2011-08-10

Analytical Comparison of Multimicrophone Probes in Measuring Acoustic Intensity

Curtis P. Wiederhold
Brigham Young University - Provo

Analytical Comparison of Multimicrophone Probes in Measuring Acoustic Intensity

Curtis P. Wiederhold

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Jonathan D. Blotter, Chair
Kent L. Gee
Scott D. Sommerfeldt

Department of Mechanical Engineering

Brigham Young University

December 2011

ABSTRACT

Analytical Comparison of Multimicrophone Probes
in Measuring Acoustic Intensity

Curtis P. Wiederhold
Department of Mechanical Engineering
Master of Science

In the late 1970s, a method was developed to estimate acoustic intensity in one dimension by taking the cross-spectral density of two closely-spaced microphone signals. Since then, multimicrophone probes have been developed to measure three-dimensional intensity as well as energy density. Their usefulness has led to the design of various types of multimicrophone probes, the most common being the four-microphone orthogonal, the four-microphone regular tetrahedron, and the six-microphone designs. These designs generally either consist of microphones suspended in space near each other or mounted on the surface of a sphere. This work analytically compares the relative merits of each probe design in measuring acoustic intensity and investigates the various finite-sum and finite-difference processing methods used with each. The analysis is limited to probes consisting of perfect point sensors in plane wave fields. The comparison is given in terms of average and maximum errors for intensity magnitude and direction as a function of angle of incidence as well as the spread between maximum and minimum errors for intensity magnitude. After existent probe geometries are reviewed, optimization techniques are introduced to predict what the optimal probe geometry would be for any given scenario. The probe is optimized to give the lowest intensity error averaged over angle of incidence of plane waves. This is done for full-space and half-space scenarios.

Keywords: Curtis Wiederhold, acoustics, multimicrophone probe, vector probe, intensity probe, energy density probe, six-microphone probe, four-microphone probe, p-p technique, optimization, genetic algorithm, spherical scattering.

ACKNOWLEDGMENTS

Having only my name listed as author of this work is misleading at best. My coworkers, graduate committee, and family members have made this research possible and made my time doing it very enjoyable and satisfying. Divine help has provided the years with optimism and perspective.

Thanks to my graduate advisor, Jonathan Blotter, for being ever-friendly and having faith enough in me to always have high expectations. Thanks to Kent Gee for countless ideas and discussions and for being an incredible model of hard work. Thanks to Scott Sommerfeldt for his valuable insights and for being remarkably accessible despite many responsibilities. Thanks to Jarom Giraud, my excellent tag-teammate, for making work enjoyable and sharing the excitement of research with me. Thanks to Derek Thomas for paving the way before me and assisting me as I began. Thanks to Blue Ridge Research and Consulting—their high-quality professional work has made them excellent to work with on this project. I am grateful for the funding from a NASA Stennis Space Center SBIR through Blue Ridge Research and Consulting and from the Rocky Mountain Space Grant Consortium. A deep appreciation goes to my parents for instilling and cultivating in me a love of learning, formal education, and work. Finally, thanks to my wife Annie for continually encouraging me to work long and hard; her support has been invaluable. My goal has always been to make her proud.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

This introduction serves as an overview of multimicrophone probes and their use and describes how the research presented here contributes to this area of acoustics. It summarizes the objectives of the research and gives an outline of the rest of the thesis. A description of the computer code developed is also included.

1.1 Problem Statement

The most widely used measuring device in acoustics is the microphone, used to measure the pressure fluctuations of acoustic waves. Multimicrophone probes consist of more than one microphone and thus measure this pressure at more than one point in space, allowing particle velocity, acoustic intensity, and acoustic energy density to be estimated—quantities useful in characterizing sound fields. However, there is no standard design or implementation for such probes. Probe design factors such as how many microphones are used, how the microphones are arranged relative to each other, and how the microphones are mounted all influence the accuracy of the measurements of these probes. Probe accuracy also depends on implementation factors such as what acoustic quantity is being measured, what finite-sum and finite-difference estimations are used, and how the probe is oriented in the acoustic field.

This work focuses on analyzing the design and implementation of multimicrophone probes by comparing the errors of various existent probes and methods. It also describes optimization techniques that can be used to predict situation-specific lowest-error designs.

1.2 Background

Since the publications of Fahy[1] and Pavic[2] in 1977, multimicrophone probes have been widely studied and used to analyze acoustic fields. For example, probes developed at BYU are currently being used by the Air Force to characterize the acoustic fields of jets which could help in the design of quieter jet engines. This would help reduce the more than $1 billion per year the U.S. Department of Defense spends on the treatment and compensation for noise-induced hearing loss in military personnel.[3]

In another example, researchers at BYU have used such a probe to produce plots like those in Figure 1-1, which shows intensity vectors as measured at an Orion GMD solid rocket motor static firing at ATK Space Systems in Promontory, Utah. This figure shows intensity vectors for 50 Hz, 150 Hz, and 500 Hz with the rocket nozzle at the origin and the shear layer shown as the black line. The figure shows that the 50 Hz energy originates at a location downstream of the 500 Hz energy source. Such plots help in the understanding and modeling of rocket noise fields. Other acoustic fields can likewise be measured and analyzed with multimicrophone probes; they have proven to be versatile and useful acoustic measurement tools.

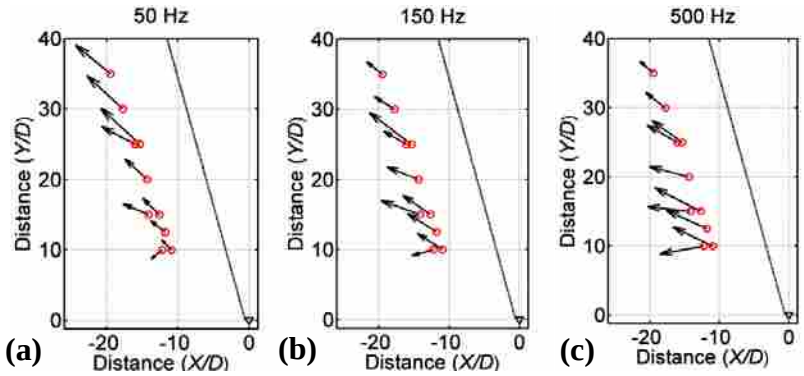

Figure 1-1: Vector Plots of Intensity at 50 Hz (a), 150 Hz (b), and 500 Hz (c) in the Sound Field of an Orion GMD Solid Rocket Motor Static Firing.

This work will principally examine three common designs of multimicrophone probes, examples of which are shown in Figure 1-2: the six-microphone design,[4,5] the four-microphone regular tetrahedron design,[6-8] and the four-microphone orthogonal design.[9,10] This figure also serves to illustrate the three typical ways microphones are mounted in a multimicrophone probe. The first method is to merely have the microphones suspended near each other, typically mounted in some sort of framework like the probe shown in Figure 1-2(b). In this mounting method, the framework is made to be thin so that scattering can be minimized. Scattering is typically neglected in the analysis of such probes as it is dependent on the framework geometry and because the framework is typically asymmetric. Pascal and Li[11] published a highly relevant work in 2008, analytically comparing various probe geometries with no scattering. In their work they examined the errors of such probes in calculating intensity and energy density.

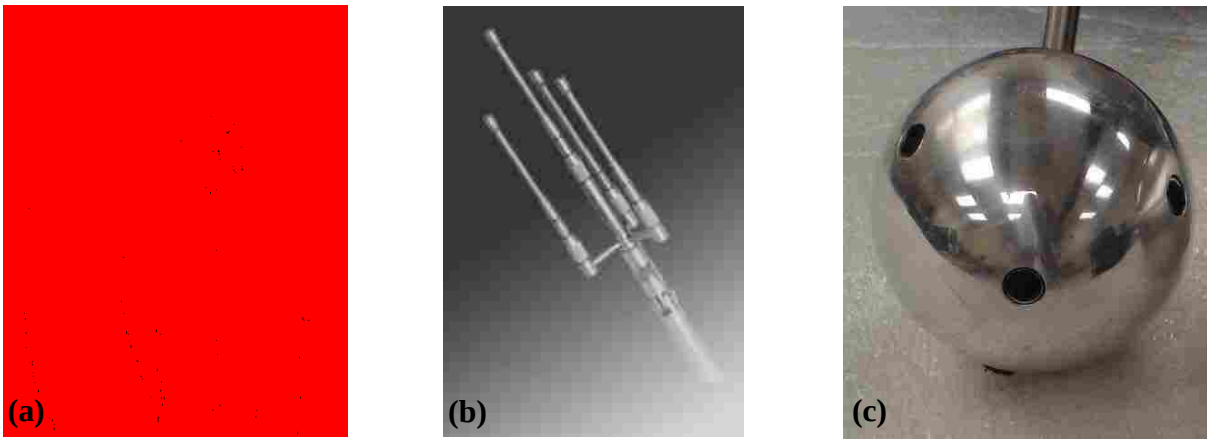

Figure 1-2: The Six-Microphone G.R.A.S. Vector Intensity Probe Type 50VI (a), the Four-Microphone Regular Tetrahedron Ono Sokki MI-6420 Tetra-Phone (b), and the Four-Microphone Spherical Orthogonal Probe Made by Locey.[12]

The second mounting method is to use spacers between the faces of the various microphones like the probe shown in Figure 1-2(a). Spacers have been shown to be beneficial for

one-dimensional, two-microphone probes[13] by stabilizing the "acoustic distance" between the microphones when they are in the face-to-face arrangement. However, Moschioni *et al.*[14] pointed out that these beneficial results are probe specific and "prevent the probe from being effective in low-frequency intensity measurements." Because of these drawbacks and because the scattering off such spacers is difficult to predict, they are not studied in this work.

The third mounting method is to have the microphones embedded on the surface of a rigid sphere as shown in Figure 1-2(c). This is advantageous because the scattering off a hard sphere is predictable.[15,16] Elko[17] and Parkins *et al.*[5] examined this idea for one-dimensional, two-microphone probes, showing that mounting microphones in a sphere generally results in less error for intensity and energy density calculations in plane wave fields. In 2004, Locey[12] numerically and experimentally compared various spherical probe types.

The objective of this work is to synthesize and further the results of previous studies, providing a thorough comparison of multimicrophone probes in plane wave fields. It will further the work of Pascal and Li by including results from spherical probes and examining new processing methods. Also, in addition to looking at maximum errors, an average error metric will be developed and used. It will build on the work of Elko and Parkins *et al.* by considering three-dimensional spherical probes. It will complement Locey's work by including non-spherical probes (assuming no scattering) in the comparison and by considering new processing methods. The error curves predicted by such analyses as the one performed in this work make probe calibration a possibility (such as done by Suzuki *et al.*[18] and Moschioni *et al.*[14]). This idea is examined and discussed as it applies to the comparison in this work.

While the existent designs of multimicrophone probes are generally intuitive and symmetric, there is no reason to believe that they represent the lowest-error designs possible.

Therefore, it is useful to explore optimization techniques that can predict lowest-error designs. Optimization using a genetic algorithm is developed in this work. As any "optimal" probe design depends on a number of factors such as which acoustic quantity is being measured and what frequency range is of interest, the objective is to create an algorithm that will predict an optimal design based on a set of input parameters. Preliminary results are given and discussed with recommendations for future work.

1.3 Objectives and Scope of Research

- Compare multimicrophone probe designs.
    - Geometric arrangement: compare six-microphone, four-microphone regular tetrahedron, and four-microphone orthogonal arrangements.
    - Mounting method: compare results from microphones mounted in an external framework to those mounted in a rigid sphere.
    - Use optimization methods to predict situation-specific optimal probe designs.
- Compare multimicrophone probe implementations.
    - Intensity calculation: compare processing methods to calculate intensity.
    - Probe orientation: investigate max and average errors dependent on probe orientation as well as error spread.
    - Calibration: investigate feasibility of using calibration curves.

The scope of this research is limited to analyzing probes in plane wave fields, leaving analysis in reactive fields as future work. However, Pascal and Li[11] note that, based on the results of studies of one-dimensional intensity probes, studying multimicrophone probes in plane wave

fields “could sufficiently provide a realistic estimate of the errors and the high-frequency limit of the use of the probes.” Another important limitation is that the microphones are assumed to be perfect point sensors. However, some preliminary work with phase errors is introduced in the optimization work in Chapter 5. Finally, this work is limited in that it does not examine the “wave vector method,” a promising processing method suggested by Thomas[19] in 2008.

1.4 Computer Code Used

Computer code was developed for this work in MATLAB to compare the various probe designs and implementations. The two main MATLAB scripts used are included as Appendix B. The first, “Multimicrophone_Probe_Analysis.m,” calculates the errors associated with the six-microphone, the four-microphone regular tetrahedron, and the four-microphone orthogonal probes. An explanation of the simulation performed by this script is given in Chapters 2 and 3. This script uses the included function files “Angle_Calc.m,” “Bias_Calc.m,” “Intensity_Calc.m,” and “Taylor_Velocity.m.” The scripts “Chapter_2_Figures.m” and “Chapter_3_Figures.m” produce the plots found in Chapters 2 and 3 using the data output by “Multimicrophone_Probe_Analysis.m.” The script “Genetic_Algorithm.m,” predicts an optimal probe design based on a chosen fitness function by using a genetic algorithm. Discussion of this script and its output is given in Chapter 4. It uses the included function files “Intensity_Error.m,” and “Design_Checker.m,” and these function files in turn use the function files “Angle_Calc.m,” “Bias_Calc.m,” “Intensity_Calc.m,” and “Taylor_Velocity.m” (the same function files used by “Multimicrophone_Probe_Analysis.m”). After an optimization is run, the script “Chapter_4_Figures.m” can then be used to produce the plots found in Chapter 4.

1.5 Thesis Outline

Chapter 2 examines the four-microphone orthogonal probe, looking at the errors associated with the various processing methods that can be used for calculating intensity. New processing methods are also suggested and analyzed. Finally, the design consideration of whether to mount the microphones in a sphere or not is studied and cross-spectral formulas for all processing methods are given. Chapter 3 performs an analysis similar to that in Chapter 2, but does so for the four-microphone regular tetrahedron and six-microphone probes. These results are then compared with the results from Chapter 2 and it is shown which probes and processing methods have the lowest error in estimating intensity magnitude and direction. Chapters 2 and 3 represent work submitted for journal publication.

Chapter 4 then transitions away from examining existent probe designs and explores how an optimal probe design might be predicted. Optimization of a multimicrophone probe based on a chosen fitness function is explored using a genetic algorithm. Conclusions and recommendations for future work are then given in Chapter 5.

2 ORTHOGONAL PROBE COMPARISON

This chapter presents a paper submitted to the Journal of the Acoustical Society of America for archival publication. The formatting was modified to make it consistent with this thesis.

2.1 Contributing Authors

Curtis P. Wiederhold
Department of Mechanical Engineering, Brigham Young University, Provo, Utah 84602

Kent L. Gee
Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602

Jonathan D. Blotter
Department of Mechanical Engineering, Brigham Young University, Provo, Utah 84602

Scott D. Sommerfeldt
Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602

2.2 Abstract

Of the various designs of three-dimensional multimicrophone probes, the four-microphone orthogonal design (consisting of one microphone at an origin position with the other three microphones equally spaced along the three coordinate axes) is commonly used due to its simplicity of design and use. Several distinct processing methods to estimate active acoustic intensity with the orthogonal probe have been suggested; however, the relative merits of each method have not been thoroughly studied. This comparative study is an analytical investigation

of the errors associated with each method. Orthogonal probes consisting of matched point sensor microphones both embedded and not embedded on the surface of a rigid sphere are considered. Results are given for plane wave fields over all angles of incidence. It is shown that the lowest error in measuring intensity magnitude results from having the microphones in a sphere and using just one microphone for the pressure estimate. For measuring intensity direction the lowest error results from having the microphones in a sphere, using a Taylor approximation to estimate the particle velocity at one point, and using the average pressure of three of the microphones.

2.3 Introduction

Active intensity is an acoustic energy quantity useful in characterizing sound fields with applications including sound source localization. To calculate this quantity at a given point in space, the pressure and particle velocity at that point must be simultaneously known. The pressure can be estimated using a finite sum between two or more microphones and the velocity with a finite-difference calculation, which is known as the p-p technique.[1,2,20] With two microphones the intensity can be estimated in one dimension while with four or more microphones it is possible to get a complete three-dimensional estimation.[4]

Such probes, capable also of estimating energy density, are referred to as multimicrophone probes (also known as vector probes, intensity probes, or energy density sensors) and have been in wide use since the 1980s. Multimicrophone probes now come in a variety of designs, the three most common being the four-microphone orthogonal design,[9,10] the four-microphone regular tetrahedron design,[6,7] and the six-microphone design.[4,5] In this work the four-microphone orthogonal design will be investigated by comparing the various implementations of this probe design in estimating intensity.

As shown in Figure 2-1(a), the orthogonal design consists of one microphone at an "origin" position (labeled microphone 1) with the other three microphones (labeled 2, 3, and 4) equidistant from the first microphone along the three coordinate axes. It has previously been referred to as the "cubic" probe,[11,21] but "orthogonal" is preferred here because the probe does not consist of microphones at all the vertices of a cube.

Many multimicrophone probes consist of microphones suspended in space near each other, with acoustic scattering avoided by making the microphone-holding fixture as small as possible.[8,22] Such probes are typically analyzed assuming that the microphones are point sensors floating in space, thus neglecting any effects from scattering.[10,11] It has alternatively been suggested that the microphones be embedded on the surface of a hard sphere, a situation where the scattering is predictable.[17] It was shown that, for a two-microphone case, having the sensors embedded in a sphere results in slightly less error in measuring acoustic quantities in both active and reactive fields when the microphone responses are matched. However, when phase and amplitude mismatch are introduced, the effect is lost and the errors are dominated by the mismatches. But in both cases (with matched or mismatched microphones), the spherical scattering effectively allows for the microphone separation distance in spherical designs to be more compact by a factor of 2/3.[5,17] Due to the benefits resulting from the spherical scattering, a spherical orthogonal probe design has been suggested and a prototype made, resulting in a probe as shown in Figure 2-1(b).[12] Though work has been done for the two-microphone case,[5,17] a thorough study has not been done to investigate whether mounting the microphones on a sphere (spherical scattering) or having them "freely suspended" (assuming no scattering) results in more accurate measurements for orthogonal probes. In this paper results from such a probe design will be compared to those from a freely suspended probe design.

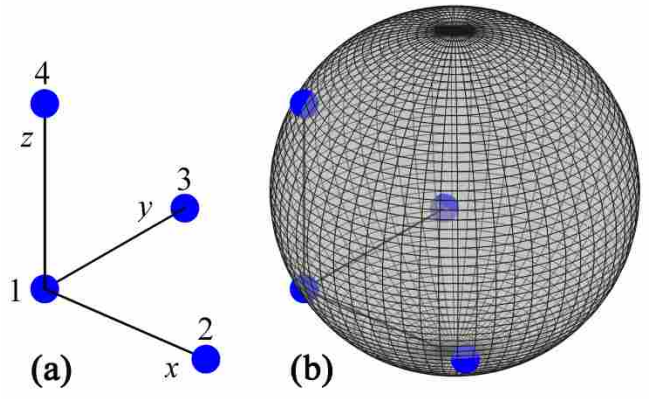

Figure 2-1: Four-Microphone Orthogonal Probe Designs with Microphones Freely Suspended (a) and Mounted on the Surface of a Sphere (b), Dots Representing Microphone Locations.

The various methods used to estimate the intensity vector using an orthogonal multimicrophone probe along with the consideration of whether or not to mount the microphones on a sphere lead to 18 intensity calculation combinations considered in this paper. Section 2.4 outlines the various processing methods that can be used with the orthogonal probe and Section 2.5 presents the cross-spectral formulas that correspond to each processing method. The analytical model that is used to compare the methods is described in Section 2.6, results are presented and discussed in Section 2.7, and concluding remarks are given in Section 2.8.

2.4 Intensity Estimation Methods

Active acoustic intensity is a measure of the net flow of energy carried by an acoustic wave through a unit area. Using complex notation, the time-averaged intensity is given by:

$$\mathbf{I} = \frac{1}{2}\mathrm{Re}\{p\mathbf{v}^*\}, \tag{2-1}$$

where p is the complex pressure, $\mathbf{v}$ is the complex particle velocity vector, the complex conjugate is denoted by "*," and the real part is denoted by "Re." The various orthogonal probe processing methods arise from different ways to estimate the particle velocity and the pressure in this expression.

2.4.1 Particle Velocity Estimation

There are two ways the particle velocity has been estimated with orthogonal probes. Commonly, a finite-difference approximation between each of the two-microphone pairs along the *x*-, *y*-, and *z*-axes is used, thereby estimating *x*-, *y*-, and *z*-components of the velocity.[10,11,23,24] This is done by starting with the time-harmonic linear Euler's equation (using the $e^{j\omega t}$ time convention):

$$\mathbf{v} = \frac{j\nabla p}{\rho\omega}, \tag{2-2}$$

where ρ is the air density, ω the angular frequency, and j the imaginary unit. The gradient of the pressure is estimated by a finite-difference estimate between microphone pairs lying along each coordinate axis, resulting in the particle velocity expression:

$$\mathbf{v} = \begin{cases} v_x \approx \dfrac{j(p_2 - p_1)}{2h\rho\omega} \\ v_y \approx \dfrac{j(p_3 - p_1)}{2h\rho\omega} \\ v_z \approx \dfrac{j(p_4 - p_1)}{2h\rho\omega} \end{cases}, \tag{2-3}$$

with $2h$ being the distance from the origin microphone to any of the three other microphones and p_1, p_2, p_3, and p_4 being the complex pressures at each microphone location. If the microphones are mounted on the surface of a sphere, Elko[17] found that the separation distance must be multiplied by a low-frequency correction factor of 3/2. In this case, $3h$ would be used in Eq.

(2-3) instead of 2*h*. This correction factor is needed for an accurate low-frequency estimate of the pressure gradient using the finite-difference method. It represents an increased effective distance between the microphones due to the scattering introduced by the sphere and is the reason the spherical probe can effectively be made more compact by a factor of 2/3.

Using Eq. (2-3) to estimate the particle velocity results in each component of the velocity being estimated at a different point in space and so is referred to in this work as the "three points" velocity estimate. Rasmussen[4] has suggested that this is an important weakness of the orthogonal design. To offset this, Cazzolato and Ghan[21] have proposed using a weighted pressure estimate to put the point where the pressure is estimated closer to the three points where the velocity is estimated—an idea described more fully in the next section on pressure estimation. Alternatively, Locey[12] suggested using a first-order Taylor approximation of the velocity utilizing the finite-difference results of all 6 two-microphone pairs to put the three velocity estimates at one point. With the origin microphone at (0,0,0) this method gives an estimate of all three components of the velocity at the point (*h*,*h*,*h*) in the following manner:

The *x*-component of the particle velocity is approximated as

$$v_x(h,h,h) \approx v_x(h,0,0) + h\left.\frac{\partial v_x}{\partial y}\right|_{(h,0,0)} + h\left.\frac{\partial v_x}{\partial z}\right|_{(h,0,0)}. \tag{2-4}$$

The partial derivatives with respect to the *y*- and *z*-directions could themselves be approximated by first-order Taylor approximations:

$$\begin{aligned}\left.\frac{\partial v_x}{\partial y}\right|_{(h,0,0)} &\approx \frac{v_x(h,h,0) - v_x(h,0,0)}{h},\\ \left.\frac{\partial v_x}{\partial z}\right|_{(h,0,0)} &\approx \frac{v_x(h,0,h) - v_x(h,0,0)}{h},\end{aligned} \tag{2-5}$$

leading to the final expression

$$v_x(h,h,h) \approx v_x(h,h,0) + v_x(h,0,h) - v_x(h,0,0). \quad (2\text{-}6)$$

These three terms correspond to velocities estimated at the three points shown in Figure 2-2.

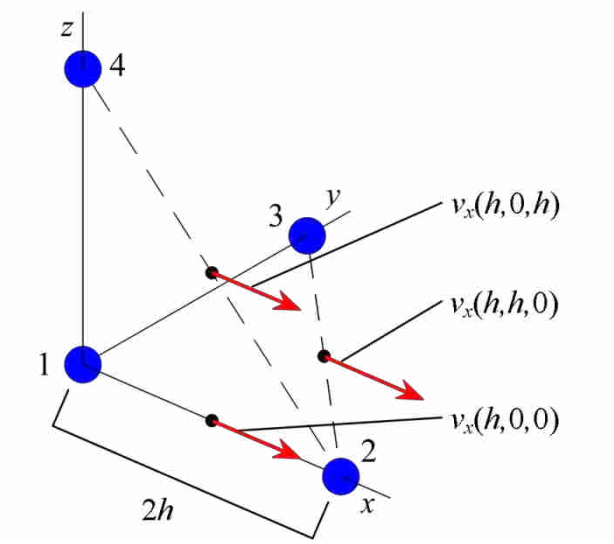

Figure 2-2: Velocity Vectors Needed to Calculate Taylor Approximation of *X*-Direction Velocity at Point (*h*,*h*,*h*).

The *x*-direction velocity at (*h*,0,0), the point midway between microphones 1 and 2 is calculated normally; however, the *x*-velocity estimates between microphones 2 and 3 and between microphones 2 and 4 require extra calculations because finite differencing gives velocities with both *x*- and *y*-components (they would lie along the direction of the dashed lines in Figure 2-2). The problem is overcome by using the velocity information from other pairs of microphones to make an estimate of the direction of the velocity. The *x*-direction velocity at point (*h*,*h*,0) is used as an example. A diagonal velocity (v_{diag}) between the microphones 2 and 3 can be calculated at this point using a finite-difference approximation. Then, the angle of the three-dimensional particle velocity of the incoming sound wave in the *xy*-plane can be

approximated using the *x*-direction velocity (obtained from microphones 1 and 2) and the *y*-direction velocity estimate (obtained from microphones 1 and 3) using the equation

$$\theta_{xy} \approx \tan^{-1}\frac{|v_y(0,h,0)|}{|v_x(h,0,0)|}, \tag{2-7}$$

where vertical bars give the absolute value of the complex velocity. However, the angle θ_{xy} must be multiplied by a factor of -1 if the difference in phase between $v_x(h,0,0)$ and $v_y(0,h,0)$ is between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians, that is if:

$$\frac{\pi}{2} \leq \left|\arg\left(\frac{v_x(h,0,0)}{v_y(0,h,0)}\right)\right| \leq \frac{3\pi}{2}\ , \tag{2-8}$$

where "arg" denotes the angle of the complex number and the vertical bars give the absolute value of the real number. This step is necessary because Eq. (2-7) uses only the magnitudes of the complex velocities—not enough information to uniquely identify the direction of the acoustic wave.

With this angle and the diagonal velocity estimated, the *x*-direction velocity at (*h*,*h*,0) can then be estimated as follows. The velocity v_{diag} is considered to be a component of the *xy*-plane component of the actual particle velocity. The estimate of this actual velocity in the *xy*-plane is called v_{xy} and its direction is estimated as θ_{xy}. The *x*-direction velocity at (*h*,*h*,0) is then the *x*-component of the velocity v_{xy} as shown in Figure 2-3 and is thus calculated by

$$v_x(h,h,0) \approx v_{diag}(h,h,0)\frac{\cos(\theta_{xy})}{\cos\left(\frac{3\pi}{4}-\theta_{xy}\right)}. \tag{2-9}$$

The final term needed is the *x*-direction velocity at point (*h*,0,*h*) and is obtained in a manner similar to the velocity at point (*h*,*h*,0) but in the *xz*-plane instead. An approximation of the *x*-direction velocity at (*h*,*h*,*h*) can then be calculated using Eq. (2-6). A similar formulation leads to

$$v_y(h,h,h) \approx v_y(h,h,0) + v_y(0,h,h) - v_y(0,h,0), \tag{2-10}$$

for the *y* velocity and

$$v_z(h,h,h) \approx v_z(h,0,h) + v_z(0,h,h) - v_z(0,0,h), \tag{2-11}$$

for the *z* velocity. A three-dimensional velocity vector at point (*h*,*h*,*h*) is then fully estimated.

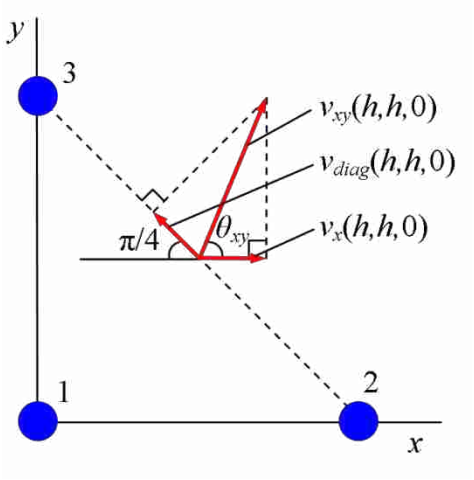

Figure 2-3: Diagram of Method Used to Estimate *X*-Direction Velocity from Diagonal Velocity.

There are thus two ways to estimate particle velocity: first, the typical method of estimating the velocity at three points in space and, second, using a Taylor approximation to put all components of the velocity estimate at the point (*h*,*h*,*h*).

2.4.2 Pressure Estimation

To calculate active intensity, pressure must also be estimated. There are five methods to estimate the pressure that will be considered in this work. First, the pressure can be estimated as the average pressure of the four microphones:[25]

$$p \approx \frac{p_1 + p_2 + p_3 + p_4}{4}. \tag{2-12}$$

Second, the pressure can be estimated as the pressure from the origin microphone:[11]

$$p \approx p_1. \tag{2-13}$$

Third, the pressure can be estimated using a weighted average of the pressures favoring the origin microphone:

$$p \approx \frac{1}{3}\left(\frac{p_1 + p_2}{2} + \frac{p_1 + p_3}{2} + \frac{p_1 + p_4}{2}\right). \tag{2-14}$$

The weighted pressure estimate has been suggested as an alternative to the normal average as it puts the pressure estimate at the point $\left(\frac{h}{3}, \frac{h}{3}, \frac{h}{3}\right)$, which is closer than the point $\left(\frac{h}{2}, \frac{h}{2}, \frac{h}{2}\right)$ of the normal average to the three points at (h,0,0), (0,h,0), and (0,0,h) where the velocity is estimated when the three points velocity estimate is used.[21] However, when the Taylor expansion is used the velocity estimate is at (h,h,h), and so the closest pressure estimate would be given by the average of the three non-origin microphones:

$$p \approx \frac{p_2 + p_3 + p_4}{3}, \tag{2-15}$$

giving a pressure estimate at $\left(\frac{2h}{3}, \frac{2h}{3}, \frac{2h}{3}\right)$. This pressure estimate is unique to this work.

Finally, a different pressure estimate can be used for each of the three orthogonal directions. When this pressure estimate is used with the three points velocity estimate, the orthogonal probe is essentially being used as 3 one-dimensional probes, such as the probe used by Vandenhout *et al*.[23] The intensity in each orthogonal direction is calculated using the average of only the pressures measured by the two microphones along that direction. For example, the x-direction intensity would be calculated by:

$$\begin{aligned} I_x &= \frac{1}{2}\mathrm{Re}\{p_x v_x^*\} \\ &= \frac{1}{2}\mathrm{Re}\left\{\left(\frac{p_1 + p_2}{2}\right) v_x^*\right\}, \end{aligned} \tag{2-16}$$

and the total intensity is then the Euclidean norm of the three orthogonal intensity estimates. When this pressure estimate is used in conjunction with the three points velocity estimate, the pressure and velocity estimates both lie at the same point for each orthogonal intensity estimate.

2.4.3 Summary of Estimation Methods

The two ways of estimating the particle velocity combined with the five ways of estimating the pressure lead to ten ways of estimating intensity with an orthogonal probe. These processing methods can either be used with microphones mounted in a sphere or not, resulting in 20 total probe types that will be analyzed. This work will investigate the errors of each of these methods in measuring the magnitude and the angle of active intensity for orthogonal probes.

Table 2-1 summarizes all intensity processing types and identifies each by an abbreviation. If the probe type has the microphones mounted on a sphere, the abbreviation “S” is used. In estimating the pressure, “1” signifies that the pressure from one microphone is used, “3” for a three-microphone average, “4” for a four-microphone average, “4W” for a weighted four-microphone average, and “3 1D” signifies that three separate pressure estimates are used. The abbreviation “T” is used if the Taylor approximation is used for the velocity estimate instead of the typical three points estimate. A forward slash is used to separate the design consideration of mounting the microphones on a sphere or not from the processing considerations of pressure and velocity estimates.

Table 2-1: Summary and Abbreviations of Considered Methods Used to Estimate Intensity with the Orthogonal Probe. (Continued on Next Page)

Abbreviation	Scattering	Pressure Estimate	Velocity Estimate
1	None	$p = p_1$	Three points
3	None	$p = \frac{p_2 + p_3 + p_4}{3}$	Three points
4	None	$p = \frac{p_1 + p_2 + p_3 + p_4}{4}$	Three points
4W	None	$p = \frac{1}{3}\left(\frac{p_1 + p_2}{2} + \frac{p_1 + p_3}{2} + \frac{p_1 + p_4}{2}\right)$	Three points
3 1D	None	$p_x = \frac{p_1 + p_2}{2}, p_y = \frac{p_3 + p_4}{2}, p_z = \frac{p_5 + p_6}{2}$	Three points
1.T	None	$p = p_1$	Taylor approximation
3.T	None	$p = \frac{p_2 + p_3 + p_4}{3}$	Taylor approximation
4.T	None	$p = \frac{p_1 + p_2 + p_3 + p_4}{4}$	Taylor approximation
4W.T	None	$p = \frac{1}{3}\left(\frac{p_1 + p_2}{2} + \frac{p_1 + p_3}{2} + \frac{p_1 + p_4}{2}\right)$	Taylor approximation
3 1D.T	None	$p_x = \frac{p_1 + p_2}{2}, p_y = \frac{p_3 + p_4}{2}, p_z = \frac{p_5 + p_6}{2}$	Taylor approximation
S/1	Spherical	$p = p_1$	Three points
S/3	Spherical	$p = \frac{p_2 + p_3 + p_4}{3}$	Three points
S/4	Spherical	$p = \frac{p_1 + p_2 + p_3 + p_4}{4}$	Three points
S/4W	Spherical	$p = \frac{1}{3}\left(\frac{p_1 + p_2}{2} + \frac{p_1 + p_3}{2} + \frac{p_1 + p_4}{2}\right)$	Three points
S/3 1D	Spherical	$p_x = \frac{p_1 + p_2}{2}, p_y = \frac{p_3 + p_4}{2}, p_z = \frac{p_5 + p_6}{2}$	Three points
S/1.T	Spherical	$p = p_1$	Taylor approximation

Table 2-1 Continued

Abbreviation	Scattering	Pressure Estimate	Velocity Estimate
S/3.T	Spherical	$p = \frac{p_2 + p_3 + p_4}{3}$	Taylor approximation
S/4.T	Spherical	$p = \frac{p_1 + p_2 + p_3 + p_4}{4}$	Taylor approximation
S/4W.T	Spherical	$p = \frac{1}{3}\left(\frac{p_1 + p_2}{2} + \frac{p_1 + p_3}{2} + \frac{p_1 + p_4}{2}\right)$	Taylor approximation
S/3 1D.T	Spherical	$p_x = \frac{p_1 + p_2}{2}, p_y = \frac{p_3 + p_4}{2}, p_z = \frac{p_5 + p_6}{2}$	Taylor approximation

2.5 Cross-Spectral Formulations

Cross-spectral formulas allow the intensity to be calculated in the frequency domain using the cross-spectra from the different time signals measured by the microphones. The one-sided cross-spectral density for a zero-mean process is defined as

$$\begin{aligned} G_{mn}(\omega) &= C_{mn}(\omega) + jQ_{mn}(\omega) \\ &= \lim_{T\to\infty} \frac{2}{T} \mathrm{E}[P_m^*(\omega, T) P_n(\omega, T)] , \end{aligned} \tag{2-17}$$

for $\omega \geq 0$, where P_m and P_n are the Fourier transforms of the pressures from the m^{th} and n^{th} microphones, respectively, over time T. The expectation operator is denoted by E[ ]. The real part of the cross-spectral density G is defined as the cospectral density, C, and the imaginary part as the quadspectral density, Q. Some of these equations have been presented by Pascal and Li,[11] but all are given here for completeness. The formulas given are for probes with microphones freely suspended in space: the variable h must be multiplied by 3/2 if instead the microphones are mounted in a sphere. Though not explicitly written in the following equations for brevity, all cross-spectral densities (and hence intensities) are functions of frequency. The equation for the probe type abbreviated "1" is

$$\mathbf{I}_1 = \begin{cases} I_x \approx \dfrac{Q_{21}}{2h\rho\omega} \\ I_y \approx \dfrac{Q_{31}}{2h\rho\omega} \\ I_z \approx \dfrac{Q_{41}}{2h\rho\omega} \end{cases} . \tag{2-18}$$

Probe type “4”:

$$\mathbf{I}_4 = \begin{cases} I_x \approx \dfrac{1}{8h\rho\omega}(2Q_{21} + Q_{31} + Q_{41} - Q_{32} - Q_{42}) \\ I_y \approx \dfrac{1}{8h\rho\omega}(2Q_{31} + Q_{21} + Q_{41} + Q_{32} - Q_{43}) \\ I_z \approx \dfrac{1}{8h\rho\omega}(2Q_{41} + Q_{21} + Q_{31} + Q_{42} + Q_{43}) \end{cases} . \tag{2-19}$$

Probe type “4W”:

$$\mathbf{I}_{4\mathrm{W}} = \begin{cases} I_x \approx \dfrac{1}{12h\rho\omega}(4Q_{21} + Q_{31} + Q_{41} - Q_{32} - Q_{42}) \\ I_y \approx \dfrac{1}{12h\rho\omega}(4Q_{31} + Q_{21} + Q_{41} + Q_{32} - Q_{43}) \\ I_z \approx \dfrac{1}{12h\rho\omega}(4Q_{41} + Q_{21} + Q_{31} + Q_{42} + Q_{43}) \end{cases} . \tag{2-20}$$

Probe type “1.T”:

$$\mathbf{I}_{1.\mathrm{T}} = \begin{cases} I_x \approx A_1\left[A_2(Q_{31} - Q_{21}) + A_3(Q_{41} - Q_{21}) - \sqrt{2}Q_{21}\right] \\ I_y \approx A_1\left[A_4(Q_{31} - Q_{21}) + A_5(Q_{41} - Q_{31}) - \sqrt{2}Q_{31}\right] \\ I_z \approx A_1\left[A_6(Q_{41} - Q_{21}) + A_7(Q_{41} - Q_{31}) - \sqrt{2}Q_{41}\right] \end{cases} , \tag{2-21}$$

with:

$$A_1 = \frac{1}{2\sqrt{2}h\rho\omega}$$

$$A_2 = \frac{\cos\left(\theta_{xy}\right)}{\cos\left(\frac{3\pi}{4} - \theta_{xy}\right)}$$

$$A_3 = \frac{\cos(\theta_{xz})}{\cos\left(\frac{3\pi}{4} - \theta_{xz}\right)}$$

$$A_4 = \frac{\sin\left(\theta_{xy}\right)}{\cos\left(\frac{3\pi}{4} - \theta_{xy}\right)}$$

$$A_5 = \frac{\cos\left(\theta_{yz}\right)}{\cos\left(\frac{3\pi}{4} - \theta_{yz}\right)}$$

$$A_6 = \frac{\sin(\theta_{xz})}{\cos\left(\frac{3\pi}{4} - \theta_{xz}\right)}$$

$$A_7 = \frac{\sin\left(\theta_{yz}\right)}{\cos\left(\frac{3\pi}{4} - \theta_{yz}\right)} \quad , \tag{2-22}$$

where:

$$\theta_{xy} = \tan^{-1}\sqrt{\frac{G_{33} + G_{11} - 2C_{13}}{G_{22} + G_{11} - 2C_{12}}}$$

$$\theta_{xz} = \tan^{-1}\sqrt{\frac{G_{44} + G_{11} - 2C_{14}}{G_{22} + G_{11} - 2C_{12}}} \quad . \tag{2-23}$$

$$\theta_{yz} = \tan^{-1}\sqrt{\frac{G_{44} + G_{11} - 2C_{14}}{G_{33} + G_{11} - 2C_{13}}}$$

However, these angles might need to be multiplied by -1 depending on the phase relations between the different velocity components as shown by Eq. (2-8) for θ_{xy}. But, for this step to be performed in terms of cross-spectra, the phase between intensity components is compared instead of velocity components. This assumes that the pressure estimates of the

intensities are equivalent, which they are not; however, the assumption serves as a satisfactory approximation. Mathematically, this step is thus given by:

$$\begin{aligned}
\theta_{xy} &= -\theta_{xy} \quad \text{if} \quad \frac{\pi}{2} \leq \left|\arg\left(\frac{Q_{21}}{Q_{31}}\right)\right| \leq \frac{3\pi}{2} \\
\theta_{xz} &= -\theta_{xz} \quad \text{if} \quad \frac{\pi}{2} \leq \left|\arg\left(\frac{Q_{21}}{Q_{41}}\right)\right| \leq \frac{3\pi}{2} \quad , \\
\theta_{yz} &= -\theta_{yz} \quad \text{if} \quad \frac{\pi}{2} \leq \left|\arg\left(\frac{Q_{31}}{Q_{41}}\right)\right| \leq \frac{3\pi}{2}
\end{aligned} \tag{2-24}$$

Probe type “4.T”:

$$\mathbf{I}_{4.\mathrm{T}} = \begin{cases} I_x \approx A_8\left(A_2A_9 + A_3A_{10} - \sqrt{2}A_{11}\right) \\ I_y \approx A_8\left(A_4A_9 + A_5A_{12} - \sqrt{2}A_{13}\right) \\ I_z \approx A_8\left(A_6A_{10} + A_7A_{12} - \sqrt{2}A_{14}\right) \end{cases} , \tag{2-25}$$

with:

$$\begin{aligned}
A_8 &= \frac{1}{8\sqrt{2}h\rho\omega} \\
A_9 &= Q_{31} - Q_{21} + 2Q_{32} - Q_{43} + Q_{42} \\
A_{10} &= Q_{41} - Q_{21} + 2Q_{42} + Q_{43} + Q_{32} \\
A_{11} &= 2Q_{21} - Q_{32} + Q_{31} - Q_{42} + Q_{41} \\
A_{12} &= Q_{41} - Q_{31} + Q_{42} - Q_{32} + 2Q_{43} \\
A_{13} &= 2Q_{31} + Q_{32} + Q_{21} - Q_{43} + Q_{41} \\
A_{14} &= 2Q_{41} + Q_{42} + Q_{21} + Q_{43} + Q_{31} \quad .
\end{aligned} \tag{2-26}$$

Probe type “4W.T”:

$$\mathbf{I}_{\text{4W.T}} = \begin{cases} I_x \approx A_{15}\left(A_2A_{16} + A_3A_{17} - \sqrt{2}A_{18}\right) \\ I_y \approx A_{15}\left(A_4A_{16} + A_5A_{19} - \sqrt{2}A_{20}\right) \\ I_z \approx A_{15}\left(A_6A_{17} + A_7A_{19} - \sqrt{2}A_{21}\right) \end{cases}, \tag{2-27}$$

with:

$$A_{15} = \frac{1}{12\sqrt{2}h\rho\omega}$$

$$A_{16} = 3Q_{31} - 3Q_{21} + 2Q_{32} - Q_{43} + Q_{42}$$

$$A_{17} = 3Q_{41} - 3Q_{21} + 2Q_{42} + Q_{43} + Q_{32}$$

$$A_{18} = 4Q_{21} - Q_{32} + Q_{31} - Q_{42} + Q_{41}$$

$$A_{19} = 3Q_{41} - 3Q_{31} + Q_{42} - Q_{32} + 2Q_{43}$$

$$A_{20} = 4Q_{31} + Q_{32} + Q_{21} - Q_{43} + Q_{41}$$

$$A_{21} = 4Q_{41} + Q_{42} + Q_{21} + Q_{43} + Q_{31} \quad . \tag{2-28}$$

It was found that probe type “3 1D” gives formulas identical to that of probe type “1” and so they are not included. This result means that it is equivalent to process the orthogonal probe data as 3 one-dimensional probes or to process the data by estimating the pressure using the origin microphone. This is a non-obvious result stemming from properties of complex numbers and is proven mathematically in Appendix A.

2.6 Analytical Model

An indication of the relative merit of any one implementation can be obtained by examining the errors of an analytical probe consisting of perfect point sensors simulated in a plane wave field. In order to directly compare the orthogonal probe types, a suitable dimensionless variable is needed. The variable *ka* is used where *k* is the wavenumber and *a* the

radius of the sphere for spherical probes, or, in the case of freely suspended probes, it is the radius of an imaginary sphere that is circumscribed for the four microphone points. This radius relates to the microphone separation distance $2h$ through the relation $h = \frac{\sqrt{6}}{3}a$. However, since at low frequencies the effective separation distance of microphones mounted in a sphere is 3/2 times greater than for those not mounted in a sphere, to directly compare the two at higher frequencies, a spherical probe is compared to a freely suspended probe that is 3/2 times larger. That is, the error at any given frequency ka for a freely suspended probe is compared to the error at $\frac{3}{2}ka$ for a spherical probe. This approach was used by both Elko[17] and Parkins *et al*.[5]

The time-harmonic complex pressure at any microphone on the surface of a rigid sphere of radius a in a plane wave field consists of incident and scattered pressure. Taking the center of the sphere as the origin, the total pressure can be given (using the $e^{j\omega t}$ time convention) by the series solution:[16]

$$p_i = \frac{P_0}{j(ka)^2}\sum_{n=0}^{\infty}\frac{j^n(2n+1)P_n(\cos\theta)}{h_n^{(2)'}(ka)}, \tag{2-29}$$

where p_i is the pressure at the i^{th} microphone, P_0 the amplitude of the plane wave, $P_n(\cos\theta)$ the Legendre polynomial of order n, θ the angle between the incident plane wave and the i^{th} microphone, and $h_n^{(2)'}$ the derivative of the spherical Hankel function of the second type of order n. When scattering is not accounted for, the pressure at the i^{th} microphone on the surface of an imaginary sphere of radius a is just the incident pressure given by

$$p_i = P_0 e^{jka\cos\theta}. \tag{2-30}$$

Higher frequencies generally require more terms in the expansion to adequately calculate the scattering. Figure 2-4 shows the percent error of the pressure as a fraction of the actual pressure as more terms in the series are included for a plane wave directly incident on a

microphone location (θ = 0). Other angles of incidence yield similar results. The "actual" pressure is calculated using a very large number of terms in the series solution. Results up to $ka = \frac{\pi}{2}$ for freely suspended designs (corresponding to the sphere diameter being a half-wavelength) and $\frac{3}{2}ka = \frac{\pi}{2}$ for spherical designs are considered in this paper, and so 25 terms is shown to be more than sufficient in the scattering calculation. For the probe types with microphones not mounted in a sphere, scattering was neglected as it would vary dependent on the size and configuration of the microphones and holders.

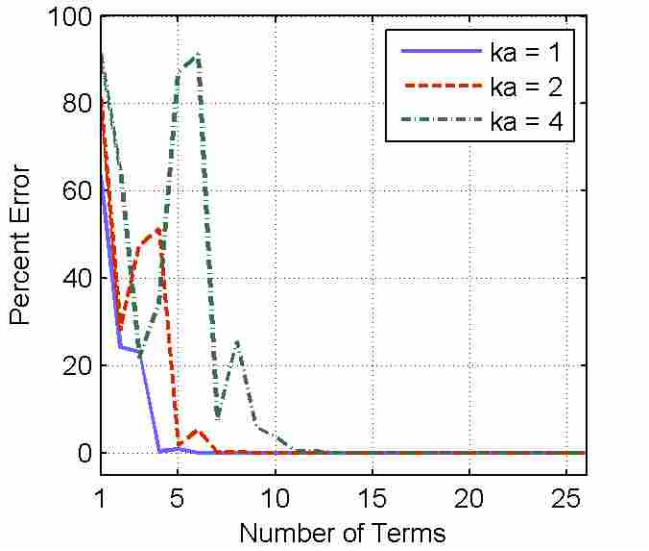

Figure 2-4: Magnitude of Percent Error in the Scattered Pressure Calculation as a Function of the Number of Terms Used in the Infinite Expansion.

The measurement error of any probe design is dependent on the angle of incidence of the travelling plane wave in relation to the probe. Certain angles of incidence correspond to underestimation of intensity while others correspond to overestimation. This is illustrated in Figure 2-5(a) which shows the intensity magnitude error of the orthogonal probe type "S/1" as an example. The figure shows the error (in dB) of the probe estimate relative to the actual intensity

magnitude when the probe is exposed to plane waves at different angles of incidence for $\frac{3}{2}ka = \frac{\pi}{2}$. For example, if the monochromatic plane wave is incident on the top of the probe, it is expected that the probe will underestimate the intensity magnitude by about 2.3 dB. The colorbar indicates that most angles of incidence result in underestimation; however, overestimation is present as well. As both are undesirable only the absolute values of the errors are examined in this study, resulting in the situation illustrated in Figure 2-5(b).

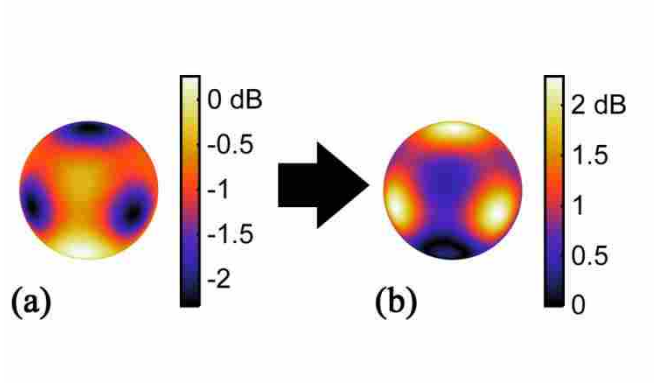

Figure 2-5: Intensity Magnitude Errors as a Function of Incidence Angle (a) and Corresponding Absolute Value Errors (b) for Probe Type "S/1" at the Highest Frequency Considered.

Three indicative metrics are used for probe type comparison. First, maximum error, corresponding to the worst possible probe orientation, is plotted to show the extreme errors associated with each implementation. For example, this metric is calculated to be 2.3 dB for Figure 2-5, corresponding to the error seen at four different angles of incidence (one on the backside is not seen). Second, an average error metric is used. This metric is a measure of the magnitude of how much error is expected if the probe were to be randomly oriented in a sound field and is calculated as the mean of the dB error magnitudes. In the example of Figure 2-5, the

mean error magnitude is 1.1 dB. Third, the effects of different processing methods on the spread between the maximum and minimum intensity magnitude errors are summarized. This metric must be calculated before taking the absolute value of the errors; in the example of Figure 2-5, it is seen to be about 2.5 dB. The intensity magnitude error spread is a useful metric because it indicates how well the probe type can be calibrated for magnitude. Better calibration can be achieved if the spread is smaller.

The simulation calculated intensity error at angles of incidence that were equally spaced using the angular spherical coordinates θ and φ. Figure 2-6 shows the simulated incidence angles as the points of intersection of the latitude and longitude lines.

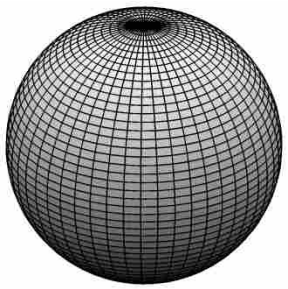

Figure 2-6: Angles of Incidence Considered Shown as the Intersection Points of Lines of Latitude and Longitude.

The error, in dB for intensity magnitude, at an incidence angle (θ_i,φ_j) was calculated using the following:

$$I_{mag\ err\ (\theta_i,\varphi_j)} = \left| 10 \log \frac{\|\mathbf{I}_{est\ (\theta_i,\varphi_j)}\|}{\|\mathbf{I}_{exact}\|} \right|, \tag{2-31}$$

where the estimated intensity magnitude $\|\mathbf{I}_{est\ (i,j)}\|$ is specific to incidence angle (θ_i,φ_j) while the actual intensity magnitude $\|\mathbf{I}_{exact}\|$ is not. The intensity direction error was calculated in degrees using

$$I_{dir\ err\ (\theta_i,\varphi_j)} = \frac{180}{\pi}\cos^{-1}\left(\frac{\mathbf{I}_{est\ (\theta_i,\varphi_j)} \cdot \mathbf{I}_{exact}}{\|\mathbf{I}_{est\ (\theta_i,\varphi_j)}\|\|\mathbf{I}_{exact}\|}\right), \tag{2-32}$$

where the angle is calculated by dividing the dot product by the magnitude values.

However, this setup results in more angles of incidence near the poles as can be seen in Figure 2-6, so an average error value would favor these areas of incidence angle. The average error metric was therefore calculated by taking the absolute value of the errors at all incidence angles and then multiplying these errors by an appropriate weighting function, w_i:

$$\overline{I_{err}} = \frac{\sum_{j=1}^{m}\sum_{i=1}^{n} w_i \left|I_{err\ (\theta_i,\varphi_j)}\right|}{m\sum_{i=1}^{n} w_i}, \tag{2-33}$$

where I_{err} is either $I_{mag\ err}$ or $I_{dir\ err}$ with $n = \frac{\theta_{step}}{\pi} + 1$ and $m = \frac{\varphi_{step}}{\pi} + 1$. This is similar to the process used by Leishman *et al.*[26] and Monson *et al.*[27] in which equal-angle, three-dimensional directivity measurements were area-weighted.

The weighting function converted the results from the equal-angle spacing used in the simulation to equal-area spacing, more appropriate in calculating average error as it does not favor the poles. The weighting function is given by:

$$w_i = \cos\theta_i - \cos\theta_{i+1}, \tag{2-34}$$

with

$$\theta_i = \begin{cases} 0, & i = 1 \\ \theta_{step}(i-2) + \frac{\theta_{step}}{2}, & i = 2,3,\ldots,n-1 \\ \pi, & i = n \end{cases}. \tag{2-35}$$

A suitable step size was found to be $\theta_{step} = \varphi_{step} = \frac{\pi}{50}$. The step size needed to be suitably small to achieve a fine enough mesh of sample points, but also needed to be chosen carefully because certain step sizes (such as $\theta_{step} = \varphi_{step} = \frac{s\pi}{4}$ for $s = 1,2,3, ...$) led to some very large errors when using the Taylor approximation. Using these step sizes resulted in one specific sample point that had very large errors due to the geometric properties of the probe and the Taylor approximation processing method for calculating intensity. Therefore using an appropriate step size meant avoiding such particular angles of incidence from being sampled, thus avoiding the large errors and giving more meaningful results. These large errors were seen to affect the maximum error metric but not the average error metric as the one large-error angle of incidence was "averaged out" with all the other angles of incidence. However, no large-error problems were seen to occur for the step size $\pi/50$.

2.7 Results

As discussed previously, it was found that for intensity the results using three separate pressure estimates with the three points velocity estimate—that is, using the probe as essentially 3 one-dimensional probes ("3 1D")—were equivalent to those when using the pressure of the origin microphone with a three points velocity estimate ("1"). For this reason probe type "3 1D" is not plotted. Five types of pressure estimates multiplied by two types of velocity estimates (one using and one not using the Taylor expansion), minus one because two of these methods are equivalent results in nine processing methods. These processing methods can be used with probes mounted in a sphere or not, making 18 total probe types whose errors are plotted.

2.7.1 Intensity Magnitude Errors

Figure 2-7 shows the average and maximum intensity magnitude errors of the probe types not mounted in a sphere. In all of the following figures, graphs (a) and (c) correspond to the particle velocity being estimated by a three points velocity estimate whereas graphs (b) and (d) correspond to the velocity estimate using the first-order Taylor approximation. The different pressure estimates are given as the different lines in the plots. The top two graphs (a) and (b) show average errors and the bottom two (c) and (d) show maximum errors.

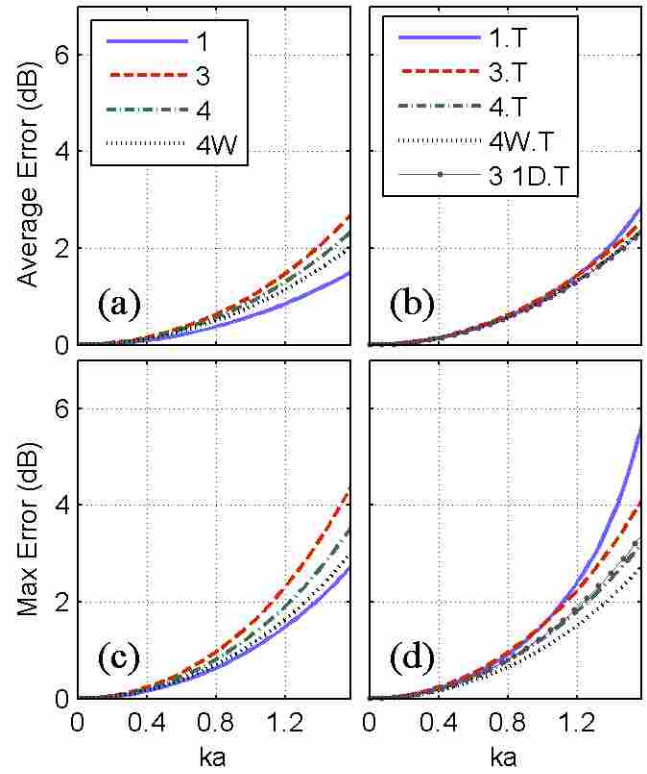

Figure 2-7: Average Intensity Magnitude Errors for Freely Suspended Designs Using Either the Three Points (a) or Taylor Approximation (b) Velocity Estimate and Corresponding Maximum Errors (c) and (d).

Graphs (a) and (c) of Figure 2-7 show that the lowest average and maximum errors when a three points velocity estimate is used result from using the pressure of the origin microphone ("1"). Conversely, when the Taylor approximation is used, the origin microphone pressure gives the most error ("1.T"). Comparing left and right graphs, lower errors are seen when a three points velocity estimate is used instead of the Taylor approximation. However, all average errors differ by less than 2 dB at the highest frequency considered. Overall, the best processing method for calculating intensity magnitude when not using a sphere is to use a three points velocity estimate and the origin microphone for the pressure estimate ("1").

Results for the probe types with microphones mounted in a sphere are shown in Figure 2-8 and show similar results. Again, the best combination is to use the origin microphone pressure with a three points velocity estimate ("S/1").

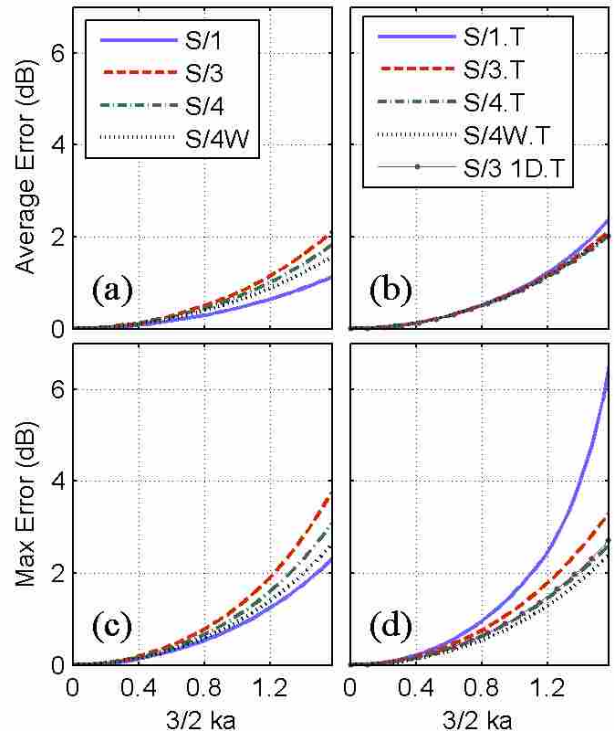

Figure 2-8: Average Intensity Magnitude Errors for Spherical Designs Using Either the Three Points (a) or Taylor Approximation (b) Velocity Estimate and Corresponding Maximum Errors (c) and (d).

Comparing Figures 2-7 and 2-8 reveals that overall the spherical designs perform slightly better than their freely suspended counterparts. The probe type that exhibits the least average and max error in calculating intensity magnitude is shown to result from using the origin pressure microphone without the Taylor approximation with the microphones embedded on a sphere ("S/1").

2.7.2 Intensity Direction Errors

The errors in estimating the intensity direction are important because often three-dimensional intensity measurements are used for source localization. In these situations, the direction errors tend to be unacceptable at an upper-frequency limit that is lower than that for the magnitude errors. The direction errors are plotted here in degrees, referring to the angle between the three-dimensional actual vector intensity and the vector intensity estimated by the probe. The errors for the probe types not mounted in a sphere are shown in Figure 2-9.

Probe type "1," where only one microphone is used for the pressure estimate, shows a few degrees more error in everything but the max error in the three points velocity estimate case—graph (c). Using the average of three microphones works best when the Taylor approximation is used ("3.T"). Comparing left to right graphs shows that, in contrast to the intensity magnitude results, the Taylor approximation probe types outperform the three points types and so the probe type with the lowest error is shown to be "3.T".

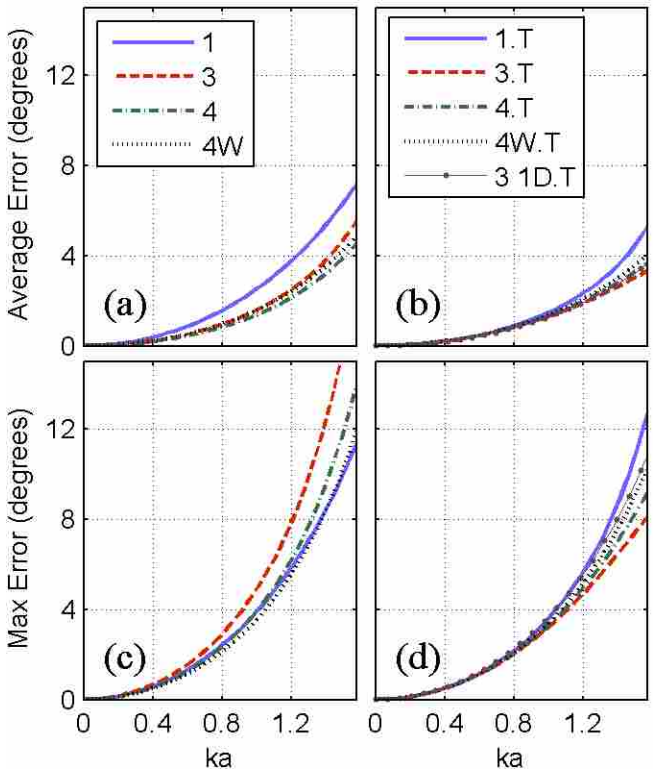

Figure 2-9: Average Intensity Direction Errors for Freely Suspended Designs Using Either the Three Points (a) or Taylor Approximation (b) Velocity Estimate and Corresponding Maximum Errors (c) and (d).

The direction errors for probe types mounted in a sphere are shown next in Figure 2-10. Of the pressure estimates, the three microphone average ("3" and "3.T") exhibits the lowest errors by a fraction of a degree in all but graph (c). The Taylor approximation is seen to be advantageous, resulting in a few degrees less error. Comparing Figures 2-9 and 2-10, the spherical designs all exhibit equal or less error than the corresponding freely suspended designs. The lowest-error method for estimating intensity direction is thus to use a spherical probe with the pressure being the average of the three non-origin microphones and using the Taylor approximation for the particle velocity ("S/3.T"). A summary of probe types that had the lowest errors is shown in Table 2-2.

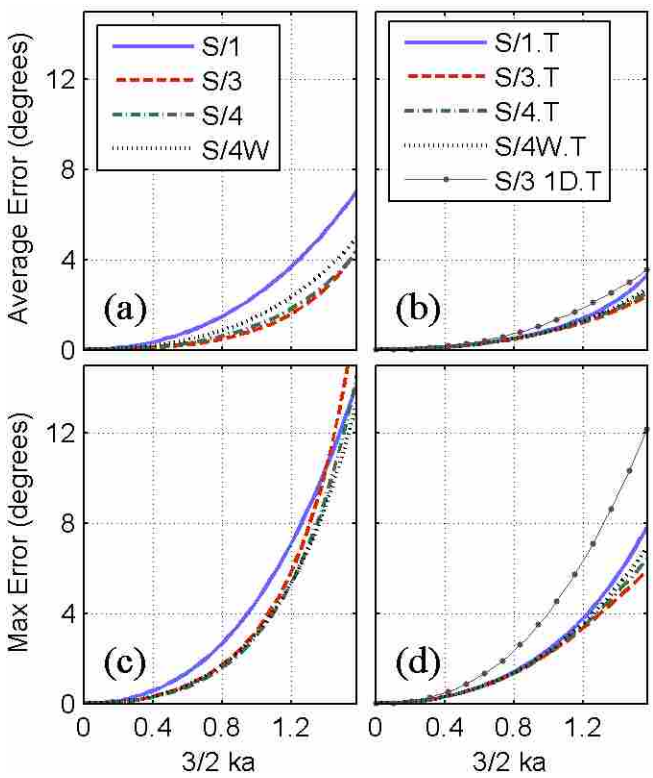

Figure 2-10: Average Intensity Direction Errors for Spherical Designs Using Either the Three Points (a) or Taylor Approximation (b) Velocity Estimate and Corresponding Maximum Errors (c) and (d).

Table 2-2: Probe Type with Least Error for Each Probe Design and Quantity of Interest.

Quantity	Non-Sphere Design	Sphere Design	Overall
Intensity Magnitude	1	S/1	S/1
Intensity Direction	3.T	S/3.T	S/3.T

2.7.3 Effect of Using Taylor Approximation

The effect of using the Taylor approximation for the velocity estimate instead of the three points estimate can be analyzed by looking at the difference in error between any Taylor approximation probe type and its corresponding three points probe type. Table 2-3 shows these results for average intensity magnitude error, intensity magnitude error spread, and average intensity direction error at the highest frequency considered. Examining the errors only at the highest frequency considered is sufficient as all error curves have been seen to be monotonically increasing with generally no one curve crossing over any other curve; that is, the probe types generally remain in the same order of highest to lowest error independent of frequency. A positive value indicates that using the Taylor approximation led to more error than using the three points estimate. For example, the first row of the table shows that when using one microphone for the pressure estimate, using the Taylor approximation (probe type “1.T”) gives 1.36 dB more error in estimating intensity magnitude than using the three points estimate (probe type “1”).

The table indicates that using the Taylor approximation is beneficial by up to a few degrees when estimating intensity direction, but not beneficial for intensity magnitude (except for probe type “3”). The magnitude error spread is benefitted by the Taylor approximation in half of the cases.

Table 2-3: Error Difference Between Taylor Approximation and Three Points Probe Types at the Highest Frequency Considered.

Probe Type	Magnitude (dB)	Magnitude Spread (dB)	Direction (degrees)
1	1.36	2.46	-1.87
3	-0.13	1.92	-2.19
4	0.02	0.92	-0.86
4W	0.35	-0.27	-0.83
3 1D	0.84	0.65	-3.50
S/1	1.24	3.09	-1.15
S/3	0.00	-0.02	-0.36
S/4	0.19	-1.00	-0.07
S/4W	0.47	-1.84	-0.23
S/3 1D	0.88	-1.16	-1.38

2.7.4 Effect of Mounting Microphones in Sphere

A similar analysis can be done to compare spherical and freely suspended designs. Table 2-4 shows the error difference between spherical and freely suspended designs at the highest frequency considered. A positive value indicates that the spherical design resulted in more error than its freely suspended counterpart. As seen before, mounting the microphones in a sphere is seen to result in less error for both intensity magnitude and direction. However, for magnitude spread, only some of the probe types are benefitted by the spherical scattering.

Table 2-4: Error Difference Between Spherical and Freely Suspended Probe Designs at the Highest Frequency Considered.

Probe Type	Magnitude (dB)	Magnitude Spread (dB)	Direction (degrees)
1	-0.36	0.69	-4.34
3	-0.57	0.49	-3.73
4	-0.50	0.66	-2.98
4W	-0.45	0.75	-3.10
1.T	-0.48	1.32	-3.62
3.T	-0.44	-1.45	-1.90
4.T	-0.33	-1.25	-2.18
4W.T	-0.32	-0.82	-2.50
3 1D.T	-0.32	-1.13	-2.22

2.8 Conclusions

Looking at general trends in the analytical simulations shows that spherical designs exhibit lower error than their freely suspended counterparts. Also seen is that using the Taylor approximation for the particle velocity estimate gives better probe accuracy than the three points estimate for intensity direction, but worse accuracy for intensity magnitude. The idea of using a weighted average of the microphones to put the pressure estimate at a point nearer to the three points where the three points velocity is estimated ("4W" and "S/4W") is shown to be useful in that it gives the lowest max errors for intensity direction. However, methods other than "4W" exhibit lower max errors for intensity magnitude and lower average errors for both magnitude and direction. The idea of using a three-microphone pressure estimate when the Taylor

approximation is used ("3.T" and "S/3.T") to get the velocity and pressure estimates as close together as possible does lead to the lowest errors for direction. However, it does not for magnitude.

If the quantity of interest is intensity magnitude, the best results come from probe type "S/1": a spherical design using the origin microphone for the pressure estimate and the three points velocity estimate. Alternatively, if intensity direction is desired, probe type "S/3.T" gives lowest error: a spherical design with pressure estimated by the average of the three non-origin microphones and the Taylor approximation used to estimate velocity. This study suggests then that lowest errors are attained when the microphones of the orthogonal probe are mounted on the surface of a sphere and if, using computational tools, the data recorded from the microphones is processed one way for intensity magnitude and another for intensity direction. A prototype of this type of probe is described by Locey[12] and Oldham.[28]

The results of this study are useful in showing the relative merit of each processing method; however, most all of the average intensity magnitude errors were within 1 to 2 dB of each other at the highest frequency considered, a fairly negligible amount. For intensity direction, the best probe types were about four degrees better than the worst types, which is also close to negligible, but likely more significant than the intensity magnitude error spread.

For the intensity magnitude results, average error curves similar to those shown in this paper could be advantageously used as calibration curves for any particular processing method, effectively decreasing the maximum error at any frequency by the amount of average error at that frequency. However, instead of using the absolute value average error, the signed average error would need to be used for the calibration. A similar process would not work for intensity direction, as the incidence angle is a two-dimensional quantity, whereas the intensity direction

error is a one-dimensional angle. Further work could examine the direction errors in terms of the two components θ and φ, possibly allowing for the development of intensity direction calibration curves.

3 ORTHOGONAL, REGULAR TETRAHEDRON, AND SIX-MICROPHONE PROBE COMPARISON

This chapter presents a paper submitted to the Journal of the Acoustical Society of America for archival publication. The formatting was modified to make it consistent with this thesis.

3.1 Contributing Authors

Curtis P. Wiederhold
Department of Mechanical Engineering, Brigham Young University, Provo, Utah 84602

Kent L. Gee
Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602

Jonathan D. Blotter
Department of Mechanical Engineering, Brigham Young University, Provo, Utah 84602

Scott D. Sommerfeldt
Department of Physics and Astronomy, Brigham Young University, Provo, Utah 84602

3.2 Abstract

Three common designs of multimicrophone probes used to measure acoustic intensity are the four-microphone regular tetrahedron, the four-microphone orthogonal, and the six-microphone designs. Various finite-sum and finite-difference processing methods exist for each of these designs. An analytical comparative analysis is performed to investigate the errors associated with each probe design and processing method. Probes consisting of matched point sensor

microphones both embedded and not embedded on the surface of a rigid sphere are considered. Results are given for plane wave fields in terms of average and maximum errors as a function of angle of incidence.

3.3 Introduction

In the late 1970s, a method was developed to estimate active intensity in one dimension by taking the cross-spectral density of two closely-spaced microphone signals.[1,2,20] This process, called the p-p technique,[29] estimates pressure at the point midway between the two microphones as the average of the two and estimates the particle velocity at the same point using a finite-difference approximation. With pressure and particle velocity estimated, active intensity can be calculated, making the p-p technique a powerful technique in characterizing noise fields.

A two-microphone probe measures the component of the vector intensity along the line from one transducer's acoustic center to the other. To get a fully three-dimensional measurement of the intensity, the p-p technique has since been developed for use with probes consisting of four or more microphones. Using at least four microphones means the particle velocity can be estimated in three dimensions, allowing three-dimensional intensity as well as energy density to be calculated. Such probes are called multimicrophone probes in this work, but have also been referred to as vector probes, intensity probes, and energy density sensors. These multimicrophone probes have found wide use and exist in various configurations, the most common designs including the four-microphone regular tetrahedron design shown in Figure 3-1(a),[6-8] the six-microphone design shown in Figure 3-1(c),[4,5] and the four-microphone orthogonal design shown in Figure 3-1(e).[9,10] Typically, scattering of the sound field being measured by the probe has been avoided by making the fixture holding the microphones as small

as possible.[8,22] Alternatively, microphones have been mounted on the surface of a hard sphere, making the scattering predictable and possibly beneficial.[5,17] Such "spherical probes" are illustrated in Figures 3-1(b), (d), and (f).

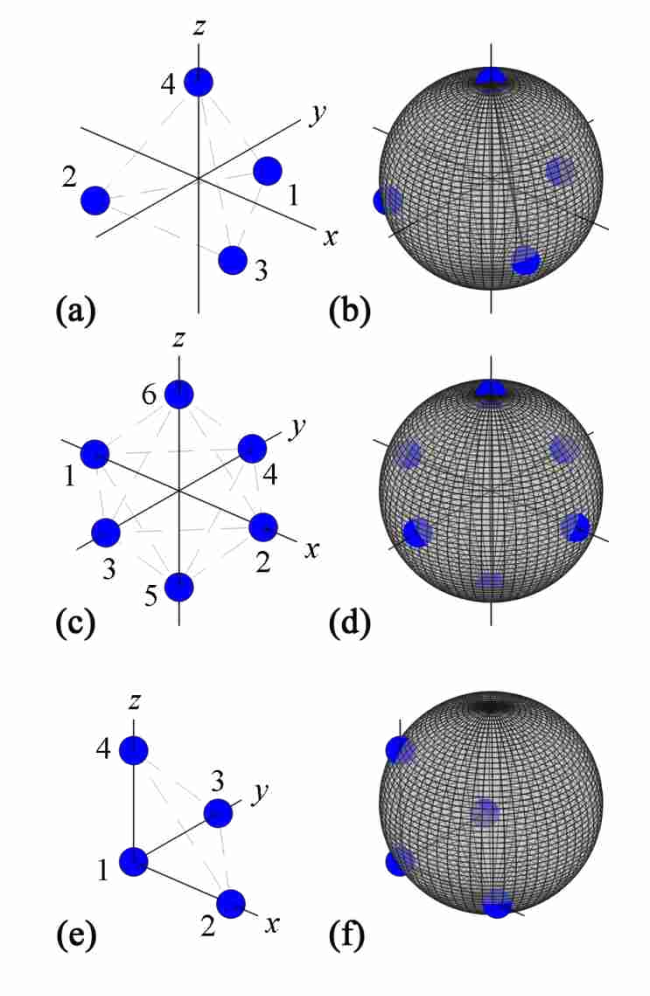

Figure 3-1: Probe Designs for Regular Tetrahedron (a), Six-Microphone (c), and Orthogonal (e) Probes and the Corresponding Designs (b), (d), and (f) with Microphones Embedded on the Surface of a Hard Sphere, Dots Representing Microphone Locations.

Even for multimicrophone probes consisting of perfectly phase and amplitude matched point sensors, the useable bandwidth is limited due to errors in the finite-sum and finite-difference approximations inherent in the p-p technique. These errors are dependent on design

and implementation factors. Design factors include the number of microphones used, the arrangement of the microphones relative to each other, and the fixture holding the microphones. Implementation factors include how the probe is oriented in the sound field and how the pressures recorded by the microphones are processed to calculate intensity or energy density.

This paper considers the relative merits of various multimicrophone probe designs and implementations. The measurement errors of multimicrophone probes consisting of perfect point sensors in plane wave fields will be analyzed. This work is similar to that of Pascal and Li,[11] but builds on it in that it examines probe implementations not previously considered and explores the effect of having the microphones mounted on a rigid sphere. Also, in addition to looking at maximum errors, an average error metric is used in evaluating these probes. This work serves as a companion paper to a work that investigated orthogonal probes.[30] The results from that paper are here compared to results from regular tetrahedron and six-microphone probes.

Section 3.4 develops the processing methods used by the regular tetrahedron and six-microphone probes. Section 3.5 gives the cross-spectral formulas associated with each processing method. The analytical model used for comparison of the probe designs and processing methods is described in Section 3.6. The results of the comparison are found in Section 3.7, and concluding remarks in Section 3.8.

3.4 Intensity Estimation Methods

Letting p be the complex pressure and $\mathbf{v}$ the complex particle velocity, the active intensity is given by:

$$\mathbf{I} = \frac{1}{2}\mathrm{Re}\{p\mathbf{v}^*\}, \tag{3-1}$$

where "*" denotes the complex conjugate. With a multimicrophone probe there exist a number of ways to estimate this quantity, mainly arising from the various ways *p* can be estimated. The processing methods will be explained for the regular tetrahedron and the six-microphone probe. The processing methods for the orthogonal probe can be found in a companion paper.[30] (Chapter 2 in this thesis).

3.4.1 Regular Tetrahedron Probe Estimation Methods

The regular tetrahedron probe consists of four microphones located at the vertices of a tetrahedron whose faces are all equilateral triangles. In this arrangement all the microphones are equidistant from each other. The microphones of this probe have either been implemented in a "freely suspended"[31] fashion as illustrated in Figure 3-1(a) or mounted on the surface of a hard sphere[12] as illustrated in Figure 3-1(b).

The particle velocity can be obtained using the time-harmonic linear Euler's equation (using the $e^{j\omega t}$ time convention):

$$\mathbf{v} = \frac{j\nabla p}{\rho\omega}, \tag{3-2}$$

where j is the imaginary unit $\sqrt{-1}$, ρ the density of the fluid, and ω the angular frequency. The gradient of the pressure is estimated by a finite-difference approximation using the systematic method developed by Pascal and Li.[11] It is dependent on the location of the microphones in the coordinate system and for the configuration shown in Figures 3-1(a) and (b) is given by:

$$\mathbf{v} = \begin{cases} v_x \approx \dfrac{\sqrt{6}j(p_3 - p_2)}{4a\rho\omega} \\ v_y \approx \dfrac{\sqrt{2}j(2p_1 - p_2 - p_3)}{4a\rho\omega} \\ v_z \approx \dfrac{j(3p_4 - p_1 - p_2 - p_3)}{4a\rho\omega} \end{cases} . \tag{3-3}$$

where p_i is the pressure at the i^{th} microphone and a the radius of the sphere that is circumscribed by the four microphones. The other needed quantity in Eq. (3-1) is the pressure p and is typically approximated as the average of the four microphones:

$$p \approx \frac{p_1 + p_2 + p_3 + p_4}{4} ; \tag{3-4}$$

however, it is unclear what the effect would be if instead the pressure from a single microphone was used. As this type of pressure estimate led to low intensity magnitude errors for the orthogonal probe,[30] this pressure estimate is examined in this paper and is given by:

$$p \approx p_1, \tag{3-5}$$

The two pressure estimates combined with the consideration of whether to have the microphones embedded on the surface of a sphere or not lead to four combinations of regular tetrahedron probe types that will be considered.

3.4.2 Six-Microphone Probe Estimation Methods

The six-microphone probe consists of six microphones arranged along the three Cartesian axes and can also be freely suspended[4] as shown in Figure 3-1(c) or mounted on the surface of a hard sphere[5] as shown in Figure 3-1(d). It was suggested as an alternative to the four-microphone probes as its traditional use required only each pair of microphones to be matched (as opposed to four matched microphones) to get an accurate measurement. This is because, in the traditional use, the intensity in each Cartesian direction was estimated using only the two microphones

along that axis. The six-microphone probe design is also easier to orient in a sound field than the regular tetrahedron probe as each two-microphone pair lies along an orthogonal axis. Having three matched pairs allows a measurement of the three orthogonal components of the intensity vector to be taken. For example, the *x*-direction intensity is estimated by:

$$I_x \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1 + p_2}{2} v_x^*\right\}. \tag{3-6}$$

The pressure estimate for this direction of the intensity is seen to be the average pressure of the two microphones along the *x*-axis. The *y*- and *z*-intensities are calculated similarly, using only the two microphones along each axis. Thus, there is a different pressure estimate for the *x*-, *y*-, and *z*-directions, unlike the regular tetrahedron probe where only one global pressure value was used.

Pascal and Li[11] showed that substantially less bias results for the six-microphone probe in measuring intensity if just one pressure estimate is used for all orthogonal directions and is given as the average of the pressures of all six microphones:

$$p \approx \frac{p_1 + p_2 + p_3 + p_4 + p_5 + p_6}{6}\ . \tag{3-7}$$

However, this method requires that all six microphones be matched for an accurate measurement. For this work, a third considered pressure estimate is examined where the pressure from just one microphone is used:

$$p \approx p_1\ . \tag{3-8}$$

Microphone one is used here, but, since all angles of incidence are considered, using any other microphone would give equivalent results in this analysis. This pressure estimate also requires all six microphones to be matched.

The particle velocity can be calculated using Euler's equation as before. The needed pressure gradient is again calculated using a finite-difference estimate resulting in the following:

$$\mathbf{v} = \begin{cases} v_x \approx \dfrac{j(p_2 - p_1)}{2a\rho\omega} \\ v_y \approx \dfrac{j(p_4 - p_3)}{2a\rho\omega} \\ v_z \approx \dfrac{j(p_6 - p_5)}{2a\rho\omega} \end{cases} . \tag{3-9}$$

Another method to estimate intensity using the six-microphone probe was developed by Moschioni *et al.*[14] The typical velocity estimate given in Eq. (3-9) results in using just the microphone pairs that lie along the three orthogonal directions, even though there are a total of 15 microphone pairs in a six-microphone probe. These other 12 pairs have a microphone separation distance that is smaller than that of the three on-axis pairs by a factor of $1/\sqrt{2}$. This smaller separation distance extends the upper limit in using a finite-difference approximation. This fact leads to the suggestion that all microphone pairs in the six-microphone probe be used to estimate the intensity by using an averaging procedure. A finite-sum pressure estimate and a finite-difference velocity estimate between each microphone pair results in 15 intensity estimates. Each of these estimates is a component of the full vector intensity along the axis of each microphone pair. These 15 intensities are averaged with frequency-dependent weighting factors given by Moschioni *et al.*:

$$\mathbf{I} = \begin{cases} I_x \approx \alpha I_{12} + \beta(I_{13} + I_{14} + I_{15} + I_{16} + I_{32} + I_{42} + I_{52} + I_{62}) \\ I_y \approx \alpha I_{34} + \beta(I_{31} + I_{32} + I_{35} + I_{36} + I_{14} + I_{24} + I_{54} + I_{64}) \\ I_z \approx \alpha I_{56} + \beta(I_{51} + I_{52} + I_{53} + I_{54} + I_{16} + I_{26} + I_{36} + I_{46}) \end{cases}, \tag{3-10}$$

where I_{ij} is the intensity from microphone i to microphone j and α and β are weighting factors given by:

$$\alpha = \frac{\sin^2\left(\frac{2a\omega}{c}\right)}{4\sin^2\left(\frac{2a\omega}{\sqrt{2}c}\right) + \sin^2\left(\frac{2a\omega}{c}\right)}$$
$$\beta = \frac{\sin^2\left(\frac{2a\omega}{\sqrt{2}c}\right)}{4\sqrt{2}\sin^2\left(\frac{2a\omega}{\sqrt{2}c}\right) + \sqrt{2}\sin^2\left(\frac{2a\omega}{c}\right)}\ , \tag{3-11}$$

where c is the sound speed. This procedure was found by Moschioni *et al*. to increase the bandwidth of the six-microphone probe and so will be considered here.

In summary, four processing methods for the six-microphone probe are examined. First, using it as 3 one-dimensional probes; second, using the average pressure of the six microphones; third, using microphone one for the pressure estimate; and fourth, using all microphone pairs and averaging the resulting components of the intensity. These four methods, along with the design consideration of whether or not to embed the microphones on the surface of a sphere, account for the eight processing methods for the six-microphone probe that will be compared in this analysis.

3.4.3 Summary of Estimation Methods

For simplification an abbreviation scheme is used to refer to each probe type. The regular tetrahedron probe is abbreviated as "4R" and the six-microphone probe as "6". If the microphones of the probe are mounted on the surface of a sphere, the abbreviation "S" is used. The pressure estimate is abbreviated as the number of microphones averaged for the estimate: "1" if one microphone is used, etc. When the six-microphone probe is used as 3 one-dimensional probes, the abbreviation "3 1D" is used and when the average intensity of all 15 microphone pairs is used the abbreviation is given as "15 1D". A forward slash is used to separate the design from the implementation considerations. Table 3-1 gives a summary of all considered probes and processing methods.

Table 3-1: Summary of Considered Probe Types and Their Abbreviations.

Abbreviation	Probe Type	Scattering	Pressure Estimate
4R/1	Reg. tetrahedron	None	$p = p_1$
4R/4	Reg. tetrahedron	None	$p = \frac{p_1+p_2+p_3+p_4}{4}$
4R.S/1	Reg. tetrahedron	Spherical	$p = p_1$
4R.S/4	Reg. tetrahedron	Spherical	$p = \frac{p_1+p_2+p_3+p_4}{4}$
6/1	Six-microphone	None	$p = p_1$
6/6	Six-microphone	None	$p = \frac{p_1+p_2+p_3+p_4+p_5+p_6}{6}$
6/3 1D	Six-microphone	None	$p_x = \frac{p_1+p_2}{2}, p_y = \frac{p_3+p_4}{2}, p_z = \frac{p_5+p_6}{2}$
6/15 1D	Six-microphone	None	15 pressure estimates
6.S/1	Six-microphone	Spherical	$p = p_1$
6.S/6	Six-microphone	Spherical	$p = \frac{p_1+p_2+p_3+p_4+p_5+p_6}{6}$
6.S/3 1D	Six-microphone	Spherical	$p_x = \frac{p_1+p_2}{2}, p_y = \frac{p_3+p_4}{2}, p_z = \frac{p_5+p_6}{2}$
6.S/15 1D	Six-microphone	Spherical	15 pressure estimates

Results for the regular tetrahedron and six-microphone probes are compared to the results given in the companion paper[30] for orthogonal probes. Abbreviations used in this paper for orthogonal probes are as given in the companion paper with the extra abbreviation "4O" indicating that the probe type is the four-microphone orthogonal probe.

3.5 Cross-Spectral Formulations

Calculating intensity in practice is most commonly done in terms of auto- and cross-correlations between microphone signals. The cross-spectral expressions for all probe types listed in Table 3-1 are given, including some already given in Pascal and Li.[11] They are all given here for completeness.

The one-sided cross-spectral density for a zero-mean process is defined as:

$$\begin{aligned} G_{mn}(\omega) &= C_{mn}(\omega) + jQ_{mn}(\omega) \\ &= \lim_{T\to\infty} \frac{2}{T} \mathrm{E}[P_m^*(\omega,T)P_n(\omega,T)] \,, \end{aligned} \tag{3-12}$$

for $\omega \geq 0$, where C and Q are respectively the real and imaginary parts of the cross-spectral density G. The expectation operator is denoted by E[ ] and P_m and P_n are the Fourier transforms of the pressures from the m^{th} and n^{th} microphones over time T. The equations are given for probes consisting of freely suspended microphones. If instead the intensity is to be calculated for probes consisting of microphones embedded on a sphere, the radius a must be multiplied by 3/2 as shown by Elko.[17] For example, the expression for probe type "4R.S/1" is the same as that for probe type "4R/1" with $\frac{3}{2}a$ substituted for a. These expressions are specific to the particular microphone numbering system and orientation of the probe in the coordinate system and so these expressions are valid only for the probes as defined in Figure 3-1. Though not explicitly written in the following equations for brevity, all cross-spectral densities (and hence intensities) are functions of frequency.

The expression for probe type “4R/1” is given by:

$$\mathbf{I}_{4\mathrm{R}/1}=\begin{cases} I_x \approx \dfrac{\sqrt{6}}{4a\rho\omega}(-Q_{21}+Q_{31}) \\ I_y \approx \dfrac{\sqrt{2}}{4a\rho\omega}(-Q_{21}-Q_{31}) \\ I_z \approx \dfrac{1}{4a\rho\omega}(-Q_{21}-Q_{31}+3Q_{41}) \end{cases} . \tag{3-13}$$

“4R/4”:

$$\mathbf{I}_{4\mathrm{R}/4}=\begin{cases} I_x \approx \dfrac{\sqrt{6}}{16a\rho\omega}(-Q_{21}+Q_{31}+2Q_{32}+Q_{42}-Q_{43}) \\ I_y \approx \dfrac{\sqrt{2}}{16a\rho\omega}(-3Q_{21}-3Q_{31}-2Q_{41}+Q_{42}+Q_{43}) \\ I_z \approx \dfrac{1}{4a\rho\omega}(Q_{41}+Q_{42}+Q_{43}) \end{cases} . \tag{3-14}$$

“6/1”:

$$\mathbf{I}_{6/1}=\begin{cases} I_x \approx \dfrac{1}{2a\rho\omega}(Q_{21}) \\ I_y \approx \dfrac{1}{2a\rho\omega}(-Q_{31}+Q_{41}) \\ I_z \approx \dfrac{1}{2a\rho\omega}(-Q_{51}+Q_{61}) \end{cases} . \tag{3-15}$$

“6/6”:

$$\mathbf{I}_{6/6}=\begin{cases} I_x \approx \dfrac{1}{12a\rho\omega}(2Q_{21}+Q_{31}-Q_{32}+Q_{41}-Q_{42}+Q_{51}-Q_{52}+Q_{61}-Q_{62}) \\ I_y \approx \dfrac{1}{12a\rho\omega}(-Q_{31}-Q_{32}+Q_{41}+Q_{42}+2Q_{43}+Q_{53}-Q_{54}+Q_{63}-Q_{64}). \\ I_z \approx \dfrac{1}{12a\rho\omega}(-Q_{51}-Q_{52}-Q_{53}-Q_{54}+Q_{61}+Q_{62}+Q_{63}+Q_{64}+2Q_{65}) \end{cases} \tag{3-16}$$

"6/3 1D":

$$\mathbf{I}_{6/3\ 1\mathrm{D}} = \begin{cases} I_x \approx \dfrac{Q_{21}}{2a\rho\omega} \\ I_y \approx \dfrac{Q_{43}}{2a\rho\omega} \\ I_z \approx \dfrac{Q_{65}}{2a\rho\omega} \end{cases} . \tag{3-17}$$

The expressions for the "15 1D" probe types are given as the average of intensities from all two-microphone pairs of the probe. The intensity between any two microphones *i* and *j* is estimated as:

$$\mathbf{I}_{ij} \approx \frac{Q_{ij}}{d\rho\omega} , \tag{3-18}$$

where *d* is the distance between the two microphones ($d = 2a$ for three of the microphone pairs and $d = \sqrt{2}a$ for the other 12 microphone pairs). These intensities are then averaged according to Eqs. (3-10) and (3-11).

3.6 Analytical Model

The model used for comparison simulates plane waves incident on matched point sensor probes. Simulating ideal probes in plane wave fields serves as a starting point for analysis and gives an indication of the relative merit between probe types. The angle of incidence of the plane wave on the probe affects the probe's accuracy and so multiple angles of incidence are considered. It was found that using a step size of $\pi/50$ in the θ and φ spherical directions of the angle of incidence is sufficient in calculating the metrics used for comparison.

Three metrics are used for probe comparison. The first is maximum error, which corresponds to the angle of incidence that results in the most measurement error for a given probe type at a given frequency. If a probe was randomly put in a plane wave field, this metric

gives a bound to the largest error that would be expected as a function of frequency. The second metric is average error. This metric represents the amount of error that would be expected on average if a probe were randomly oriented in a plane wave field. These two metrics are examined in measuring both the magnitude and direction of the intensity vector. The third metric is that of the spread of the intensity magnitude errors. It is the difference between the maximum and the minimum intensity magnitude errors. The spread is an indication of how well the probe type can be calibrated for intensity magnitude; better calibration can be achieved is the spread is smaller.

The intensity direction error is defined as the angle (in degrees) between the angle of incidence of the incoming plane wave and the estimated angle of incidence as measured by the probe. The equation for this is:

$$I_{dir\ err\ (\theta_i,\varphi_j)} = \frac{180}{\pi}\cos^{-1}\left(\frac{\mathbf{I}_{est\ (\theta_i,\varphi_j)} \cdot \mathbf{I}_{exact}}{\|\mathbf{I}_{est\ (\theta_i,\varphi_j)}\|\|\mathbf{I}_{exact}\|}\right), \tag{3-19}$$

where the dot product between $\mathbf{I}_{est\ (\theta_i,\varphi_j)}$, the estimated intensity vector at incidence angle (θ_i,φ_j), and $\mathbf{I}_{exact}$, the actual intensity, is taken and then divided by the magnitudes of the two vectors. The intensity magnitude error is given in dB and defined as follows:

$$I_{mag\ err\ (\theta_i,\varphi_j)} = \left|10\log\frac{\|\mathbf{I}_{est\ (\theta_i,\varphi_j)}\|}{\|\mathbf{I}_{exact}\|}\right|. \tag{3-20}$$

The absolute value is taken as both under and overestimation of the intensity magnitude are undesirable. This is necessary also so an average error can be calculated. Without doing so, negative errors would average out with positive errors. It is also necessary to multiply the errors by an appropriate weighting function to convert the results from an "equal angle" approach arising from using a constant step size in the θ and φ coordinates to an "equal area" approach that is appropriate for calculating average error.[26,27,30]

The probe types either have the microphones mounted on a sphere or freely suspended in space. In the sphere-mounted case, spherical scattering is accounted for;[16] however, scattering off any fixture holding the sphere is neglected. In the case of freely suspended microphones, scattering is neglected entirely as it would be dependent on the geometry of the fixture holding the microphones. Hence, it is expected that the results from the simulation will match experimental results more closely for the spherical probes than for the freely suspended probes.

The spherical scattering has been shown to effectively increase the separation distance between microphones at low frequencies, and so, the 3/2 correction factor must be used as mentioned earlier for the finite-difference expressions to give accurate results. The dimensionless frequency chosen for plotting results from freely suspended probes is ka where k is the wavenumber and a the radius of an imaginary sphere that is circumscribed by the four microphone points. For spherical probes, the errors are plotted as a function of $\frac{3}{2}ka$ where a is the radius of the sphere on which the microphones are mounted. Because of the needed correction factor, the spherical probe is directly compared to a freely suspended probe that is 3/2 times larger, an approach used by both Elko[17] and Parkins *et al.*[5] Hence, the errors at any frequency ka of a freely suspended probe are directly compared to the errors at $\frac{3}{2}ka$ for a spherical probe.

3.7 Results and Discussion

3.7.1 Regular Tetrahedron Probe Results

The regular tetrahedron probe is examined first—errors in estimating intensity magnitude are shown in Figure 3-2 and intensity direction in Figure 3-3. In the following figures the top two plots, (a) and (b), show average errors and the bottom two, (c) and (d), the maximum errors. The left plots, (a) and (c), are for probes with microphones freely suspended in space (no scattering);

whereas the right plots, (b) and (d), show errors for probes with microphones mounted on a sphere (spherical scattering).

Graphs (a) and (b) of Figure 3-2 show that using one microphone (probe types “4R/1” and “4R.S/1”) for the pressure estimate results in slightly better intensity magnitude average error. However, in Figure 3-2(c) and (d) it is shown that there are angles of incidence that give higher maximum error than using the average of the four microphones for the pressure estimate (probe types “4R/4” and “4R.S/4”). Comparing left and right graphs in Figure 3-2 shows that mounting the microphones in a sphere results in lower intensity magnitude error for the regular tetrahedron probe.

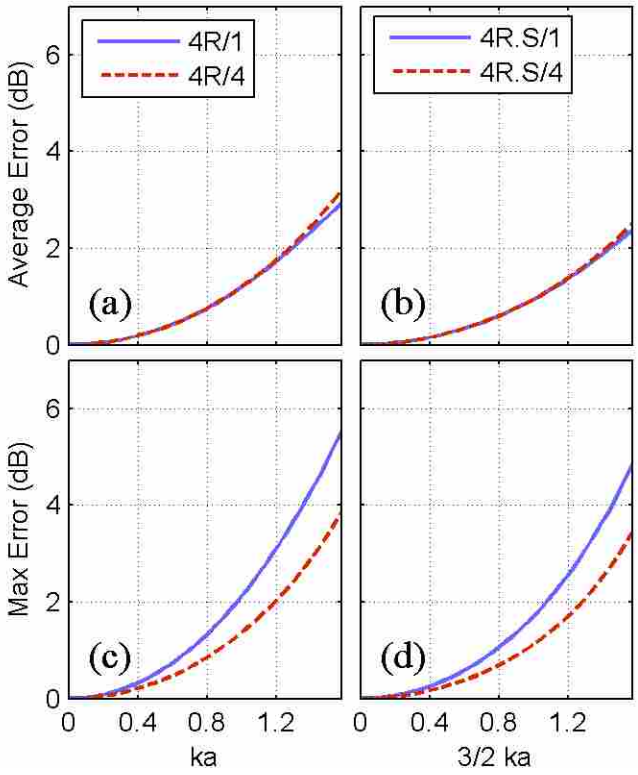

Figure 3-2: Average Intensity Magnitude Errors for Freely Suspended (a) and Spherical (b) Regular Tetrahedron Probe Designs and Corresponding Maximum Errors (c) and (d).

For intensity direction, Figure 3-3 indicates that the one microphone pressure estimate gives significantly worse average and maximum results than using the average of the four microphones. Comparing left and right graphs in Figure 3-3 shows that, as before, mounting the microphones in a sphere gives improved results.

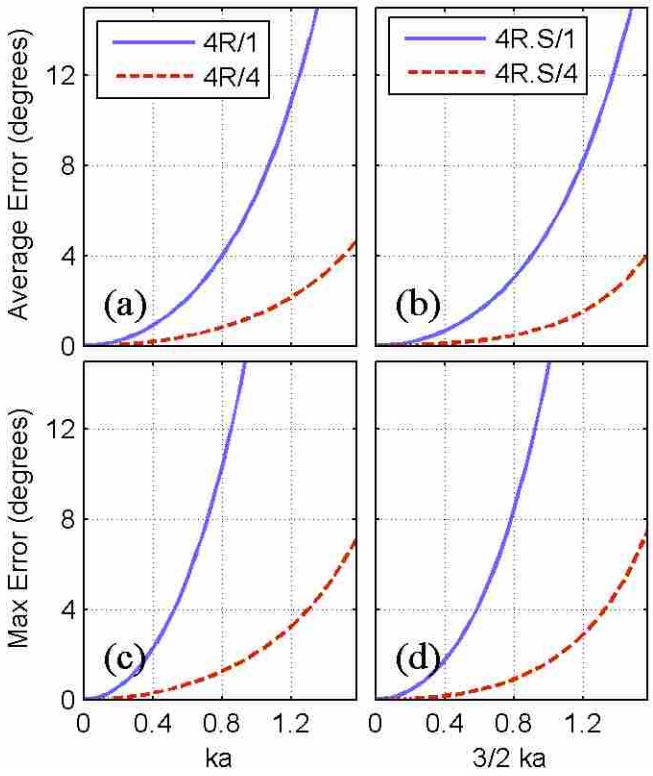

Figure 3-3: Average Intensity Direction Errors for Freely Suspended (a) and Spherical (b) Regular Tetrahedron Probe Designs and Corresponding Maximum Errors (c) and (d).

3.7.2 Six-Microphone Probe Results

The six-microphone probe results are shown in Figures 3-4 and 3-5. For this probe, the best processing results are shown to come from the method of averaging intensity estimates from all 15 microphone pairs (“6/15 1D” and “6.S/15 1D”). Spherical scattering is seen to result in

lower error in all cases. Some of the graphs have overlapping lines: “6/1” and “6/3 1D” overlap in Figure 3-4(c) while “6/1” overlaps with “6/6” in Figures 3-5(a), (b), and (c). In Figure 3-5(c) the maximum intensity direction error for probe type “6/1” is seen to rise sharply at approximately $ka = \frac{\pi}{2}$; once this frequency limit is reached, there are angles of incidence that result in very high intensity direction error for both “6/1” and “6/3 1D”.

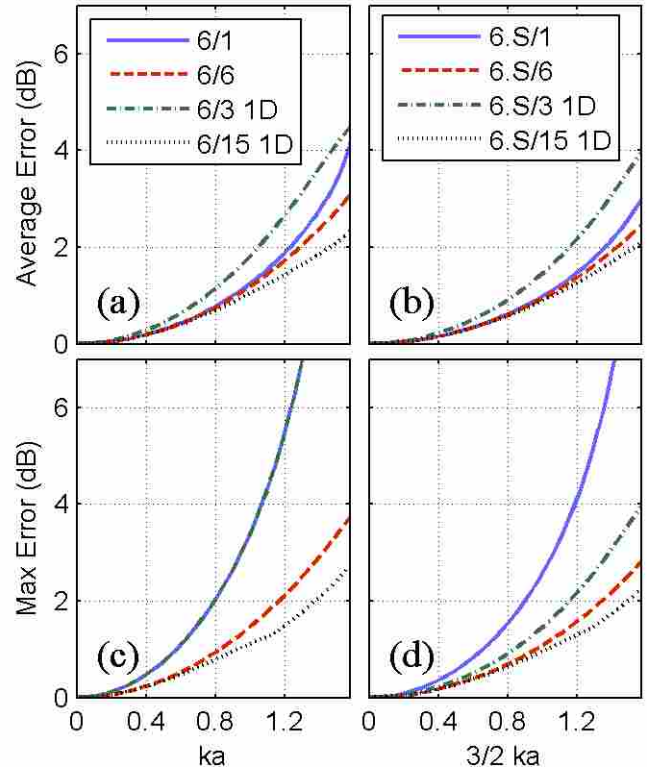

Figure 3-4: Average Intensity Magnitude Errors for Freely Suspended (a) and Spherical (b) Six-Microphone Probe Designs and Corresponding Maximum Errors (c) and (d).

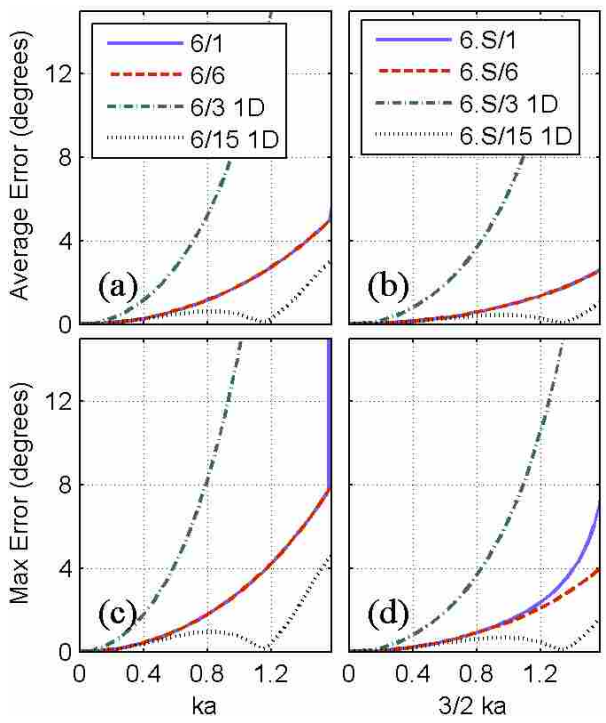

Figure 3-5: Average Intensity Direction Errors for Freely Suspended (a) and Spherical (b) Six-Microphone Probe Designs and Corresponding Maximum Errors (c) and (d).

3.7.3 Comparison of Probe Geometries

The lowest-error processing methods of the different probe geometries are now compared. For the regular tetrahedron geometry, the processing method resulting in the lowest error was to use the average pressure of the four microphones ("4R/4" and "4R.S/4"). For the six-microphone geometry, the lowest-error processing method was using all 15 intensities ("6/15 1D" and "6.S/15 1D"). Previous work[30] shows that for the orthogonal geometry the lowest intensity magnitude error resulted from using one microphone for the pressure estimate ("4O/1" and "4O.S/1"). For intensity direction, the lowest error resulted from using a weighted average of the four microphones for the pressure and a Taylor approximation of the velocity ("4O/4W.T" and "4O.S/4W.T"). These lowest-error processing methods for each probe design are compared for intensity magnitude in Figure 3-6 and intensity direction in Figure 3-7.

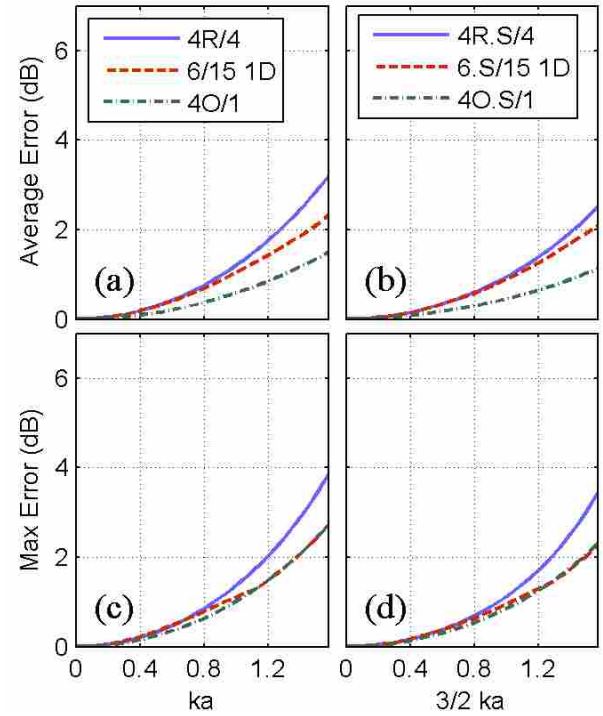

Figure 3-6: Average Intensity Magnitude Errors for the Lowest-Error Freely Suspended (a) and Spherical (b) Probe Designs and Corresponding Maximum Errors (c) and (d).

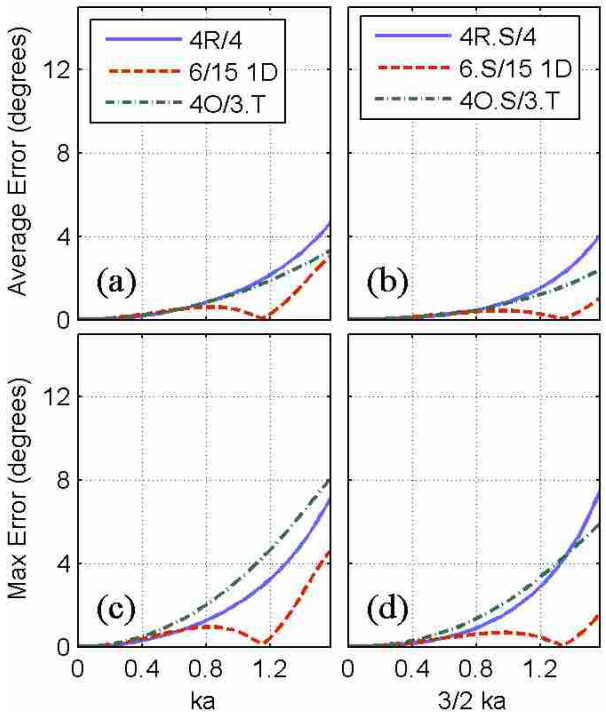

Figure 3-7: Average Intensity Direction Errors for the Lowest-Error Freely Suspended (a) and Spherical (b) Probe Designs and Corresponding Maximum Errors (c) and (d).

The orthogonal probe types (“4O/1” and “4O.S/1”) are shown in Figure 3-6 to, on average, measure the intensity magnitude the most accurately by about 1 or 2 dB at the highest frequency considered. However, in Figure 3-7 the six-microphone probe is shown to be more accurate for intensity direction. In these figures, spherical designs all exhibit lower average error than their freely suspended counterparts.

3.7.4 Effect of Mounting Microphones in Sphere

To analyze more directly the effect of mounting the microphones in a sphere, the error difference between each spherical design and its corresponding freely suspended design is given in Table 3-2. The errors reported are only for the highest frequency considered, which is generally an indication of the errors at all other lower frequencies. This is true because the error curves all follow roughly the same monotonically increasing trend, except for the “6/15 1D” probe type. A positive value indicates that the spherical design resulted in more error than its freely suspended counterpart. For example, the first row of the table shows that for the regular tetrahedron using one microphone for the pressure estimate, mounting the microphones in a sphere leads to 0.55 dB less error in estimating intensity magnitude at the highest frequency considered. The results are similar to those found for the orthogonal probe types—the spherical designs exhibit lower error for all probe types for both magnitude and direction. The magnitude spread is also seen to be reduced by mounting the microphones in a sphere for all probe types except “4R/4.”

Table 3-2: Error Difference Between Spherical and Freely Suspended Probe Designs at the Highest Frequency Considered.

Probe Type	Magnitude (dB)	Magnitude Spread (dB)	Direction (degrees)
4R/1	-0.55	-0.31	-15.9
4R/4	-0.68	0.09	-3.12
6/1	-1.14	-14.4	-3.40
6/6	-0.61	-0.44	-3.00
6/3 1D	-0.52	-9.35	-26.7
6/15 1D	-0.22	-0.43	-2.68

3.7.5 Summary of Lowest-Error Probe Types

A summary of the ten lowest-error probe types for each quantity is given in Table 3-3. The probe type is given with the amount of error of that type at the highest frequency considered in parentheses. The table shows that for intensity magnitude, the lowest errors are achieved with the orthogonal probe types. However, the magnitude spread is lowest for six-microphone probe types. For intensity direction, the lowest error comes from the "6.S/15 1D" probe type but the next five lowest-error types are all orthogonal probes.

Table 3-3: Ten Lowest-Error Probe Types for Each Quantity of Interest.

Rank	Magnitude (dB)	Magnitude Spread (dB)	Direction (degrees)
1	4O.S/1 (1.13)	6.S/15 1D (0.32)	6.S/15 1D (0.34)
2	4O/1 (1.49)	6.S/6 (0.57)	4O.S/3 1D.T (1.41)
3	4O.S/4W (1.57)	6/15 1D (0.75)	4O.S/3.T (1.42)
4	4O.S/4 (1.83)	6/6 (1.01)	4O.S/4.T (1.47)
5	4O.S/3 1D.T (2.01)	4O.S/4W.T (1.08)	4O.S/4W.T (1.52)
6	4O.S/4.T (2.02)	4R/4 (1.12)	4O.S/4 (1.54)
7	4O/4W (2.02)	4R.S/4 (1.21)	4R.S/4 (1.54)
8	4O.S/4W.T (2.04)	4O.S/3 1D.T (1.42)	4O.S/1.T (1.64)
9	6.S/15 1D (2.10)	4O/1 (1.89)	4O.S/4W (1.76)
10	4O.S/3 (2.11)	4O/4W.T (1.90)	4O.S/3 (1.78)

3.7.6 Calibration

Because the errors considered in this work can be predicted, calibration of the probe can occur in post-processing. An important consideration in calibrating probes, and thus in the relative merit of any particular probe type, is the error spread between the maximum and minimum error obtained as a function of angle of incidence. The smaller the error spread, the better a probe type can be calibrated for intensity magnitude. There are two ways the calibration could be done. First, the calibration could be to the mean between the maximum and minimum errors. This would effectively make the new maximum error at any frequency equal to one-half the value of the error spread. Second, the calibration could be to the error value averaged over all angles of incidence. This calibration is advantageous in that the average error at any frequency

would then be zero; however, the maximum error at any frequency would not necessarily be equal to one-half the value of the error spread.

It may also be possible for the probes to be calibrated for intensity direction, but the direction errors would have to be known in terms of the θ and φ spherical angles as opposed to merely the angle between the actual and estimated intensity vectors (which is what is reported in this work). However, for this to be possible there would need to be a one-to-one correspondence between each actual angle of incidence and estimated angle of incidence. That is, if one estimated angle of incidence corresponded to more than one actual angle of incidence, an accurate calibration would not be possible. If an intensity direction calibration were possible, a more accurate intensity magnitude calibration could also then be applied that was specific to the calibrated angle of incidence. This calibration method, however, is possibly not feasible in plane wave fields, not to mention reactive fields.

3.7.7 Definition of *kd*

Results given in this paper and similar analyses[5,11,30] are largely dependent on the choice of the length *d* in the dimensionless frequency *kd*. In the analysis done by Pascal and Li,[11] this distance for the regular tetrahedron is chosen to be the distance between any two microphones: for the six-microphone probe it is the distance between any two microphones that lie on the same axis (e.g. the distance between microphones 1 and 2), and for the orthogonal probe it is the distance between the "origin" microphone and any of the other three microphones. Therefore, any results reported in this paper which are also shown in Pascal and Li differ accordingly. The *x*-axis is scalable depending on how the dimensionless frequency is defined. While this approach

gives an indication of the relative merit of any particular probe design, it is not general because it is completely dependent on the definition of *d* for each probe geometry.

If phase mismatch were introduced into the comparison, the definition of the distance *d* could become less important. Accounting for phase mismatch introduces low-frequency errors because the phase error becomes large relative to the acoustic phase change between microphones due to the larger wavelengths. For example, two microphones separated by 0.03 m measure a 65 Hz tone with about a 2 degree phase difference. Therefore, a 1 degree phase mismatch between the two microphones would introduce large errors into the measurement; whereas, at 3,500 Hz, a 1 degree phase error is fairly negligible compared to the 110 degree phase difference resulting from a 0.03 m separation distance.

With phase mismatch introduced, any particular probe design then has a high-frequency limit as well as a low-frequency limit where the estimation error is greater than a chosen threshold. This is useful because the frequency range between the low- and high-frequency limit is independent of the definition of the separation distance *d*. For example, suppose a particular probe design was shown to have less than an arbitrary threshold of 2 degrees of error in estimating the intensity direction between a *kd* of 0.2 to 2 while another design was only able to meet the criteria from a *kd* of 0.2 to 1. In the first case, the high-frequency cutoff was 10 times greater than that of the low while, for the second case, it was 5 times greater. Comparing the value of 10 to 5 then indicates that the first probe has a larger bandwidth of low error than the second probe, regardless of the definition of the distance *d*. This then serves as a better indicator in comparing probe geometries.

However, this is left as future work. In this study, the value of *d* is chosen as it is because the idea is to compare both designs mounted in a sphere to those not in a sphere. For designs not

mounted in a sphere, the length d was chosen to be the radius of an imaginary sphere circumscribing the four or six sensors. For designs mounted in a sphere, results were plotted using the length $\frac{3}{2}a$ where a is the radius of the actual sphere.

3.8 Conclusions

The intensity measurement errors of various configurations of multimicrophone probes consisting of perfect point sensors were analyzed in plane wave fields over all angles of incidence. The probe type with the lowest errors for intensity magnitude was shown to be "4O.S/1"—the orthogonal probe with sensors mounted on a sphere, using the "origin" microphone for the pressure estimate. This orthogonal probe design had from 1 to 2 dB less average error than the lowest-error processing method for the regular tetrahedron or six-microphone designs at the highest frequency considered. However, if calibration is used the lowest-error probe type for intensity magnitude becomes "6.S/15 1D"—the six-microphone probe using a weighted average of the 15 intensities calculated from the 15 microphone pairs of the probe. The difference between the highest- and lowest-error processing methods at the highest frequency considered for average intensity magnitude error was about 2 dB for the six-microphone design and negligible for the regular tetrahedron design.

For intensity direction, the lowest errors resulted from probe type "6.S/15 1D." The average intensity direction error of this probe at the highest frequency considered was about 2 to 3 degrees less than the lowest-error processing methods of the other two probe geometries. The difference between the highest- and lowest-error processing methods was much more substantial for intensity direction than for magnitude as certain processing methods resulted in very large direction errors.

Mounting sensors in a sphere was shown to give lower errors than having the sensors freely suspended for all probe geometries and processing methods. As scattering was neglected for the freely suspended case, it is expected that experimental results for spherical probes would more closely match the analytical results presented in this paper than would experimental results for freely suspended probes. However, the lower errors of spherical designs were all less than 1 dB for average intensity magnitude errors at the highest frequency considered for all probe geometries and processing methods. For intensity direction, spherical designs had at most 2 degrees less average error at the highest frequency considered. Thus, spherical designs are shown to measure intensity with less error, but not significantly less.

Comparisons between probe designs are highly dependent on how the distance d is defined in the dimensionless frequency kd. Future work includes introducing phase mismatch errors and comparing the probes based on how large of bandwidth a probe type can measure before reaching a determined threshold error level. Future work also includes examining the feasibility of calibrating intensity direction errors.

4 PROBE DESIGN OPTIMIZATION

4.1 Introduction

Three probe designs—the four-microphone regular tetrahedron, the four-microphone orthogonal, and the six-microphone probe—have been analyzed and compared. In this chapter, optimization techniques are developed to predict what an optimal four-microphone probe design would be in a user-specified measurement situation. The three probes considered previously have designs that are simple and generally symmetrical, making them relatively intuitive and easy to analyze. The designs of the four-microphone orthogonal and six-microphone probes also make them easy to orient in a sound field. However, there is no guarantee that these three designs necessarily represent the lowest-error design possible.

Optimization techniques can be employed to suggest a probe design that would give the lowest errors for a particular situation. Factors that would affect an optimal probe design would be the frequency range of interest, the acoustic quantity of interest, the type of sound field (e.g. active or reactive), and phase and amplitude mismatch of the microphones. The optimization situation can be highly specific (e.g. error in intensity magnitude at one frequency) or can be more general (e.g. sum of the errors in intensity magnitude and direction over a chosen frequency range). While this flexibility makes such optimization techniques powerful, it also makes it difficult to know how best to utilize them.

Optimization is performed here for four-microphone probes with matched point sensors. Plane wave fields are considered over all angles of incidence. The optimization is limited to analyzing probes mounted on a sphere as any optimization performed for freely suspended designs merely results in designs where the sensors are as close together as possible, a result that occurs because the sensors are perfectly matched. With matched sensors and no scattering, the errors from the finite-sum and finite-difference estimates of the pressure and particle velocity can be minimized by simply minimizing the separation distance between the sensors.

Section 4.2 gives an overview of the optimization problem and describes the genetic algorithm used. Results of optimization runs are given in Section 4.3 and conclusions and suggestions for future work are given in Section 4.4.

4.2 Optimization Techniques

The probe design optimization consists of three parts: the objective function, the design variables, and the constraints. The objective function is the real-valued function which is to be minimized. The design variables are the inputs to the objective function—the variables that are adjusted in the optimization. The constraints are the limits put on the design variables.

The objective function for this optimization is chosen to be the amount of error resulting from any particular probe design and so is to be minimized in the optimization. The objective function can be chosen to be the error in the calculation of pressure, particle velocity, intensity, and/or energy density; at any frequency or frequency range; at any or all angles of incidence.

The design variables are the θ and φ angles of the four microphone positions. As the radius of the sphere is fixed, each microphone position can be specified by two coordinates, giving a total of eight design variables.

The first constraint is the range of values that the design variables are allowed to take. As they represent spherical coordinates, the θ angles are constrained to be between 0 and π radians while the φ angles must be between 0 and 2π radians. The other constraint is that the four microphones cannot all lie in the same plane. If they do, a three-dimensional estimation of the particle velocity is not possible, and the design is labeled infeasible. The optimization algorithm is designed to avoid such designs by assigning them a penalty function. That is, if such a design is encountered, an arbitrarily large value is output by the objective function.

For the objective function to be evaluated for any given microphone arrangement, a finite-difference estimate of the pressure gradient must be calculated. An estimate is made according to the systematic method described by Pascal and Li.[11] First, the input design variables are translated from spherical coordinates to Cartesian coordinates. Then, a matrix **M** is constructed as follows:

$$\mathbf{M} = \begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix}, \tag{4-1}$$

where the position of the i^{th} microphone is given by (x_i,y_i,z_i). A matrix of pressures is also needed and given by:

$$\mathbf{P} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \tag{4-2}$$

where p_i is the pressure at the i^{th} microphone. Inverting **M** and multiplying it by **P** then gives a finite-difference estimate of the pressure gradient according to:

$$\mathbf{D} = \begin{bmatrix} p \\ \dfrac{\partial p}{\partial x} \\ \dfrac{\partial p}{\partial y} \\ \dfrac{\partial p}{\partial z} \end{bmatrix} \approx \mathbf{M}^{-1}\mathbf{P} \tag{4-3}$$

The last three rows are the three orthogonal components of the pressure gradient. The first row is a weighted average of the four microphone pressures—weighted in such a manner as to put the estimate of the pressure at the origin of the coordinate system. Though the method varies somewhat, the gradient estimations in this process are equivalent to those output by the method developed by Thomas,[19] and represent a "finite-difference, least-squares estimate of the gradient."

The estimates of the pressure and pressure gradient can then be used to calculate intensity or energy density, which require particle velocity and pressure estimates. The particle velocity is estimated using the estimated pressure gradient in the time-harmonic linear Euler's equation (using the $e^{j\omega t}$ time convention):

$$\mathbf{v} = \frac{j\nabla p}{\rho\omega}, \tag{4-4}$$

where j is the imaginary unit, ρ the density of the fluid, and ω the angular frequency. The pressure estimate used is the average pressure of the four microphones:

$$p \approx \frac{p_1 + p_2 + p_3 + p_4}{4}. \tag{4-5}$$

which is only equal to the quantity given in the first row of Eq. (4-3) if the four microphones are in the regular tetrahedron configuration. While Eq. (4-5) is estimating the pressure at the centroid of the four microphone points, row one of Eq. (4-3) is estimating the pressure at the origin point (the center of the sphere). While the pressure estimate of Eq. (4-3) may be used or any of the

pressure estimates discussed in Chapters 2 and 3, the average pressure is chosen here as it was seen to result in lower errors (averaged over angle of incidence) in most cases for intensity calculation for the regular tetrahedron and orthogonal designs when a velocity estimate using Eq. (4-3) was used. This average pressure estimate resulted in higher errors only in calculating intensity magnitude with the orthogonal design, but at the highest frequency considered $\left(\frac{3}{2}ka = \frac{\pi}{2}\right)$ this error was less than 1 dB higher than the other pressure estimates.

The constraints of the optimization are given mathematically by:

$$0 \leq \theta_i \leq \pi$$

$$0 \leq \varphi_i < 2\pi$$

$$\text{rank}(\mathbf{M}) = 4 \tag{4-6}$$

where θ_i and φ_i are the spherical coordinates of the i^{th} microphone and the rank of the matrix **M** is constrained to be four (that is, the matrix must be full-rank).

The angle of incidence step size resolution was chosen to be π/20. Though the resolution used in Chapters 2 and 3 was π/50, a coarser resolution was desired for the computational load to not become prohibitive. It was seen that for the processing method used (velocity estimate given by Eq. (4-3), pressure estimate given as the average of the four microphones), π/20 is an adequate step size. Also in the interest of computational load, only 15 terms were used in the infinite series expansion to calculate pressure scattered off the sphere. Figure 2-4 shows that this still gives accurate pressure calculations up to $ka = 4$.

4.2.1 Genetic Algorithm

As the design variables of this optimization problem are continuous, gradient-based optimization techniques could be used. Of the gradient-based methods, an algorithm capable of

performing constrained nonlinear optimization was needed. Of these, the active-set algorithm was chosen as it lends itself well to the dense nature of the matrices in this optimization problem. Central-difference derivatives were used to improve accuracy in the algorithm. While viable results were obtained using this method, it was found that in general slightly better results were found using a genetic algorithm technique. Also, the genetic algorithm used arrived at solutions significantly quicker. It is likely that the genetic algorithm method worked better because of its ability to avoid local minima, of which there are many in this problem.

The genetic algorithm is a heuristic optimization method that mimics evolutionary biology, effectively searching large design spaces and "evolving" toward the global optimum. A useful general description of genetic algorithms is given by Duke,[32] while more detailed information can be found in many texts. The important characteristics of the genetic algorithm implemented for this optimization include selection, crossover, mutation, and elitism.

A large population of randomly chosen designs is created and then evaluated using the objective function. Selection then refers to the process of choosing which of these designs will be used to create the next generation of designs. Tournament selection is used—a process where the best design of a random subset of the population is selected. The best design is defined to be the one that returns the smallest value from the objective function. This selection process is repeated until a sufficient number of designs are chosen for crossover to be performed.

Crossover is then performed on the chosen designs. The style of crossover used in this optimization problem involves each design creating a new design in a one-to-one correspondence (that is, one "parent" design makes one "child" design). It is a method that has been termed "parthenogenesis" by Duke and is described mathematically by:

$$y_i = \alpha r_i + x_i \,, \tag{4-7}$$

where x_i is the i^{th} design variable of the parent design, y_i is the i^{th} design variable of the child design, and r_i is a zero-mean normally distributed random number whose standard deviation controls how much change can occur in the crossover process. The variable α is a dynamic factor that also controls how much change occurs in crossover, but is not random in nature and changes as the algorithm progresses according to:

$$\alpha = \left(1 - \frac{n-1}{N}\right)^{\beta}, \tag{4-8}$$

where n is the number of the current generation, N is the number of total generations, and β is the user-specified factor that controls how dynamic α is in controlling the nature of the crossover.

Dynamic mutation is then performed on the child designs, a process wherein random changes are made in order to introduce more variety in the population. First, a random number is chosen that determines whether mutation is performed on a given child design. If mutation is to be performed, a uniformly distributed random number, $r_{i,mut}$, is then chosen between the maximum and minimum value of the design variable (between 0 and 2π for the φ angle variables and between 0 and π for the θ angle variables). If this number is greater than the current value of the design variable ($r_{i,mut} \geq y_i$) then the design variable is modified according to:

$$\begin{aligned} y_i &= y_{i,min} + \left(r_{i,mut} - y_{i,min}\right)^{\alpha}\left(y_i - y_{i,min}\right)^{1-\alpha} \\ &= {r_{i,mut}}^{\alpha} {y_i}^{1-\alpha} \end{aligned} \tag{4-9}$$

where $y_{i,min}$ is the minimum value allowable of the i^{th} design variable. Since this value is 0 for all design variables, the equation is simplified. The dynamic factor α is the same one used in crossover. If the random number is less than the current value of the design variable ($r_{i,mut} < y_i$) then the design variable is instead given by:

$$y_i = y_{i,max} + \left(y_{i,max} - r_{i,mut}\right)^{\alpha}\left(y_{i,max} - y_i\right)^{1-\alpha} \tag{4-10}$$

where $y_{i,max}$ is the maximum value allowable for the i^{th} design variable.

This mutation process is similar to the parthenogenesis crossover in that each selected design is modified to become the new child design. However, they differ importantly in how they modify each design. The crossover process slightly modifies each design; it occurs to a larger number of the designs but changes them very little, allowing any particular design to "settle" into a minimum, be it local or global. The mutation process, on the other hand, occurs to only a few designs in each generation; however, the change is drastic. This process introduces variability in each generation, helping the algorithm not get stuck in local minima.

Finally, elitism is performed wherein the child and parent designs are pooled together and compared using the objective function. The best designs of this pool become the new parent designs of the next generation and the algorithm iterates. After a given number of generations, the algorithm terminates.

4.3 Results

The results of a few optimization runs are presented in this section. All runs involve four-microphone spherical probe designs optimized to errors in measuring intensity.

4.3.1 Spherical Probe Optimized for Intensity Direction Maximum Error at $ka = \pi/3$

The genetic algorithm was used to minimize the error for intensity direction at $ka = \frac{\pi}{3}$, the highest frequency considered in Chapters 2 and 3. The maximum intensity direction error as a function of angle of incidence was used, though optimizing to average error gave similar results. The optimized design is shown in Figure 4-1. The design looks very similar to that of the regular tetrahedron design, though rotated. Comparing the intensity magnitude and direction errors of this design to those of the regular tetrahedron and orthogonal probe in Figure 4-2 shows that

indeed this design is essentially equivalent to the regular tetrahedron design (the lines overlap). The results are plotted as a function of $\frac{3}{2}ka$ for consistency with the figures in Chapters 2 and 3.

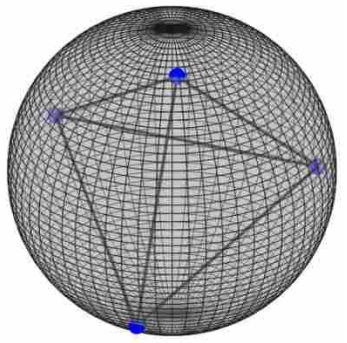

Figure 4-1: Optimized Design for Intensity Direction at $ka = \pi/3$.

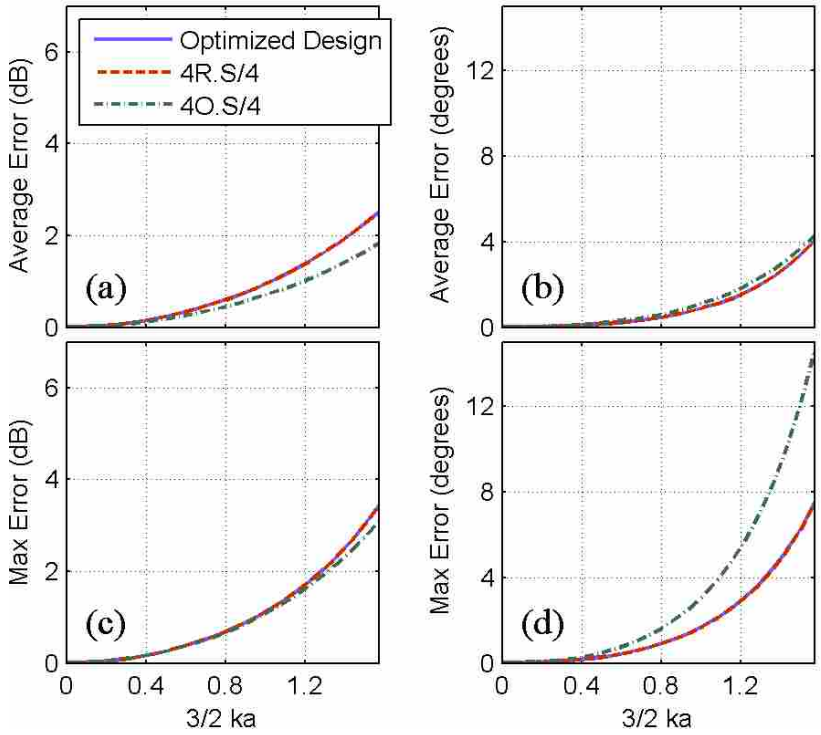

Figure 4-2: Intensity Magnitude Average (a) and Maximum (c) Errors and Intensity Direction Average (b) and Maximum (d) Errors Including the Errors of a Probe Optimized for Intensity Direction.

4.3.2 Spherical Probe Optimized for Intensity Magnitude Average Error at $ka = \pi/3$

If the design is optimized to intensity magnitude instead of intensity direction, the optimized design consists of all four microphones being clumped closely together, as shown in Figure 4-3. This design leads to very low intensity magnitude errors in this analytical study as shown as a function of frequency in Figure 4-5, but represents an infeasible design. This is because, first, the microphone size would limit how closely they could be spaced on a sphere and, second, mismatches between sensors placed closely together would lead to large errors, especially at lower frequencies.

Similar results were obtained for optimizations run at the lower frequency $ka = \frac{\pi}{48}$: when optimizing for intensity direction, the regular tetrahedron design resulted and when optimizing for intensity magnitude, an infeasible clumping of the microphones occurred.

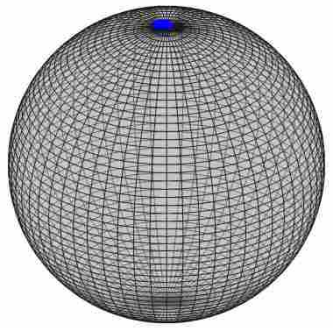

Figure 4-3: Optimized Design for Intensity Magnitude at $ka = \pi/3$.

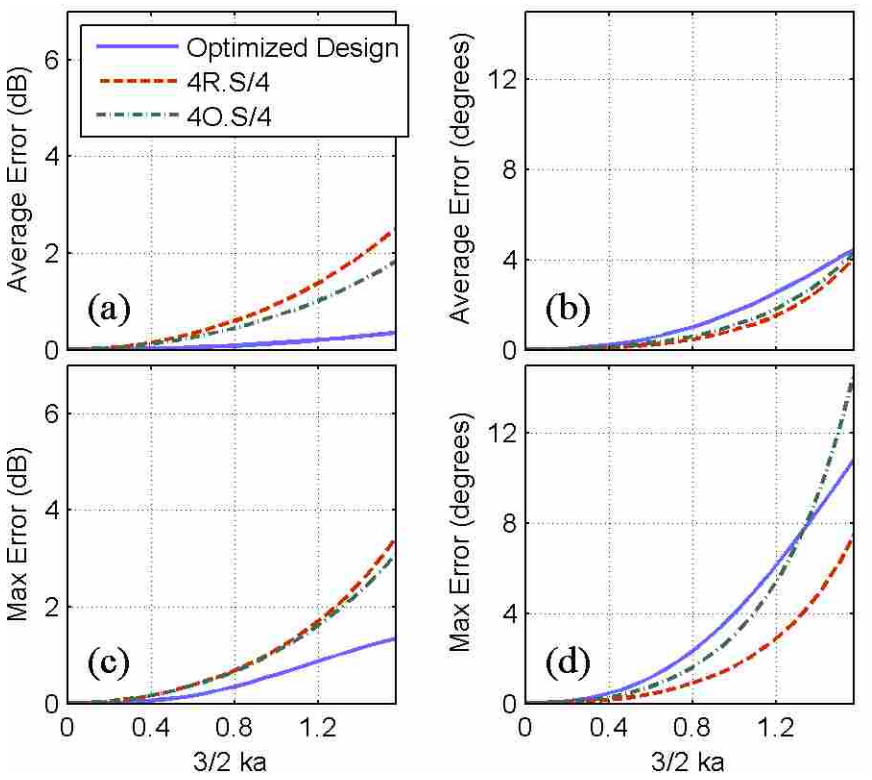

Figure 4-4: Intensity Magnitude Average (a) and Maximum (c) Errors and Intensity Magnitude Average (b) and Maximum (d) Errors Including the Errors of a Probe Optimized for Intensity Magnitude.

4.3.3 Spherical Probe Optimized for Intensity Direction Average Error at $ka = \pi/3$ in a Half Space

While the direction of the acoustic wave incident on a multimicrophone probe is typically not known, at times the general direction of the sound source may be known. This knowledge of the sound field can be advantageous and provide opportunities for improved performance. The genetic algorithm was used to optimize the probe for intensity direction average error when the sound is known to be incident on one of the two halves of the sphere. The design shown in Figure 4-5 results.

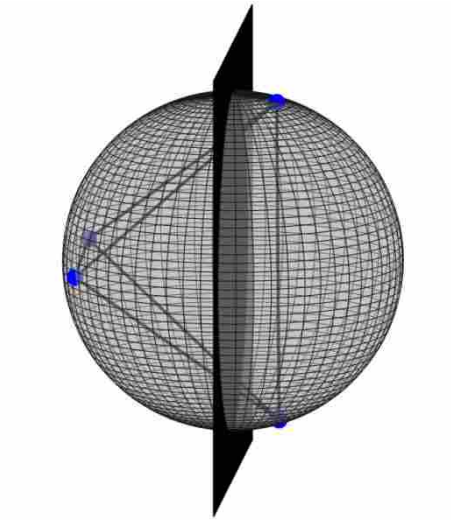

Figure 4-5: Optimized Design for Intensity Direction at $ka = \pi/3$ for a Half Space Where Errors Are Only Considered for the Right Half of the Sphere.

Only the errors on the right side of the sphere (sides of the sphere are indicated by the black plane) are considered in this half-space optimization. The intensity direction errors of this design at $ka = \frac{\pi}{3}$ as a function of angle of incidence are shown in Figure 4-6. It is seen that the optimized probe does have the lowest direction errors on the right half of the sphere; however, there are areas on the other side of the sphere that result in very high errors. The orthogonal probe, also an asymmetric design, exhibits this same quality to a degree. Figure 4-7 shows the errors associated with these same designs for intensity magnitude. The optimized design is seen to have areas of low error, but they do not align with the areas of low error for intensity direction.

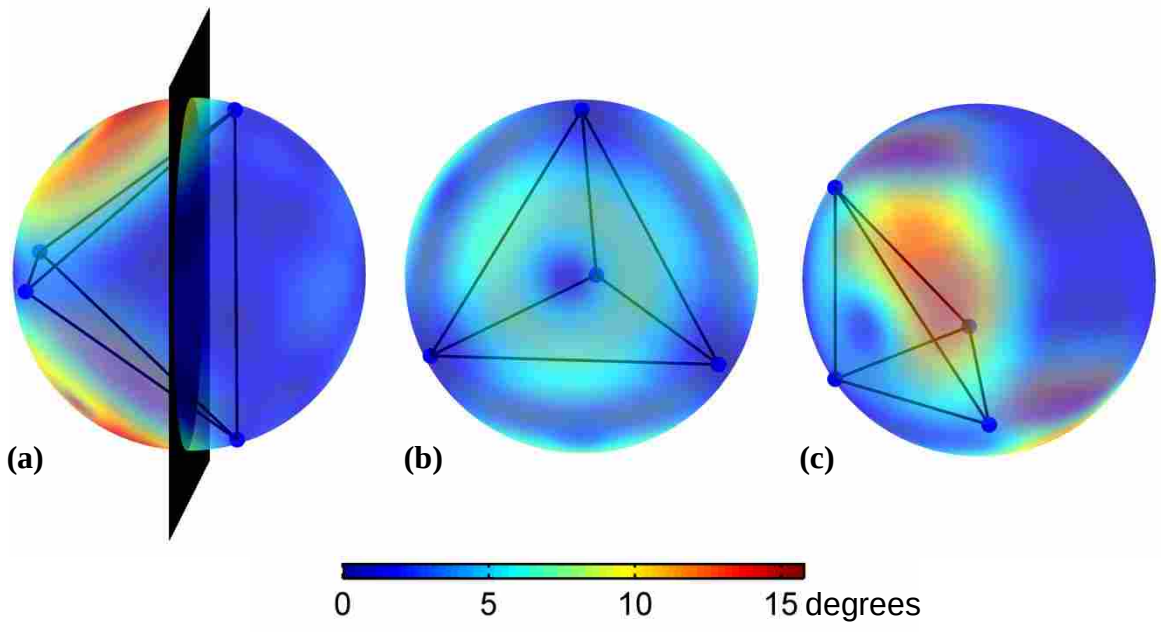

Figure 4-6: Intensity Direction Errors for Optimized (a), Regular Tetrahedron (b), and Orthogonal (c) Designs.

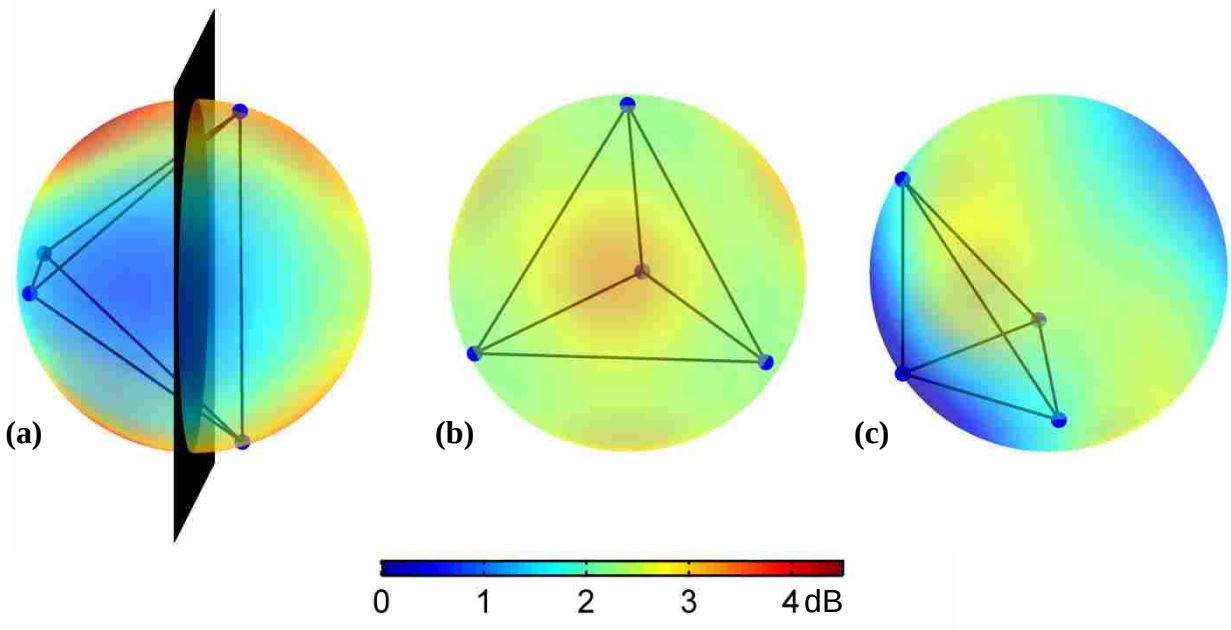

Figure 4-7: Intensity Magnitude Errors for Optimized (a), Regular Tetrahedron (b), and Orthogonal (c) Designs.

Plotting the errors as a function of frequency gives an idea of how such a probe would do at lower frequencies. Figure 4-8 shows the average and maximum errors of the three designs for only half of the angles of incidence, those on the right half of the spheres in Figures 4-6 and 4-7. This comparison is somewhat misleading, however, because the regular tetrahedron and orthogonal designs are not oriented in the sound field in a manner for them to give the lowest errors. But the plot does allow for superficial comparison. The plot shows that the optimized design does have the lowest errors in graph (b), average intensity direction, while not having significantly larger errors in graphs (b), (c), and (d). Thus, if some prior knowledge of source direction is had, an optimized probe can be used to achieve lower intensity direction errors without substantially increasing intensity magnitude errors.

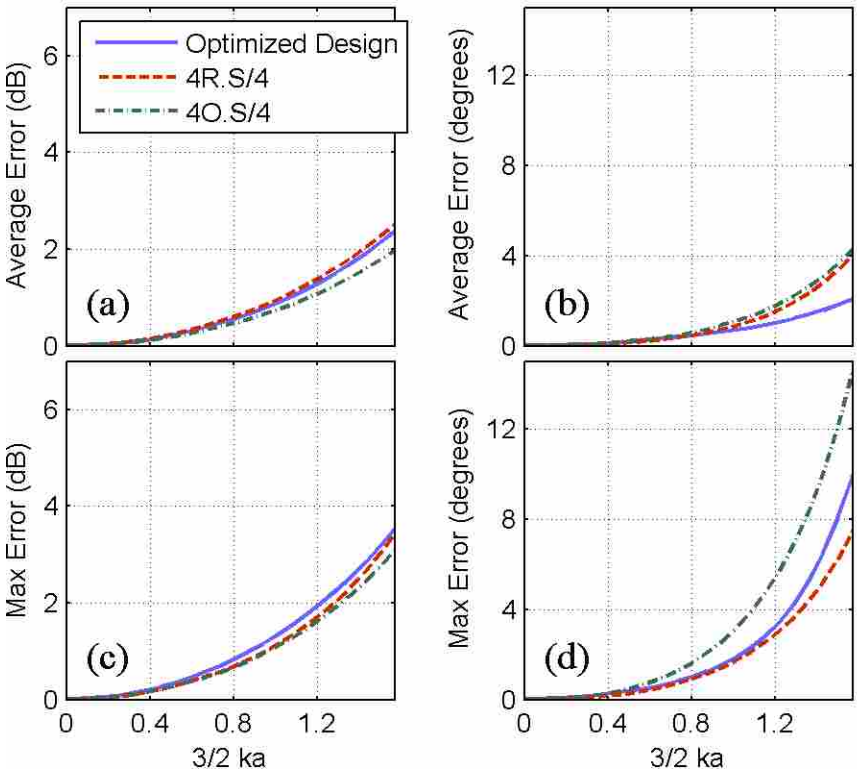

Figure 4-8: Intensity Magnitude Average (a) and Maximum (c) Errors and Intensity Direction Average (b) and Maximum (d) Errors in a Half Space Including the Errors of a Probe Optimized for Intensity Direction in a Half Space.

4.3.4 Spherical Probe Optimized for Intensity Magnitude Average Error at $ka = \pi/3$ in a Half Space

If a probe were instead to be optimized for intensity magnitude in a half space, the result has the sensors clumped closely together as shown in Figure 4-9. This is similar to what was seen in the full-space optimization for intensity magnitude at the same frequency, the lowest errors are obtained by having the microphones very close together—an infeasible design.

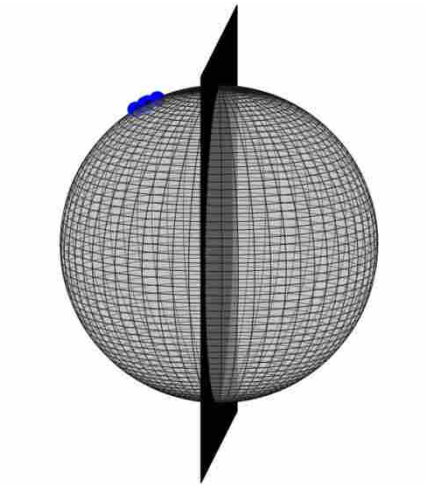

Figure 4-9: Optimized Design for Intensity Magnitude at $ka = \pi/3$ for a Half Space Where Errors Are Only Donsidered for the Right Half of the Sphere.

4.4 Conclusions

Optimization runs at $ka = \frac{\pi}{3}$ and $\frac{\pi}{48}$ showed that the regular tetrahedron design represents an optimal design for intensity direction in plane wave fields. This, however, does not take into account the various processing methods available with multimicrophone probes as was analyzed in Chapters 2 and 3. Future optimizations could have processing methods as design variables. Optimizing for intensity magnitude was shown to result in an infeasible design. Future work

could involve modeling microphone mismatches and then optimizing for intensity magnitude, as infeasible designs would then likely not result.

The results for the half-space optimization runs showed that low intensity direction errors can be obtained, but somewhat at the sacrifice of intensity magnitude results. But, as discussed in Chapters 2 and 3, intensity magnitude may be calibrated for, generally making the intensity direction errors more important. It was seen that if instead the probe is optimized for intensity magnitude, an infeasible design results. Future work could involve optimizing a design for both direction and magnitude errors in a half space. Or, instead of looking at a half space, optimization for intensity direction could be run in a larger or smaller space, likely resulting in designs that differ less or more, respectively, from the regular tetrahedron design.

5 CONCLUSIONS

5.1 Summary

The work of this thesis has been to examine the errors of multimicrophone probes in estimating acoustic intensity. It has been limited to plane wave fields with the probes consisting of matched point sensors. The maximum and average errors as a function of angle of incidence were reported for both intensity magnitude and direction. Error spread for intensity magnitude was also summarized. Probes with freely suspended point sensors have been analyzed as well as probes with sensors mounted in a hard sphere.

First, the errors in calculating intensity with the four-microphone orthogonal multimicrophone probe were examined. It was found that using a Taylor approximation for the velocity estimate led to less error in calculating intensity direction, but more error for intensity magnitude. It was seen that having the sensors mounted in a sphere led to less average errors for all processing methods. It also led to less maximum errors in most but not all processing methods.

This same trend was found in comparing the spherical and freely suspended regular tetrahedron and six-microphone probes; however, the errors were not significantly lower. Comparing probe designs to each other was found to be highly dependent on the definition of the dimensionless parameter *kd*. The lowest-error probe design and processing method combinations are summarized in Table 5-1, a repeat of Table 3-3.

Table 5-1: Ten Lowest-Error Probe Types for Each Quantity of Interest.

Rank	Magnitude (dB)	Magnitude Spread (dB)	Direction (degrees)
1	4O.S/1 (1.13)	6.S/15 1D (0.32)	6.S/15 1D (0.34)
2	4O/1 (1.49)	6.S/6 (0.57)	4O.S/3 1D.T (1.41)
3	4O.S/4W (1.57)	6/15 1D (0.75)	4O.S/3.T (1.42)
4	4O.S/4 (1.83)	6/6 (1.01)	4O.S/4.T (1.47)
5	4O.S/3 1D.T (2.01)	4O.S/4W.T (1.08)	4O.S/4W.T (1.52)
6	4O.S/4.T (2.02)	4R/4 (1.12)	4O.S/4 (1.54)
7	4O/4W (2.02)	4R.S/4 (1.21)	4R.S/4 (1.54)
8	4O.S/4W.T (2.04)	4O.S/3 1D.T (1.42)	4O.S/1.T (1.64)
9	6.S/15 1D (2.10)	4O/1 (1.89)	4O.S/4W (1.76)
10	4O.S/3 (2.11)	4O/4W.T (1.90)	4O.S/3 (1.78)

Using a genetic algorithm to optimize four-microphone spherical probes was shown to be able to predict designs that would minimize error. However, the optimized designs generally had unintended side effects such as having higher errors for quantities that were not being optimized for or predicting infeasible designs. But future work could include introducing phase and amplitude mismatches and having the probe optimized to a sum of the errors of the different quantities. The genetic algorithm has been shown to work for the cases tested and could continue to lead to better understanding of four-microphone probe design.

5.2 Recommendations

Future work recommended includes:

- Looking at energy density errors in addition to intensity errors.
- Including phase and sensitivity mismatch between sensors.
- Examining errors of multimicrophone probes in more complicated fields.
- Investigating feasibility of intensity direction calibration.
- Comparing analytical results to experimental results.
- Exploring new optimization scenarios.

REFERENCES

1. F. J. Fahy, "Measurement of acoustic intensity using the cross-spectral density of two microphone signals," J. Acoust. Soc. Am. **62**, 1057–1059 (1977).

2. G. Pavic, "Measurement of sound intensity," J. Sound Vib. **51**, 533-545 (1977).

3. R. McKinley, Workshop on military impulsive noise, Feb. 1998.

4. G. Rasmussen, "Measurement of vector fields," in *Proceedings of the Second International Congress on Acoustic Intensity*, Senlis, France (1985), pp. 53-58.

5. J. W. Parkins, S. D. Sommerfeldt, and J. Tichy, "Error analysis of a practical energy density sensor," J. Acoust. Soc. Am. **108**, 211-222 (2000).

6. L. M. C. Santos, C. C. Rodrigues, and J. L. Bento Coelho, "Measuring the three-dimensional acoustic intensity vector with a four-microphone probe," in *Proceedings of Inter-Noise*, Newport Beach, CA (1989), pp. 965-968.

7. R. Hickling and A. W. Brown, "Determining the direction to a sound source in air using vector sound-intensity probes," J. Acoust. Soc. Am. **129**, 219-224 (2011).

8. H. Suzuki, S. Oguro, M. Anzai, and T. Ono, "Performance evaluation of a three dimensional intensity probe," J. Acoust. Soc. Jpn. **16**, 233-238 (1995).

9. M. Schumacher, *A transducer and processing system to measure total acoustic energy density*, M.S. thesis, University of Texas at Austin, Austin, TX, 1984.

10. B. S. Cazzolato and C. H. Hansen, "Errors arising from three-dimensional energy density sensing in one-dimensional sound fields," J. Sound Vib. **236**, 375-400 (2000).

11. J.-C. Pascal and J.-F. Li, "A systematic method to obtain 3D finite-difference formulations for acoustic intensity and other energy quantities," J. Sound Vib. **310**, 1093-1111 (2008).

12. L. L. Locey, *Analysis and comparison of three acoustic energy density probes*, M.S. thesis, Brigham Young University, Provo, UT, 2004.

13. F. Jacobsen, V. Cutanda, and P. M. Juhl, "A numerical and experimental investigation of the performance of sound intensity probes at high frequencies," J Acoust Soc Am **103**, 953-961 (1997).

14. G. Moschioni, B. Saggin, and M. Tarabini, "3-D Sound Intensity Measurements: Accuracy Enhancements With Virtual-Instrument-Based Technology," IEEE Trans. Instrum. Meas. **57**, 1820-1829 (2008).

15. J. J. Bowman, T. B. A. Senior, P. L. E. Uslenghi, and J. S. Asvestas, *Electromagnetic and acoustic scattering by simple shapes* (North-Holland Pub. Co., Amsterdam, 1969),

16. P. M. Morse and K. U. Ingard, *Theoretical acoustics* (McGraw-Hill, New York, 1968), pp. 418-421.

17. G. W. Elko, "An acoustic vector-field probe with calculable obstacle bias," in *Proceedings of Noise-Con 91*, Tarrytown, NY (1991), pp. 525-532.

18. H. Suzuki, S. Oguro, and T. Ono, "A sensitivity correction method for a three-dimensional sound intensity probe," J. Acoust. Soc. Jpn. **21**, 259-265 (2000).

19. D. C. Thomas, *Theory and estimation of acoustic intensity and energy density*, M.S. thesis, Brigham Young University, Provo, UT, 2008.

20. J. Y. Chung, "Cross-spectral method of measuring acoustic intensity without error caused by instrument phase mismatch," J. Acoust. Soc. Am. **64**, 1613-1616 (1978).

21. B. S. Cazzolato and J. Ghan, "Frequency domain expressions for the estimation of time averaged acoustic energy density," J. Acoust. Soc. Am. **117**, 3750–3756 (2005).

22. B. S. Cazzolato and C. H. Hansen, "Errors in the measurement of acoustic energy density in one-dimensional sound fields," J. Sound Vib. **236**, 801-831 (2000).

23. J. Vandenhout, P. Sas, and R. Snoeys, "Measurement, accuracy and interpretation of real and imaginary intensity patterns in the near field of complex radiators," in *Proceedings of the Second International Congress on Acoustic Intensity*, Senlis, France (1985), pp. 121-128.

24. K. H. Miah and E. L. Hixon, "Design and performance evaluation of a broadband three dimensional acoustic intensity measuring system," J. Acoust. Soc. Am. **127**, 2338-2346 (2010).

25. J. A. Moryl and E. L. Hixson, "A total acoustic energy density sensor with applications to energy density measurement in a reverberation room," in *Proceedings of Inter-Noise*, Beijing, China (1987), pp. 1195-1198.

26. T. W. Leishman, S. Rollins, and H. M. Smith, "An experimental evaluation of regular polyhedron loudspeakers as omnidirectional sources of sound," J Acoust Soc Am **120**, 1411-1422 (2006).

27. B. B. Monson, S. D. Sommerfeldt, and K. L. Gee, "Improving compactness for active noise control of a small axial cooling fan," Noise Control Eng. J. **55**, 397-407 (2007).

28. J. R. Oldham, *Development of a multiple microphone probe calibrator*, M.S. thesis, Brigham Young University, Provo, UT, 2007.

29. F. J. Fahy, *Sound intensity*, 2nd ed. (E & FN Spon, London, 1995), pp. 91-97.

30. C. P. Wiederhold, "Analytical comparison of processing methods in measuring acoustic intensity with orthogonal multimicrophone probes," submitted for publication, J. Acoust. Soc. Am., (2011).

31. K. Hori, "4 microphones power advanced, 3-dimensional sound intensity measuring system," J. Electronic Eng. **31**, 47-49 (1994).

32. C. R. Duke, *Optimization of control source and error sensor locations in free field active noise control*, M.S. thesis, Brigham Young University, Provo, UT, 2007.

APPENDIX A. MATHEMATICAL EQUIVALENCE OF ORTHOGONAL PROBE TYPES "3 1D" AND "1"

The expression for the vector active intensity if the orthogonal probe is implemented as 3 one-dimensional probes ("3 1D") is given using complex notation by

$$\mathbf{I} = \begin{cases} I_x \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1 + p_2}{2} v_x^*\right\} \\ I_y \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1 + p_3}{2} v_y^*\right\} \\ I_z \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1 + p_4}{2} v_z^*\right\} \end{cases}, \tag{A-1}$$

where p_1 is the pressure from the origin microphone and p_2, p_3, and p_4 are the pressures from the microphones on the *x*-, *y*-, and *z*-axes; particle velocities in the three orthogonal directions are given by v_x, v_y, and v_z; the complex conjugate is denoted by "*;" and the real part is denoted by "Re." Using the time-harmonic linear Euler's equation

$$\mathbf{v} = \frac{j\nabla p}{\rho\omega}, \tag{A-2}$$

with ρ being the fluid density and ω the angular frequency in conjunction with a first-order finite-difference approximation of the pressure gradient, the following is obtained and then simplified:

$$\mathbf{I} = \begin{cases} I_x \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_2}{2}\cdot\left(\frac{j(p_2-p_1)}{2h\rho\omega}\right)^*\right\} \\ I_y \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_3}{2}\cdot\left(\frac{j(p_3-p_1)}{2h\rho\omega}\right)^*\right\} \\ I_z \approx \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_4}{2}\cdot\left(\frac{j(p_4-p_1)}{2h\rho\omega}\right)^*\right\} \end{cases}$$

$$= \begin{cases} \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_2}{2}\cdot\frac{-j({p_2}^*-{p_1}^*)}{2h\rho\omega}\right\} \\ \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_3}{2}\cdot\frac{-j({p_3}^*-{p_1}^*)}{2h\rho\omega}\right\} \\ \frac{1}{2}\mathrm{Re}\left\{\frac{p_1+p_4}{2}\cdot\frac{-j({p_4}^*-{p_1}^*)}{2h\rho\omega}\right\} \end{cases}$$

$$= \begin{cases} \frac{1}{8h\rho\omega}\mathrm{Im}\{(p_1+p_2)({p_2}^*-{p_1}^*)\} \\ \frac{1}{8h\rho\omega}\mathrm{Im}\{(p_1+p_3)({p_3}^*-{p_1}^*)\} \\ \frac{1}{8h\rho\omega}\mathrm{Im}\{(p_1+p_4)({p_4}^*-{p_1}^*)\} \end{cases}$$

$$= \begin{cases} \frac{1}{8h\rho\omega}\mathrm{Im}\{p_1{p_2}^*-p_1{p_1}^*+p_2{p_2}^*-p_2{p_1}^*\} \\ \frac{1}{8h\rho\omega}\mathrm{Im}\{p_1{p_3}^*-p_1{p_1}^*+p_3{p_3}^*-p_3{p_1}^*\} \\ \frac{1}{8h\rho\omega}\mathrm{Im}\{p_1{p_4}^*-p_1{p_1}^*+p_4{p_4}^*-p_4{p_1}^*\} \end{cases}$$

$$= \begin{cases} \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_2}^*\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_3}^*\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_4}^*\} \end{cases}, \qquad \text{(A-3)}$$

where $2h$ is the separation distance from microphone 1 to the other three microphones and j the imaginary unit.

If instead the pressure estimate of the orthogonal probe is taken to be the pressure of the origin microphone ("1") the intensity is:

$$\mathbf{I} = \begin{cases} I_x \approx \frac{1}{2}\mathrm{Re}\{p_1 v_x^*\} \\ I_y \approx \frac{1}{2}\mathrm{Re}\{p_1 v_y^*\} \\ I_z \approx \frac{1}{2}\mathrm{Re}\{p_1 v_z^*\} \end{cases}$$

$$= \begin{cases} \frac{1}{2}\mathrm{Re}\left\{p_1\left(\frac{j(p_2 - p_1)}{2h\rho\omega}\right)^*\right\} \\ \frac{1}{2}\mathrm{Re}\left\{p_1\left(\frac{j(p_3 - p_1)}{2h\rho\omega}\right)^*\right\} \\ \frac{1}{2}\mathrm{Re}\left\{p_1\left(\frac{j(p_4 - p_1)}{2h\rho\omega}\right)^*\right\} \end{cases}$$

$$= \begin{cases} \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1({p_2}^* - {p_1}^*)\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1({p_3}^* - {p_1}^*)\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1({p_4}^* - {p_1}^*)\} \end{cases}$$

$$= \begin{cases} \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_2}^*\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_3}^*\} \\ \frac{1}{4h\rho\omega}\mathrm{Im}\{p_1{p_4}^*\} \end{cases} . \quad \text{(A-4)}$$

As Eqs. (A-3) and (A-4) are equivalent, the two processing methods are equivalent in calculating intensity.

APPENDIX B. MATLAB CODE

B.1 Multimicrophone_Probe_Analysis.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Multimicrophone_Probe_Analysis.m                                      %
% Date: 08 August 2011                                                        %
% Author: Curtis Wiederhold (based on code by Derek Thomas)                   %
%                                                                             %
% Description: This Matlab script performs a freq domain analytical           %
% comparison of three types of multimicrophone probes: the four mic           %
% orthogonal probe, the four mic regular tetrahedron probe, and the           %
% six mic probe. It compares the processing methods that have been            %
% suggested to use with each of these probes. Incoming plane waves are        %
% simulated at different incidence angles. For spherical-type probes,         %
% spherical scattering is calculated.                                         %
%                                                                             %
% Functions called by script:                                                 %
% Taylor_Velocity.m                                                           %
% Intensity_Calc.m                                                            %
% Angle_Calc.m                                                                %
% Bias_Calc.m                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all;

tic

% Set step size for angles of incidence
res = 50;
res_step = pi/res;

% Specify frequencies
freq1 = 10;
freq2 = 100;
freq3 = 200:200:8000;
freq = horzcat(freq1,freq2,freq3);

% For loop to calculate values at each frequency.
q = 1;
for f=(freq(1:length(freq)))
    disp(int2str(f));
```

```
w = 2*pi*f;
c = 343;
rho = 1.21;
k = w/c;
M = 25; % Number of terms to include in the scattering expansion.
a = 0.0254/2; % Radius of the probe.
% For loop for theta angle of incoming plane wave.
o = 1;
for thetak=0:res_step:pi
    % For loop for phi angle of incoming plane wave.
    % Only goes to 2*pi-res_step so that in average error calculation
    % some angles of incidence aren't included twice.
    n=1;
    for phik = 0:res_step:2*pi-res_step

        % Wave vector components of the plane wave.
        kx = -k*cos(phik)*sin(thetak);
        ky = -k*sin(phik)*sin(thetak);
        kz = -k*cos(thetak);

        p0 = 1; % Amplitude of the incident wave.

        %%%%%%% Microphone positions of probes given in spherical
        %%%%%%% coordinates and dot products of these positions with
        %%%%%%% angle of incidence of incoming plane wave.
        %%%%%%% First Subscript:
        %%%%%%% 1 = Orthogonal probe
        %%%%%%% 2 = Regular tetrahedron probe
        %%%%%%% 3 = 6-microphone probe

        % Orthogonal probe
        mictheta1(1) = acos(-1/sqrt(3));
        micphi1(1) = atan2(-1/sqrt(3),1/sqrt(3));
        rk11 = -(-1/2*(cos(mictheta1(1)-thetak)*(1+cos(micphi1(1)...
            -phik))+cos(mictheta1(1)+thetak)*(1-cos(micphi1(1)-phik))));
        mictheta1(2) = acos(-1/sqrt(3));
        micphi1(2) = atan2(1/sqrt(3),-1/sqrt(3));
        rk12 = -(-1/2*(cos(mictheta1(2)-thetak)*(1+cos(micphi1(2)...
            -phik))+cos(mictheta1(2)+thetak)*(1-cos(micphi1(2)-phik))));
        mictheta1(3) = acos(-1/sqrt(3));
        micphi1(3) = atan2(-1/sqrt(3),-1/sqrt(3));
        rk13 = -(-1/2*(cos(mictheta1(3)-thetak)*(1+cos(micphi1(3)...
            -phik))+cos(mictheta1(3)+thetak)*(1-cos(micphi1(3)-phik))));
        mictheta1(4) = acos(1/sqrt(3));
        micphi1(4) = atan2(-1/sqrt(3),-1/sqrt(3));
        rk14 = -(-1/2*(cos(mictheta1(4)-thetak)*(1+cos(micphi1(4)...
            -phik))+cos(mictheta1(4)+thetak)*(1-cos(micphi1(4)-phik))));

        % Regular tetrahedron probe
        mictheta2(1) = 0;
        micphi2(1) = 0;
        rk21 = -(-1/2*(cos(mictheta2(1)-thetak)*(1+cos(micphi2(1)...
            -phik))+cos(mictheta2(1)+thetak)*(1-cos(micphi2(1)-phik))));
        mictheta2(2) = 1.910633237;
        micphi2(2) = -pi/3;
```

```
rk22 = -(-1/2*(cos(mictheta2(2)-thetak)*(1+cos(micphi2(2)...
    -phik))+cos(mictheta2(2)+thetak)*(1-cos(micphi2(2)-phik))));
mictheta2(3) = 1.910633237;
micphi2(3) = pi;
rk23 = -(-1/2*(cos(mictheta2(3)-thetak)*(1+cos(micphi2(3)...
    -phik))+cos(mictheta2(3)+thetak)*(1-cos(micphi2(3)-phik))));
mictheta2(4) = 1.910633237;
micphi2(4) = pi/3;
rk24 = -(-1/2*(cos(mictheta2(4)-thetak)*(1+cos(micphi2(4)...
    -phik))+cos(mictheta2(4)+thetak)*(1-cos(micphi2(4)-phik))));

% 6-microphone probe
mictheta3(1) = 0;
micphi3(1) = 0;
rk31 = -(-1/2*(cos(mictheta3(1)-thetak)*(1+cos(micphi3(1)...
    -phik))+cos(mictheta3(1)+thetak)*(1-cos(micphi3(1)-phik))));
mictheta3(2) = pi/2;
micphi3(2) = 0;
rk32 = -(-1/2*(cos(mictheta3(2)-thetak)*(1+cos(micphi3(2)...
    -phik))+cos(mictheta3(2)+thetak)*(1-cos(micphi3(2)-phik))));
mictheta3(3) = pi/2;
micphi3(3) = pi/2;
rk33 = -(-1/2*(cos(mictheta3(3)-thetak)*(1+cos(micphi3(3)...
    -phik))+cos(mictheta3(3)+thetak)*(1-cos(micphi3(3)-phik))));
mictheta3(4) = pi/2;
micphi3(4) = pi;
rk34 = -(-1/2*(cos(mictheta3(4)-thetak)*(1+cos(micphi3(4)...
    -phik))+cos(mictheta3(4)+thetak)*(1-cos(micphi3(4)-phik))));
mictheta3(5) = pi/2;
micphi3(5) = 3*pi/2;
rk35 = -(-1/2*(cos(mictheta3(5)-thetak)*(1+cos(micphi3(5)...
    -phik))+cos(mictheta3(5)+thetak)*(1-cos(micphi3(5)-phik))));
mictheta3(6) = pi;
micphi3(6) = 0;
rk36 = -(-1/2*(cos(mictheta3(6)-thetak)*(1+cos(micphi3(6)...
    -phik))+cos(mictheta3(6)+thetak)*(1-cos(micphi3(6)-phik))));

% Calculate the incident pressures at microphones 1 to 4
% (pi1-pi4) and the first terms of the scattered pressures
% (ps1-ps4) for the spherical probe types.
pi11 = p0*exp(1i*k*a*rk11);
pi12 = p0*exp(1i*k*a*rk12);
pi13 = p0*exp(1i*k*a*rk13);
pi14 = p0*exp(1i*k*a*rk14);

pi21 = p0*exp(1i*k*a*rk21);
pi22 = p0*exp(1i*k*a*rk22);
pi23 = p0*exp(1i*k*a*rk23);
pi24 = p0*exp(1i*k*a*rk24);

pi31 = p0*exp(1i*k*a*rk31);
pi32 = p0*exp(1i*k*a*rk32);
pi33 = p0*exp(1i*k*a*rk33);
pi34 = p0*exp(1i*k*a*rk34);
pi35 = p0*exp(1i*k*a*rk35);
```

```
pi36 = p0*exp(1i*k*a*rk36);

% Initialize the first two Legendre polynomials for
% the recurrence relations to calculate scattered pressure
% "mm2" means "order m minus 2", "mp1" means "order m plus 1"
P11mm2 = 1;
P12mm2 = 1;
P13mm2 = 1;
P14mm2 = 1;

P21mm2 = 1;
P22mm2 = 1;
P23mm2 = 1;
P24mm2 = 1;

P31mm2 = 1;
P32mm2 = 1;
P33mm2 = 1;
P34mm2 = 1;
P35mm2 = 1;
P36mm2 = 1;

P11mm1 = rk11;
P12mm1 = rk12;
P13mm1 = rk13;
P14mm1 = rk14;

P21mm1 = rk21;
P22mm1 = rk22;
P23mm1 = rk23;
P24mm1 = rk24;

P31mm1 = rk31;
P32mm1 = rk32;
P33mm1 = rk33;
P34mm1 = rk34;
P35mm1 = rk35;
P36mm1 = rk36;


m = 0;  % initial order for Bessel functions
% Calculate the necessary Bessel and Hankel functions for
% scattering calculation.
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));

Am = (-1i)^m*(2*m+1)/dhm;
p11 = Am*P11mm2;
p12 = Am*P12mm2;
p13 = Am*P13mm2;
p14 = Am*P14mm2;
```

```
p21 = Am*P21mm2;
p22 = Am*P22mm2;
p23 = Am*P23mm2;
p24 = Am*P24mm2;

p31 = Am*P31mm2;
p32 = Am*P32mm2;
p33 = Am*P33mm2;
p34 = Am*P34mm2;
p35 = Am*P35mm2;
p36 = Am*P36mm2;

% Change order for Bessel functions, then proceed same as
% above.
m = 1;
% Calculate the necessary Bessel and Hankel functions.
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));
Am = (-1i)^m*(2*m+1)/dhm;

% Now add results from order m=0 results to those of m=1.
p11 = p11+Am*P11mm1;
p12 = p12+Am*P12mm1;
p13 = p13+Am*P13mm1;
p14 = p14+Am*P14mm1;

p21 = p21+Am*P21mm1;
p22 = p22+Am*P22mm1;
p23 = p23+Am*P23mm1;
p24 = p24+Am*P24mm1;

p31 = p31+Am*P31mm1;
p32 = p32+Am*P32mm1;
p33 = p33+Am*P33mm1;
p34 = p34+Am*P34mm1;
p35 = p35+Am*P35mm1;
p36 = p36+Am*P36mm1;

% Now that you have 2 calculated, recurrence relations can be
% used to calculate the rest.
for m = 2:M;
    % Calculate the necessary Legendre polynomials.
    P11m = 2*rk11*P11mm1-P11mm2-1/m*(rk11*P11mm1-P11mm2);
    P12m = 2*rk12*P12mm1-P12mm2-1/m*(rk12*P12mm1-P12mm2);
    P13m = 2*rk13*P13mm1-P13mm2-1/m*(rk13*P13mm1-P13mm2);
    P14m = 2*rk14*P14mm1-P14mm2-1/m*(rk14*P14mm1-P14mm2);

    P21m = 2*rk21*P21mm1-P21mm2-1/m*(rk21*P21mm1-P21mm2);
    P22m = 2*rk22*P22mm1-P22mm2-1/m*(rk22*P22mm1-P22mm2);
    P23m = 2*rk23*P23mm1-P23mm2-1/m*(rk23*P23mm1-P23mm2);
    P24m = 2*rk24*P24mm1-P24mm2-1/m*(rk24*P24mm1-P24mm2);
```

```
P31m = 2*rk31*P31mm1-P31mm2-1/m*(rk31*P31mm1-P31mm2);
P32m = 2*rk32*P32mm1-P32mm2-1/m*(rk32*P32mm1-P32mm2);
P33m = 2*rk33*P33mm1-P33mm2-1/m*(rk33*P33mm1-P33mm2);
P34m = 2*rk34*P34mm1-P34mm2-1/m*(rk34*P34mm1-P34mm2);
P35m = 2*rk35*P35mm1-P35mm2-1/m*(rk35*P35mm1-P35mm2);
P36m = 2*rk36*P36mm1-P36mm2-1/m*(rk36*P36mm1-P36mm2);

% Calculate the necessary Bessel and Hankel functions.
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));

Am = (-1i)^m*(2*m+1)/dhm;

p11 = p11+Am*P11m;
p12 = p12+Am*P12m;
p13 = p13+Am*P13m;
p14 = p14+Am*P14m;

p21 = p21+Am*P21m;
p22 = p22+Am*P22m;
p23 = p23+Am*P23m;
p24 = p24+Am*P24m;

p31 = p31+Am*P31m;
p32 = p32+Am*P32m;
p33 = p33+Am*P33m;
p34 = p34+Am*P34m;
p35 = p35+Am*P35m;
p36 = p36+Am*P36m;

% Update the Legendre polynomials for the recurrence
% relation.
P11mm2 = P11mm1;
P12mm2 = P12mm1;
P13mm2 = P13mm1;
P14mm2 = P14mm1;

P21mm2 = P21mm1;
P22mm2 = P22mm1;
P23mm2 = P23mm1;
P24mm2 = P24mm1;

P31mm2 = P31mm1;
P32mm2 = P32mm1;
P33mm2 = P33mm1;
P34mm2 = P34mm1;
P35mm2 = P35mm1;
P36mm2 = P36mm1;

P11mm1 = P11m;
```

```
    P12mm1 = P12m;
    P13mm1 = P13m;
    P14mm1 = P14m;

    P21mm1 = P21m;
    P22mm1 = P22m;
    P23mm1 = P23m;
    P24mm1 = P24m;

    P31mm1 = P31m;
    P32mm1 = P32m;
    P33mm1 = P33m;
    P34mm1 = P34m;
    P35mm1 = P35m;
    P36mm1 = P36m;
end

p11 = conj(p11*1i*p0/(k*a)^2);
p12 = conj(p12*1i*p0/(k*a)^2);
p13 = conj(p13*1i*p0/(k*a)^2);
p14 = conj(p14*1i*p0/(k*a)^2);

p21 = conj(p21*1i*p0/(k*a)^2);
p22 = conj(p22*1i*p0/(k*a)^2);
p23 = conj(p23*1i*p0/(k*a)^2);
p24 = conj(p24*1i*p0/(k*a)^2);

p31 = conj(p31*1i*p0/(k*a)^2);
p32 = conj(p32*1i*p0/(k*a)^2);
p33 = conj(p33*1i*p0/(k*a)^2);
p34 = conj(p34*1i*p0/(k*a)^2);
p35 = conj(p35*1i*p0/(k*a)^2);
p36 = conj(p36*1i*p0/(k*a)^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%        EXACT CALCULATIONS      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Particle velocity
vx = kx/rho/w*p0;
vy = ky/rho/w*p0;
vz = kz/rho/w*p0;
v(o,n) = (abs(vx).^2+abs(vy).^2+abs(vz).^2).^(1/2);

% Intensity
Ix = 0.5*real(p0*conj(vx));
Iy = 0.5*real(p0*conj(vy));
Iz = 0.5*real(p0*conj(vz));
I(o,n) = (abs(Ix).^2+abs(Iy).^2+abs(Iz).^2).^(1/2);

% Energy density
W(o,n) = rho*abs(v(o,n))^2/4 + abs(p0)^2/4/rho/c^2;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%        FINITE-DIFF CALCULATIONS    %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Pressure Calculations
%%%%%%% 1st subscript: probe type
%%%%%%% 2nd: pressure estimation type

pi_avg12 = pi13;
p_avg12 = p13;
pi_avg13 = (pi11+pi12+pi13+pi14)/4;
p_avg13 = (p11+p12+p13+p14)/4;
pi_avg14 = ((pi11+pi13)/2+(pi12+pi13)/2+(pi14+pi13)/2)/3;
p_avg14 = ((p11+p13)/2+(p12+p13)/2+(p14+p13)/2)/3;
pi_avg15 = (pi11+pi12+pi14)/3;
p_avg15 = (p11+p12+p14)/3;

pi_avg21 = (pi21+pi22+pi23+pi24)/4;
p_avg21 = (p21+p22+p23+p24)/4;
pi_avg22 = pi21;
p_avg22 = p21;

pi_avg32 = (pi31+pi32+pi33+pi34+pi35+pi36)/6;
p_avg32 = (p31+p32+p33+p34+p35+p36)/6;
pi_avg33 = pi31;
p_avg33 = p31;

Pi12(o,n) = abs(pi_avg12);
P12(o,n) = abs(p_avg12);
Pi13(o,n) = abs(pi_avg13);
P13(o,n) = abs(p_avg13);
Pi14(o,n) = abs(pi_avg14);
P14(o,n) = abs(p_avg14);
Pi15(o,n) = abs(pi_avg15);
P15(o,n) = abs(p_avg15);

Pi21(o,n) = abs(pi_avg21);
P21(o,n) = abs(p_avg21);
Pi22(o,n) = abs(pi_avg22);
P22(o,n) = abs(p_avg22);

Pi32(o,n) = abs(pi_avg32);
P32(o,n) = abs(p_avg32);
Pi33(o,n) = abs(pi_avg33);
P33(o,n) = abs(p_avg33);


%%%%%%% Velocity Calculations
%%%%%%% 1st subscript: probe type
%%%%%%% 2nd: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%          sphere w/o 3/2 factor
```

```
%%%%%%% 3rd: 1 = estimation at three points, 2 = estimation
%%%%%%%          at one point using Taylor expansion

vx111=1i/rho/(2*a/sqrt(3))/w*(pi11-pi13);
vy111=1i/rho/(2*a/sqrt(3))/w*(pi12-pi13);
vz111=1i/rho/(2*a/sqrt(3))/w*(pi14-pi13);
v111(o,n) = (abs(vx111).^2+abs(vy111).^2+abs(vz111).^2).^(1/2);
vx121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p11-p13);
vy121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p12-p13);
vz121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p14-p13);
v121(o,n) = (abs(vx121).^2+abs(vy121).^2+abs(vz121).^2).^(1/2);
vx131=1i/rho/(2*a/sqrt(3))/w*(p11-p13);
vy131=1i/rho/(2*a/sqrt(3))/w*(p12-p13);
vz131=1i/rho/(2*a/sqrt(3))/w*(p14-p13);
v131(o,n) = (abs(vx131).^2+abs(vy131).^2+abs(vz131).^2).^(1/2);

[vx112 vy112 vz112 v112(o,n)] = ...
    Taylor_Velocity(pi11,pi12,pi13,pi14,w,rho,a,1);
[vx122 vy122 vz122 v122(o,n)] = ...
    Taylor_Velocity(p11,p12,p13,p14,w,rho,a,3/2);
[vx132 vy132 vz132 v132(o,n)] = ...
    Taylor_Velocity(p11,p12,p13,p14,w,rho,a,1);

vx211=1i/rho/4/a/w*sqrt(2)*(pi22-2*pi23+pi24);
vy211=1i/rho/4/a/w*sqrt(6)*(-pi22+pi24);
vz211=1i/rho/4/a/w*(3*pi21-pi22-pi23-pi24);
v211(o,n) = (abs(vx211).^2+abs(vy211).^2+abs(vz211).^2).^(1/2);
vx221=1i/rho/4/(a*3/2)/w*sqrt(2)*(p22-2*p23+p24);
vy221=1i/rho/4/(a*3/2)/w*sqrt(6)*(-p22+p24);
vz221=1i/rho/4/(a*3/2)/w*(3*p21-p22-p23-p24);
v221(o,n) = (abs(vx221).^2+abs(vy221).^2+abs(vz221).^2).^(1/2);
vx231=1i/rho/4/a/w*sqrt(2)*(p22-2*p23+p24);
vy231=1i/rho/4/a/w*sqrt(6)*(-p22+p24);
vz231=1i/rho/4/a/w*(3*p21-p22-p23-p24);
v231(o,n) = (abs(vx231).^2+abs(vy231).^2+abs(vz231).^2).^(1/2);

vx311=1i/rho/(2*a)/w*(pi32-pi34);
vy311=1i/rho/(2*a)/w*(pi33-pi35);
vz311=1i/rho/(2*a)/w*(pi31-pi36);
v311(o,n) = (abs(vx311).^2+abs(vy311).^2+abs(vz311).^2).^(1/2);
vx321=1i/rho/(2*a*3/2)/w*(p32-p34);
vy321=1i/rho/(2*a*3/2)/w*(p33-p35);
vz321=1i/rho/(2*a*3/2)/w*(p31-p36);
v321(o,n) = (abs(vz321).^2+abs(vy321).^2+abs(vx321).^2).^(1/2);
vx331=1i/rho/(2*a)/w*(p32-p34);
vy331=1i/rho/(2*a)/w*(p33-p35);
vz331=1i/rho/(2*a)/w*(p31-p36);
v331(o,n) = (abs(vx331).^2+abs(vy331).^2+abs(vz331).^2).^(1/2);

% Calculations for six-microphone probe described in Moschioni
% et al. First two subscripts are the two microphones betweeen
% which the intensity is being calculated.
v42_311=1i/rho/(2*a)/w*(pi32-pi34);
v42_321=1i/rho/(2*a*3/2)/w*(p32-p34);
v42_331=1i/rho/(2*a)/w*(p32-p34);
```

```
v45_311=1i/rho/(2*a/sqrt(2))/w*(pi35-pi34);
v45_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p35-p34);
v45_331=1i/rho/(2*a/sqrt(2))/w*(p35-p34);
v43_311=1i/rho/(2*a/sqrt(2))/w*(pi33-pi34);
v43_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p33-p34);
v43_331=1i/rho/(2*a/sqrt(2))/w*(p33-p34);
v46_311=1i/rho/(2*a/sqrt(2))/w*(pi36-pi34);
v46_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p36-p34);
v46_331=1i/rho/(2*a/sqrt(2))/w*(p36-p34);
v41_311=1i/rho/(2*a/sqrt(2))/w*(pi31-pi34);
v41_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p31-p34);
v41_331=1i/rho/(2*a/sqrt(2))/w*(p31-p34);
v52_311=1i/rho/(2*a/sqrt(2))/w*(pi32-pi35);
v52_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p32-p35);
v52_331=1i/rho/(2*a/sqrt(2))/w*(p32-p35);
v32_311=1i/rho/(2*a/sqrt(2))/w*(pi32-pi33);
v32_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p32-p33);
v32_331=1i/rho/(2*a/sqrt(2))/w*(p32-p33);
v62_311=1i/rho/(2*a/sqrt(2))/w*(pi32-pi36);
v62_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p32-p36);
v62_331=1i/rho/(2*a/sqrt(2))/w*(p32-p36);
v12_311=1i/rho/(2*a/sqrt(2))/w*(pi32-pi31);
v12_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p32-p31);
v12_331=1i/rho/(2*a/sqrt(2))/w*(p32-p31);
v53_311=1i/rho/(2*a)/w*(pi33-pi35);
v53_321=1i/rho/(2*a*3/2)/w*(p33-p35);
v53_331=1i/rho/(2*a)/w*(p33-p35);
v51_311=1i/rho/(2*a/sqrt(2))/w*(pi31-pi35);
v51_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p31-p35);
v51_331=1i/rho/(2*a/sqrt(2))/w*(p31-p35);
v54_311=-v45_311;
v54_321=-v45_321;
v54_331=-v45_331;
v56_311=1i/rho/(2*a/sqrt(2))/w*(pi36-pi35);
v56_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p36-p35);
v56_331=1i/rho/(2*a/sqrt(2))/w*(p36-p35);
v13_311=1i/rho/(2*a/sqrt(2))/w*(pi33-pi31);
v13_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p33-p31);
v13_331=1i/rho/(2*a/sqrt(2))/w*(p33-p31);
v23_311=-v32_311;
v23_321=-v32_321;
v23_331=-v32_331;
v63_311=1i/rho/(2*a/sqrt(2))/w*(pi33-pi36);
v63_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p33-p36);
v63_331=1i/rho/(2*a/sqrt(2))/w*(p33-p36);
v61_311=1i/rho/(2*a)/w*(pi31-pi36);
v61_321=1i/rho/(2*a*3/2)/w*(p31-p36);
v61_331=1i/rho/(2*a)/w*(p31-p36);
v64_311=-v46_311;
v64_321=-v46_321;
v64_331=-v46_331;
v65_311=1i/rho/(2*a/sqrt(2))/w*(pi35-pi36);
v65_321=1i/rho/(2*a*3/2/sqrt(2))/w*(p35-p36);
v65_331=1i/rho/(2*a/sqrt(2))/w*(p35-p36);
v21_311=-v12_311;
v21_321=-v12_321;
```

```
v21_331=-v12_331;
v31_311=-v13_311;
v31_321=-v13_321;
v31_331=-v13_331;


%%%%%%% Intensity Calculations
%%%%%%% 1st subscript: type of probe
%%%%%%% 2nd: how p was calculated
%%%%%%% 3rd: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%      sphere w/o 3/2 factor
%%%%%%% 4th: 1 = three points, 2 = Tayloy approximation

Ix1111 = 0.5*real((pi11+pi13)/2*conj(vx111));
Iy1111 = 0.5*real((pi12+pi13)/2*conj(vy111));
Iz1111 = 0.5*real((pi14+pi13)/2*conj(vz111));
I1111(o,n) = ...
    (abs(Ix1111).^2+abs(Iy1111).^2+abs(Iz1111).^2).^(1/2);
Ix1121 = 0.5*real((p11+p13)/2*conj(vx121));
Iy1121 = 0.5*real((p12+p13)/2*conj(vy121));
Iz1121 = 0.5*real((p14+p13)/2*conj(vz121));
I1121(o,n) = ...
    (abs(Ix1121).^2+abs(Iy1121).^2+abs(Iz1121).^2).^(1/2);
Ix1131 = 0.5*real((p11+p13)/2*conj(vx131));
Iy1131 = 0.5*real((p12+p13)/2*conj(vy131));
Iz1131 = 0.5*real((p14+p13)/2*conj(vz131));
I1131(o,n) = ...
    (abs(Ix1131).^2+abs(Iy1131).^2+abs(Iz1131).^2).^(1/2);
[Ix1211 Iy1211 Iz1211 I1211(o,n)] = ...
    Intensity_Calc(pi_avg12,vx111,vy111,vz111);
[Ix1221 Iy1221 Iz1221 I1221(o,n)] = ...
    Intensity_Calc(p_avg12,vx121,vy121,vz121);
[Ix1231 Iy1231 Iz1231 I1231(o,n)] = ...
    Intensity_Calc(p_avg12,vx131,vy131,vz131);
[Ix1311 Iy1311 Iz1311 I1311(o,n)] = ...
    Intensity_Calc(pi_avg13,vx111,vy111,vz111);
[Ix1321 Iy1321 Iz1321 I1321(o,n)] = ...
    Intensity_Calc(p_avg13,vx121,vy121,vz121);
[Ix1331 Iy1331 Iz1331 I1331(o,n)] = ...
    Intensity_Calc(p_avg13,vx131,vy131,vz131);
[Ix1411 Iy1411 Iz1411 I1411(o,n)] = ...
    Intensity_Calc(pi_avg14,vx111,vy111,vz111);
[Ix1421 Iy1421 Iz1421 I1421(o,n)] = ...
    Intensity_Calc(p_avg14,vx121,vy121,vz121);
[Ix1431 Iy1431 Iz1431 I1431(o,n)] = ...
    Intensity_Calc(p_avg14,vx131,vy131,vz131);
[Ix1511 Iy1511 Iz1511 I1511(o,n)] = ...
    Intensity_Calc(pi_avg15,vx111,vy111,vz111);
[Ix1521 Iy1521 Iz1521 I1521(o,n)] = ...
    Intensity_Calc(p_avg15,vx121,vy121,vz121);
[Ix1531 Iy1531 Iz1531 I1531(o,n)] = ...
    Intensity_Calc(p_avg15,vx131,vy131,vz131);

Ix1112 = 0.5*real((pi11+pi13)/2*conj(vx112));
Iy1112 = 0.5*real((pi12+pi13)/2*conj(vy112));
```

```
Iz1112 = 0.5*real((pi14+pi13)/2*conj(vz112));
I1112(o,n) = ...
    (abs(Ix1112).^2+abs(Iy1112).^2+abs(Iz1112).^2).^(1/2);
Ix1122 = 0.5*real((p11+p13)/2*conj(vx122));
Iy1122 = 0.5*real((p12+p13)/2*conj(vy122));
Iz1122 = 0.5*real((p14+p13)/2*conj(vz122));
I1122(o,n) = ...
    (abs(Ix1122).^2+abs(Iy1122).^2+abs(Iz1122).^2).^(1/2);
Ix1132 = 0.5*real((p11+p13)/2*conj(vx132));
Iy1132 = 0.5*real((p12+p13)/2*conj(vy132));
Iz1132 = 0.5*real((p14+p13)/2*conj(vz132));
I1132(o,n) = ...
    (abs(Ix1132).^2+abs(Iy1132).^2+abs(Iz1132).^2).^(1/2);
[Ix1212 Iy1212 Iz1212 I1212(o,n)] = ...
    Intensity_Calc(pi_avg12,vx112,vy112,vz112);
[Ix1222 Iy1222 Iz1222 I1222(o,n)] = ...
    Intensity_Calc(p_avg12,vx122,vy122,vz122);
[Ix1232 Iy1232 Iz1232 I1232(o,n)] = ...
    Intensity_Calc(p_avg12,vx132,vy132,vz132);
[Ix1312 Iy1312 Iz1312 I1312(o,n)] = ...
    Intensity_Calc(pi_avg13,vx112,vy112,vz112);
[Ix1322 Iy1322 Iz1322 I1322(o,n)] = ...
    Intensity_Calc(p_avg13,vx122,vy122,vz122);
[Ix1332 Iy1332 Iz1332 I1332(o,n)] = ...
    Intensity_Calc(p_avg13,vx132,vy132,vz132);
[Ix1412 Iy1412 Iz1412 I1412(o,n)] = ...
    Intensity_Calc(pi_avg14,vx112,vy112,vz112);
[Ix1422 Iy1422 Iz1422 I1422(o,n)] = ...
    Intensity_Calc(p_avg14,vx122,vy122,vz122);
[Ix1432 Iy1432 Iz1432 I1432(o,n)] = ...
    Intensity_Calc(p_avg14,vx132,vy132,vz132);
[Ix1512 Iy1512 Iz1512 I1512(o,n)] = ...
    Intensity_Calc(pi_avg15,vx112,vy112,vz112);
[Ix1522 Iy1522 Iz1522 I1522(o,n)] = ...
    Intensity_Calc(p_avg15,vx122,vy122,vz122);
[Ix1532 Iy1532 Iz1532 I1532(o,n)] = ...
    Intensity_Calc(p_avg15,vx132,vy132,vz132);

[Ix2111 Iy2111 Iz2111 I2111(o,n)] = ...
    Intensity_Calc(pi_avg21,vx211,vy211,vz211);
[Ix2121 Iy2121 Iz2121 I2121(o,n)] = ...
    Intensity_Calc(p_avg21,vx221,vy221,vz221);
[Ix2131 Iy2131 Iz2131 I2131(o,n)] = ...
    Intensity_Calc(p_avg21,vx231,vy231,vz231);
[Ix2211 Iy2211 Iz2211 I2211(o,n)] = ...
    Intensity_Calc(pi_avg22,vx211,vy211,vz211);
[Ix2221 Iy2221 Iz2221 I2221(o,n)] = ...
    Intensity_Calc(p_avg22,vx221,vy221,vz221);
[Ix2231 Iy2231 Iz2231 I2231(o,n)] = ...
    Intensity_Calc(p_avg22,vx231,vy231,vz231);

Ix3111 = 0.5*real((pi32+pi34)/2*conj(vx311));
Iy3111 = 0.5*real((pi33+pi35)/2*conj(vy311));
Iz3111 = 0.5*real((pi31+pi36)/2*conj(vz311));
I3111(o,n) = ...
    (abs(Ix3111).^2+abs(Iy3111).^2+abs(Iz3111).^2).^(1/2);
```

```
Ix3121 = 0.5*real((p32+p34)/2*conj(vx321));
Iy3121 = 0.5*real((p33+p35)/2*conj(vy321));
Iz3121 = 0.5*real((p31+p36)/2*conj(vz321));
I3121(o,n) = ...
    (abs(Ix3121).^2+abs(Iy3121).^2+abs(Iz3121).^2).^(1/2);
Ix3131 = 0.5*real((p32+p34)/2*conj(vx331));
Iy3131 = 0.5*real((p33+p35)/2*conj(vy331));
Iz3131 = 0.5*real((p31+p36)/2*conj(vz331));
I3131(o,n) = ...
    (abs(Ix3131).^2+abs(Iy3131).^2+abs(Iz3131).^2).^(1/2);
[Ix3211 Iy3211 Iz3211 I3211(o,n)] = ...
    Intensity_Calc(pi_avg32,vx311,vy311,vz311);
[Ix3221 Iy3221 Iz3221 I3221(o,n)] = ...
    Intensity_Calc(p_avg32,vx321,vy321,vz321);
[Ix3231 Iy3231 Iz3231 I3231(o,n)] = ...
    Intensity_Calc(p_avg32,vx331,vy331,vz331);
[Ix3311 Iy3311 Iz3311 I3311(o,n)] = ...
    Intensity_Calc(pi_avg33,vx311,vy311,vz311);
[Ix3321 Iy3321 Iz3321 I3321(o,n)] = ...
    Intensity_Calc(p_avg33,vx321,vy321,vz321);
[Ix3331 Iy3331 Iz3331 I3331(o,n)] = ...
    Intensity_Calc(p_avg33,vx331,vy331,vz331);

% Calculations for Moschioni et al. probe.
delxl = 2*a/(c/f);

uI01 = 2*sqrt(2)./abs(sin(2*pi*delxl));
uI02 = 2*sqrt(2)./abs(sin(2*pi*delxl./sqrt(2)));

alpha = .5*(uI02).^2./(2*(uI01).^2+.5*(uI02).^2);
beta = 2*(uI01).^2./(2*(uI01).^2+.5*(uI02).^2);

I42_311 = 0.5*real((pi34+pi32)/2*conj(v42_311));
I42_321 = 0.5*real((p34+p32)/2*conj(v42_321));
I42_331 = 0.5*real((p34+p32)/2*conj(v42_331));
I45_311 = 0.5*real((pi34+pi35)/2*conj(v45_311));
I45_321 = 0.5*real((p34+p35)/2*conj(v45_321));
I45_331 = 0.5*real((p34+p35)/2*conj(v45_331));
I43_311 = 0.5*real((pi34+pi33)/2*conj(v43_311));
I43_321 = 0.5*real((p34+p33)/2*conj(v43_321));
I43_331 = 0.5*real((p34+p33)/2*conj(v43_331));
I46_311 = 0.5*real((pi34+pi36)/2*conj(v46_311));
I46_321 = 0.5*real((p34+p36)/2*conj(v46_321));
I46_331 = 0.5*real((p34+p36)/2*conj(v46_331));
I41_311 = 0.5*real((pi34+pi31)/2*conj(v41_311));
I41_321 = 0.5*real((p34+p31)/2*conj(v41_321));
I41_331 = 0.5*real((p34+p31)/2*conj(v41_331));
I52_311 = 0.5*real((pi35+pi32)/2*conj(v52_311));
I52_321 = 0.5*real((p35+p32)/2*conj(v52_321));
I52_331 = 0.5*real((p35+p32)/2*conj(v52_331));
I32_311 = 0.5*real((pi33+pi32)/2*conj(v32_311));
I32_321 = 0.5*real((p33+p32)/2*conj(v32_321));
I32_331 = 0.5*real((p33+p32)/2*conj(v32_331));
I62_311 = 0.5*real((pi36+pi32)/2*conj(v62_311));
I62_321 = 0.5*real((p36+p32)/2*conj(v62_321));
```

```
I62_331 = 0.5*real((p36+p32)/2*conj(v62_331));
I12_311 = 0.5*real((pi31+pi32)/2*conj(v12_311));
I12_321 = 0.5*real((p31+p32)/2*conj(v12_321));
I12_331 = 0.5*real((p31+p32)/2*conj(v12_331));

I53_311 = 0.5*real((pi35+pi33)/2*conj(v53_311));
I53_321 = 0.5*real((p35+p33)/2*conj(v53_321));
I53_331 = 0.5*real((p35+p33)/2*conj(v53_331));
I51_311 = 0.5*real((pi35+pi31)/2*conj(v51_311));
I51_321 = 0.5*real((p35+p31)/2*conj(v51_321));
I51_331 = 0.5*real((p35+p31)/2*conj(v51_331));
I54_311 = 0.5*real((pi35+pi34)/2*conj(v54_311));
I54_321 = 0.5*real((p35+p34)/2*conj(v54_321));
I54_331 = 0.5*real((p35+p34)/2*conj(v54_331));
I56_311 = 0.5*real((pi35+pi36)/2*conj(v56_311));
I56_321 = 0.5*real((p35+p36)/2*conj(v56_321));
I56_331 = 0.5*real((p35+p36)/2*conj(v56_331));
I13_311 = 0.5*real((pi31+pi33)/2*conj(v13_311));
I13_321 = 0.5*real((p31+p33)/2*conj(v13_321));
I13_331 = 0.5*real((p31+p33)/2*conj(v13_331));
I23_311 = 0.5*real((pi32+pi33)/2*conj(v23_311));
I23_321 = 0.5*real((p32+p33)/2*conj(v23_321));
I23_331 = 0.5*real((p32+p33)/2*conj(v23_331));
I63_311 = 0.5*real((pi36+pi33)/2*conj(v63_311));
I63_321 = 0.5*real((p36+p33)/2*conj(v63_321));
I63_331 = 0.5*real((p36+p33)/2*conj(v63_331));

I61_311 = 0.5*real((pi36+pi31)/2*conj(v61_311));
I61_321 = 0.5*real((p36+p31)/2*conj(v61_321));
I61_331 = 0.5*real((p36+p31)/2*conj(v61_331));
I64_311 = 0.5*real((pi36+pi34)/2*conj(v64_311));
I64_321 = 0.5*real((p36+p34)/2*conj(v64_321));
I64_331 = 0.5*real((p36+p34)/2*conj(v64_331));
I65_311 = 0.5*real((pi36+pi35)/2*conj(v65_311));
I65_321 = 0.5*real((p36+p35)/2*conj(v65_321));
I65_331 = 0.5*real((p36+p35)/2*conj(v65_331));
I21_311 = 0.5*real((pi32+pi31)/2*conj(v21_311));
I21_321 = 0.5*real((p32+p31)/2*conj(v21_321));
I21_331 = 0.5*real((p32+p31)/2*conj(v21_331));
I31_311 = 0.5*real((pi33+pi31)/2*conj(v31_311));
I31_321 = 0.5*real((p33+p31)/2*conj(v31_321));
I31_331 = 0.5*real((p33+p31)/2*conj(v31_331));

Ix3411 = alpha*I42_311+beta/4/sqrt(2)*(I45_311+I43_311...
    +I46_311+I41_311+I52_311+I32_311+I62_311+I12_311);
Ix3421 = alpha*I42_321+beta/4/sqrt(2)*(I45_321+I43_321...
    +I46_321+I41_321+I52_321+I32_321+I62_321+I12_321);
Ix3431 = alpha*I42_331+beta/4/sqrt(2)*(I45_331+I43_331...
    +I46_331+I41_331+I52_331+I32_331+I62_331+I12_331);

Iy3411 = alpha*I53_311+beta/4/sqrt(2)*(I51_311+I52_311...
    +I54_311+I56_311+I13_311+I23_311+I43_311+I63_311);
Iy3421 = alpha*I53_321+beta/4/sqrt(2)*(I51_321+I52_321...
    +I54_321+I56_321+I13_321+I23_321+I43_321+I63_321);
Iy3431 = alpha*I53_331+beta/4/sqrt(2)*(I51_331+I52_331...
```

```
    +I54_331+I56_331+I13_331+I23_331+I43_331+I63_331);

Iz3411 = alpha*I61_311+beta/4/sqrt(2)*(I62_311+I63_311...
    +I64_311+I65_311+I21_311+I31_311+I41_311+I51_311);
Iz3421 = alpha*I61_321+beta/4/sqrt(2)*(I62_321+I63_321...
    +I64_321+I65_321+I21_321+I31_321+I41_321+I51_321);
Iz3431 = alpha*I61_331+beta/4/sqrt(2)*(I62_331+I63_331...
    +I64_331+I65_331+I21_331+I31_331+I41_331+I51_331);

I3411(o,n) = ...
    (abs(Ix3411).^2+abs(Iy3411).^2+abs(Iz3411).^2).^(1/2);
I3421(o,n) = ...
    (abs(Ix3421).^2+abs(Iy3421).^2+abs(Iz3421).^2).^(1/2);
I3431(o,n) = ...
    (abs(Ix3431).^2+abs(Iy3431).^2+abs(Iz3431).^2).^(1/2);


%%%%%%% Energy Density Calculations
%%%%%%% 1st subscript: type of probe
%%%%%%% 2nd: how p was calculated
%%%%%%% 3rd: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%      sphere w/o 3/2 factor
%%%%%%% 4th: 1 = no Taylor, 2 = with Taylor

W1211(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg12)^2/4/rho/c^2;
W1221(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1231(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1311(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg13)^2/4/rho/c^2;
W1321(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1331(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1411(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg14)^2/4/rho/c^2;
W1421(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1431(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1511(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg15)^2/4/rho/c^2;
W1521(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg15)^2/4/rho/c^2;
W1531(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg15)^2/4/rho/c^2;
W2111(o,n) = rho*abs(v211(o,n))^2/4 + abs(pi_avg21)^2/4/rho/c^2;
W2121(o,n) = rho*abs(v221(o,n))^2/4 + abs(p_avg21)^2/4/rho/c^2;
W2131(o,n) = rho*abs(v231(o,n))^2/4 + abs(p_avg21)^2/4/rho/c^2;
W2211(o,n) = rho*abs(v211(o,n))^2/4 + abs(pi_avg22)^2/4/rho/c^2;
W2221(o,n) = rho*abs(v221(o,n))^2/4 + abs(p_avg22)^2/4/rho/c^2;
W2231(o,n) = rho*abs(v231(o,n))^2/4 + abs(p_avg22)^2/4/rho/c^2;
W3211(o,n) = rho*abs(v311(o,n))^2/4 + abs(pi_avg32)^2/4/rho/c^2;
W3221(o,n) = rho*abs(v321(o,n))^2/4 + abs(p_avg32)^2/4/rho/c^2;
W3231(o,n) = rho*abs(v331(o,n))^2/4 + abs(p_avg32)^2/4/rho/c^2;
W3311(o,n) = rho*abs(v311(o,n))^2/4 + abs(pi_avg33)^2/4/rho/c^2;
W3321(o,n) = rho*abs(v321(o,n))^2/4 + abs(p_avg33)^2/4/rho/c^2;
W3331(o,n) = rho*abs(v331(o,n))^2/4 + abs(p_avg33)^2/4/rho/c^2;
W1212(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg12)^2/4/rho/c^2;
W1222(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1232(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1312(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg13)^2/4/rho/c^2;
W1322(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1332(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1412(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg14)^2/4/rho/c^2;
```

```
W1422(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1432(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1512(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg15)^2/4/rho/c^2;
W1522(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg15)^2/4/rho/c^2;
W1532(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg15)^2/4/rho/c^2;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%        DIRECTION CALCULATIONS      %%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


Ivect = [Ix Iy Iz];
vvect = [vx vy vz];

I1111ang(o,n,q) = Angle_Calc(Ivect,Ix1111,Iy1111,Iz1111);
I1121ang(o,n,q) = Angle_Calc(Ivect,Ix1121,Iy1121,Iz1121);
I1131ang(o,n,q) = Angle_Calc(Ivect,Ix1131,Iy1131,Iz1131);
I1211ang(o,n,q) = Angle_Calc(Ivect,Ix1211,Iy1211,Iz1211);
I1221ang(o,n,q) = Angle_Calc(Ivect,Ix1221,Iy1221,Iz1221);
I1231ang(o,n,q) = Angle_Calc(Ivect,Ix1231,Iy1231,Iz1231);
I1311ang(o,n,q) = Angle_Calc(Ivect,Ix1311,Iy1311,Iz1311);
I1321ang(o,n,q) = Angle_Calc(Ivect,Ix1321,Iy1321,Iz1321);
I1331ang(o,n,q) = Angle_Calc(Ivect,Ix1331,Iy1331,Iz1331);
I1411ang(o,n,q) = Angle_Calc(Ivect,Ix1411,Iy1411,Iz1411);
I1421ang(o,n,q) = Angle_Calc(Ivect,Ix1421,Iy1421,Iz1421);
I1431ang(o,n,q) = Angle_Calc(Ivect,Ix1431,Iy1431,Iz1431);
I1511ang(o,n,q) = Angle_Calc(Ivect,Ix1511,Iy1511,Iz1511);
I1521ang(o,n,q) = Angle_Calc(Ivect,Ix1521,Iy1521,Iz1521);
I1531ang(o,n,q) = Angle_Calc(Ivect,Ix1531,Iy1531,Iz1531);

I2111ang(o,n,q) = Angle_Calc(Ivect,Ix2111,Iy2111,Iz2111);
I2121ang(o,n,q) = Angle_Calc(Ivect,Ix2121,Iy2121,Iz2121);
I2131ang(o,n,q) = Angle_Calc(Ivect,Ix2131,Iy2131,Iz2131);
I2211ang(o,n,q) = Angle_Calc(Ivect,Ix2211,Iy2211,Iz2211);
I2221ang(o,n,q) = Angle_Calc(Ivect,Ix2221,Iy2221,Iz2221);
I2231ang(o,n,q) = Angle_Calc(Ivect,Ix2231,Iy2231,Iz2231);

I3111ang(o,n,q) = Angle_Calc(Ivect,Ix3111,Iy3111,Iz3111);
I3121ang(o,n,q) = Angle_Calc(Ivect,Ix3121,Iy3121,Iz3121);
I3131ang(o,n,q) = Angle_Calc(Ivect,Ix3131,Iy3131,Iz3131);
I3211ang(o,n,q) = Angle_Calc(Ivect,Ix3211,Iy3211,Iz3211);
I3221ang(o,n,q) = Angle_Calc(Ivect,Ix3221,Iy3221,Iz3221);
I3231ang(o,n,q) = Angle_Calc(Ivect,Ix3231,Iy3231,Iz3231);
I3311ang(o,n,q) = Angle_Calc(Ivect,Ix3311,Iy3311,Iz3311);
I3321ang(o,n,q) = Angle_Calc(Ivect,Ix3321,Iy3321,Iz3321);
I3331ang(o,n,q) = Angle_Calc(Ivect,Ix3331,Iy3331,Iz3331);
I3411ang(o,n,q) = Angle_Calc(Ivect,Ix3411,Iy3411,Iz3411);
I3421ang(o,n,q) = Angle_Calc(Ivect,Ix3421,Iy3421,Iz3421);
I3431ang(o,n,q) = Angle_Calc(Ivect,Ix3431,Iy3431,Iz3431);

I1112ang(o,n,q) = Angle_Calc(Ivect,Ix1112,Iy1112,Iz1112);
I1122ang(o,n,q) = Angle_Calc(Ivect,Ix1122,Iy1122,Iz1122);
```

```
        I1132ang(o,n,q) = Angle_Calc(Ivect,Ix1132,Iy1132,Iz1132);
        I1212ang(o,n,q) = Angle_Calc(Ivect,Ix1212,Iy1212,Iz1212);
        I1222ang(o,n,q) = Angle_Calc(Ivect,Ix1222,Iy1222,Iz1222);
        I1232ang(o,n,q) = Angle_Calc(Ivect,Ix1232,Iy1232,Iz1232);
        I1312ang(o,n,q) = Angle_Calc(Ivect,Ix1312,Iy1312,Iz1312);
        I1322ang(o,n,q) = Angle_Calc(Ivect,Ix1322,Iy1322,Iz1322);
        I1332ang(o,n,q) = Angle_Calc(Ivect,Ix1332,Iy1332,Iz1332);
        I1412ang(o,n,q) = Angle_Calc(Ivect,Ix1412,Iy1412,Iz1412);
        I1422ang(o,n,q) = Angle_Calc(Ivect,Ix1422,Iy1422,Iz1422);
        I1432ang(o,n,q) = Angle_Calc(Ivect,Ix1432,Iy1432,Iz1432);
        I1512ang(o,n,q) = Angle_Calc(Ivect,Ix1512,Iy1512,Iz1512);
        I1522ang(o,n,q) = Angle_Calc(Ivect,Ix1522,Iy1522,Iz1522);
        I1532ang(o,n,q) = Angle_Calc(Ivect,Ix1532,Iy1532,Iz1532);

        v111ang(o,n,q) = Angle_Calc(vvect,vx111,vy111,vz111);
        v121ang(o,n,q) = Angle_Calc(vvect,vx121,vy121,vz121);
        v131ang(o,n,q) = Angle_Calc(vvect,vx131,vy131,vz131);

        v211ang(o,n,q) = Angle_Calc(vvect,vx211,vy211,vz211);
        v221ang(o,n,q) = Angle_Calc(vvect,vx221,vy221,vz221);
        v231ang(o,n,q) = Angle_Calc(vvect,vx231,vy231,vz231);

        v311ang(o,n,q) = Angle_Calc(vvect,vx311,vy311,vz311);
        v321ang(o,n,q) = Angle_Calc(vvect,vx321,vy321,vz321);
        v331ang(o,n,q) = Angle_Calc(vvect,vx331,vy331,vz331);

        v112ang(o,n,q) = Angle_Calc(vvect,vx112,vy112,vz112);
        v122ang(o,n,q) = Angle_Calc(vvect,vx122,vy122,vz122);
        v132ang(o,n,q) = Angle_Calc(vvect,vx132,vy132,vz132);


        n = n+1;
    end
    o = o+1;
end

phi=0:res_step:2*pi-res_step;
theta=0:res_step:pi;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Finite Diff Bias Calculations   %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[I1111b{1}(:,:,q) I1111b{2}(q) I1111b{3}(q) I1111b{4}(q) I1111b{5}(q)...
    I1111b{6}(q) I1111b{7}(q) I1111b{8}(q) I1111b{9}(q)] = ...
    Bias_Calc(res_step,I1111,I,10);
[I1121b{1}(:,:,q) I1121b{2}(q) I1121b{3}(q) I1121b{4}(q) I1121b{5}(q)...
    I1121b{6}(q) I1121b{7}(q) I1121b{8}(q) I1121b{9}(q)] = ...
    Bias_Calc(res_step,I1121,I,10);
[I1131b{1}(:,:,q) I1131b{2}(q) I1131b{3}(q) I1131b{4}(q) I1131b{5}(q)...
    I1131b{6}(q) I1131b{7}(q) I1131b{8}(q) I1131b{9}(q)] = ...
    Bias_Calc(res_step,I1131,I,10);
```

```
[I1211b{1}(:,:,q) I1211b{2}(q) I1211b{3}(q) I1211b{4}(q) I1211b{5}(q)...
    I1211b{6}(q) I1211b{7}(q) I1211b{8}(q) I1211b{9}(q)] = ...
    Bias_Calc(res_step,I1211,I,10);
[I1221b{1}(:,:,q) I1221b{2}(q) I1221b{3}(q) I1221b{4}(q) I1221b{5}(q)...
    I1221b{6}(q) I1221b{7}(q) I1221b{8}(q) I1221b{9}(q)] = ...
    Bias_Calc(res_step,I1221,I,10);
[I1231b{1}(:,:,q) I1231b{2}(q) I1231b{3}(q) I1231b{4}(q) I1231b{5}(q)...
    I1231b{6}(q) I1231b{7}(q) I1231b{8}(q) I1231b{9}(q)] = ...
    Bias_Calc(res_step,I1231,I,10);
[I1311b{1}(:,:,q) I1311b{2}(q) I1311b{3}(q) I1311b{4}(q) I1311b{5}(q)...
    I1311b{6}(q) I1311b{7}(q) I1311b{8}(q) I1311b{9}(q)] = ...
    Bias_Calc(res_step,I1311,I,10);
[I1321b{1}(:,:,q) I1321b{2}(q) I1321b{3}(q) I1321b{4}(q) I1321b{5}(q)...
    I1321b{6}(q) I1321b{7}(q) I1321b{8}(q) I1321b{9}(q)] = ...
    Bias_Calc(res_step,I1321,I,10);
[I1331b{1}(:,:,q) I1331b{2}(q) I1331b{3}(q) I1331b{4}(q) I1331b{5}(q)...
    I1331b{6}(q) I1331b{7}(q) I1331b{8}(q) I1331b{9}(q)] = ...
    Bias_Calc(res_step,I1331,I,10);
[I1411b{1}(:,:,q) I1411b{2}(q) I1411b{3}(q) I1411b{4}(q) I1411b{5}(q)...
    I1411b{6}(q) I1411b{7}(q) I1411b{8}(q) I1411b{9}(q)] = ...
    Bias_Calc(res_step,I1411,I,10);
[I1421b{1}(:,:,q) I1421b{2}(q) I1421b{3}(q) I1421b{4}(q) I1421b{5}(q)...
    I1421b{6}(q) I1421b{7}(q) I1421b{8}(q) I1421b{9}(q)] = ...
    Bias_Calc(res_step,I1421,I,10);
[I1431b{1}(:,:,q) I1431b{2}(q) I1431b{3}(q) I1431b{4}(q) I1431b{5}(q)...
    I1431b{6}(q) I1431b{7}(q) I1431b{8}(q) I1431b{9}(q)] = ...
    Bias_Calc(res_step,I1431,I,10);
[I1511b{1}(:,:,q) I1511b{2}(q) I1511b{3}(q) I1511b{4}(q) I1511b{5}(q)...
    I1511b{6}(q) I1511b{7}(q) I1511b{8}(q) I1511b{9}(q)] = ...
    Bias_Calc(res_step,I1511,I,10);
[I1521b{1}(:,:,q) I1521b{2}(q) I1521b{3}(q) I1521b{4}(q) I1521b{5}(q)...
    I1521b{6}(q) I1521b{7}(q) I1521b{8}(q) I1521b{9}(q)] = ...
    Bias_Calc(res_step,I1521,I,10);
[I1531b{1}(:,:,q) I1531b{2}(q) I1531b{3}(q) I1531b{4}(q) I1531b{5}(q)...
    I1531b{6}(q) I1531b{7}(q) I1531b{8}(q) I1531b{9}(q)] = ...
    Bias_Calc(res_step,I1531,I,10);


[I2111b{1}(:,:,q) I2111b{2}(q) I2111b{3}(q) I2111b{4}(q) I2111b{5}(q)...
    I2111b{6}(q) I2111b{7}(q) I2111b{8}(q) I2111b{9}(q)] = ...
    Bias_Calc(res_step,I2111,I,10);
[I2121b{1}(:,:,q) I2121b{2}(q) I2121b{3}(q) I2121b{4}(q) I2121b{5}(q)...
    I2121b{6}(q) I2121b{7}(q) I2121b{8}(q) I2121b{9}(q)] = ...
    Bias_Calc(res_step,I2121,I,10);
[I2131b{1}(:,:,q) I2131b{2}(q) I2131b{3}(q) I2131b{4}(q) I2131b{5}(q)...
    I2131b{6}(q) I2131b{7}(q) I2131b{8}(q) I2131b{9}(q)] = ...
    Bias_Calc(res_step,I2131,I,10);
[I2211b{1}(:,:,q) I2211b{2}(q) I2211b{3}(q) I2211b{4}(q) I2211b{5}(q)...
    I2211b{6}(q) I2211b{7}(q) I2211b{8}(q) I2211b{9}(q)] = ...
    Bias_Calc(res_step,I2211,I,10);
[I2221b{1}(:,:,q) I2221b{2}(q) I2221b{3}(q) I2221b{4}(q) I2221b{5}(q)...
    I2221b{6}(q) I2221b{7}(q) I2221b{8}(q) I2221b{9}(q)] = ...
    Bias_Calc(res_step,I2221,I,10);
[I2231b{1}(:,:,q) I2231b{2}(q) I2231b{3}(q) I2231b{4}(q) I2231b{5}(q)...
    I2231b{6}(q) I2231b{7}(q) I2231b{8}(q) I2231b{9}(q)] = ...
    Bias_Calc(res_step,I2231,I,10);
```

```
[I3111b{1}(:,:,q) I3111b{2}(q) I3111b{3}(q) I3111b{4}(q) I3111b{5}(q)...
    I3111b{6}(q) I3111b{7}(q) I3111b{8}(q) I3111b{9}(q)] = ...
    Bias_Calc(res_step,I3111,I,10);
[I3121b{1}(:,:,q) I3121b{2}(q) I3121b{3}(q) I3121b{4}(q) I3121b{5}(q)...
    I3121b{6}(q) I3121b{7}(q) I3121b{8}(q) I3121b{9}(q)] = ...
    Bias_Calc(res_step,I3121,I,10);
[I3131b{1}(:,:,q) I3131b{2}(q) I3131b{3}(q) I3131b{4}(q) I3131b{5}(q)...
    I3131b{6}(q) I3131b{7}(q) I3131b{8}(q) I3131b{9}(q)] = ...
    Bias_Calc(res_step,I3131,I,10);
[I3211b{1}(:,:,q) I3211b{2}(q) I3211b{3}(q) I3211b{4}(q) I3211b{5}(q)...
    I3211b{6}(q) I3211b{7}(q) I3211b{8}(q) I3211b{9}(q)] = ...
    Bias_Calc(res_step,I3211,I,10);
[I3221b{1}(:,:,q) I3221b{2}(q) I3221b{3}(q) I3221b{4}(q) I3221b{5}(q)...
    I3221b{6}(q) I3221b{7}(q) I3221b{8}(q) I3221b{9}(q)] = ...
    Bias_Calc(res_step,I3221,I,10);
[I3231b{1}(:,:,q) I3231b{2}(q) I3231b{3}(q) I3231b{4}(q) I3231b{5}(q)...
    I3231b{6}(q) I3231b{7}(q) I3231b{8}(q) I3231b{9}(q)] = ...
    Bias_Calc(res_step,I3231,I,10);
[I3311b{1}(:,:,q) I3311b{2}(q) I3311b{3}(q) I3311b{4}(q) I3311b{5}(q)...
    I3311b{6}(q) I3311b{7}(q) I3311b{8}(q) I3311b{9}(q)] = ...
    Bias_Calc(res_step,I3311,I,10);
[I3321b{1}(:,:,q) I3321b{2}(q) I3321b{3}(q) I3321b{4}(q) I3321b{5}(q)...
    I3321b{6}(q) I3321b{7}(q) I3321b{8}(q) I3321b{9}(q)] = ...
    Bias_Calc(res_step,I3321,I,10);
[I3331b{1}(:,:,q) I3331b{2}(q) I3331b{3}(q) I3331b{4}(q) I3331b{5}(q)...
    I3331b{6}(q) I3331b{7}(q) I3331b{8}(q) I3331b{9}(q)] = ...
    Bias_Calc(res_step,I3331,I,10);
[I3411b{1}(:,:,q) I3411b{2}(q) I3411b{3}(q) I3411b{4}(q) I3411b{5}(q)...
    I3411b{6}(q) I3411b{7}(q) I3411b{8}(q) I3411b{9}(q)] = ...
    Bias_Calc(res_step,I3411,I,10);
[I3421b{1}(:,:,q) I3421b{2}(q) I3421b{3}(q) I3421b{4}(q) I3421b{5}(q)...
    I3421b{6}(q) I3421b{7}(q) I3421b{8}(q) I3421b{9}(q)] = ...
    Bias_Calc(res_step,I3421,I,10);
[I3431b{1}(:,:,q) I3431b{2}(q) I3431b{3}(q) I3431b{4}(q) I3431b{5}(q)...
    I3431b{6}(q) I3431b{7}(q) I3431b{8}(q) I3431b{9}(q)] = ...
    Bias_Calc(res_step,I3431,I,10);


[I1112b{1}(:,:,q) I1112b{2}(q) I1112b{3}(q) I1112b{4}(q) I1112b{5}(q)...
    I1112b{6}(q) I1112b{7}(q) I1112b{8}(q) I1112b{9}(q)] = ...
    Bias_Calc(res_step,I1112,I,10);
[I1122b{1}(:,:,q) I1122b{2}(q) I1122b{3}(q) I1122b{4}(q) I1122b{5}(q)...
    I1122b{6}(q) I1122b{7}(q) I1122b{8}(q) I1122b{9}(q)] = ...
    Bias_Calc(res_step,I1122,I,10);
[I1132b{1}(:,:,q) I1132b{2}(q) I1132b{3}(q) I1132b{4}(q) I1132b{5}(q)...
    I1132b{6}(q) I1132b{7}(q) I1132b{8}(q) I1132b{9}(q)] = ...
    Bias_Calc(res_step,I1132,I,10);
[I1212b{1}(:,:,q) I1212b{2}(q) I1212b{3}(q) I1212b{4}(q) I1212b{5}(q)...
    I1212b{6}(q) I1212b{7}(q) I1212b{8}(q) I1212b{9}(q)] = ...
    Bias_Calc(res_step,I1212,I,10);
[I1222b{1}(:,:,q) I1222b{2}(q) I1222b{3}(q) I1222b{4}(q) I1222b{5}(q)...
    I1222b{6}(q) I1222b{7}(q) I1222b{8}(q) I1222b{9}(q)] = ...
    Bias_Calc(res_step,I1222,I,10);
[I1232b{1}(:,:,q) I1232b{2}(q) I1232b{3}(q) I1232b{4}(q) I1232b{5}(q)...
    I1232b{6}(q) I1232b{7}(q) I1232b{8}(q) I1232b{9}(q)] = ...
    Bias_Calc(res_step,I1232,I,10);
[I1312b{1}(:,:,q) I1312b{2}(q) I1312b{3}(q) I1312b{4}(q) I1312b{5}(q)...
```

```
    I1312b{6}(q) I1312b{7}(q) I1312b{8}(q) I1312b{9}(q)] = ...
    Bias_Calc(res_step,I1312,I,10);
[I1322b{1}(:,:,q) I1322b{2}(q) I1322b{3}(q) I1322b{4}(q) I1322b{5}(q)...
    I1322b{6}(q) I1322b{7}(q) I1322b{8}(q) I1322b{9}(q)] = ...
    Bias_Calc(res_step,I1322,I,10);
[I1332b{1}(:,:,q) I1332b{2}(q) I1332b{3}(q) I1332b{4}(q) I1332b{5}(q)...
    I1332b{6}(q) I1332b{7}(q) I1332b{8}(q) I1332b{9}(q)] = ...
    Bias_Calc(res_step,I1332,I,10);
[I1412b{1}(:,:,q) I1412b{2}(q) I1412b{3}(q) I1412b{4}(q) I1412b{5}(q)...
    I1412b{6}(q) I1412b{7}(q) I1412b{8}(q) I1412b{9}(q)] = ...
    Bias_Calc(res_step,I1412,I,10);
[I1422b{1}(:,:,q) I1422b{2}(q) I1422b{3}(q) I1422b{4}(q) I1422b{5}(q)...
    I1422b{6}(q) I1422b{7}(q) I1422b{8}(q) I1422b{9}(q)] = ...
    Bias_Calc(res_step,I1422,I,10);
[I1432b{1}(:,:,q) I1432b{2}(q) I1432b{3}(q) I1432b{4}(q) I1432b{5}(q)...
    I1432b{6}(q) I1432b{7}(q) I1432b{8}(q) I1432b{9}(q)] = ...
    Bias_Calc(res_step,I1432,I,10);
[I1512b{1}(:,:,q) I1512b{2}(q) I1512b{3}(q) I1512b{4}(q) I1512b{5}(q)...
    I1512b{6}(q) I1512b{7}(q) I1512b{8}(q) I1512b{9}(q)] = ...
    Bias_Calc(res_step,I1512,I,10);
[I1522b{1}(:,:,q) I1522b{2}(q) I1522b{3}(q) I1522b{4}(q) I1522b{5}(q)...
    I1522b{6}(q) I1522b{7}(q) I1522b{8}(q) I1522b{9}(q)] = ...
    Bias_Calc(res_step,I1522,I,10);
[I1532b{1}(:,:,q) I1532b{2}(q) I1532b{3}(q) I1532b{4}(q) I1532b{5}(q)...
    I1532b{6}(q) I1532b{7}(q) I1532b{8}(q) I1532b{9}(q)] = ...
    Bias_Calc(res_step,I1532,I,10);

[Pi12b{1}(:,:,q) Pi12b{2}(q) Pi12b{3}(q) Pi12b{4}(q) Pi12b{5}(q) ...
    Pi12b{6}(q) Pi12b{7}(q) Pi12b{8}(q) Pi12b{9}(q)] = ...
    Bias_Calc(res_step,Pi12,p0,20);
[Pi13b{1}(:,:,q) Pi13b{2}(q) Pi13b{3}(q) Pi13b{4}(q) Pi13b{5}(q) ...
    Pi13b{6}(q) Pi13b{7}(q) Pi13b{8}(q) Pi13b{9}(q)] = ...
    Bias_Calc(res_step,Pi13,p0,20);
[Pi14b{1}(:,:,q) Pi14b{2}(q) Pi14b{3}(q) Pi14b{4}(q) Pi14b{5}(q) ...
    Pi14b{6}(q) Pi14b{7}(q) Pi14b{8}(q) Pi14b{9}(q)] = ...
    Bias_Calc(res_step,Pi14,p0,20);
[Pi15b{1}(:,:,q) Pi15b{2}(q) Pi15b{3}(q) Pi15b{4}(q) Pi15b{5}(q) ...
    Pi15b{6}(q) Pi15b{7}(q) Pi15b{8}(q) Pi15b{9}(q)] = ...
    Bias_Calc(res_step,Pi15,p0,20);
[P12b{1}(:,:,q) P12b{2}(q) P12b{3}(q) P12b{4}(q) P12b{5}(q) ...
    P12b{6}(q) P12b{7}(q) P12b{8}(q) P12b{9}(q)] = ...
    Bias_Calc(res_step,P12,p0,20);
[P13b{1}(:,:,q) P13b{2}(q) P13b{3}(q) P13b{4}(q) P13b{5}(q) ...
    P13b{6}(q) P13b{7}(q) P13b{8}(q) P13b{9}(q)] = ...
    Bias_Calc(res_step,P13,p0,20);
[P14b{1}(:,:,q) P14b{2}(q) P14b{3}(q) P14b{4}(q) P14b{5}(q) ...
    P14b{6}(q) P14b{7}(q) P14b{8}(q) P14b{9}(q)] = ...
    Bias_Calc(res_step,P14,p0,20);
[P15b{1}(:,:,q) P15b{2}(q) P15b{3}(q) P15b{4}(q) P15b{5}(q) ...
    P15b{6}(q) P15b{7}(q) P15b{8}(q) P15b{9}(q)] = ...
    Bias_Calc(res_step,P15,p0,20);

[Pi21b{1}(:,:,q) Pi21b{2}(q) Pi21b{3}(q) Pi21b{4}(q) Pi21b{5}(q) ...
    Pi21b{6}(q) Pi21b{7}(q) Pi21b{8}(q) Pi21b{9}(q)] = ...
    Bias_Calc(res_step,Pi21,p0,20);
[Pi22b{1}(:,:,q) Pi22b{2}(q) Pi22b{3}(q) Pi22b{4}(q) Pi22b{5}(q) ...
```

```
    Pi22b{6}(q) Pi22b{7}(q) Pi22b{8}(q) Pi22b{9}(q)] = ...
    Bias_Calc(res_step,Pi22,p0,20);
[P21b{1}(:,:,q) P21b{2}(q) P21b{3}(q) P21b{4}(q) P21b{5}(q) ...
    P21b{6}(q) P21b{7}(q) P21b{8}(q) P21b{9}(q)] = ...
    Bias_Calc(res_step,P21,p0,20);
[P22b{1}(:,:,q) P22b{2}(q) P22b{3}(q) P22b{4}(q) P22b{5}(q) ...
    P22b{6}(q) P22b{7}(q) P22b{8}(q) P22b{9}(q)] = ...
    Bias_Calc(res_step,P22,p0,20);

[Pi32b{1}(:,:,q) Pi32b{2}(q) Pi32b{3}(q) Pi32b{4}(q) Pi32b{5}(q) ...
    Pi32b{6}(q) Pi32b{7}(q) Pi32b{8}(q) Pi32b{9}(q)] = ...
    Bias_Calc(res_step,Pi32,p0,20);
[Pi33b{1}(:,:,q) Pi33b{2}(q) Pi33b{3}(q) Pi33b{4}(q) Pi33b{5}(q) ...
    Pi33b{6}(q) Pi33b{7}(q) Pi33b{8}(q) Pi33b{9}(q)] = ...
    Bias_Calc(res_step,Pi33,p0,20);
[P32b{1}(:,:,q) P32b{2}(q) P32b{3}(q) P32b{4}(q) P32b{5}(q) ...
    P32b{6}(q) P32b{7}(q) P32b{8}(q) P32b{9}(q)] = ...
    Bias_Calc(res_step,P32,p0,20);
[P33b{1}(:,:,q) P33b{2}(q) P33b{3}(q) P33b{4}(q) P33b{5}(q) ...
    P33b{6}(q) P33b{7}(q) P33b{8}(q) P33b{9}(q)] = ...
    Bias_Calc(res_step,P33,p0,20);

[v111b{1}(:,:,q) v111b{2}(q) v111b{3}(q) v111b{4}(q) v111b{5}(q) ...
    v111b{6}(q) v111b{7}(q) v111b{8}(q) v111b{9}(q)] = ...
    Bias_Calc(res_step,v111,v,20);
[v121b{1}(:,:,q) v121b{2}(q) v121b{3}(q) v121b{4}(q) v121b{5}(q) ...
    v121b{6}(q) v121b{7}(q) v121b{8}(q) v121b{9}(q)] = ...
    Bias_Calc(res_step,v121,v,20);
[v131b{1}(:,:,q) v131b{2}(q) v131b{3}(q) v131b{4}(q) v131b{5}(q) ...
    v131b{6}(q) v131b{7}(q) v131b{8}(q) v131b{9}(q)] = ...
    Bias_Calc(res_step,v131,v,20);

[v211b{1}(:,:,q) v211b{2}(q) v211b{3}(q) v211b{4}(q) v211b{5}(q) ...
    v211b{6}(q) v211b{7}(q) v211b{8}(q) v211b{9}(q)] = ...
    Bias_Calc(res_step,v211,v,20);
[v221b{1}(:,:,q) v221b{2}(q) v221b{3}(q) v221b{4}(q) v221b{5}(q) ...
    v221b{6}(q) v221b{7}(q) v221b{8}(q) v221b{9}(q)] = ...
    Bias_Calc(res_step,v221,v,20);
[v231b{1}(:,:,q) v231b{2}(q) v231b{3}(q) v231b{4}(q) v231b{5}(q) ...
    v231b{6}(q) v231b{7}(q) v231b{8}(q) v231b{9}(q)] = ...
    Bias_Calc(res_step,v231,v,20);

[v311b{1}(:,:,q) v311b{2}(q) v311b{3}(q) v311b{4}(q) v311b{5}(q) ...
    v311b{6}(q) v311b{7}(q) v311b{8}(q) v311b{9}(q)] = ...
    Bias_Calc(res_step,v311,v,20);
[v321b{1}(:,:,q) v321b{2}(q) v321b{3}(q) v321b{4}(q) v321b{5}(q) ...
    v321b{6}(q) v321b{7}(q) v321b{8}(q) v321b{9}(q)] = ...
    Bias_Calc(res_step,v321,v,20);
[v331b{1}(:,:,q) v331b{2}(q) v331b{3}(q) v331b{4}(q) v331b{5}(q) ...
    v331b{6}(q) v331b{7}(q) v331b{8}(q) v331b{9}(q)] = ...
    Bias_Calc(res_step,v331,v,20);

[v112b{1}(:,:,q) v112b{2}(q) v112b{3}(q) v112b{4}(q) v112b{5}(q) ...
    v112b{6}(q) v112b{7}(q) v112b{8}(q) v112b{9}(q)] = ...
    Bias_Calc(res_step,v112,v,20);
```

```
[v122b{1}(:,:,q) v122b{2}(q) v122b{3}(q) v122b{4}(q) v122b{5}(q) ...
    v122b{6}(q) v122b{7}(q) v122b{8}(q) v122b{9}(q)] = ...
    Bias_Calc(res_step,v122,v,20);
[v132b{1}(:,:,q) v132b{2}(q) v132b{3}(q) v132b{4}(q) v132b{5}(q) ...
    v132b{6}(q) v132b{7}(q) v132b{8}(q) v132b{9}(q)] = ...
    Bias_Calc(res_step,v132,v,20);

[W1211b{1}(:,:,q) W1211b{2}(q) W1211b{3}(q) W1211b{4}(q) W1211b{5}(q)...
    W1211b{6}(q) W1211b{7}(q) W1211b{8}(q) W1211b{9}(q)] = ...
    Bias_Calc(res_step,W1211,W,10);
[W1221b{1}(:,:,q) W1221b{2}(q) W1221b{3}(q) W1221b{4}(q) W1221b{5}(q)...
    W1221b{6}(q) W1221b{7}(q) W1221b{8}(q) W1221b{9}(q)] = ...
    Bias_Calc(res_step,W1221,W,10);
[W1231b{1}(:,:,q) W1231b{2}(q) W1231b{3}(q) W1231b{4}(q) W1231b{5}(q)...
    W1231b{6}(q) W1231b{7}(q) W1231b{8}(q) W1231b{9}(q)] = ...
    Bias_Calc(res_step,W1231,W,10);
[W1311b{1}(:,:,q) W1311b{2}(q) W1311b{3}(q) W1311b{4}(q) W1311b{5}(q)...
    W1311b{6}(q) W1311b{7}(q) W1311b{8}(q) W1311b{9}(q)] = ...
    Bias_Calc(res_step,W1311,W,10);
[W1321b{1}(:,:,q) W1321b{2}(q) W1321b{3}(q) W1321b{4}(q) W1321b{5}(q)...
    W1321b{6}(q) W1321b{7}(q) W1321b{8}(q) W1321b{9}(q)] = ...
    Bias_Calc(res_step,W1321,W,10);
[W1331b{1}(:,:,q) W1331b{2}(q) W1331b{3}(q) W1331b{4}(q) W1331b{5}(q)...
    W1331b{6}(q) W1331b{7}(q) W1331b{8}(q) W1331b{9}(q)] = ...
    Bias_Calc(res_step,W1331,W,10);
[W1411b{1}(:,:,q) W1411b{2}(q) W1411b{3}(q) W1411b{4}(q) W1411b{5}(q)...
    W1411b{6}(q) W1411b{7}(q) W1411b{8}(q) W1411b{9}(q)] = ...
    Bias_Calc(res_step,W1411,W,10);
[W1421b{1}(:,:,q) W1421b{2}(q) W1421b{3}(q) W1421b{4}(q) W1421b{5}(q)...
    W1421b{6}(q) W1421b{7}(q) W1421b{8}(q) W1421b{9}(q)] = ...
    Bias_Calc(res_step,W1421,W,10);
[W1431b{1}(:,:,q) W1431b{2}(q) W1431b{3}(q) W1431b{4}(q) W1431b{5}(q)...
    W1431b{6}(q) W1431b{7}(q) W1431b{8}(q) W1431b{9}(q)] = ...
    Bias_Calc(res_step,W1431,W,10);
[W1511b{1}(:,:,q) W1511b{2}(q) W1511b{3}(q) W1511b{4}(q) W1511b{5}(q)...
    W1511b{6}(q) W1511b{7}(q) W1511b{8}(q) W1511b{9}(q)] = ...
    Bias_Calc(res_step,W1511,W,10);
[W1521b{1}(:,:,q) W1521b{2}(q) W1521b{3}(q) W1521b{4}(q) W1521b{5}(q)...
    W1521b{6}(q) W1521b{7}(q) W1521b{8}(q) W1521b{9}(q)] = ...
    Bias_Calc(res_step,W1521,W,10);
[W1531b{1}(:,:,q) W1531b{2}(q) W1531b{3}(q) W1531b{4}(q) W1531b{5}(q)...
    W1531b{6}(q) W1531b{7}(q) W1531b{8}(q) W1531b{9}(q)] = ...
    Bias_Calc(res_step,W1531,W,10);

[W2111b{1}(:,:,q) W2111b{2}(q) W2111b{3}(q) W2111b{4}(q) W2111b{5}(q)...
    W2111b{6}(q) W2111b{7}(q) W2111b{8}(q) W2111b{9}(q)] = ...
    Bias_Calc(res_step,W2111,W,10);
[W2121b{1}(:,:,q) W2121b{2}(q) W2121b{3}(q) W2121b{4}(q) W2121b{5}(q)...
    W2121b{6}(q) W2121b{7}(q) W2121b{8}(q) W2121b{9}(q)] = ...
    Bias_Calc(res_step,W2121,W,10);
[W2131b{1}(:,:,q) W2131b{2}(q) W2131b{3}(q) W2131b{4}(q) W2131b{5}(q)...
    W2131b{6}(q) W2131b{7}(q) W2131b{8}(q) W2131b{9}(q)] = ...
    Bias_Calc(res_step,W2131,W,10);
[W2211b{1}(:,:,q) W2211b{2}(q) W2211b{3}(q) W2211b{4}(q) W2211b{5}(q)...
    W2211b{6}(q) W2211b{7}(q) W2211b{8}(q) W2211b{9}(q)] = ...
    Bias_Calc(res_step,W2211,W,10);
```

```
[W2221b{1}(:,:,q) W2221b{2}(q) W2221b{3}(q) W2221b{4}(q) W2221b{5}(q)...
    W2221b{6}(q) W2221b{7}(q) W2221b{8}(q) W2221b{9}(q)] = ...
    Bias_Calc(res_step,W2221,W,10);
[W2231b{1}(:,:,q) W2231b{2}(q) W2231b{3}(q) W2231b{4}(q) W2231b{5}(q)...
    W2231b{6}(q) W2231b{7}(q) W2231b{8}(q) W2231b{9}(q)] = ...
    Bias_Calc(res_step,W2231,W,10);

[W3211b{1}(:,:,q) W3211b{2}(q) W3211b{3}(q) W3211b{4}(q) W3211b{5}(q)...
    W3211b{6}(q) W3211b{7}(q) W3211b{8}(q) W3211b{9}(q)] = ...
    Bias_Calc(res_step,W3211,W,10);
[W3221b{1}(:,:,q) W3221b{2}(q) W3221b{3}(q) W3221b{4}(q) W3221b{5}(q)...
    W3221b{6}(q) W3221b{7}(q) W3221b{8}(q) W3221b{9}(q)] = ...
    Bias_Calc(res_step,W3221,W,10);
[W3231b{1}(:,:,q) W3231b{2}(q) W3231b{3}(q) W3231b{4}(q) W3231b{5}(q)...
    W3231b{6}(q) W3231b{7}(q) W3231b{8}(q) W3231b{9}(q)] = ...
    Bias_Calc(res_step,W3231,W,10);
[W3311b{1}(:,:,q) W3311b{2}(q) W3311b{3}(q) W3311b{4}(q) W3311b{5}(q)...
    W3311b{6}(q) W3311b{7}(q) W3311b{8}(q) W3311b{9}(q)] = ...
    Bias_Calc(res_step,W3311,W,10);
[W3321b{1}(:,:,q) W3321b{2}(q) W3321b{3}(q) W3321b{4}(q) W3321b{5}(q)...
    W3321b{6}(q) W3321b{7}(q) W3321b{8}(q) W3321b{9}(q)] = ...
    Bias_Calc(res_step,W3321,W,10);
[W3331b{1}(:,:,q) W3331b{2}(q) W3331b{3}(q) W3331b{4}(q) W3331b{5}(q)...
    W3331b{6}(q) W3331b{7}(q) W3331b{8}(q) W3331b{9}(q)] = ...
    Bias_Calc(res_step,W3331,W,10);

[W1212b{1}(:,:,q) W1212b{2}(q) W1212b{3}(q) W1212b{4}(q) W1212b{5}(q)...
    W1212b{6}(q) W1212b{7}(q) W1212b{8}(q) W1212b{9}(q)] = ...
    Bias_Calc(res_step,W1212,W,10);
[W1222b{1}(:,:,q) W1222b{2}(q) W1222b{3}(q) W1222b{4}(q) W1222b{5}(q)...
    W1222b{6}(q) W1222b{7}(q) W1222b{8}(q) W1222b{9}(q)] = ...
    Bias_Calc(res_step,W1222,W,10);
[W1232b{1}(:,:,q) W1232b{2}(q) W1232b{3}(q) W1232b{4}(q) W1232b{5}(q)...
    W1232b{6}(q) W1232b{7}(q) W1232b{8}(q) W1232b{9}(q)] = ...
    Bias_Calc(res_step,W1232,W,10);
[W1312b{1}(:,:,q) W1312b{2}(q) W1312b{3}(q) W1312b{4}(q) W1312b{5}(q)...
    W1312b{6}(q) W1312b{7}(q) W1312b{8}(q) W1312b{9}(q)] = ...
    Bias_Calc(res_step,W1312,W,10);
[W1322b{1}(:,:,q) W1322b{2}(q) W1322b{3}(q) W1322b{4}(q) W1322b{5}(q)...
    W1322b{6}(q) W1322b{7}(q) W1322b{8}(q) W1322b{9}(q)] = ...
    Bias_Calc(res_step,W1322,W,10);
[W1332b{1}(:,:,q) W1332b{2}(q) W1332b{3}(q) W1332b{4}(q) W1332b{5}(q)...
    W1332b{6}(q) W1332b{7}(q) W1332b{8}(q) W1332b{9}(q)] = ...
    Bias_Calc(res_step,W1332,W,10);
[W1412b{1}(:,:,q) W1412b{2}(q) W1412b{3}(q) W1412b{4}(q) W1412b{5}(q)...
    W1412b{6}(q) W1412b{7}(q) W1412b{8}(q) W1412b{9}(q)] = ...
    Bias_Calc(res_step,W1412,W,10);
[W1422b{1}(:,:,q) W1422b{2}(q) W1422b{3}(q) W1422b{4}(q) W1422b{5}(q)...
    W1422b{6}(q) W1422b{7}(q) W1422b{8}(q) W1422b{9}(q)] = ...
    Bias_Calc(res_step,W1422,W,10);
[W1432b{1}(:,:,q) W1432b{2}(q) W1432b{3}(q) W1432b{4}(q) W1432b{5}(q)...
    W1432b{6}(q) W1432b{7}(q) W1432b{8}(q) W1432b{9}(q)] = ...
    Bias_Calc(res_step,W1432,W,10);
[W1512b{1}(:,:,q) W1512b{2}(q) W1512b{3}(q) W1512b{4}(q) W1512b{5}(q)...
    W1512b{6}(q) W1512b{7}(q) W1512b{8}(q) W1512b{9}(q)] = ...
    Bias_Calc(res_step,W1512,W,10);
```

```
[W1522b{1}(:,:,q) W1522b{2}(q) W1522b{3}(q) W1522b{4}(q) W1522b{5}(q)...
    W1522b{6}(q) W1522b{7}(q) W1522b{8}(q) W1522b{9}(q)] = ...
    Bias_Calc(res_step,W1522,W,10);
[W1532b{1}(:,:,q) W1532b{2}(q) W1532b{3}(q) W1532b{4}(q) W1532b{5}(q)...
    W1532b{6}(q) W1532b{7}(q) W1532b{8}(q) W1532b{9}(q)] = ...
    Bias_Calc(res_step,W1532,W,10);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Direction Bias Calculations   %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[I1111angb{1}(:,:,q) I1111angb{2}(q) I1111angb{3}(q) I1111angb{4}(q)...
    I1111angb{5}(q) I1111angb{6}(q) I1111angb{7}(q) I1111angb{8}(q)...
    I1111angb{9}(q)] = Bias_Calc(res_step,I1111ang(:,:,q));
[I1121angb{1}(:,:,q) I1121angb{2}(q) I1121angb{3}(q) I1121angb{4}(q)...
    I1121angb{5}(q) I1121angb{6}(q) I1121angb{7}(q) I1121angb{8}(q)...
    I1121angb{9}(q)] = Bias_Calc(res_step,I1121ang(:,:,q));
[I1131angb{1}(:,:,q) I1131angb{2}(q) I1131angb{3}(q) I1131angb{4}(q)...
    I1131angb{5}(q) I1131angb{6}(q) I1131angb{7}(q) I1131angb{8}(q)...
    I1131angb{9}(q)] = Bias_Calc(res_step,I1131ang(:,:,q));
[I1211angb{1}(:,:,q) I1211angb{2}(q) I1211angb{3}(q) I1211angb{4}(q)...
    I1211angb{5}(q) I1211angb{6}(q) I1211angb{7}(q) I1211angb{8}(q)...
    I1211angb{9}(q)] = Bias_Calc(res_step,I1211ang(:,:,q));
[I1221angb{1}(:,:,q) I1221angb{2}(q) I1221angb{3}(q) I1221angb{4}(q)...
    I1221angb{5}(q) I1221angb{6}(q) I1221angb{7}(q) I1221angb{8}(q)...
    I1221angb{9}(q)] = Bias_Calc(res_step,I1221ang(:,:,q));
[I1231angb{1}(:,:,q) I1231angb{2}(q) I1231angb{3}(q) I1231angb{4}(q)...
    I1231angb{5}(q) I1231angb{6}(q) I1231angb{7}(q) I1231angb{8}(q)...
    I1231angb{9}(q)] = Bias_Calc(res_step,I1231ang(:,:,q));
[I1311angb{1}(:,:,q) I1311angb{2}(q) I1311angb{3}(q) I1311angb{4}(q)...
    I1311angb{5}(q) I1311angb{6}(q) I1311angb{7}(q) I1311angb{8}(q)...
    I1311angb{9}(q)] = Bias_Calc(res_step,I1311ang(:,:,q));
[I1321angb{1}(:,:,q) I1321angb{2}(q) I1321angb{3}(q) I1321angb{4}(q)...
    I1321angb{5}(q) I1321angb{6}(q) I1321angb{7}(q) I1321angb{8}(q)...
    I1321angb{9}(q)] = Bias_Calc(res_step,I1321ang(:,:,q));
[I1331angb{1}(:,:,q) I1331angb{2}(q) I1331angb{3}(q) I1331angb{4}(q)...
    I1331angb{5}(q) I1331angb{6}(q) I1331angb{7}(q) I1331angb{8}(q)...
    I1331angb{9}(q)] = Bias_Calc(res_step,I1331ang(:,:,q));
[I1411angb{1}(:,:,q) I1411angb{2}(q) I1411angb{3}(q) I1411angb{4}(q)...
    I1411angb{5}(q) I1411angb{6}(q) I1411angb{7}(q) I1411angb{8}(q)...
    I1411angb{9}(q)] = Bias_Calc(res_step,I1411ang(:,:,q));
[I1421angb{1}(:,:,q) I1421angb{2}(q) I1421angb{3}(q) I1421angb{4}(q)...
    I1421angb{5}(q) I1421angb{6}(q) I1421angb{7}(q) I1421angb{8}(q)...
    I1421angb{9}(q)] = Bias_Calc(res_step,I1421ang(:,:,q));
[I1431angb{1}(:,:,q) I1431angb{2}(q) I1431angb{3}(q) I1431angb{4}(q)...
    I1431angb{5}(q) I1431angb{6}(q) I1431angb{7}(q) I1431angb{8}(q)...
    I1431angb{9}(q)] = Bias_Calc(res_step,I1431ang(:,:,q));
[I1511angb{1}(:,:,q) I1511angb{2}(q) I1511angb{3}(q) I1511angb{4}(q)...
    I1511angb{5}(q) I1511angb{6}(q) I1511angb{7}(q) I1511angb{8}(q)...
    I1511angb{9}(q)] = Bias_Calc(res_step,I1511ang(:,:,q));
[I1521angb{1}(:,:,q) I1521angb{2}(q) I1521angb{3}(q) I1521angb{4}(q)...
    I1521angb{5}(q) I1521angb{6}(q) I1521angb{7}(q) I1521angb{8}(q)...
    I1521angb{9}(q)] = Bias_Calc(res_step,I1521ang(:,:,q));
```

```
[I1531angb{1}(:,:,q) I1531angb{2}(q) I1531angb{3}(q) I1531angb{4}(q)...
    I1531angb{5}(q) I1531angb{6}(q) I1531angb{7}(q) I1531angb{8}(q)...
    I1531angb{9}(q)] = Bias_Calc(res_step,I1531ang(:,:,q));

[I2111angb{1}(:,:,q) I2111angb{2}(q) I2111angb{3}(q) I2111angb{4}(q)...
    I2111angb{5}(q) I2111angb{6}(q) I2111angb{7}(q) I2111angb{8}(q)...
    I2111angb{9}(q)] = Bias_Calc(res_step,I2111ang(:,:,q));
[I2121angb{1}(:,:,q) I2121angb{2}(q) I2121angb{3}(q) I2121angb{4}(q)...
    I2121angb{5}(q) I2121angb{6}(q) I2121angb{7}(q) I2121angb{8}(q)...
    I2121angb{9}(q)] = Bias_Calc(res_step,I2121ang(:,:,q));
[I2131angb{1}(:,:,q) I2131angb{2}(q) I2131angb{3}(q) I2131angb{4}(q)...
    I2131angb{5}(q) I2131angb{6}(q) I2131angb{7}(q) I2131angb{8}(q)...
    I2131angb{9}(q)] = Bias_Calc(res_step,I2131ang(:,:,q));
[I2211angb{1}(:,:,q) I2211angb{2}(q) I2211angb{3}(q) I2211angb{4}(q)...
    I2211angb{5}(q) I2211angb{6}(q) I2211angb{7}(q) I2211angb{8}(q)...
    I2211angb{9}(q)] = Bias_Calc(res_step,I2211ang(:,:,q));
[I2221angb{1}(:,:,q) I2221angb{2}(q) I2221angb{3}(q) I2221angb{4}(q)...
    I2221angb{5}(q) I2221angb{6}(q) I2221angb{7}(q) I2221angb{8}(q)...
    I2221angb{9}(q)] = Bias_Calc(res_step,I2221ang(:,:,q));
[I2231angb{1}(:,:,q) I2231angb{2}(q) I2231angb{3}(q) I2231angb{4}(q)...
    I2231angb{5}(q) I2231angb{6}(q) I2231angb{7}(q) I2231angb{8}(q)...
    I2231angb{9}(q)] = Bias_Calc(res_step,I2231ang(:,:,q));

[I3111angb{1}(:,:,q) I3111angb{2}(q) I3111angb{3}(q) I3111angb{4}(q)...
    I3111angb{5}(q) I3111angb{6}(q) I3111angb{7}(q) I3111angb{8}(q)...
    I3111angb{9}(q)] = Bias_Calc(res_step,I3111ang(:,:,q));
[I3121angb{1}(:,:,q) I3121angb{2}(q) I3121angb{3}(q) I3121angb{4}(q)...
    I3121angb{5}(q) I3121angb{6}(q) I3121angb{7}(q) I3121angb{8}(q)...
    I3121angb{9}(q)] = Bias_Calc(res_step,I3121ang(:,:,q));
[I3131angb{1}(:,:,q) I3131angb{2}(q) I3131angb{3}(q) I3131angb{4}(q)...
    I3131angb{5}(q) I3131angb{6}(q) I3131angb{7}(q) I3131angb{8}(q)...
    I3131angb{9}(q)] = Bias_Calc(res_step,I3131ang(:,:,q));
[I3211angb{1}(:,:,q) I3211angb{2}(q) I3211angb{3}(q) I3211angb{4}(q)...
    I3211angb{5}(q) I3211angb{6}(q) I3211angb{7}(q) I3211angb{8}(q)...
    I3211angb{9}(q)] = Bias_Calc(res_step,I3211ang(:,:,q));
[I3221angb{1}(:,:,q) I3221angb{2}(q) I3221angb{3}(q) I3221angb{4}(q)...
    I3221angb{5}(q) I3221angb{6}(q) I3221angb{7}(q) I3221angb{8}(q)...
    I3221angb{9}(q)] = Bias_Calc(res_step,I3221ang(:,:,q));
[I3231angb{1}(:,:,q) I3231angb{2}(q) I3231angb{3}(q) I3231angb{4}(q)...
    I3231angb{5}(q) I3231angb{6}(q) I3231angb{7}(q) I3231angb{8}(q)...
    I3231angb{9}(q)] = Bias_Calc(res_step,I3231ang(:,:,q));
[I3311angb{1}(:,:,q) I3311angb{2}(q) I3311angb{3}(q) I3311angb{4}(q)...
    I3311angb{5}(q) I3311angb{6}(q) I3311angb{7}(q) I3311angb{8}(q)...
    I3311angb{9}(q)] = Bias_Calc(res_step,I3311ang(:,:,q));
[I3321angb{1}(:,:,q) I3321angb{2}(q) I3321angb{3}(q) I3321angb{4}(q)...
    I3321angb{5}(q) I3321angb{6}(q) I3321angb{7}(q) I3321angb{8}(q)...
    I3321angb{9}(q)] = Bias_Calc(res_step,I3321ang(:,:,q));
[I3331angb{1}(:,:,q) I3331angb{2}(q) I3331angb{3}(q) I3331angb{4}(q)...
    I3331angb{5}(q) I3331angb{6}(q) I3331angb{7}(q) I3331angb{8}(q)...
    I3331angb{9}(q)] = Bias_Calc(res_step,I3331ang(:,:,q));
[I3411angb{1}(:,:,q) I3411angb{2}(q) I3411angb{3}(q) I3411angb{4}(q)...
    I3411angb{5}(q) I3411angb{6}(q) I3411angb{7}(q) I3411angb{8}(q)...
    I3411angb{9}(q)] = Bias_Calc(res_step,I3411ang(:,:,q));
[I3421angb{1}(:,:,q) I3421angb{2}(q) I3421angb{3}(q) I3421angb{4}(q)...
    I3421angb{5}(q) I3421angb{6}(q) I3421angb{7}(q) I3421angb{8}(q)...
    I3421angb{9}(q)] = Bias_Calc(res_step,I3421ang(:,:,q));
```

```
[I3431angb{1}(:,:,q) I3431angb{2}(q) I3431angb{3}(q) I3431angb{4}(q)...
    I3431angb{5}(q) I3431angb{6}(q) I3431angb{7}(q) I3431angb{8}(q)...
    I3431angb{9}(q)] = Bias_Calc(res_step,I3431ang(:,:,q));

[I1112angb{1}(:,:,q) I1112angb{2}(q) I1112angb{3}(q) I1112angb{4}(q)...
    I1112angb{5}(q) I1112angb{6}(q) I1112angb{7}(q) I1112angb{8}(q)...
    I1112angb{9}(q)] = Bias_Calc(res_step,I1112ang(:,:,q));
[I1122angb{1}(:,:,q) I1122angb{2}(q) I1122angb{3}(q) I1122angb{4}(q)...
    I1122angb{5}(q) I1122angb{6}(q) I1122angb{7}(q) I1122angb{8}(q)...
    I1122angb{9}(q)] = Bias_Calc(res_step,I1122ang(:,:,q));
[I1132angb{1}(:,:,q) I1132angb{2}(q) I1132angb{3}(q) I1132angb{4}(q)...
    I1132angb{5}(q) I1132angb{6}(q) I1132angb{7}(q) I1132angb{8}(q)...
    I1132angb{9}(q)] = Bias_Calc(res_step,I1132ang(:,:,q));
[I1212angb{1}(:,:,q) I1212angb{2}(q) I1212angb{3}(q) I1212angb{4}(q)...
    I1212angb{5}(q) I1212angb{6}(q) I1212angb{7}(q) I1212angb{8}(q)...
    I1212angb{9}(q)] = Bias_Calc(res_step,I1212ang(:,:,q));
[I1222angb{1}(:,:,q) I1222angb{2}(q) I1222angb{3}(q) I1222angb{4}(q)...
    I1222angb{5}(q) I1222angb{6}(q) I1222angb{7}(q) I1222angb{8}(q)...
    I1222angb{9}(q)] = Bias_Calc(res_step,I1222ang(:,:,q));
[I1232angb{1}(:,:,q) I1232angb{2}(q) I1232angb{3}(q) I1232angb{4}(q)...
    I1232angb{5}(q) I1232angb{6}(q) I1232angb{7}(q) I1232angb{8}(q)...
    I1232angb{9}(q)] = Bias_Calc(res_step,I1232ang(:,:,q));
[I1312angb{1}(:,:,q) I1312angb{2}(q) I1312angb{3}(q) I1312angb{4}(q)...
    I1312angb{5}(q) I1312angb{6}(q) I1312angb{7}(q) I1312angb{8}(q)...
    I1312angb{9}(q)] = Bias_Calc(res_step,I1312ang(:,:,q));
[I1322angb{1}(:,:,q) I1322angb{2}(q) I1322angb{3}(q) I1322angb{4}(q)...
    I1322angb{5}(q) I1322angb{6}(q) I1322angb{7}(q) I1322angb{8}(q)...
    I1322angb{9}(q)] = Bias_Calc(res_step,I1322ang(:,:,q));
[I1332angb{1}(:,:,q) I1332angb{2}(q) I1332angb{3}(q) I1332angb{4}(q)...
    I1332angb{5}(q) I1332angb{6}(q) I1332angb{7}(q) I1332angb{8}(q)...
    I1332angb{9}(q)] = Bias_Calc(res_step,I1332ang(:,:,q));
[I1412angb{1}(:,:,q) I1412angb{2}(q) I1412angb{3}(q) I1412angb{4}(q)...
    I1412angb{5}(q) I1412angb{6}(q) I1412angb{7}(q) I1412angb{8}(q)...
    I1412angb{9}(q)] = Bias_Calc(res_step,I1412ang(:,:,q));
[I1422angb{1}(:,:,q) I1422angb{2}(q) I1422angb{3}(q) I1422angb{4}(q)...
    I1422angb{5}(q) I1422angb{6}(q) I1422angb{7}(q) I1422angb{8}(q)...
    I1422angb{9}(q)] = Bias_Calc(res_step,I1422ang(:,:,q));
[I1432angb{1}(:,:,q) I1432angb{2}(q) I1432angb{3}(q) I1432angb{4}(q)...
    I1432angb{5}(q) I1432angb{6}(q) I1432angb{7}(q) I1432angb{8}(q)...
    I1432angb{9}(q)] = Bias_Calc(res_step,I1432ang(:,:,q));
[I1512angb{1}(:,:,q) I1512angb{2}(q) I1512angb{3}(q) I1512angb{4}(q)...
    I1512angb{5}(q) I1512angb{6}(q) I1512angb{7}(q) I1512angb{8}(q)...
    I1512angb{9}(q)] = Bias_Calc(res_step,I1512ang(:,:,q));
[I1522angb{1}(:,:,q) I1522angb{2}(q) I1522angb{3}(q) I1522angb{4}(q)...
    I1522angb{5}(q) I1522angb{6}(q) I1522angb{7}(q) I1522angb{8}(q)...
    I1522angb{9}(q)] = Bias_Calc(res_step,I1522ang(:,:,q));
[I1532angb{1}(:,:,q) I1532angb{2}(q) I1532angb{3}(q) I1532angb{4}(q)...
    I1532angb{5}(q) I1532angb{6}(q) I1532angb{7}(q) I1532angb{8}(q)...
    I1532angb{9}(q)] = Bias_Calc(res_step,I1532ang(:,:,q));

[v111angb{1}(:,:,q) v111angb{2}(q) v111angb{3}(q) v111angb{4}(q)...
    v111angb{5}(q) v111angb{6}(q) v111angb{7}(q) v111angb{8}(q)...
    v111angb{9}(q)] = Bias_Calc(res_step,v111ang(:,:,q));
[v121angb{1}(:,:,q) v121angb{2}(q) v121angb{3}(q) v121angb{4}(q)...
    v121angb{5}(q) v121angb{6}(q) v121angb{7}(q) v121angb{8}(q)...
    v121angb{9}(q)] = Bias_Calc(res_step,v121ang(:,:,q));
```

```
    [v131angb{1}(:,:,q) v131angb{2}(q) v131angb{3}(q) v131angb{4}(q)...
        v131angb{5}(q) v131angb{6}(q) v131angb{7}(q) v131angb{8}(q)...
        v131angb{9}(q)] = Bias_Calc(res_step,v131ang(:,:,q));

    [v211angb{1}(:,:,q) v211angb{2}(q) v211angb{3}(q) v211angb{4}(q)...
        v211angb{5}(q) v211angb{6}(q) v211angb{7}(q) v211angb{8}(q)...
        v211angb{9}(q)] = Bias_Calc(res_step,v211ang(:,:,q));
    [v221angb{1}(:,:,q) v221angb{2}(q) v221angb{3}(q) v221angb{4}(q)...
        v221angb{5}(q) v221angb{6}(q) v221angb{7}(q) v221angb{8}(q)...
        v221angb{9}(q)] = Bias_Calc(res_step,v221ang(:,:,q));
    [v231angb{1}(:,:,q) v231angb{2}(q) v231angb{3}(q) v231angb{4}(q)...
        v231angb{5}(q) v231angb{6}(q) v231angb{7}(q) v231angb{8}(q)...
        v231angb{9}(q)] = Bias_Calc(res_step,v231ang(:,:,q));

    [v311angb{1}(:,:,q) v311angb{2}(q) v311angb{3}(q) v311angb{4}(q)...
        v311angb{5}(q) v311angb{6}(q) v311angb{7}(q) v311angb{8}(q)...
        v311angb{9}(q)] = Bias_Calc(res_step,v311ang(:,:,q));
    [v321angb{1}(:,:,q) v321angb{2}(q) v321angb{3}(q) v321angb{4}(q)...
        v321angb{5}(q) v321angb{6}(q) v321angb{7}(q) v321angb{8}(q)...
        v321angb{9}(q)] = Bias_Calc(res_step,v321ang(:,:,q));
    [v331angb{1}(:,:,q) v331angb{2}(q) v331angb{3}(q) v331angb{4}(q)...
        v331angb{5}(q) v331angb{6}(q) v331angb{7}(q) v331angb{8}(q)...
        v331angb{9}(q)] = Bias_Calc(res_step,v331ang(:,:,q));

    [v112angb{1}(:,:,q) v112angb{2}(q) v112angb{3}(q) v112angb{4}(q)...
        v112angb{5}(q) v112angb{6}(q) v112angb{7}(q) v112angb{8}(q)...
        v112angb{9}(q)] = Bias_Calc(res_step,v112ang(:,:,q));
    [v122angb{1}(:,:,q) v122angb{2}(q) v122angb{3}(q) v122angb{4}(q)...
        v122angb{5}(q) v122angb{6}(q) v122angb{7}(q) v122angb{8}(q)...
        v122angb{9}(q)] = Bias_Calc(res_step,v122ang(:,:,q));
    [v132angb{1}(:,:,q) v132angb{2}(q) v132angb{3}(q) v132angb{4}(q)...
        v132angb{5}(q) v132angb{6}(q) v132angb{7}(q) v132angb{8}(q)...
        v132angb{9}(q)] = Bias_Calc(res_step,v132ang(:,:,q));

    % Set up meshgrid for plotting.
    [PHI,THETA]=meshgrid(phi,theta);
    X = cos(PHI).*sin(THETA);
    Y = sin(PHI).*sin(THETA);
    Z = cos(THETA);

    q = q+1;
    toc
end

ka = 2*pi*freq*a/c;
set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',20);
set(0,'DefaultAxesFontWeight','demi')
set(0,'DefaultAxesLineWidth',2);
set(0,'DefaultLineLineWidth',2);
set(0,'DefaultLineMarkersize',8);
toc
save Multimicrophone_Probe_Analysis.mat
```

B.1.1 Angle_Calc.m

```
function ang = Angle_Calc(vect,x,y,z)

vect1 = [x y z];
ang = abs(acos(dot(vect,vect1)/(norm(vect)*norm(vect1))))*180/pi;

end
```

B.1.2 Bias_Calc.m

```
function [b bmax bmin berrmax berrmin bstd bavg berrstd berravg] = ...
    Bias_Calc(res_step,X2,X,val)

% If statements determine type of acoustic quantity and do appropriate bias
% calculation.
if (nargin == 4) && (val == 10)
    b = 10*log10(X2./X);
elseif (nargin == 4) && (val == 20)
    b = 20*log10(X2./X);
elseif nargin < 4
    b = X2;
end

% Max and min.
bmax = max(max(b));
bmin = min(min(b));

% Absolute value max and min.
berrmax = max(max(abs(b)));
berrmin = min(min(abs(b)));

% Weighting function needed to calculate average error.
thetavect = [0,(0:res_step:pi)+res_step/2];
thetavect(end) = pi;
for i=1:size(X2,1)
    btemp(i,:) = b(i,:)*(cos(thetavect(i))-cos(thetavect(i+1)));
    weights(i) = (cos(thetavect(i))-cos(thetavect(i+1)));
end
berravg = sum(sum(abs(btemp)))/sum(weights)/size(X2,2);
bavg = sum(sum(btemp))/sum(weights)/size(X2,2);

% Standard deviation calculations.
% These expressions need to be corrected!
% Need to do weighted standard deviation.
bstd = std(reshape(btemp,1,[]));
% bavg = sum(sum(btemp))/sum(weights)/size(X2,2);
berrstd = std(reshape(abs(btemp),1,[]));
% berrstd = sum(sum(weights.*(abs(btemp)-berravg).^2))/sum(weights);
end
```

B.1.3 Intensity_Calc.m

```
function [Ix Iy Iz I] = Intensity_Calc(p,vx,vy,vz)

Ix = 0.5*real(p*conj(vx));
Iy = 0.5*real(p*conj(vy));
Iz = 0.5*real(p*conj(vz));
I = (abs(Ix).^2+abs(Iy).^2+abs(Iz).^2).^(1/2);

end
```

B.1.4 Taylor_Velocity.m

```
% Code from Lance Locey thesis with slight modifications to work with
% "Multimicrophone_Probe_Analysis.m"

function [vx vy vz v] = Taylor_Velocity(p1,p2,p3,p4,w,rho,a,factor)

Ux13=(p1-p3)/1i/w/rho/(2*a*factor/sqrt(3));
Uy23=(p2-p3)/1i/w/rho/(2*a*factor/sqrt(3));
Uz43=(p4-p3)/1i/w/rho/(2*a*factor/sqrt(3));
Ux31=-Ux13;
Uy32=-Uy23;
Uz34=-Uz43;

U12=(p1-p2)/1i/w/rho/(sqrt(8)*a*factor/sqrt(3));
U41=(p4-p1)/1i/w/rho/(sqrt(8)*a*factor/sqrt(3));
U14=-U41;
U24=(p2-p4)/1i/w/rho/(sqrt(8)*a*factor/sqrt(3));

% determine the phase angle of the on axis velocities
Ux31pha=angle(Ux31);
Uy32pha=angle(Uy32);
Uz34pha=angle(Uz34);


% XY plane

% difference in phase of Ux and Uy
phadiffxy=180/pi*(Ux31pha-Uy32pha);
theta=atan2(abs(Uy32),abs(Ux31));

% if the phase difference is between 90 and 270 degrees, change the sign of
% theta
if ( abs(phadiffxy) > 90 && abs(phadiffxy) < 270)
    theta=-theta;
end
```

```
% thetaa is the angle as measured from the diagonal. This gives the correct
% sign on the diagonal
thetaa=3*pi/4-theta;
U12p=U12./cos(thetaa); % this line may not need the ./
Uy12=U12p.*sin(theta);
Ux12=U12p.*cos(theta);


% XZ Plane

% difference in phase of Ux and Uz
phadiffxz=180/pi*(Ux31pha-Uz34pha); % difference in phase of Ux and Uz
betaxz=atan2(abs(Uz34),abs(Ux31)); % Ux31 points in the positive x direction,
Uz34 points in the positive z direction

% if the phase difference is between 90 and 270 degrees, change the sign of
% betaxz
if ( abs(phadiffxz) > 90 && abs(phadiffxz) < 270)
    betaxz=-betaxz;
end
betaa=3*pi/4-betaxz;
U14p=U14./cos(betaa);
Ux14=U14p.*cos(betaxz);
Uz14=U14p.*sin(betaxz);


% YZ Plane

% difference in phase of Uy and Uz
phadiffyz=180/pi*(Uy32pha-Uz34pha);
alphayz=atan2(abs(Uz34),abs(Uy32));

% if the phase difference is between 90 and 270 degrees, change the sign of
% alphayz
if ( abs(phadiffyz) > 90 && abs(phadiffyz) < 270)
    alphayz=-alphayz;
end
alphaa=3*pi/4-alphayz;
U24p=U24./cos(alphaa);
Uz24=U24p.*sin(alphayz);
Uy24=U24p.*cos(alphayz);

%Taylor Series expansion
Ux=Ux12-Ux31+Ux14;
Uy=Uy12-Uy32+Uy24;
Uz=Uz14-Uz34+Uz24;

vx = Ux;
vy = Uy;
vz = Uz;

v = (abs(vx).^2+abs(vy).^2+abs(vz).^2).^(1/2);

end
```

B.2 Chapter_2_Figures.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Chapter_2_Figures.m                                                    %
% Date: 08 August 2011                                                         %
% Author: Curtis Wiederhold                                                    %
%                                                                              %
% Description: This Matlab script plots Figs. 2-1, 2-2, 2-5, 2-6, 2-7    %
% 2-8, 2-9, and 2-10 of my thesis. For Figs. 2-5 and 2-7 through 2-10,   %
% the data output by "Multimicrophone_Probe_Analysis.m" must be loaded.  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Figure 2-1

clear all;
close all;

figure

axes('position',[.3 -.23 .8 1.5])
[x y z] = sphere(60);
patch(surf2patch(x,y,z,z),'EdgeColor',[.1 .1 .1],'FaceColor',[.7 .7
.7],'LineWidth',.4);
view(3)
axis vis3d
axis off
alpha(0.5)
view([41 30])
hold on
plot3(1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',1)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),1/sqrt(3),50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',1)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),...
    1/sqrt(3),50),'-k','LineWidth',1)
text(.5,.6,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

axes('position',[-.1 -.23 .8 1.5])
sphere
hold on
alpha(0)
shading interp
plot3(1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ok','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
```

```
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',11,'MarkerEdgeColor','none')
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',1)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),1/sqrt(3),50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',1)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),...
    1/sqrt(3),50),'-k','LineWidth',1)
axis vis3d
view([41 30])
axis off
text(1.2,.6,'x','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')
text(1.06,1.44,'y','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')
text(.45,1.8,'z','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')
text(.5,.6,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

text(1.37,.8,'2','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(1.26,1.58,'3','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(.52,2.13,'4','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(.38,1,'1','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 2.1]);
print -dtiffn -r1600 Figure1


%% Figure 2-2

clear all;
close all;

figure
axes('position',[-.3 -.4 2 2])
sphere
hold on
alpha(0)
shading interp

plot3(linspace(-1/sqrt(3),.8,50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),.8,50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),.8,50),'-k','LineWidth',.5)
```

```
annotation('arrow',[0.46 0.65],[0.192 0.12],'Color','r','LineWidth',1.5)
plot3(0,-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','k','MarkerSize',5,...
    'MarkerEdgeColor','none')

plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    linspace(1/sqrt(3),-1/sqrt(3),50),'--k','LineWidth',.5)
annotation('arrow',[0.458 0.65],[0.477 0.405],'Color','r','LineWidth',1.5)
plot3(0,-1/sqrt(3),0,'ob','MarkerFaceColor','k','MarkerSize',5,...
    'MarkerEdgeColor','none')

plot3(linspace(-1/sqrt(3),1/sqrt(3),50),linspace(1/sqrt(3),...
    -1/sqrt(3),50),-1/sqrt(3).*ones(50),'--k','LineWidth',.5)
annotation('arrow',[0.702 0.915],[0.315 0.233],'Color','r','LineWidth',1.5)
plot3(0,0,-1/sqrt(3),'ob','MarkerFaceColor','k','MarkerSize',5,...
    'MarkerEdgeColor','none')

plot3(1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',14,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',14,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',14,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',14,'MarkerEdgeColor','none')

axis vis3d
view([41 30])
axis off

text(2.35,.88,'x','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')
text(2.17,2.48,'y','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')
text(.8,3.35,'z','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal','FontAngle','Italic')

text(2.35,1.15,'2','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(1.87,2.37,'3','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(1.05,3.15,'4','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')
text(.67,1.6,'1','Units','inches','FontSize',12,'FontName','Times',...
    'FontWeight','normal')


set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 2.4 2.7]);
print -dtiffn -r1600 Figure2


%% Figure 2-5
```

```
set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75; .42 .25 .4;...
    0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


cmin = -10;
cmax = 10;
maxval = 1;
CMRmap=[0 0 0;.15 .15 .5;.3 .15 .75;.6 .2 .50;1 .25 .15;.9 .5 0;...
    .9 .75 .1;.9 .9 .5;1 1 1];
xtemp = 1:8/63:9;
xtemp1 = 1:9;
for i = 1:3
    sCMRmap(:,i) = spline(xtemp1,CMRmap(:,i),xtemp)';
end
sCMRmap = abs(sCMRmap/max(max(sCMRmap)));
phot = sCMRmap;

figure
for n=46
% for n=1:length(freq)
    clf
    colormap(phot);
    n
    axes('position',[-.02 0 .3 1])
    surf(X,Y,Z,I1221b{1}(:,:,n),'Edgecolor','none')
    shading interp
    axis ([-maxval maxval -maxval maxval -maxval maxval])
    axis vis3d
    axis off
    colorbar('position',[.27 .2 .03 .6],'YTick',[-2 -1.5 -1 -.5 0],...
        'YTickLabel',{'-2','-1.5','-1','-0.5','0 dB'})
    text(.1,0,'(a)','Units','inches','FontSize',14,'FontName','Times',...
        'FontWeight','demi')
%     caxis([cmin cmax])
%     text(text1_x,text1_y,text1_z,'1')
%     text(text2_x,text2_y,text2_z,'2')
%     text(text3_x,text3_y,text3_z,'3')
%     text(text4_x,text4_y,text4_z,'4')
%     annotation('arrow',[0.42 0.57],[0.5 0.5],'Color','k',...
%         'LineWidth',1,'HeadStyle','Plain','MarkerEdgeColor','none')

    axes('position',[.54 0 .3 1])
    surf(X,Y,Z,abs(I1221b{1}(:,:,n)),...
        'Edgecolor','none')
    shading interp
    axis ([-maxval maxval -maxval maxval -maxval maxval])
    axis vis3d
```

```
    axis off
    colorbar('position',[.83 .2 .03 .6],'YTick',[0.01 0.5 1 1.5 2],...
        'YTickLabel',{'0','0.5','1','1.5','2 dB'})
    text(.1,0,'(b)','Units','inches','FontSize',14,'FontName','Times',...
        'FontWeight','demi')
    %     caxis([cmin cmax])
%     text(text1_x,text1_y,text1_z,'1')
%     text(text2_x,text2_y,text2_z,'2')
%     text(text3_x,text3_y,text3_z,'3')
%     text(text4_x,text4_y,text4_z,'4')

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [2 1 3+3/8 2]);
    % set(gcf, 'Position', [200 100 450 550]);
    % set(gcf,'PaperPositionMode','auto ')
    print -dtiffn -r1600 Figure5

    pause
end


%% Figure 2-6

close all;
clear all;

figure
axes('position',[-.25 -.25 1.5 1.5])
sphere(50)
axis vis3d
axis off
% shading interp
colormap gray
alpha(1)
view([41 30])

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 1.5 1.5]);
print -dtiffn -r1600 Figure6


%% Figure 2-7

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);
```

```
xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
ytickvect = [0 2 4 6];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx = 'ka';
ka = 2*pi*freq*a/c;
% ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(ka,I1211b{9},'-',ka,I1511b{9},'--',...
    ka,I1311b{9},'-.',ka,I1411b{9},':')
% xlabel('ka')
ylabel(labely1)
legend('1','3','4','4W','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I1212b{9},'-',ka,I1512b{9},'--',...
    ka,I1312b{9},'-.',ka,I1412b{9},':')
hold on
I=1:3:size(I1112b{9});
test = [.42 .25 .4];
plot(ka,I1112b{9},'.-','Visible','off','LineWidth',.7,'MarkerSize',8,...
    'MarkerEdgeColor',test,'Color',test);
plot(ka,I1112b{9},'-',ka(I),I1112b{9}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
legend('1.T','3.T','4.T','4W.T','3 1D.T','Location','NorthWest')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
```

```
plot(ka,I1211b{4},'-',ka,I1511b{4},'--',...
    ka,I1311b{4},'-.',ka,I1411b{4},':')
xlabel(labelx)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I1212b{4},'-',ka,I1512b{4},'--',...
    ka,I1312b{4},'-.',ka,I1412b{4},':')
hold on
plot(ka,I1112b{4},'.-','Visible','off','LineWidth',.7,'MarkerSize',8,...
    'MarkerEdgeColor',test,'Color',test);
plot(ka,I1112b{4},'-',ka(I),I1112b{4}(I),'.','LineWidth',.7,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);
% set(gcf, 'Position', [200 100 450 550]);
% set(gcf,'PaperPositionMode','auto')
print -dtiffn -r200 Figure7

%% Figure 2-8

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
```

```
ytickvect = [0 2 4 6];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
labelx = '3/2 ka';
% labelx = 'kb';
% ka = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(ka,I1221b{9},'-',ka,I1521b{9},'--',...
    ka,I1321b{9},'-.',ka,I1421b{9},':')
% xlabel('ka')
ylabel(labely1)
legend('S/1','S/3','S/4','S/4W','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I1222b{9},'-',ka,I1522b{9},'--',...
    ka,I1322b{9},'-.',ka,I1422b{9},':')
hold on
I=1:3:size(I1112b{9});
test = [.42 .25 .4];
plot(ka,I1122b{9},'.-','Visible','off','LineWidth',.7,'MarkerSize',...
    8,'MarkerEdgeColor',test,'Color',test);
plot(ka,I1122b{9},'-',ka(I),I1122b{9}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
legend('S/1.T','S/3.T','S/4.T','S/4W.T','S/3 1D.T','Location','NorthWest')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(ka,I1221b{4},'-',ka,I1521b{4},'--',...
    ka,I1321b{4},'-.',ka,I1421b{4},':')
```

```
xlabel(labelx)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I1222b{4},'-',ka,I1522b{4},'--',...
    ka,I1322b{4},'-.',ka,I1422b{4},':')
hold on
plot(ka,I1122b{4},'.-','Visible','off','LineWidth',.7,'MarkerSize',8,...
    'MarkerEdgeColor',test,'Color',test);
plot(ka,I1122b{4},'-',ka(I),I1122b{4}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);
% set(gcf, 'Position', [200 100 450 550]);
% set(gcf,'PaperPositionMode','auto')
print -dtiffn -r200 Figure8


%% Figure 2-9

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);



xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 15;
ytickvect = [0 4 8 12];
```

```
% labely = 'Error (dB)';
labely1 = 'Average Error (degrees)';
labely2 = 'Max Error (degrees)';
% labelx = 'ka';
labelx = 'ka';
ka = 2*pi*freq*a/c;
% ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(ka,I1211angb{9},'-',ka,I1511angb{9},'--',...
    ka,I1311angb{9},'-.',ka,I1411angb{9},':')
legend('1','3','4','4W','Location','NorthWest')
% xlabel('ka')
ylabel(labely1)
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I1212angb{9},'-',ka,I1512angb{9},'--',...
    ka,I1312angb{9},'-.',ka,I1412angb{9},':')
hold on
I=1:3:size(I1112b{9});
test = [.42 .25 .4];
plot(ka,I1112angb{9},'.-','Visible','off','LineWidth',.7,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
plot(ka,I1112angb{9},'-',ka(I),I1112angb{9}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
legend('1.T','3.T','4.T','4W.T','3 1D.T','Location','NorthWest')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(ka,I1211angb{4},'-',ka,I1511angb{4},'--',...
    ka,I1311angb{4},'-.',ka,I1411angb{4},':')
xlabel(labelx)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
```

```
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I1212angb{4},'-',ka,I1512angb{4},'--',...
    ka,I1312angb{4},'-.',ka,I1412angb{4},':')
hold on
plot(ka,I1112angb{4},'.-','Visible','off','LineWidth',.7,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
plot(ka,I1112angb{4},'-',ka(I),I1112angb{4}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);
% set(gcf, 'Position', [200 100 450 550]);
% set(gcf,'PaperPositionMode','auto')
print -dtiffn -r200 Figure9

%% Figure 2-10

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 15;
ytickvect = [0 4 8 12];
% labely = 'Error (dB)';
labely1 = 'Average Error (degrees)';
labely2 = 'Max Error (degrees)';
labelx = '3/2 ka';
```

```
% labelx = 'kb';
% ka = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(ka,I1221angb{9},'-',ka,I1521angb{9},'--',...
    ka,I1321angb{9},'-.',ka,I1421angb{9},':')
legend('S/1','S/3','S/4','S/4W','Location','NorthWest')
% xlabel('ka')
ylabel(labely1)
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I1222angb{9},'-',ka,I1522angb{9},'--',...
    ka,I1322angb{9},'-.',ka,I1422angb{9},':')
hold on
I=1:3:size(I1112b{9});
test = [.42 .25 .4];
plot(ka,I1122angb{9},'.-','Visible','off','LineWidth',.7,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
plot(ka,I1122angb{9},'-',ka(I),I1122angb{9}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
legend('S/1.T','S/3.T','S/4.T','S/4W.T','S/3 1D.T','Location','NorthWest')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(ka,I1221angb{4},'-',ka,I1521angb{4},'--',...
    ka,I1321angb{4},'-.',ka,I1421angb{4},':')
xlabel(labelx)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
```

```
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I1222angb{4},'-',ka,I1522angb{4},'--',...
    ka,I1322angb{4},'-.',ka,I1422angb{4},':')
hold on
plot(ka,I1122angb{4},'.-','Visible','off','LineWidth',.7,'MarkerSize',8,...
    'MarkerEdgeColor',test,'Color',test);
plot(ka,I1122angb{4},'-',ka(I),I1122angb{4}(I),'.','LineWidth',.71,...
    'MarkerSize',8,'MarkerEdgeColor',test,'Color',test);
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);
% set(gcf, 'Position', [200 100 450 550]);
% set(gcf,'PaperPositionMode','auto')
print -dtiffn -r200 Figure10
```

B.3 Chapter_3_Figures.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Chapter_3_Figures.m                                                    %
% Date: 08 August 2011                                                         %
% Author: Curtis Wiederhold                                                    %
%                                                                              %
% Description: This Matlab script plots all the figures in Chapter 3           %
% of my thesis. For Figs. 3-2 through 3-7, the data output by                  %
% "Multimicrophone_Probe_Analysis.m" must be loaded.                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Figure 3-1

clear all;
close all;

width = .81;
height = .51;

dotsize = 11;

figure

% Reg Tetra

axes('position',[.35 .6 width height])
[x y z] = sphere(60);
patch(surf2patch(x,y,z,z),'EdgeColor',[.1 .1 .1],'FaceColor',[.7 .7 .7],...
    'LineWidth',.4);
view([41 30])
axis vis3d
axis off
alpha(0.5)
hold on
plot3(0,0,1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,2*sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(-sqrt(6)/3,-sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b','MarkerSize',...
    dotsize,'MarkerEdgeColor','none')
plot3(sqrt(6)/3,-sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b','MarkerSize',...
    dotsize,'MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(0,2*sqrt(2)/3,50),...
    linspace(1,-1/3,50),'-k','LineWidth',.1)
plot3(linspace(0,-sqrt(6)/3,50),linspace(0,-sqrt(2)/3,50),...
    linspace(1,-1/3,50),'-k','LineWidth',.1)
plot3(linspace(0,sqrt(6)/3,50),linspace(0,-sqrt(2)/3,50),...
    linspace(1,-1/3,50),'-k','LineWidth',1)
plot3(linspace(0,-sqrt(6)/3,50),linspace(2*sqrt(2)/3,-sqrt(2)/3,50),...
    linspace(-1/3,-1/3,50),'-k','LineWidth',.1)
plot3(linspace(0,sqrt(6)/3,50),linspace(2*sqrt(2)/3,-sqrt(2)/3,50),...
    linspace(-1/3,-1/3,50),'-k','LineWidth',.1)
plot3(linspace(-sqrt(6)/3,sqrt(6)/3,50),linspace(-sqrt(2)/3,...
    -sqrt(2)/3,50),linspace(-1/3,-1/3,50),'-k','LineWidth',.1)
```

```
plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'-k',...
    'LineWidth',.5)
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
text(.3,.6,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

axes('position',[-.10 .6 width height])
sphere
hold on
plot3(0,0,1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,2*sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b','MarkerSize',...
    dotsize,'MarkerEdgeColor','none')
plot3(-sqrt(6)/3,-sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(sqrt(6)/3,-sqrt(2)/3,-1/3,'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(0,2*sqrt(2)/3,50),linspace(1,-1/3,50),...
    '--k','LineWidth',.1)
plot3(linspace(0,-sqrt(6)/3,50),linspace(0,-sqrt(2)/3,50),...
    linspace(1,-1/3,50),'--k','LineWidth',.1)
plot3(linspace(0,sqrt(6)/3,50),linspace(0,-sqrt(2)/3,50),...
    linspace(1,-1/3,50),'--k','LineWidth',.1)
plot3(linspace(0,-sqrt(6)/3,50),linspace(2*sqrt(2)/3,-sqrt(2)/3,50),...
    linspace(-1/3,-1/3,50),'--k','LineWidth',.1)
plot3(linspace(0,sqrt(6)/3,50),linspace(2*sqrt(2)/3,-sqrt(2)/3,50),...
    linspace(-1/3,-1/3,50),'--k','LineWidth',.1)
plot3(linspace(-sqrt(6)/3,sqrt(6)/3,50),linspace(-sqrt(2)/3,...
    -sqrt(2)/3,50),linspace(-1/3,-1/3,50),'--k','LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'-k',...
    'LineWidth',.5)
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
alpha(0)
shading interp
view([41 30])
axis vis3d
axis off
text(.4,.6,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')


% 6 Mic

axes('position',[.35 .275 width height])
[x y z] = sphere(60);
patch(surf2patch(x,y,z,z),'EdgeColor',[.1 .1 .1],'FaceColor',...
    [.7 .7 .7],'LineWidth',.4);
view([41 30])
axis vis3d
```

```
axis off
alpha(0.5)
hold on
plot3(0,0,1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,1,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(1,0,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,0,-1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,-1,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(-1,0,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'-k',...
    'LineWidth',.5)
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)

plot3(linspace(0,0,50),linspace(0,0,50),linspace(1,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(1,-1,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,-1,50),linspace(0,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,1,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,1,50),linspace(0,0,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,-1,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(0,0,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,1,50),linspace(1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(1,0,50),linspace(0,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,0,50),linspace(0,0,50),linspace(0,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,0,50),linspace(0,-1,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,-1,50),linspace(-1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(0,0,50),linspace(-1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(-1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
text(.3,.6,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

axes('position',[-.13 .275 width height])
```

```
sphere
hold on
alpha(0)
shading interp
plot3(0,0,1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,1,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(1,0,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,0,-1,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(0,-1,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(-1,0,0,'ob','MarkerFaceColor','b','MarkerSize',dotsize,...
    'MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'-k',...
    'LineWidth',.5)
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'-k',...
    'LineWidth',.5)

plot3(linspace(0,0,50),linspace(0,0,50),linspace(1,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(1,-1,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,-1,50),linspace(0,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,1,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,1,50),linspace(0,0,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,-1,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(0,0,50),linspace(1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,1,50),linspace(1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(1,0,50),linspace(0,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,0,50),linspace(0,0,50),linspace(0,-1,50),'--k',...
    'LineWidth',.1)
plot3(linspace(1,0,50),linspace(0,-1,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,0,50),linspace(0,-1,50),linspace(-1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(0,0,50),linspace(-1,0,50),'--k',...
    'LineWidth',.1)
plot3(linspace(0,-1,50),linspace(-1,0,50),linspace(0,0,50),'--k',...
    'LineWidth',.1)
view([41 30])
axis vis3d
axis off
```

```
text(.5,.6,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

cow = .4;
cow2 = .45;


% Orthogonal

axes('position',[.35 -.05 width height])
[x y z] = sphere(60);
patch(surf2patch(x,y,z,z),'EdgeColor',[.1 .1 .1],'FaceColor',[.7 .7 .7],...
    'LineWidth',.4);
view([41 30])
axis vis3d
axis off
alpha(0.5)
hold on
plot3(1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(linspace(-1/sqrt(3),.9,50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),.9,50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),.9,50),'-k','LineWidth',.5)
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),1/sqrt(3),50),...
    -1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),linspace(1/sqrt(3),...
    -1/sqrt(3),50),-1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),linspace(1/sqrt(3),-1/sqrt(3),50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)
plot3(linspace(1/sqrt(3),-1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)

plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
text(.3,.6,'(f)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')
```

```
axes('position',[-.06 -.05 width height])
sphere
hold on
% plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
%     -1/sqrt(3).*ones(50),'-k','LineWidth',1)
% plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),1/sqrt(3),50),...
%     -1/sqrt(3).*ones(50),'-k','LineWidth',1)
% plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
%     linspace(-1/sqrt(3),1/sqrt(3),50),'-k','LineWidth',1)
plot3(linspace(-1/sqrt(3),.9,50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),.9,50),...
    -1/sqrt(3).*ones(50),'-k','LineWidth',.5)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),.9,50),'-k','LineWidth',.5)
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    -1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),linspace(-1/sqrt(3),1/sqrt(3),50),...
    -1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)
plot3(linspace(-1/sqrt(3),1/sqrt(3),50),linspace(1/sqrt(3),...
    -1/sqrt(3),50),-1/sqrt(3).*ones(50),'--k','LineWidth',.1)
plot3(-1/sqrt(3).*ones(50),linspace(1/sqrt(3),-1/sqrt(3),50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)
plot3(linspace(1/sqrt(3),-1/sqrt(3),50),-1/sqrt(3).*ones(50),...
    linspace(-1/sqrt(3),1/sqrt(3),50),'--k','LineWidth',.1)
plot3(1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),-1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')
plot3(-1/sqrt(3),-1/sqrt(3),1/sqrt(3),'ob','MarkerFaceColor','b',...
    'MarkerSize',dotsize,'MarkerEdgeColor','none')

plot3(linspace(0,0,50),linspace(0,0,50),linspace(1.4,-1.4,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
plot3(linspace(0,0,50),linspace(1.4,-1.4,50),linspace(0,0,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
plot3(linspace(1.4,-1.4,50),linspace(0,0,50),linspace(0,0,50),'o',...
    'MarkerFaceColor','none','MarkerEdgeColor','none')
alpha(0)
shading interp
view([41 30])
axis vis3d
axis off
text(0.27,.6,'(e)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 5]);
```

```
print -dtiffn -r1600 Figure1

%% Figure 3-2

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
ytickvect = [0 2 4 6];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I2211b{9},'-',kb,I2111b{9},'--')
% xlabel('ka')
ylabel(labely1)
legend('4R/1','4R/4','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I2221b{9},'-',ka,I2121b{9},'--')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
```

```
legend('4R.S/1','4R.S/4','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I2211b{4},'-',kb,I2111b{4},'--')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I2221b{4},'-',ka,I2121b{4},'--')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure2

%% Figure 3-3

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
```

```
ylimit = 15;
ytickvect = [0 4 8 12];
labely1 = 'Average Error (degrees)';
labely2 = 'Max Error (degrees)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I2211angb{9},'-',kb,I2111angb{9},'--')
% xlabel('ka')
ylabel(labely1)
legend('4R/1','4R/4','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I2221angb{9},'-',ka,I2121angb{9},'--')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
legend('4R.S/1','4R.S/4','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I2211angb{4},'-',kb,I2111angb{4},'--')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
```

```
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I2221angb{4},'-',ka,I2121angb{4},'--')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure3

%% Figure 3-4

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
ytickvect = [0 2 4 6];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I3311b{9},'-',kb,I3211b{9},'--',...
    kb,I3111b{9},'-.',kb,I3411b{9},':')
```

```
% xlabel('ka')
ylabel(labely1)
legend('6/1','6/6','6/3 1D','6/15 1D','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I3321b{9},'-',ka,I3221b{9},'--',...
    ka,I3121b{9},'-.',ka,I3421b{9},':')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
legend('6.S/1','6.S/6','6.S/3 1D','6.S/15 1D','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I3311b{4},'-',kb,I3211b{4},'--',...
    kb,I3111b{4},'-.',kb,I3411b{4},':')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I3321b{4},'-',ka,I3221b{4},'--',...
    ka,I3121b{9},'-.',ka,I3421b{4},':')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName',...
```

```
    'Times','FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure4

%% Figure 3-5

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);


xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 15;
ytickvect = [0 4 8 12];
labely1 = 'Average Error (degrees)';
labely2 = 'Max Error (degrees)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I3311angb{9},'-',kb,I3211angb{9},'--',...
    kb,I3111angb{9},'-.',kb,I3411angb{9},':')
% xlabel('ka')
ylabel(labely1)
legend('6/1','6/6','6/3 1D','6/15 1D','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(222)
axes('position',[.57 .55 .42 .44])
```

```
plot(ka,I3321angb{9},'-',ka,I3221angb{9},'--',...
    ka,I3121angb{9},'-.',ka,I3421angb{9},':')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
legend('6.S/1','6.S/6','6.S/3 1D','6.S/15 1D','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I3311angb{4},'-',kb,I3211angb{4},'--',...
    kb,I3111angb{4},'-.',kb,I3411angb{4},':')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I3321angb{4},'-',ka,I3221angb{4},'--',...
    ka,I3121angb{9},'-.',ka,I3421angb{4},':')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)

xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure5

%% Figure 3-6

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
```

```
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);



xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
ytickvect = [0 2 4 6];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I2121b{9},'-',ka,I3421b{9},'--',ka,I1221b{9},'-.')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
legend('4R.S/4','6.S/15 1D','4O.S/1','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I2111b{9},'-',kb,I3411b{9},'--',kb,I1211b{9},'-.')
% xlabel('ka')
ylabel(labely1)
legend('4R/4','6/15 1D','4O/1','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')
```

```
% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I2111b{4},'-',kb,I3411b{4},'--',kb,I1211b{4},'-.')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I2121b{4},'-',ka,I3421b{4},'--',ka,I1221b{4},'-.')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure6


%% Figure 3-7

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);

xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 15;
ytickvect = [0 4 8 12];
labely1 = 'Average Error (degrees)';
```

```
labely2 = 'Max Error (degrees)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = 'ka';
labelx2 = '3/2 ka';
kb = 2*pi*freq*a/c;
ka = 2*pi*freq*3/2*a/c;

figure

% subplot(222)
axes('position',[.57 .55 .42 .44])
plot(ka,I2121angb{9},'-',ka,I3421angb{9},'--',...
    ka,I1522angb{9},'-.')
set(gca,'YTickLabel','')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
% xlabel('ka')
% ylabel('Error (dB)')
legend('4R.S/4','6.S/15 1D','4O.S/3.T','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(221)
axes('position',[.12 .55 .42 .44])
plot(kb,I2111angb{9},'-',kb,I3411angb{9},'--',...
    kb,I1512angb{9},'-.')
% xlabel('ka')
ylabel(labely1)
legend('4R/4','6/15 1D','4O/3.T','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')


% subplot(223)
axes('position',[.12 .09 .42 .44])
plot(kb,I2111angb{4},'-',kb,I3411angb{4},'--',...
    kb,I1512angb{4},'-.')
xlabel(labelx1)
ylabel(labely2)
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
```

```
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

% subplot(224)
axes('position',[.57 .09 .42 .44])
plot(ka,I2121angb{4},'-',ka,I3421angb{4},'--',...
    ka,I1522angb{4},'-.')
set(gca,'YTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlabel(labelx2)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName',...
    'Times','FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 3+3/8 4]);

print -dtiffn -r200 Figure7
```

B.4 Genetic_Algorithm.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Genetic_Algorithm.m                                                      %
% Date: 08 August 2011                                                           %
% Author: Curtis Wiederhold                                                      %
%                                                                                %
% Description: This Matlab script uses a genetic algorithm to predict           %
% an optimal four-microphone spherical probe design based on what error         %
% quantity is being optimized to. The error quantity is specified as the        %
% value returned from the function file "Intensity_Error.m." The other          %
% function: "Design_Checker.m" outputs errors at all frequencies (not           %
% just at the frequency to which to optimization was run) which can then        %
% be used with "Chapter_4_Figures.m" to produce error plots.                     %
%                                                                                %
% Functions called by script:                                                    %
% Intensity_Error.m                                                              %
% Design_Checker.m                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;

tic
filename = 'NoPhase_4501Hz_Imagavg_15run_res20.mat';
endmessage = 'mag avg 4501 Hz, 15 runs';
runs = 15;
gensize = 500;
M = 100;
crossprob = 0.5;
crossdev = 5;
mutprob = 0.25;
tournsize = 50;
eta = 0.5;
beta = 0.1;
parent = zeros(8,gensize);
% freq = 5000;
% freq = [1 6752];
% freq = [50 2000 4000 6752];
freq = 4501;
% freq = 281;
res = 20;
res_step = pi/res;

for s=1:runs

    for n=1:gensize
        Rrank = 3;
        while Rrank<4
            thetas = randi([0 180],4,1);
            phis = randi([0 359],4,1);
            x1 = sin(thetas(1)).*cos(phis(1));
            y1 = sin(thetas(1)).*sin(phis(1));
            z1 = cos(thetas(1));
            x2 = sin(thetas(2)).*cos(phis(2));
```

```
        y2 = sin(thetas(2)).*sin(phis(2));
        z2 = cos(thetas(2));
        x3 = sin(thetas(3)).*cos(phis(3));
        y3 = sin(thetas(3)).*sin(phis(3));
        z3 = cos(thetas(3));
        x4 = sin(thetas(4)).*cos(phis(4));
        y4 = sin(thetas(4)).*sin(phis(4));
        z4 = cos(thetas(4));
        R = [1 x1 y1 z1; 1 x2 y2 z2; 1 x3 y3 z3; 1 x4 y4 z4];
        parent(:,n) = [thetas(1) phis(1) thetas(2) phis(2) ...
            thetas(3) phis(3) thetas(4) phis(4)];
        Rrank = rank(R);
    end

end
fitness2 = Intensity_Error(parent,freq,res);
for k=1:M
    disp(['Run: ',int2str(s),' of ',int2str(runs),'. Generation: ',...
        int2str(k),' of ',int2str(M),'. I mag avg 4501 Hz run.'])
    toc
    alpha = (1-(k-1)/M)^beta;
    fitness = fitness2;
    child = parent;

    % Crossover, Parthenogensis style
    for n=1:gensize
        Rrank1 = 3;
        while Rrank1<4
            % Selection, Tournament Style
            tempvar = find(fitness==min(fitness(randi(gensize,1,...
                tournsize))));
            fatherind(n) = tempvar(1);

            if rand<=crossprob
                child(:,n) = alpha.*randn(8,1).*crossdev...
                    +parent(:,fatherind(n));
            else
                child(:,n) = parent(:,fatherind(n));
            end

            x1 = sin(child(1,n)).*cos(child(2,n));
            y1 = sin(child(1,n)).*sin(child(2,n));
            z1 = cos(child(1,n));
            x2 = sin(child(3,n)).*cos(child(4,n));
            y2 = sin(child(3,n)).*sin(child(4,n));
            z2 = cos(child(3,n));
            x3 = sin(child(5,n)).*cos(child(6,n));
            y3 = sin(child(5,n)).*sin(child(6,n));
            z3 = cos(child(5,n));
            x4 = sin(child(7,n)).*cos(child(8,n));
            y4 = sin(child(7,n)).*sin(child(8,n));
            z4 = cos(child(7,n));
            R = [1 x1 y1 z1; 1 x2 y2 z2; 1 x3 y3 z3; 1 x4 y4 z4];
            Rrank1 = rank(R);
        end
```

```
        end

        % Mutation, Dynamic Style
        for j=1:size(child,2)
            Rrank = 3;
            while Rrank<4
                for n=1:size(child,1)
                    if rand<mutprob
                        if mod(n,2)
                            r = randi([0 180]);
                            if r<=child(n,j)
                                child(n,j) = round(r^alpha*child(n,j)...
                                    ^(1-alpha));
                            else
                                child(n,j) = round(180-(180-r)^alpha...
                                    *(180-child(n,j))^(1-alpha));
                            end
                        else
                            r = randi([0 359]);
                            if r<=child(n,j)
                                child(n,j) = round(r^alpha*child(n,j)...
                                    ^(1-alpha));
                            else
                                child(n,j) = round(359-(359-r)^alpha...
                                    *(359-child(n,j))^(1-alpha));
                            end
                        end
                    end
                end
                x1 = sin(child(1,j)).*cos(child(2,j));
                y1 = sin(child(1,j)).*sin(child(2,j));
                z1 = cos(child(1,j));
                x2 = sin(child(3,j)).*cos(child(4,j));
                y2 = sin(child(3,j)).*sin(child(4,j));
                z2 = cos(child(3,j));
                x3 = sin(child(5,j)).*cos(child(6,j));
                y3 = sin(child(5,j)).*sin(child(6,j));
                z3 = cos(child(5,j));
                x4 = sin(child(7,j)).*cos(child(8,j));
                y4 = sin(child(7,j)).*sin(child(8,j));
                z4 = cos(child(7,j));
                R = [1 x1 y1 z1; 1 x2 y2 z2; 1 x3 y3 z3; 1 x4 y4 z4];
                Rrank = rank(R);
            end
%             child(:,j) = sortrows(child(:,j));
        end

        % Elitism. Find best candidates for next generation from both
        % parents and children
        fitness2 = Intensity_Error(child,freq,res);
        pool(1:gensize) = fitness;
        pool(gensize+1:2*gensize) = fitness2;
        despool(:,1:gensize) = parent;
        despool(:,gensize+1:2*gensize) = child;
        bestfitness(k) = min(pool);
        avgfitness(k) = mean(pool);
```

```
        % Find best designs, make them the new children
        for n=1:gensize
            temp2 = find(pool==min(pool));
            index = temp2(1);
            child2(:,n) = despool(:,index);
            fitness2(n) = pool(index);
            pool(index) = max(pool);
        end

        bestdesign(:,k) = child2(:,1);
        parent = child2;

    end

    opt(:,s) = bestdesign(:,end);
    fopt(s) = bestfitness(end);
    bestdesigns(:,:,s) = bestdesign;
    bestfitnesses(:,s) = bestfitness;
    avgfitnesses(:,s) = avgfitness;
    finalgen(:,:,s) = child2;

    toc

end


totaltime=toc

% bestdesign(:,end)

[I4121b I4121angb I1321b I1321angb I2121b I2121angb f] =...
    Design_Checker(bestdesigns(:,end,find(fopt==min(fopt))));

save(filename)
disp(endmessage)
```

B.4.1 Intensity_Error.m

```
function [Ierr] = Intensity_Error(x,freq,res)

res_step = pi/res;

% gensize = size(chromosome,3);
if size(x,1)==1
    x=x';
end
gensize = size(x,2);
% mictheta(1,1:gensize) = 0;
% micphi(1,1:gensize) = 0;
% micphi(2:4,:) = chromosome(1:3,2,:)*pi/180;
% mictheta(2:4,:) = chromosome(1:3,1,:)*pi/180;
micphi = [x(2,:); x(4,:); x(6,:); x(8,:)].*pi/180;
mictheta = [x(1,:); x(3,:); x(5,:); x(7,:)].*pi/180;


x1 = sin(mictheta(1,:)).*cos(micphi(1,:));
y1 = sin(mictheta(1,:)).*sin(micphi(1,:));
z1 = cos(mictheta(1,:));
x2 = sin(mictheta(2,:)).*cos(micphi(2,:));
y2 = sin(mictheta(2,:)).*sin(micphi(2,:));
z2 = cos(mictheta(2,:));
x3 = sin(mictheta(3,:)).*cos(micphi(3,:));
y3 = sin(mictheta(3,:)).*sin(micphi(3,:));
z3 = cos(mictheta(3,:));
x4 = sin(mictheta(4,:)).*cos(micphi(4,:));
y4 = sin(mictheta(4,:)).*sin(micphi(4,:));
z4 = cos(mictheta(4,:));


% Find least-squares estimate of pressure gradient
% Commented process follows described on p. 28 of Derek's thesis
% Uncommented process follows method described by Pascal (Eqs 6-8)

for s=1:gensize
%     r12 = [x2(s)-x1(s) y2(s)-y1(s) z2(s)-z1(s)];
%     r13 = [x3(s)-x1(s) y3(s)-y1(s) z3(s)-z1(s)];
%     r14 = [x4(s)-x1(s) y4(s)-y1(s) z4(s)-z1(s)];
%     r23 = [x3(s)-x2(s) y3(s)-y2(s) z3(s)-z2(s)];
%     r24 = [x4(s)-x2(s) y4(s)-y2(s) z4(s)-z2(s)];
%     r34 = [x4(s)-x3(s) y4(s)-y3(s) z4(s)-z3(s)];

%     R(:,:,s) = [r12; r13; r14; r23; r24; r34];

    Mpascal(:,:,s) = [1 x1(s) y1(s) z1(s); 1 x2(s) y2(s) z2(s);...
        1 x3(s) y3(s) z3(s); 1 x4(s) y4(s) z4(s)];
end


% for loop to calculate values at each frequency
```

```
q = 1;
for f=(freq(1:length(freq)))
    % f=freq;
    w = 2*pi*f;
    c = 343;
    rho = 1.21;
    k = w/c;
    M = 15; % Number of terms to include in the scattering expansion
    a = 0.0254/2; % Radius of the probe
    % Direction of plane wave propagation
    o = 1;
    for thetak=0:res_step:pi
        n=1;
        % If you want to run a half space scenario this line of code should
        % instead be: for phik = 0:res_step:pi
        for phik = 0:res_step:2*pi

            % wave vector components of the plane wave
            kx = -k*cos(phik)*sin(thetak);
            ky = -k*sin(phik)*sin(thetak);
            kz = -k*cos(thetak);

            p0 = 1; % Amplitude of the incident wave

            rk1 = -(-1./2.*(cos(mictheta(1,:)-thetak).*...
                (1+cos(micphi(1,:)-phik))+cos(mictheta(1,:)+thetak).*...
                (1-cos(micphi(1,:)-phik))));
            rk2 = -(-1./2.*(cos(mictheta(2,:)-thetak).*...
                (1+cos(micphi(2,:)-phik))+cos(mictheta(2,:)+thetak).*...
                (1-cos(micphi(2,:)-phik))));
            rk3 = -(-1./2.*(cos(mictheta(3,:)-thetak).*...
                (1+cos(micphi(3,:)-phik))+cos(mictheta(3,:)+thetak).*...
                (1-cos(micphi(3,:)-phik))));
            rk4 = -(-1./2.*(cos(mictheta(4,:)-thetak).*...
                (1+cos(micphi(4,:)-phik))+cos(mictheta(4,:)+thetak).*...
                (1-cos(micphi(4,:)-phik))));

            % Calculate the incident pressures (pi1-pi4) and the first
            % terms of the scattered pressures (ps1-ps4)
            % Equation for plane wave: p=p0*exp(-j*k*r)
            %             pi1 = p0*exp(1i*k*a*rk1);
            %             pi2 = p0*exp(1i*k*a*rk2);
            %             pi3 = p0*exp(1i*k*a*rk3);
            %             pi4 = p0*exp(1i*k*a*rk4);

            % Initialize the first two Legendre polynomials for the
            % recurrence relations
            P1mm2(1:gensize) = 1;
            P2mm2(1:gensize) = 1;
            P3mm2(1:gensize) = 1;
            P4mm2(1:gensize) = 1;

            P1mm1 = rk1;
            P2mm1 = rk2;
            P3mm1 = rk3;
```

```
P4mm1 = rk4;

m = 0;  % initial order for Bessel functions
% Calculate the necessary Bessel and Hankel functions
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a); % order m minus 1
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a); % order m plus 1
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));

Am = (-1i)^m*(2*m+1)/dhm;
p1 = Am*P1mm2;
p2 = Am*P2mm2;
p3 = Am*P3mm2;
p4 = Am*P4mm2;

m = 1;
% Calculate the necessary Bessel and Hankel functions
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));
Am = (-1i)^m*(2*m+1)/dhm;

% now add results from order m=0 results to those of m=1
p1 = p1+Am*P1mm1;
p2 = p2+Am*P2mm1;
p3 = p3+Am*P3mm1;
p4 = p4+Am*P4mm1;

% Now that you have 2 calculated, recurrence relations can be
% used to calculate the rest.
for m = 2:M;
    % Use a recurrence relation to calculate the Legendre polys
    P1m = 2*rk1.*P1mm1-P1mm2-1./m.*(rk1.*P1mm1-P1mm2);
    P2m = 2*rk2.*P2mm1-P2mm2-1./m.*(rk2.*P2mm1-P2mm2);
    P3m = 2*rk3.*P3mm1-P3mm2-1./m.*(rk3.*P3mm1-P3mm2);
    P4m = 2*rk4.*P4mm1-P4mm2-1./m.*(rk4.*P4mm1-P4mm2);

    % Calculate the necessary Bessel and Hankel functions
    jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
    jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
    ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
    ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
    dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));
    %                 Am = p0*(2*m+1)*1i^m*h2m*djm/dhm;
    Am = (-1i)^m*(2*m+1)/dhm;

    p1 = p1+Am*P1m;
    p2 = p2+Am*P2m;
    p3 = p3+Am*P3m;
    p4 = p4+Am*P4m;
```

```
    % Update the Legendre polynomials for the recurrence
    % relation
    P1mm2 = P1mm1;
    P2mm2 = P2mm1;
    P3mm2 = P3mm1;
    P4mm2 = P4mm1;

    P1mm1 = P1m;
    P2mm1 = P2m;
    P3mm1 = P3m;
    P4mm1 = P4m;
end

% total pressure = incident + scattered pressures
p1 = conj(p1*1i*p0/(k*a)^2);
p2 = conj(p2*1i*p0/(k*a)^2);
p3 = conj(p3*1i*p0/(k*a)^2);
p4 = conj(p4*1i*p0/(k*a)^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%        EXACT CALCULATIONS      %%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% particle velocity
vx = kx/rho/w*p0;
vy = ky/rho/w*p0;
vz = kz/rho/w*p0;
v(o,n) = (abs(vx).^2+abs(vy).^2+abs(vz).^2).^(1/2);

% Intensity calculations
Ix = 0.5*real(p0*conj(vx));
Iy = 0.5*real(p0*conj(vy));
Iz = 0.5*real(p0*conj(vz));
I(o,n) = (abs(Ix).^2+abs(Iy).^2+abs(Iz).^2).^(1/2);

%             % Energy density
%             W(o,n) = abs(rho*v(o,n)^2/2 + p0.^2/2/rho/c^2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%        FINITE-DIFF CALCULATIONS   %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Pressure Calculations

%             pi_avg = (pi1+pi2+pi3+pi4)/4;
p_avg = (p1+p2+p3+p4)/4;

%             Pi(o,n) = abs(pi_avg);
P(o,n,:) = abs(p_avg);
```

```
            %%%%%%% Velocity Calculations
            for s=1:gensize
%                 delp1 = [pi2-pi1 pi3-pi1 pi4-pi1 pi3-pi2 pi4-pi2...
%                     pi4-pi3].';
%                 delp2 = [p2(s)-p1(s) p3(s)-p1(s) p4(s)-p1(s) ...
%                     p3(s)-p2(s) p4(s)-p2(s) p4(s)-p3(s)].';
%                 gradp1 = R\delp1;
%                 gradp2(:,s) = R(:,:,s)\squeeze(delp2);

                Ppascal = [p1(s); p2(s); p3(s); p4(s)];
                Dpascal(:,s) = Mpascal(:,:,s)\Ppascal;
            end
%             gradp2 = squeeze(gradp2);

            %             % velocity finite-difference approx. w/o sphere
            %             % (1 = no sphere)
            %             vx1=1i/rho/a/w*gradp1(1);
            %             vy1=1i/rho/a/w*gradp1(2);
            %             vz1=1i/rho/a/w*gradp1(3);
            %       v1(o,n) = (abs(vx1).^2+abs(vy1).^2+abs(vz1).^2).^(1/2);

            % approximations w/ sphere
            % (2 = with sphere)
%             vx2=1i/rho/(a*3/2)/w*gradp2(1,:);
%             vy2=1i/rho/(a*3/2)/w*gradp2(2,:);
%             vz2=1i/rho/(a*3/2)/w*gradp2(3,:);
            vx2=1i/rho/(a*3/2)/w*Dpascal(2,:);
            vy2=1i/rho/(a*3/2)/w*Dpascal(3,:);
            vz2=1i/rho/(a*3/2)/w*Dpascal(4,:);
            v2(o,n,:) = (abs(vx2).^2+abs(vy2).^2+abs(vz2).^2).^(1/2);

            %             vx1=1i/rho/4/a/w*sqrt(2)*(pi2-2*pi3+pi4);
            %             vy1=1i/rho/4/a/w*sqrt(6)*(-pi2+pi4);
            %             vz1=1i/rho/4/a/w*(3*pi1-pi2-pi3-pi4);
            %       v1(o,n) = (abs(vx1).^2+abs(vy1).^2+abs(vz1).^2).^(1/2);
            %             vx2=1i/rho/4/(a*3/2)/w*sqrt(2)*(p2-2*p3+p4);
            %             vy2=1i/rho/4/(a*3/2)/w*sqrt(6)*(-p2+p4);
            %             vz2=1i/rho/4/(a*3/2)/w*(3*p1-p2-p3-p4);
            %       v2(o,n) = (abs(vx2).^2+abs(vy2).^2+abs(vz2).^2).^(1/2);

            %             % approximations w/ sphere but without 3/2 factor
            %             % (3 = with sphere w/o 3/2 factor)
            %             vx3=-j/rho/(a)/w*gradp2(1);
            %             vy3=-j/rho/(a)/w*gradp2(2);
            %             vz3=-j/rho/(a)/w*gradp2(3);
            %       v3(o,n) = (abs(vx3).^2+abs(vy3).^2+abs(vz3).^2).^(1/2);


            %%%%%%% Intensity Calculations

            %  [Ix1 Iy1 Iz1 I1(o,n)] = Intensity_Calc(pi_avg,vx1,vy1,vz1);
            Ix2 = 0.5.*real(p_avg.*conj(vx2));
            Iy2 = 0.5.*real(p_avg.*conj(vy2));
```

```
            Iz2 = 0.5.*real(p_avg.*conj(vz2));
            I2(o,n,:) = (abs(Ix2).^2+abs(Iy2).^2+abs(Iz2).^2).^(1/2);
            %   [Ix2 Iy2 Iz2 I2(o,n)] = Intensity_Calc(p_avg,vx2,vy2,vz2);
            %   [Ix3 Iy3 Iz3 I3(o,n)] = Intensity_Calc(p_avg,vx3,vy3,vz3);

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%       DIRECTION CALCULATIONS     %%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


            Ivect = [Ix Iy Iz];
            %             vvect = [vx vy vz];

            %             I1ang(o,n,q) = Angle_Calc(Ivect,Ix1,Iy1,Iz1);

            for s=1:gensize
%                 vect1 = [Ix2(s) Iy2(s) Iz2(s)];
%                 I2ang(o,n,s) = abs(acos(dot(Ivect,vect1)/...
%                     (norm(Ivect)*norm(vect1))))*180/pi;
                I2ang(o,n,s) = Angle_Calc(Ivect,Ix2(s),Iy2(s),Iz2(s));
            end

            %         I2ang(o,n,q) = Angle_Calc(Ivect,Ix2,Iy2,Iz2);
            %             I3ang(o,n,q) = Angle_Calc(Ivect,Ix3,Iy3,Iz3);

            %             v1ang(o,n,q) = Angle_Calc(vvect,vx1,vy,vz1);
            %             v2ang(o,n,q) = Angle_Calc(vvect,vx2,vy2,vz2);
            %             v3ang(o,n,q) = Angle_Calc(vvect,vx3,vy3,vz3);


            n = n+1;
        end
        o = o+1;
    end


    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%  Finite Diff Bias Calculations   %%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % I2(o,n,:)
    % [I1b{1}(:,:,q) I1b{2}(q) I1b{3}(q) I1b{4}(q) I1b{5}(q) I1b{6}(q) ...
    %         I1b{7}(q) I1b{8}(q) I1b{9}(q)] = Bias_Calc(res_step,I,I1,10);
    for s=1:gensize
        [I2b{1}(:,:,s) I2b{2}(s) I2b{3}(s) I2b{4}(s) I2b{5}(s)...
            I2b{6}(s) I2b{7}(s) I2b{8}(s) I2b{9}(s)] = ...
            Bias_Calc(res_step,I,I2(:,:,s),10);
    end

    % [I3b{1}(:,:,q) I3b{2}(q) I3b{3}(q) I3b{4}(q) I3b{5}(q) I3b{6}(q) ...
    %         I3b{7}(q) I3b{8}(q) I3b{9}(q)] = Bias_Calc(res_step,I,I3,10);
```

```
%  [Pib{1}(:,:,q) Pib{2}(q) Pib{3}(q) Pib{4}(q) Pib{5}(q) Pib{6}(q) ...
%        Pib{7}(q) Pib{8}(q) Pib{9}(q)] = Bias_Calc(res_step,p0,Pi,20);
%     [Pb{1}(:,:,q) Pb{2}(q) Pb{3}(q) Pb{4}(q) Pb{5}(q) Pb{6}(q) ...
%         Pb{7}(q) Pb{8}(q) Pb{9}(q)] = Bias_Calc(res_step,p0,P,20);
%
%  [v1b{1}(:,:,q) v1b{2}(q) v1b{3}(q) v1b{4}(q) v1b{5}(q) v1b{6}(q) ...
%        v1b{7}(q) v1b{8}(q) v1b{9}(q)] = Bias_Calc(res_step,v,v1,20);
%  [v2b{1}(:,:,q) v2b{2}(q) v2b{3}(q) v2b{4}(q) v2b{5}(q) v2b{6}(q) ...
%        v2b{7}(q) v2b{8}(q) v2b{9}(q)] = Bias_Calc(res_step,v,v2,20);
%  [v3b{1}(:,:,q) v3b{2}(q) v3b{3}(q) v3b{4}(q) v3b{5}(q) v3b{6}(q) ...
%        v3b{7}(q) v3b{8}(q) v3b{9}(q)] = Bias_Calc(res_step,v,v3,20);



    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%  Direction Bias Calculations   %%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     [I1angb{1}(:,:,q) I1angb{2}(q) I1angb{3}(q) I1angb{4}(q)...
%         I1angb{5}(q) I1angb{6}(q) I1angb{7}(q) I1angb{8}(q)...
%         I1angb{9}(q)] = Bias_Calc(res_step,I1ang(:,:,q));
    for s=1:gensize
        [I2angb{1}(:,:,s) I2angb{2}(s) I2angb{3}(s) I2angb{4}(s)...
            I2angb{5}(s) I2angb{6}(s) I2angb{7}(s) I2angb{8}(s)...
            I2angb{9}(s)] = Bias_Calc(res_step,I2ang(:,:,s));
    end
%     [I3angb{1}(:,:,q) I3angb{2}(q) I3angb{3}(q) I3angb{4}(q)...
%         I3angb{5}(q) I3angb{6}(q) I3angb{7}(q) I3angb{8}(q)...
%         I3angb{9}(q)] = Bias_Calc(res_step,I3ang(:,:,q));

%     [v1angb{1}(:,:,q) v1angb{2}(q) v1angb{3}(q) v1angb{4}(q)...
%         v1angb{5}(q) v1angb{6}(q) v1angb{7}(q) v1angb{8}(q)...
%         v1angb{9}(q)] = Bias_Calc(res_step,v1ang(:,:,q));
%     [v2angb{1}(:,:,q) v2angb{2}(q) v2angb{3}(q) v2angb{4}(q)...
%         v2angb{5}(q) v2angb{6}(q) v2angb{7}(q) v2angb{8}(q)...
%         v2angb{9}(q)] = Bias_Calc(res_step,v2ang(:,:,q));
%     [v3angb{1}(:,:,q) v3angb{2}(q) v3angb{3}(q) v3angb{4}(q)...
%         v3angb{5}(q) v3angb{6}(q) v3angb{7}(q) v3angb{8}(q)...
%         v3angb{9}(q)] = Bias_Calc(res_step,v3ang(:,:,q));

    %     q = q+1;
    % end

    %         Ierr(s) = I2angb{9};
    % size(I2b)
    % I2b{9}
    % I2b{4}
    % I2angb{9}
    % I2angb{4}
    % pause

    Iavg(q,:) = I2b{9};
    Imax(q,:) = I2b{4};
```

```
    Iangavg(q,:) = I2angb{9};
    Iangmax(q,:) = I2angb{4};

    q = q+1;
end

% Ierr = I2b{9}+I2b{4}+.5*I2angb{9}+.5*I2angb{4};
Ierr = sum(Iavg,1);
% Ierr = Iangavg(1,:);

for s=1:gensize
%     if rank(R(:,:,s))<3
    if rank(Mpascal(:,:,s))<4
        Ierr(s) = 10000;
%         error('BAD RANK! REDO YOUR CODE!')
%         'BAD RANK! REDO YOUR CODE!'
    end
end


end
```

B.4.2 Design_Checker.m

```
function [I4121b I4121angb I1321b I1321angb I2121b I2121angb freq] = ...
    Design_Checker(x)

res = 50;
res_step = pi/res;

% specify frequencies to sweep
freq1 = 10;
freq2 = 100;
freq3 = 500:200:8000;
freq = horzcat(freq1,freq2,freq3);
% freq = 7740:1:7745;
% freq = 21492; % ka=5
% freq = 4298; % ka=1
% freq = 5000;

% Cubic Probe mic positions
mictheta1(1) = acos(-1/sqrt(3));
micphi1(1) = atan2(-1/sqrt(3),1/sqrt(3));
mictheta1(2) = acos(-1/sqrt(3));
micphi1(2) = atan2(1/sqrt(3),-1/sqrt(3));
mictheta1(3) = acos(-1/sqrt(3));
micphi1(3) = atan2(-1/sqrt(3),-1/sqrt(3));
mictheta1(4) = acos(1/sqrt(3));
micphi1(4) = atan2(-1/sqrt(3),-1/sqrt(3));

% Regular Tetrahedron Probe mic positions
mictheta2(1) = 0;
micphi2(1) = 0;
mictheta2(2) = 1.910633237;
micphi2(2) = -pi/3;
mictheta2(3) = 1.910633237;
micphi2(3) = pi;
mictheta2(4) = 1.910633237;
micphi2(4) = pi/3;

% 6 Mic Probe mic positions
mictheta3(1) = 0;
micphi3(1) = 0;
mictheta3(2) = pi/2;
micphi3(2) = 0;
mictheta3(3) = pi/2;
micphi3(3) = pi/2;
mictheta3(4) = pi/2;
micphi3(4) = pi;
mictheta3(5) = pi/2;
micphi3(5) = 3*pi/2;
mictheta3(6) = pi;
micphi3(6) = 0;

% Genetic Algorithm Probe mic positions
if size(x,1)==1
```

```
    x=x';
end
micphi4 = [x(2,:); x(4,:); x(6,:); x(8,:)].*pi/180;
mictheta4 = [x(1,:); x(3,:); x(5,:); x(7,:)].*pi/180;

% GA Probe code for least squares estimate of gradient
x1 = sin(mictheta4(1))*cos(micphi4(1));
y1 = sin(mictheta4(1))*sin(micphi4(1));
z1 = cos(mictheta4(1));
x2 = sin(mictheta4(2))*cos(micphi4(2));
y2 = sin(mictheta4(2))*sin(micphi4(2));
z2 = cos(mictheta4(2));
x3 = sin(mictheta4(3))*cos(micphi4(3));
y3 = sin(mictheta4(3))*sin(micphi4(3));
z3 = cos(mictheta4(3));
x4 = sin(mictheta4(4))*cos(micphi4(4));
y4 = sin(mictheta4(4))*sin(micphi4(4));
z4 = cos(mictheta4(4));

r12 = [x2-x1 y2-y1 z2-z1]';
r13 = [x3-x1 y3-y1 z3-z1]';
r14 = [x4-x1 y4-y1 z4-z1]';
r23 = [x3-x2 y3-y2 z3-z2]';
r24 = [x4-x2 y4-y2 z4-z2]';
r34 = [x4-x3 y4-y3 z4-z3]';

% d12 = sqrt(sum(r12.^2));
% d13 = sqrt(sum(r13.^2));
% d14 = sqrt(sum(r14.^2));
% d23 = sqrt(sum(r23.^2));
% d24 = sqrt(sum(r24.^2));
% d34 = sqrt(sum(r34.^2));
% 2/sqrt(3)./[d12 d13 d14 d23 d24 d34]
% mind = min([d12 d13 d14 d23 d24 d34])
% factor = (2/sqrt(3))/mind

R = [r12 r13 r14 r23 r24 r34]';
if rank(R)<3
    error('bad rank');
end



% for loop to calculate values at each frequency
q = 1;
for f=(freq(1:length(freq)))
    f
    w = 2*pi*f;
    c = 343;
    rho = 1.21;
    k = w/c;
    M = 25; % Number of terms to include in the scattering expansion
    a = 0.0254/2; % Radius of the probe
    % Direction of plane wave propagation
    o = 1;
```

```
for thetak=0:res_step:pi
    n=1;
    % If you want to run a half space scenario this line of code should
    % instead be: for phik = 0:res_step:pi
    for phik = 0:res_step:2*pi

        % wave vector components of the plane wave
        kx = -k*cos(phik)*sin(thetak);
        ky = -k*sin(phik)*sin(thetak);
        kz = -k*cos(thetak);

        p0 = 1; % Amplitude of the incident wave


        %%%%%%% First Subscript:
        %%%%%%% 1 = Orthogonal Probe
        %%%%%%% 2 = Regular Tetrahedral Probe
        %%%%%%% 3 = 6 Microphone Probe
        %%%%%%% 4 = GA Probe

        % Orthogonal Probe dot products
        rk11 = -(-1/2*(cos(mictheta1(1)-thetak)*(1+cos(micphi1(1)-...
            phik))+cos(mictheta1(1)+thetak)*(1-cos(micphi1(1)-phik))));
        rk12 = -(-1/2*(cos(mictheta1(2)-thetak)*(1+cos(micphi1(2)-...
            phik))+cos(mictheta1(2)+thetak)*(1-cos(micphi1(2)-phik))));
        rk13 = -(-1/2*(cos(mictheta1(3)-thetak)*(1+cos(micphi1(3)-...
            phik))+cos(mictheta1(3)+thetak)*(1-cos(micphi1(3)-phik))));
        rk14 = -(-1/2*(cos(mictheta1(4)-thetak)*(1+cos(micphi1(4)-...
            phik))+cos(mictheta1(4)+thetak)*(1-cos(micphi1(4)-phik))));

        % Regular Probe Tetrahedronn dot products
        rk21 = -(-1/2*(cos(mictheta2(1)-thetak)*(1+cos(micphi2(1)-...
            phik))+cos(mictheta2(1)+thetak)*(1-cos(micphi2(1)-phik))));
        rk22 = -(-1/2*(cos(mictheta2(2)-thetak)*(1+cos(micphi2(2)-...
            phik))+cos(mictheta2(2)+thetak)*(1-cos(micphi2(2)-phik))));
        rk23 = -(-1/2*(cos(mictheta2(3)-thetak)*(1+cos(micphi2(3)-...
            phik))+cos(mictheta2(3)+thetak)*(1-cos(micphi2(3)-phik))));
        rk24 = -(-1/2*(cos(mictheta2(4)-thetak)*(1+cos(micphi2(4)-...
            phik))+cos(mictheta2(4)+thetak)*(1-cos(micphi2(4)-phik))));

        % 6 Microphone Probe dot products
        rk31 = -(-1/2*(cos(mictheta3(1)-thetak)*(1+cos(micphi3(1)-...
            phik))+cos(mictheta3(1)+thetak)*(1-cos(micphi3(1)-phik))));
        rk32 = -(-1/2*(cos(mictheta3(2)-thetak)*(1+cos(micphi3(2)-...
            phik))+cos(mictheta3(2)+thetak)*(1-cos(micphi3(2)-phik))));
        rk33 = -(-1/2*(cos(mictheta3(3)-thetak)*(1+cos(micphi3(3)-...
            phik))+cos(mictheta3(3)+thetak)*(1-cos(micphi3(3)-phik))));
        rk34 = -(-1/2*(cos(mictheta3(4)-thetak)*(1+cos(micphi3(4)-...
            phik))+cos(mictheta3(4)+thetak)*(1-cos(micphi3(4)-phik))));
        rk35 = -(-1/2*(cos(mictheta3(5)-thetak)*(1+cos(micphi3(5)-...
            phik))+cos(mictheta3(5)+thetak)*(1-cos(micphi3(5)-phik))));
        rk36 = -(-1/2*(cos(mictheta3(6)-thetak)*(1+cos(micphi3(6)-...
            phik))+cos(mictheta3(6)+thetak)*(1-cos(micphi3(6)-phik))));

        % Genetic Algorithm dot products
```

```
rk41 = -(-1/2*(cos(mictheta4(1)-thetak)*(1+cos(micphi4(1)-...
    phik))+cos(mictheta4(1)+thetak)*(1-cos(micphi4(1)-phik))));
rk42 = -(-1/2*(cos(mictheta4(2)-thetak)*(1+cos(micphi4(2)-...
    phik))+cos(mictheta4(2)+thetak)*(1-cos(micphi4(2)-phik))));
rk43 = -(-1/2*(cos(mictheta4(3)-thetak)*(1+cos(micphi4(3)-...
    phik))+cos(mictheta4(3)+thetak)*(1-cos(micphi4(3)-phik))));
rk44 = -(-1/2*(cos(mictheta4(4)-thetak)*(1+cos(micphi4(4)-...
    phik))+cos(mictheta4(4)+thetak)*(1-cos(micphi4(4)-phik))));

% Calculate the incident pressures (pi1-pi4) and the first
% terms of the scattered pressures (ps1-ps4)
% Equation for plane wave: p=p0*exp(-j*k*r)
pi11 = p0*exp(1i*k*a*rk11);
pi12 = p0*exp(1i*k*a*rk12);
pi13 = p0*exp(1i*k*a*rk13);
pi14 = p0*exp(1i*k*a*rk14);

pi21 = p0*exp(1i*k*a*rk21);
pi22 = p0*exp(1i*k*a*rk22);
pi23 = p0*exp(1i*k*a*rk23);
pi24 = p0*exp(1i*k*a*rk24);

pi31 = p0*exp(1i*k*a*rk31);
pi32 = p0*exp(1i*k*a*rk32);
pi33 = p0*exp(1i*k*a*rk33);
pi34 = p0*exp(1i*k*a*rk34);
pi35 = p0*exp(1i*k*a*rk35);
pi36 = p0*exp(1i*k*a*rk36);

pi41 = p0*exp(1i*k*a*rk41);
pi42 = p0*exp(1i*k*a*rk42);
pi43 = p0*exp(1i*k*a*rk43);
pi44 = p0*exp(1i*k*a*rk44);

% Initialize the first two Legendre polynomials for the
% recurrence relations
% "mm2" means "order m minus 2"
P11mm2 = 1;
P12mm2 = 1;
P13mm2 = 1;
P14mm2 = 1;

P21mm2 = 1;
P22mm2 = 1;
P23mm2 = 1;
P24mm2 = 1;

P31mm2 = 1;
P32mm2 = 1;
P33mm2 = 1;
P34mm2 = 1;
P35mm2 = 1;
P36mm2 = 1;

P41mm2 = 1;
```

```
P42mm2 = 1;
P43mm2 = 1;
P44mm2 = 1;

P11mm1 = rk11;
P12mm1 = rk12;
P13mm1 = rk13;
P14mm1 = rk14;

P21mm1 = rk21;
P22mm1 = rk22;
P23mm1 = rk23;
P24mm1 = rk24;

P31mm1 = rk31;
P32mm1 = rk32;
P33mm1 = rk33;
P34mm1 = rk34;
P35mm1 = rk35;
P36mm1 = rk36;

P41mm1 = rk41;
P42mm1 = rk42;
P43mm1 = rk43;
P44mm1 = rk44;


m = 0;  % initial order for Bessel functions
% Calculate the necessary Bessel and Hankel functions
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a); % order m minus 1
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a); % order m plus 1
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));

Am = (-1i)^m*(2*m+1)/dhm;
p11 = Am*P11mm2;
p12 = Am*P12mm2;
p13 = Am*P13mm2;
p14 = Am*P14mm2;

p21 = Am*P21mm2;
p22 = Am*P22mm2;
p23 = Am*P23mm2;
p24 = Am*P24mm2;

p31 = Am*P31mm2;
p32 = Am*P32mm2;
p33 = Am*P33mm2;
p34 = Am*P34mm2;
p35 = Am*P35mm2;
p36 = Am*P36mm2;

p41 = Am*P41mm2;
```

```
p42 = Am*P42mm2;
p43 = Am*P43mm2;
p44 = Am*P44mm2;

m = 1;  % change order for Bessel functions
% Calculate the necessary Bessel and Hankel functions
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));
Am = (-1i)^m*(2*m+1)/dhm;

% now add results from order m=0 results to those of m=1
p11 = p11+Am*P11mm1;
p12 = p12+Am*P12mm1;
p13 = p13+Am*P13mm1;
p14 = p14+Am*P14mm1;

p21 = p21+Am*P21mm1;
p22 = p22+Am*P22mm1;
p23 = p23+Am*P23mm1;
p24 = p24+Am*P24mm1;

p31 = p31+Am*P31mm1;
p32 = p32+Am*P32mm1;
p33 = p33+Am*P33mm1;
p34 = p34+Am*P34mm1;
p35 = p35+Am*P35mm1;
p36 = p36+Am*P36mm1;

p41 = p41+Am*P41mm1;
p42 = p42+Am*P42mm1;
p43 = p43+Am*P43mm1;
p44 = p44+Am*P44mm1;

% Now that you have 2 calculated, recurrence relations can be
% used to calculate the rest.
for m = 2:M;
    % Use a recurrence relation to calculate the Legendre
    % polynomials
    P11m = 2*rk11*P11mm1-P11mm2-1/m*(rk11*P11mm1-P11mm2);
    P12m = 2*rk12*P12mm1-P12mm2-1/m*(rk12*P12mm1-P12mm2);
    P13m = 2*rk13*P13mm1-P13mm2-1/m*(rk13*P13mm1-P13mm2);
    P14m = 2*rk14*P14mm1-P14mm2-1/m*(rk14*P14mm1-P14mm2);

    P21m = 2*rk21*P21mm1-P21mm2-1/m*(rk21*P21mm1-P21mm2);
    P22m = 2*rk22*P22mm1-P22mm2-1/m*(rk22*P22mm1-P22mm2);
    P23m = 2*rk23*P23mm1-P23mm2-1/m*(rk23*P23mm1-P23mm2);
    P24m = 2*rk24*P24mm1-P24mm2-1/m*(rk24*P24mm1-P24mm2);

    P31m = 2*rk31*P31mm1-P31mm2-1/m*(rk31*P31mm1-P31mm2);
    P32m = 2*rk32*P32mm1-P32mm2-1/m*(rk32*P32mm1-P32mm2);
    P33m = 2*rk33*P33mm1-P33mm2-1/m*(rk33*P33mm1-P33mm2);
    P34m = 2*rk34*P34mm1-P34mm2-1/m*(rk34*P34mm1-P34mm2);
```

```
P35m = 2*rk35*P35mm1-P35mm2-1/m*(rk35*P35mm1-P35mm2);
P36m = 2*rk36*P36mm1-P36mm2-1/m*(rk36*P36mm1-P36mm2);

P41m = 2*rk41*P41mm1-P41mm2-1/m*(rk41*P41mm1-P41mm2);
P42m = 2*rk42*P42mm1-P42mm2-1/m*(rk42*P42mm1-P42mm2);
P43m = 2*rk43*P43mm1-P43mm2-1/m*(rk43*P43mm1-P43mm2);
P44m = 2*rk44*P44mm1-P44mm2-1/m*(rk44*P44mm1-P44mm2);

% Calculate the necessary Bessel and Hankel functions
jmm1 = sqrt(pi/(2*k*a))*besselj(m+1/2-1,k*a);
jmp1 = sqrt(pi/(2*k*a))*besselj(m+1/2+1,k*a);
ymm1 = sqrt(pi/(2*k*a))*bessely(m+1/2-1,k*a);
ymp1 = sqrt(pi/(2*k*a))*bessely(m+1/2+1,k*a);
dhm = 1/(2*m+1)*(m*(jmm1+1i*ymm1)-(m+1)*(jmp1+1i*ymp1));
%           Am = p0*(2*m+1)*1i^m*h2m*djm/dhm;
Am = (-1i)^m*(2*m+1)/dhm;

p11 = p11+Am*P11m;
p12 = p12+Am*P12m;
p13 = p13+Am*P13m;
p14 = p14+Am*P14m;

p21 = p21+Am*P21m;
p22 = p22+Am*P22m;
p23 = p23+Am*P23m;
p24 = p24+Am*P24m;

p31 = p31+Am*P31m;
p32 = p32+Am*P32m;
p33 = p33+Am*P33m;
p34 = p34+Am*P34m;
p35 = p35+Am*P35m;
p36 = p36+Am*P36m;

p41 = p41+Am*P41m;
p42 = p42+Am*P42m;
p43 = p43+Am*P43m;
p44 = p44+Am*P44m;

% Update the Legendre polynomials for the recurrence
% relation
P11mm2 = P11mm1;
P12mm2 = P12mm1;
P13mm2 = P13mm1;
P14mm2 = P14mm1;

P21mm2 = P21mm1;
P22mm2 = P22mm1;
P23mm2 = P23mm1;
P24mm2 = P24mm1;

P31mm2 = P31mm1;
P32mm2 = P32mm1;
P33mm2 = P33mm1;
```

```
    P34mm2 = P34mm1;
    P35mm2 = P35mm1;
    P36mm2 = P36mm1;

    P41mm2 = P41mm1;
    P42mm2 = P42mm1;
    P43mm2 = P43mm1;
    P44mm2 = P44mm1;

    P11mm1 = P11m;
    P12mm1 = P12m;
    P13mm1 = P13m;
    P14mm1 = P14m;

    P21mm1 = P21m;
    P22mm1 = P22m;
    P23mm1 = P23m;
    P24mm1 = P24m;

    P31mm1 = P31m;
    P32mm1 = P32m;
    P33mm1 = P33m;
    P34mm1 = P34m;
    P35mm1 = P35m;
    P36mm1 = P36m;

    P41mm1 = P41m;
    P42mm1 = P42m;
    P43mm1 = P43m;
    P44mm1 = P44m;
end

% total pressure = incident + scattered pressures
p11 = conj(p11*1i*p0/(k*a)^2);
p12 = conj(p12*1i*p0/(k*a)^2);
p13 = conj(p13*1i*p0/(k*a)^2);
p14 = conj(p14*1i*p0/(k*a)^2);

p21 = conj(p21*1i*p0/(k*a)^2);
p22 = conj(p22*1i*p0/(k*a)^2);
p23 = conj(p23*1i*p0/(k*a)^2);
p24 = conj(p24*1i*p0/(k*a)^2);

p31 = conj(p31*1i*p0/(k*a)^2);
p32 = conj(p32*1i*p0/(k*a)^2);
p33 = conj(p33*1i*p0/(k*a)^2);
p34 = conj(p34*1i*p0/(k*a)^2);
p35 = conj(p35*1i*p0/(k*a)^2);
p36 = conj(p36*1i*p0/(k*a)^2);

p41 = conj(p41*1i*p0/(k*a)^2);
p42 = conj(p42*1i*p0/(k*a)^2);
p43 = conj(p43*1i*p0/(k*a)^2);
p44 = conj(p44*1i*p0/(k*a)^2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%        EXACT CALCULATIONS      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% particle velocity
vx = kx/rho/w*p0;
vy = ky/rho/w*p0;
vz = kz/rho/w*p0;
v(o,n) = (abs(vx).^2+abs(vy).^2+abs(vz).^2).^(1/2);

% Intensity calculations
Ix = 0.5*real(p0*conj(vx));
Iy = 0.5*real(p0*conj(vy));
Iz = 0.5*real(p0*conj(vz));
I(o,n) = (abs(Ix).^2+abs(Iy).^2+abs(Iz).^2).^(1/2);

% Energy density
W(o,n) = rho*abs(v(o,n))^2/4 + abs(p0)^2/4/rho/c^2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%        FINITE-DIFF CALCULATIONS   %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%% Pressure Calculations
%%%%%%% 1st index = probe type
%%%%%%% 2nd subscript = pressure estimation type

pi_avg12 = pi13;
p_avg12 = p13;
pi_avg13 = (pi11+pi12+pi13+pi14)/4;
p_avg13 = (p11+p12+p13+p14)/4;
pi_avg14 = ((pi11+pi13)/2+(pi12+pi13)/2+(pi14+pi13)/2)/3;
p_avg14 = ((p11+p13)/2+(p12+p13)/2+(p14+p13)/2)/3;

pi_avg21 = (pi21+pi22+pi23+pi24)/4;
p_avg21 = (p21+p22+p23+p24)/4;
pi_avg22 = pi21;
p_avg22 = p21;

pi_avg32 = (pi31+pi32+pi33+pi34+pi35+pi36)/6;
p_avg32 = (p31+p32+p33+p34+p35+p36)/6;
pi_avg33 = pi31;
p_avg33 = p31;

pi_avg41 = (pi41+pi42+pi43+pi44)/4;
p_avg41 = (p41+p42+p43+p44)/4;
pi_avg42 = pi41;
p_avg42 = p41;
```

```
Pi12(o,n) = abs(pi_avg12);
P12(o,n) = abs(p_avg12);
Pi13(o,n) = abs(pi_avg13);
P13(o,n) = abs(p_avg13);
Pi14(o,n) = abs(pi_avg14);
P14(o,n) = abs(p_avg14);

Pi21(o,n) = abs(pi_avg21);
P21(o,n) = abs(p_avg21);
Pi22(o,n) = abs(pi_avg22);
P22(o,n) = abs(p_avg22);

Pi32(o,n) = abs(pi_avg32);
P32(o,n) = abs(p_avg32);
Pi33(o,n) = abs(pi_avg33);
P33(o,n) = abs(p_avg33);

Pi41(o,n) = abs(pi_avg41);
P41(o,n) = abs(p_avg41);
Pi42(o,n) = abs(pi_avg42);
P42(o,n) = abs(p_avg42);


%%%%%%% Velocity Calculations
%%%%%%% 1st subscript: probe type
%%%%%%% 2nd: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%          sphere w/o 3/2 factor
%%%%%%% 3rd: 1 = estimation at three points, 2 = estimation
%%%%%%%          at origin using Taylor expansion

vx111=1i/rho/(2*a/sqrt(3))/w*(pi11-pi13);
vy111=1i/rho/(2*a/sqrt(3))/w*(pi12-pi13);
vz111=1i/rho/(2*a/sqrt(3))/w*(pi14-pi13);
v111(o,n) = (abs(vx111).^2+abs(vy111).^2+abs(vz111).^2).^(1/2);
vx121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p11-p13);
vy121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p12-p13);
vz121=1i/rho/(2*a*(3/2)/sqrt(3))/w*(p14-p13);
v121(o,n) = (abs(vx121).^2+abs(vy121).^2+abs(vz121).^2).^(1/2);
vx131=1i/rho/(2*a/sqrt(3))/w*(p11-p13);
vy131=1i/rho/(2*a/sqrt(3))/w*(p12-p13);
vz131=1i/rho/(2*a/sqrt(3))/w*(p14-p13);
v131(o,n) = (abs(vx131).^2+abs(vy131).^2+abs(vz131).^2).^(1/2);

vx211=1i/rho/4/a/w*sqrt(2)*(pi22-2*pi23+pi24);
vy211=1i/rho/4/a/w*sqrt(6)*(-pi22+pi24);
vz211=1i/rho/4/a/w*(3*pi21-pi22-pi23-pi24);
v211(o,n) = (abs(vx211).^2+abs(vy211).^2+abs(vz211).^2).^(1/2);
vx221=1i/rho/4/(a*3/2)/w*sqrt(2)*(p22-2*p23+p24);
vy221=1i/rho/4/(a*3/2)/w*sqrt(6)*(-p22+p24);
vz221=1i/rho/4/(a*3/2)/w*(3*p21-p22-p23-p24);
v221(o,n) = (abs(vx221).^2+abs(vy221).^2+abs(vz221).^2).^(1/2);
vx231=1i/rho/4/a/w*sqrt(2)*(p22-2*p23+p24);
vy231=1i/rho/4/a/w*sqrt(6)*(-p22+p24);
vz231=1i/rho/4/a/w*(3*p21-p22-p23-p24);
v231(o,n) = (abs(vx231).^2+abs(vy231).^2+abs(vz231).^2).^(1/2);
```

```
vx311=1i/rho/(2*a)/w*(pi32-pi34);
vy311=1i/rho/(2*a)/w*(pi33-pi35);
vz311=1i/rho/(2*a)/w*(pi31-pi36);
v311(o,n) = (abs(vx311).^2+abs(vy311).^2+abs(vz311).^2).^(1/2);
vx321=1i/rho/(2*a*3/2)/w*(p32-p34);
vy321=1i/rho/(2*a*3/2)/w*(p33-p35);
vz321=1i/rho/(2*a*3/2)/w*(p31-p36);
v321(o,n) = (abs(vz321).^2+abs(vy321).^2+abs(vx321).^2).^(1/2);
vx331=1i/rho/(2*a)/w*(p32-p34);
vy331=1i/rho/(2*a)/w*(p33-p35);
vz331=1i/rho/(2*a)/w*(p31-p36);
v331(o,n) = (abs(vx331).^2+abs(vy331).^2+abs(vz331).^2).^(1/2);

[vx112 vy112 vz112 v112(o,n)] = ...
    Taylor_Velocity(pi11,pi12,pi13,pi14,w,rho,a,1);
[vx122 vy122 vz122 v122(o,n)] = ...
    Taylor_Velocity(p11,p12,p13,p14,w,rho,a,3/2);
[vx132 vy132 vz132 v132(o,n)] = ...
    Taylor_Velocity(p11,p12,p13,p14,w,rho,a,1);

delpi = [pi42-pi41 pi43-pi41 pi44-pi41...
    pi43-pi42 pi44-pi42 pi44-pi43].';
delp = [p42-p41 p43-p41 p44-p41 p43-p42 p44-p42 p44-p43].';
gradpi = R\delpi;
gradp = R\delp;
vx411=1i/rho/(a)/w*gradpi(1);
vy411=1i/rho/(a)/w*gradpi(2);
vz411=1i/rho/(a)/w*gradpi(3);
v411(o,n) = (abs(vx411).^2+abs(vy411).^2+abs(vz411).^2).^(1/2);
vx421=1i/rho/(a*3/2)/w*gradp(1);
vy421=1i/rho/(a*3/2)/w*gradp(2);
vz421=1i/rho/(a*3/2)/w*gradp(3);
v421(o,n) = (abs(vx421).^2+abs(vy421).^2+abs(vz421).^2).^(1/2);
vx431=1i/rho/(a)/w*gradp(1);
vy431=1i/rho/(a)/w*gradp(2);
vz431=1i/rho/(a)/w*gradp(3);
v431(o,n) = (abs(vx431).^2+abs(vy431).^2+abs(vz431).^2).^(1/2);


%%%%%%% Intensity Calculations
%%%%%%% 1st subscript: type of probe
%%%%%%% 2nd subscript: how p was calculated
%%%%%%% 3rd subscript: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%                sphere w/o 3/2 factor
%%%%%%% 4th subscript: 1 = no Taylor, 2 = with Taylor

Ix1111 = 0.5*real((pi11+pi13)/2*conj(vx111));
Iy1111 = 0.5*real((pi12+pi13)/2*conj(vy111));
Iz1111 = 0.5*real((pi14+pi13)/2*conj(vz111));
I1111(o,n) = ...
    (abs(Ix1111).^2+abs(Iy1111).^2+abs(Iz1111).^2).^(1/2);
Ix1121 = 0.5*real((p11+p13)/2*conj(vx121));
Iy1121 = 0.5*real((p12+p13)/2*conj(vy121));
Iz1121 = 0.5*real((p14+p13)/2*conj(vz121));
```

```
I1121(o,n) = ...
    (abs(Ix1121).^2+abs(Iy1121).^2+abs(Iz1121).^2).^(1/2);
Ix1131 = 0.5*real((p11+p13)/2*conj(vx131));
Iy1131 = 0.5*real((p12+p13)/2*conj(vy131));
Iz1131 = 0.5*real((p14+p13)/2*conj(vz131));
I1131(o,n) = ...
    (abs(Ix1131).^2+abs(Iy1131).^2+abs(Iz1131).^2).^(1/2);
[Ix1211 Iy1211 Iz1211 I1211(o,n)] = ...
    Intensity_Calc(pi_avg12,vx111,vy111,vz111);
[Ix1221 Iy1221 Iz1221 I1221(o,n)] = ...
    Intensity_Calc(p_avg12,vx121,vy121,vz121);
[Ix1231 Iy1231 Iz1231 I1231(o,n)] = ...
    Intensity_Calc(p_avg12,vx131,vy131,vz131);
[Ix1311 Iy1311 Iz1311 I1311(o,n)] = ...
    Intensity_Calc(pi_avg13,vx111,vy111,vz111);
[Ix1321 Iy1321 Iz1321 I1321(o,n)] = ...
    Intensity_Calc(p_avg13,vx121,vy121,vz121);
[Ix1331 Iy1331 Iz1331 I1331(o,n)] = ...
    Intensity_Calc(p_avg13,vx131,vy131,vz131);
[Ix1411 Iy1411 Iz1411 I1411(o,n)] = ...
    Intensity_Calc(pi_avg14,vx111,vy111,vz111);
[Ix1421 Iy1421 Iz1421 I1421(o,n)] = ...
    Intensity_Calc(p_avg14,vx121,vy121,vz121);
[Ix1431 Iy1431 Iz1431 I1431(o,n)] = ...
    Intensity_Calc(p_avg14,vx131,vy131,vz131);

[Ix2111 Iy2111 Iz2111 I2111(o,n)] = ...
    Intensity_Calc(pi_avg21,vx211,vy211,vz211);
[Ix2121 Iy2121 Iz2121 I2121(o,n)] = ...
    Intensity_Calc(p_avg21,vx221,vy221,vz221);
[Ix2131 Iy2131 Iz2131 I2131(o,n)] = ...
    Intensity_Calc(p_avg21,vx231,vy231,vz231);
[Ix2211 Iy2211 Iz2211 I2211(o,n)] = ...
    Intensity_Calc(pi_avg22,vx211,vy211,vz211);
[Ix2221 Iy2221 Iz2221 I2221(o,n)] = ...
    Intensity_Calc(p_avg22,vx221,vy221,vz221);
[Ix2231 Iy2231 Iz2231 I2231(o,n)] = ...
    Intensity_Calc(p_avg22,vx231,vy231,vz231);

Ix3111 = 0.5*real((pi32+pi34)/2*conj(vx311));
Iy3111 = 0.5*real((pi33+pi35)/2*conj(vy311));
Iz3111 = 0.5*real((pi31+pi36)/2*conj(vz311));
I3111(o,n) = ...
    (abs(Ix3111).^2+abs(Iy3111).^2+abs(Iz3111).^2).^(1/2);
Ix3121 = 0.5*real((p32+p34)/2*conj(vx321));
Iy3121 = 0.5*real((p33+p35)/2*conj(vy321));
Iz3121 = 0.5*real((p31+p36)/2*conj(vz321));
I3121(o,n) = ...
    (abs(Ix3121).^2+abs(Iy3121).^2+abs(Iz3121).^2).^(1/2);
Ix3131 = 0.5*real((p32+p34)/2*conj(vx331));
Iy3131 = 0.5*real((p33+p35)/2*conj(vy331));
Iz3131 = 0.5*real((p31+p36)/2*conj(vz331));
I3131(o,n) = ...
    (abs(Ix3131).^2+abs(Iy3131).^2+abs(Iz3131).^2).^(1/2);
[Ix3211 Iy3211 Iz3211 I3211(o,n)] = ...
    Intensity_Calc(pi_avg32,vx311,vy311,vz311);
```

```
[Ix3221 Iy3221 Iz3221 I3221(o,n)] = ...
    Intensity_Calc(p_avg32,vx321,vy321,vz321);
[Ix3231 Iy3231 Iz3231 I3231(o,n)] = ...
    Intensity_Calc(p_avg32,vx331,vy331,vz331);
[Ix3311 Iy3311 Iz3311 I3311(o,n)] = ...
    Intensity_Calc(pi_avg33,vx311,vy311,vz311);
[Ix3321 Iy3321 Iz3321 I3321(o,n)] = ...
    Intensity_Calc(p_avg33,vx321,vy321,vz321);
[Ix3331 Iy3331 Iz3331 I3331(o,n)] = ...
    Intensity_Calc(p_avg33,vx331,vy331,vz331);

Ix1112 = 0.5*real((pi11+pi13)/2*conj(vx112));
Iy1112 = 0.5*real((pi12+pi13)/2*conj(vy112));
Iz1112 = 0.5*real((pi14+pi13)/2*conj(vz112));
I1112(o,n) = ...
    (abs(Ix1112).^2+abs(Iy1112).^2+abs(Iz1112).^2).^(1/2);
Ix1122 = 0.5*real((p11+p13)/2*conj(vx122));
Iy1122 = 0.5*real((p12+p13)/2*conj(vy122));
Iz1122 = 0.5*real((p14+p13)/2*conj(vz122));
I1122(o,n) = ...
    (abs(Ix1122).^2+abs(Iy1122).^2+abs(Iz1122).^2).^(1/2);
Ix1132 = 0.5*real((p11+p13)/2*conj(vx132));
Iy1132 = 0.5*real((p12+p13)/2*conj(vy132));
Iz1132 = 0.5*real((p14+p13)/2*conj(vz132));
I1132(o,n) = ...
    (abs(Ix1132).^2+abs(Iy1132).^2+abs(Iz1132).^2).^(1/2);
[Ix1212 Iy1212 Iz1212 I1212(o,n)] = ...
    Intensity_Calc(pi_avg12,vx112,vy112,vz112);
[Ix1222 Iy1222 Iz1222 I1222(o,n)] = ...
    Intensity_Calc(p_avg12,vx122,vy122,vz122);
[Ix1232 Iy1232 Iz1232 I1232(o,n)] = ...
    Intensity_Calc(p_avg12,vx132,vy132,vz132);
[Ix1312 Iy1312 Iz1312 I1312(o,n)] = ...
    Intensity_Calc(pi_avg13,vx112,vy112,vz112);
[Ix1322 Iy1322 Iz1322 I1322(o,n)] = ...
    Intensity_Calc(p_avg13,vx122,vy122,vz122);
[Ix1332 Iy1332 Iz1332 I1332(o,n)] = ...
    Intensity_Calc(p_avg13,vx132,vy132,vz132);
[Ix1412 Iy1412 Iz1412 I1412(o,n)] = ...
    Intensity_Calc(pi_avg14,vx112,vy112,vz112);
[Ix1422 Iy1422 Iz1422 I1422(o,n)] = ...
    Intensity_Calc(p_avg14,vx122,vy122,vz122);
[Ix1432 Iy1432 Iz1432 I1432(o,n)] = ...
    Intensity_Calc(p_avg14,vx132,vy132,vz132);

[Ix4111 Iy4111 Iz4111 I4111(o,n)] = ...
    Intensity_Calc(pi_avg41,vx411,vy411,vz411);
[Ix4121 Iy4121 Iz4121 I4121(o,n)] = ...
    Intensity_Calc(p_avg41,vx421,vy421,vz421);
[Ix4131 Iy4131 Iz4131 I4131(o,n)] = ...
    Intensity_Calc(p_avg41,vx431,vy431,vz431);
[Ix4211 Iy4211 Iz4211 I4211(o,n)] = ...
    Intensity_Calc(pi_avg42,vx411,vy411,vz411);
[Ix4221 Iy4221 Iz4221 I4221(o,n)] = ...
    Intensity_Calc(p_avg42,vx421,vy421,vz421);
[Ix4231 Iy4231 Iz4231 I4231(o,n)] = ...
```

```
    Intensity_Calc(p_avg42,vx431,vy431,vz431);


%%%%%%% Energy Density Calculations
%%%%%%% 1st subscript: type of probe
%%%%%%% 2nd subscript: how p was calculated
%%%%%%% 3rd subscript: 1 = no sphere, 2 = with sphere, 3 = with
%%%%%%%                sphere w/o 3/2 factor
%%%%%%% 4th subscript: 1 = no Taylor, 2 = with Taylor

W1211(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg12)^2/4/rho/c^2;
W1221(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1231(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1311(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg13)^2/4/rho/c^2;
W1321(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1331(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1411(o,n) = rho*abs(v111(o,n))^2/4 + abs(pi_avg14)^2/4/rho/c^2;
W1421(o,n) = rho*abs(v121(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1431(o,n) = rho*abs(v131(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W2111(o,n) = rho*abs(v211(o,n))^2/4 + abs(pi_avg21)^2/4/rho/c^2;
W2121(o,n) = rho*abs(v221(o,n))^2/4 + abs(p_avg21)^2/4/rho/c^2;
W2131(o,n) = rho*abs(v231(o,n))^2/4 + abs(p_avg21)^2/4/rho/c^2;
W2211(o,n) = rho*abs(v211(o,n))^2/4 + abs(pi_avg22)^2/4/rho/c^2;
W2221(o,n) = rho*abs(v221(o,n))^2/4 + abs(p_avg22)^2/4/rho/c^2;
W2231(o,n) = rho*abs(v231(o,n))^2/4 + abs(p_avg22)^2/4/rho/c^2;
W3211(o,n) = rho*abs(v311(o,n))^2/4 + abs(pi_avg32)^2/4/rho/c^2;
W3221(o,n) = rho*abs(v321(o,n))^2/4 + abs(p_avg32)^2/4/rho/c^2;
W3231(o,n) = rho*abs(v331(o,n))^2/4 + abs(p_avg32)^2/4/rho/c^2;
W3311(o,n) = rho*abs(v311(o,n))^2/4 + abs(pi_avg33)^2/4/rho/c^2;
W3321(o,n) = rho*abs(v321(o,n))^2/4 + abs(p_avg33)^2/4/rho/c^2;
W3331(o,n) = rho*abs(v331(o,n))^2/4 + abs(p_avg33)^2/4/rho/c^2;
W1212(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg12)^2/4/rho/c^2;
W1222(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1232(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg12)^2/4/rho/c^2;
W1312(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg13)^2/4/rho/c^2;
W1322(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1332(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg13)^2/4/rho/c^2;
W1412(o,n) = rho*abs(v112(o,n))^2/4 + abs(pi_avg14)^2/4/rho/c^2;
W1422(o,n) = rho*abs(v122(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W1432(o,n) = rho*abs(v132(o,n))^2/4 + abs(p_avg14)^2/4/rho/c^2;
W4111(o,n) = rho*abs(v411(o,n))^2/4 + abs(pi_avg41)^2/4/rho/c^2;
W4121(o,n) = rho*abs(v421(o,n))^2/4 + abs(p_avg41)^2/4/rho/c^2;
W4131(o,n) = rho*abs(v431(o,n))^2/4 + abs(p_avg41)^2/4/rho/c^2;
W4211(o,n) = rho*abs(v411(o,n))^2/4 + abs(pi_avg42)^2/4/rho/c^2;
W4221(o,n) = rho*abs(v421(o,n))^2/4 + abs(p_avg42)^2/4/rho/c^2;
W4231(o,n) = rho*abs(v431(o,n))^2/4 + abs(p_avg42)^2/4/rho/c^2;



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%        DIRECTION CALCULATIONS      %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
Ivect = [Ix Iy Iz];
vvect = [vx vy vz];

I1111ang(o,n,q) = Angle_Calc(Ivect,Ix1111,Iy1111,Iz1111);
I1121ang(o,n,q) = Angle_Calc(Ivect,Ix1121,Iy1121,Iz1121);
I1131ang(o,n,q) = Angle_Calc(Ivect,Ix1131,Iy1131,Iz1131);
I1211ang(o,n,q) = Angle_Calc(Ivect,Ix1211,Iy1211,Iz1211);
I1221ang(o,n,q) = Angle_Calc(Ivect,Ix1221,Iy1221,Iz1221);
I1231ang(o,n,q) = Angle_Calc(Ivect,Ix1231,Iy1231,Iz1231);
I1311ang(o,n,q) = Angle_Calc(Ivect,Ix1311,Iy1311,Iz1311);
I1321ang(o,n,q) = Angle_Calc(Ivect,Ix1321,Iy1321,Iz1321);
I1331ang(o,n,q) = Angle_Calc(Ivect,Ix1331,Iy1331,Iz1331);
I1411ang(o,n,q) = Angle_Calc(Ivect,Ix1411,Iy1411,Iz1411);
I1421ang(o,n,q) = Angle_Calc(Ivect,Ix1421,Iy1421,Iz1421);
I1431ang(o,n,q) = Angle_Calc(Ivect,Ix1431,Iy1431,Iz1431);

I2111ang(o,n,q) = Angle_Calc(Ivect,Ix2111,Iy2111,Iz2111);
I2121ang(o,n,q) = Angle_Calc(Ivect,Ix2121,Iy2121,Iz2121);
I2131ang(o,n,q) = Angle_Calc(Ivect,Ix2131,Iy2131,Iz2131);
I2211ang(o,n,q) = Angle_Calc(Ivect,Ix2211,Iy2211,Iz2211);
I2221ang(o,n,q) = Angle_Calc(Ivect,Ix2221,Iy2221,Iz2221);
I2231ang(o,n,q) = Angle_Calc(Ivect,Ix2231,Iy2231,Iz2231);

I3111ang(o,n,q) = Angle_Calc(Ivect,Ix3111,Iy3111,Iz3111);
I3121ang(o,n,q) = Angle_Calc(Ivect,Ix3121,Iy3121,Iz3121);
I3131ang(o,n,q) = Angle_Calc(Ivect,Ix3131,Iy3131,Iz3131);
I3211ang(o,n,q) = Angle_Calc(Ivect,Ix3211,Iy3211,Iz3211);
I3221ang(o,n,q) = Angle_Calc(Ivect,Ix3221,Iy3221,Iz3221);
I3231ang(o,n,q) = Angle_Calc(Ivect,Ix3231,Iy3231,Iz3231);
I3311ang(o,n,q) = Angle_Calc(Ivect,Ix3311,Iy3311,Iz3311);
I3321ang(o,n,q) = Angle_Calc(Ivect,Ix3321,Iy3321,Iz3321);
I3331ang(o,n,q) = Angle_Calc(Ivect,Ix3331,Iy3331,Iz3331);

I1112ang(o,n,q) = Angle_Calc(Ivect,Ix1112,Iy1112,Iz1112);
I1122ang(o,n,q) = Angle_Calc(Ivect,Ix1122,Iy1122,Iz1122);
I1132ang(o,n,q) = Angle_Calc(Ivect,Ix1132,Iy1132,Iz1132);
I1212ang(o,n,q) = Angle_Calc(Ivect,Ix1212,Iy1212,Iz1212);
I1222ang(o,n,q) = Angle_Calc(Ivect,Ix1222,Iy1222,Iz1222);
I1232ang(o,n,q) = Angle_Calc(Ivect,Ix1232,Iy1232,Iz1232);
I1312ang(o,n,q) = Angle_Calc(Ivect,Ix1312,Iy1312,Iz1312);
I1322ang(o,n,q) = Angle_Calc(Ivect,Ix1322,Iy1322,Iz1322);
I1332ang(o,n,q) = Angle_Calc(Ivect,Ix1332,Iy1332,Iz1332);
I1412ang(o,n,q) = Angle_Calc(Ivect,Ix1412,Iy1412,Iz1412);
I1422ang(o,n,q) = Angle_Calc(Ivect,Ix1422,Iy1422,Iz1422);
I1432ang(o,n,q) = Angle_Calc(Ivect,Ix1432,Iy1432,Iz1432);

I4111ang(o,n,q) = Angle_Calc(Ivect,Ix4111,Iy4111,Iz4111);
I4121ang(o,n,q) = Angle_Calc(Ivect,Ix4121,Iy4121,Iz4121);
I4131ang(o,n,q) = Angle_Calc(Ivect,Ix4131,Iy4131,Iz4131);
I4211ang(o,n,q) = Angle_Calc(Ivect,Ix4211,Iy4211,Iz4211);
I4221ang(o,n,q) = Angle_Calc(Ivect,Ix4221,Iy4221,Iz4221);
I4231ang(o,n,q) = Angle_Calc(Ivect,Ix4231,Iy4231,Iz4231);

v111ang(o,n,q) = Angle_Calc(vvect,vx111,vy111,vz111);
```

```
        v121ang(o,n,q) = Angle_Calc(vvect,vx121,vy121,vz121);
        v131ang(o,n,q) = Angle_Calc(vvect,vx131,vy131,vz131);

        v211ang(o,n,q) = Angle_Calc(vvect,vx211,vy211,vz211);
        v221ang(o,n,q) = Angle_Calc(vvect,vx221,vy221,vz221);
        v231ang(o,n,q) = Angle_Calc(vvect,vx231,vy231,vz231);

        v311ang(o,n,q) = Angle_Calc(vvect,vx311,vy311,vz311);
        v321ang(o,n,q) = Angle_Calc(vvect,vx321,vy321,vz321);
        v331ang(o,n,q) = Angle_Calc(vvect,vx331,vy331,vz331);

        v112ang(o,n,q) = Angle_Calc(vvect,vx112,vy112,vz112);
        v122ang(o,n,q) = Angle_Calc(vvect,vx122,vy122,vz122);
        v132ang(o,n,q) = Angle_Calc(vvect,vx132,vy132,vz132);

        v411ang(o,n,q) = Angle_Calc(vvect,vx411,vy411,vz411);
        v421ang(o,n,q) = Angle_Calc(vvect,vx421,vy421,vz421);
        v431ang(o,n,q) = Angle_Calc(vvect,vx431,vy431,vz431);


        n = n+1;
    end
    o = o+1;
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Finite Diff Bias Calculations   %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[I1111b{1}(:,:,q) I1111b{2}(q) I1111b{3}(q) I1111b{4}(q) I1111b{5}(q)...
    I1111b{6}(q) I1111b{7}(q) I1111b{8}(q) I1111b{9}(q)] = ...
    Bias_Calc(res_step,I1111,I,10);
[I1121b{1}(:,:,q) I1121b{2}(q) I1121b{3}(q) I1121b{4}(q) I1121b{5}(q)...
    I1121b{6}(q) I1121b{7}(q) I1121b{8}(q) I1121b{9}(q)] = ...
    Bias_Calc(res_step,I1121,I,10);
[I1131b{1}(:,:,q) I1131b{2}(q) I1131b{3}(q) I1131b{4}(q) I1131b{5}(q)...
    I1131b{6}(q) I1131b{7}(q) I1131b{8}(q) I1131b{9}(q)] = ...
    Bias_Calc(res_step,I1131,I,10);
[I1211b{1}(:,:,q) I1211b{2}(q) I1211b{3}(q) I1211b{4}(q) I1211b{5}(q)...
    I1211b{6}(q) I1211b{7}(q) I1211b{8}(q) I1211b{9}(q)] = ...
    Bias_Calc(res_step,I1211,I,10);
[I1221b{1}(:,:,q) I1221b{2}(q) I1221b{3}(q) I1221b{4}(q) I1221b{5}(q)...
    I1221b{6}(q) I1221b{7}(q) I1221b{8}(q) I1221b{9}(q)] = ...
    Bias_Calc(res_step,I1221,I,10);
[I1231b{1}(:,:,q) I1231b{2}(q) I1231b{3}(q) I1231b{4}(q) I1231b{5}(q)...
    I1231b{6}(q) I1231b{7}(q) I1231b{8}(q) I1231b{9}(q)] = ...
    Bias_Calc(res_step,I1231,I,10);
[I1311b{1}(:,:,q) I1311b{2}(q) I1311b{3}(q) I1311b{4}(q) I1311b{5}(q)...
    I1311b{6}(q) I1311b{7}(q) I1311b{8}(q) I1311b{9}(q)] = ...
    Bias_Calc(res_step,I1311,I,10);
```

```
[I1321b{1}(:,:,q) I1321b{2}(q) I1321b{3}(q) I1321b{4}(q) I1321b{5}(q)...
    I1321b{6}(q) I1321b{7}(q) I1321b{8}(q) I1321b{9}(q)] = ...
    Bias_Calc(res_step,I1321,I,10);
[I1331b{1}(:,:,q) I1331b{2}(q) I1331b{3}(q) I1331b{4}(q) I1331b{5}(q)...
    I1331b{6}(q) I1331b{7}(q) I1331b{8}(q) I1331b{9}(q)] = ...
    Bias_Calc(res_step,I1331,I,10);
[I1411b{1}(:,:,q) I1411b{2}(q) I1411b{3}(q) I1411b{4}(q) I1411b{5}(q)...
    I1411b{6}(q) I1411b{7}(q) I1411b{8}(q) I1411b{9}(q)] = ...
    Bias_Calc(res_step,I1411,I,10);
[I1421b{1}(:,:,q) I1421b{2}(q) I1421b{3}(q) I1421b{4}(q) I1421b{5}(q)...
    I1421b{6}(q) I1421b{7}(q) I1421b{8}(q) I1421b{9}(q)] = ...
    Bias_Calc(res_step,I1421,I,10);
[I1431b{1}(:,:,q) I1431b{2}(q) I1431b{3}(q) I1431b{4}(q) I1431b{5}(q)...
    I1431b{6}(q) I1431b{7}(q) I1431b{8}(q) I1431b{9}(q)] = ...
    Bias_Calc(res_step,I1431,I,10);

[I2111b{1}(:,:,q) I2111b{2}(q) I2111b{3}(q) I2111b{4}(q) I2111b{5}(q)...
    I2111b{6}(q) I2111b{7}(q) I2111b{8}(q) I2111b{9}(q)] = ...
    Bias_Calc(res_step,I2111,I,10);
[I2121b{1}(:,:,q) I2121b{2}(q) I2121b{3}(q) I2121b{4}(q) I2121b{5}(q)...
    I2121b{6}(q) I2121b{7}(q) I2121b{8}(q) I2121b{9}(q)] = ...
    Bias_Calc(res_step,I2121,I,10);
[I2131b{1}(:,:,q) I2131b{2}(q) I2131b{3}(q) I2131b{4}(q) I2131b{5}(q)...
    I2131b{6}(q) I2131b{7}(q) I2131b{8}(q) I2131b{9}(q)] = ...
    Bias_Calc(res_step,I2131,I,10);
[I2211b{1}(:,:,q) I2211b{2}(q) I2211b{3}(q) I2211b{4}(q) I2211b{5}(q)...
    I2211b{6}(q) I2211b{7}(q) I2211b{8}(q) I2211b{9}(q)] = ...
    Bias_Calc(res_step,I2211,I,10);
[I2221b{1}(:,:,q) I2221b{2}(q) I2221b{3}(q) I2221b{4}(q) I2221b{5}(q)...
    I2221b{6}(q) I2221b{7}(q) I2221b{8}(q) I2221b{9}(q)] = ...
    Bias_Calc(res_step,I2221,I,10);
[I2231b{1}(:,:,q) I2231b{2}(q) I2231b{3}(q) I2231b{4}(q) I2231b{5}(q)...
    I2231b{6}(q) I2231b{7}(q) I2231b{8}(q) I2231b{9}(q)] = ...
    Bias_Calc(res_step,I2231,I,10);

[I3111b{1}(:,:,q) I3111b{2}(q) I3111b{3}(q) I3111b{4}(q) I3111b{5}(q)...
    I3111b{6}(q) I3111b{7}(q) I3111b{8}(q) I3111b{9}(q)] = ...
    Bias_Calc(res_step,I3111,I,10);
[I3121b{1}(:,:,q) I3121b{2}(q) I3121b{3}(q) I3121b{4}(q) I3121b{5}(q)...
    I3121b{6}(q) I3121b{7}(q) I3121b{8}(q) I3121b{9}(q)] = ...
    Bias_Calc(res_step,I3121,I,10);
[I3131b{1}(:,:,q) I3131b{2}(q) I3131b{3}(q) I3131b{4}(q) I3131b{5}(q)...
    I3131b{6}(q) I3131b{7}(q) I3131b{8}(q) I3131b{9}(q)] = ...
    Bias_Calc(res_step,I3131,I,10);
[I3211b{1}(:,:,q) I3211b{2}(q) I3211b{3}(q) I3211b{4}(q) I3211b{5}(q)...
    I3211b{6}(q) I3211b{7}(q) I3211b{8}(q) I3211b{9}(q)] = ...
    Bias_Calc(res_step,I3211,I,10);
[I3221b{1}(:,:,q) I3221b{2}(q) I3221b{3}(q) I3221b{4}(q) I3221b{5}(q)...
    I3221b{6}(q) I3221b{7}(q) I3221b{8}(q) I3221b{9}(q)] = ...
    Bias_Calc(res_step,I3221,I,10);
[I3231b{1}(:,:,q) I3231b{2}(q) I3231b{3}(q) I3231b{4}(q) I3231b{5}(q)...
    I3231b{6}(q) I3231b{7}(q) I3231b{8}(q) I3231b{9}(q)] = ...
    Bias_Calc(res_step,I3231,I,10);
[I3311b{1}(:,:,q) I3311b{2}(q) I3311b{3}(q) I3311b{4}(q) I3311b{5}(q)...
    I3311b{6}(q) I3311b{7}(q) I3311b{8}(q) I3311b{9}(q)] = ...
    Bias_Calc(res_step,I3311,I,10);
```

```
[I3321b{1}(:,:,q) I3321b{2}(q) I3321b{3}(q) I3321b{4}(q) I3321b{5}(q)...
    I3321b{6}(q) I3321b{7}(q) I3321b{8}(q) I3321b{9}(q)] = ...
    Bias_Calc(res_step,I3321,I,10);
[I3331b{1}(:,:,q) I3331b{2}(q) I3331b{3}(q) I3331b{4}(q) I3331b{5}(q)...
    I3331b{6}(q) I3331b{7}(q) I3331b{8}(q) I3331b{9}(q)] = ...
    Bias_Calc(res_step,I3331,I,10);

[I1112b{1}(:,:,q) I1112b{2}(q) I1112b{3}(q) I1112b{4}(q) I1112b{5}(q)...
    I1112b{6}(q) I1112b{7}(q) I1112b{8}(q) I1112b{9}(q)] = ...
    Bias_Calc(res_step,I1112,I,10);
[I1122b{1}(:,:,q) I1122b{2}(q) I1122b{3}(q) I1122b{4}(q) I1122b{5}(q)...
    I1122b{6}(q) I1122b{7}(q) I1122b{8}(q) I1122b{9}(q)] = ...
    Bias_Calc(res_step,I1122,I,10);
[I1132b{1}(:,:,q) I1132b{2}(q) I1132b{3}(q) I1132b{4}(q) I1132b{5}(q)...
    I1132b{6}(q) I1132b{7}(q) I1132b{8}(q) I1132b{9}(q)] = ...
    Bias_Calc(res_step,I1132,I,10);
[I1212b{1}(:,:,q) I1212b{2}(q) I1212b{3}(q) I1212b{4}(q) I1212b{5}(q)...
    I1212b{6}(q) I1212b{7}(q) I1212b{8}(q) I1212b{9}(q)] = ...
    Bias_Calc(res_step,I1212,I,10);
[I1222b{1}(:,:,q) I1222b{2}(q) I1222b{3}(q) I1222b{4}(q) I1222b{5}(q)...
    I1222b{6}(q) I1222b{7}(q) I1222b{8}(q) I1222b{9}(q)] = ...
    Bias_Calc(res_step,I1222,I,10);
[I1232b{1}(:,:,q) I1232b{2}(q) I1232b{3}(q) I1232b{4}(q) I1232b{5}(q)...
    I1232b{6}(q) I1232b{7}(q) I1232b{8}(q) I1232b{9}(q)] = ...
    Bias_Calc(res_step,I1232,I,10);
[I1312b{1}(:,:,q) I1312b{2}(q) I1312b{3}(q) I1312b{4}(q) I1312b{5}(q)...
    I1312b{6}(q) I1312b{7}(q) I1312b{8}(q) I1312b{9}(q)] = ...
    Bias_Calc(res_step,I1312,I,10);
[I1322b{1}(:,:,q) I1322b{2}(q) I1322b{3}(q) I1322b{4}(q) I1322b{5}(q)...
    I1322b{6}(q) I1322b{7}(q) I1322b{8}(q) I1322b{9}(q)] = ...
    Bias_Calc(res_step,I1322,I,10);
[I1332b{1}(:,:,q) I1332b{2}(q) I1332b{3}(q) I1332b{4}(q) I1332b{5}(q)...
    I1332b{6}(q) I1332b{7}(q) I1332b{8}(q) I1332b{9}(q)] = ...
    Bias_Calc(res_step,I1332,I,10);
[I1412b{1}(:,:,q) I1412b{2}(q) I1412b{3}(q) I1412b{4}(q) I1412b{5}(q)...
    I1412b{6}(q) I1412b{7}(q) I1412b{8}(q) I1412b{9}(q)] = ...
    Bias_Calc(res_step,I1412,I,10);
[I1422b{1}(:,:,q) I1422b{2}(q) I1422b{3}(q) I1422b{4}(q) I1422b{5}(q)...
    I1422b{6}(q) I1422b{7}(q) I1422b{8}(q) I1422b{9}(q)] = ...
    Bias_Calc(res_step,I1422,I,10);
[I1432b{1}(:,:,q) I1432b{2}(q) I1432b{3}(q) I1432b{4}(q) I1432b{5}(q)...
    I1432b{6}(q) I1432b{7}(q) I1432b{8}(q) I1432b{9}(q)] = ...
    Bias_Calc(res_step,I1432,I,10);

[I4111b{1}(:,:,q) I4111b{2}(q) I4111b{3}(q) I4111b{4}(q) I4111b{5}(q)...
    I4111b{6}(q) I4111b{7}(q) I4111b{8}(q) I4111b{9}(q)] = ...
    Bias_Calc(res_step,I4111,I,10);
[I4121b{1}(:,:,q) I4121b{2}(q) I4121b{3}(q) I4121b{4}(q) I4121b{5}(q)...
    I4121b{6}(q) I4121b{7}(q) I4121b{8}(q) I4121b{9}(q)] = ...
    Bias_Calc(res_step,I4121,I,10);
[I4131b{1}(:,:,q) I4131b{2}(q) I4131b{3}(q) I4131b{4}(q) I4131b{5}(q)...
    I4131b{6}(q) I4131b{7}(q) I4131b{8}(q) I4131b{9}(q)] = ...
    Bias_Calc(res_step,I4131,I,10);
[I4211b{1}(:,:,q) I4211b{2}(q) I4211b{3}(q) I4211b{4}(q) I4211b{5}(q)...
    I4211b{6}(q) I4211b{7}(q) I4211b{8}(q) I4211b{9}(q)] = ...
    Bias_Calc(res_step,I4211,I,10);
```

```
[I4221b{1}(:,:,q) I4221b{2}(q) I4221b{3}(q) I4221b{4}(q) I4221b{5}(q)...
    I4221b{6}(q) I4221b{7}(q) I4221b{8}(q) I4221b{9}(q)] = ...
    Bias_Calc(res_step,I4221,I,10);
[I4231b{1}(:,:,q) I4231b{2}(q) I4231b{3}(q) I4231b{4}(q) I4231b{5}(q)...
    I4231b{6}(q) I4231b{7}(q) I4231b{8}(q) I4231b{9}(q)] = ...
    Bias_Calc(res_step,I4231,I,10);

[Pi12b{1}(:,:,q) Pi12b{2}(q) Pi12b{3}(q) Pi12b{4}(q) Pi12b{5}(q)...
    Pi12b{6}(q) Pi12b{7}(q) Pi12b{8}(q) Pi12b{9}(q)] = ...
    Bias_Calc(res_step,Pi12,p0,20);
[Pi13b{1}(:,:,q) Pi13b{2}(q) Pi13b{3}(q) Pi13b{4}(q) Pi13b{5}(q)...
    Pi13b{6}(q) Pi13b{7}(q) Pi13b{8}(q) Pi13b{9}(q)] = ...
    Bias_Calc(res_step,Pi13,p0,20);
[Pi14b{1}(:,:,q) Pi14b{2}(q) Pi14b{3}(q) Pi14b{4}(q) Pi14b{5}(q)...
    Pi14b{6}(q) Pi14b{7}(q) Pi14b{8}(q) Pi14b{9}(q)] = ...
    Bias_Calc(res_step,Pi14,p0,20);
[P12b{1}(:,:,q) P12b{2}(q) P12b{3}(q) P12b{4}(q) P12b{5}(q)...
    P12b{6}(q) P12b{7}(q) P12b{8}(q) P12b{9}(q)] = ...
    Bias_Calc(res_step,P12,p0,20);
[P13b{1}(:,:,q) P13b{2}(q) P13b{3}(q) P13b{4}(q) P13b{5}(q)...
    P13b{6}(q) P13b{7}(q) P13b{8}(q) P13b{9}(q)] = ...
    Bias_Calc(res_step,P13,p0,20);
[P14b{1}(:,:,q) P14b{2}(q) P14b{3}(q) P14b{4}(q) P14b{5}(q)...
    P14b{6}(q) P14b{7}(q) P14b{8}(q) P14b{9}(q)] = ...
    Bias_Calc(res_step,P14,p0,20);

[Pi21b{1}(:,:,q) Pi21b{2}(q) Pi21b{3}(q) Pi21b{4}(q) Pi21b{5}(q)...
    Pi21b{6}(q) Pi21b{7}(q) Pi21b{8}(q) Pi21b{9}(q)] = ...
    Bias_Calc(res_step,Pi21,p0,20);
[Pi22b{1}(:,:,q) Pi22b{2}(q) Pi22b{3}(q) Pi22b{4}(q) Pi22b{5}(q)...
    Pi22b{6}(q) Pi22b{7}(q) Pi22b{8}(q) Pi22b{9}(q)] = ...
    Bias_Calc(res_step,Pi22,p0,20);
[P21b{1}(:,:,q) P21b{2}(q) P21b{3}(q) P21b{4}(q) P21b{5}(q)...
    P21b{6}(q) P21b{7}(q) P21b{8}(q) P21b{9}(q)] = ...
    Bias_Calc(res_step,P21,p0,20);
[P22b{1}(:,:,q) P22b{2}(q) P22b{3}(q) P22b{4}(q) P22b{5}(q)...
    P22b{6}(q) P22b{7}(q) P22b{8}(q) P22b{9}(q)] = ...
    Bias_Calc(res_step,P22,p0,20);

[Pi32b{1}(:,:,q) Pi32b{2}(q) Pi32b{3}(q) Pi32b{4}(q) Pi32b{5}(q)...
    Pi32b{6}(q) Pi32b{7}(q) Pi32b{8}(q) Pi32b{9}(q)] = ...
    Bias_Calc(res_step,Pi32,p0,20);
[Pi33b{1}(:,:,q) Pi33b{2}(q) Pi33b{3}(q) Pi33b{4}(q) Pi33b{5}(q)...
    Pi33b{6}(q) Pi33b{7}(q) Pi33b{8}(q) Pi33b{9}(q)] = ...
    Bias_Calc(res_step,Pi33,p0,20);
[P32b{1}(:,:,q) P32b{2}(q) P32b{3}(q) P32b{4}(q) P32b{5}(q)...
    P32b{6}(q) P32b{7}(q) P32b{8}(q) P32b{9}(q)] = ...
    Bias_Calc(res_step,P32,p0,20);
[P33b{1}(:,:,q) P33b{2}(q) P33b{3}(q) P33b{4}(q) P33b{5}(q)...
    P33b{6}(q) P33b{7}(q) P33b{8}(q) P33b{9}(q)] = ...
    Bias_Calc(res_step,P33,p0,20);

[Pi41b{1}(:,:,q) Pi41b{2}(q) Pi41b{3}(q) Pi41b{4}(q) Pi41b{5}(q)...
    Pi41b{6}(q) Pi41b{7}(q) Pi41b{8}(q) Pi41b{9}(q)] = ...
    Bias_Calc(res_step,Pi41,p0,20);
```

```
[Pi42b{1}(:,:,q) Pi42b{2}(q) Pi42b{3}(q) Pi42b{4}(q) Pi42b{5}(q)...
    Pi42b{6}(q) Pi42b{7}(q) Pi42b{8}(q) Pi42b{9}(q)] = ...
    Bias_Calc(res_step,Pi42,p0,20);
[P41b{1}(:,:,q) P41b{2}(q) P41b{3}(q) P41b{4}(q) P41b{5}(q)...
    P41b{6}(q) P41b{7}(q) P41b{8}(q) P41b{9}(q)] = ...
    Bias_Calc(res_step,P41,p0,20);
[P42b{1}(:,:,q) P42b{2}(q) P42b{3}(q) P42b{4}(q) P42b{5}(q)...
    P42b{6}(q) P42b{7}(q) P42b{8}(q) P42b{9}(q)] = ...
    Bias_Calc(res_step,P42,p0,20);

[v111b{1}(:,:,q) v111b{2}(q) v111b{3}(q) v111b{4}(q) v111b{5}(q)...
    v111b{6}(q) v111b{7}(q) v111b{8}(q) v111b{9}(q)] = ...
    Bias_Calc(res_step,v111,v,20);
[v121b{1}(:,:,q) v121b{2}(q) v121b{3}(q) v121b{4}(q) v121b{5}(q)...
    v121b{6}(q) v121b{7}(q) v121b{8}(q) v121b{9}(q)] = ...
    Bias_Calc(res_step,v121,v,20);
[v131b{1}(:,:,q) v131b{2}(q) v131b{3}(q) v131b{4}(q) v131b{5}(q)...
    v131b{6}(q) v131b{7}(q) v131b{8}(q) v131b{9}(q)] = ...
    Bias_Calc(res_step,v131,v,20);

[v211b{1}(:,:,q) v211b{2}(q) v211b{3}(q) v211b{4}(q) v211b{5}(q)...
    v211b{6}(q) v211b{7}(q) v211b{8}(q) v211b{9}(q)] = ...
    Bias_Calc(res_step,v211,v,20);
[v221b{1}(:,:,q) v221b{2}(q) v221b{3}(q) v221b{4}(q) v221b{5}(q)...
    v221b{6}(q) v221b{7}(q) v221b{8}(q) v221b{9}(q)] = ...
    Bias_Calc(res_step,v221,v,20);
[v231b{1}(:,:,q) v231b{2}(q) v231b{3}(q) v231b{4}(q) v231b{5}(q)...
    v231b{6}(q) v231b{7}(q) v231b{8}(q) v231b{9}(q)] = ...
    Bias_Calc(res_step,v231,v,20);

[v311b{1}(:,:,q) v311b{2}(q) v311b{3}(q) v311b{4}(q) v311b{5}(q)...
    v311b{6}(q) v311b{7}(q) v311b{8}(q) v311b{9}(q)] = ...
    Bias_Calc(res_step,v311,v,20);
[v321b{1}(:,:,q) v321b{2}(q) v321b{3}(q) v321b{4}(q) v321b{5}(q)...
    v321b{6}(q) v321b{7}(q) v321b{8}(q) v321b{9}(q)] = ...
    Bias_Calc(res_step,v321,v,20);
[v331b{1}(:,:,q) v331b{2}(q) v331b{3}(q) v331b{4}(q) v331b{5}(q)...
    v331b{6}(q) v331b{7}(q) v331b{8}(q) v331b{9}(q)] = ...
    Bias_Calc(res_step,v331,v,20);

[v112b{1}(:,:,q) v112b{2}(q) v112b{3}(q) v112b{4}(q) v112b{5}(q)...
    v112b{6}(q) v112b{7}(q) v112b{8}(q) v112b{9}(q)] = ...
    Bias_Calc(res_step,v112,v,20);
[v122b{1}(:,:,q) v122b{2}(q) v122b{3}(q) v122b{4}(q) v122b{5}(q)...
    v122b{6}(q) v122b{7}(q) v122b{8}(q) v122b{9}(q)] = ...
    Bias_Calc(res_step,v122,v,20);
[v132b{1}(:,:,q) v132b{2}(q) v132b{3}(q) v132b{4}(q) v132b{5}(q)...
    v132b{6}(q) v132b{7}(q) v132b{8}(q) v132b{9}(q)] = ...
    Bias_Calc(res_step,v132,v,20);

[v411b{1}(:,:,q) v411b{2}(q) v411b{3}(q) v411b{4}(q) v411b{5}(q)...
    v411b{6}(q) v411b{7}(q) v411b{8}(q) v411b{9}(q)] = ...
    Bias_Calc(res_step,v411,v,20);
[v421b{1}(:,:,q) v421b{2}(q) v421b{3}(q) v421b{4}(q) v421b{5}(q)...
    v421b{6}(q) v421b{7}(q) v421b{8}(q) v421b{9}(q)] = ...
```

```
    Bias_Calc(res_step,v421,v,20);
[v431b{1}(:,:,q) v431b{2}(q) v431b{3}(q) v431b{4}(q) v431b{5}(q)...
    v431b{6}(q) v431b{7}(q) v431b{8}(q) v431b{9}(q)] = ...
    Bias_Calc(res_step,v431,v,20);

[W1211b{1}(:,:,q) W1211b{2}(q) W1211b{3}(q) W1211b{4}(q) W1211b{5}(q)...
    W1211b{6}(q) W1211b{7}(q) W1211b{8}(q) W1211b{9}(q)] = ...
    Bias_Calc(res_step,W1211,W,10);
[W1221b{1}(:,:,q) W1221b{2}(q) W1221b{3}(q) W1221b{4}(q) W1221b{5}(q)...
    W1221b{6}(q) W1221b{7}(q) W1221b{8}(q) W1221b{9}(q)] = ...
    Bias_Calc(res_step,W1221,W,10);
[W1231b{1}(:,:,q) W1231b{2}(q) W1231b{3}(q) W1231b{4}(q) W1231b{5}(q)...
    W1231b{6}(q) W1231b{7}(q) W1231b{8}(q) W1231b{9}(q)] = ...
    Bias_Calc(res_step,W1231,W,10);
[W1311b{1}(:,:,q) W1311b{2}(q) W1311b{3}(q) W1311b{4}(q) W1311b{5}(q)...
    W1311b{6}(q) W1311b{7}(q) W1311b{8}(q) W1311b{9}(q)] = ...
    Bias_Calc(res_step,W1311,W,10);
[W1321b{1}(:,:,q) W1321b{2}(q) W1321b{3}(q) W1321b{4}(q) W1321b{5}(q)...
    W1321b{6}(q) W1321b{7}(q) W1321b{8}(q) W1321b{9}(q)] = ...
    Bias_Calc(res_step,W1321,W,10);
[W1331b{1}(:,:,q) W1331b{2}(q) W1331b{3}(q) W1331b{4}(q) W1331b{5}(q)...
    W1331b{6}(q) W1331b{7}(q) W1331b{8}(q) W1331b{9}(q)] = ...
    Bias_Calc(res_step,W1331,W,10);
[W1411b{1}(:,:,q) W1411b{2}(q) W1411b{3}(q) W1411b{4}(q) W1411b{5}(q)...
    W1411b{6}(q) W1411b{7}(q) W1411b{8}(q) W1411b{9}(q)] = ...
    Bias_Calc(res_step,W1411,W,10);
[W1421b{1}(:,:,q) W1421b{2}(q) W1421b{3}(q) W1421b{4}(q) W1421b{5}(q)...
    W1421b{6}(q) W1421b{7}(q) W1421b{8}(q) W1421b{9}(q)] = ...
    Bias_Calc(res_step,W1421,W,10);
[W1431b{1}(:,:,q) W1431b{2}(q) W1431b{3}(q) W1431b{4}(q) W1431b{5}(q)...
    W1431b{6}(q) W1431b{7}(q) W1431b{8}(q) W1431b{9}(q)] = ...
    Bias_Calc(res_step,W1431,W,10);

[W2111b{1}(:,:,q) W2111b{2}(q) W2111b{3}(q) W2111b{4}(q) W2111b{5}(q)...
    W2111b{6}(q) W2111b{7}(q) W2111b{8}(q) W2111b{9}(q)] = ...
    Bias_Calc(res_step,W2111,W,10);
[W2121b{1}(:,:,q) W2121b{2}(q) W2121b{3}(q) W2121b{4}(q) W2121b{5}(q)...
    W2121b{6}(q) W2121b{7}(q) W2121b{8}(q) W2121b{9}(q)] = ...
    Bias_Calc(res_step,W2121,W,10);
[W2131b{1}(:,:,q) W2131b{2}(q) W2131b{3}(q) W2131b{4}(q) W2131b{5}(q)...
    W2131b{6}(q) W2131b{7}(q) W2131b{8}(q) W2131b{9}(q)] = ...
    Bias_Calc(res_step,W2131,W,10);
[W2211b{1}(:,:,q) W2211b{2}(q) W2211b{3}(q) W2211b{4}(q) W2211b{5}(q)...
    W2211b{6}(q) W2211b{7}(q) W2211b{8}(q) W2211b{9}(q)] = ...
    Bias_Calc(res_step,W2211,W,10);
[W2221b{1}(:,:,q) W2221b{2}(q) W2221b{3}(q) W2221b{4}(q) W2221b{5}(q)...
    W2221b{6}(q) W2221b{7}(q) W2221b{8}(q) W2221b{9}(q)] = ...
    Bias_Calc(res_step,W2221,W,10);
[W2231b{1}(:,:,q) W2231b{2}(q) W2231b{3}(q) W2231b{4}(q) W2231b{5}(q)...
    W2231b{6}(q) W2231b{7}(q) W2231b{8}(q) W2231b{9}(q)] = ...
    Bias_Calc(res_step,W2231,W,10);

[W3211b{1}(:,:,q) W3211b{2}(q) W3211b{3}(q) W3211b{4}(q) W3211b{5}(q)...
    W3211b{6}(q) W3211b{7}(q) W3211b{8}(q) W3211b{9}(q)] = ...
    Bias_Calc(res_step,W3211,W,10);
[W3221b{1}(:,:,q) W3221b{2}(q) W3221b{3}(q) W3221b{4}(q) W3221b{5}(q)...
```

```
    W3221b{6}(q) W3221b{7}(q) W3221b{8}(q) W3221b{9}(q)] = ...
    Bias_Calc(res_step,W3221,W,10);
[W3231b{1}(:,:,q) W3231b{2}(q) W3231b{3}(q) W3231b{4}(q) W3231b{5}(q)...
    W3231b{6}(q) W3231b{7}(q) W3231b{8}(q) W3231b{9}(q)] = ...
    Bias_Calc(res_step,W3231,W,10);
[W3311b{1}(:,:,q) W3311b{2}(q) W3311b{3}(q) W3311b{4}(q) W3311b{5}(q)...
    W3311b{6}(q) W3311b{7}(q) W3311b{8}(q) W3311b{9}(q)] = ...
    Bias_Calc(res_step,W3311,W,10);
[W3321b{1}(:,:,q) W3321b{2}(q) W3321b{3}(q) W3321b{4}(q) W3321b{5}(q)...
    W3321b{6}(q) W3321b{7}(q) W3321b{8}(q) W3321b{9}(q)] = ...
    Bias_Calc(res_step,W3321,W,10);
[W3331b{1}(:,:,q) W3331b{2}(q) W3331b{3}(q) W3331b{4}(q) W3331b{5}(q)...
    W3331b{6}(q) W3331b{7}(q) W3331b{8}(q) W3331b{9}(q)] = ...
    Bias_Calc(res_step,W3331,W,10);

[W1212b{1}(:,:,q) W1212b{2}(q) W1212b{3}(q) W1212b{4}(q) W1212b{5}(q)...
    W1212b{6}(q) W1212b{7}(q) W1212b{8}(q) W1212b{9}(q)] = ...
    Bias_Calc(res_step,W1212,W,10);
[W1222b{1}(:,:,q) W1222b{2}(q) W1222b{3}(q) W1222b{4}(q) W1222b{5}(q)...
    W1222b{6}(q) W1222b{7}(q) W1222b{8}(q) W1222b{9}(q)] = ...
    Bias_Calc(res_step,W1222,W,10);
[W1232b{1}(:,:,q) W1232b{2}(q) W1232b{3}(q) W1232b{4}(q) W1232b{5}(q)...
    W1232b{6}(q) W1232b{7}(q) W1232b{8}(q) W1232b{9}(q)] = ...
    Bias_Calc(res_step,W1232,W,10);
[W1312b{1}(:,:,q) W1312b{2}(q) W1312b{3}(q) W1312b{4}(q) W1312b{5}(q)...
    W1312b{6}(q) W1312b{7}(q) W1312b{8}(q) W1312b{9}(q)] = ...
    Bias_Calc(res_step,W1312,W,10);
[W1322b{1}(:,:,q) W1322b{2}(q) W1322b{3}(q) W1322b{4}(q) W1322b{5}(q)...
    W1322b{6}(q) W1322b{7}(q) W1322b{8}(q) W1322b{9}(q)] = ...
    Bias_Calc(res_step,W1322,W,10);
[W1332b{1}(:,:,q) W1332b{2}(q) W1332b{3}(q) W1332b{4}(q) W1332b{5}(q)...
    W1332b{6}(q) W1332b{7}(q) W1332b{8}(q) W1332b{9}(q)] = ...
    Bias_Calc(res_step,W1332,W,10);
[W1412b{1}(:,:,q) W1412b{2}(q) W1412b{3}(q) W1412b{4}(q) W1412b{5}(q)...
    W1412b{6}(q) W1412b{7}(q) W1412b{8}(q) W1412b{9}(q)] = ...
    Bias_Calc(res_step,W1412,W,10);
[W1422b{1}(:,:,q) W1422b{2}(q) W1422b{3}(q) W1422b{4}(q) W1422b{5}(q)...
    W1422b{6}(q) W1422b{7}(q) W1422b{8}(q) W1422b{9}(q)] = ...
    Bias_Calc(res_step,W1422,W,10);
[W1432b{1}(:,:,q) W1432b{2}(q) W1432b{3}(q) W1432b{4}(q) W1432b{5}(q)...
    W1432b{6}(q) W1432b{7}(q) W1432b{8}(q) W1432b{9}(q)] = ...
    Bias_Calc(res_step,W1432,W,10);

[W4111b{1}(:,:,q) W4111b{2}(q) W4111b{3}(q) W4111b{4}(q) W4111b{5}(q)...
    W4111b{6}(q) W4111b{7}(q) W4111b{8}(q) W4111b{9}(q)] = ...
    Bias_Calc(res_step,W4111,W,10);
[W4121b{1}(:,:,q) W4121b{2}(q) W4121b{3}(q) W4121b{4}(q) W4121b{5}(q)...
    W4121b{6}(q) W4121b{7}(q) W4121b{8}(q) W4121b{9}(q)] = ...
    Bias_Calc(res_step,W4121,W,10);
[W4131b{1}(:,:,q) W4131b{2}(q) W4131b{3}(q) W4131b{4}(q) W4131b{5}(q)...
    W4131b{6}(q) W4131b{7}(q) W4131b{8}(q) W4131b{9}(q)] = ...
    Bias_Calc(res_step,W4131,W,10);
[W4211b{1}(:,:,q) W4211b{2}(q) W4211b{3}(q) W4211b{4}(q) W4211b{5}(q)...
    W4211b{6}(q) W4211b{7}(q) W4211b{8}(q) W4211b{9}(q)] = ...
    Bias_Calc(res_step,W4211,W,10);
[W4221b{1}(:,:,q) W4221b{2}(q) W4221b{3}(q) W4221b{4}(q) W4221b{5}(q)...
```

```
    W4221b{6}(q) W4221b{7}(q) W4221b{8}(q) W4221b{9}(q)] = ...
    Bias_Calc(res_step,W4221,W,10);
[W4231b{1}(:,:,q) W4231b{2}(q) W4231b{3}(q) W4231b{4}(q) W4231b{5}(q)...
    W4231b{6}(q) W4231b{7}(q) W4231b{8}(q) W4231b{9}(q)] = ...
    Bias_Calc(res_step,W4231,W,10);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  Direction Bias Calculations   %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


[I1111angb{1}(:,:,q) I1111angb{2}(q) I1111angb{3}(q) I1111angb{4}(q)...
    I1111angb{5}(q) I1111angb{6}(q) I1111angb{7}(q) I1111angb{8}(q)...
    I1111angb{9}(q)] = Bias_Calc(res_step,I1111ang(:,:,q));
[I1121angb{1}(:,:,q) I1121angb{2}(q) I1121angb{3}(q) I1121angb{4}(q)...
    I1121angb{5}(q) I1121angb{6}(q) I1121angb{7}(q) I1121angb{8}(q)...
    I1121angb{9}(q)] = Bias_Calc(res_step,I1121ang(:,:,q));
[I1131angb{1}(:,:,q) I1131angb{2}(q) I1131angb{3}(q) I1131angb{4}(q)...
    I1131angb{5}(q) I1131angb{6}(q) I1131angb{7}(q) I1131angb{8}(q)...
    I1131angb{9}(q)] = Bias_Calc(res_step,I1131ang(:,:,q));
[I1211angb{1}(:,:,q) I1211angb{2}(q) I1211angb{3}(q) I1211angb{4}(q)...
    I1211angb{5}(q) I1211angb{6}(q) I1211angb{7}(q) I1211angb{8}(q)...
    I1211angb{9}(q)] = Bias_Calc(res_step,I1211ang(:,:,q));
[I1221angb{1}(:,:,q) I1221angb{2}(q) I1221angb{3}(q) I1221angb{4}(q)...
    I1221angb{5}(q) I1221angb{6}(q) I1221angb{7}(q) I1221angb{8}(q)...
    I1221angb{9}(q)] = Bias_Calc(res_step,I1221ang(:,:,q));
[I1231angb{1}(:,:,q) I1231angb{2}(q) I1231angb{3}(q) I1231angb{4}(q)...
    I1231angb{5}(q) I1231angb{6}(q) I1231angb{7}(q) I1231angb{8}(q)...
    I1231angb{9}(q)] = Bias_Calc(res_step,I1231ang(:,:,q));
[I1311angb{1}(:,:,q) I1311angb{2}(q) I1311angb{3}(q) I1311angb{4}(q)...
    I1311angb{5}(q) I1311angb{6}(q) I1311angb{7}(q) I1311angb{8}(q)...
    I1311angb{9}(q)] = Bias_Calc(res_step,I1311ang(:,:,q));
[I1321angb{1}(:,:,q) I1321angb{2}(q) I1321angb{3}(q) I1321angb{4}(q)...
    I1321angb{5}(q) I1321angb{6}(q) I1321angb{7}(q) I1321angb{8}(q)...
    I1321angb{9}(q)] = Bias_Calc(res_step,I1321ang(:,:,q));
[I1331angb{1}(:,:,q) I1331angb{2}(q) I1331angb{3}(q) I1331angb{4}(q)...
    I1331angb{5}(q) I1331angb{6}(q) I1331angb{7}(q) I1331angb{8}(q)...
    I1331angb{9}(q)] = Bias_Calc(res_step,I1331ang(:,:,q));
[I1411angb{1}(:,:,q) I1411angb{2}(q) I1411angb{3}(q) I1411angb{4}(q)...
    I1411angb{5}(q) I1411angb{6}(q) I1411angb{7}(q) I1411angb{8}(q)...
    I1411angb{9}(q)] = Bias_Calc(res_step,I1411ang(:,:,q));
[I1421angb{1}(:,:,q) I1421angb{2}(q) I1421angb{3}(q) I1421angb{4}(q)...
    I1421angb{5}(q) I1421angb{6}(q) I1421angb{7}(q) I1421angb{8}(q)...
    I1421angb{9}(q)] = Bias_Calc(res_step,I1421ang(:,:,q));
[I1431angb{1}(:,:,q) I1431angb{2}(q) I1431angb{3}(q) I1431angb{4}(q)...
    I1431angb{5}(q) I1431angb{6}(q) I1431angb{7}(q) I1431angb{8}(q)...
    I1431angb{9}(q)] = Bias_Calc(res_step,I1431ang(:,:,q));


[I2111angb{1}(:,:,q) I2111angb{2}(q) I2111angb{3}(q) I2111angb{4}(q)...
    I2111angb{5}(q) I2111angb{6}(q) I2111angb{7}(q) I2111angb{8}(q)...
    I2111angb{9}(q)] = Bias_Calc(res_step,I2111ang(:,:,q));
[I2121angb{1}(:,:,q) I2121angb{2}(q) I2121angb{3}(q) I2121angb{4}(q)...
    I2121angb{5}(q) I2121angb{6}(q) I2121angb{7}(q) I2121angb{8}(q)...
```

```
    I2121angb{9}(q)] = Bias_Calc(res_step,I2121ang(:,:,q));
[I2131angb{1}(:,:,q) I2131angb{2}(q) I2131angb{3}(q) I2131angb{4}(q)...
    I2131angb{5}(q) I2131angb{6}(q) I2131angb{7}(q) I2131angb{8}(q)...
    I2131angb{9}(q)] = Bias_Calc(res_step,I2131ang(:,:,q));
[I2211angb{1}(:,:,q) I2211angb{2}(q) I2211angb{3}(q) I2211angb{4}(q)...
    I2211angb{5}(q) I2211angb{6}(q) I2211angb{7}(q) I2211angb{8}(q)...
    I2211angb{9}(q)] = Bias_Calc(res_step,I2211ang(:,:,q));
[I2221angb{1}(:,:,q) I2221angb{2}(q) I2221angb{3}(q) I2221angb{4}(q)...
    I2221angb{5}(q) I2221angb{6}(q) I2221angb{7}(q) I2221angb{8}(q)...
    I2221angb{9}(q)] = Bias_Calc(res_step,I2221ang(:,:,q));
[I2231angb{1}(:,:,q) I2231angb{2}(q) I2231angb{3}(q) I2231angb{4}(q)...
    I2231angb{5}(q) I2231angb{6}(q) I2231angb{7}(q) I2231angb{8}(q)...
    I2231angb{9}(q)] = Bias_Calc(res_step,I2231ang(:,:,q));

[I3111angb{1}(:,:,q) I3111angb{2}(q) I3111angb{3}(q) I3111angb{4}(q)...
    I3111angb{5}(q) I3111angb{6}(q) I3111angb{7}(q) I3111angb{8}(q)...
    I3111angb{9}(q)] = Bias_Calc(res_step,I3111ang(:,:,q));
[I3121angb{1}(:,:,q) I3121angb{2}(q) I3121angb{3}(q) I3121angb{4}(q)...
    I3121angb{5}(q) I3121angb{6}(q) I3121angb{7}(q) I3121angb{8}(q)...
    I3121angb{9}(q)] = Bias_Calc(res_step,I3121ang(:,:,q));
[I3131angb{1}(:,:,q) I3131angb{2}(q) I3131angb{3}(q) I3131angb{4}(q)...
    I3131angb{5}(q) I3131angb{6}(q) I3131angb{7}(q) I3131angb{8}(q)...
    I3131angb{9}(q)] = Bias_Calc(res_step,I3131ang(:,:,q));
[I3211angb{1}(:,:,q) I3211angb{2}(q) I3211angb{3}(q) I3211angb{4}(q)...
    I3211angb{5}(q) I3211angb{6}(q) I3211angb{7}(q) I3211angb{8}(q)...
    I3211angb{9}(q)] = Bias_Calc(res_step,I3211ang(:,:,q));
[I3221angb{1}(:,:,q) I3221angb{2}(q) I3221angb{3}(q) I3221angb{4}(q)...
    I3221angb{5}(q) I3221angb{6}(q) I3221angb{7}(q) I3221angb{8}(q)...
    I3221angb{9}(q)] = Bias_Calc(res_step,I3221ang(:,:,q));
[I3231angb{1}(:,:,q) I3231angb{2}(q) I3231angb{3}(q) I3231angb{4}(q)...
    I3231angb{5}(q) I3231angb{6}(q) I3231angb{7}(q) I3231angb{8}(q)...
    I3231angb{9}(q)] = Bias_Calc(res_step,I3231ang(:,:,q));
[I3311angb{1}(:,:,q) I3311angb{2}(q) I3311angb{3}(q) I3311angb{4}(q)...
    I3311angb{5}(q) I3311angb{6}(q) I3311angb{7}(q) I3311angb{8}(q)...
    I3311angb{9}(q)] = Bias_Calc(res_step,I3311ang(:,:,q));
[I3321angb{1}(:,:,q) I3321angb{2}(q) I3321angb{3}(q) I3321angb{4}(q)...
    I3321angb{5}(q) I3321angb{6}(q) I3321angb{7}(q) I3321angb{8}(q)...
    I3321angb{9}(q)] = Bias_Calc(res_step,I3321ang(:,:,q));
[I3331angb{1}(:,:,q) I3331angb{2}(q) I3331angb{3}(q) I3331angb{4}(q)...
    I3331angb{5}(q) I3331angb{6}(q) I3331angb{7}(q) I3331angb{8}(q)...
    I3331angb{9}(q)] = Bias_Calc(res_step,I3331ang(:,:,q));

[I1112angb{1}(:,:,q) I1112angb{2}(q) I1112angb{3}(q) I1112angb{4}(q)...
    I1112angb{5}(q) I1112angb{6}(q) I1112angb{7}(q) I1112angb{8}(q)...
    I1112angb{9}(q)] = Bias_Calc(res_step,I1112ang(:,:,q));
[I1122angb{1}(:,:,q) I1122angb{2}(q) I1122angb{3}(q) I1122angb{4}(q)...
    I1122angb{5}(q) I1122angb{6}(q) I1122angb{7}(q) I1122angb{8}(q)...
    I1122angb{9}(q)] = Bias_Calc(res_step,I1122ang(:,:,q));
[I1132angb{1}(:,:,q) I1132angb{2}(q) I1132angb{3}(q) I1132angb{4}(q)...
    I1132angb{5}(q) I1132angb{6}(q) I1132angb{7}(q) I1132angb{8}(q)...
    I1132angb{9}(q)] = Bias_Calc(res_step,I1132ang(:,:,q));
[I1212angb{1}(:,:,q) I1212angb{2}(q) I1212angb{3}(q) I1212angb{4}(q)...
    I1212angb{5}(q) I1212angb{6}(q) I1212angb{7}(q) I1212angb{8}(q)...
    I1212angb{9}(q)] = Bias_Calc(res_step,I1212ang(:,:,q));
[I1222angb{1}(:,:,q) I1222angb{2}(q) I1222angb{3}(q) I1222angb{4}(q)...
    I1222angb{5}(q) I1222angb{6}(q) I1222angb{7}(q) I1222angb{8}(q)...
```

```
    I1222angb{9}(q)] = Bias_Calc(res_step,I1222ang(:,:,q));
[I1232angb{1}(:,:,q) I1232angb{2}(q) I1232angb{3}(q) I1232angb{4}(q)...
    I1232angb{5}(q) I1232angb{6}(q) I1232angb{7}(q) I1232angb{8}(q)...
    I1232angb{9}(q)] = Bias_Calc(res_step,I1232ang(:,:,q));
[I1312angb{1}(:,:,q) I1312angb{2}(q) I1312angb{3}(q) I1312angb{4}(q)...
    I1312angb{5}(q) I1312angb{6}(q) I1312angb{7}(q) I1312angb{8}(q)...
    I1312angb{9}(q)] = Bias_Calc(res_step,I1312ang(:,:,q));
[I1322angb{1}(:,:,q) I1322angb{2}(q) I1322angb{3}(q) I1322angb{4}(q)...
    I1322angb{5}(q) I1322angb{6}(q) I1322angb{7}(q) I1322angb{8}(q)...
    I1322angb{9}(q)] = Bias_Calc(res_step,I1322ang(:,:,q));
[I1332angb{1}(:,:,q) I1332angb{2}(q) I1332angb{3}(q) I1332angb{4}(q)...
    I1332angb{5}(q) I1332angb{6}(q) I1332angb{7}(q) I1332angb{8}(q)...
    I1332angb{9}(q)] = Bias_Calc(res_step,I1332ang(:,:,q));
[I1412angb{1}(:,:,q) I1412angb{2}(q) I1412angb{3}(q) I1412angb{4}(q)...
    I1412angb{5}(q) I1412angb{6}(q) I1412angb{7}(q) I1412angb{8}(q)...
    I1412angb{9}(q)] = Bias_Calc(res_step,I1412ang(:,:,q));
[I1422angb{1}(:,:,q) I1422angb{2}(q) I1422angb{3}(q) I1422angb{4}(q)...
    I1422angb{5}(q) I1422angb{6}(q) I1422angb{7}(q) I1422angb{8}(q)...
    I1422angb{9}(q)] = Bias_Calc(res_step,I1422ang(:,:,q));
[I1432angb{1}(:,:,q) I1432angb{2}(q) I1432angb{3}(q) I1432angb{4}(q)...
    I1432angb{5}(q) I1432angb{6}(q) I1432angb{7}(q) I1432angb{8}(q)...
    I1432angb{9}(q)] = Bias_Calc(res_step,I1432ang(:,:,q));

[I4111angb{1}(:,:,q) I4111angb{2}(q) I4111angb{3}(q) I4111angb{4}(q)...
    I4111angb{5}(q) I4111angb{6}(q) I4111angb{7}(q) I4111angb{8}(q)...
    I4111angb{9}(q)] = Bias_Calc(res_step,I4111ang(:,:,q));
[I4121angb{1}(:,:,q) I4121angb{2}(q) I4121angb{3}(q) I4121angb{4}(q)...
    I4121angb{5}(q) I4121angb{6}(q) I4121angb{7}(q) I4121angb{8}(q)...
    I4121angb{9}(q)] = Bias_Calc(res_step,I4121ang(:,:,q));
[I4131angb{1}(:,:,q) I4131angb{2}(q) I4131angb{3}(q) I4131angb{4}(q)...
    I4131angb{5}(q) I4131angb{6}(q) I4131angb{7}(q) I4131angb{8}(q)...
    I4131angb{9}(q)] = Bias_Calc(res_step,I4131ang(:,:,q));
[I4211angb{1}(:,:,q) I4211angb{2}(q) I4211angb{3}(q) I4211angb{4}(q)...
    I4211angb{5}(q) I4211angb{6}(q) I4211angb{7}(q) I4211angb{8}(q)...
    I4211angb{9}(q)] = Bias_Calc(res_step,I4211ang(:,:,q));
[I4221angb{1}(:,:,q) I4221angb{2}(q) I4221angb{3}(q) I4221angb{4}(q)...
    I4221angb{5}(q) I4221angb{6}(q) I4221angb{7}(q) I4221angb{8}(q)...
    I4221angb{9}(q)] = Bias_Calc(res_step,I4221ang(:,:,q));
[I4231angb{1}(:,:,q) I4231angb{2}(q) I4231angb{3}(q) I4231angb{4}(q)...
    I4231angb{5}(q) I4231angb{6}(q) I4231angb{7}(q) I4231angb{8}(q)...
    I4231angb{9}(q)] = Bias_Calc(res_step,I4231ang(:,:,q));

[v111angb{1}(:,:,q) v111angb{2}(q) v111angb{3}(q) v111angb{4}(q)...
    v111angb{5}(q) v111angb{6}(q) v111angb{7}(q) v111angb{8}(q)...
    v111angb{9}(q)] = Bias_Calc(res_step,v111ang(:,:,q));
[v121angb{1}(:,:,q) v121angb{2}(q) v121angb{3}(q) v121angb{4}(q)...
    v121angb{5}(q) v121angb{6}(q) v121angb{7}(q) v121angb{8}(q)...
    v121angb{9}(q)] = Bias_Calc(res_step,v121ang(:,:,q));
[v131angb{1}(:,:,q) v131angb{2}(q) v131angb{3}(q) v131angb{4}(q)...
    v131angb{5}(q) v131angb{6}(q) v131angb{7}(q) v131angb{8}(q)...
    v131angb{9}(q)] = Bias_Calc(res_step,v131ang(:,:,q));

[v211angb{1}(:,:,q) v211angb{2}(q) v211angb{3}(q) v211angb{4}(q)...
    v211angb{5}(q) v211angb{6}(q) v211angb{7}(q) v211angb{8}(q)...
    v211angb{9}(q)] = Bias_Calc(res_step,v211ang(:,:,q));
[v221angb{1}(:,:,q) v221angb{2}(q) v221angb{3}(q) v221angb{4}(q)...
```

```
        v221angb{5}(q) v221angb{6}(q) v221angb{7}(q) v221angb{8}(q)...
        v221angb{9}(q)] = Bias_Calc(res_step,v221ang(:,:,q));
    [v231angb{1}(:,:,q) v231angb{2}(q) v231angb{3}(q) v231angb{4}(q)...
        v231angb{5}(q) v231angb{6}(q) v231angb{7}(q) v231angb{8}(q)...
        v231angb{9}(q)] = Bias_Calc(res_step,v231ang(:,:,q));

    [v311angb{1}(:,:,q) v311angb{2}(q) v311angb{3}(q) v311angb{4}(q)...
        v311angb{5}(q) v311angb{6}(q) v311angb{7}(q) v311angb{8}(q)...
        v311angb{9}(q)] = Bias_Calc(res_step,v311ang(:,:,q));
    [v321angb{1}(:,:,q) v321angb{2}(q) v321angb{3}(q) v321angb{4}(q)...
        v321angb{5}(q) v321angb{6}(q) v321angb{7}(q) v321angb{8}(q)...
        v321angb{9}(q)] = Bias_Calc(res_step,v321ang(:,:,q));
    [v331angb{1}(:,:,q) v331angb{2}(q) v331angb{3}(q) v331angb{4}(q)...
        v331angb{5}(q) v331angb{6}(q) v331angb{7}(q) v331angb{8}(q)...
        v331angb{9}(q)] = Bias_Calc(res_step,v331ang(:,:,q));

    [v112angb{1}(:,:,q) v112angb{2}(q) v112angb{3}(q) v112angb{4}(q)...
        v112angb{5}(q) v112angb{6}(q) v112angb{7}(q) v112angb{8}(q)...
        v112angb{9}(q)] = Bias_Calc(res_step,v112ang(:,:,q));
    [v122angb{1}(:,:,q) v122angb{2}(q) v122angb{3}(q) v122angb{4}(q)...
        v122angb{5}(q) v122angb{6}(q) v122angb{7}(q) v122angb{8}(q)...
        v122angb{9}(q)] = Bias_Calc(res_step,v122ang(:,:,q));
    [v132angb{1}(:,:,q) v132angb{2}(q) v132angb{3}(q) v132angb{4}(q)...
        v132angb{5}(q) v132angb{6}(q) v132angb{7}(q) v132angb{8}(q)...
        v132angb{9}(q)] = Bias_Calc(res_step,v132ang(:,:,q));

    [v411angb{1}(:,:,q) v411angb{2}(q) v411angb{3}(q) v411angb{4}(q)...
        v411angb{5}(q) v411angb{6}(q) v411angb{7}(q) v411angb{8}(q)...
        v411angb{9}(q)] = Bias_Calc(res_step,v411ang(:,:,q));
    [v421angb{1}(:,:,q) v421angb{2}(q) v421angb{3}(q) v421angb{4}(q)...
        v421angb{5}(q) v421angb{6}(q) v421angb{7}(q) v421angb{8}(q)...
        v421angb{9}(q)] = Bias_Calc(res_step,v421ang(:,:,q));
    [v431angb{1}(:,:,q) v431angb{2}(q) v431angb{3}(q) v431angb{4}(q)...
        v431angb{5}(q) v431angb{6}(q) v431angb{7}(q) v431angb{8}(q)...
        v431angb{9}(q)] = Bias_Calc(res_step,v431ang(:,:,q));



    q = q+1;

end
phi=0:res_step:2*pi;
theta=0:res_step:pi;

[PHI,THETA]=meshgrid(phi,theta);

X = cos(PHI).*sin(THETA);
Y = sin(PHI).*sin(THETA);
Z = cos(THETA);

end
```

B.5 Chapter_4_Figures.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Name: Chapter_4_Figures.m                                                    %
% Date: 08 August 2011                                                         %
% Author: Curtis Wiederhold                                                    %
%                                                                              %
% Description: This Matlab script plots the figures in Chapter 4               %
% of my thesis. Data must be loaded from an optimization scenario              %
% run using "Genetic_Aglorithm.m"                                              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Figures 4-1, 4-3, 4-5, 4-9

% Need to clear "alpha" because it is a variable in the genetic algorithm,
% but it is also a Matlab function that is used here.
clear alpha

figure
for n=1:runs
% for n=7
    clf

    dotsize = 7;
    linesize = 5;

    % Rotate mic positions if you want the mics in a different orientation
    % (such as having mic 1 be on top of the sphere).
    mic1(1,1) = sind(opt(1,n)).*cosd(opt(2,n));
    mic1(2,1) = sind(opt(1,n)).*sind(opt(2,n));
    mic1(3,1) = cosd(opt(1,n));
    mic2(1,1) = sind(opt(3,n)).*cosd(opt(4,n));
    mic2(2,1) = sind(opt(3,n)).*sind(opt(4,n));
    mic2(3,1) = cosd(opt(3,n));
    mic3(1,1) = sind(opt(5,n)).*cosd(opt(6,n));
    mic3(2,1) = sind(opt(5,n)).*sind(opt(6,n));
    mic3(3,1) = cosd(opt(5,n));
    mic4(1,1) = sind(opt(7,n)).*cosd(opt(8,n));
    mic4(2,1) = sind(opt(7,n)).*sind(opt(8,n));
    mic4(3,1) = cosd(opt(7,n));

%     r12 = norm(mic2-mic1);
%     r13 = norm(mic3-mic1);
%     r14 = norm(mic4-mic1);
%     r23 = norm(mic3-mic2);
%     r24 = norm(mic4-mic2);
%     r34 = norm(mic4-mic3);
%
%     mic1dist = r12+r13+r14;
%     mic2dist = r12+r23+r24;
%     mic3dist = r13+r23+r34;
%     mic4dist = r14+r24+r34;
%
%     dist = [mic1dist mic2dist mic3dist mic4dist];
%     smallmic = find(min(dist)==dist);
```

```
%     for m=1:2:size(opt,1)
%         opt(m,n)=opt(m,n)-opt(2*smallmic-1,n);
%         opt(m+1,n)=opt(m+1,n)-opt(2*smallmic,n);
%     end

    % Specify angle in degrees for rotation about each axis.
    xang = 0*pi/180;
    yang = 0*pi/180;
    zang = 0*pi/180;

    Rx = [1 0 0; 0 cos(xang) -sin(xang); 0 sin(xang) cos(xang)];
    Ry = [cos(yang) 0 sin(yang); 0 1 0; -sin(yang) 0 cos(yang)];
    Rz = [cos(zang) -sin(zang) 0; sin(zang) cos(zang) 0; 0 0 1];

    mic1 = Rx*Ry*Rz*mic1;
    mic2 = Rx*Ry*Rz*mic2;
    mic3 = Rx*Ry*Rz*mic3;
    mic4 = Rx*Ry*Rz*mic4;



    axes('position',[-.15 -.15 1.3 1.3])
    title(['Design Number ',int2str(n)])
    [x y z] = sphere(60);
    patch(surf2patch(x,y,z,z),'EdgeColor',[.1 .1 .1],'FaceColor',...
        [.7 .7 .7],'LineWidth',.4);
    view([180 30])
    axis vis3d
    axis off
    alpha(0.5);
    hold on
    plot3(mic1(1),mic1(2),mic1(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic2(1),mic2(2),mic2(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic3(1),mic3(2),mic3(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic4(1),mic4(2),mic4(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(linspace(mic1(1),mic2(1),50),linspace(mic1(2),mic2(2),50),...
        linspace(mic1(3),mic2(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic3(1),50),linspace(mic1(2),mic3(2),50),...
        linspace(mic1(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic4(1),50),linspace(mic1(2),mic4(2),50),...
        linspace(mic1(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic3(1),50),linspace(mic2(2),mic3(2),50),...
        linspace(mic2(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic4(1),50),linspace(mic2(2),mic4(2),50),...
        linspace(mic2(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic3(1),mic4(1),50),linspace(mic3(2),mic4(2),50),...
        linspace(mic3(3),mic4(3),50),'-k','LineWidth',linesize)
```

```
%     % Uncomment for Figures 4-5, 4-9.
%     % Puts in black patch plane for half-space designs.
%     v = 1.3;
%     patch([v v -v -v],[0 0 0 0],[v -v -v v],'k')

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [2 1 2 2]);

    print -dtiffn -r600 Figure

    pause
end

%% Figures 4-6(a) and 4-7(a)

close all;

cmin = 0;
% cmax = 30;
cmax = 6;

clear alpha;
set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',12);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);

res_step = pi/50;

phi=0:res_step:2*pi;
theta=0:res_step:pi;

[PHI,THETA]=meshgrid(phi,theta);

X = cos(PHI).*sin(THETA);
Y = sin(PHI).*sin(THETA);
Z = cos(THETA);

dotsize = 7;
linesize = 5;

% Specify which run's optimized design you want to use.
n=1;

mic1(1,1) = sind(opt(1,n)).*cosd(opt(2,n));
mic1(2,1) = sind(opt(1,n)).*sind(opt(2,n));
mic1(3,1) = cosd(opt(1,n));
mic2(1,1) = sind(opt(3,n)).*cosd(opt(4,n));
mic2(2,1) = sind(opt(3,n)).*sind(opt(4,n));
mic2(3,1) = cosd(opt(3,n));
mic3(1,1) = sind(opt(5,n)).*cosd(opt(6,n));
```

```
mic3(2,1) = sind(opt(5,n)).*sind(opt(6,n));
mic3(3,1) = cosd(opt(5,n));
mic4(1,1) = sind(opt(7,n)).*cosd(opt(8,n));
mic4(2,1) = sind(opt(7,n)).*sind(opt(8,n));
mic4(3,1) = cosd(opt(7,n));

figure
% for n=1:length(f)
    clf
    axes('position',[-.12 -.04 1.1 1.1])
    surf(X,Y,Z,I4121b{1}(:,:,find(f==4500)),'Edgecolor','none')
    alpha(.5)
    hold on
    plot3(mic1(1),mic1(2),mic1(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic2(1),mic2(2),mic2(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic3(1),mic3(2),mic3(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic4(1),mic4(2),mic4(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(linspace(mic1(1),mic2(1),50),linspace(mic1(2),mic2(2),50),...
        linspace(mic1(3),mic2(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic3(1),50),linspace(mic1(2),mic3(2),50),...
        linspace(mic1(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic4(1),50),linspace(mic1(2),mic4(2),50),...
        linspace(mic1(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic3(1),50),linspace(mic2(2),mic3(2),50),...
        linspace(mic2(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic4(1),50),linspace(mic2(2),mic4(2),50),...
        linspace(mic2(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic3(1),mic4(1),50),linspace(mic3(2),mic4(2),50),...
        linspace(mic3(3),mic4(3),50),'-k','LineWidth',linesize)
    shading interp
%     axis ([-maxval maxval -maxval maxval -maxval maxval])
    axis vis3d
    axis off
    view([95 10])
%     ytickvect = [0 10 20 30];
%     yticklabel = {'0','10','20','30 degrees'};
    ytickvect = [0 2 4 6];
    yticklabel = {'0','2','4','6 dB'};
%     colorbar('position',[.75 .3 .02 .4])
    colorbar('position',[.75 .3 .02 .4],'YTick',ytickvect,'YTickLabel',...
        yticklabel)
    caxis([cmin cmax])

    % Puts in black patch plane for half-space designs.
    v = 1.3;
    patch([v v -v -v],[0 0 0 0],[v -v -v v],'k')

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [2 1 4 4]);
    % set(gcf, 'Position', [200 100 450 550]);
    % set(gcf,'PaperPositionMode','auto')
```

```
    print -dtiffn -r600 Figure

%     pause
% end

%% Figures 4-6(b) and 4-7(b)

close all;

cmin = 0;
% cmax = 30;
cmax = 6;

clear alpha;

res_step = pi/50;

phi=0:res_step:2*pi;
theta=0:res_step:pi;

[PHI,THETA]=meshgrid(phi,theta);

X = cos(PHI).*sin(THETA);
Y = sin(PHI).*sin(THETA);
Z = cos(THETA);

dotsize = 7;
linesize = 5;

mictheta(1) = 0;
micphi(1) = 0;
mictheta(2) = 1.910633237;
micphi(2) = -pi/3;
mictheta(3) = 1.910633237;
micphi(3) = pi;
mictheta(4) = 1.910633237;
micphi(4) = pi/3;

mic1(1,1) = sin(mictheta(1)).*cos(micphi(1));
mic1(2,1) = sin(mictheta(1)).*sin(micphi(1));
mic1(3,1) = cos(mictheta(1));
mic2(1,1) = sin(mictheta(2)).*cos(micphi(2));
mic2(2,1) = sin(mictheta(2)).*sin(micphi(2));
mic2(3,1) = cos(mictheta(2));
mic3(1,1) = sin(mictheta(3)).*cos(micphi(3));
mic3(2,1) = sin(mictheta(3)).*sin(micphi(3));
mic3(3,1) = cos(mictheta(3));
mic4(1,1) = sin(mictheta(4)).*cos(micphi(4));
mic4(2,1) = sin(mictheta(4)).*sin(micphi(4));
mic4(3,1) = cos(mictheta(4));

figure
% for n=1:length(freq)
    clf
```

```
    axes('position',[-.12 -.04 1.1 1.1])
    surf(X,Y,Z,I2121b{1}(:,:,end),'Edgecolor','none')
    alpha(.5)
    hold on
    plot3(mic1(1),mic1(2),mic1(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic2(1),mic2(2),mic2(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic3(1),mic3(2),mic3(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic4(1),mic4(2),mic4(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(linspace(mic1(1),mic2(1),50),linspace(mic1(2),mic2(2),50),...
        linspace(mic1(3),mic2(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic3(1),50),linspace(mic1(2),mic3(2),50),...
        linspace(mic1(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic4(1),50),linspace(mic1(2),mic4(2),50),...
        linspace(mic1(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic3(1),50),linspace(mic2(2),mic3(2),50),...
        linspace(mic2(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic4(1),50),linspace(mic2(2),mic4(2),50),...
        linspace(mic2(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic3(1),mic4(1),50),linspace(mic3(2),mic4(2),50),...
        linspace(mic3(3),mic4(3),50),'-k','LineWidth',linesize)
    shading interp
%     axis ([-maxval maxval -maxval maxval -maxval maxval])
    axis vis3d
    axis off
    view([95 10])
%     ytickvect = [0 10 20 30];
%     yticklabel = {'0','10','20','30 degrees'};
    ytickvect = [0 2 4 6];
    yticklabel = {'0','2','4','6 dB'};
%     colorbar('position',[.75 .3 .02 .4])
    colorbar('position',[.75 .3 .02 .4],'YTick',ytickvect,...
        'YTickLabel',yticklabel)
    caxis([cmin cmax])

    % Puts in patch plane for half-space designs
    v = 1.3;
    patch([v v -v -v],[0 0 0 0],[v -v -v v],'FaceColor','none',...
        'EdgeColor','none')

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [2 1 4 4]);

    print -dtiffn -r600 Figure

%     pause
% end

%% Figures 4-6(c) and 4-7(c)

close all;
```

```
cmin = 0;
% cmax = 30;
cmax = 6;

clear alpha;

res_step = pi/50;

phi=0:res_step:2*pi;
theta=0:res_step:pi;

[PHI,THETA]=meshgrid(phi,theta);

X = cos(PHI).*sin(THETA);
Y = sin(PHI).*sin(THETA);
Z = cos(THETA);

dotsize = 7;
linesize = 5;

mictheta(1) = acos(-1/sqrt(3));
micphi(1) = atan2(-1/sqrt(3),1/sqrt(3));
mictheta(2) = acos(-1/sqrt(3));
micphi(2) = atan2(1/sqrt(3),-1/sqrt(3));
mictheta(3) = acos(-1/sqrt(3));
micphi(3) = atan2(-1/sqrt(3),-1/sqrt(3));
mictheta(4) = acos(1/sqrt(3));
micphi(4) = atan2(-1/sqrt(3),-1/sqrt(3));

mic1(1,1) = sin(mictheta(1)).*cos(micphi(1));
mic1(2,1) = sin(mictheta(1)).*sin(micphi(1));
mic1(3,1) = cos(mictheta(1));
mic2(1,1) = sin(mictheta(2)).*cos(micphi(2));
mic2(2,1) = sin(mictheta(2)).*sin(micphi(2));
mic2(3,1) = cos(mictheta(2));
mic3(1,1) = sin(mictheta(3)).*cos(micphi(3));
mic3(2,1) = sin(mictheta(3)).*sin(micphi(3));
mic3(3,1) = cos(mictheta(3));
mic4(1,1) = sin(mictheta(4)).*cos(micphi(4));
mic4(2,1) = sin(mictheta(4)).*sin(micphi(4));
mic4(3,1) = cos(mictheta(4));

figure
% for n=1:length(freq)
    clf
    axes('position',[-.12 -.04 1.1 1.1])
    surf(X,Y,Z,I1321b{1}(:,:,end),'Edgecolor','none')
    alpha(.5)
    hold on
    plot3(mic1(1),mic1(2),mic1(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic2(1),mic2(2),mic2(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
```

```
    plot3(mic3(1),mic3(2),mic3(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(mic4(1),mic4(2),mic4(3),'ob','MarkerFaceColor','b',...
        'MarkerSize',dotsize,'MarkerEdgeColor','none')
    plot3(linspace(mic1(1),mic2(1),50),linspace(mic1(2),mic2(2),50),...
        linspace(mic1(3),mic2(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic3(1),50),linspace(mic1(2),mic3(2),50),...
        linspace(mic1(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic1(1),mic4(1),50),linspace(mic1(2),mic4(2),50),...
        linspace(mic1(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic3(1),50),linspace(mic2(2),mic3(2),50),...
        linspace(mic2(3),mic3(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic2(1),mic4(1),50),linspace(mic2(2),mic4(2),50),...
        linspace(mic2(3),mic4(3),50),'-k','LineWidth',linesize)
    plot3(linspace(mic3(1),mic4(1),50),linspace(mic3(2),mic4(2),50),...
        linspace(mic3(3),mic4(3),50),'-k','LineWidth',linesize)
    shading interp
%     axis ([-maxval maxval -maxval maxval -maxval maxval])
    axis vis3d
    axis off
    view([41 20])
%     ytickvect = [0 10 20 30];
%     yticklabel = {'0','10','20','30 degrees'};
    ytickvect = [0 2 4 6];
    yticklabel = {'0','2','4','6 dB'};
%     colorbar('position',[.75 .3 .02 .4])
    colorbar('position',[.75 .3 .02 .4],'YTick',ytickvect,...
        'YTickLabel',yticklabel)
    caxis([cmin cmax])

    % Puts in patch plane for half-space designs.
    v = 1.3;
    patch([v v -v -v],[0 0 0 0],[v -v -v v],'FaceColor','none',...
        'EdgeColor','none')

    set(gcf, 'PaperPositionMode', 'manual');
    set(gcf, 'PaperUnits', 'inches');
    set(gcf, 'PaperPosition', [2 1 4 4]);

    print -dtiffn -r600 Figure

%     pause
% end
%% Figures 4-2, 4-4, and 4-8

set(0,'DefaultAxesFontName','Arial');
set(0,'DefaultAxesFontSize',10);
set(0,'DefaultAxesFontWeight','normal')
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1.5);
set(0,'DefaultLineMarkersize',8);
colvect=[.4 .4 1; .85 .16 0; .23 .43 .33; 0 0 0; 0 .75 .75;...
    .42 .25 .4; 0,0,0; 0,0,1; 0.87 .5 0];
set(0,'DefaultAxesColorOrder',colvect);
```

```
xlimit = pi/2;
xtickvect = [0 0.4 0.8 1.2];
ylimit = 7;
ytickvect = [0 2 4 6];
ylimit2 = 15;
ytickvect2 = [0 4 8 12];
labely1 = 'Average Error (dB)';
labely2 = 'Max Error (dB)';
% labely = 'Error (degrees)';
% labelx = 'ka';
labelx1 = '3/2 ka';
labelx2 = '3/2 ka';

a = .0254/2;
c = 343;

ka = 2*pi*f*3/2*a/c;

figure


% subplot(222)
axes('position',[.6 .55 .38 .44])
plot(ka,I4121angb{9},'-',ka,I2121angb{9},'--',ka,I1321angb{9},'-.')
set(gca,'XTickLabel','')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect2)
% xlabel('ka')
ylabel('Average Error (degrees)')
% legend('Optimized Design','4R.S/4','4O.S/4','Location','NorthWest')
xlim([0 xlimit])
ylim([0 ylimit2])
grid on
text(.1,.23,'(b)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(221)
axes('position',[.07 .55 .38 .44])
plot(ka,I4121b{9},'-',ka,I2121b{9},'--',ka,I1321b{9},'-.')
% xlabel('ka')
ylabel('Average Error (dB)')
legend('Optimized Design','4R.S/4','4O.S/4','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'XTickLabel','')
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(a)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')


% subplot(223)
axes('position',[.07 .09 .38 .44])
plot(ka,I4121b{4},'-',ka,I2121b{4},'--',ka,I1321b{4},'-.')
```

```
xlabel(labelx1)
ylabel('Max Error (dB)');
% legend('One','Average','Weighted','Location','NorthWest')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect)
xlim([0 xlimit])
ylim([0 ylimit])
grid on
text(.1,.23,'(c)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

% subplot(224)
axes('position',[.6 .09 .38 .44])
plot(ka,I4121angb{4},'-',ka,I2121angb{4},'--',ka,I1321angb{4},'-.')
set(gca,'XTick',xtickvect)
set(gca,'YTick',ytickvect2)
xlabel(labelx2)
ylabel('Max Error (degrees)')
xlim([0 xlimit])
ylim([0 ylimit2])
grid on
text(.1,.23,'(d)','Units','inches','FontSize',14,'FontName','Times',...
    'FontWeight','demi')

set(gcf, 'PaperPositionMode', 'manual');
set(gcf, 'PaperUnits', 'inches');
set(gcf, 'PaperPosition', [2 1 4.5 4]);
% set(gcf, 'Position', [200 100 450 550]);
% set(gcf,'PaperPositionMode','auto')
print -dtiffn -r200 testtesttest

%% Plot average and best fitnesses per generation

figure
for s=1:runs
    subplot(211)
    plot(1:M,avgfitnesses(:,s))
    xlim([1 M])

    subplot(212)
    plot(1:M,bestfitnesses(:,s))
    xlim([1 M])
    pause
end
```