# ANALYSIS AND COMPARISON OF THREE ACOUSTIC ENERGY DENSITY PROBES

By

Lance L. Locey

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Physics and Astronomy

Brigham Young University

December 2004

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Lance L. Locey

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| _____ | _____ |
| Date | Scott D. Sommerfeldt, chair |
| _____ | _____ |
| Date | Jonathan D. Blotter |
| _____ | _____ |
| Date | Timothy W. Leishman |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Lance L. Locey in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript was satisfactory to the graduate committee and was ready for submission to the university library.

_____ _____

Date

Scott D. Sommerfeldt
Chair, Graduate Committee

Accepted for the Department

_____

Ross L. Spencer
Graduate Coordinator

Accepted for the College

_____

G. Rex Bryce, Associate Dean
College of Physical and
Mathematical Sciences

ABSTRACT

ANALYSIS AND COMPARISON OF THREE ACOUSTIC ENERGY DENSITY PROBES

Lance L. Locey

Department of Physics and Astronomy

Master of Science

Traditional methods for the investigation of sound fields generally rely on a microphone to convert sound pressure into an electrical signal which can be recorded, displayed, and so forth. The squared sound pressure is directly related to potential energy density. Consequently, the measurement of sound pressure alone does not inherently provide insight into the total energy density of the sound field. Specifically, no information about the kinetic energy density of the sound field is available from this measurement alone. However, it is possible to use two microphones to estimate particle velocity. The squared particle velocity magnitude is directly related to kinetic energy density. The two energy quantities combine to yield total acoustic energy density.

The purpose of this work is to investigate and compare three probes designed to measure acoustic energy density. It is also to determine which probe may be most practically implemented in real-world applications. All three designs are based on a rigid

spherical housing and are referred to as follows: the six microphone probe, the tetrahedron probe, and the orthogonal probe. The six microphone probe is so named because it is made up of six microphones, with one pair of microphones oriented along each Cartesian axis. The tetrahedron probe is so named because it has four microphones, one at each of the vertices of a regular tetrahedron. The orthogonal probe has four microphones positioned in such a way that the lines drawn from an origin microphone to the other microphones form an orthogonal set.

The majority of the work presented in this thesis uses Matlab to numerically predict the behavior of the probes. Four numeric models are used to predict the behavior of the three different probes. The models match the geometric arrangement of the various probes. A simple experiment also shows how the probes respond to a source in an anechoic environment.

The results of the numeric modeling indicate that the orthogonal probe has the greatest useable frequency range. Both the tetrahedron and the orthogonal probes have a greater frequency range than the six microphone probe. However, in the simple experiment, the orthogonal probe did not measure energy density as accurately as the tetrahedron probe when the orthogonal probe was rotated, such that no Cartesian axis was parallel to the radial axis of the source. The data indicate that more work must be done before a decision can be made between the tetrahedron probe and the orthogonal probe. It is clear that it is possible to measure acoustic energy density in 3-dimensions using only four microphones, instead of six.

# ACKNOWLEDGMENTS

I would like to acknowledge several people. The first person I would like to acknowledge is my advisor, Scott Sommerfeldt. Dr. Sommerfeldt spent countless hours teaching me how to think critically and creatively. He rarely gave me the answer outright; rather he would ask me questions that would direct me to discover the correct answer on my own. In doing so, Dr. Sommerfeldt taught me how to formulate questions that would lead to the correct solution. The second person I would like to acknowledge is Timothy Leishman. Dr. Leishman worked with me on several projects prior to starting this work. His example helped me develop the habit of checking myself. He also taught me the importance of always asking the question, “Do I believe this result?” Lastly, I would like to acknowledge Kent Gee. While an undergraduate, Kent acted as a mentor to me. He helped me to feel like I was a part of the Acoustics Research Group at BYU. He also helped me get my first acoustics related job as an intern at NASA Glenn Research Center in Cleveland, Ohio. These three individuals have all contributed to my success, and I am grateful to each of them.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

As acoustics has developed as a science, the typical quantity used to monitor the properties of sound fields is acoustic pressure. A major reason for this is that the pressure can be measured using a microphone, a relatively straightforward transducer. However, in many cases, acoustic pressure may not provide a complete description, or even the desired description, of the field. An alternative method of describing the field is in terms of acoustic energy density.

## 1.1 What is Acoustic Energy Density?

Acoustic energy density may be defined as the amount of sound energy per unit volume enclosed within an infinitesimally small volume at a point in space. Energy is comprised of two parts: kinetic energy and potential energy. Kinetic energy is the energy of motion. In an acoustic field, kinetic energy exists as particles of fluid oscillate about their equilibrium positions. Potential energy is the energy of position. When particles of fluid are compressed, their potential energy increases as the fluid seeks to expand. A good example of energy transfer between potential and kinetic energy is in a mass-spring system. When a spring is compressed, potential energy is stored within the spring. As the mass is released, the spring expands and potential energy in the spring is converted into kinetic energy of the mass. When the speed of the mass reaches a maximum, all the potential energy in the system has been converted into kinetic energy. As the spring stretches, the mass begins to slow as kinetic energy is converted to potential energy. Eventually, the mass will stop for a brief instant. At that point, the system will have

converted all the kinetic energy of the mass into potential energy that is stored in the spring. Fluid particles exhibit this same behavior.

As mentioned above, acoustic energy density refers to the amount of kinetic and potential energy per unit volume stored in an acoustic field within an infinitesimally small volume. It is generally impractical to measure acoustic properties at all locations within an acoustic field. However, acoustic energy density measurements yield more information at one measurement point than a traditional pressure measurement. Consequently, acoustic energy density is a more comprehensive measurement.

### 1.2 Kinetic Energy Density

Time averaged acoustic kinetic energy density is described by the following equation

$$T = \frac{\rho_0 \vec{v} \cdot \vec{v}^*}{4}, \tag{1-1}$$

where $T$ is the acoustic kinetic energy density, $\rho_0$ is the ambient density of the fluid (assumed to be air in this thesis), and $\vec{v}$ is the complex acoustic particle velocity. Acoustic kinetic energy density thus depends on the square of particle velocity magnitude. Particle velocity is a vector quantity and has magnitude, phase, and a direction.

Euler showed that particle velocity can be determined from a pressure gradient as

$$\rho_0 \frac{\partial \vec{v}(t)}{\partial t} = -\nabla p(t), \tag{1-2}$$

where $\vec{v}(t)$ and $p(t)$ are real functions of time. Then, if harmonic time dependence is assumed,

$$\vec{v} = \frac{-1}{j\omega\rho_0}\nabla p \quad , \tag{1-3}$$

where again, $\vec{v}$ is the complex acoustic particle velocity, $p$ is the complex pressure, and $\omega$ is the angular frequency. In practice, a pressure gradient in one dimension can be estimated by taking the difference in pressure between two closely spaced microphones:

$$\hat{\vec{v}} \approx \frac{\hat{p}_2 - \hat{p}_1}{j\omega\rho_0 d}\hat{d} \quad , \tag{1-4}$$

where $\hat{p}_1$ and $\hat{p}_2$ are the pressure amplitudes measured by the microphones. The complex particle velocity amplitude, $\hat{\vec{v}}$, is estimated at the midpoint between $p_1$ and $p_2$. The separation between the microphones is $d$, and the velocity is estimated along the line joining the two microphones, designated by the unit vector, $\hat{d}$. This estimation method has been shown to be sufficiently accurate to predict particle velocity for many applications,[1] and is often referred to as the two-microphone technique.

Error conditions depend on the spacing between the microphones and the wavelength of interest (as well as the accuracy of the microphones). A shorter separation distance between microphones allows for a higher frequency estimate at the expense of low frequency accuracy. When two microphones are closely spaced relative to a wavelength, the two microphones measure essentially the same pressure. The difference between them is small, and an error can occur due to microphone mismatch. A larger separation distance allows for a better lower frequency estimate, but errors enter at high frequencies due to the courser finite difference approximation. A larger separation distance limits high frequency accuracy because the gradient is estimated using only the first term of the Taylor series expansion. This term becomes less accurate as the

microphone spacing becomes larger relative to wavelength. A larger separation distance makes for a longer minimum wavelength, which leads to a lower high frequency limit. It is useful to have varying separation distances to allow for a broader frequency range when making measurements using this technique.

### 1.3 Potential Energy Density

The time averaged acoustic potential energy density goes as the pressure squared, as shown in the following equation

$$U = \frac{pp^*}{4\rho_0 c^2} = \frac{|\hat{p}|^2}{4\rho_0 c^2}, \tag{1-5}$$

where U is the acoustic potential energy density, $\hat{p}$ is the complex pressure amplitude, $\rho_0$ is the density of air, and $c$ is the speed of sound in air. Just like acoustic kinetic energy density, acoustic potential energy density is a scalar. Potential energy density is easily obtained by measuring the acoustic pressure. Pressure is measured with a microphone.

### 1.4 Relevant Work

Others have investigated some aspects of acoustic energy density. Elko investigated the scattering effects of a hard sphere intended as a probe housing.[2] He found that when estimating velocity with microphones embedded on the surface of a sphere, it is necessary to include a factor of $\frac{3}{2}$ in the separation distance when approximating the particle velocity using the two microphone method. He further found that the introduction of the sphere reduced the rise in the pressure estimate error as a function of frequency, which was then followed by a drop in the error as a function of frequency, which essentially extended the usable frequency range of the probe in question by increasing the *ka* value where the error exceeded 1 dB. Parkins furthered

Elko's work by investigating the effects of microphone mismatch in conjunction with the scattering effects of a hard sphere.[3] Parkins, et al.[4] have also used acoustic energy density as a performance function in active noise control problems. In the case of active noise control, they found that minimizing acoustic energy density yielded better global performance than minimizing the squared pressure. The Japanese company, Ono Sokki,[5] has developed an intensity probe that uses only four microphones. Although its intended purpose is to measure intensity, the probe could be used to measure energy density as well.

## 1.5 Motivations for the Research

This research is motivated by several factors. First, NASA is interested in the possibility of studying the radiated fields of rocket exhaust plumes by measuring the acoustic energy density of the sound field. It is hoped that an array of acoustic energy density sensors will provide more information about the sound field created by the exhaust gases than could be obtained with a similarly sized array of traditional microphones. Additionally, it is hoped that sound power, or sound energy per unit time, will be derivable from acoustic energy density measurements. Consequently, NASA has funded this research in an effort to promote the development of these capabilities.

Another motivation for this work is purely commercial. Larson Davis, a PCB Group company, is interested in marketing an acoustic energy density probe. To date, no such probe is commercially available. Acoustic intensity probes have the hardware capability to measure acoustic energy density, but that is not their primary purpose. Intensity probes are also considerably more expensive than the acoustic energy density probes anticipated by this study. The difference in cost is largely due to the quality and

matching of microphones used in acoustic intensity probes. To measure acoustic intensity, microphones must be very closely matched both in magnitude and phase, which dramatically increases their costs. Based on research by Parkins, it is anticipated that the precision required for acoustic intensity will not be required to accurately measure acoustic energy density.[3] It is anticipated that the acoustic energy density probes may use inexpensive electret microphones.

Lastly, this research is motivated by the fact that acoustic energy density remains largely unexplored as a means of characterizing sound fields. The majority of work done in acoustic energy density has been applied to active noise control problems. It is not known how well acoustic energy density could be used to characterize quantities related to room acoustics or other specific problems such as sound field equalization. This research will contribute to other research efforts by providing more information about acoustic energy density probes for a variety of applications.

## 1.6 Objectives of the Thesis

- **Measure Acoustic Energy Density using 4 microphones**
- **Further scattering work done by Elko and Parkins up to *ka*=2**
- **Quantify errors due to imperfectly matched microphones**
- **Demonstrate agreement between numerical model & simple experiment**

In addition to its general objective of increasing understanding of energy density measurements, this thesis has four specific objectives. The first objective is to determine if it is possible to measure acoustic energy density in three dimensions using four microphones instead of six. Additionally, this thesis will explore which of two four-microphone probe configurations yields more accurate measurements and, if so, over

what frequency range. The target error range for the probe is +/- 1 dB, where the error is a measure of the measured energy density compared to the actual energy density. If it is possible to measure three-dimensional acoustic energy density with only four microphones, the author will address what causes one probe configuration to be more accurate than the other. The second objective of this research will be to further the work done by Elko and Parkins, to investigate how scattering about a hard sphere specifically affects acoustic energy density measurements. The third objective will be to numerically investigate and quantify errors introduced when using imperfect microphones to measure acoustic energy density. The final objective is to obtain basic measurement agreement between a numerical model and a physical prototype when measuring acoustic energy density.

## 1.7 Scope of Research

This thesis is limited to several specific aspects of an overall research effort. The first area of investigation is the implementation of numeric models to predict acoustic energy density when measuring pressure at several points on a sphere. Three different microphone orientations will be investigated, corresponding to a six microphone probe, tetrahedron probe, and orthogonal probe. The first numerical model used to predict acoustic energy density uses a monopole source and ideal point sensors that mimic the geometry of the probe in question. The second model builds upon the first and includes the effects of scattering due to a hard sphere. The last part of the numerical investigation explores the effects of microphone magnitude and phase mismatch, based on work done by Parkins. The nondimensional frequencies investigated cover the range up to a value of $ka$=2, where $k$ is the acoustic wave number and $a$ is the radius of the sphere. The last

aspect of the research will be to use a simple experiment to verify that it is possible to measure acoustic energy density using prototype energy density probes that match the geometry of the numerical models.

## 1.8 Plan of Development

The next two chapters will discuss numerical and experimental methods. The numerical methods chapter describes how energy density is estimated using four models. It also addresses how bias errors in potential energy density, kinetic energy density, intensity and total energy density due to microphone magnitude and phase mismatch were investigated. The experimental methods chapter discusses how acoustic energy density is measured using prototype probes that have microphones which match the sensor layout of the numerical models. It elaborates on the equipment used in the experiment and how it was set up. Subsequent chapters present and discuss numerical and experimental results, and draw pertinent conclusions.

# CHAPTER 2

## NUMERICAL METHODS

This chapter discusses how four numeric models were used to predict the behavior of three acoustic energy density probes. The first model is based on a simple point source radiating into free space. The resultant pressure field is then sampled at four or six locations, based on a random position vector and the geometry of the probe in question. This first model will be referred to as the multi-point sensor model because it samples the pressure field at four or six points in space, but omits the hard sphere in which the microphones will eventually be embedded. The second model builds upon the first, and takes into account the effects of scattering due to a rigid sphere. The second model will be referred to as the multi-point spherical sensor model. For the second model, a plane wave is assumed to be propagating in the negative z direction, with the probe located at the origin.

The third and fourth models are based on work done by Parkins[3]. Parkins referred to his first model as the two-point sensor. This is the third model implemented in this work and it will be referred to by the same name. Parkins' second model, which he referred to as the spherical sensor, was also implemented in this work as the fourth model, and will be referred to as the two-point spherical sensor model. The author expanded Parkins' work to include both a two-point sensor and two-point spherical sensor that matches the geometry of the tetrahedron and orthogonal probes, in order to have a valid comparison of all three probe types. The two-point and two-point spherical sensors are one dimensional representations of the corresponding probe geometries.

Matlab was used to numerically predict the behavior of the acoustic energy density probes. The actual code used can be found in the Appendix. This chapter describes how the probes were modeled in Matlab and the equations used to predict energy density based on the pressure field both with and without the effects of scattering due to a hard sphere. Energy density estimates were based on complex pressure measurements made by a virtual energy density probe. In all cases, the energy density was estimated at the geometric center of the microphones.

## 2.1 Virtual Energy Density Probe

The first step in developing the Matlab models was to establish three virtual energy density probes. The virtual probes were a Matlab representation of the actual physical probes. The virtual probes had measurement, or sensor, points that corresponded to the microphone locations of the related actual probes. Each virtual probe was modeled after the appropriate corresponding physical probe design: six microphone, tetrahedron, and orthogonal. The radius of each virtual probe was designated by a variable, $a$, so that both the large and small physical probe designs could be easily modeled and compared.

## 2.2 Microphone Locations

The first virtual probe had six microphones and was referred to as the virtual six microphone probe. If the virtual six microphone probe were centered at the origin of a Cartesian coordinate system, pressure measurement locations would be located on the surface of the sphere and have coordinates given by Table 2.1, where $a$ is the radius of the sphere. Figure 2.1 shows the locations of the microphones, given a sphere with radius of 1.

| Microphone number | x coordinate | y coordinate | z coordinate |
|---|---|---|---|
| 1 | +*a* | 0 | 0 |
| 2 | -*a* | 0 | 0 |
| 3 | 0 | +*a* | 0 |
| 4 | 0 | -*a* | 0 |
| 5 | 0 | 0 | +*a* |
| 6 | 0 | 0 | -*a* |

**Table 2.1 Microphone Coordinates - Six Microphone Probe**

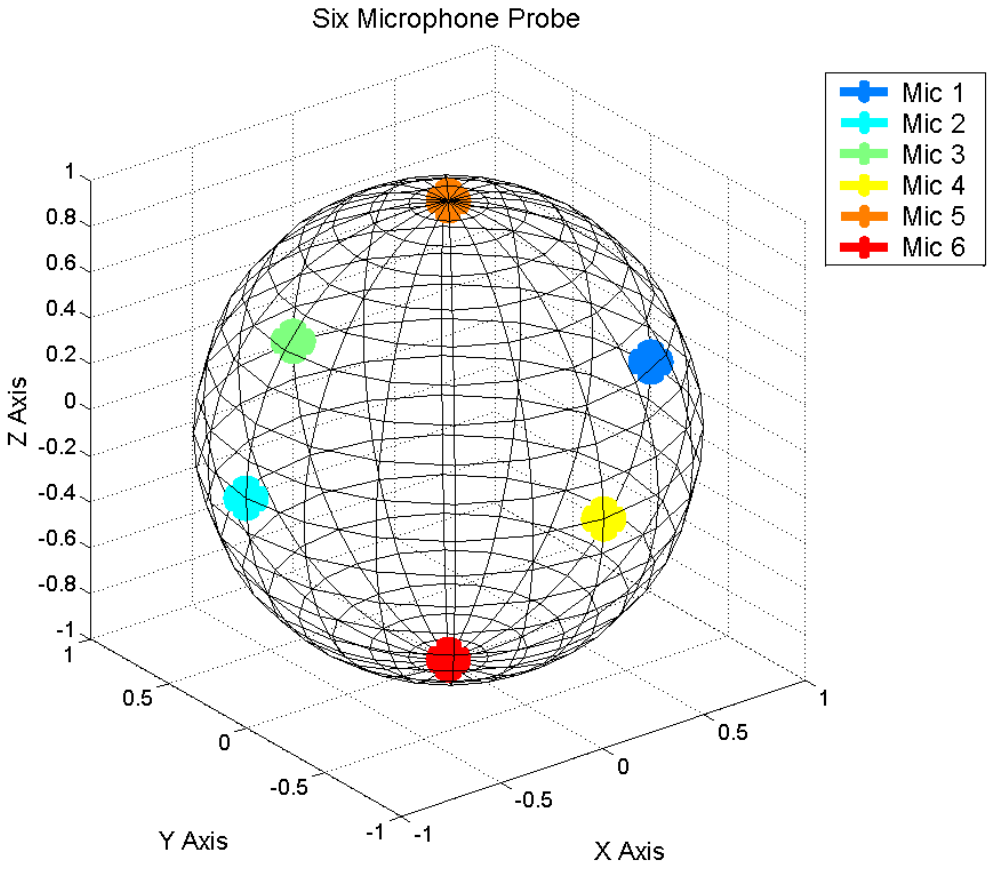


**Figure 2.1 Microphone Locations - Six Microphone Probe**

Similarly, the second virtual probe was referred to as the virtual tetrahedron probe and had pressure measurement locations at the vertices of a regular tetrahedron. Table 2.2 shows the coordinates of the tetrahedron probe microphones, given that the center of the tetrahedron was placed at the origin. Again, *a* refers to the radius of the sphere. Figure 2.2 shows the locations of the microphones, given a sphere with radius of 1.

| Microphone number | x coordinate | y coordinate | z coordinate |
|---|---|---|---|
| 1 | 0 | $\frac{2\sqrt{2}a}{3}$ | $\frac{-a}{3}$ |
| 2 | $\frac{-a\sqrt{6}}{3}$ | $\frac{-a\sqrt{2}}{3}$ | $\frac{-a}{3}$ |
| 3 | $\frac{a\sqrt{6}}{3}$ | $\frac{-a\sqrt{2}}{3}$ | $\frac{-a}{3}$ |
| 4 | 0 | 0 | $a$ |

**Table 2.2 Microphone Coordinates - Tetrahedron Probe**

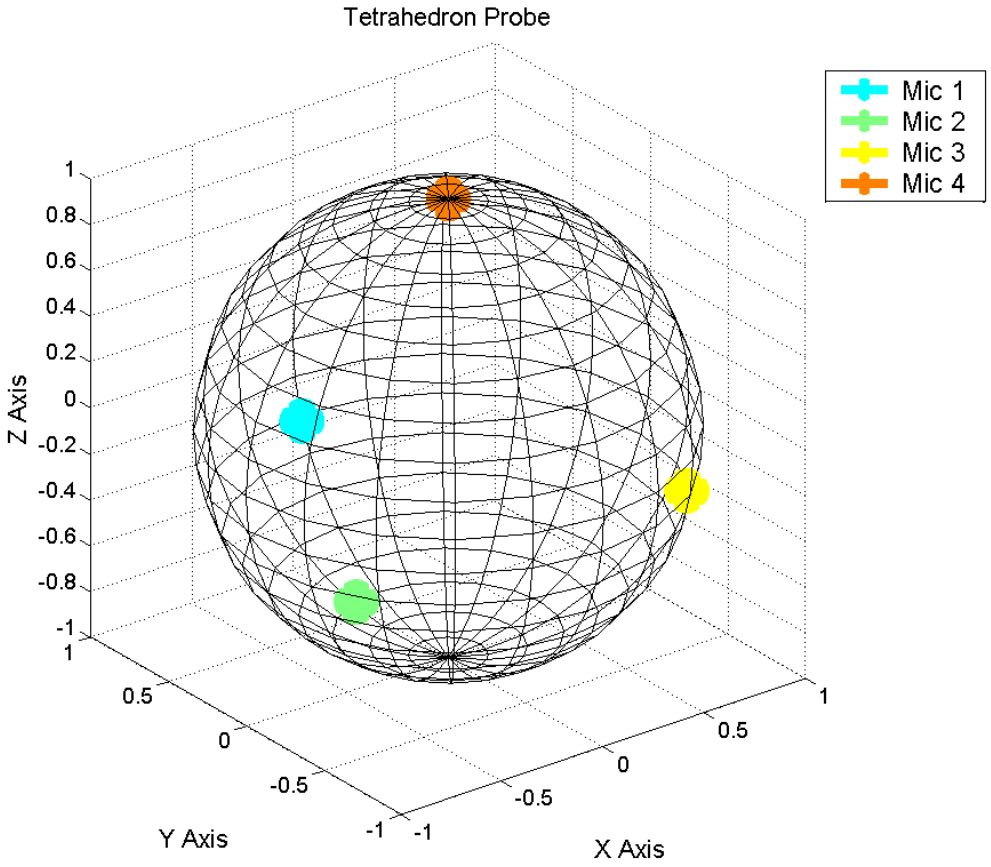


**Figure 2.2 Microphone Locations - Tetrahedron Probe**

The last virtual probe was referred to as the virtual orthogonal probe. The virtual orthogonal probe was so named because if lines were drawn between microphone number three (as indicated in the table below) and the other three microphone locations, the three lines drawn would form an orthogonal set in a Cartesian coordinate system, with microphone number 3 located at the origin of the 'local' coordinate system. The center of the orthogonal probe had coordinates of [0,0,0], and the four microphones had coordinates dictated by Table 2.3. Figure 2.3 shows the locations of the microphones, given a sphere with radius of 1.

| Microphone number | x coordinate | y coordinate | z coordinate |
|---|---|---|---|
| 1 | $\frac{a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ |
| 2 | $\frac{-a}{\sqrt{3}}$ | $\frac{a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ |
| 3 | $\frac{-a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ |
| 4 | $\frac{-a}{\sqrt{3}}$ | $\frac{-a}{\sqrt{3}}$ | $\frac{a}{\sqrt{3}}$ |

**Table 2.3 Microphone Coordinates - Orthogonal Probe**

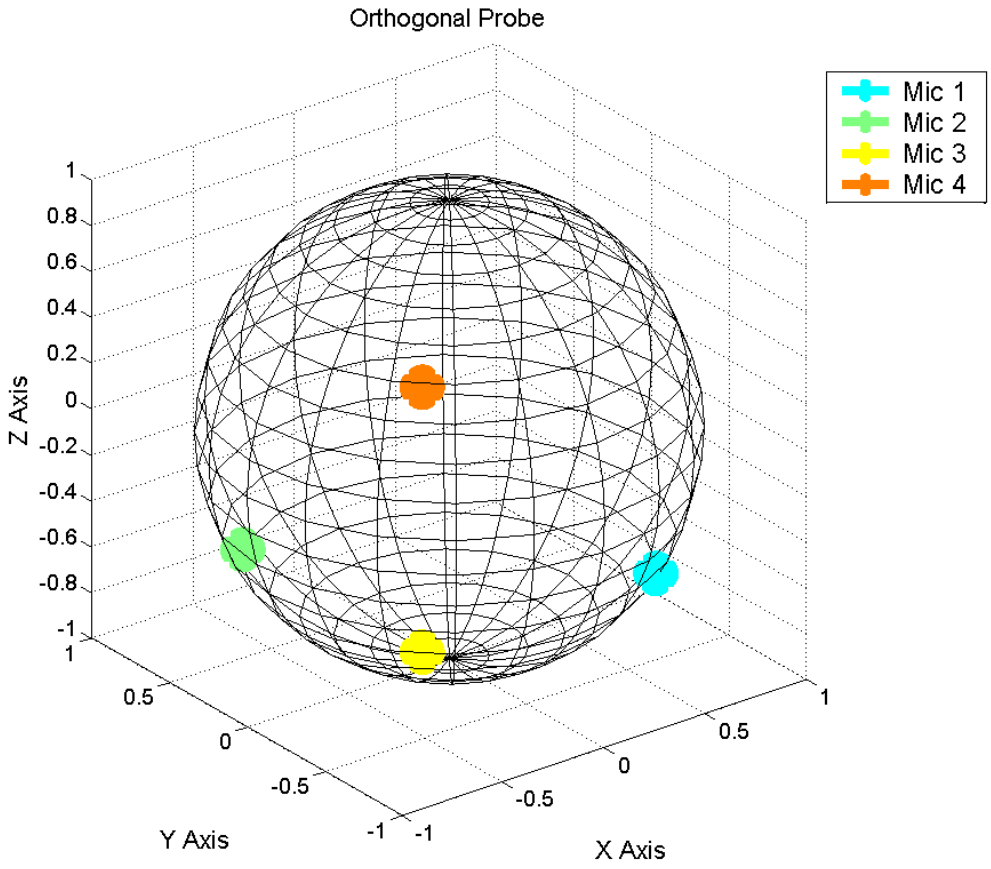


**Figure 2.3 Microphone Locations - Orthogonal Probe**

## 2.3 Multi Point and Multi Point Spherical Matlab Models

Four models were used to numerically demonstrate how the probes could be used to measure acoustic energy density. This section will discuss the first two models, the multi-point model and the multi-point spherical model. The two models differ in two ways: first, the way that the sound field was measured by the idealized sensors, and second, the inclusion of a constant when estimating velocity using microphones embedded on the surface of a sphere.[2] The simple model was referred to as the point sensor model and is similar to Parkins' point sensor model. The more complicated model was referred to as the spherical sensor model and is similar to Parkins' spherical sensor model. Once the pressure at the sensor locations was determined, the two models predicted velocity and then energy density using the same algorithm. The only difference between the two is the factor of 3/2 used when estimating the particle velocity in the presence of the hard sphere. Elko showed that this factor accounts for the scattering effects of the sphere.[2] Put another way, the first model was made up of a monopole point source at the origin. The resultant pressure field was then sampled at four or six locations, depending on the probe geometry, and the energy density was estimated at the center of virtual probe, based on the sampled pressures. The second model calculated the incident and scattered pressure on the surface of a hard sphere, due to a plane wave, at four or six locations on the sphere. It then estimated the potential energy density based on the average of the pressures at the four or six microphone locations. It also estimated the velocity at the center of the sphere and estimated the kinetic energy density. The estimated values were then compared with exact values, as described in subsection 2.4, for both models.

### 2.3.1 Pressure Estimate for Multi Point Sensor Model

The first model propagated the pressure field of a simple monopole source into free space. It then estimated the pressure at points in space specified by the geometry of the probe being modeled, according to the following equation

$$p = \frac{Ae^{-jk|\vec{r}'|}}{|\vec{r}'|} \tag{2-1}$$

where $p$ was the complex pressure, $A$ was the complex monopole amplitude, $k$ was the acoustic wave number, and $\vec{r}'$ was the position vector that points from the origin to the desired microphone location. The vector $\vec{r}'$ was determined by the following equation

$$\vec{r}' = \vec{r} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2-2}$$

where $\vec{r}$ is the vector from the origin to the center of the probe, and $x$, $y$, and $z$ are the coordinates of the microphone, relative to the center of the probe. The two equations above were used to determine the pressure at the four or six measurement points, depending on the probe type. The four or six pressure values were averaged and then used to estimate the potential energy density, as described in section 2.5.

### 2.3.2 Pressure Estimate- Multi Point Spherical Sensor Model

The second model takes into account scattering effects caused by the rigid sphere. It also takes into account the effects caused by a reflecting surface with a uniform complex reflection coefficient, $\hat{R}$. The model assumed a rigid sphere centered at the origin. Perfect point microphones were located on the surface of the sphere at locations described in the tables above. The sphere was then rotated by random angles $\alpha$, $\varphi$, and $\theta$ about the y, x, and z axes, according to the following equations:

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} R_y \end{bmatrix} \begin{bmatrix} R_x \end{bmatrix} \begin{bmatrix} R_z \end{bmatrix} \tag{2-3}$$

$$R_y(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{2-4}$$

$$R_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \sin(\varphi) \\ 0 & -\sin(\varphi) & \cos(\varphi) \end{bmatrix} \tag{2-5}$$

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2-6}$$

$$\begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} \cos(\alpha)\cos(\theta) + \sin(\alpha)\sin(\varphi)\sin(\theta) & \cos(\alpha)\sin(\theta) - \sin(\alpha)\sin(\varphi)\cos(\theta) & \sin(\alpha)\cos(\varphi) \\ -\cos(\varphi)\sin(\theta) & \cos(\varphi)\cos(\theta) & \sin(\varphi) \\ -\sin(\alpha)\cos(\theta) + \cos(\alpha)\sin(\varphi)\sin(\theta) & -\sin(\alpha)\sin(\theta) - \cos(\alpha)\sin(\varphi)\cos(\theta) & \cos(\alpha)\cos(\varphi) \end{bmatrix} \tag{2-7}$$

where $x$, $y$, and $z$ were the coordinates of the microphones as outlined in the tables above. The angles $\alpha$, $\varphi$, and $\theta$ were generated by using the Matlab random number generator, `RANDN`, multiplied by $2\pi$. The resultant random angles ranged between $-2\pi$ and $2\pi$. After rotating the sphere, a plane wave was propagated in the negative z direction, past the sphere. The angle between the z axis and each microphone was computed and designated $\theta$. This angle $\theta$, not to be confused with the angle of rotation about the z axis, was then used in the scattered pressure equation below (2-9) and the incident pressure equation (2-8) to determine the scattered and incident pressure at each microphone. The incident pressure was determined by the following equation:

$$P_{inc}(ka, \theta) = P_+\left(e^{-jka\cos(\theta)}\right) \tag{2-8}$$

where $P_+$ was the amplitude of the plane wave, $\theta$ was the angle measured from the z axis to the vector pointing from the origin to the microphone position, and $ka$ was the acoustic

wave number multiplied by the radius of the sphere. The scattered pressure was determined by

$$P_{sc}\left(ka,\theta,\hat{R}\right)=\frac{P_{+}}{j(ka)^{2}}\sum_{n=0}^{\infty}\frac{j^{n}(2n+1)\left(\hat{R}+(-1)^{n}\right)P_{n}(\cos(\theta))}{h_{n}^{(2)'}(ka)} \quad (2\text{-}9)$$

where $\theta$ was the angle made by the microphone and the z axis, $P_n(\cos\theta)$ was a Legendre function of order $n$, and $h_n^{(2)'}$ was the derivative of the spherical Hankel function of the second kind of order $n$.[6] Parkins indicated that only thirteen terms were necessary in determining the scattered pressure.[3] Including more terms did not change the results noticeably. The pressure field on the surface of the rigid sphere was created by the sum of the incident and scattered pressure[6], based on the following equation

$$P_{tot} = P_{inc} + P_{sc}\,. \quad (2\text{-}10)$$

## 2.4 Particle Velocity Estimate

### 2.4.1. Particle Velocity Number Direction System

A number and direction system was established to reduce confusion when describing particle velocity. Particle velocity is a vector quantity. As such, it can be broken down into x, y, and z components. Euler's equation can be numerically implemented to predict particle velocity at the midpoint between two closely spaced microphones. A numbering system was devised to indicate which direction would be the direction of positive particle velocity. The system starts with a capital V, which identifies the quantity as particle velocity. The V was followed by two subscript integers. The first subscript integer corresponds to the first microphone used to estimate the particle velocity. The second integer corresponds to the second microphone used to estimate the particle velocity. The particle velocity was estimated at the midpoint between the two

microphones and taken to be positive in the direction of the second subscript integer. The last subscript character corresponds to the x, y, or z component. If no component was specified, the velocity vector was in the direction of the second microphone, on the line between the two microphones. (It should be noted that the particle velocity values are probe dependent, due to the different geometries, and are not interchangeable.) For example, $V_{34y}$ corresponded to the particle velocity in the y direction, estimated at the midpoint between the third and fourth microphone. In the case of the six microphone virtual probe, the third and fourth microphone were both situated along the y axis. The third microphone was located on the positive y axis while the fourth microphone was located on the negative y axis. When particle velocity $V_{34y}$ was positive, as measured by the virtual six microphone probe, the particle velocity pointed from the origin, which was also the midpoint between microphones three and four, towards the fourth microphone, or in the negative y direction. Conversely, when particle velocity $V_{43y}$ was positive, it pointed from the midpoint toward microphone number three, which was in the positive y direction.

### 2.4.2 Particle Velocity and the Hard Sphere

Elko indicated that a frequency independent correction factor was necessary when approximating velocity by using microphones embedded in the surface of a hard sphere. That factor was $\frac{3}{2}$, and is incorporated by multiplying the microphone spacing by 3/2 in calculating the velocity estimate. Table 2.4 shows the nondimensional apparent microphone separation values, the *kd* values, used when computing the particle velocities, both with and without scattering. Both *a* and *b* represented the distance from the center of the probe to the microphone, which is also *a*, the radius of the sphere. The three

quantities were equal, but *a* referred the two-point spherical sensor, *b* referred to the two point sensor, and a referred to the multi point sensors. The orthogonal probe had two separation distances, depending on which two microphones were being compared. The separation distance between microphones 1 and 2 is greater than the separation between microphones 1 and 3, hence the need for two separation distances when computing particle velocity.

| | Two-Point Sensor (no Sphere) | Spherical Sensor |
|---|---|---|
| Six Microphone Probe | $2kb$ | $3ka$ |
| Tetrahedron Probe | $\sqrt{3}kb$ | $\frac{3\sqrt{3}ka}{2}$ |
| Orthogonal Probe Short Separation Distance | $\frac{2\sqrt{3}kb}{3}$ | $\sqrt{3}ka$ |
| Orthogonal Probe Long Separation Distance | $\sqrt{3}kb$ | $\frac{3\sqrt{3}ka}{2}$ |

**Table 2.4 Effective *kd* Values, both with (spherical sensor) and without (two-point sensor) scattering.**

### 2.4.3 Particle Velocity Estimate: Six Microphone Probe

The particle velocity estimation process was slightly more complicated than the pressure estimation process. Of the three probes, the six microphone virtual probe had the simplest geometry and the most direct method to predict particle velocity. It will be described first. As described above, the virtual six microphone probe had microphones arranged in three pairs. Each pair of microphones was situated along one coordinate axis and could be used to estimate particle velocity in that direction—the first pair of microphones estimated velocity $V_{21x}$, the second pair of microphones estimated velocity

$V_{43y}$, and the third pair of microphones estimated velocity $V_{65z}$. The difference in pressure between each pair of microphone pressure locations was computed, and yielded a total of three pressure gradient estimates. Euler's equation was then implemented to estimate the velocity, according to:

$$V_{21x} \approx \frac{p_2 - p_1}{j\omega\rho_0 d} \tag{2-11}$$

where $p_1$ and $p_2$ were the pressures measured by two closely spaced microphones, $\omega$ was the angular frequency, $\rho_0$ was the density of air, and $d$ was the separation distance between the two microphones. The particle velocity estimates $V_{43y}$ and $V_{65z}$ were estimated in the same manner as $V_{21x}$.

### 2.4.4 Particle Velocity Estimate: Tetrahedron Probe

The particle velocity estimation method for the virtual tetrahedron probe will be described next. The virtual tetrahedron probe particle velocity estimation method was compared with work done by Hori[5]. The four microphones of the virtual tetrahedron probe made up six possible pressure gradients which could be used to estimate a total of six particle velocities. These velocities were $V_{12}$, $V_{13}$, $V_{14}$, $V_{23}$, $V_{24}$, and $V_{34}$. As noted above, particle velocity was a vector quantity and could be broken down into vector components. Table 2.5 illustrates the sign of the velocity in the x, y, and z direction of each positive particle velocity vector.

| Velocity Vector | $\hat{x}$ | $\hat{y}$ | $\hat{z}$ |
|---|---|---|---|
| $V_{12}$ | Negative | Positive | None |
| $V_{13}$ | Positive | Positive | None |
| $V_{14}$ | None | Positive | Positive |
| $V_{23}$ | Positive | None | None |
| $V_{24}$ | Positive | Negative | Positive |
| $V_{34}$ | Negative | Negative | Positive |

**Table 2.5 Tetra Probe- Direction of Positive Velocity**

For example, if $V_{34}$ was positive, the x and y components of velocity would be negative, while the z component would be positive. The following equation was used to derive the velocity estimate at the center of the probe in the x direction.

$$V_x = C_1\left[V_{32x} + V_{31x} + V_{12x} + V_{34x} + V_{42x}\right] \tag{2-12}$$

The above equation was the summation of all the velocity components in the x direction. The unknown, $C_1$, was found numerically to be -½. A similar constant of -½ was necessary for the y and z direction velocity estimates as well. The various terms are adapted from Eq. (2-7) as follows:

$$V_{32x} \approx \frac{p_3 - p_2}{j\omega\rho_0 d} \tag{2-13}$$

$$V_{31x} \approx \frac{p_3 - p_1}{j\omega\rho_0 d}\sin\left(30^\circ\right) \tag{2-14}$$

$$V_{12x} \approx \frac{p_1 - p_2}{j\omega\rho_0 d}\sin\left(30^\circ\right) \tag{2-15}$$

$$V_{34x} \approx \frac{p_3 - p_4}{j\omega\rho_0 d}\frac{\cos\left(30^\circ\right)}{\sqrt{3}} \tag{2-16}$$

$$V_{42x} \approx \frac{p_4 - p_2}{j\omega\rho_0 d}\frac{\cos(30^\circ)}{\sqrt{3}} \tag{2-17}$$

Substituting these expressions into Eq. (2-12) yields

$$V_x \approx \frac{C_1(-2p_2 + 2p_3)}{j\omega\rho_0 d} \tag{2-18}$$

where again $C_1$ was -½. Note that when estimating the velocity associated with the multi-point spherical model, a factor of 3/2 must be included in the separation distance to correct for the scattering effects of the sphere, as indicated by Elko.[2] The velocity in the y direction at the center of the probe was found in a similar manner, using the follow equations:

$$V_y = C_2\left[V_{12y} + V_{13y} + V_{14y} + V_{42y} + V_{43y}\right] \tag{2-19}$$

$$V_{12y} \approx \frac{p_1 - p_2}{j\omega\rho_0 d}\cos(30^\circ) \tag{2-20}$$

$$V_{13y} \approx \frac{p_1 - p_3}{j\omega\rho_0 d}\cos(30^\circ) \tag{2-21}$$

$$V_{14y} \approx \frac{p_1 - p_4}{j\omega\rho_0 d}\left(\frac{1}{\sqrt{3}}\right) \tag{2-22}$$

$$V_{42y} \approx \frac{p_4 - p_2}{j\omega\rho_0 d}\left(\frac{1}{\sqrt{3}}\right)\sin(30^\circ) \tag{2-23}$$

$$V_{43y} \approx \frac{p_4 - p_3}{j\omega\rho_0 d}\left(\frac{1}{\sqrt{3}}\right)\sin(30^\circ) \tag{2-24}$$

The result of combining the above equations yielded

$$V_y \approx \frac{C_2}{j\omega\rho_0 d}\frac{1}{\sqrt{3}}\left[4p_1 - 2p_2 - 2p_3\right] \tag{2-25}$$

The equations used to estimate the particle velocity in the z direction were as follows:

$$V_z = C_3\left[V_{34z} + V_{24z} + V_{14z}\right] \tag{2-26}$$

$$V_{34z} = \frac{p_3 - p_4}{j\omega\rho_0 d}\frac{\sqrt{2}}{\sqrt{3}} \tag{2-27}$$

$$V_{24z} = \frac{p_2 - p_4}{j\omega\rho_0 d}\frac{\sqrt{2}}{\sqrt{3}} \tag{2-28}$$

$$V_{14z} = \frac{p_1 - p_4}{j\omega\rho_0 d}\frac{\sqrt{2}}{\sqrt{3}} \tag{2-29}$$

$$V_z = \frac{C_3}{j\omega\rho_0 d}\frac{\sqrt{2}}{\sqrt{3}}\left[3p_4 - p_3 - p_2 - p_1\right] \tag{2-30}$$

Again, $C_2$ and $C_3$ were -½. Note that when estimating the velocity associated with the spherical model, a factor of 3/2 must again be included in the separation distance to correct for the scattering effects of the sphere, as indicated by Elko.[2]

### 2.4.5 Particle Velocity Estimate: Orthogonal Probe

#### 2.4.5.1 Orthogonal Probe 'On Axis' Velocity

The particle velocity estimation method for the virtual orthogonal probe was the most complicated of the three probe designs. As in the case of the tetrahedron virtual probe, the four microphones allow for six possible pressure gradients. Three of these pressure gradients can be used to estimate velocity along the local orthogonal axes. The differences in pressure between $p_3$-$p_1$, $p_3$-$p_2$, and $p_3$-$p_4$ make up the three velocity estimates that have only one Cartesian component. These three single component velocity estimates will be termed 'on axis' velocity estimates and are referred to as $V_{31x}$, $V_{32y}$, and $V_{34z}$. They are referred to as 'on axis' velocity estimates because they each represent the estimated velocity along only one axis.

#### 2.4.5.2 Orthogonal Probe Diagonal Velocity Estimate

The other three velocity estimates are termed 'diagonal' velocity estimates and are referred to as $V_{12}$, $V_{14}$, and $V_{24}$. These velocity estimates are termed 'diagonal' velocity estimates because they estimate velocity on the diagonal between the local unit vectors and consequently these velocity estimates have components in two Cartesian directions, not just one. Subsection 2.4.5.3 will describe how particle velocity at the center of the sphere was estimated given the velocity estimates $V_{31}$, $V_{32}$, $V_{34}$, $V_{12}$, $V_{14}$, and $V_{24}$.

The diagonal velocity estimate $V_{12}$ lies in the local x-y plane. It was determined by using a numerical approximation of Euler's equation. Because the velocity was a vector quantity, it could be decomposed into an x and y component of velocity. There were several steps in this process. The first step was to use the magnitude of the on axis velocity components, $V_{31x}$ and $V_{32y}$, to estimate an angle, α given by

$$\alpha = \tan^{-1}\left(\left|\frac{V_{32y}}{V_{31x}}\right|\right). \tag{2-31}$$

The angle α represents the angle made between the projection of the unknown particle velocity onto the x-y plane. However, in determining the sign of the angle α, it was necessary to compare the phase information of the two particle velocities.

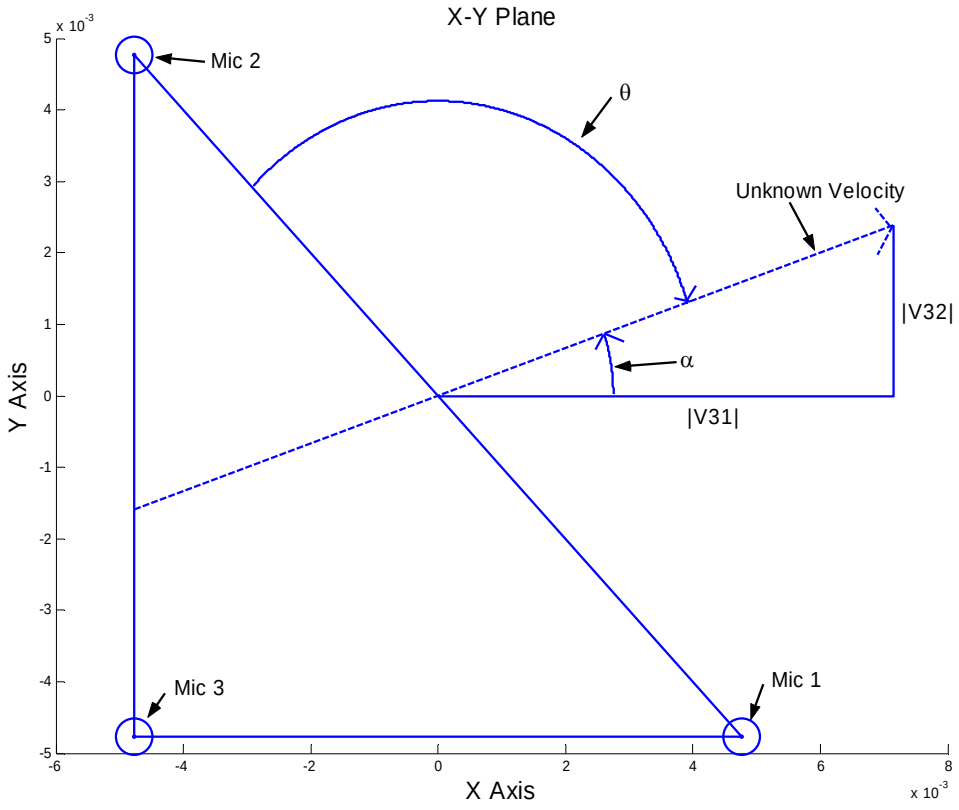


**Figure 2.4: Projection of Unknown Velocity onto X-Y Plane, Orthogonal Probe**

The sign of the angle α was reversed, depending on the phase difference between $V_{31}$ and $V_{32}$. A difference in phase between $V_{31}$ and $V_{32}$ of π/2 to 3π/2 radians necessitated that α be multiplied by -1. Once the angle α was determined, the unknown particle velocity could be estimated based on the measured diagonal velocity and a new angle θ.

The angle θ was measured in the clockwise direction from the diagonal to the projection of the unknown particle velocity onto the x-y plane. See Figure 2.4. The angle θ was determined by subtracting α from 3π/4.

$$\theta = \frac{3\pi}{4} - \alpha \tag{2-32}$$

The unknown predicted velocity, $V_{12p}$, could then be estimated. The subscript *p* referred to the predicted unknown velocity projected onto the x-y plane. Given the velocity $V_{12}$ and the angle θ, the unknown velocity $V_{12p}$ could be estimated as

$$V_{12p} = \frac{V_{12}}{\cos(\theta)} \tag{2-33}$$

The unknown velocity $V_{12p}$ could then be decomposed into x and y components by multiplying by sin(α) and cos(α).

$$V_{12x} = V_{12p}\cos(\alpha) \tag{2-34}$$

$$V_{12y} = V_{12p}\sin(\alpha) \tag{2-35}$$

The same process was used to decompose the velocity for the other two diagonal particle velocity values. Table 2.6 below shows the direction of the particle velocity when a given velocity vector was positive.

| Velocity Vector | $\hat{x}$ | $\hat{y}$ | $\hat{z}$ |
|---|---|---|---|
| $V_{12}$ | Negative | Positive | None |
| $V_{13}$ | Negative | None | None |
| $V_{14}$ | Negative | None | Positive |
| $V_{23}$ | None | Negative | None |
| $V_{24}$ | None | Negative | Positive |
| $V_{34}$ | None | None | Positive |

**Table 2.6 Orthogonal Probe- Direction of Positive velocity**

#### 2.4.5.3 Taylor Series Expansion

The velocity at the center of the virtual orthogonal probe was estimated by combining ‘on axis’ and ‘diagonal’ velocities. The three ‘diagonal’ velocities estimate

velocity in two directions. For example, $V_{12}$ has a velocity component in x as well as in y. The particle velocity $V_{14}$ has a component in x as well as in z. The particle velocity $V_{31x}$ only measures velocity in the x direction. The $V_{31x}$ component alone was not sufficient to estimate the velocity in the x direction at the center of the sphere because the midpoint between microphones three and one was not at the exact center of the sphere. To compensate for the offset in the estimate position of $V_{31x}$, the x components of $V_{12}$ and $V_{14}$ were used, in a manner similar to a Taylor series approximation. The equations below describe how this was accomplished, in the x direction. For simplicity, normalized coordinates were used. For additional clarification on the orthogonal probe, see Section 3.4. The particle velocity $V_x$ at coordinates [ ½, ½, ½] was the desired particle velocity, at the center of the probe, in the x direction.

$$V_x\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) = V_x\left(\frac{1}{2},0,0\right) + \left.\frac{\partial V_x}{\partial y}\right|_0 \delta y + \left.\frac{\partial V_x}{\partial z}\right|_0 \delta z \tag{2-36}$$

$$V_x\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) = V_x\left(\frac{1}{2},0,0\right) + \frac{\left[V_x\left(\frac{1}{2},\frac{1}{2},0\right) - V_x\left(\frac{1}{2},0,0\right)\right]\delta y}{\delta y} + \frac{\left[V_x\left(\frac{1}{2},0,\frac{1}{2}\right) - V_x\left(\frac{1}{2},0,0\right)\right]\delta z}{\delta z} \tag{2-37}$$

$$V_x\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right) = V_x\left(\frac{1}{2},\frac{1}{2},0\right) + V_x\left(\frac{1}{2},0,\frac{1}{2}\right) - V_x\left(\frac{1}{2},0,0\right) \tag{2-38}$$

The overall estimate of the particle velocity, in the x direction at the center of the probe, was

$$V_x = V_{21x} + V_{41x} - V_{31x} \tag{2-39}$$

The same technique was used for $V_y$ and $V_z$.

## 2.5 Kinetic, Potential, and Total Energy Density Estimate

Once the x, y, and z particle velocities had been estimated for all three probe types, together with the averaged pressure, it was possible to estimate kinetic, potential, and total energy density. The kinetic energy density was estimated using the following equations.

$$T_x = \frac{\rho_0 \vec{v}_x \cdot \vec{v}_x{}^*}{4} \tag{2-40}$$

$$T_y = \frac{\rho_0 \vec{v}_y \cdot \vec{v}_y{}^*}{4} \tag{2-41}$$

$$T_z = \frac{\rho_0 \vec{v}_z \cdot \vec{v}_z{}^*}{4} \tag{2-42}$$

$$T_{total} = T_x + T_y + T_z \tag{2-43}$$

The potential energy density was estimated with the following equation

$$U_{ave} = \frac{p_{ave} p_{ave}{}^*}{4\rho_0 c^2} \tag{2-44}$$

,

where $p_{ave}$ is the average complex pressure amplitude of the four or six pressures. The sum of potential and kinetic energy density yields total energy density.

$$e = U_{ave} + T_{total} \tag{2-45}$$

## 2.6 Exact Solution

Once the estimate for the total energy density had been determined, it was necessary to compare it with the exact solution. The exact solution was determined in one of two ways, depending on the numerical model. In the first model, the exact solution was determined by first finding the pressure at the center of the sphere using equations (2-1) and (2-2). Then the velocity was determined by the following equation

$$\vec{v} = \frac{\hat{r}\left(1 - \frac{j}{kr}\right)p}{\rho_0 c}, \tag{2-46}$$

where $p$ was the exact complex pressure at the center of the sphere, and $kr$ was the acoustic wave number times the radial distance from the source.[7]

In the case of the second model, the pressure at the center of the sphere was taken to be $P_+$, the plane wave source strength. The exact potential energy density is then

$$U_{exact} = \frac{P_+^{\,2}}{4\rho_0 c^2} \tag{2-47}$$

The velocity at the center of the sphere was taken to be $\frac{P_+}{\rho_0 c}$ in the negative z direction.

The rotation matrix was applied to the exact velocity and the resultant rotated velocities were used to determine the total exact kinetic energy density, using equations (2-40) through (2-43). The errors were then plotted using the bias equations below.

## 2.7 Bias Errors

The expressions for the four bias errors are defined by

$$U_{b,dB} \equiv 10\log U_{bias} = 10\log\left|\frac{\hat{P}_{ec}}{\hat{P}_c}\right|^2 \tag{2-48}$$

$$T_{b,dB} \equiv 10\log T_{bias} = 10\log\left|\frac{\hat{V}_{e\theta}}{\hat{V}_\theta}\right|^2 \tag{2-49}$$

$$e_{b,dB} \equiv 10\log e_{bias} = 10\log\left|\frac{e_{e\theta}}{e_\theta}\right| \tag{2-50}$$

$$I_{b,dB} \equiv 10\log I_{bias} = 10\log\left|\frac{I_{e\theta}}{I_\theta}\right|^2. \tag{2-51}$$

The four bias errors above were composed of both an estimate and an exact value. The exact value, the value in the denominator, was determined either at the center of the probe or in the θ direction. The estimated value, the value in the numerator, was also determined for the same location and direction.

## 2.8 Two Point and Two Point Spherical Sensor Models

Parkins[3] investigated the effects of scattering from a rigid sphere as it applies to measuring potential, kinetic, and total energy density, as well as intensity. He also investigated the effects of mismatched microphones on these measurements. Parkins' investigation was limited to frequencies up to $ka = 0.75$, where $k$ was the acoustic wave number and $a$ was the radius of the sphere. His model also took into account room effects by incorporating a complex reflection coefficient, $\hat{R}$, into his analysis. This section will describe how the author implemented Parkins' equations and extended his analysis up to $ka$=2. Additionally, this section will describe how Parkins' method was expanded to include both the tetrahedron probe and the orthogonal probe.

Parkins' defined his problem by using a single axis spherical sensor and a two-point sensor. The spherical sensor consisted of two microphones, $m_1$ and $m_2$, embedded on the surface of a hard sphere with a diameter of $2a$, and is referred to as the two-point spherical sensor. The two-point sensor consisted of two microphones separated in space by a distance $2b$. To clarify, both $a$ and $b$ were the same linear distance, which is also the radius of the sphere. Parkins used the two different variables to distinguish between the two-point sensor model, which uses separation distance $2b$, and the two-point spherical sensor model, which uses separation distance $2a$. Both the two-point sensor and the two-point spherical sensor were centered at the origin. The two-point and two-point spherical

sensor are representations of the six microphone probe. In expanding Parkins' work, the author included both a two-point sensor and a two-point spherical sensor with microphones separated by 120º to represent the tetrahedron probe. Additionally, a two-point sensor and a two-point spherical sensor with microphones separated by 70.52º were used to represent the orthogonal probe. For clarification, see Figure 2.5.

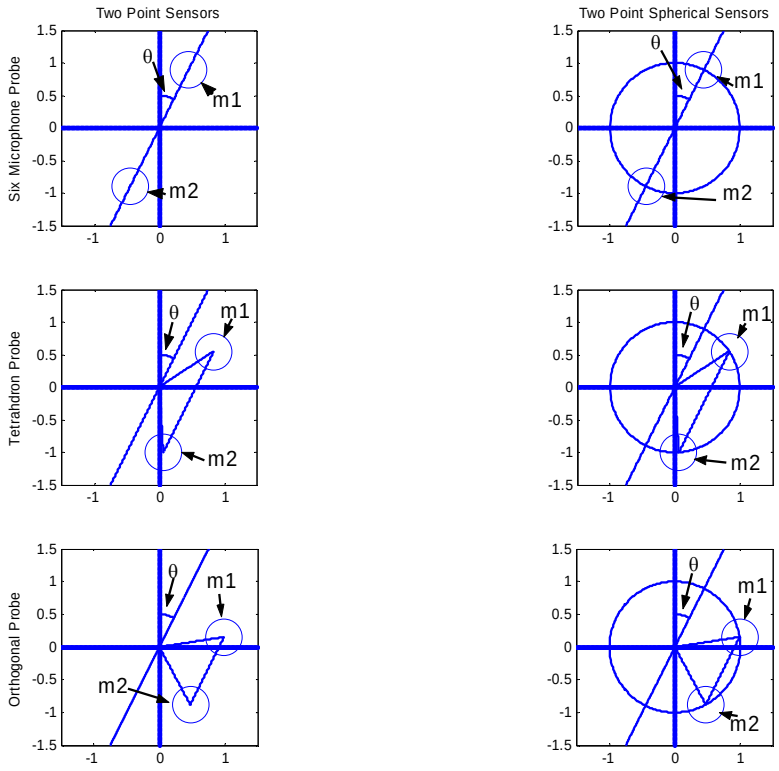


**Figure 2.5: Microphone Positions: Two-Point Sensor and Two-Point Spherical Sensor**

In his model, Parkins established two plane waves, $P_+$ and $P_-$, that were incident on both the two-point and two-point spherical sensors, from opposite directions producing a standing wave. The angle between the incident wave traveling in the negative z direction and the line joining the two microphones was θ. Parkins indicated that the errors were greatest when θ was 0, corresponding to the direction of propagation

for the plane wave. Thus, in this analysis, unless otherwise specified, θ was set to a value of 0. The complex reflection coefficient of the standing wave was defined by

$$\hat{R} \equiv \frac{\hat{P}_-}{\hat{P}_+} = \eta e^{j\xi} \tag{2-52}$$

where $\hat{P}_+$ and $\hat{P}_-$ are the complex amplitudes of the plane waves in the positive and negative z-directions at the origin, when there was no sphere present. Microphone $m_1$ was taken to have a phase mismatch, $\delta_p$, and a sensitivity mismatch, $\delta_m$, with respect to $m_2$ given by

$$\frac{\hat{P}_{m1}}{\hat{P}_{m2}} = \delta_m e^{j\delta_p} \tag{2-53}$$

when the microphones were both in the same pressure field. The sensitivity mismatch in decibels was given by

$$\delta_{m,dB} = 20\log(\delta_m) \tag{2-54}$$

In each case, only two microphones were used, so only the acoustic parameters along the microphone axis (which will be referred to as the θ component) could be measured. Parkins defined the θ components of potential energy density, kinetic energy density, total energy density, and intensity by

$$U_\theta \equiv \frac{\left|\hat{P}_c\right|^2}{12\rho_0 c^2} \tag{2-55}$$

$$T_\theta \equiv \frac{\rho_0\left|\hat{V}_\theta\right|^2}{4} \tag{2-56}$$

$$e_\theta \equiv \frac{\left|\hat{P}_c\right|^2}{12\rho_0 c^2} + \frac{\rho_0\left|\hat{V}_\theta\right|^2}{4} \tag{2-57}$$

$$I_\theta \equiv \tfrac{1}{2}\Re e\{P_c V_\theta *\}. \tag{2-58}$$

$\hat{P}_c$ was the complex pressure at the center of the sphere, and $\hat{V}_\theta$ was the complex velocity in the θ direction. Using the finite sum and finite difference approximations, the estimated pressure and velocity at the origin were

$$\hat{P}_{ec} = \frac{P_{m2} + P_{m1}}{2} \tag{2-59}$$

$$\hat{V}_{e\theta} = \frac{P_{m2} - P_{m1}}{j\rho_0 ckd} \tag{2-60}$$

where *d* was a constant equal to the effective microphone separation distance. Table 2. compares the different effective *kd* values for each probe.

To determine the bias errors, approximations for the acoustical quantities using the finite sum and difference equations were compared to the exact values at the origin. Calculating the bias errors in decibels provided an appropriate scale for graphing. The exact acoustic values at the origin are given by

$$\hat{P}_c = \left(1 + \hat{R}\right) \tag{2-61}$$

$$\hat{V}_\theta = \frac{\left(1 - \hat{R}\right)\cos\theta}{\rho_0 c} \tag{2-62}$$

$$e_\theta = \frac{1}{12\rho_0 c^2}\left(\left|1 + \hat{R}\right|^2 + 3\left|1 - \hat{R}\right|^2 \cos^2\theta\right) \tag{2-63}$$

$$I_\theta = \frac{\left(1 - \left|\hat{R}\right|^2\right)\cos\theta}{2\rho_0 c}. \tag{2-64}$$

### 2.8.1 Excess Pressure

To determine the bias errors of the spherical sensors, the pressure on the surface of the sphere, due to scattering, was determined. The geometry was shown by Figure 2.2. Two plane waves, $P_+$ and $P_-$, form a standing wave with reflection coefficient $\hat{R}$. The rigid sphere was centered at the origin. The complex pressure on the surface of the sphere in a standing wave field was

$$P_s\left(ka,\theta,\hat{R},\hat{P}_+\right)=\frac{\hat{P}_+}{j(ka)^2}\sum_{n=0}^{\infty}\frac{j^n(2n+1)\left(\hat{R}+(-1)^n\right)P_n(\cos(\theta))}{h_n^{(2)'}(ka)} \tag{2-65}$$

where *ka* was the acoustic wave number times the radius of the sphere, $\theta$ was the angle made by the microphone and the z axis, $\hat{R}$ was the complex reflection coefficient, $P_+$ was the plane wave amplitude, $P_n$ was a Legendre function of order *n*, and $h_n^{(2)'}$ was the derivative of the spherical Hankel function of the second kind of order *n*.[5] Parkins indicated that only thirteen terms were necessary to compute the sum. The pressure on this same surface, with the sphere removed, was given by

$$P_f\left(ka,\theta,\hat{R},P_+\right)=P_+\left(e^{-jka\cos(\theta)}+\hat{\mathrm{R}}e^{jka\cos(\theta)}\right). \tag{2-66}$$

The complex excess pressure was defined by

$$P_{ex}\equiv\frac{P_s}{P_f} \tag{2-67}$$

and is plotted as a function of incident angle, θ, in the results section, Figure 4.13.

### 2.8.2 Bias errors for Two-Point Sensors

For the two-point sensor, the bias errors can be calculated using equations 2-48 through 2-51 and 2-55 through 2-58. The normalized complex pressures at the two

microphones of the six microphone two-point sensor can be represented using the following equations

$$\hat{P}_{m1} = \delta_m e^{j\delta_p}\left(e^{-jkb\cos\theta} + \hat{\mathrm{R}}e^{jkb\cos\theta}\right) \tag{2-68}$$

$$\hat{P}_{m2} = \left(e^{jkb\cos\theta} + \hat{\mathrm{R}}e^{-jkb\cos\theta}\right) \tag{2-69}$$

For the tetrahedron two-point sensor, the normalized complex pressures at the microphones can be found using the following equations

$$\hat{P}_{m1} = \delta_m e^{j\delta_p}\left(e^{-jkb\cos\theta+\pi/6} + \hat{\mathrm{R}}e^{jkb\cos\theta+\pi/6}\right) \tag{2-70}$$

$$\hat{P}_{m2} = \left(e^{jkb\cos\theta-\pi/6} + \hat{\mathrm{R}}e^{-jkb\cos\theta-\pi/6}\right). \tag{2-71}$$

The factor of $\pi/6$ was necessary to rotate the tetrahedron so that the line between the two microphones was parallel with the incidence angle $\theta$.

For the orthogonal two-point sensor, the normalized complex pressures at the microphones can be found using the following equations

$$\hat{P}_{m1} = \delta_m e^{j\delta_p}\left(e^{-jkb\cos\theta+.9553} + \hat{\mathrm{R}}e^{jkb\cos\theta+.9553}\right) \tag{2-72}$$

$$\hat{P}_{m2} = \left(e^{jkb\cos\theta-.9553} + \hat{\mathrm{R}}e^{-jkb\cos\theta-.9553}\right). \tag{2-73}$$

Again, the factor of 0.9553 was necessary to rotate the probe so that the line between the two microphones was parallel with the incidence angle $\theta$.

### 2.8.3 Bias Errors for Two Point Spherical Sensors

The bias errors for the spherical sensors are more difficult to calculate, because of the infinite series. Parkins indicated that only thirteen terms of the series were necessary. The normalized complex pressures at the two microphones of the six microphone two-point spherical sensor are

$$\hat{P}_{m1} = \delta_m e^{j\delta_p} \frac{P_s\left(kd, \theta, \hat{R}, \hat{P}_+\right)}{\hat{P}_+} \tag{2-74}$$

$$\hat{P}_{m2} = \frac{P_s\left(kd, \theta + \pi, \hat{R}, \hat{P}_+\right)}{\hat{P}_+}, \tag{2-75}$$

where *kd* is given by Table 2.. The normalized complex pressures at the two microphones for the tetrahedron probe are

$$\hat{P}_{m1} = \delta_m e^{j\delta_p} \frac{P_s\left(kd, \theta + \pi/6, \hat{R}, \hat{P}_+\right)}{\hat{P}_+} \tag{2-76}$$

$$\hat{P}_{m2} = \frac{P_s\left(kd, \theta + 5\pi/6, \hat{R}, \hat{P}_+\right)}{\hat{P}_+}, \tag{2-77}$$

where *kd* is given by Table 2..The normalized complex pressures at the two microphones for the orthogonal probe are

$$\hat{P}_{m1} = \delta_m e^{j\delta_p} \frac{P_s\left(kd, \theta + .9553, \hat{R}, \hat{P}_+\right)}{\hat{P}_+} \tag{2-78}$$

$$\hat{P}_{m2} = \frac{P_s\left(kd, \theta + 2.1863, \hat{R}, \hat{P}_+\right)}{\hat{P}_+}, \tag{2-79}$$

where *kd* is given by Table 2.. The results for the bias errors due to magnitude and phase mismatch can be found in the results section.

# CHAPTER 3

# EXPERIMENTAL METHODS

## 3.1 Probe Description

The energy density probes were composed of several major parts: an aluminum sphere, a steel tube, and a metal body. The sphere was made from machined aluminum. The steel tube connected the sphere to the metal body and provided a way to connect the microphones to their associated electronics located inside the metal body. The metal body was a piece of extruded aluminum. The ends were fitted with aluminum end pieces and were attached to the extruded aluminum by four machine screws, two screws for each end piece. Within the metal body were several Larson Davis Digital Sensor Interface Transmitters, or DSITs, based on the DSS Digital Sensing System. The DSITs used in the probes were all model DSIT-I2A. A 10 lead ribbon cable connected the metal body and the A/D converters to the DSS-R receiver board. The DSS-R resided within the DSS chassis. The DSS chassis was connected to a computer via an Ethernet cable.

**Figure 3.1: Large and Small Tetrahedron Probe, 2 DSITs, and Aluminum Case**

The energy density probes all used Panasonic Electret microphones, model number WM-61A, to convert acoustic pressure to a voltage. All microphones were embedded in the surface of the sphere and are secured in place by an epoxy. Each microphone had a black ground wire and a colored signal wire. The colored signal wires corresponded to the colored dots on the surface of the sphere, adjacent to each microphone. The microphone wires were fed through the steel tube and connected to the DSITs. See Figure 3.1. Each DSIT was capable of measuring two channels of data simultaneously. It should be noted here that depending on the number of microphones in the probe, the metal body housed either two or three DSITs to accommodate either four or six channels of data acquisition.

**Figure 3.2: Large Six Microphone Probe, Large Tetrahedron Probe, Large Orthogonal Probe, Small Six Microphone Probe, Small Tetrahedron Probe, Small Orthogonal Probe**

In total, there were three probe designs-- the six-microphone probe, the tetrahedron-microphone probe, and the orthogonal-microphone probe. The tetrahedron

and orthogonal probes both made use of only four microphones as compared to the first design, the six microphone probe. Each probe had a small radius version and a large radius version. The large radius probes all had a 0.0254 m radius. The smaller four microphone probes, the tetrahedron and orthogonal probes, had a radius of 0.00825 m. The small six microphone probe had a radius of 0.0106 meters. See Figure 3.2.

The small six microphone probe had a larger radius than the other four microphone probes because the six microphones embedded on the surface of the aluminum sphere occupied more volume within the sphere than the four microphone probes. In other words, the aluminum sphere of the six microphone probe was larger to accommodate the additional two extra microphones. The following sections will describe each specific probe design in more detail.

## 3.2 The Six Microphone Probe

As stated above, the six microphone probe came in two sizes. The smaller sized probe had a radius of 0.0106 m. The larger sized probe had a radius of 0.0254 m. The six-microphone probe had three orthogonal pairs of microphones. Each pair of microphones was parallel to one of the three Cartesian coordinate axes. The coordinates of each microphone are found in Table 2.1. The six microphone probe was modeled after work done by others and represents the current state of the art when measuring velocity in three dimensions.[8] However, no one currently uses a sphere in conjunction with the six microphone configuration. The separation distance between the microphones on the six microphone probe was $2a$, where $a$ was the radius of the probe.

$$d = 2a \tag{3-1}$$

## 3.3 The Tetrahedron Probe

The tetrahedron probe also came in two sizes. The smaller probe radius was 0.00825 meters. The larger probe radius was 0.0254 meters. The tetrahedron probe had microphones located at the vertices of a regular tetrahedron. The coordinates of the four microphones were described in Table 2.. The coordinates of three microphones were in a plane below the center of the sphere, and the other microphone was positioned directly above the center of the sphere. The separation distance between the microphones was

$$d = \frac{2\sqrt{6}a}{3} \tag{3-2}$$

where a was the radius of the probe. It should be noted that the separation distance on the tetrahedron probe was the same as the long separation distance on the orthogonal probe, which will be described below.

## 3.4 The Orthogonal Probe

The orthogonal probe also came in two sizes. The smaller probe radius was 0.00825 m. The larger probe radius was 0.0254 m. The orthogonal probe was made up of four microphones. To describe the geometry of the orthogonal probe, it was helpful to first start with a Cartesian coordinate system with an origin and three orthogonal unit vectors. One microphone was located at the origin and the other three microphones were located at the tips of the Cartesian unit vectors. So, one microphone had coordinates [0,0,0] at the local origin. The other microphones had normalized positions [1,0,0], [0,1,0] and [0,0,1]. The center of the sphere would then be located at a normalized position [½, ½, ½], and all four microphones would lie on the surface of the sphere. Alternately, Table 2. lists the coordinates of the four microphones, given that the center of the probe was located at the origin.

There were two separation distances between the microphones of the orthogonal probe. The short separation distance, or the distance from the local origin along each of the unit vectors, was

$$d = \frac{2a}{\sqrt{3}} \tag{3-3}$$

where $a$ was the radius of the probe and $d$ was the separation distance. The long separation distance, or the distance between the tips of each unit vector, was

$$d = \frac{2\sqrt{6}a}{3} \tag{3-4}$$

It should be noted that the long separation distance was the same separation distance as found on the tetrahedron probe described above. For additional clarification, see Figure 2.4.

### 3.5 Additional Hardware

A Mackie Studio monitor, model number HR824 serial number (21)EL13673, served as an acoustic source. The source was positioned in the corner of the large anechoic chamber located within the Eyring Science Center at BYU. The Input Sensitivity knob on the back of the Mackie was set to maximum. The low frequency cutoff switch was set to 37 Hz (normal). The High Frequency switch was set to 0 (normal). The power mode was set to the On position. The Mackie was placed atop a speaker platform, such that the acoustic center of the source was approximately 2 m above the wire floor of the chamber. The Mackie was connected to a 120 volt A/C outlet via a black power cable.

The Mackie was connected to the source terminal of a Hewlett Packard HP35670A 2 channel FFT analyzer via a BNC cable that passed through the chamber

cable tunnel into the control room. On the Mackie side of the BNC cable, a BNC to RCA adapter was used to connect the BNC cable to the RCA input on the Mackie. The other end of the BNC cable was connected to the source terminal of an HP FFT analyzer that was located in the control room. The HP analyzer served as a tone generator. The output of the HP analyzer was initially set to 1 volt peak.

A Larson Davis Digital Sensing System (DSS) accomplished data acquisition. The DSS used was model number DSS-CH, serial number 0127. The DSS chassis was equipped with master board model DSS-ME and receiver board model DSS-R. The DSS was connected to a laptop computer via Ethernet cables and a Link-Max network switch model number LM-208. The laptop computer had an Intel Pentium 4 processor, running Microsoft XP Home Edition. Larson Davis WaveFront software versions 1.0.0.0 through 1.0.4.2 were used to interface with the DSS hardware and export the data to a comma delimited text file.

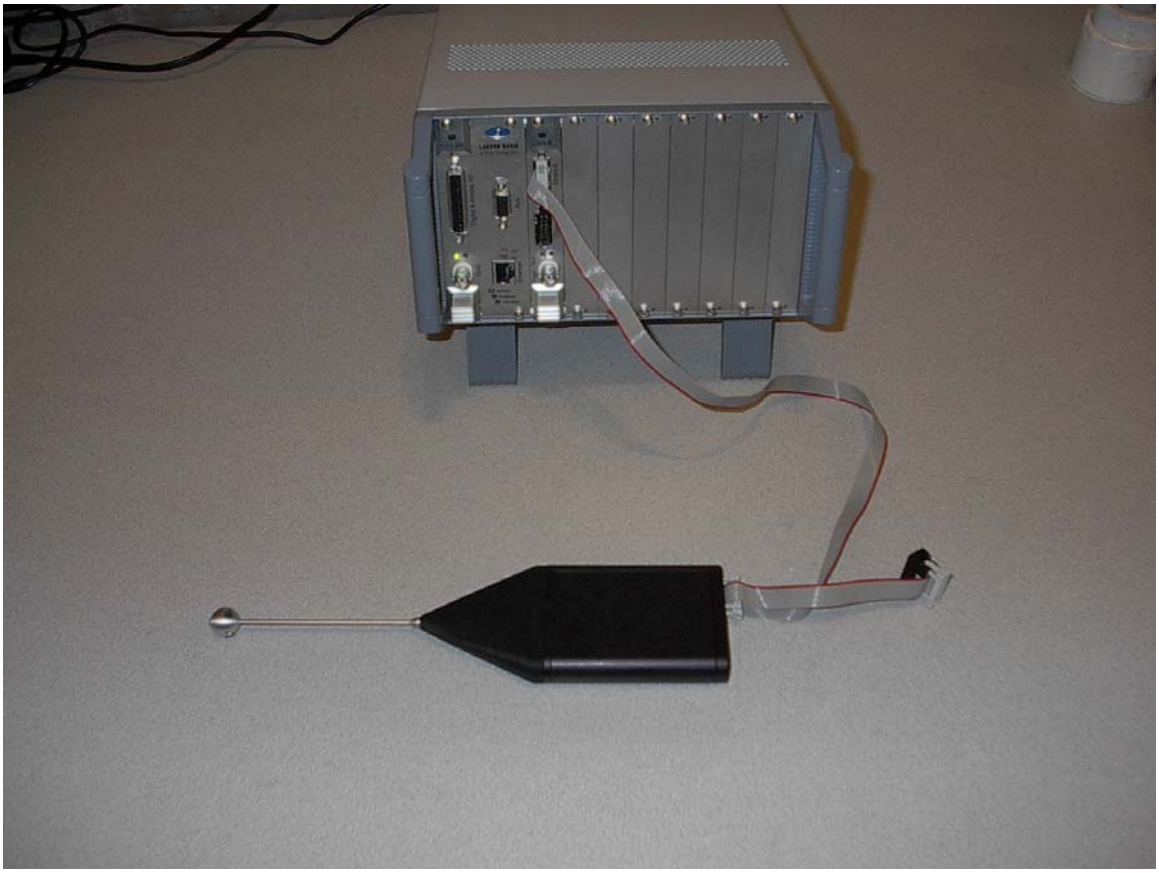

**Figure 3.3: DSS Chassis connected to Small Tetrahedron Probe. (note black terminating resistor )**

The probes were connected to the DSS using a grey 10 lead ribbon cable. The cable was passed through the chamber cable tunnel from the control room into the anechoic chamber. Nylon cable ties were used to secure the ribbon cable to a metal framework within the chamber. A single lead of the ribbon cable was colored red to identify the first wire when crimping connectors to the ribbon cable. The ribbon cable was fitted with a connector and terminating resistor at the chamber end of the cable. See Figure 3.3. The connector was installed as specified in the DSS documentation and was necessary for the DSS-R receiver board to function properly. The other end of the ribbon cable was connected to the DSS-R via a connector, which was also crimped to the ribbon cable. A third connector was crimped to the ribbon cable approximately 30 cm from the

terminating resistor, in between the terminating resistor and the control room connector. This last connector served as the connection point for the probes, and was inserted into the base of the metal body of the probe under test. The probes were mounted individually to a microphone stand, at nominally the same elevation as the acoustic center of the Mackie speaker. The probes were oriented in two ways, depending on the test. The first orientation was such that the microphone connected to DSIT channel 22 was pointed directly at the acoustic source. The second orientation was such that one axis of the probe was lined up with the radial axis of the source.

### 3.6 Sensor Calibration

A TMS ½ inch ICP microphone, model number 426C01, serial number 2330, was used as a reference microphone. It was connected via a BNC cable to input channel 1 of an HP35670A 2 channel FFT analyzer. The analyzer was configured to act as an ICP power supply for the TMS ½ inch microphone. The TMS microphone was inserted into a Larson Davis Precision Acoustic Calibrator, model number CAL200, serial number 0244. The Calibrator was activated and provided a constant source of 94 dB SPL re 20 μPa at 1 kHz. The HP analyzer used the built-in calibration routine, with the calibrator as a known source, to determine the sensitivity of the TMS microphone.

The large radius energy density probes were calibrated relative to the TMS ½ inch microphone above. A relative calibration was accomplished by means of a calibration device fabricated by Kent Gee. The calibration device was fabricated using a PVC cup and a 1 ¾ inch diameter loudspeaker. See Figure 3.4. The loudspeaker was connected to the source terminal of the HP35670A analyzer, which provided a source tone of 250 Hz with an initial setting of 1 volt peak output. The source tone of 250 Hz was chosen so that

the sound field inside the PVC cup would be uniform. The calibrated TMS ½ inch microphone was inserted into the side of the PVC cup and connected to input channel 1 of the HP35670A 2 channel FFT analyzer. The analyzer was configured to measure dB SPL, as outlined above. The large radius probe was inserted into the mouth of the calibration device, with the microphone to be calibrated facing directly towards the loudspeaker. Putty was used to create an airtight seal between the mouth of the calibration device and the probe. The probe was connected to the data acquisition system, as described above. WaveFront was used to look at the frequency response of the microphone to be calibrated in real time. The level of the voltage supplied to the loudspeaker in the calibration device was increased until harmonic distortion of the probe mounted microphone at 500 Hz (the first harmonic of the source tone) rose just above the noise floor, as displayed by the WaveFront software. The SPL, as indicated by the calibrated TMS ½ inch microphone, was recorded and inputted into the WaveFront Units/Calibration submenu. WaveFront then calculated and recorded the sensitivity of the probe mounted microphone in Pa/volt. The process was repeated for each microphone within the probe, and for all three large probes.

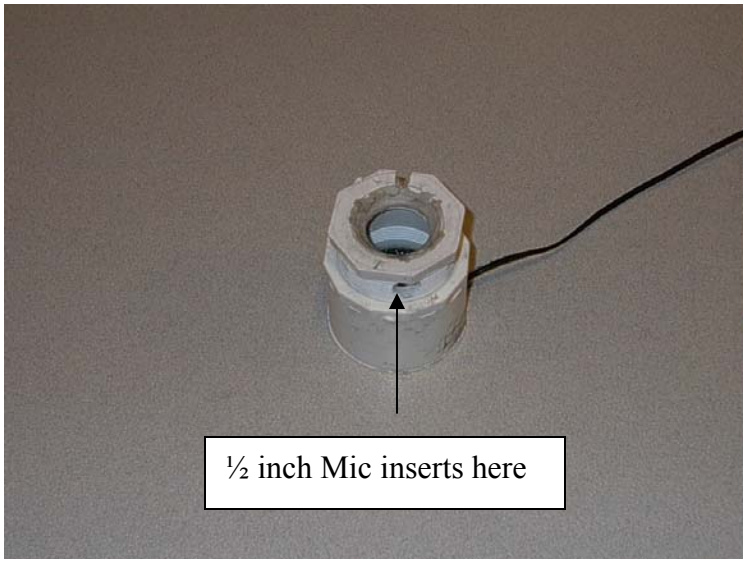


**Figure 3.4: PVC Large Probe Calibrator**

The small radius energy density probe was calibrated using a Larson Davis Precision Acoustic Calibrator, model number CAL200, serial number 0244. It should be noted that the Calibrator was not originally designed to calibrate microphones embedded on the surface of a sphere. Consequently, the exact sound pressure level exposed to the sphere mounted microphones was unknown. However, when used with a standard ½ inch microphone, the calibrator produced a steady tone of 1 kHz at 94 dB SPL re 20 μPa.

The Probe was connected to the data acquisition system as described above. The calibrator was placed over the microphone to be calibrated and the calibrator was activated. A rubber band was used to ensure constant pressure between the surface of the probe and the calibrator. The WaveFront software had a built in calibration system found under the Units/Calibration submenu. The calibration SPL level of 94 dB was input, along with the 20 μPa reference, and the software calculated the microphone sensitivity in Pa/Volt. The software stored the sensitivity value for future use. (The author recognizes that at 94 dB SPL the electret microphones within the energy density probes begin to exhibit harmonic distortion. However, with the resources available, it was felt

that this still provided the best immediate solution. Future work should include the development of an improved calibration apparatus.)

## 3.7 Measurements

Experimental verification was done at Brigham Young University in the anechoic chamber, located in the Eyring Science Center. The hardware was assembled and set up as described in Section 3.5. Measurements were made over the course of several days.

Data was taken at four positions in the anechoic chamber on the diagonal line from one corner of the chamber to the opposite corner of the chamber. The source and probes were positioned on the diagonal line to facilitate the greatest separation distance between the source and probes.

### 3.7.1 Measurement Locations

The first position where data was taken was referred to as position A. Position A was 1.5 m from the source. The probe was oriented so that the microphone connected to DSIT channel 22 was pointing directly at the source. The probe was attached to a microphone stand and elevated to nominally the same height as the source. Positions B and C were identical to Position A except Position B was 3 m from the source and Position C was 6 m from the source. Again, all three positions were located on the diagonal line between two opposite corners of the anechoic chamber. The last position, Position R, was identical to position C, except for the orientation of the probe. The probe orientation at Position R was randomly chosen such that no microphone was pointed directly at the source.

At each measurement position, a ½ inch TMS ICP microphone was also present. The TMS ICP microphone was positioned at the same elevation and radial distance from

the source as the probe; however the microphone was separated from the probe by nominally a foot, to prevent mutual interference.

### 3.7.2 Exporting Data

The DSS system had the capability to export the pressure time waveform data of each microphone after a measurement was made. The time waveform data for each microphone of the probe under test was exported to a single text file. For each microphone, the text file had a short header followed by a long column of numbers. The column of numbers represented the pressure values measured by the particular microphone during the time interval that data was taken. The pressure values were represented with 7 digits of precision, followed by a three digit exponent. The header described certain aspects of the data run and the specific channel, such as probe type, sampling frequency, calibration data, and microphone sensitivity. The text file had either four or six header- data combinations, depending on the probe type.

A filename scheme was developed to reduce confusion when accessing the time waveform data stored in the text files mentioned above. Each filename had five characters followed by the .txt extension. The first character corresponded to the measurement position of the probe when the data was taken. The first character was either A, B, C, or R. The second character designated the probe size, either L for the large probe or S for the small probe. The third character designated the probe type and was made up of the first letter of the probe under test. S identified the six microphone probe as the probe under test. T identified the tetrahedron probe as the probe under test. O identified the orthogonal probe as the probe under test. The fourth and fifth digits represented the measurement number for averaging purposes.

The data was organized into folders corresponding to probe size and type. For example, the directory titled ‘Large_Six’ had data taken by the large six microphone probe at all four measurement locations. Ten measurements were made at each location to assist in averaging. Consequently, the ‘Large_Six’ folder had a total of 40 text files. A folder was made for each of the six probes. In total, two hundred and forty measurement runs were made.

# CHAPTER 4

## NUMERICAL RESULTS AND DISCUSSION

This chapter presents and discusses the numerical results. In total, the work presented includes four models. The first two were developed and implemented by the author. The last two were developed by Parkins, and expanded upon by the author. The four models assume a point sensing element which has no area. The four models are also similar in that for one dimension, all four models have the same geometric configuration. The author's models are both three dimensional representations of the physical probes. The Parkins' models are only one dimensional. This section will show the numerical results, show how all the models are similar and different, and explain the significance of the results.

The Parkins' work was originally implemented to explore the effects of microphone magnitude and phase mismatch in conjunction with scattering effects caused by a rigid sphere. However, upon implementation, it became clear that Parkins' use of three different complex reflection coefficients provided additional insight into the effects of various enclosures on the performance of the energy density probes. Consequently, the results caused by the different reflection coefficients will be included.

The rest of the chapter is presented as follows. The next subsection will present the results of the multi-point model. The following subsection will discuss the results of the multi-point spherical model. Continuing on, the results obtained by the modified Parkins' model will be discussed in the case of perfect microphones, for the six microphone model, the tetrahedron model, and the orthogonal model. The next subsection will compare the modified Parkins' two-point and spherical models in the case

of perfectly matched microphones. That will be followed by an investigation of the excess pressure on the surface of the sphere in an enclosure with various reflection coefficients. The last subsection will discuss the effects of the different reflection coefficients in conjunction with microphone magnitude and phase mismatching.

## 4.1 Multi Point Sensor Model

### 4.1.1 Multi Point Sensor Energy Density and Pressure Error

Figure 4.1 shows the pressure and energy density bias errors, equations 2-48 and 2-50, for the three multi-point sensors after averaging the results from 100 random position vectors. The orthogonal probe has the highest pressure bias error. The tetrahedron and six microphone probes have almost identical pressure bias errors. The energy density bias errors for the orthogonal probe are the closest to 0 dB, followed by the tetrahedron probe and then the six microphone probe. All bias errors approach 0 as *ka* goes to 0. The energy density errors are monotonically decreasing, indicating greater underestimation, while the pressure errors are monotonically increasing, indicating greater overestimation.

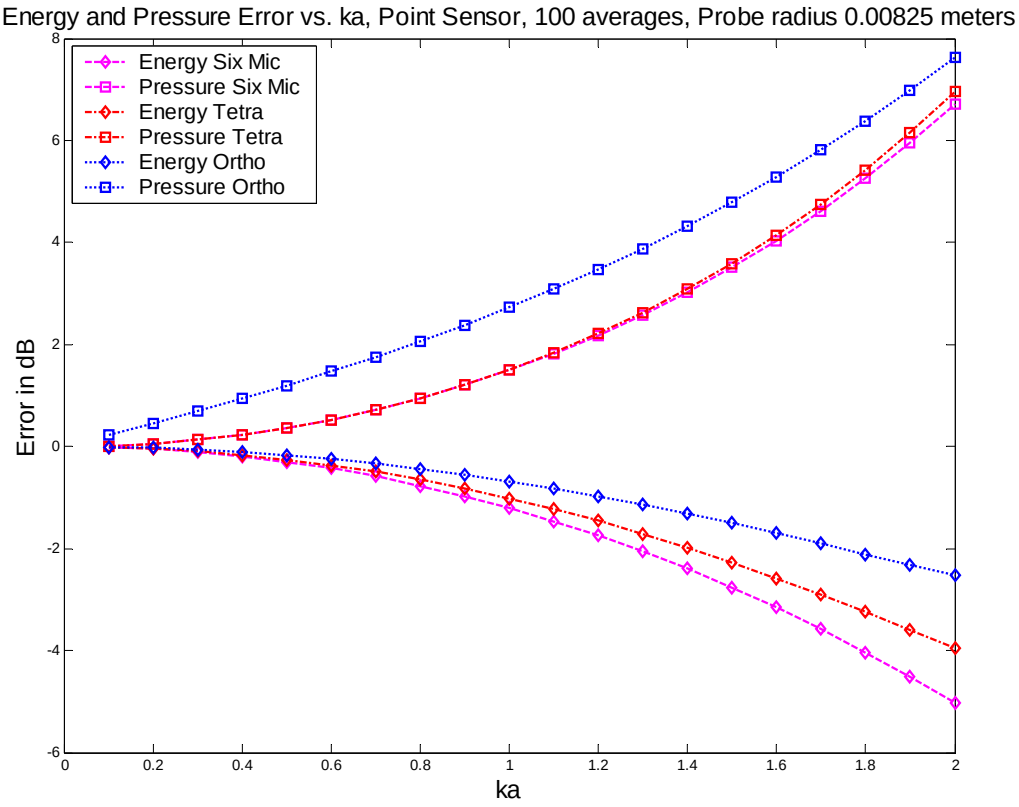


**Figure 4.1: Energy density and pressure error for multi point sensors, 100 averages.**

### 4.1.2 Multi Point Sensor Velocity Error

The velocity magnitude errors, shown in Figure 4.2, follow similar trends for each probe type. The velocity magnitude error is determined by comparing the square of the estimated magnitude with the square of the actual magnitude. The six microphone probe has the highest error. This is most likely due to the larger separation distance between the microphones. The orthogonal probe, on the other hand, has the lowest error. This is due to the short separation distance between the microphones found in the orthogonal probe. The tetrahedron probe error falls between the six microphone probe and the orthogonal probe. All three probes demonstrate low errors as *ka* approaches zero. This behavior is to be expected. The negative bias errors suggest that the velocity magnitude is

underestimated. The velocity error and pressure error influence the overall energy density error. Given the trends of the energy density error, velocity error, and pressure error, it appears that the energy density error follows the patterns of velocity error more than the pressure error.

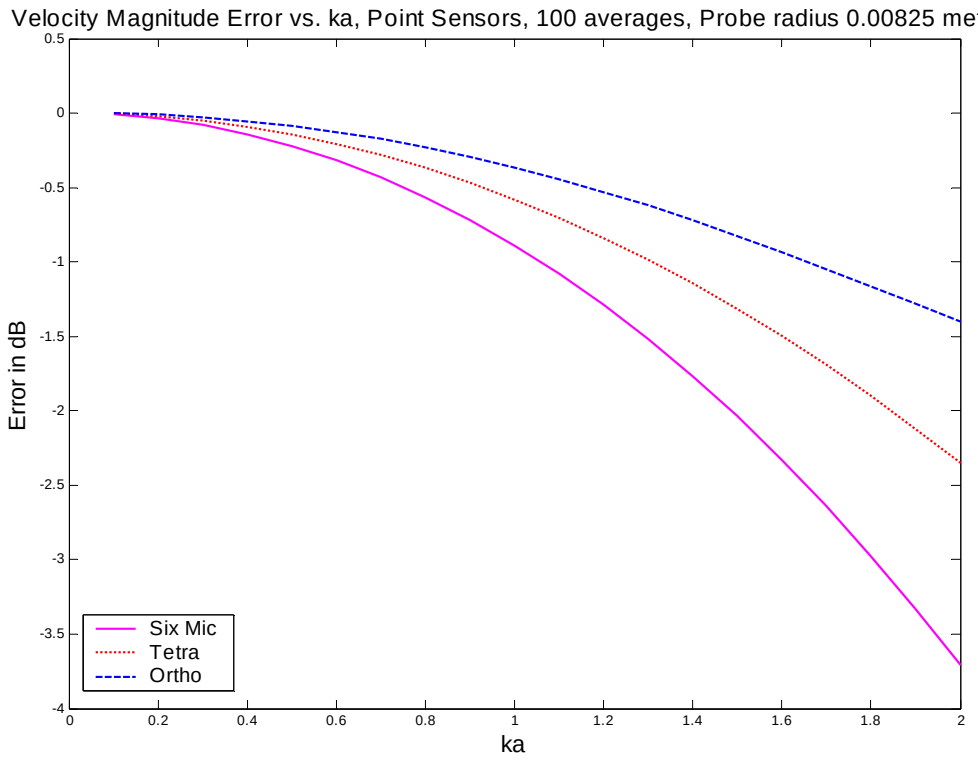


**Figure 4.2: Multi Point Sensor Velocity Magnitude Error**

### 4.1.3 Multi Point Sensor Intensity Error

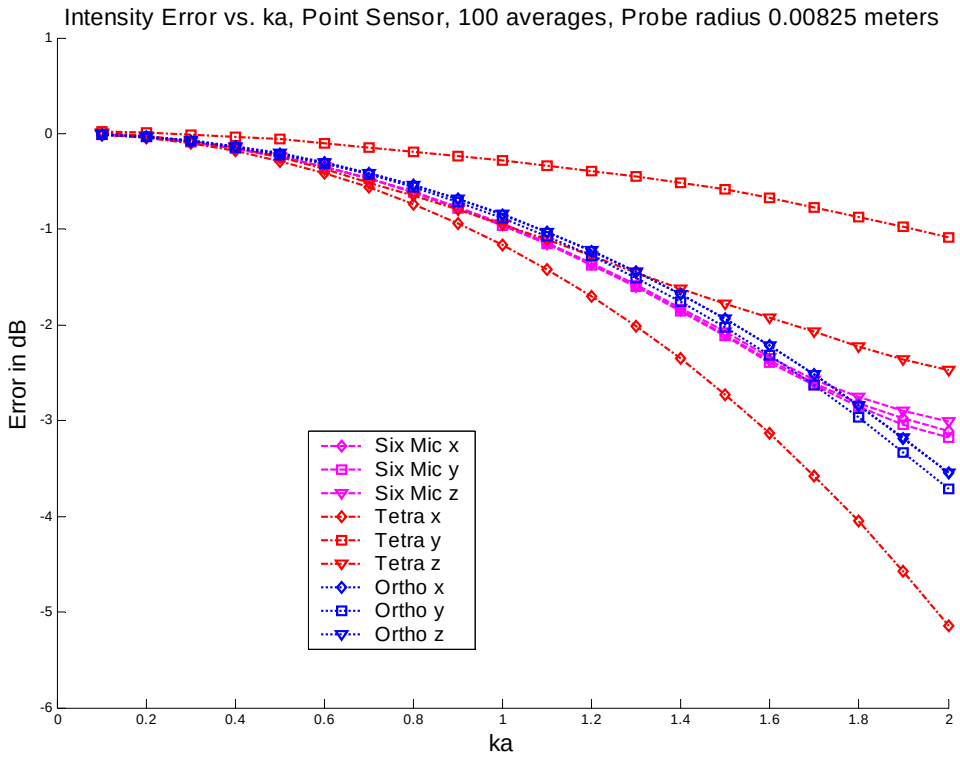


**Figure 4.3: Intensity magnitude errors for point sensors, 100 averages.**

The x, y, and z component intensity magnitude errors, shown in Figure 4.3, follow a similar pattern for all three probe designs. The errors are all monotonically decreasing, except for the six microphone probe, which experiences a point of inflection around a *ka* value of 1.6. The orthogonal probe has the most consistent error curve while the six microphone probe shows a change of behavior near *ka* of 1.6. The six microphone probe initially performs about the same as the orthogonal probe. The tetrahedron probe has the widest variety of errors. This is caused by the velocity estimate. When the velocity in the y or z direction is low, the tetrahedron probe has a tendency to overestimate that component of velocity. As a result, the intensity is not accurately predicted. However, up to a *ka* of 1, the three probes perform relatively well, and have errors less than 2 dB.

## 4.2 Multi Point Spherical Sensor Model

### 4.2.1 Multi Point Spherical Sensor Energy Density and Pressure Error

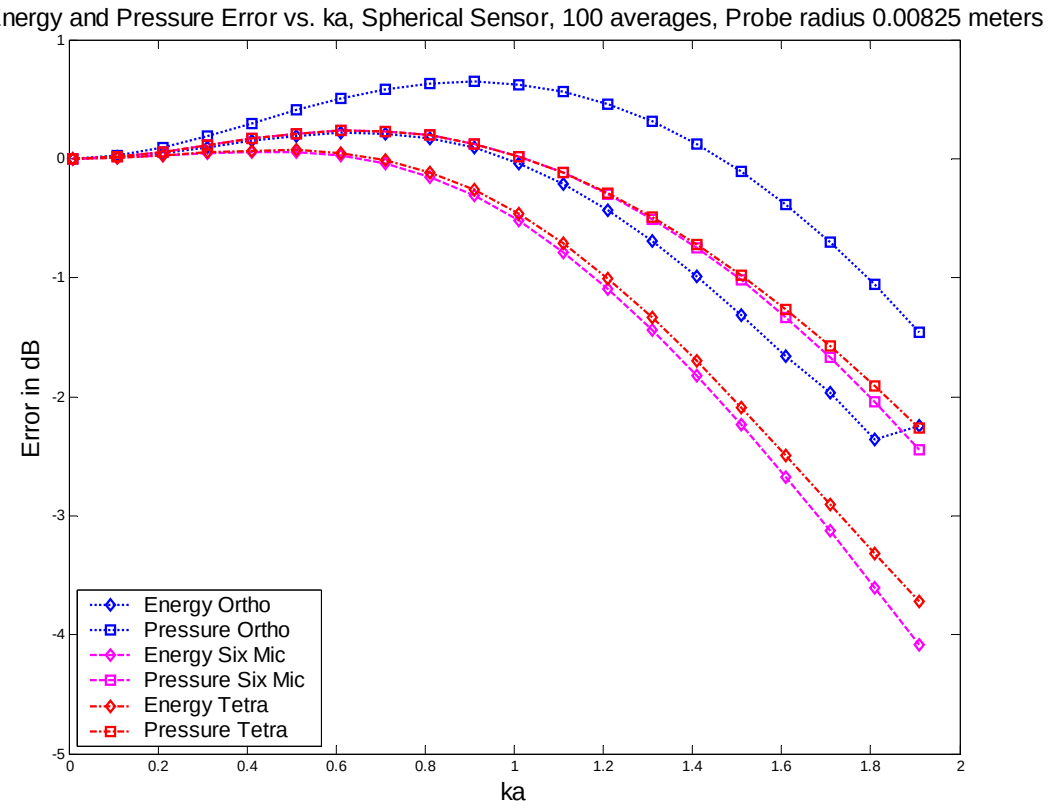


**Figure 4.4: Multi Point Spherical Sensor Energy Density and Pressure Error**

Figure 4.4 shows the results for measuring pressure and energy density using the multi-point spherical sensor. The multi-point spherical sensor model assumes perfect microphones. The model takes into account scattering effects associated with the hard spherical probe, but does not incorporate any room effects due to a reflection coefficient. The energy density and pressure errors for the multi-point spherical sensors are surprisingly low. The tetrahedron and six microphone probe have almost identical errors both for energy density and pressure. The orthogonal probe pressure error follows a shape similar to the other probes, but peaks at a *ka* value around 0.9, as opposed to the peak at

*ka* of 0.6 for the tetrahedron and six microphone probes. The orthogonal probe begins to show problems in the energy density errors when *ka* is equal to 1.8. Such a high *ka* value is the upper limit of the process used to estimate particle velocity for the orthogonal probe. The rise in the pressure error of the orthogonal probe is due to what Elko characterized as the effects of a sphere when measuring pressure, i.e. Elko showed that the pressure error is reduced through the presence of the sphere,[3] as can be seen from Fig. 4.1 and Fig. 4.2. The effect is enhanced by the fact that the microphones are physically closer to each other than in the other probes. However, all probes show the same trend, but the orthogonal probe is the most pronounced of the three.

### 4.2.2 Multi Point Spherical Sensor Velocity Error

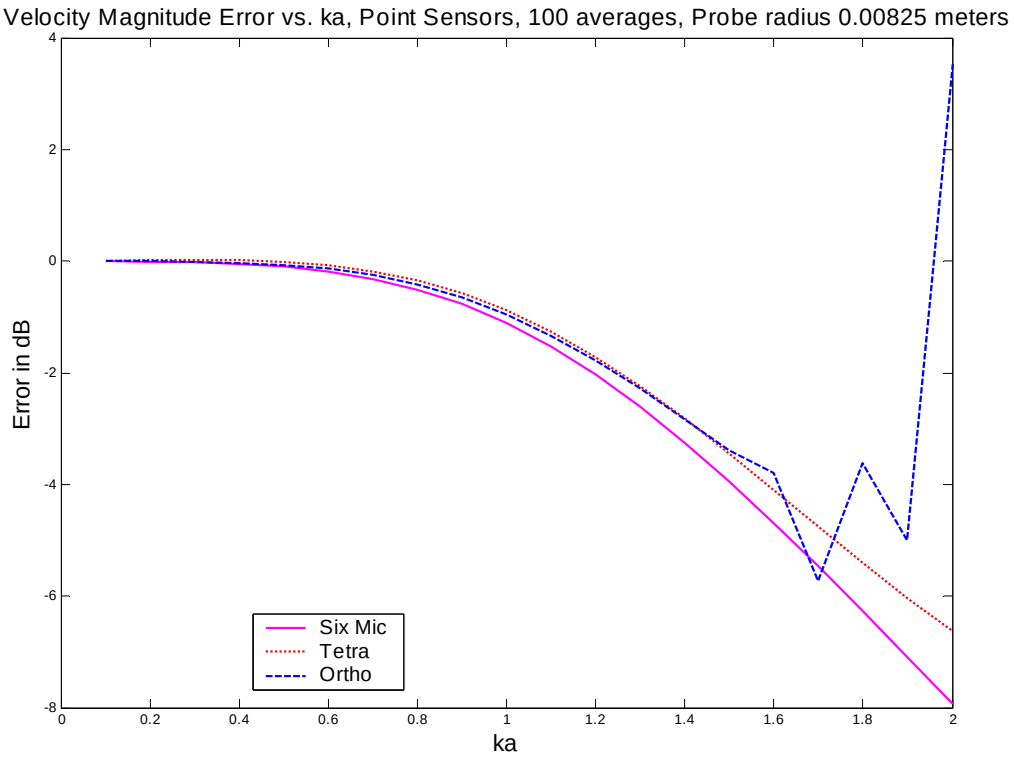


**Figure 4.5: Multi Point Spherical Sensor Velocity Error**

The velocity magnitude errors for the multi-point spherical sensor, shown in Fig. 4.5, all follow the same pattern. The orthogonal probe errors exhibit problems when *ka* is greater than 1.4. This breakdown occurs because the probe inaccurately estimates the angles associated with the projection of the unknown velocity. The figure does indicate that the tetrahedron and orthogonal probe outperform the six microphone probe, but not by much. All three probe designs approach zero error as *ka* goes to zero. The error of all three probes at *ka* equal to 1 is nearly 1 dB.

### 4.2.3 Multi Point Spherical Sensor Intensity Error

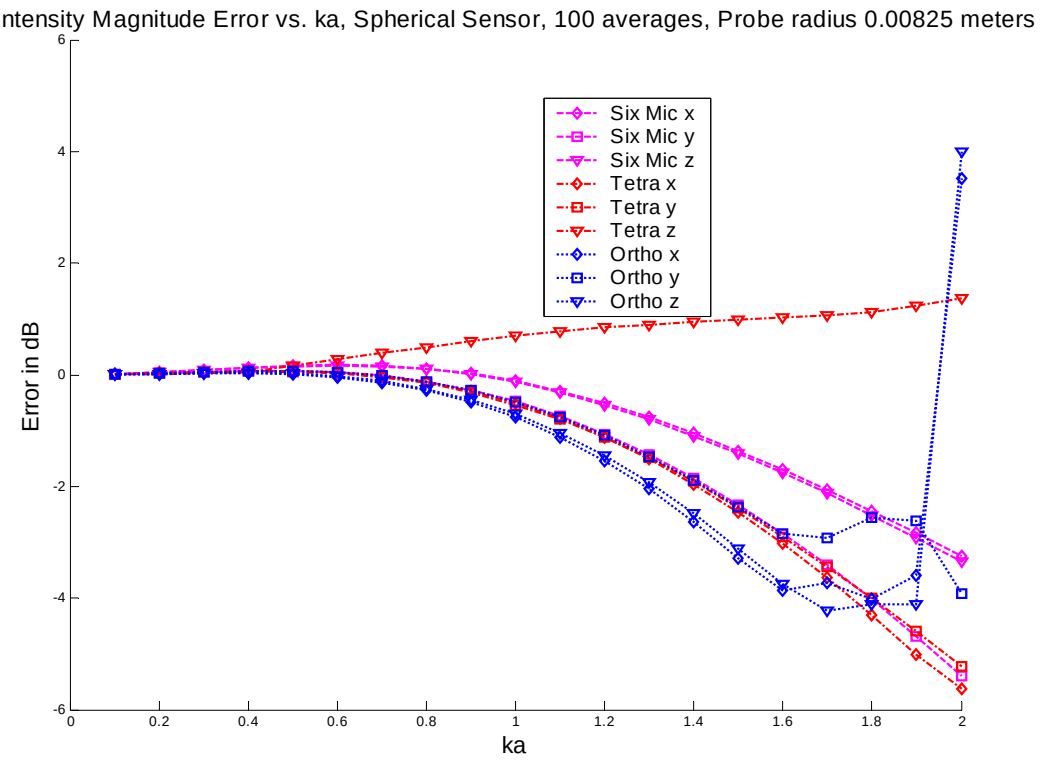


**Figure 4.6: Multi Point Spherical Sensor Intensity Error**

Figure 4.6 shows the intensity error for each component of intensity, given the multi-point spherical sensor. The six microphone probe tends to estimate the intensity better than the other two probes. The multi-point spherical sensor orthogonal probe demonstrates problems above *ka* of 1.5, in the same region where the velocity errors are problematic. These errors are caused by an inaccurate estimation of the angles involved with the orthogonal probe estimation model.

## 4.3 Two-Point vs. Multi-Point Sensor

A brief discussion of the similarities and differences between the two-point and multi-point models is advantageous. The two-point model is essentially a one dimensional representation of the multi-point model. The results shown for the multi-

point models in the previous subsections represent an average of 100 random locations or sensor orientations for the multi-point models. The following subsections all have the same orientation, where θ is set to 0. When comparing the four models, the two-point sensor is similar to the multi-point sensor and the two-point spherical sensor is similar to multi-point spherical sensor.

## 4.4 Two-point Sensor Results, Perfectly Matched Microphones

### 4.4.1 Six Microphone Two-point Sensor

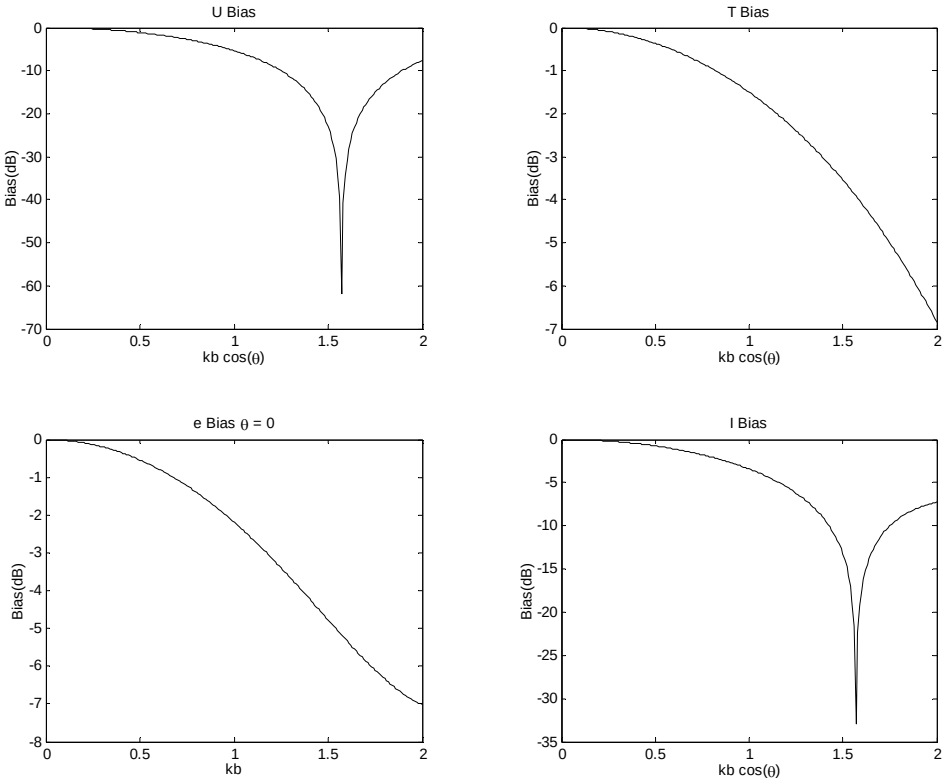


**Figure: 4.7 Bias errors of a two-point sensor with perfectly matched microphones for potential energy density (U), kinetic energy density (T), total energy density (e), and magnitude of intensity (I) measurements, Six Microphone probe.**

Figure 4.7 shows the results for perfectly matched microphones in the six microphone two-point sensor model. The results are displayed as a function of kbcos(θ),

the way that Parkins showed his results. However, theta was taken to be zero, because Parkins indicated that this yielded the highest errors. Additionally, Parkins used *a* instead of *b* to differentiate between his two-point sensor and his spherical sensor. The bias errors for potential energy density and intensity show a sharp dip just above a *kb* value of 1.5. This suggests that the sensor is not able to estimate the pressure accurately at that frequency.

### 4.4.2 Tetrahedron Two-Point Sensor

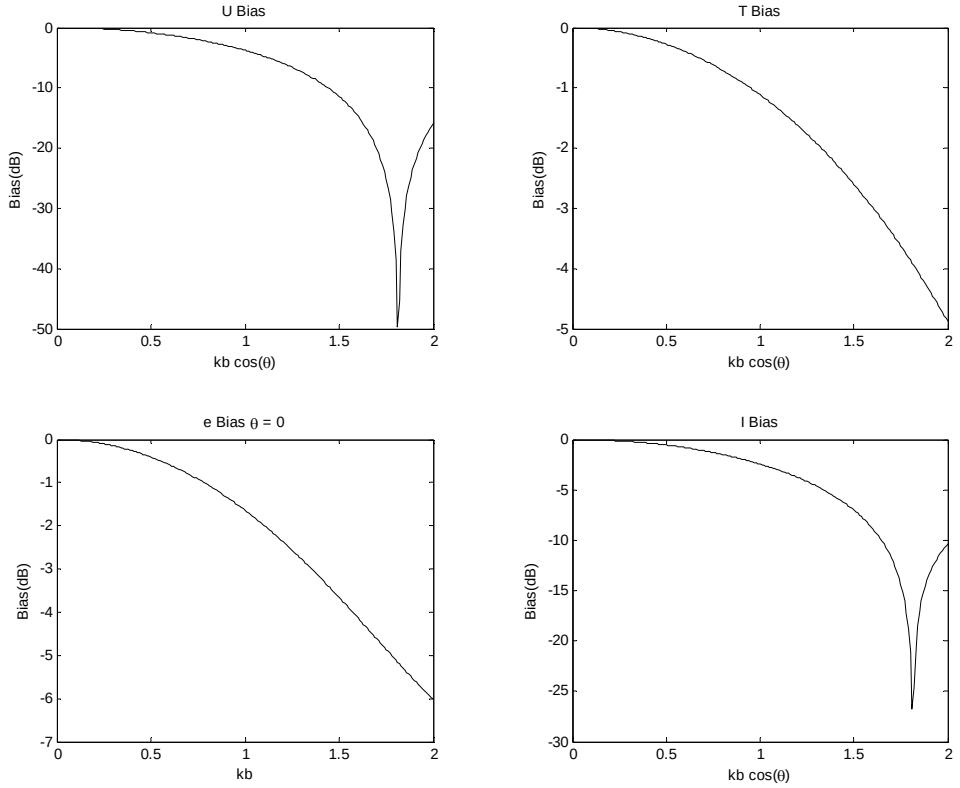


**Figure 4.8: Bias errors of a two-point sensor with perfectly matched microphones for potential energy density (U), kinetic energy density (T), total energy density (e), and magnitude of intensity (I) measurements, Tetrahedron probe**

Figure 4.8 shows the bias errors for the two-point tetrahedron sensor with perfectly matched microphones. The tetrahedron two-point sensor errors follow the six

microphone two-point sensor errors very closely. The main difference is the shift of the dips in potential energy density bias and intensity bias. This shift is caused by the shorter separation distance between the microphones. It should be noted that according to Pakins, the energy bias is not a function of θ, so it is plotted only as a function of kb, with θ=0.

### 4.4.3 Orthogonal Two-point Sensor

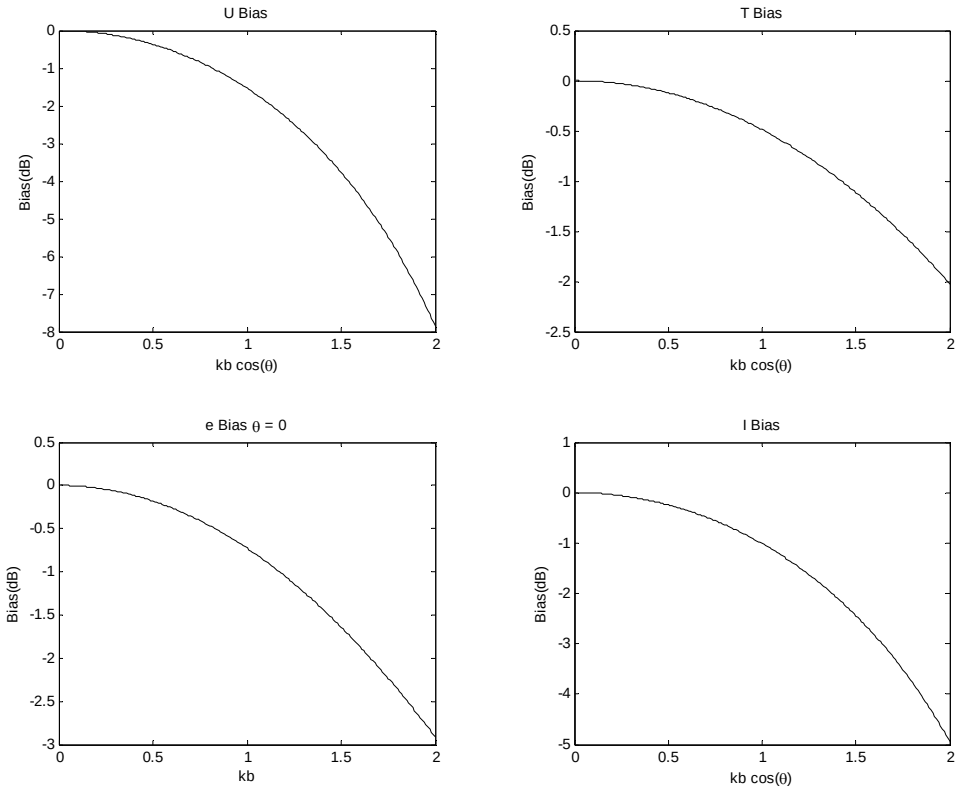


**Figure 4.9: Bias errors of a two-point sensor with perfectly matched microphones for potential energy density (U), kinetic energy density (T), total energy density (e), and magnitude of intensity (I) measurements, Orthogonal probe**

Figure 4.9 shows the errors for the orthogonal two-point sensor with perfectly matched microphones. The orthogonal two-point sensor exhibits the best errors out of the three probe types. The shorter separation distance of the orthogonal probe changes the scaling of *kb*, which yields a more accurate measurement, when comparing similar *kb*

values. The potential energy density bias drops off the fastest, but the energy density bias follows the trend of the kinetic energy density bias more than the potential energy density bias. This suggests that the energy density bias errors follow the kinetic energy density bias, which is based on velocity. This is consistent with results from the multi-point sensor model in Section 4.2.1.

## 4.5 Two-point Sensor vs. Two-point Spherical Sensor

When comparing the two-point sensor with the two-point spherical sensor, Parkins indicated that it is necessary to apply a correction factor. Table 2. shows the different values of *kd* when using the two-point sensor and the two-point spherical sensor. Parkins chose the letter *b* to correspond to the radius of his spherical sensor contrasted with the letter *a* to correspond to the radius of his two-point sensor. Both *a* and *b* represent the same value, which is the radius of the sphere. Consequently, when comparing the velocity estimate of the two models, $ka = \frac{2}{3}kb$.

### 4.5.1 Six Microphone Two-point vs. Two-point Spherical Sensor

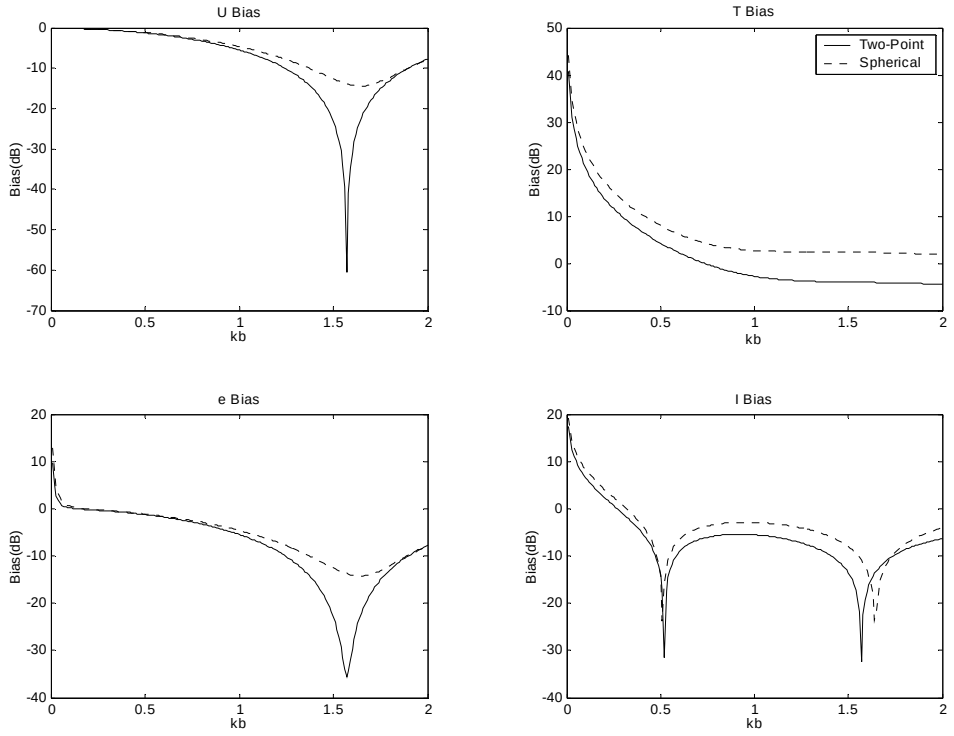


**Figure 4.10: Bias errors of a spherical sensor and two-point sensor for microphones having a 1º phase and .25 dB sensitivity mismatch. Sensors are in a standing wave field with reflection coefficient R= .97 and incident angle θ= 0. Potential energy density bias (U). Kinetic energy density density bias (T). Total energy density density bias (e). Magnitude of intensity bias (I). Six Micropone Probe**

Figure 4.10 shows the bias errors for a two-point sensor with mismatched microphones compared to a spherical sensor with imperfectly matched microphones. The potential energy density bias for the spherical sensor was not as pronounced as the two-point sensor. The energy density bias has a dip at roughly the same kb value as the potential energy density bias. The intensity bias has two dips, one at approximately kb = 0.5 and another at approximately kb= 1.6. The spherical sensor is not as sensitive to dips in the potential energy density bias as the two-point sensor. This smoothing effect is seen in the potential energy density bias, the overall energy density bias, and in the intensity bias.

### 4.5.2 Tetrahedron Two-point vs. Two-point Spherical Sensor

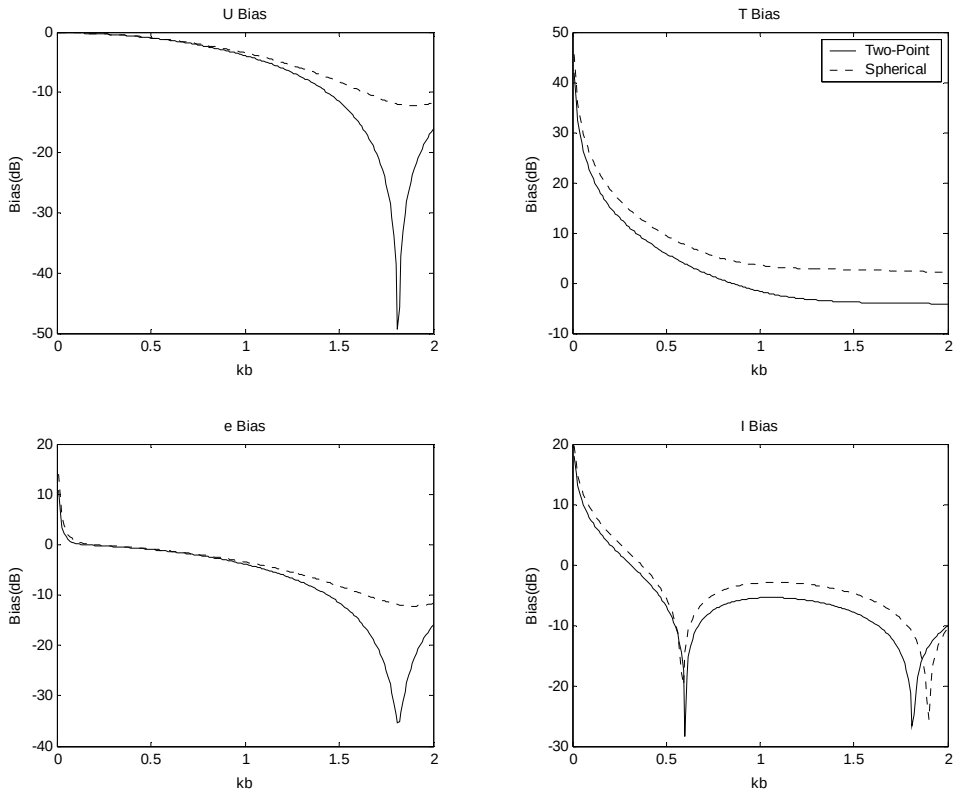


**Figure 4.11: Bias errors of a spherical sensor and two-point sensor for microphones having a 1º phase and .25 dB sensitivity mismatch. Sensors are in a standing wave field with reflection coefficient R= .97 and incident angle θ= 0. Potential energy density bias (U). Kinetic energy density density bias (T). Total energy density density bias (e). Magnitude of intensity bias (I). Tetrahedron Probe**

The comparison of the tetrahedron two-point and the two-point spherical models show the same kb stretch as seen in the other results. The energy density bias error has the same shape as the potential energy density error. However, the energy density error is not as pronounced as the potential energy density error. The intensity bias error has two dips. The second dip, at around *kb* of 1.8, is most likely caused by the error in estimating pressure. The locations of the dips are at nominally the same *kb* values as the minima in the potential energy density error. The first dips, near *kb* of 0.6, are most likely caused by a phase issue that cannot be seen in these magnitude plots. Because intensity is formed

by multiplying the pressure and velocity, it is very sensitive to problems in the phase. It is very possible that the pressure near *kb* of 0.6 goes through a phase change, but this was not investigated explicitly. The energy density estimate in the vicinity of *kb* equal to 0.6 is very accurate, but the shape of the error curve suggests that it may be a point of inflection.

### 4.5.3 Orthogonal Two-point vs. Two-point Spherical Sensor

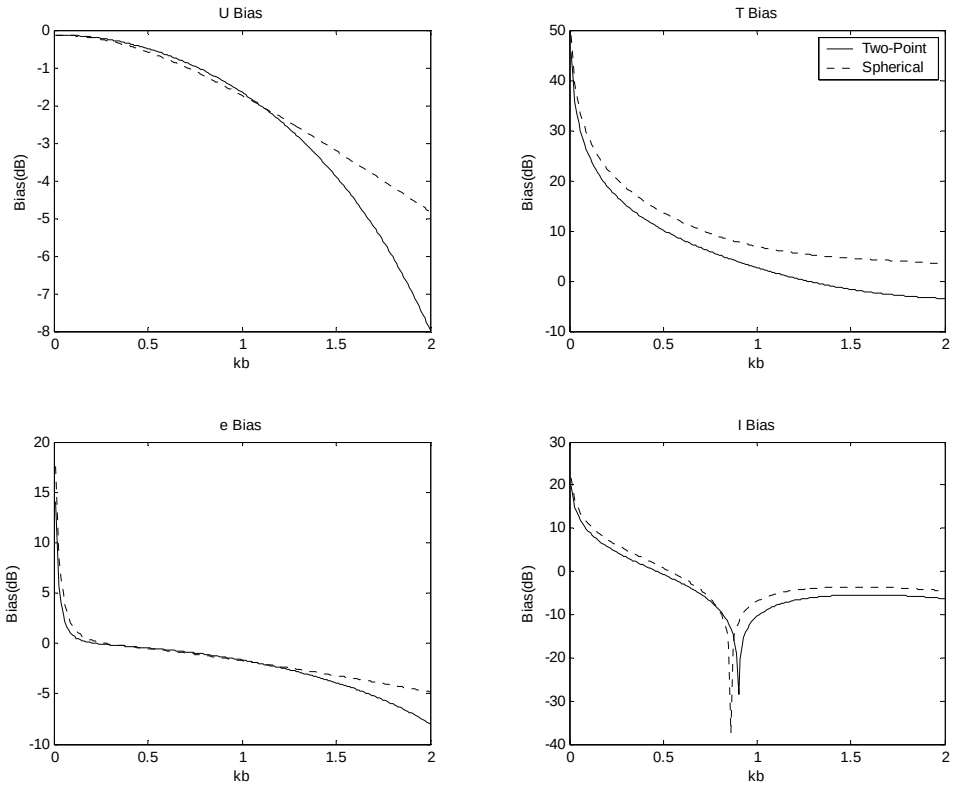


**Figure 4.12: Bias errors of a spherical sensor and two-point sensor for microphones having a 1º phase and .25 dB sensitivity mismatch. Sensors are in a standing wave field with reflection coefficient R= .97 and incident angle θ= 0. Potential energy density bias (U). Kinetic energy density density bias (T). Total energy density density bias (e). Magnitude of intensity bias (I). Orthogonal Probe**

The orthogonal probe comparison again shows what happens when the microphones are brought close together. The errors all show the same patterns, but the minima and distinguishing features are all shifted. The energy density bias errors are very

high when *kb* approaches zero. This is because of how the kinetic energy density is over estimated as *kb* approaches zero.

## 4.6 Two-point Spherical Sensor Excess Pressure

Figure 4.13 shows the excess pressure magnitude and excess phase on the surface of a hard sphere for various reflection coefficients and *ka* values. As *ka* increases, the scattering effects are more pronounced. In the case when $\hat{R} = 0$, (the anechoic case), the

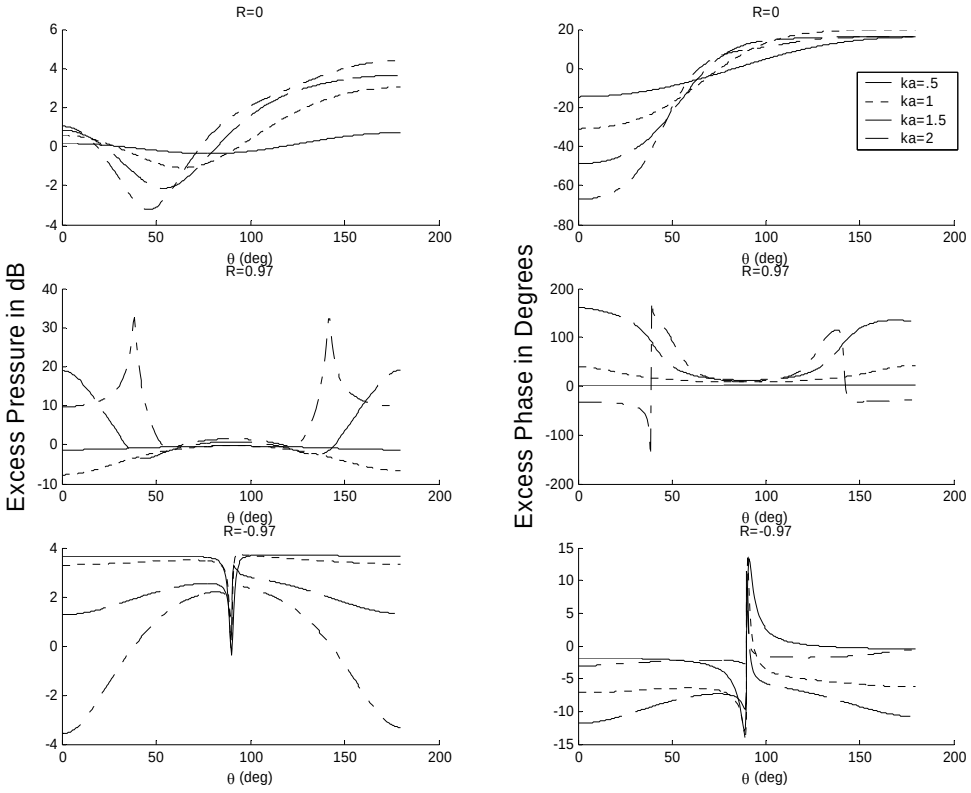


**Figure 4.13: Excess pressure magnitude and phase on the surface of a hard sphere for various reflection coefficients, as a function of incidence angle.**

excess pressure on the surface of the sphere goes through a phase change. The trend becomes more pronounced as *ka* increases. The dip in excess pressure shifts towards θ = 0 as *ka* increases. The sign of the excess phase changes from negative to positive when θ is about 60º. In the case when $\hat{R} = 0.97$, (the nearly rigid case), the excess pressure

fluctuates considerably as a function of θ. The excess pressure behavior is symmetric about θ = 90º. When *ka* = 2 and θ = 30º and 120º, the excess pressure has a maximum. The excess phase experiences a point of discontinuity and changes sign as well.

In the case when $\hat{R}$ = -0.97, (the pressure release case), the excess pressure fluctuates considerably as a function of θ. The excess pressure behavior was symmetric about θ = 90º and experiences a major dip at θ = 90º. Aside from the dip at 90º, the excess pressure is nominally flat at *ka* = 0.5 and approaches a bell shape at *ka* =2. At θ = 90º the excess phase experiences a point of discontinuity and changes sign as well.

## 4.7 Two-point Spherical Sensor Results

The following sub sections present the results of the two-point spherical sensor model for all three probe types and several complex reflection coefficients with mismatched microphones. A discussion of the significance of the results will follow the figures. In every case, the angle of incidence, θ, is taken to be zero.

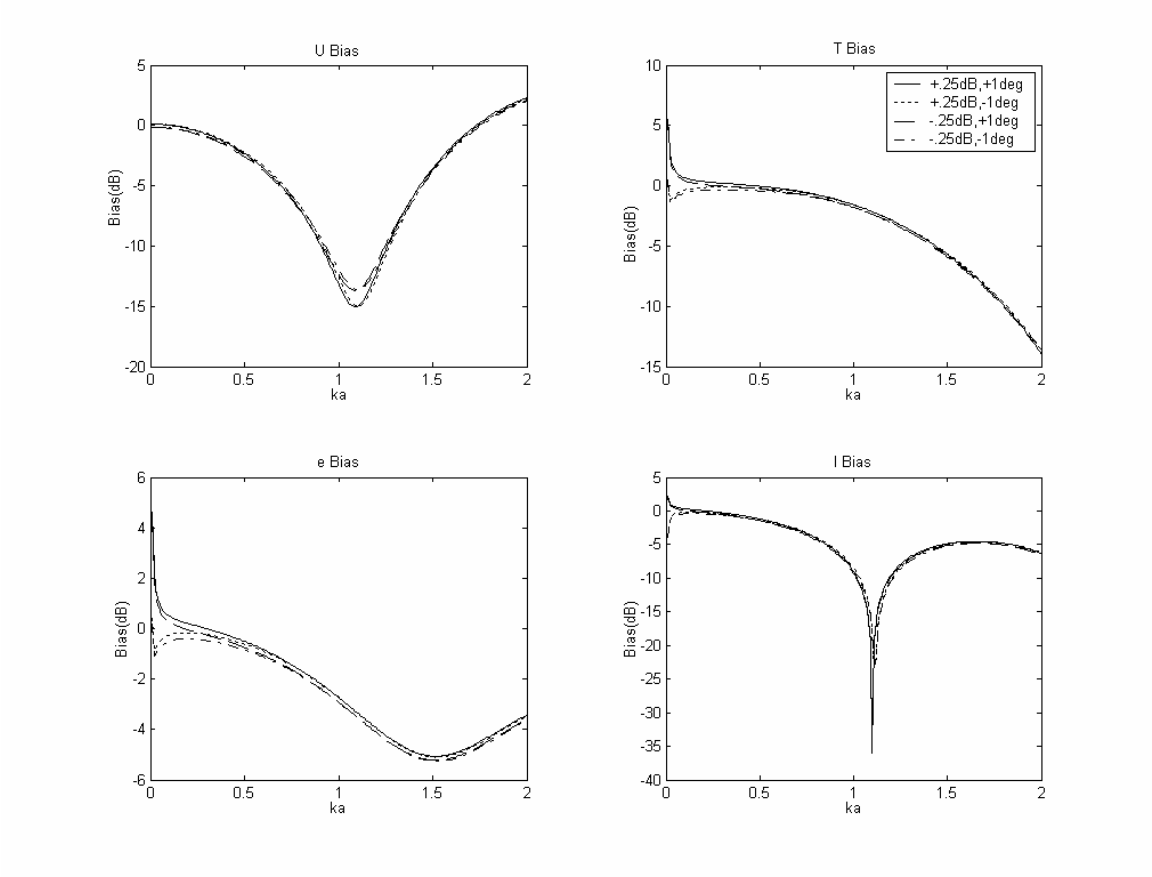


**Figure 4.14: Bias errors of a spherical sensor with mismatched microphones. R=0, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Six Microphone Probe**

Figure 4.14 shows the bias errors for a spherical sensor with mismatched microphones in the case when $\hat{R}$ =0. The angle of incidence, θ, is zero. The figure shows the results for the model that matches the six microphone geometry in one dimension. Scattering effects are taken into account in the model. The potential energy density bias errors steadily increase (i.e. increasing underestimation) with increasing *ka* until *ka* was nominally 1, at which point the bias errors reach the point where the potential energy density is underestimated most. The errors then decrease with increasing *ka* until the potential energy density is eventually overestimated by about 2 dB at *ka* =2. The kinetic energy density bias is increasingly underestimated as frequency increases. The total

energy density bias has a minimum at *ka* = 1.5 (maximum error) and has the same small *ka* effects that the kinetic energy density bias exhibits. The intensity magnitude bias error is the worst at $ka \approx 1.1$.

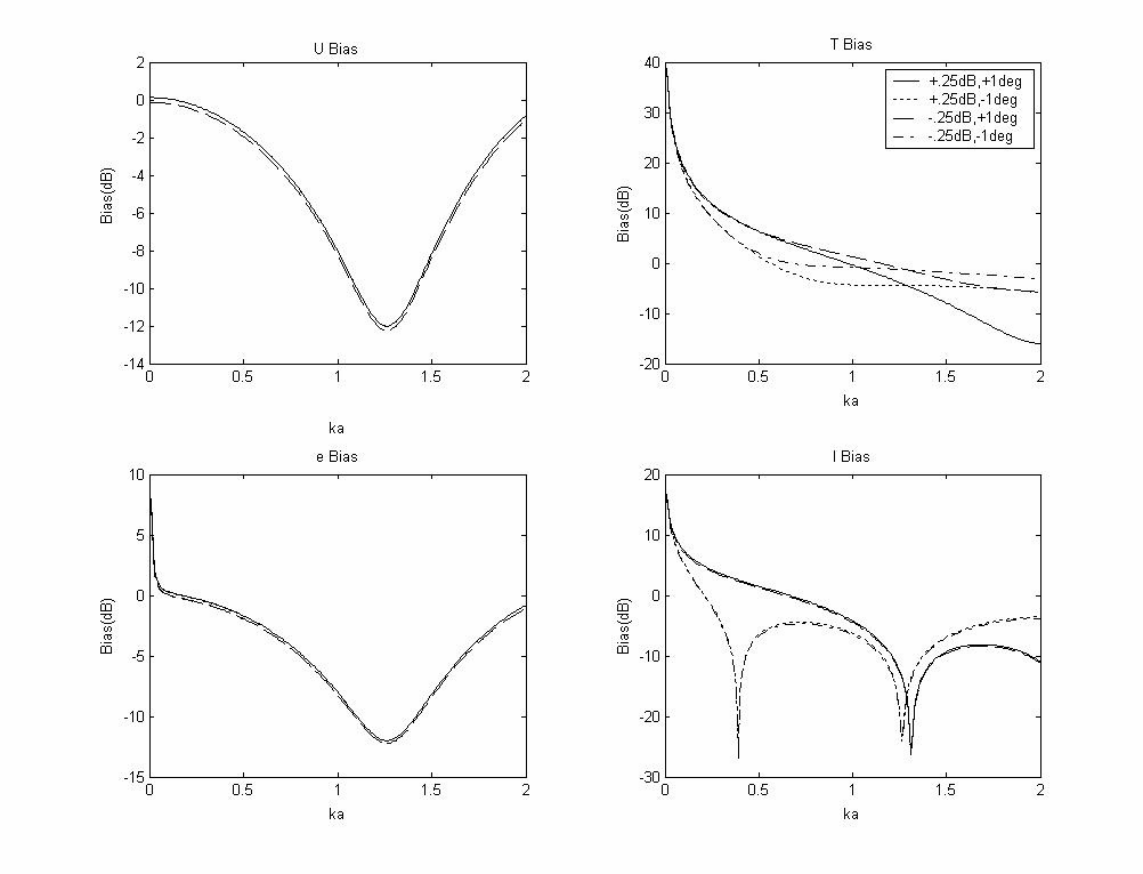


**Figure 4.15: Bias errors of a spherical sensor with mismatched microphones. R=0, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Tetrahedron Probe**

Figure 4.15 shows the results for the tetrahedron two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to zero, which represents the anechoic case. The angle of incidence, θ, is zero.

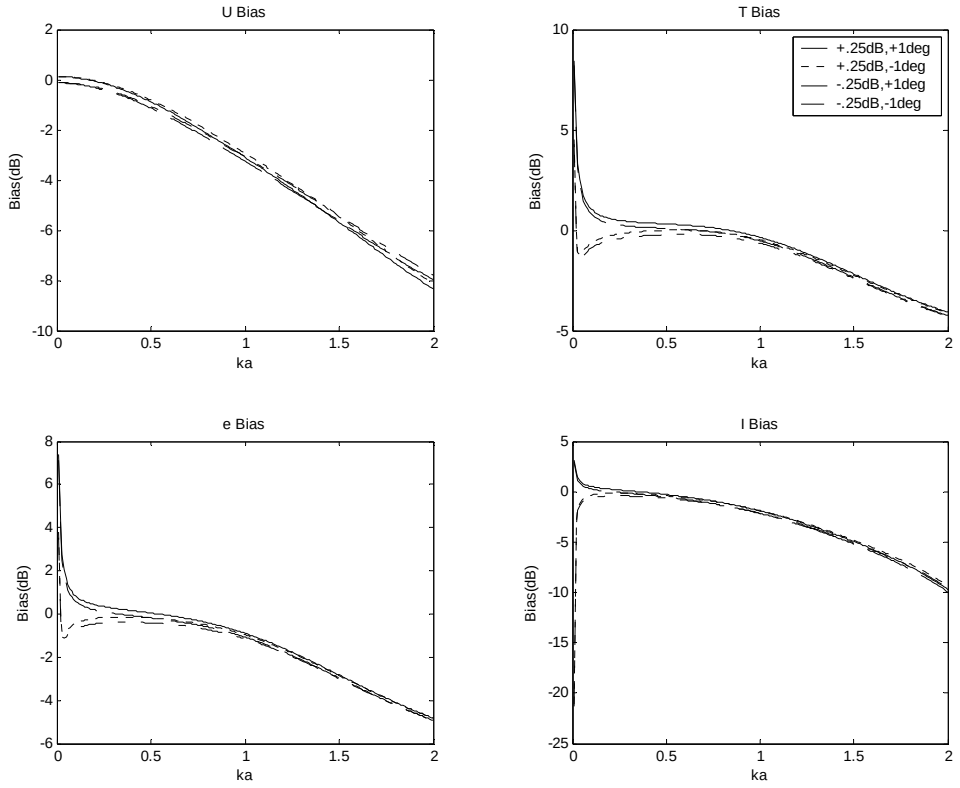


**Figure 4.16: Bias errors of a spherical sensor with mismatched microphones. R=0, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Orthogonal Probe**

Figure 4.16 shows the results for the orthogonal two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to zero, which represents the anechoic case. The angle of incidence, θ, is zero.

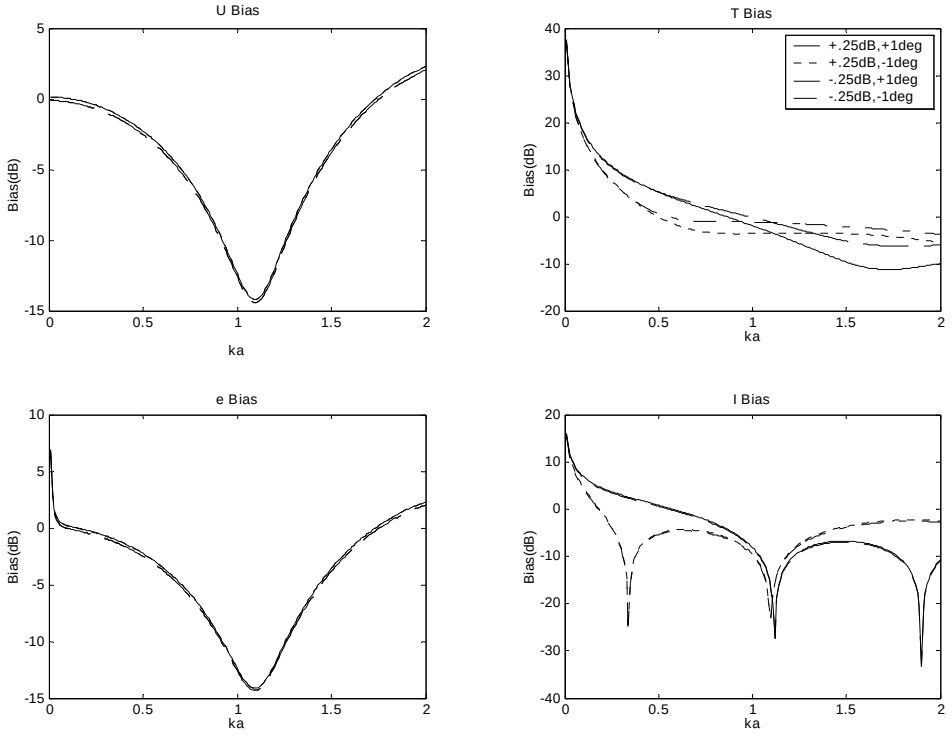


**Figure 4.17: Bias errors of a spherical sensor with mismatched microphones. R=.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Six Microphone Probe**

Figure 4.17 shows the results for the six microphone two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to .97, which represents the nearly rigid enclosure case. The angle of incidence, θ, is zero.

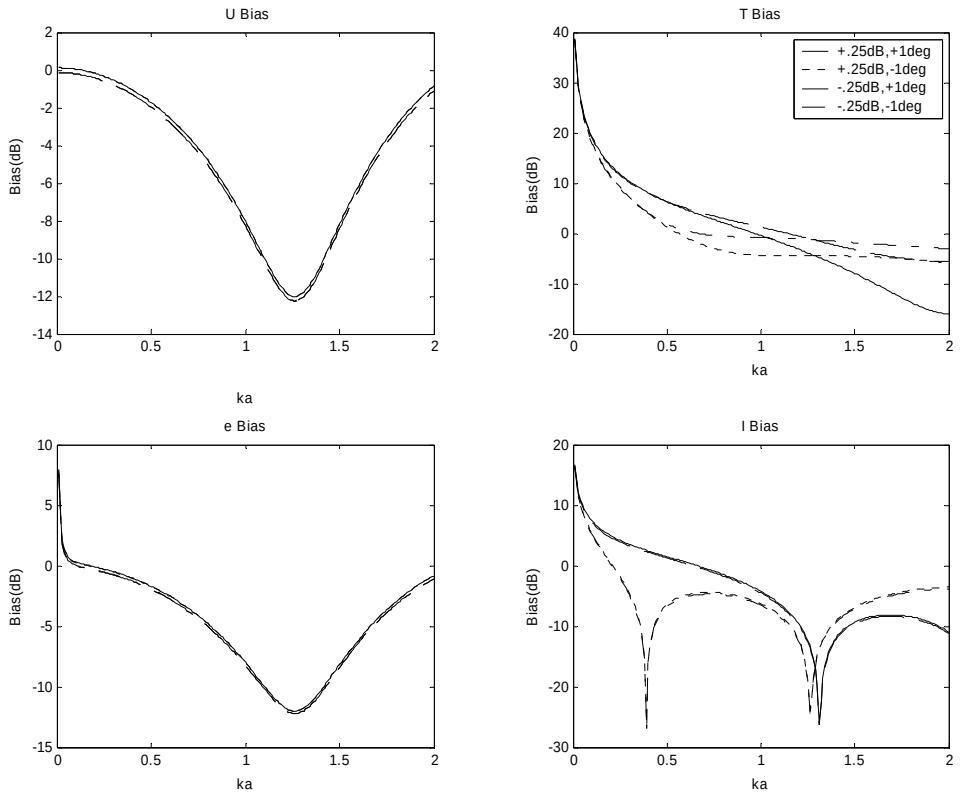


**Figure 4.18: Bias errors of a spherical sensor with mismatched microphones. R=.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Tetrahedron Probe**

Figure 4.18 shows the results for the tetrahedron two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to .97, which represents the nearly rigid enclosure case. The angle of incidence, θ, is zero.

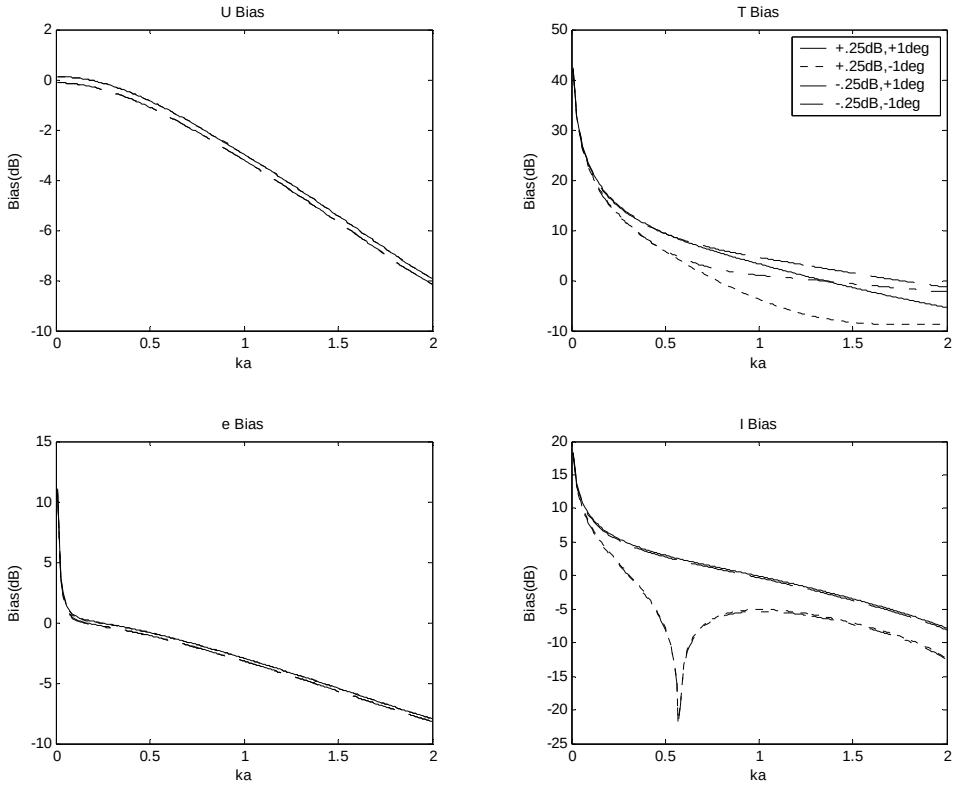


**Figure 4.19: Bias errors of a spherical sensor with mismatched microphones. R=.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Orthogonal Probe**

Figure 4.19 shows the results for the orthogonal two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to .97, which represents the nearly rigid enclosure case. The angle of incidence, θ, is zero.

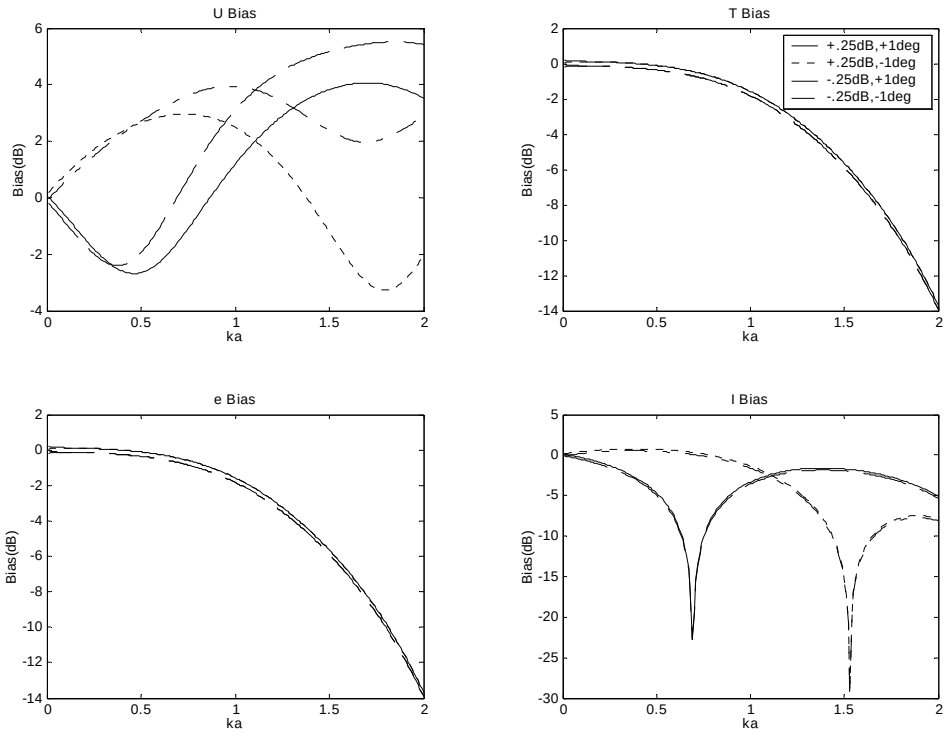


**Figure 4.20: Bias errors of a spherical sensor with mismatched microphones. R=-.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Six Microphone Probe**

Figure 4.20 shows the results for the six microphone two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to -.97, which represents the pressure release case. The angle of incidence, θ, is zero.

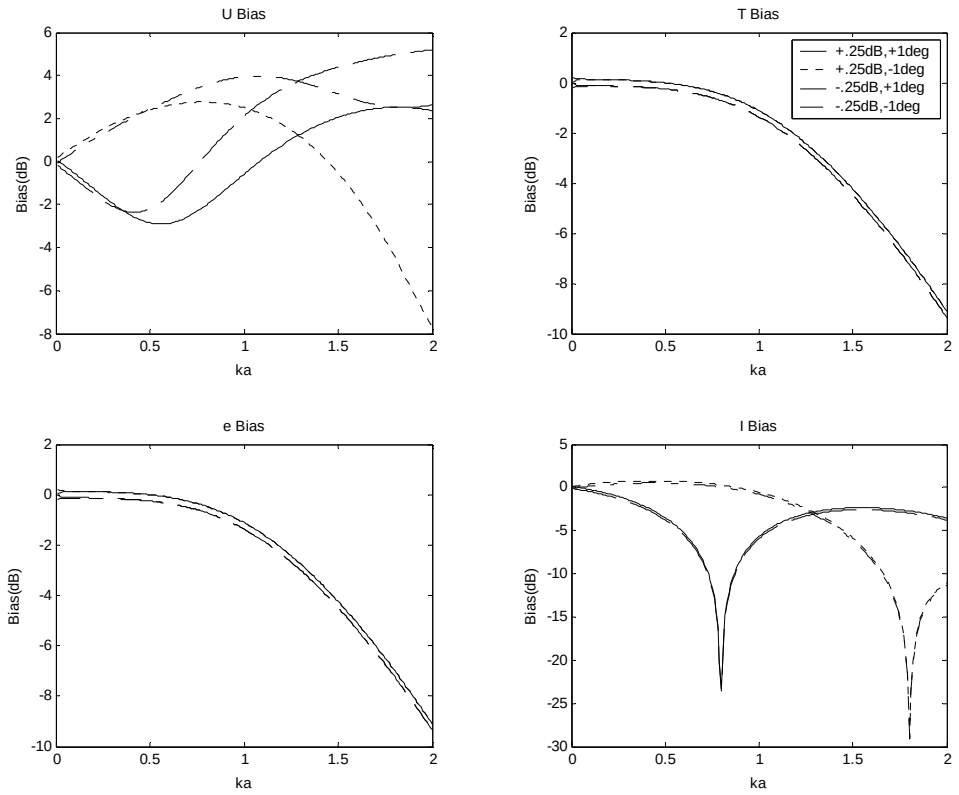


**Figure 4.21: Bias errors of a spherical sensor with mismatched microphones. R=-.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Tetrahedron Probe**

Figure 4.21 shows the results for the tetrahedron two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to -.97, which represents the pressure release case. The angle of incidence, θ, is zero.

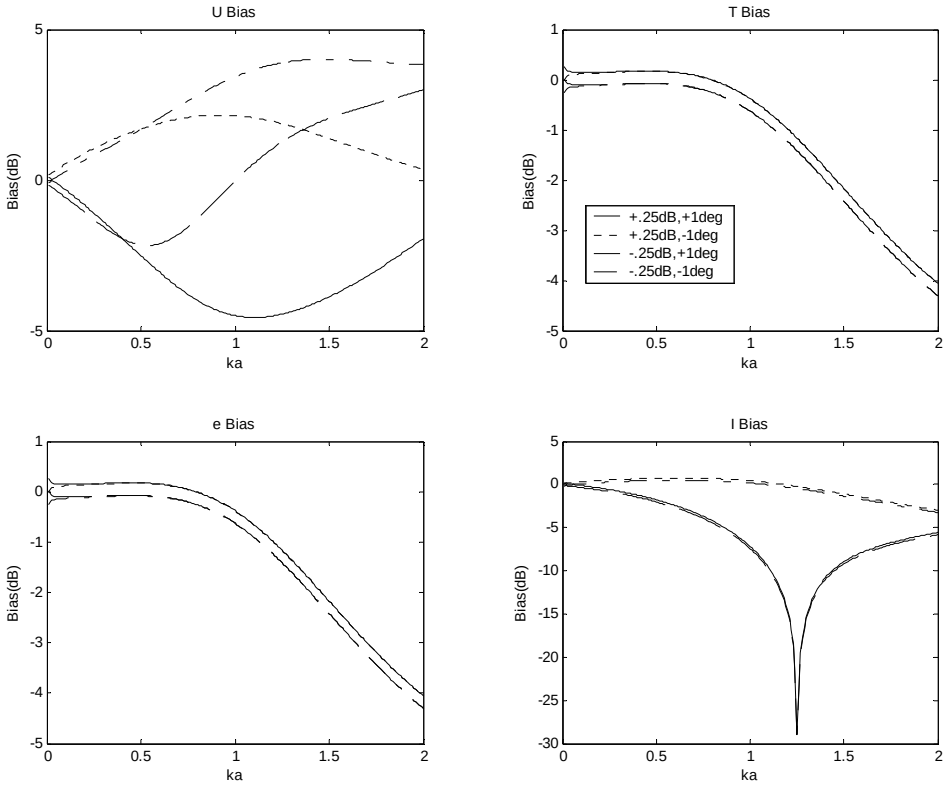


**Figure 4.22: Bias errors of a spherical sensor with mismatched microphones. R=-.97, θ=0. Potential energy density bias (U). Kinetic energy density bias (T). Total energy density bias (e). Magnitude of intensity bias (I). Orthogonal Probe**

Figure 4.22 shows the results for the orthogonal two-point spherical sensor with mismatched microphones. The complex reflection coefficient, $\hat{R}$, is equal to -.97, which represents the pressure release case. The angle of incidence, θ, is zero.

## 4.8 Discussion of Two-point Sensor Results

When comparing Figure 4.14, Figure 4.15, and Figure 4.16, the first notion that comes to mind is how all the figures follow the same trend. The minima in the figures initially suggest a scaling factor, due to the differences in separation distance. When comparing the six microphone probe (Figure 4.14) to the tetrahedron probe (Figure 4.15), the minima of the potential energy density are around *ka* of 1.1 and *ka* of 1.25. If the

frequencies in Figure 4.14 were multiplied by 1.15, the result would be nearly identical to Figure 4.15. The ratio of *kd* between the six microphone model and the tetrahedron model is 0.866. Thus, the appropriate scale factor for these two models would be the inverse of 0.866, which is nominally 1.15.

Applying this same logic when comparing the six microphone model to the orthogonal model would yield a scale factor of approximately 1.7. However, when comparing Figure 4.14 and Figure 4.16, simply scaling the frequencies of the six microphone model by 1.7 is not sufficient to yield the orthogonal results.

It is hypothesized that in the case of the orthogonal probe, two effects are taking place. The first is the *kd* scaling, as mentioned above. The second has to do with the excess pressure, as shown in Figure 4.13. It is hypothesized that the orthogonal probe microphone placement lines up with the excess pressure in such a way that is beneficial in reducing the error.

# CHAPTER 5

## EXPERIMENTAL RESULTS

This chapter will discuss the experimental results. The energy density measurements made by the large probes will be presented. The energy density measurements made by the small probes resulted in erroneous data. The erroneous data is the result of imperfect calibration methods for the small probes. This problem will be discussed in greater detail later.

The figures in this chapter represent the average energy density spectra that were measured by the given probe, at the various locations. At the end of each section, a graph shows the average results at 500 Hz for each probe measurement location—A, B, C, and R (see Section 3.7.1). The measurements indicate that the large six microphone probe and the large tetrahedron probe both performed as would be expected, with regard to 1/r decay. A doubling of distance from the source should yield a 6 dB drop in energy density, and the figures indicate the same. The large orthogonal probe shows the same 6 dB drop; however it does not perform as well in the rotated position. The log scale in the figures all use $3*10^{-15}\,\frac{Joules}{m^3}$ as the reference value.

## 5.1 Large Six Microphone Probe

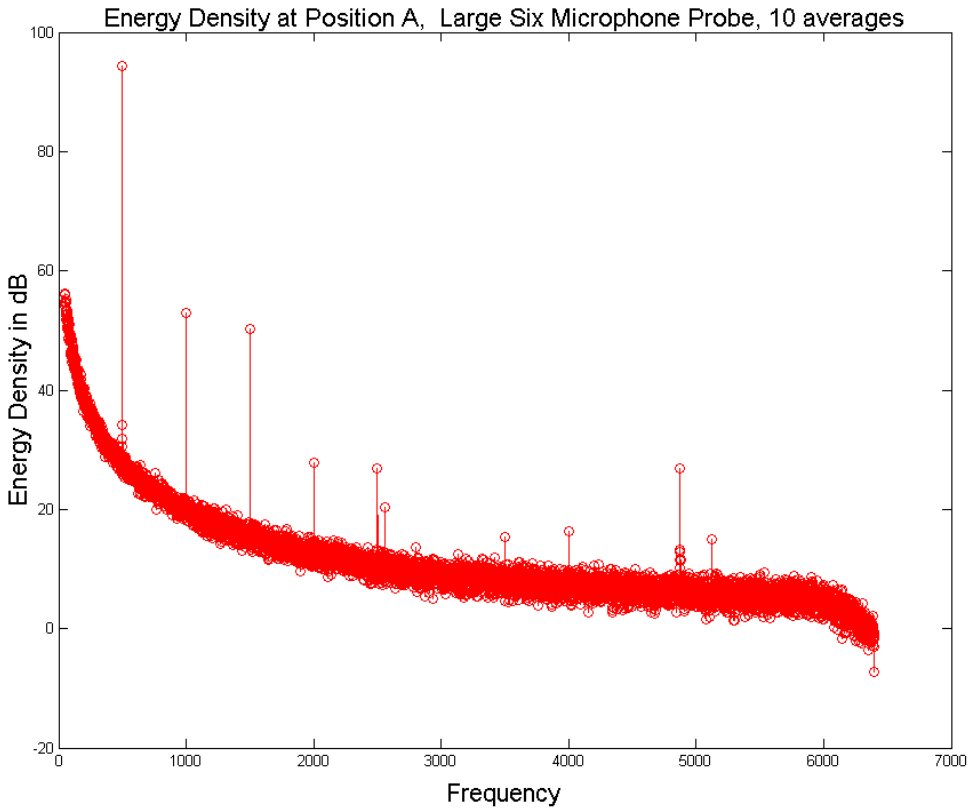


**Figure 5.1: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position A, Six Microphone Probe**

Figures 5.1 through 5.4 show the energy density spectrum measured by the six microphone probe at the respective measurement position (A,B,C,R). Each figure shows harmonic distortion that is nominally 50 dB below the signal. This distortion is caused by either the source or the analog to digital converter, not the microphones.

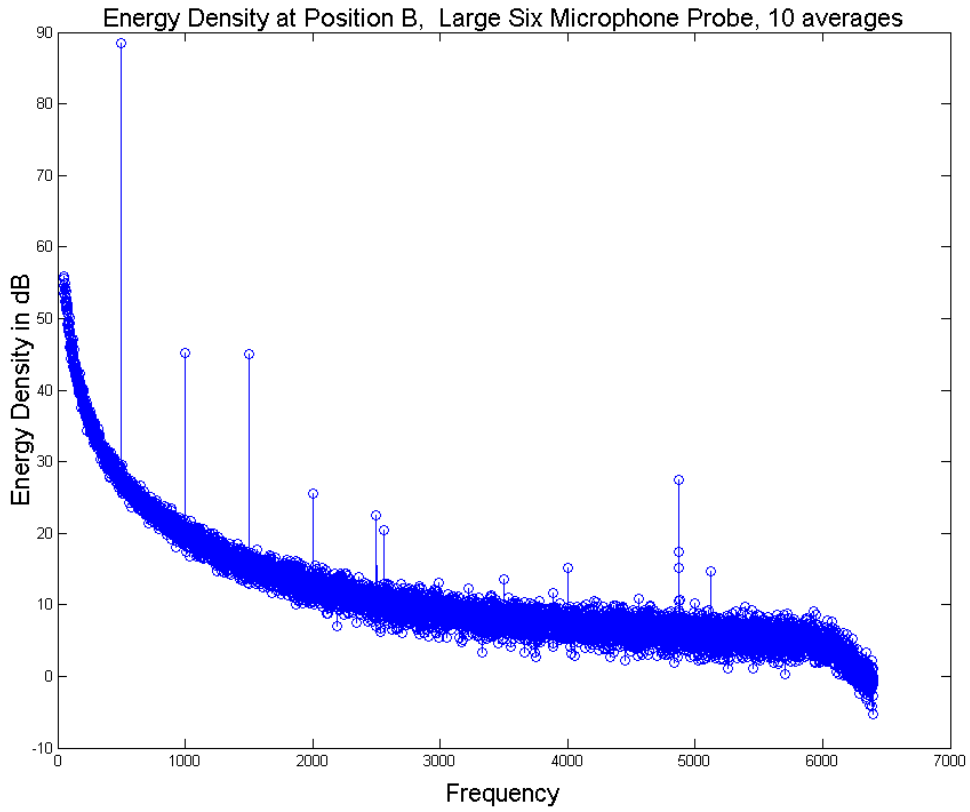


**Figure 5.2: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position B, Six Microphone Probe**

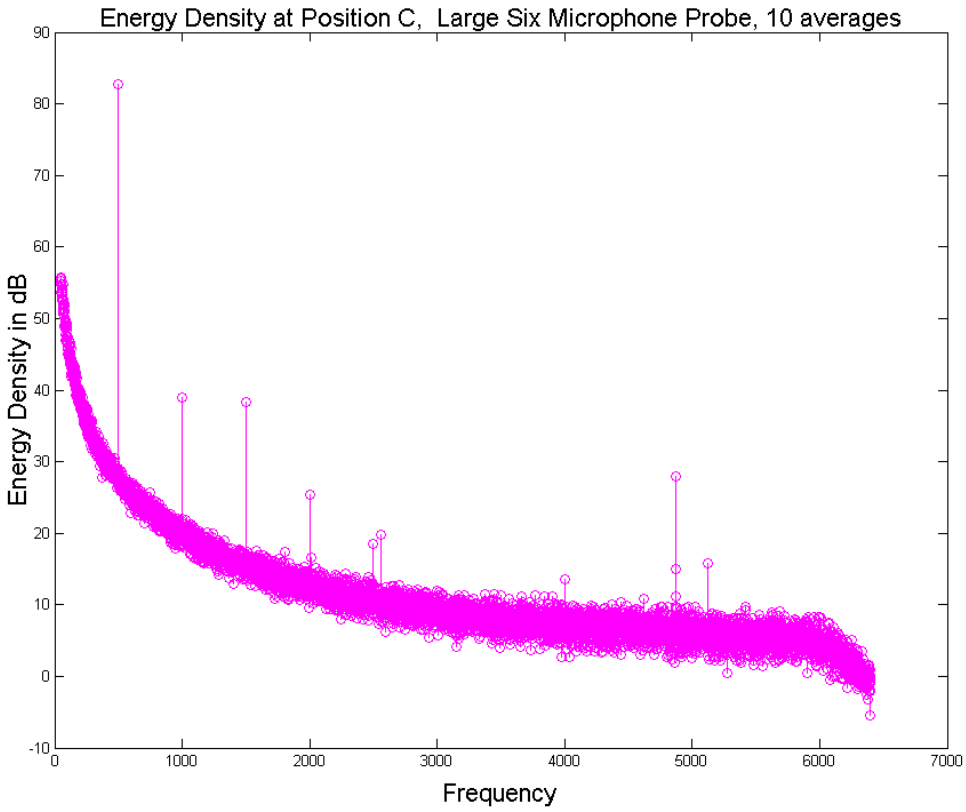


**Figure 5.3: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position C, Six Microphone Probe**

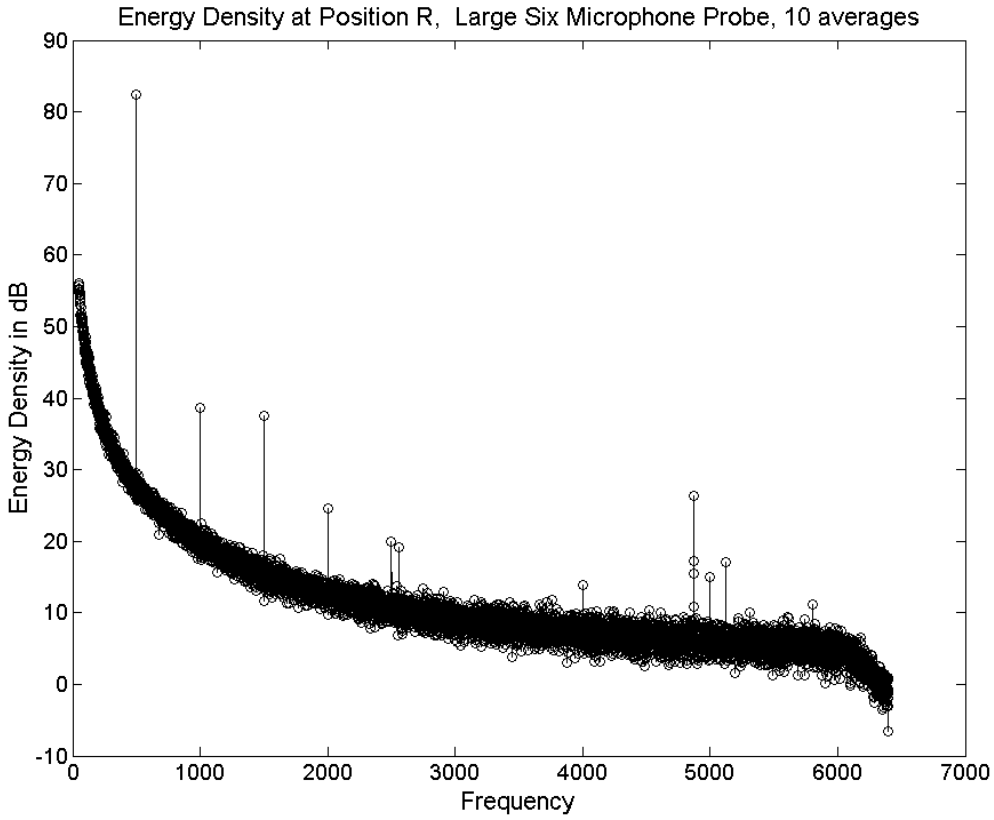


**Figure 5.4: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position R, Six Microphone Probe**

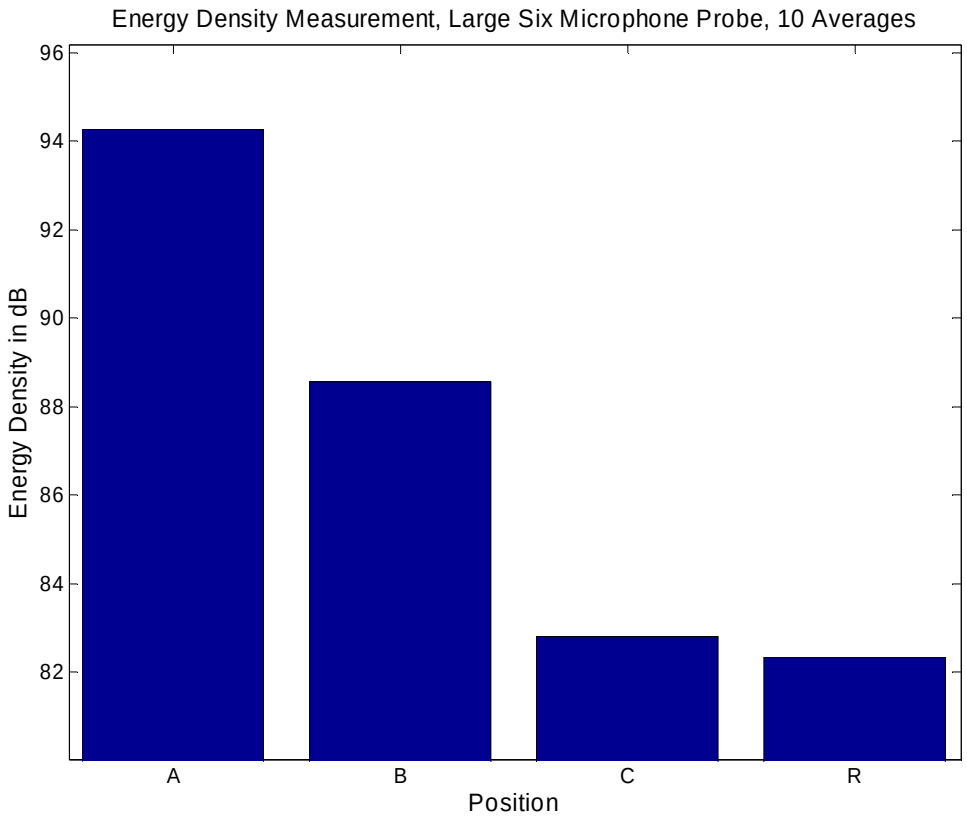


**Figure 5.5: Energy Density at 500 Hz, Large Six Microphone Probe**

Figure 5.5 shows the acoustic energy density, measured in decibels at 500 Hz, by the large six microphone probe. The probe behaves as expected, with the energy density falling off at 6 dB per doubling of distance. Additionally, the probe is not affected by rotations.

## 5.2 Large Tetrahedron Probe

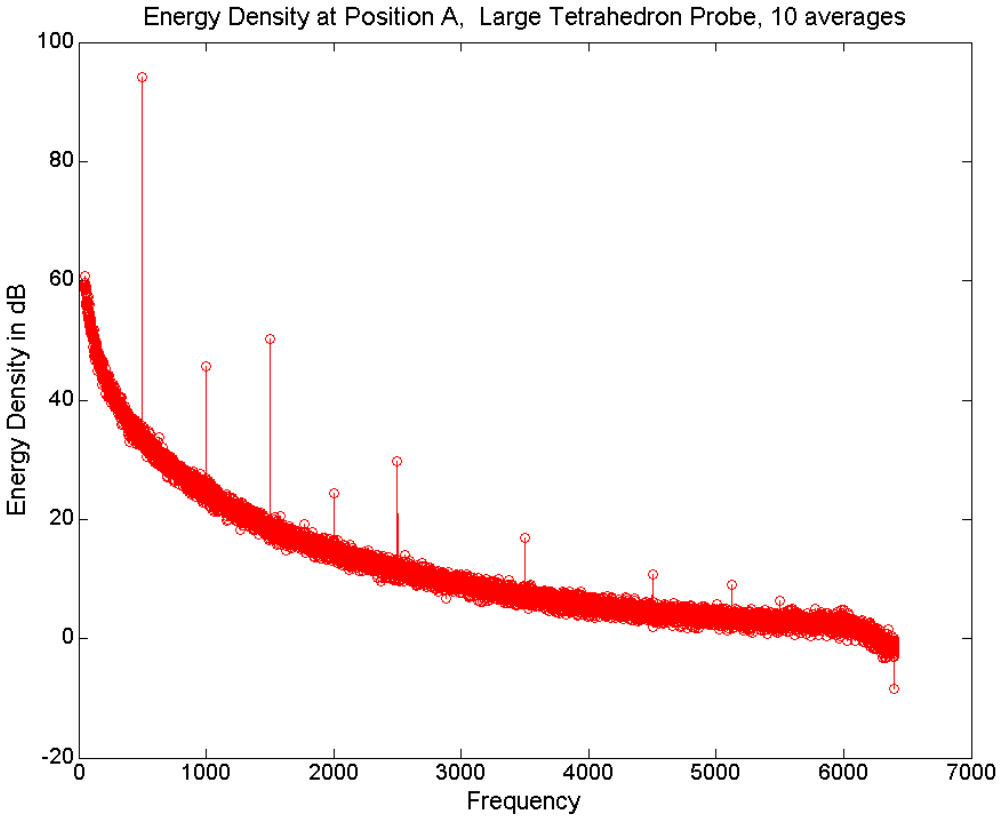


**Figure 5.6: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position A, Large Tetrahedron Probe**

Figures 5.6 through 5.9 show the energy density spectrum measured by the tetrahedron probe at the respective measurement position (A,B,C,R). Each figure shows harmonic distortion that is nominally 50 dB below the signal. This distortion is caused by either the source or the analog to digital converter, not the microphones.

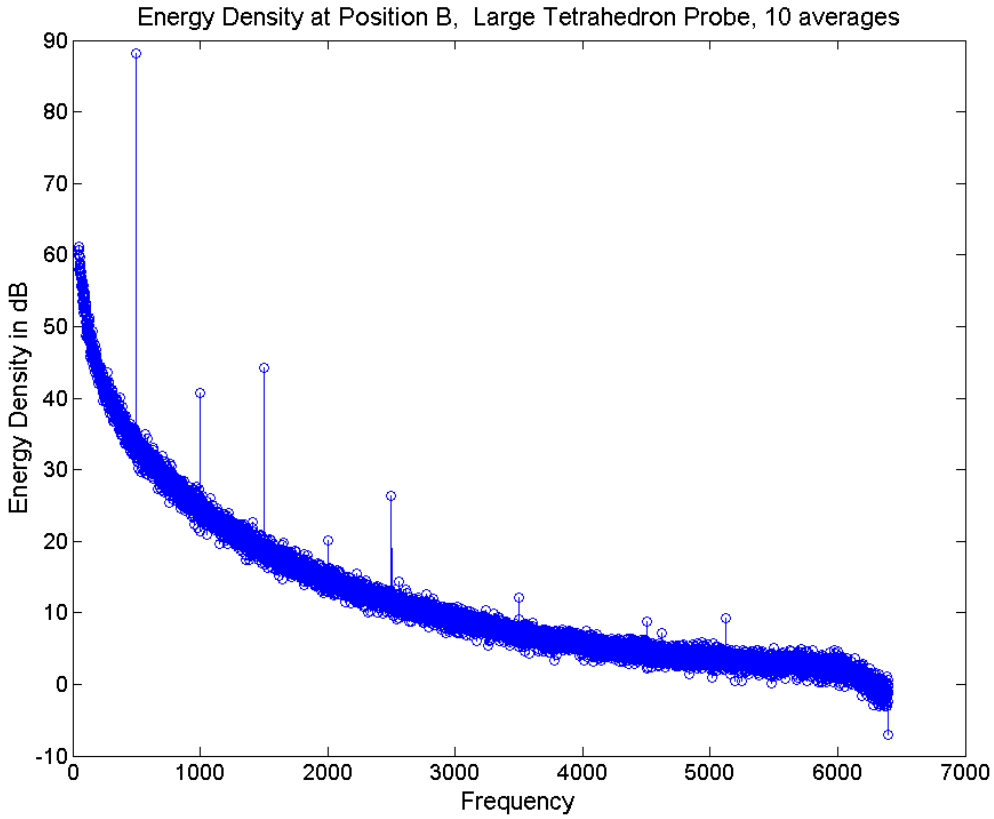


**Figure 5.7: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position B, Tetrahedron Probe**

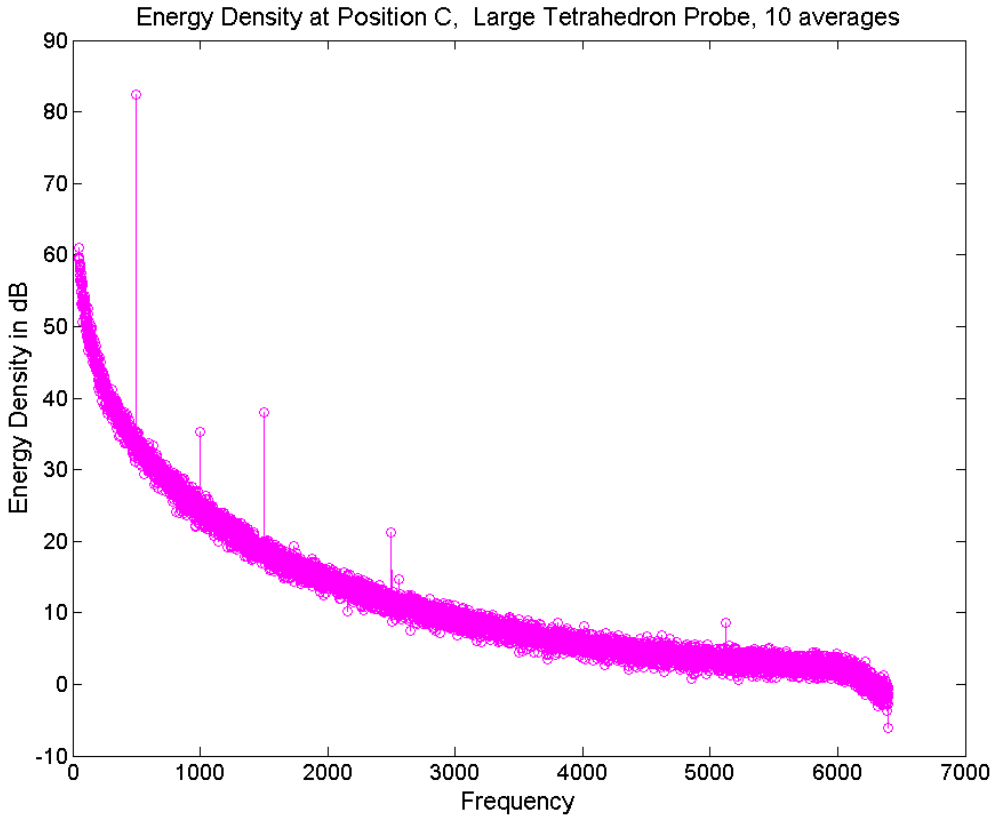


**Figure 5.8: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position C, Large Tetrahedron Probe**

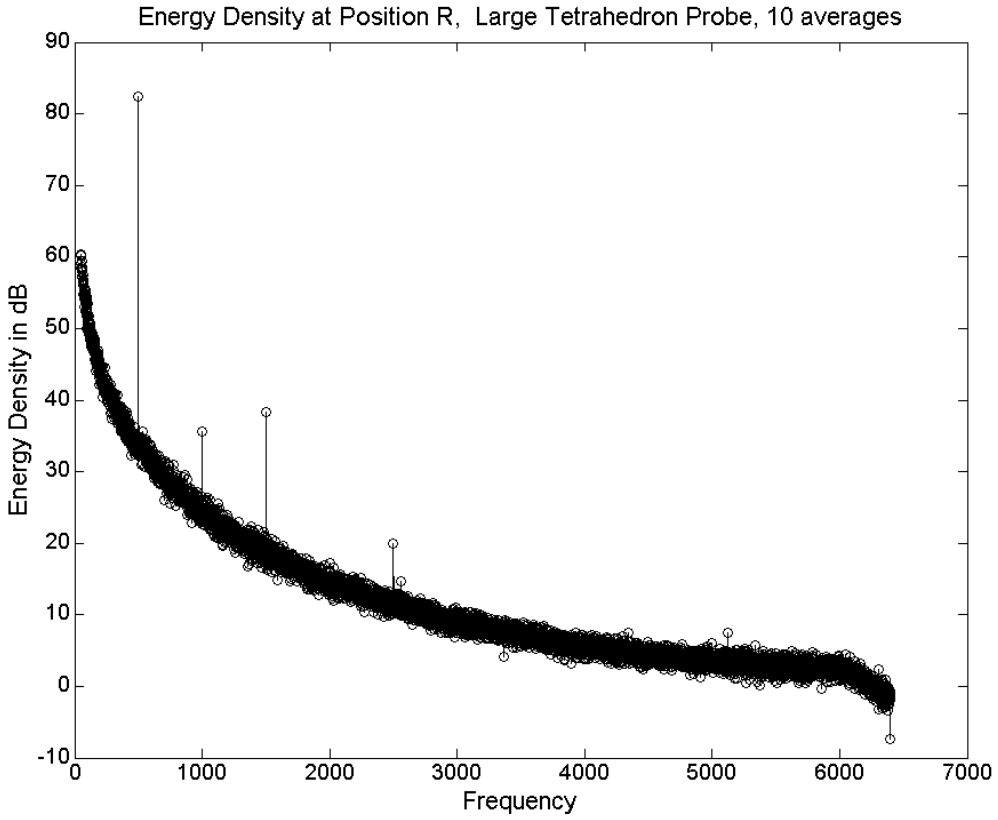


**Figure 5.9: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position R, Large Tetrahedron Probe**

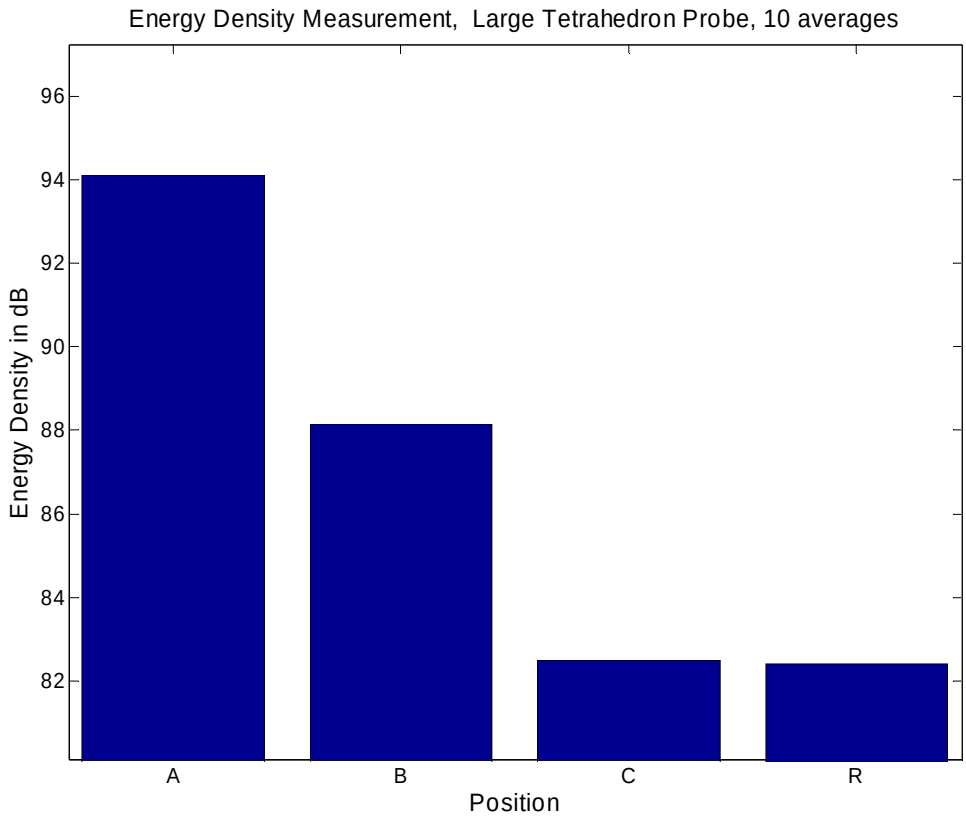


**Figure 5.10: Energy Density at 500 Hz, Large Tetrahedron Probe**

Figure 5.10 shows energy density, measured in decibels at 500 Hz, by the tetrahedron probe. The measured energy density falls off as 1/r, or 6 dB per doubling of distance. The tetrahedron probe behaves well in the rotated position and slightly better than the six microphone probe.

## 5.3 Large Orthogonal Probe

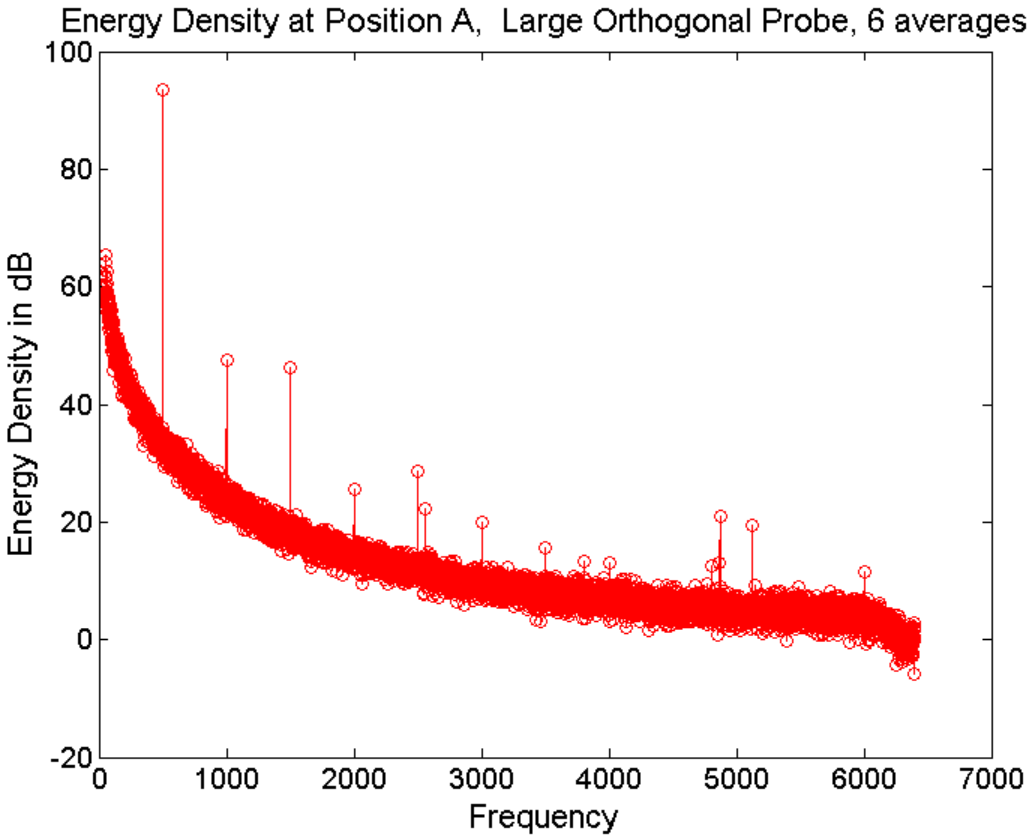


**Figure 5.11: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position A, Large Orthogonal Probe**

Figures 5.11 through 5.14 show the energy density spectrum measured by the orthogonal probe at the respective measurement position (A,B,C,R). Each figure shows harmonic distortion that is nominally 50 dB below the signal. This distortion is caused by either the source or the analog to digital converter, not the microphones.

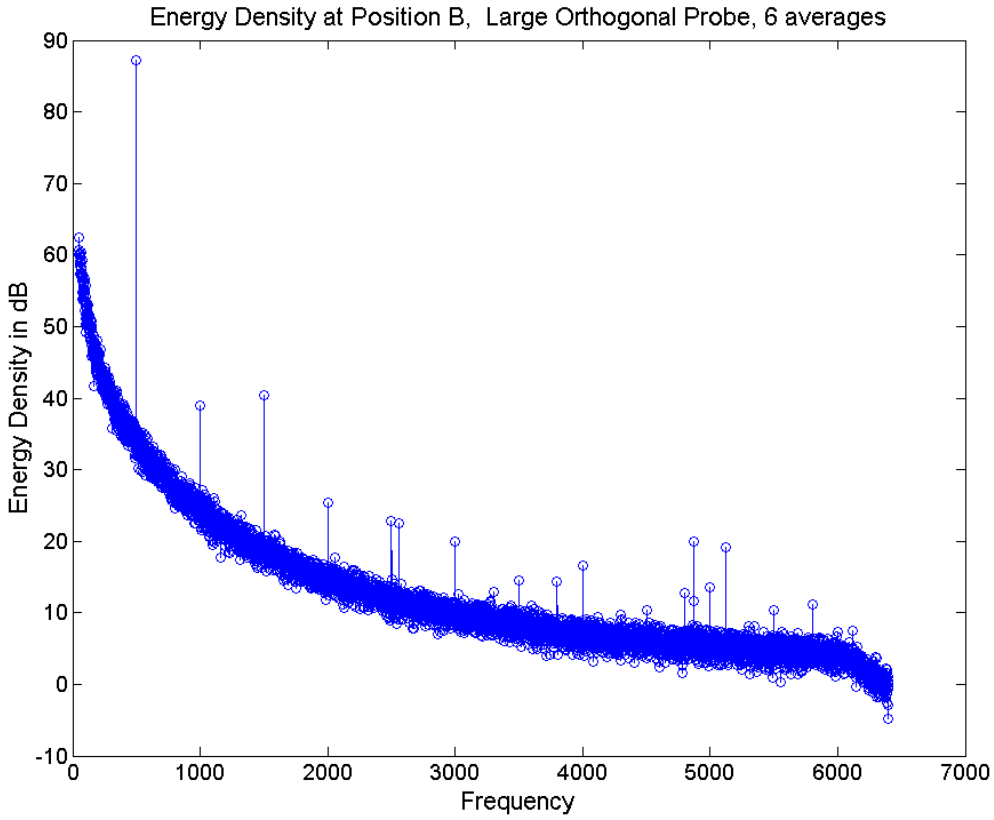


**Figure 5.12: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position B, Large Orthogonal Probe**

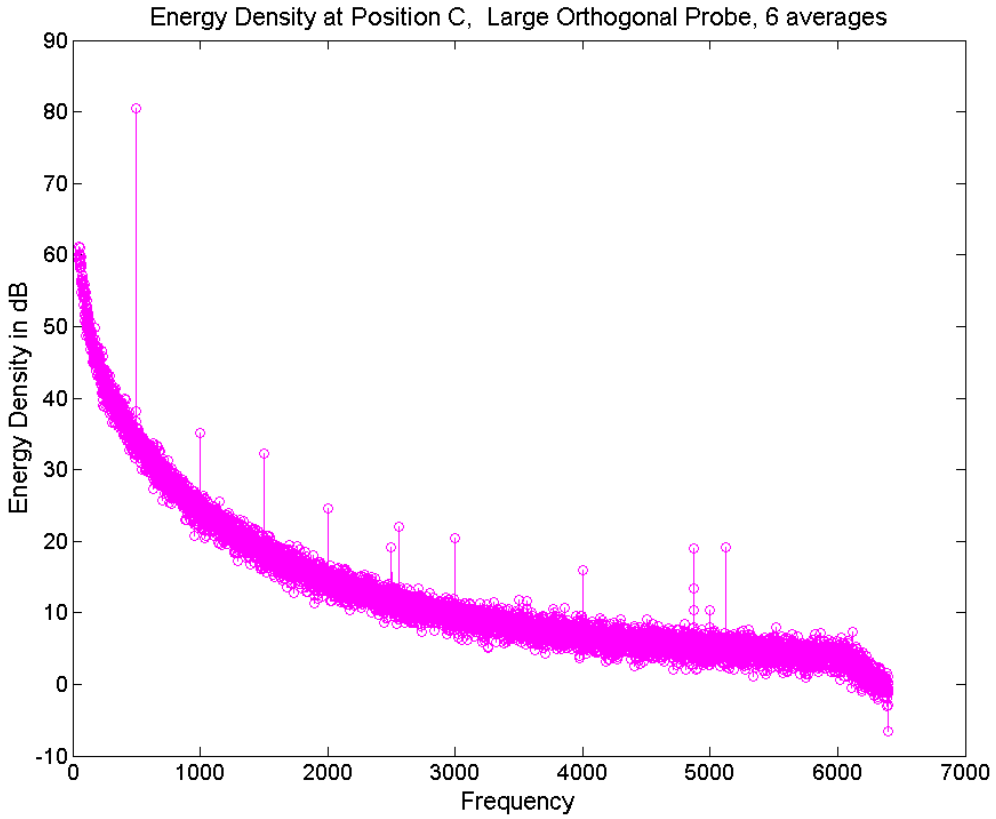


**Figure 5.13: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position C, Large Orthogonal Probe**

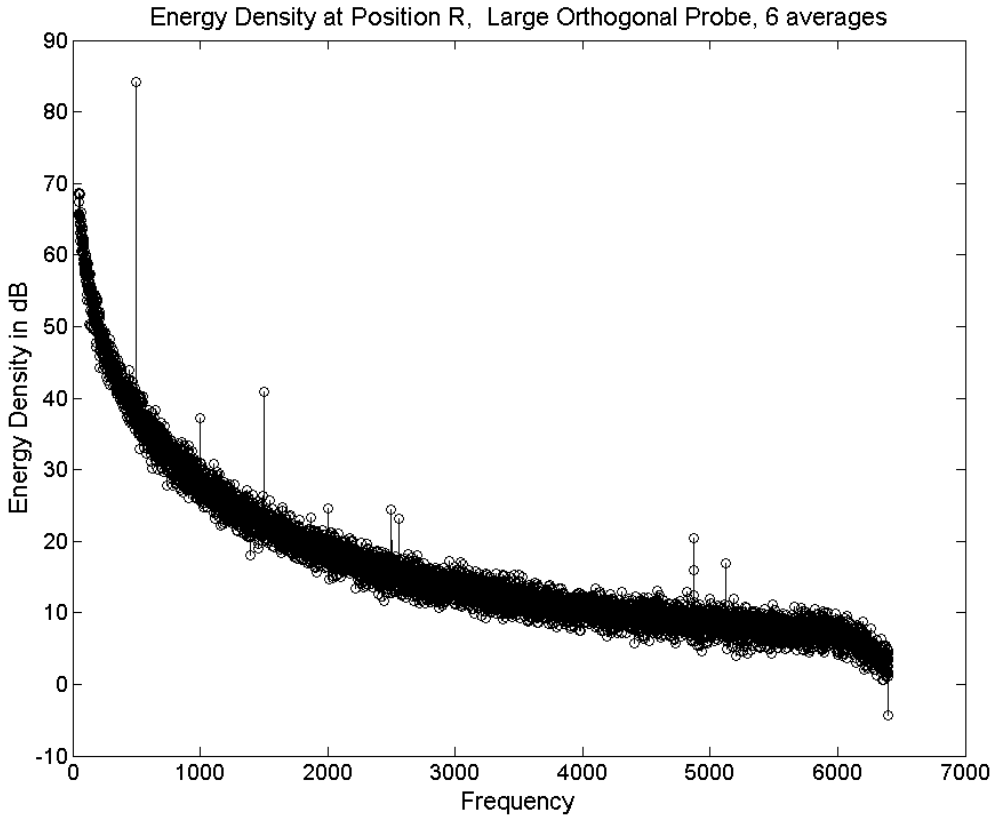


**Figure 5.14: Energy Density Spectrum (re $3x10^{-15}$ J/m^3), Position R, Large Orthogonal Probe**

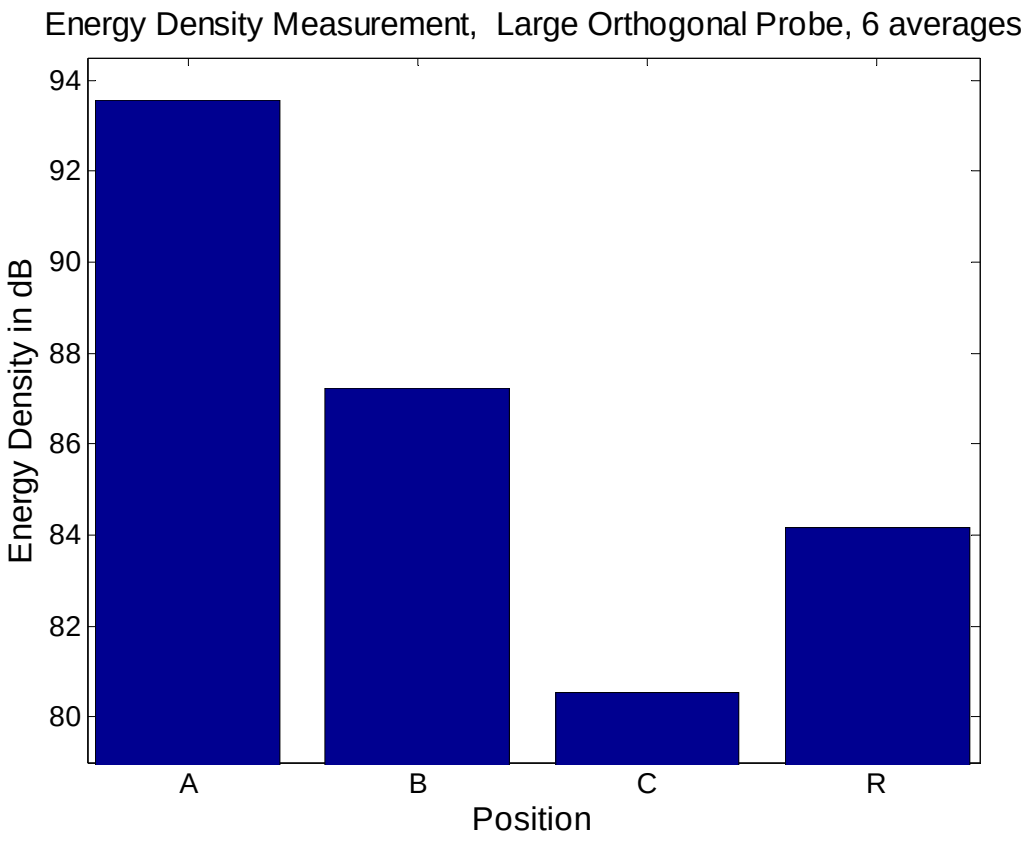


**Figure 5.15: Energy Density at 500 Hz, Orthogonal Large Probe**

Figure 5.15 shows that the orthogonal probe is not as accurate at measuring energy density as the six microphone and tetrahedron probes. The difference in acoustic energy density between positions B and C is closer to 7 dB than the ideal 6 dB. It is possible that the orthogonal probe algorithm for estimating particle velocity is more sensitive to calibration error than the six microphone and tetrahedron probes are. It is also possible that the probe was not positioned at exactly 6 m from the source. The measured acoustic energy density in the rotated position is higher than expected. Again, this could be due to calibration errors, position errors, or a combination of both.

## 5.4 Discussion of Experimental Results

Several conclusions can be drawn from these limited experimental results. The first conclusion that can be made based on the experimental results is that the orthogonal probe appears to be the most susceptible to measurement errors when the probe is in the rotated position. This conclusion is drawn from figure 5.15. The second conclusion that can be made is that the tetrahedron probe appears to be very capable of making accurate acoustic energy density measurements.

## 5.5 Errors and Limitations

Errors can be introduced by several factors. One large group of measurement errors are caused by hardware imperfections. For example, it was generally assumed that a properly calibrated microphone will always yield the same value when measuring a given acoustic field. However, one possible source of error is microphone drift, when a microphone experiences a change in frequency response over time. This can be caused by a variety of factors, such as a change in the temperature of the microphone electronics or imperfections in the materials used to construct the microphone. Microphone drift can cause errors when many measurements are made during some time interval, when the microphones are powered on continuously for an extended period. Another potential error can be caused by improperly matched microphones. It is difficult to determine how significant this error was, given that the probes were assembled outside of the author's control.

Each microphone will have both a magnitude and phase response to a given field. If two microphones are exposed to identical acoustic fields, in principle they should both have identical frequency responses. In practice, however, it is possible that the two

microphones will have a slight relative phase mismatch and potentially a magnitude mismatch as well. These differences can then create errors when measuring pressure, and consequently, estimating both kinetic and potential energy density. Parkins' work investigated some of the consequences of microphone mismatch, as described earlier in the previous chapter.

Other errors that can exist in a measurement include positioning errors and estimation errors. When estimating particle velocity, the numerical implementation of Euler's equation requires that the separation distance between microphones be some fixed value. If the actual separation distance was different from the value used to calculate the particle velocity, an error will result. Additionally, errors are introduced when performing a finite sum to estimate pressure (bias error) or a finite difference to estimate particle velocity (finite difference error).

One benefit of measuring energy density was that the dominant error was generally not numerically significant. Because energy density was comprised of the sum of kinetic and potential energy density, a high error term usually has a low numeric value, when compared to the other term. Thus, it has little overall effect in estimating the total acoustic energy density. In other words, if the potential energy density estimate is under estimated by one order of magnitude, but it is three orders of magnitude less than the kinetic energy density measurement, then when the two terms are added together, the error in the pressure estimate will have little effect on the overall error.

# CHAPTER 6

# CONCLUSIONS

Several conclusions can be drawn from this work. The first conclusion is that it is possible to measure 3-dimensional acoustic energy density accurately with only four microphones. The numerical results suggest that the orthogonal probe has the potential for the greatest useable frequency range. However, the initial experimental results suggest that the orthogonal probe may have trouble accurately measuring acoustic energy density at some probe orientations. Therefore, it appears that more work should be done in order to determine why the orthogonal probe did not perform experimentally as expected, as outlined in previous chapters.

It is clear that the effects due to scattering about the probe can be beneficial. As ka increases, the differences in excess pressure, as shown in Figure 4.13, lead to greater variation in the pressure field. This may create an upper limit to the useable frequency range.

Currently, the numerical results suggest that the orthogonal probe holds the most promise for measuring acoustic energy density. However, the experimental results suggest that the orthogonal probe is not as accurate as the tetrahedron probe in the case when the orthogonal probe is not oriented with one measurement axis along the radial axis of the source. Consequently, more work is necessary to determine which probe is better. For example, additional testing could be done to explore which probe is the least susceptible to error as a function of rotation. It is clear, though, that it is very feasible to measure acoustic energy density with four microphones embedded on the surface of a sphere.

## 6.1 Significance and implications of results

The numerical results suggest that the orthogonal probe is best suited to measure acoustic energy density because it has the broadest useable frequency range of the three probe designs. Acceptable errors are considered to be within +/- 1 dB. According to Figure 4.4, the orthogonal probe has the highest upper frequency limit, with a limit of 1 dB at $\approx$ 1.4 *ka*. This compared with the six microphone probe and the tetrahedron probe, which have a l dB limit at *ka* $\approx$ 1.1. Figure 4.16 shows what happens when the orthogonal probe is placed in an anechoic environment with mismatched microphones. The energy density bias errors of the orthogonal probe at *ka* =1 are approximately 1 dB. Figure 4.14 shows what happens when the six microphone probe is placed in an anechoic environment with mismatched microphones. The energy density bias errors of the six microphone probe at *ka* =1 are approximately 3 dB. The difference between the six microphone probe and the orthogonal probe suggests that the orthogonal probe should be more accurate at measuring energy density.

## 6.2 Limitations of Results

The numerical results are limited to nondimensional *ka* values up to 2. For the large probe, a ka value of 2 corresponds to roughly 2150 Hz. In the case of the small probe, a ka value of 2 corresponds to roughly 6600 Hz. The experimental results are only valid for 500 Hz, because that was the frequency of the source tone. All other signals were due to either noise or harmonic distortion.

## 6.3 Recommendations for Future Work

There are several areas where additional work could be done. In the numerical domain, more work could be done to investigate the effects of scattering as a function of

*ka*. Such an investigation would yield an increased understanding of how scattering about the sphere changes with increased frequency. The work presented in this thesis assumed that the angle of incidence, $\theta$, for the two-point and two-point spherical sensor, had the greatest error when $\theta$=0. This assumption should be investigated for the case of the tetrahedron and orthogonal probes, to verify that it is correct. Additionally, the thirteen terms used in the scattered pressure, equation (2-5), should be checked to ensure that no more terms are necessary for values of *ka* up to 2. A long term objective of this research should be to derive sound power measurements from acoustic energy density measurements.

Experimentally, work should be done in several areas. The first step to improve the experiment would be to develop a better method for calibrating the small diameter probes. The current method for calibrating the probes has a very high possibility for air leaks, which can lead to an inaccurate calibration.

Once the probes have been calibrated, a method should be developed to verify that the probes accurately measure acoustic energy density. One possible method could involve the use of an intensity probe to determine particle velocity, due to a source at some known distance from the source. A calibrated microphone could then be used to determine pressure at the same point. The energy density at that point could then be calculated and compared with the results obtained by the probes.

Another task would be to test the probes in conjunction with various reflecting surfaces. It would then be advantageous to compare the numerical results with the experimental results to see if the numeric models compare with the experimental data. Another experiment would be to measure acoustic energy density in an environment

where two or more sound sources exist to see if the probes can distinguish between different sound sources.

## REFERENCES

1 J. Ghan, B. S. Cazzolato, and S. D. Snyder, "Expression for the estimation of time-averaged acoustic energy density using the two-microphone method (L)," Journal of the Acoustical Society of America, Vol. 113, No. 5, pp. 2404–2407, (2003).

2 G. W. Elko, "An acoustic vector-field probe with calculable obstacle bias," Proc. Noise-Con '91, Tarrytown, NY, 1991, pp. 525-532.

3 J. W. Parkins, *Active Minimization of Energy Density in a Three-Dimensional Enclosure,* (Ph.D. dissertation, The Pennsylvania State University, University Park, PA, 1998) pp. 37-58.

4 J. W. Parkins, S. D. Sommerfeldt, and J. Tichy, "Narrowband and broadband active control in an enclosure using the acoustic energy density," Journal of the Acoustical Society of America, Vol. 108, No. 1, pp. 192-203 (2000).

5 K. Hori, "4 Microphones Power Advanced, 3-Dimensional Sound Intensity Measuring System," Journal of Electronic Engineering, Vol. 31, No. 326, (1994) pp. 47-49.

6 P. M. Morse and K. U. Ingard, Theoretical Acoustics, (McGraw- Hill Book Company, New York, 1968), pg. 419.

7 L. E. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders, Fundamentals of Acoustics, 4th ed. (Wiley, New York, 2000) pg 129.

8 www.gras.dk July 28th, 2004.

## Appendix

The following is the Matlab code used in this project.

```
% AVPLOT    actual velocity plot

%    AVPLOT('r,A') plots the actual velocity at a location r, given a
%    source strength of A. Assumes perfect microphones and a monopole
%    sourece at the origin.
%
%    This function is designed to help me understand what is really happening
%    in terms of velocity at a point, as a function of frequency. It uses the
%    velocity function that computes the actual velocity at a point, as a
%    function of frequency. avplot(r,A)
function avplot(r,A)

rho=1.21; % density of air
c=343; % speed of sound
f=1:6300; % freqency vector
omega=2*pi*f; % angular freqency
k=omega/c; % wave number vector
iterations=1;
for a=1:iterations
   v(:,:,a)=velocity(r(:,a),k,A); % velocity at r, as a function of frequency
   figure
   plot(f,v(1,:,a),'-<',f,v(2,:,a),'-v',f,v(3,:,a),'-.')
   legend
   title('Velocity as a function of frequency')
   xlabel(['freqency',' r: ' num2str(r(:,a)')])
   legend('X velocity','Y velocity','Z velocity')

end
```

```
% This program draws circles
clc
close all
clear

%%%%%%%%%%%% two microphones--- six mic representation
%Plot x and y axis and NO circle
% x=-1:.01:1;
% y=sqrt(1-x.^2);
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
subplot(3,2,1)
% plot(x,y,x,-y,'b-',lx,0,'b-',0,ly,'b-','LineWidth',2)
plot(lx,0,'b-',0,ly,'b-','LineWidth',2)
axis([-1.5,1.5,-1.5,1.5]);axis square
hold on

% Plot line y=2x, this is the 'theta' line
x=-.75:.01:.75;
y=2*x;
plot(x,y,'b-','LineWidth',2)

% Plot the microphones

plot(1/sqrt(5),2/sqrt(5),'MarkerSize',24,'Marker','o')
plot(-1/sqrt(5),-2/sqrt(5),'MarkerSize',24,'Marker','o')
title('Two Point Sensors')
ylabel('Six Microphone Probe')

% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)

hold off
subplot(3,2,2)

%Plot x and y axis and circle
x=-1:.01:1;
y=sqrt(1-x.^2);
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
plot(x,y,x,-y,'b-',lx,0,'b-',0,ly,'b-','LineWidth',2)
title('Two Point Spherical Sensors')
hold on
axis([-1.5,1.5,-1.5,1.5]);axis square

% Plot line y=2x, this is the 'theta' line
x=-.75:.01:.75;
y=2*x;
plot(x,y,'b-','LineWidth',2)

% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)
```

```
% Plot the microphones
plot(1/sqrt(5),2/sqrt(5),'MarkerSize',24,'Marker','o')
plot(-1/sqrt(5),-2/sqrt(5),'MarkerSize',24,'Marker','o')

%%%%%%%%%%%%%% Now do the tetrahedron
subplot(3,2,3)

%Plot x and y axis and NO circle
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
% plot(x,y,x,-y,'b-',lx,0,'b-',0,ly,'b-','LineWidth',2)
plot(lx,0,'b-',0,ly,'b-','LineWidth',2)
hold on
% Plot line y=2x, this is the 'theta' line
x=-.75:.0001:.75;
y=2*x;
plot(x,y,'b-','LineWidth',2)
axis([-1.5,1.5,-1.5,1.5]);axis square

% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)

% Plot the first microphone
s1=tan(atan(.5)+pi/6)^-1; % s stands for slope
ypt1=sqrt(s1^2/(s1^2+1));
xpt1=sqrt(1/(1+s1^2))
x=0:.0001:xpt1;
y=s1*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt1,ypt1,'MarkerSize',24,'Marker','o')

% Plot the second microphone
s2=tan(atan(.5)+5*pi/6)^-1; % s stands for slope
ypt2=-sqrt(s2^2/(s2^2+1));
xpt2=sqrt(1/(1+s2^2))
x=0:.0001:xpt2;
y=s2*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt2,ypt2,'MarkerSize',24,'Marker','o')

b=ypt2-2*xpt2
x=xpt2:.0001:xpt1;
y=2*x+b;
plot(x,y,'b-','LineWidth',2)

ylabel('Tetrahdron Probe')

% Sphere Version
hold off
subplot(3,2,4)

%Plot x and y axis and circle
x=-1:.01:1;
```

```
y=sqrt(1-x.^2);
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
plot(x,y,x,-y,'b-',lx,0,'b-',0,ly,'b-','LineWidth',2)
axis([-1.5,1.5,-1.5,1.5]);axis square

hold on

% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)

% Plot line y=2x, this is the 'theta' line
x=-.75:.01:.75;
y=2*x;
plot(x,y,'b-','LineWidth',2)

% Plot the first microphone
s1=tan(atan(.5)+pi/6)^-1; % s stands for slope
ypt1=sqrt(s1^2/(s1^2+1));
xpt1=sqrt(1/(1+s1^2))
x=0:.0001:xpt1;
y=s1*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt1,ypt1,'MarkerSize',24,'Marker','o')

% Plot the second microphone
s2=tan(atan(.5)+5*pi/6)^-1; % s stands for slope
ypt2=-sqrt(s2^2/(s2^2+1));
xpt2=sqrt(1/(1+s2^2))
x=0:.0001:xpt2;
y=s2*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt2,ypt2,'MarkerSize',24,'Marker','o')

b=ypt2-2*xpt2
x=xpt2:.0001:xpt1;
y=2*x+b;
plot(x,y,'b-','LineWidth',2)

%%%%%%%%%%%%%% Now do the Orthogonal Probe

subplot(3,2,5)

%Plot x and y axis and NO circle
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
plot(lx,0,'b-',0,ly,'b-','LineWidth',2)
axis([-1.5,1.5,-1.5,1.5]);axis square

hold on
% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)
```

```
% Plot line y=2x, this is the 'theta' line
x=-.75:.0001:.75;
y=2*x;
plot(x,y,'b-','LineWidth',2)

% Plot the first microphone
s1=tan(atan(.5)+.9553)^-1; % s stands for slope
ypt1=sqrt(s1^2/(s1^2+1));
xpt1=sqrt(1/(1+s1^2))
x=0:.0001:xpt1;
y=s1*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt1,ypt1,'MarkerSize',24,'Marker','o')

% Plot the second microphone
s2=tan(atan(.5)+2.1863)^-1; % s stands for slope
ypt2=-sqrt(s2^2/(s2^2+1));
xpt2=sqrt(1/(1+s2^2))
x=0:.0001:xpt2;
y=s2*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt2,ypt2,'MarkerSize',24,'Marker','o')


b=ypt2-2*xpt2
x=xpt2:.0001:xpt1;
y=2*x+b;
plot(x,y,'b-','LineWidth',2)

ylabel('Orthogonal Probe')


% Sphere Version
hold off
subplot(3,2,6)


%Plot x and y axis and circle
x=-1:.01:1;
y=sqrt(1-x.^2);
lx=-1.5:.01:1.5;
ly=-1.5:.01:1.5;
plot(x,y,x,-y,'b-',lx,0,'b-',0,ly,'b-','LineWidth',2)
axis([-1.5,1.5,-1.5,1.5])
axis([-1.5,1.5,-1.5,1.5]);axis square

hold on
% plot the angle theta
x=0:.0001:1/(sqrt(20));
y=sqrt(.25-x.^2);
plot(x,y,'b-','Linewidth',2)

% Plot line y=2x, this is the 'theta' line
x=-.75:.01:.75;
y=2*x;
```

```
plot(x,y,'b-','LineWidth',2)

% Plot the first microphone
s1=tan(atan(.5)+.9553)^-1; % s stands for slope
ypt1=sqrt(s1^2/(s1^2+1));
xpt1=sqrt(1/(1+s1^2))
x=0:.0001:xpt1;
y=s1*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt1,ypt1,'MarkerSize',24,'Marker','o')

% Plot the second microphone
s2=tan(atan(.5)+2.1863)^-1; % s stands for slope
ypt2=-sqrt(s2^2/(s2^2+1));
xpt2=sqrt(1/(1+s2^2))
x=0:.0001:xpt2;
y=s2*x;
plot(x,y,'b-','LineWidth',2)
plot(xpt2,ypt2,'MarkerSize',24,'Marker','o')

b=ypt2-2*xpt2
x=xpt2:.0001:xpt1;
y=2*x+b;
plot(x,y,'b-','LineWidth',2)
```

```
function DSSed
% DSSED  DSS Energy Density
%    [varargout] = DSSED imports DSS data via text files. This function
%    will read a directory and process all text files. Files are processed
%    by calling DSSsix, DSStetra, and DSSortho. Energy Density values are
%    plotted, along with pressure, velocity, and intensity.

clear
clc
close all
c=343; % speed of sound

d=dir; % setup an array of structs that has info on each item in the directory
numfiles=length(d); % each item in the directory must be checked
for index=3:numfiles % the first 2 elements are '.' and '..', which is directory stuff, so skip it
   filename=upper(d(index).name); % get the filename of the first item in the directory, put name in
UPPER case

   if findstr(filename,'TXT')
      [probe,p1,p2,p3,p4,p5,p6,k,radius]=Pextract(filename);
      switch probe
         case 'Six'
            E=DSSsix(p1,p2,p3,p4,p5,p6,k,radius);
         case 'Tetra'
            E=DSStetra(p1,p2,p3,p4,k,radius);
         case 'Ortho'
            E=DSSortho(p1,p2,p3,p4,k,radius);
         otherwise
            disp('Do not know what type of probe it is')
            return
      end
      figure
      maxfig
%        hold on
      f=k*c/(2*pi);
      H=plot(f(50:length(k)),E(50:length(k)),'r-o');
      set(H,'linewidth',2);
      H=title(['Energy Density vs. Frequency, ',probe,' Probe, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
      xlabel('Frequency','fontsize',16);
      ylabel('Energy Density','fontsize',16);
      H=legend(probe);
      set(H,'fontsize',14);
   end % end if
end
```

```
% function DSSedA
% DSSEDA  DSS Energy Density Average
%    [varargout] = DSSEDA imports DSS data via text files. This function
%    will read a directory and process all text files. Files are processed
%    by calling DSSsix, DSStetra, and DSSortho. Energy Density values are
%    plotted, along with pressure, velocity, and intensity.

clear
clc
close all
c=343; % speed of sound
plotflag=0;
d=cd; % get the current path information
path(path,d) % append the current path to the path, so that Matlab can see all my functions...
CodeDir=cd % store the current directory, so Matlab can return to it...
datadir=('U:\NASA\NRG Density Project\dss data\Small_Six_2000Hz')
disp(['MOVING TO THIS DIRECTORY!!!!', datadir])
cd(datadir)
d=dir('*.txt'); % setup an array of structs that has info on each txt item in the directory
numfiles=length(d); % each item in the directory must be checked


for index=1:numfiles
   filename=upper(d(index).name); % get the filename of the first item in the directory, put name in
UPPER case

[probe,p1(:,index),p2(:,index),p3(:,index),p4(:,index),p5(:,index),p6(:,index),k,radius(index)]=Pextract(file
name);

   switch probe
      case 'Six'

E(:,index)=DSSsix(p1(:,index),p2(:,index),p3(:,index),p4(:,index),p5(:,index),p6(:,index),k,radius(index),pl
otflag);

         f=k*c/(2*pi);
         figure;(plot(f,20*log10(abs(p1(:,index))/20e-6),f,20*log10(abs(p2(:,index))/20e-
6),f,20*log10(abs(p3(:,index))/20e-6),f,20*log10(abs(p4(:,index))/20e-6)))
         title(filename)

      case 'Tetra'
         E(:,index)=DSStetra(p1(:,index),p2(:,index),p3(:,index),p4(:,index),k,radius(index),plotflag);

         f=k*c/(2*pi);
         figure;(plot(f,20*log10(abs(p1(:,index))/20e-6),f,20*log10(abs(p2(:,index))/20e-
6),f,20*log10(abs(p3(:,index))/20e-6),f,20*log10(abs(p4(:,index))/20e-6)))
         title(filename)

      case 'Ortho'

         [Enew(:,index),
E(:,index)]=DSSortho(p1(:,index),p2(:,index),p3(:,index),p4(:,index),k,radius(index),plotflag);
%          Etemp(:,index)=10*log10(E(:,index)/3e-15); % convert to dB
%          f=k*c/(2*pi);
%          figure;
%          plot(f(50:length(f)),Etemp(50:length(f),index),'b','linewidth',1.5);
```

```
%           index
%           f=k*c/(2*pi);
%
%           figure;(plot(f,20*log10(abs(p1(:,index))/20e-6),f,20*log10(abs(p2(:,index))/20e-
6),f,20*log10(abs(p3(:,index))/20e-6),f,20*log10(abs(p4(:,index))/20e-6)))
%           title(filename)

      otherwise
         disp('Do not know what type of probe it is')
         return
   end

end



% now determine where the breaks in the directory are
A=0;
B=0;
C=0;
R=0;
S=0;
for fileindex=1:length(d)
   switch upper(d(fileindex).name(1))
      case 'A'
         A=A+1;
      case 'B'
         B=B+1;
      case 'C'
         C=C+1;
      case 'R'
         R=R+1;
   end

end

% if findstr(probe,'Ortho')
%    figure
%    plot(10*log10(abs(Enew)/3e-15))
%    title('NRG of all runs')
%    figure;plot(mean(10*log10(abs(Enew)/3e-15),2))
%    title('NRG average of all the runs-- new estimation method')
%
%    figure
%    plot(k*c/2/pi,10*log10((Enew)/3e-15))
%    title('Energy based on new method')
%
%    % figure; plot(k*c/2/pi,10*log10((Enew)/3e-15))
%    % title('
%
%    % plot(k*c/2/pi,10*log10((Enew)/3e-15))
%
%    figure;plot(10*log10(mean(abs(Enew)/3e-15,2)))
%    title('Energy average-- abs-- new method')
%    % figure;plot(10*log10(mean(abs(Enew)/3e-15,2)))
%
```

```
% end % end if


% Take the average of E
E=10*log10(E/3e-15); % convert to dB
EA=mean(E(:,1:A),2); % position A
EB=mean(E(:,A+1:A+B),2); % position B
EC=mean(E(:,A+B+1:A+B+C),2); % position C
ER=mean(E(:,A+B+C+1:A+B+C+R),2); % position R

b=1:4;
bplot(1)=EA(501);
bplot(2)=EB(501);
bplot(3)=EC(501);
bplot(4)=ER(501);

% create a string, either Large or Small, for use in the title
if radius(1) == .0254
   size=' Large ';
else
   size=' Small ';
end

if A~=0
   % plot Average Energy at Position A
   figure
   maxfig
   f=k*c/(2*pi);
   H=plot(f(50:length(k)),EA(50:length(k)),'r-o');
   %     set(H,'linewidth',2);
   switch probe
      case 'Six'
         H=title(['Energy Density at Position A, ',size,probe,' Microphone Probe, ', num2str(A), '
averages'],'fontsize',16);
      case 'Tetra'
         H=title(['Energy Density at Position A, ',size,probe,'hedron Probe, ', num2str(A), '
averages'],'fontsize',16);
      case 'Ortho'
         H=title(['Energy Density at Position A, ',size,probe,'gonal Probe, ', num2str(A), '
averages'],'fontsize',16);
   end

   xlabel('Frequency','fontsize',16);
   ylabel('Energy Density in dB','fontsize',16);
   set(gca,'FontSize',14)

   % H=legend(probe);
   % set(H,'fontsize',14);
   hold on
end %end if

if B~=0
   % plot Average Energy at Position B
   figure
   maxfig
   f=k*c/(2*pi);
```

```
   H=plot(f(50:length(k)),EB(50:length(k)),'b-o');
   %     set(H,'linewidth',2);
   switch probe
      case 'Six'
         H=title(['Energy Density at Position B, ',size,probe,' Microphone Probe, ', num2str(B), '
averages'],'fontsize',16);
      case 'Tetra'
         H=title(['Energy Density at Position B, ',size,probe,'hedron Probe, ', num2str(B), '
averages'],'fontsize',16);
      case 'Ortho'
         H=title(['Energy Density at Position B, ',size,probe,'gonal Probe, ', num2str(B), '
averages'],'fontsize',16);
   end
   xlabel('Frequency','fontsize',16);
   ylabel('Energy Density in dB','fontsize',16);
   set(gca,'FontSize',14)

   % H=legend(probe);
   % set(H,'fontsize',14);
end %end if

if C~=0
   % plot Average Energy at Position C
   figure
   maxfig
   f=k*c/(2*pi);
   H=plot(f(50:length(k)),EC(50:length(k)),'m-o');
   %     set(H,'linewidth',2);
   switch probe
      case 'Six'
         H=title(['Energy Density at Position C, ',size,probe,' Microphone Probe, ', num2str(C), '
averages'],'fontsize',16);
      case 'Tetra'
         H=title(['Energy Density at Position C, ',size,probe,'hedron Probe, ', num2str(C), '
averages'],'fontsize',16);
      case 'Ortho'
         H=title(['Energy Density at Position C, ',size,probe,'gonal Probe, ', num2str(C), '
averages'],'fontsize',16);
   end
   xlabel('Frequency','fontsize',16);
   ylabel('Energy Density in dB','fontsize',16);
   set(gca,'FontSize',14)

   % H=legend(probe);
   % set(H,'fontsize',14);
end % end if

if R~=0
   % plot Average Energy at Position R
   figure
   maxfig
   f=k*c/(2*pi);
   H=plot(f(50:length(k)),ER(50:length(k)),'k-o');
   %     set(H,'linewidth',2);
   switch probe
      case 'Six'
```

```
        H=title(['Energy Density at Position R, ',size,probe,' Microphone Probe, ', num2str(R), '
averages'],'fontsize',16);
      case 'Tetra'
        H=title(['Energy Density at Position R, ',size,probe,'hedron Probe, ', num2str(R), '
averages'],'fontsize',16);
      case 'Ortho'
        H=title(['Energy Density at Position R, ',size,probe,'gonal Probe, ', num2str(R), '
averages'],'fontsize',16);
   end
   xlabel('Frequency','fontsize',16);
   ylabel('Energy Density in dB','fontsize',16);
   set(gca,'FontSize',14)

end % end if

figure
bar(bplot)
xlabel('Position','fontsize',16);
ylabel('Energy Density in dB','fontsize',16);
% set(H,'fontsize',14);
switch probe
   case 'Six'
      H=title(['Energy Density Measurement, ',size,probe,' Microphone Probe, ', num2str(min([A,B,C,R])), '
averages'],'fontsize',16);
   case 'Tetra'
      H=title(['Energy Density Measurement, ',size,probe,'hedron Probe, ', num2str(min([A,B,C,R])), '
averages'],'fontsize',16);
   case 'Ortho'
      H=title(['Energy Density Measurement, ',size,probe,'gonal Probe, ', num2str(min([A,B,C,R])), '
averages'],'fontsize',16);
end
set(gca,'XTickLabel',['A';'B';'C';'R'])
set(gca,'FontSize',14)
% % Plot the average energy of everything...
% figure
% maxfig
% plot(f(50:length(k)),10*log10(mean(E(50:length(k)))/10e-12))
% xlabel('Frequency','fontsize',16);
% ylabel('Energy Density in dB','fontsize',16);
% title('Average energy density, for all the locations...')
% figure;(plot(f,20*log10(abs(p1)/20e-6),f,20*log10(abs(p2)/20e-6),f,20*log10(abs(p3)/20e-
6),f,20*log10(abs(p4)/20e-6)))
CodeDir
cd(CodeDir) % move back to old directory
```

```
function [varargout]=DSSortho(p1,p2,p3,p4,k,radius,plotflag)
%   DSSORTHO  Energy Density based on DSS ORTHO microphone probe data.
%     [Energy] = DSSORTHO(p1,p2,p3,p4,radius,plotflag) determines energy
%     density based on the pressures measured by the Energy Density Probe.
%     Default is to show plots. If you don't want to see plots, set plotflag=0.

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 6 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 7 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSORTHO')
      return
end


rho=1.21; % density of air
c=343; % speed of sound

p=(p1+p2+p3+p4)/4; % the pressure estimate

% figure
% plot(k*c/2/pi,20*log10(abs(p1)/20e-6))
% title('p1')
% figure
% plot(k*c/2/pi,20*log10(abs(p2)/20e-6))
% title('p2')
% figure
% plot(k*c/2/pi,20*log10(abs(p3)/20e-6))
% title('p3')
% figure
% plot(k*c/2/pi,20*log10(abs(p4)/20e-6))
% title('p4')
% figure
% plot(k*c/2/pi,20*log10(abs(p1+p2+p3+p4)/4/20e-6))
% title('Average of p1-4')

U=p.*conj(p)/(2*rho*c^2); % estimate potential energy
% figure
% plot(k*c/2/pi,10*log10(U/3e-15))
% title('p p* /(2*rho*c)-- DSS ortho')

[Vx,Vy,Vz,E]=DSSvortho(p1,p2,p3,p4,k,radius,plotflag); % estimate the velocity

Tx=rho*(Vx.*conj(Vx))/2;
Ty=rho*(Vy.*conj(Vy))/2;
Tz=rho*(Vz.*conj(Vz))/2;
T=Tx+Ty+Tz;

% figure
% plot(k*c/2/pi,10*log10(T/3e-15))
% title('T -- DSS ortho')
%
```

```
% figure
% plot(k*c/2/pi,10*log10(Tx/3e-15))
% title('Tx -- DSS ortho')
%
% figure
% plot(k*c/2/pi,10*log10(Ty/3e-15))
% title('Ty -- DSS ortho')
%
% figure
% plot(k*c/2/pi,10*log10(Tz/3e-15))
% title('Tz -- DSS ortho')


nao=nargout; % number arguments outputed
if nao~=0
   varargout={E,T+U};
% varargout={K};
end

if plotflag ~=0 % if the plot flag is anything other than zero...
   p=20*log10(abs(p)/20e-6); % convert pressure into SPL
   f=k*c/(2*pi);
   figure % Plot Pressure
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(p(100:length(k))),'b-d'); % diamond
   set(H,'linewidth',2);
   H=title(['Pressure Magnitude vs. Frequency, Orthogonal Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('dB SPL','fontsize',16);
   H=legend('Pressure');
   set(H,'fontsize',14);
   hold off
end % end if
```

```
% function DSSpspectrum
% DSSPSPECTRUM  DSS Pressure Spectrum
%    [varargout] = DSSPSPECTRUM imports DSS data via text files. This function
%    will read a directory and process all text files. Files are processed
%    by calling DSSsix, DSStetra, and DSSortho. Plots the pressure spectrum
%    of the individual microphones

clear
clc
close all

ORIENT('LANDSCAPE');
close(gcf)

c=343; % speed of sound
plotflag=0; % plotflag==1 means plot the pressure and velocity
d=cd; % get the current path information
path(path,d) % append the current path to the path, so that Matlab can see all my functions...
datadir=('U:\NASA\NRG Density Project\dss data\Diagnostics')
disp(['MOVING TO THIS DIRECTORY!!!!', datadir])
cd(datadir)
d=dir('*.txt'); % setup an array of structs that has info on each txt item in the directory
numfiles=length(d); % each item in the directory must be checked


for index=1:numfiles
   filename=upper(d(index).name); % get the filename of the first item in the directory, put name in
UPPER case

[probe,p1(:,index),p2(:,index),p3(:,index),p4(:,index),p5(:,index),p6(:,index),k,radius(index)]=Pextract(file
name);
   figure
   H=plot(k*radius(index),20*log10(abs(p1(:,index)/2e-
5)),'r.',k*radius(index),20*log10(abs(p2(:,index)/2e-5)),'b-.',...
      k*radius(index),20*log10(abs(p3(:,index)/2e-5)),'g-',k*radius(index),20*log10(abs(p4(:,index)/2e-
5)),'y--');
   title(filename)
   legend('p1','p2','p3','p4')

   %    set(H,'linewidth',1.5);
   %    print


end


%%%%%%% average for all locations

figure
plot(abs(mean(20*log10(p1/2e-5),2)),'r-v')
title('average of mic 1')

figure
plot(abs(mean(20*log10(p2/2e-5),2)),'b-*')
title('average of mic 2')
```

```
figure
plot(abs(mean(20*log10(p3/2e-5),2)),'g->')
title('average of mic 3')

figure
plot(abs(mean(20*log10(p4/2e-5),2)),'y-*')
title('average of mic 4')

% figure
% plot(abs(mean(20*log10(p5/2e-5),2)),'k-<')
% title('average of mic 5')
%
% figure
% plot(abs(mean(20*log10(p6/2e-5),2)),'m-*')
% title('average of mic 6')

p1=mean(p1,2);
p2=mean(p2,2);
p3=mean(p3,2);
p4=mean(p4,2);
p5=mean(p5,2);
p6=mean(p6,2);
switch probe
   case 'Six'
      E(index,:)=DSSsix(p1,p2,p3,p4,p5,p6,k,radius(index),plotflag)';
   case 'Tetra'
      E(index,:)=DSStetra(p1,p2,p3,p4,k,radius(index),plotflag)';
   case 'Ortho'
      E(index,:)=DSSortho(p1,p2,p3,p4,k,radius(index),plotflag)';
   otherwise
      disp('Do not know what type of probe it is')
      return
end
```

```
function [varargout]=DSSsix(p1,p2,p3,p4,p5,p6,k,radius,plotflag)
%   DSSSIX  Energy Density based on DSS six microphone probe data.
%     [varargout] = DSSSIX(p1,p2,p3,p4,p5,p6,radius,plotflag) determines energy
%     density based on the pressures measured by the Energy Density Probe.
%     Default is to plot the velocity. If you don't want to plot pressure,
%     set plotflag=0.

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 8 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 9 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSsix')
      return
end

rho=1.21; % density of air
c=343; % speed of sound

p=(p1+p2+p3+p4+p5+p6)/6; % the pressure estimate
U=p.*conj(p)/(2*rho*c^2); % estimate potential energy

[Vx,Vy,Vz]=DSSvsix(p1,p2,p3,p4,p5,p6,k,radius,plotflag); % estimate the velocity
Tx=rho*(Vx.*conj(Vx))/2;
Ty=rho*(Vy.*conj(Vy))/2;
Tz=rho*(Vz.*conj(Vz))/2;
T=Tx+Ty+Tz;

nao=nargout; % number arguments outputed
if nao~=0
   varargout={T+U};
end

if plotflag~=0 % plot unless plotflag==0
   p=20*log10(abs(p)/20e-6); % convert pressure into SPL
   f=k*c/(2*pi);
   figure % Plot Pressure
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(p(100:length(k))),'b-d'); % diamond
   set(H,'linewidth',2);
   H=title(['Pressure Magnitude vs. Frequency, Six Microphone Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('dB SPL','fontsize',16);
   H=legend('Pressure');
   set(H,'fontsize',14);
   hold off
end % end if
```

```
function [varargout]=DSStetra(p1,p2,p3,p4,k,radius,plotflag)
%  DSSTETRA  Energy Density based on DSS Tetrahedron microphone probe data.
%    [varargout] = DSSTETRA(p1,p2,p3,p4,radius,plotflag) determines energy
%    density based on the pressures measured by the Energy Density Probe.
%    If you don't want to see plots, set plotflag=0.

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 6 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 7 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSTETRA')
      return
end

rho=1.21; % density of air
c=343; % speed of sound

p=(p1+p2+p3+p4)/4; % the pressure estimate
U=p.*conj(p)/(2*rho*c^2); % estimate potential energy

[Vx,Vy,Vz]=DSSvtetra(p1,p2,p3,p4,k,radius,plotflag); % estimate the velocity
Tx=rho*(Vx.*conj(Vx))/2;
Ty=rho*(Vy.*conj(Vy))/2;
Tz=rho*(Vz.*conj(Vz))/2;
T=Tx+Ty+Tz;

nao=nargout; % number arguments outputed
if nao~=0
   varargout={T+U};
end

if plotflag~=0
   p=20*log10(abs(p)/20e-6); % convert pressure into SPL
   f=k*c/(2*pi);
   figure % Plot Pressure
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(p(100:length(k))),'b-d'); % diamond
   set(H,'linewidth',2);
   H=title(['Pressure Magnitude vs. Frequency, Tetrahedron Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('dB SPL','fontsize',16);
   H=legend('Pressure');
   set(H,'fontsize',14);
   hold off
end % end if
```

```
function [Vx,Vy,Vz,E]=DSSvortho(p1,p2,p3,p4,k,radius,plotflag)
%DSSVORTHO   DSS Velocity Ortho.
%   [Vx; Vy; Vz] = DSSVORTHO(p1,p2,p3,p4,k,radius,plotflag) returns the velocity as
%   measured by the DSS with an orthogonal probe. Default is to plot the
%   velocity. If you don't want to see the velocity plot, set plotflag=0.

warning off MATLAB:divideByZero

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 6 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 7 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSvortho')
      return
end

rho=1.21; % density of air
c=343; % speed of sound
omega=k*c;

r1=[+radius/sqrt(3);-radius/sqrt(3);-radius/sqrt(3)]; % mic1 position
r2=[-radius/sqrt(3);+radius/sqrt(3);-radius/sqrt(3)]; % mic2 position
r3=[-radius/sqrt(3);-radius/sqrt(3);-radius/sqrt(3)]; % mic3 position
r4=[-radius/sqrt(3);-radius/sqrt(3);+radius/sqrt(3)]; % mic4 position

% determine Auto and Cross Spectra....

S11=mean(p1.*conj(p1),2);
S22=mean(p2.*conj(p2),2);
S33=mean(p3.*conj(p3),2);
S44=mean(p4.*conj(p4),2);
S12=mean(p1.*conj(p2),2);
S21=mean(p2.*conj(p1),2);
S13=mean(p1.*conj(p3),2);
S14=mean(p1.*conj(p4),2);
S41=mean(p4.*conj(p1),2);
S23=mean(p2.*conj(p3),2);
S32=mean(p3.*conj(p2),2);
S24=mean(p2.*conj(p4),2);
S42=mean(p4.*conj(p2),2);
S34=mean(p3.*conj(p4),2);
S43=mean(p4.*conj(p3),2);

[value,index]=max(abs(p1)); % index now corresponds to the frequency bin that has the tone, should be
501 when frequency is 500 Hz.

%On Axis Velocity Values
for temp=1:min(size(p1))
   Vx13(:,temp)=(p1(:,temp)-p3(:,temp))./(j*rho.*omega*(r1(1)-r3(1))  ); % This vector points in the - x
direction
   Vy23(:,temp)=(p2(:,temp)-p3(:,temp))./(j*rho.*omega*(r2(2)-r3(2))  ); % This vector points in the - y
direction
```

```
   Vz43(:,temp)=(p4(:,temp)-p3(:,temp))./(j*rho.*omega*(r4(3)-r3(3))  ); % This vector points in the - z
direction
end
Vx31=-Vx13; % This vector points in the + x direction
Vy32=-Vy23; % This vector points in the + y direction
Vz34=-Vz43; % This vector points in the + z direction

% determine the phase angle of the on axis velocities
Vx31pha=angle(Vx31(index));
Vy32pha=angle(Vy32(index));
Vz34pha=angle(Vz34(index));

% In the XY plane

% difference in phase of Vx and Vy
phadiffxy=180/pi*(Vx31pha-Vy32pha);
theta=atan2(abs(Vy32(index)),abs(Vx31(index)));
% if the phase difference is between 90 and 270 degrees, change the sign of
% theta
   if ( abs(phadiffxy) > 85 & abs(phadiffxy) < 275)
      theta=-theta;
   end

for temp=1:min(size(p1))
   V12(:,temp)=(p1(:,temp)-p2(:,temp))./(j*rho.*omega*sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2)); % This
vector points in the + y - x direction
end
thetaa=3*pi/4-theta; % thetaa is the angle as measured from the diagonal. This gives the correct sign on the
diagonal

V12p=V12./cos(thetaa); % this line may not need the ./
Vy12=V12p.*sin(theta);
Vx12=V12p.*cos(theta);

for temp=1:min(size(p1))
   % In the XZ Plane
   V41(:,temp)=(p4(:,temp)-p1(:,temp))./(j*rho.*omega*sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2)); % This
vector points in the - z + x direction
end
V14=-V41;

% difference in phase of Vx and Vz
phadiffxz=180/pi*(Vx31pha-Vz34pha); % difference in phase of Vx and Vz
betaxz=atan2(abs(Vz34(index)),abs(Vx31(index))); % Vx31 points in the positive x direction, Vz34 points
in the positive z direction

% if the phase difference is between 90 and 270 degrees, change the sign of
% betaxz
if ( abs(phadiffxz) > 85 & abs(phadiffxz) < 275)
   betaxz=-betaxz;
end

betaa=3*pi/4-betaxz;
V14p=V14./cos(betaa);
Vx14=V14p.*cos(betaxz);
Vz14=V14p.*sin(betaxz);
```

```
   % In the YZ Plane
for temp=1:min(size(p1))
   V24(:,temp)=(p2(:,temp)-p4(:,temp))./(j*rho.*omega*sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2)); % This
vector points in the - y + z direction
end
% difference in phase of Vy and Vz
phadiffyz=180/pi*(Vy32pha-Vz34pha); % difference in phase of Vy and Vz
alphayz=atan2(abs(Vz34(index)),abs(Vy32(index)));

% if the phase difference is between 90 and 270 degrees, change the sign of
% alphayz
if (abs(phadiffyz) > 85 & abs(phadiffyz) < 275)
          alphayz=-alphayz;
       end

alphaa=3*pi/4-alphayz;

V24p=V24./cos(alphaa);
Vz24=V24p.*sin(alphayz);
Vy24=V24p.*cos(alphayz);

%Taylor Series expansion
Vx=(Vx12-Vx31+Vx14)/1.5;
Vy=(Vy12-Vy32+Vy24)/1.5;
Vz=(Vz14-Vz34+Vz24)/1.5;

% convert the velocity values to dB SVL
VxdB=20*log10(Vx/5e-8);
VydB=20*log10(Vy/5e-8);
VzdB=20*log10(Vz/5e-8);

 if plotflag~=0 % if the plot flag is anything but zero, plot
   f=k*c/(2*pi);
   figure % Plot Velocity Magnitude
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(VxdB(100:length(k))),'b-d'); % x axis is a diamond
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VydB(100:length(k))),'r-s'); % y axis is a square
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VzdB(100:length(k))),'y-v'); % z axis is a v
   set(H,'linewidth',2);
   H=title(['Velocity Magnitude vs. Frequency, Orthogonal Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('Particle Velocity in dB SVL re 5e-8 m/s ','fontsize',16);
   H=legend('Ortho X','Ortho Y','Ortho Z');
   set(H,'fontsize',14);
   hold off

end % end if
```

```
% see notes- 4/Jun/04
Ax=1./(rho.*omega*(r1(1)-r3(1))  );
Bx=cos(theta )./(rho.*omega.*(sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2)).*cos(thetaa) );
Cx=cos(betaxz)./(rho.*omega.*(sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2)).*cos(betaa)  );
Dx=mean(Bx+Cx,2)+Ax;
Bx=mean(Bx,2);
Cx=mean(Cx,2);
Kx=rho/(2*1.5*1.5)*((Dx.*Dx).*(S11)- 2*(Dx.*Bx).*real(S12) - 2*(Dx.*Ax).*real(S13) -
2*(Dx.*Cx).*real(S14)  ...
   +(Bx.*Bx).*(S22)+ 2*(Bx.*Ax).*real(S23) + 2*(Bx.*Cx).*real(S24)  ...
   +(Ax.*Ax).*(S33)+ 2*(Ax.*Cx).*real(S34)...
   +(Cx.*Cx).*(S44) );

Ay=1./(rho.*omega*(r2(2)-r3(2))  );
By=sin(theta)./(rho.*omega.*sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2).*cos(thetaa) );
Cy=cos(alphayz)./(rho.*omega.*(sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2)).*cos(alphaa) );
Dy=-By+Ay+Cy;
Ky=rho/(2*1.5*1.5)*((Dy.*Dy).*(S22) + 2*(Dy.*By).*real(S21) - 2*(Dy.*Ay).*real(S23) -
2*(Dy.*Cy).*real(S24)  ...
   +(By.*By).*(S11)- 2*(Ay.*By).*real(S13) - 2*(By.*Cy).*real(S14)  ...
   +(Ay.*Ay).*(S33)+ 2*(Ay.*Cy).*real(S34)...
   +(Cy.*Cy).*(S44) );

Az=1./(rho.*omega*(r4(3)-r3(3))  );
Bz=sin(betaxz)./(rho.*omega.*sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2).*cos(betaa) );
Cz=sin(alphayz)./(rho.*omega.*sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2).*cos(alphaa) );
Dz=-Bz+Az-Cz;
Kz=rho/(2*1.5*1.5)*((Dz.*Dz).*(S44) + 2*(Dz.*Bz).*real(S41) - 2*(Dz.*Az).*real(S43)
+2*(Dz.*Cz).*real(S42)  ...
   +(Bz.*Bz).*(S11)- 2*(Bz.*Az).*real(S13) + 2*(Bz.*Cz).*real(S12)  ...
   +(Az.*Az).*(S33)- 2*(Az.*Cz).*real(S32)...
   +(Cz.*Cz).*(S22) );

K=Kx+Ky+Kz;
if plotflag ~=0
   Vxnew=(Dx.*p1(:,1)-Bx.*p2(:,1)-Ax.*p3(:,1)-Cx.*p4(:,1))/1.5;
   figure
   plot(k*c/2/pi,20*log10(abs(Vxnew)/5e-8))
   title('Vx dB based on one estimate ref 5e-8 m/s')

   Vynew=(Dy.*p2(:,1)+By.*p1(:,1)-Ay.*p3(:,1)-Cy.*p4(:,1))/1.5;
   figure
   plot(k*c/2/pi,20*log10(abs(Vynew)/5e-8))
   title('Vy dB based on one estimate ref 5e-8 m/s')

   Vznew=(Dz.*p4(:,1)+Bz.*p1(:,1)-Az.*p3(:,1)+Cz.*p2(:,1))/1.5;
   figure
   plot(k*c/2/pi,20*log10(abs(Vznew)/5e-8))
   title('Vz dB based on one estimate ref 5e-8 m/s')



   figure
   plot(k*c/2/pi,10*log10(K/3e-15))
   title('K -- DSS vortho re 3e-15')
```

```
    figure
    plot(k*c/2/pi,10*log10(Kx/3e-15))
    title('Kx -- DSS vortho re 3e-15')

    figure
    plot(k*c/2/pi,10*log10(Ky/3e-15))
    title('Ky -- DSS vortho re 3e-15')

    figure
    plot(k*c/2/pi,10*log10(Kz/3e-15))
    title('Kz -- DSS vortho re 3e-15')
end

p=(p1+p2+p3+p4)/4;
p=mean(p,2);
U= p.*conj(p)/(2*rho*c^2);  % where U is the potential energy
% figure
% plot(k*c/2/pi,10*log10(U/3e-15))
% title('U=p p* /(2*rho*c^2) re 3e-15 -- DSS vortho')

E= K+U; % E is the total energy density, as a function of frequency
% figure
% plot(k*c/2/pi,10*log10(E/3e-15))
% title('K+U re 3e-15')

% figure
% plot(omega(1:length(omega))/(2*pi),10*log10(abs(K(1:length(omega))/3e-15)))
% title('K re 3e-15')
% plot(k*c/2/pi,10*log10((mean(p1+p2+p3+p4,2)/4-(p1+p2+p3+p4)/4)/3e-15))
```

```
function [varargout]=DSSvsix(p1,p2,p3,p4,p5,p6,k,radius,plotflag)
%DSSVSIX   DSS Velocity Six Microphone probe.
%   [Vx; Vy; Vz] = DSSVSIX(p1,p2,p3,p4,p5,p6,k,radius,plotflag) returns the velocity as
%   measured by the DSS with a six microphone probe.
%   Default is to plot the velocity. If you don't want to plot velocity,
%   set plotflag=0.


n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 8 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 9 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSvsix')
      return
end

warning off MATLAB:divideByZero
rho=1.21; % density of air in kg/m^3
c=343;
omega=k*c; % angular velocity

% determine the particle velocity at the center of the probe
Vx=(p2-p1)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - x direction
Vy=(p4-p3)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - y direction
Vz=(p6-p5)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - z direction

nao=nargout; % number arguments outputed
if nao~=0
   varargout={Vx;Vy;Vz};
end

% convert the velocity values to dB SVL
VxdB=20*log10(Vx/5e-8);
VydB=20*log10(Vy/5e-8);
VzdB=20*log10(Vz/5e-8);

if plotflag~=0
   f=k*c/(2*pi);
   figure % Plot Velocity Magnitude
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(VxdB(100:length(k))),'b-d'); % x axis is a diamond
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VydB(100:length(k))),'r-s'); % y axis is a square
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VzdB(100:length(k))),'y-v'); % z axis is a v
   set(H,'linewidth',2);
   H=title(['Velocity Magnitude vs. Frequency, Six Microphone Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('Particle Velocity in dB SVL','fontsize',16);
   H=legend('Six X','Six Y','Six Z');
```

```
    set(H,'fontsize',14);
    hold off
end % end if
```

```
function [Vx,Vy,Vz]=DSSvtera(p1,p2,p3,p4,k,radius,plotflag)
%DSSVTETRA   DSS Velocity Tetrahedron Probe
%   [Vx, Vy, Vz] = DSSVTETRA(p1,p2,p3,p4,k,radius,plotflag) returns the velocity as
%   measured by the DSS with a tetrahedron probe. Default is to plot the
%   velocity. If you don't want to plot velocity, set plotflag=0.

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 6 % no argument for plot flag passed in, assume default is to plot
      plotflag=1;
   case 7 % plot flag argument passed in
      % don't actually do anything
   otherwise
      disp('Error-- wrong number of input arguments DSSTETRA')
      return
end

warning off MATLAB:divideByZero

rho=1.21; % density of air
c=343; % speed of sound
omega=k*c;

r1=[0                ; -2/3*radius*2^(1/2)       ;-1/3*radius];
r2=[-1/3*radius*6^(1/2); +1/9*3^(1/2)*radius*6^(1/2);-1/3*radius];
r3=[+1/3*radius*6^(1/2); +1/9*3^(1/2)*radius*6^(1/2);-1/3*radius];
r4=[0                ; 0                      ;+radius];

d=sqrt(sum((r1-r2).^2)); % seperation distance between microphones

C1=-1/2;
C2=-1/2;
C3=-1/2;
Vx=C1*mean(-2*p2+2*p3,2)./(j*omega*rho*d);
Vy=C2*mean(4*p1/sqrt(3)-2*p2/sqrt(3)-2*p3/sqrt(3),2)./(j*omega*rho*d);
Vz=C3*mean(-sqrt(2/3)*p1-sqrt(2/3)*p2-sqrt(2/3)*p3+3*sqrt(2/3)*p4,2)./(j*omega*rho*d);

% convert the velocity values to dB SVL
VxdB=20*log10(Vx/5e-8);
VydB=20*log10(Vy/5e-8);
VzdB=20*log10(Vz/5e-8);


if plotflag~=0
   f=k*c/(2*pi);
   figure % Plot Velocity Magnitude
   maxfig
   hold on
   H=plot(f(100:length(k)),abs(VxdB(100:length(k))),'b-d'); % x axis is a diamond
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VydB(100:length(k))),'r-s'); % y axis is a square
   set(H,'linewidth',2);
   H=plot(f(100:length(k)),abs(VzdB(100:length(k))),'y-v'); % z axis is a v
   set(H,'linewidth',2);
```

```
   H=title(['Velocity Magnitude vs. Frequency, Tetrahedron Probe, radius ',num2str(radius), '
meters'],'fontsize',16);
   xlabel('Frequency','fontsize',16);
   ylabel('Particle Velocity in dB SVL','fontsize',16);
   H=legend('Tetra X','Tetra Y','Tetra Z');
   set(H,'fontsize',14);
   hold off
end % end if
```

```
% EACTUAL     compute the energy of orthogonal probe
%     EACTUAL(r,k,A) computes the energy at a position r, given k,
%     and source strength A. Assumes a monopole source radiating
%     into the free field. r must be a vector with in [x;y;z]. Output is
%     a scalar value.

function output=eactual(r,k,A)

c=343;
rho=1.21; % density of air


p=pressure(r,k,A);
U=p.*conj(p)/(2*rho*c^2); % estimate potential energy

v=velocity(r,k,A); % estimate the velocity
T=rho*v.*conj(v)/2; % estimate kinetic energy
T=T(1,:)+T(2,:)+T(3,:);

output= real(U+T);
```

```
function p=edfft(p)
% EDFFT     Energy Density FFT
%     p = EDFFT(p) is a simple FFT routine that scales the FFT and returns
%     the positive frequency components only.

p=fft(p)/length(p); % fft and scaled by length
p=p(1:length(p)/2); % take only the positive frequencies, get rid of negative frequencies
p(2:(length(p)-1))=2*p(2:(length(p)-1)); % double everything but DC
```

```
% EORTHO      estimate the energy of orthogonal probe
%     EORTHO(r,k,radius,A) estimates the energy at a position r, given k,
%     radius, and source strength A. Assumes a monopole source radiating
%     into the free field. r must be a vector with in [x;y;z]. Output is
%     the complex velocity in x, y, and z directions. NOTE- ka must be a
%     scalar value. It cannot be a vector. If ka is a vector, VEST will
%     return an error.


% vest(r,f,radius) This function estimates the velocity between all four microphones and returns the
velocity components between all four mics
% Inputs are the vector r, radius of sphere, and the wave number k. This function calls the pressure
function.
% Output is a vector length equal to the vector wave number k.
% Vector notation- vector starts at first digit, ends at second digit. So,
% U12 increases from mic 1 to mic 2 REMEBER, vest must accept only one ka
% value at a time. It does not work when ka is a vector

function output=eortho(r,k,radius,A)

c=343;
omega=k*c;
rho=1.21; % density of air

r1=[r(1)+radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic1 position
r2=[r(1)-radius/sqrt(3);r(2)+radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic2 position
r3=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic3 position
r4=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)+radius/sqrt(3)]; % mic 4 position

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

p=(p1+p2+p3+p4)/4;
U=p.*conj(p)/(2*rho*c^2); % estimate potential energy

v=vortho(r,k,radius,A); % estimate the velocity

T=rho*v.*conj(v)/2; % estimate kinetic energy
T=T(1)+T(2)+T(3);

output= real(U+T); % take the real part because there is a very small imaginary part that is an artifact... (e-
22)
```

```
% ESIX       free field energy estimate for six microphone probe
%     ESIX(r,k,radius,A) estimates the energy at the center of the
%     sphere based on Euler's Equation. It outputs a scalar.
%
% Six microphone technique for estimated 3d velocity. This script accepts
% inputs of r,f,radius- r being the location of the probe, f being the
% frequency of interest, and radius being the radius of the probe.

function output=esix(r,k,radius,A)

c=343; % speed of sound in air
rho=1.21; % density of air in kg/m^3
omega=k*c; % angular velocity

% determine the locations of the six microphones and the vectors that point
% to them
r1=r+[ radius;0;0;];
r2=r+[-radius;0;0;];
r3=r+[0; radius;0;];
r4=r+[0;-radius;0;];
r5=r+[0;0; radius;];
r6=r+[0;0;-radius;];

% Determine the pressure at each microphone
p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
p5=pressure(r5,k,A);
p6=pressure(r6,k,A);

p=(p1+p2+p3+p4+p5+p6)/6;
U=p.*conj(p)/(2*rho*c^2); % estimate potential energy

% determine the particle velocity at the center of the probe
Ux=(p2-p1)./(j*rho.*omega*(r1(1)-r2(1))); % This vector points in the - x direction
Uy=(p4-p3)./(j*rho.*omega*(r3(2)-r4(2))); % This vector points in the - y direction
Uz=(p6-p5)./(j*rho.*omega*(r5(3)-r6(3))); % This vector points in the - z direction
T=rho*(Ux.*conj(Ux)+Uy.*conj(Uy)+Uz.*conj(Uz))/2;

output=U+T;
```

```
% ETETRA      estimate the energy of tetrahedron probe
%    ETETRA(r,k,radius,A) estimates the acoustic energy at a position r, given k,
%    radius, and source strength A. Assumes a monopole source radiating
%    into the free field. r must be a vector with in [x;y;z]. Output is
%    a scalar . NOTE- ka must be a scalar value. It cannot be a vector.
%    Velocity estimate based on Ono Sokki tetrahedron intensity probe.

% This function estimates the velocity between all four microphones and returns the velocity components
between all four mics
% Inputs are the vector r, radius of sphere, and the wave number k. This function calls the pressure
function.
% Output is a vector length equal to the vector wave number k.
% Vector notation- vector starts at first digit, ends at second digit. So,
% U12 increases from mic 1 to mic 2

function output=etetra(r,k,radius,A)

rho=1.21; % density of air in kg/m^3
c=343;
%%%%%%%% Error checking... see if ka is greater than a 1x1 matrix, if so,
%%%%%%%% break

if length(k)>1
   disp('Error! ka values in vest must be 1x1 only. Use a for loop.')
   disp('for temp=1:length(temp)')
   disp('     vest(r,ka(temp),radius,A)')
   disp('end')
   output=('ERROR! ka must be a scalar!');
   return;
end;


v=vtetra(r,k,radius,A); % estimate the velocity

T=rho*v.*conj(v)/2; % estimate kinetic energy
T=T(1)+T(2)+T(3);

r1=[r(1)                ; r(2)+2/3*radius*2^(1/2)       ; r(3)-1/3*radius];
r2=[r(1)-1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r3=[r(1)+1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r4=[r(1)                ; r(2)                          ; r(3)+radius];

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

p=(p1+p2+p3+p4)/4;

U= p.*conj(p)/(2*rho*c^2); % deterimine the potential energy

output= real(U+T);% take the real part because there is a very small imaginary part that is an artifact... (e-
22)
```

```
% FCTHREE   free field velocity comparison between the three probes
%     FCTHREE(r,k,radius,A) calls fvsix, vest, velocity, and vtetra to
%     determine the velocity at a given location r, for frequency f, of a
%     monopole source with strength A.
%
% this function compares the three different probe designs based on only
% the free field conditions. In other words, there are not spherical
% effects included in the matlab code.
% function [dberror_ortho, dberror_tetra, dberror_six,
deltaphase_ortho,deltaphase_tetra,deltaphase_six]=fcthree(r,f);

function [e_ortho, e_six, e_tetra, deltaphase_ortho,deltaphase_six,deltaphase_tetra]=fcthree(r,k,radius,A);

c=343;
omega=k*c;
warning off MATLAB:divideByZero
for temp=1:length(k)
   ortho=vortho(r,k(temp),radius,A);
end
six=vsix(r,k,radius,A);
tetra=vtetra(r,k,radius,A);
actual=velocity(r,k,A);

e_ortho=(abs(ortho)./abs(actual));
e_tetra=abs(tetra)./abs(actual);
e_six= (abs(six)./abs(actual));
deltaphase_ortho=abs(180/pi*(angle(actual)-angle(ortho))); % difference in phase in degrees
if deltaphase_ortho(1)>180
   deltaphase_ortho(1)=360-deltaphase_ortho(1);
end
if deltaphase_ortho(2)>180
   deltaphase_ortho(2)=360-deltaphase_ortho(2);
end
if deltaphase_ortho(3)>180
   deltaphase_ortho(3)=360-deltaphase_ortho(3);
end
deltaphase_tetra=abs(180/pi*(angle(actual)-angle(tetra))); % difference in phase in degrees
if deltaphase_tetra(1)>180
   deltaphase_tetra(1)=360-deltaphase_tetra(1);
end
if deltaphase_tetra(2)>180
   deltaphase_tetra(2)=360-deltaphase_tetra(2);
end
if deltaphase_tetra(3)>180
   deltaphase_tetra(3)=360-deltaphase_tetra(3);
end
deltaphase_six= abs(180/pi*(angle(actual)-angle(six)));   % difference in phase in degrees
if deltaphase_six(1)>180
   deltaphase_six(1)=360-deltaphase_six(1);
end
if deltaphase_six(2)>180
   deltaphase_six(2)=360-deltaphase_six(2);
end
if deltaphase_six(3)>180
   deltaphase_six(3)=360-deltaphase_six(3);
end
```

```
function [varargout]=FFE(iterations)
% FFE      Free Field Error
%    FFE(iterations) collects the pressure, velocity, intensity and energy error for
%    the various probe designs. FFE with no argument will estimate the errors of only one location
%    FFE(10) will estimate the
%    error of 10 random locations with the default of radius .00825 meters.
%    It calls the various FF error functions
%    and plots the results. It does not take into account the difference
%    in radius between the small six microphone probe and the other small
%    radius probes. The default probe radius is .00825.
clc
close all
n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      r=randn(3,1); % generate random vector
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=0;
%       r=[1.1909;1.1892;-0.0376];
      r=[+0.0376;-1.1909;1.1892;];

   case 1 % one argument passed in, number of iterations
      r=randn(3,iterations);
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=0;
   otherwise
      disp('Error-- too many input arguments')
      return
end
for index=1:iterations
   [Es(:,:,index),Et(:,:,index),Eo(:,:,index)] = FFEERROR(r(:,index),ka,radius,A(index),p); % free field
energy error
   [Ps(:,:,index),Pt(:,:,index),Po(:,:,index)] = FFPERROR(r(:,index),ka,radius,A(index),p); % free field
pressure error
   [Vs(:,:,index),Vt(:,:,index),Vo(:,:,index)] = FFVERROR(r(:,index),ka,radius,A(index),p); % free field
velocity error
   [Is(:,:,index),It(:,:,index),Io(:,:,index)] = FFIERROR(r(:,index),ka,radius,A(index),p); % free field
intensity error
end
Es=mean(Es,3);Et=mean(Et,3);Eo=mean(Eo,3); % energy mean
Ps=mean(Ps,3);Pt=mean(Pt,3);Po=mean(Po,3); % pressure mean
Vs=mean(Vs,3);Vt=mean(Vt,3);Vo=mean(Vo,3); % velocity mean
Is=mean(Is,3);It=mean(It,3);Io=mean(Io,3); % intensity mean

figure % plot Energy and Pressure Error
maxfig
H=plot(ka,Es,'m--d',ka,Ps,'m--s',ka,Et,'r-.d',ka,Pt,'r-.s',ka,Eo,'b:d',ka,Po,'b:s');
set(H,'linewidth',2);
```

```
H=legend('Energy Six Mic','Pressure Six Mic','Energy Tetra','Pressure Tetra','Energy Ortho','Pressure
Ortho',2);
set(H,'fontsize',14);
if iterations==1
   H=title(['Energy and Pressure Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Energy and Pressure Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);

figure % Plot velocity Error
maxfig
hold on
H=plot(ka,Vs(1,:),'m--d',ka,Vs(2,:),'m--s',ka,Vs(3,:),'m--v');
set(H,'linewidth',2);
H=plot(ka,Vt(1,:),'r-.d',ka,Vt(2,:),'r-.s',ka,Vt(3,:),'r-.v');
set(H,'linewidth',2);
H=plot(ka,Vo(1,:),'b:d',ka,Vo(2,:),'b:s',ka,Vo(3,:),'b:v');
set(H,'linewidth',2);

H=legend('Six Mic x','Six Mic y','Six Mic z',...
   'Tetra x','Tetra y','Tetra z',...
   'Ortho x','Ortho y','Ortho z',2);
set(H,'fontsize',14);

% tetrahedron error checking
if max(max(Vt))>5 % if the error is higher than 2 dB
   r % print r
   H=text(1,max(max(Vt))/2,num2str(r))
   set(H,'fontsize',12);
   print
end


if iterations==1

   H=title(['Velocity Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Velocity Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);

ylabel('Error in dB','fontsize',16);

figure % plot intensity error
maxfig
hold on
H=plot(ka,Is(1,:),'m--d',ka,Is(2,:),'m--s',ka,Is(3,:),'m--v');
set(H,'linewidth',2);
H=plot(ka,It(1,:),'r-.d',ka,It(2,:),'r-.s',ka,It(3,:),'r-.v');
set(H,'linewidth',2);
```

```
H=plot(ka,Io(1,:),'b:d',ka,Io(2,:),'b:s',ka,Io(3,:),'b:v');
set(H,'linewidth',2);
H=legend('Six Mic x','Six Mic y','Six Mic z',...
   'Tetra x','Tetra y','Tetra z',...
   'Ortho x','Ortho y','Ortho z',3);
set(H,'fontsize',14);
if iterations==1
   H=title(['Intensity Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Intensity Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);
```

```
%  Free Field Energy Density Function Help
%
%  Energy Functions.
%    eactual     - Calculate the actual energy at location r.
%    eortho      - Estimate the energy based on Orthogonal Probe.
%    esix        - Estimate the energy based on Six Microphone Probe.
%    etetra      - Estimate the energy based on Tetrahedron Probe.
%
%  Error Functions.
%    FFe         - Compare Pressure, Velocity, Intensity & Energy
%    FFeerror    - Plot free field energy error.
%    FFierror    - Plot free field intensity error.
%    FFperror    - Plot free field pressure error.
%    FFverror    - Plot free field velocity error.
%
%  Intensity Functions.
%    intensity   - Calculate actual intensity at a given vector location r.
%    iortho      - Estimate intensity based on Orthogonal Probe.
%    isix        - Estimate intensity based on Six Microphone Probe.
%    itetra      - Estimate intensity based on Tetrahedron Probe.
%
%  Plotting Functions.
%    avplot      - Actual Velocity Plot.
%    maxfig      - Maximize the current figure.
%    orthoplot3  - Plot a 3-d sphere with microphones according to the orthogonal probe.
%    pplot       - Plot all three probes in the same figure, animate.
%    sixplot3    - Plot a 3-d sphere with 6 microphones on the surface.
%    tetraplot3  - Plot a 3-d sphere with microphones at locations given by Ono Sokki.
%
%  Pressure Functions.
%    pressure    - Estimate the pressure for a given vector r.
%
%  Velocity Functions.
%    velocity    - Compute actual velocity at a given vector r.
%    vortho      - Estimate velocity based on orthogonal probe configuration.
%    vsix        - Estimate velocity at the center of sphere w/ six mics.
%    vtetra      - Estimate velocity based on Ono Sokki Tetrahedron.
%
%  Misc Functions.
%    fferror     - Determine Errors in Mag and Phase based on fcthree.
%    opmecheck   - Ortho Phase Magnitude error check.
%    printopt    - Set printer default to print Landscape Orientation.
%
%    See also SEedfun.

help('ffedfun');
```

```
function [varargout] = FFeerror(r,ka,radius,A,p)
% FFEERROR  free field energy error.
%    [Esix,Etetra,Eortho] = FFEERROR(r,ka,radius,A,p) collects the energy error for the various probe
designs.
%    FFEERROR then takes averages and standard deviations of the various
%    errors. It also plots the results.
%
%    FFEERROR with no argument will estimate the intensity error at one random location.
%    If one argument is passed in, example: FFEERROR(10), will estimate the
%    error of 10 random locations with the default of radius .00825 meters. If four
%    arguments are passed in, FFEERROR will estimate the intensity error at the
%    location r, given ka, radius, and source strength A. The fifth
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      r=randn(3,1); % generate random vector
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, number of iterations
      iterations=r;
      r=randn(3,iterations);
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 4
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   otherwise
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
end

rho=1.21; % density of air
c= 343; % speed of sound in air

for index=1:iterations
   ea(index,:)=eactual(r(:,index),k,A(index));
   for temp=1:length(k)
      eo(index,temp)=eortho(r(:,index),k(temp),radius,A(index));
      et(index,temp)=etetra(r(:,index),k(temp),radius,A(index));
   end
   es(index,:)=esix(r(:,index),k,radius,A(index));
end
```

```
ees=10*log10(mean(es./ea,1)); % energy error six
eet=10*log10(mean(et./ea,1)); % energy error tetra
eeo=10*log10(mean(eo./ea,1)); % energy error ortho

if p~=0
   figure
   maxfig

   H=plot(ka,ees,'m--s',ka,eet,'r-.v',ka,eeo,'b:d');
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Energy Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Energy Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);
   H=legend('Six Mic','Tetra','Ortho');
   set(H,'fontsize',14);
end
nao=nargout; % number arguments outputed
if nao~=0
   varargout={ees,eet,eeo};
end
```

```
% script to plot the error as a function of frequency

clc
close all
clear
warning off MATLAB:divideByZero

ka= .1:.1:2;
radius=.00825;
k=ka/radius;
rho=1.21; % density of air in kg/m^3
c=343; % speed of sound in m/s
iterations=1;
r=randn(3,iterations);
r=[0;1;0;];
% r=[0;0;0;];
A=sqrt(rho*c);

e_ortho=zeros(3,length(k),iterations);
e_six=zeros(3,length(k),iterations);
deltaphase_ortho=zeros(3,length(k),iterations);
deltaphase_six=zeros(3,length(k),iterations);
for a=1:iterations

   for b= 1:length(k)

      [e_ortho(:,b,a), e_six(:,b,a),
e_tetra(:,b,a),deltaphase_ortho(:,b,a),deltaphase_six(:,b,a),deltaphase_tetra(:,b,a)]=fcthree(r(:,a),k(b),radius,
A);
      %       [dberror_ortho(:,1:length(k),a), dberror_six(:,1:length(k),a), ...
      %             deltaphase_ortho(:,1:length(k),a),...
      %             deltaphase_six(:,1:length(k),a)]=fcthree(r(:,a),k);
      %



      %    [dberror_ortho(:,1:length(k),a), dberror_tetra(:,1:length(k),a), dberror_six(:,1:length(k),a), ...
      %       6    deltaphase_ortho(:,1:length(k),a),deltaphase_tetra(:,1:length(k),a),...
      %           deltaphase_six(:,1:length(k),a)]=fcthree(r(:,a),k);

   end


end
eom=mean(e_ortho,3); % Error Orthogonal Magnitude
eomstd=std(e_ortho,0,3); % Error Orthogonal Magnitude STD
eop=mean(deltaphase_ortho,3); % Error Orthogonal Phase
eopstd=std(deltaphase_ortho,0,3); % Error Orthogonal Phase STD
etm=mean(e_tetra,3);
etp=mean(deltaphase_tetra,3);
esm=mean(e_six,3); % Error Sixmic Magnitude
esmstd=std(e_six,0,3); % Error Sixmic Magnitude STD
esp=mean(deltaphase_six,3); % Error Sixmic Phase
espstd=std(deltaphase_six,0,3); % Error Sixmic Phase STD

eom=20*log10(eom); %convert into db
esm=20*log10(esm); %convert into db
```

```
eomstd=20*log10(eomstd);
esmstd=20*log10(esmstd);

% % Plot Standard Deviation as a function of frequency
% figure
%
% % plot(ka,eom(1,:),'-.',ka,eom(2,:),'--',ka,eom(3,:),':')
% subplot(2,2,1)
% plot(ka,eomstd(1,:),'-.',ka,eomstd(2,:),'--',ka,eomstd(3,:),'.:')
% title('STD of the Error vs. Frequency, Orthogonal Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Standard Deviation','fontsize',16)
% legend('STD in X','STD in Y','STD in Z')
%
% subplot(2,2,2)
% plot(ka,eopstd(1,:),'-.',ka,eopstd(2,:),'--',ka,eopstd(3,:),'.:')
% title('STD of the Phase Error vs. Frequency, Orthogonal Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Standard Deviation in Degrees','fontsize',16)
% legend('STD in X','STD in Y','STD in Z')
%
% subplot(2,2,3)
% plot(ka,esmstd(1,:),'-.',ka,esmstd(2,:),'--',ka,esmstd(3,:),'.:')
% title('STD of the Error vs. Frequency, Six Microphone Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Standard Deviation','fontsize',16)
% legend('STD in X','STD in Y','STD in Z')
%
% subplot(2,2,4)
% plot(ka,espstd(1,:),'-.',ka,espstd(2,:),'--',ka,espstd(3,:),'.:')
% title('STD of the Phase Error vs. Frequency, Six Microphone Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Standard Deviation in Degrees','fontsize',16)
% legend('STD in X','STD in Y','STD in Z')
%
% % Plot Error as a function of kauency
% figure
%
% subplot(2,2,1)
% plot(ka,eom(1,:),'-.',ka,eom(2,:),'--',ka,eom(3,:),'.:')
% title('Magnitude Error vs. Frequency, Orthogonal Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Error in dB','fontsize',16)
% legend('Error in X','Error in Y','Error in Z')
%
% subplot(2,2,2)
% plot(ka,eop(1,:),'-.',ka,eop(2,:),'--',ka,eop(3,:),'.:')
% title('Phase Error vs. Frequency, Orthogonal Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Error in Degrees','fontsize',16)
% legend('Error in X','Error in Y','Error in Z')
%
% subplot(2,2,3)
% plot(ka,esm(1,:),'-.',ka,esm(2,:),'--',ka,esm(3,:),'.:')
% title('Magnitude Error vs. Frequency, Six Microphone Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
```

```
% ylabel('Error in dB','fontsize',16)
% legend('Error in X','Error in Y','Error in Z')
%
% subplot(2,2,4)
% plot(ka,esp(1,:),'-.',ka,esp(2,:),'--',ka,esp(3,:),'.:')
% title('Phase Error vs. Frequency, Six Microphone Probe','fontsize',16)
% xlabel('Frequency in Hz','fontsize',16)
% ylabel('Error in Degrees','fontsize',16)
% legend('Error in X','Error in Y','Error in Z')

figure
subplot(2,1,1)
plot(ka,etm(1,:),ka,etm(2,:),ka,etm(3,:))
title('Magnitude Error vs. Frequency, Tetrahedron Probe','fontsize',16)
xlabel('Frequency in Hz','fontsize',16)
ylabel('Error in dB','fontsize',16)
legend('Error in X','Error in Y','Error in Z')

subplot(2,1,2)
plot(ka,etp(1,:),ka,etp(2,:),ka,etp(3,:))
title('Phase Error vs. Frequency, Tetrahedron Probe','fontsize',16)
xlabel('Frequency in Hz','fontsize',16)
ylabel('Error in Degrees','fontsize',16)
legend('Error in X','Error in Y','Error in Z')
```

```
function [varargout] = FFierror(r,ka,radius,A,p)
% FFIERROR    free field intensity Error
%     [Esix,Etetra,Eortho] = FFIERROR(r,ka,radius,A,p) determines the intensity error when using the
orthogonal,
%     six mic, and tetrahedron probes. This script calls isix, iono,
%     iortho, and iactual to estimate the intensity based on the various
%     probe geometries.
%
%     FFIERROR with no argument will estimate the intensity error at one random location.
%     If one argument is passed in, example: FFIERROR(10), will estimate the
%     error of 10 random locations based on the radius of .00825 meters. If four arguments are passed in,
%     FFIERROR will estimate the intensity error at the location r, given
%     ka, radius, and source strength A. The fifth
%     variable, p, is a flag to plot. The default is to display the plots.
%     If you don't want to see plots, set p=0
%
% ffierror stands for FreeField Intensity Error.

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      r=randn(3,1); % generate random vector
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, which corresponds to the number of iterations
      iterations=r;
      r=randn(3,iterations);
      radius=.00825; % default is the small sphere
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 4
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   otherwise
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
end

rho=1.21; % density of air
c= 343; % speed of sound in air

for index=1:iterations
   Isix(:,:,index)= isix(r(:,index),k,radius,A(index)); % estimate the intensity based on pressure gradiants
   Iortho(:,:,index)=iortho(r(:,index),k,radius,A(index));
   Itetra(:,:,index)=itetra(r(:,index),k,radius,A(index)); % estimate intensity based on Ono Sokki
   Iactual(:,:,index)=real(intensity(r(:,index),k,A(index))); %determine the actual complex intensity
```

```
end

Esix=10*log10(mean(abs(Isix./Iactual),3));
Etetra=10*log10(mean(abs(Itetra./Iactual),3));
Eortho=10*log10(mean(abs(Iortho./Iactual),3));
if p~=0
   figure
   maxfig
   hold on
   H=plot(ka,Esix(1,:),  'm--d',ka,Esix(2,:),  'm--s',ka,Esix(3,:),'m--v');
   set(H,'linewidth',2);
   H=plot(ka,Etetra(1,:),'r-.d',ka,Etetra(2,:),'r-.s',ka,Etetra(3,:),'r-.v');
   set(H,'linewidth',2);
   H=plot(ka,Eortho(1,:),'b:d',ka,Eortho(2,:),'b:s',ka,Eortho(3,:),'b:v');
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Intensity Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Intensity Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',0);
   set(H,'fontsize',14);
   hold off

   figure
   hold on
   H=plot(ka,180/pi*(mean((angle(Iactual(1,:,:))-angle(Isix(1,:,:))),3))  ,'m--
d',ka,180/pi*(mean((angle(Iactual(2,:,:))-angle(Isix(2,:,:))),3))  ,'m--
s',ka,180/pi*(mean((angle(Iactual(3,:,:))-angle(Isix(3,:,:))),3)),'m--v');
   set(H,'linewidth',2);
   H=plot(ka,180/pi*(mean((angle(Iactual(1,:,:))-angle(Itetra(1,:,:))),3)),'r-
.d',ka,180/pi*(mean((angle(Iactual(2,:,:))-angle(Itetra(2,:,:))),3)),'r-
.s',ka,180/pi*(mean((angle(Iactual(3,:,:))-angle(Itetra(3,:,:))),3)),'r-.v');
   set(H,'linewidth',2);
   H=plot(ka,180/pi*(mean((angle(Iactual(1,:,:))-
angle(Iortho(1,:,:))),3)),'b:d',ka,180/pi*(mean((angle(Iactual(1,:,:))-
angle(Iortho(1,:,:))),3)),'b:s',ka,180/pi*(mean((angle(Iactual(1,:,:))-angle(Iortho(1,:,:))),3)),'b:v');
   set(H,'linewidth',2);
   xlabel('ka','fontsize',16);
   ylabel('Error in degrees','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',...
      0);
   set(H,'fontsize',14);
   if iterations==1
      H=title(['Intensity Phase Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
```

```
      H=title(['Intensity Phase Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end

end
nao=nargout; % number arguments outputed
if nao~=0
   varargout={Esix,Etetra,Eortho};
end
```

```
function [varargout] = FFperror(r,ka,radius,A,p)
% FFPERROR  free field energy error.
%    [Esix,Etetra,Eortho] = FFPERROR(r,ka,radius,A,p) collects the pressure error for the various probe
designs.
%    FFPERROR then takes averages and standard deviations of the various
%    errors. It also plots the results.
%
%    FFPERROR with no argument will estimate the pressure error at one random location.
%    If one argument is passed in, example: FFPERROR(10), will estimate the
%    error of 10 random locations with a default radius .00825 meters. If four
%    arguments are passed in, FFPERROR will estimate the intensity error at the
%    location r, given ka, radius, and source strength A. The fifth
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      r=randn(3,1); % generate random vector
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, number of iterations
      iterations=r;
      r=randn(3,iterations);
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 4
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   otherwise
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
end

rho=1.21; % density of air
c= 343; % speed of sound in air
for index=1:iterations
   pa(index,:)=pressure(r(:,index),k,A(index));
   [p1,p2,p3,p4]=portho(r(:,index),k,radius,A(index));
   po(index,:)=(p1+p2+p3+p4)/4;
   [p1,p2,p3,p4]=ptetra(r(:,index),k,radius,A(index));
   pt(index,:)=(p1+p2+p3+p4)/4;
   [p1,p2,p3,p4,p5,p6]=psix(r(:,index),k,radius,A(index));
   ps(index,:)=(p1+p2+p3+p4+p5+p6)/6;
end
```

```
eps=abs(20*log10(mean(ps./pa,1))); % energy error six
ept=abs(20*log10(mean(pt./pa,1))); % energy error tetra
epo=abs(20*log10(mean(po./pa,1))); % energy error ortho
if p~=0
   figure
   maxfig

   H=plot(ka,eps,'m--s',ka,ept,'r-.v',ka,epo,'b:d');
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Pressure Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Pressure Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);

   H=legend('Six Mic','Tetra','Ortho');
   set(H,'fontsize',14);
end
nao=nargout; % number arguments outputed
if nao~=0
   varargout={eps,ept,epo};
end
```

```
function [varargout] = FFverror(r,ka,radius,A,p)
% FFVERROR  free field energy error.
%    [Esix,Etetra,Eortho] = FFVERROR(r,ka,radius,A,p) collects the velocity error for the various probe
designs.
%    FFVERROR then takes averages and standard deviations of the various
%    errors. It also plots the results.
%
%    FFVERROR with no argument will estimate the velocity error at one random location.
%    If one argument is passed in, example: FFVERROR(10), will estimate the
%    error of 10 random locations with a default radius .00825 meters. If four
%    arguments are passed in, FFVERROR will estimate the intensity error at the
%    location r, given ka, radius, and source strength A. The fifth
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      r=randn(3,1); % generate random vector
      iterations=1;
      radius=.00825;
      ka=.001:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, number of iterations
      iterations=r;
      r=randn(3,iterations);
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 4
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   otherwise
      % use r, k radius as passed in
      k=ka/radius;
      iterations=1;
end
rho=1.21; % density of air
c= 343; % speed of sound in air

% r=[1;.03;1.1];
% A=.2+.03*j;
for index=1:iterations
   va(:,:,index)=velocity(r(:,index),k,A(index));
   dbSVL=20*log10(abs(va)/5e-8);
   abs(va(2,:));
   for temp=1:length(k)
      vo(:,temp,index)=vortho(r(:,index),k(temp),radius,A(index));
      vt(:,temp,index)=vtetra(r(:,index),k(temp),radius,A(index));
```

```
    end
    vs(:,:,index)=vsix(r(:,index),k,radius,A(index));
end
evo=(20*log10(mean(abs(vo./va),3))); % error velocity ortho
evs=(20*log10(mean(abs(vs./va),3))); % error velocity six
evt=(20*log10(mean(abs(vt./va),3))); % error velocity tetra

pvo=mean(abs(2*pi*angle(vo)-2*pi*angle(va)),3); % p stands for phase
pvoU=max(2*pi*angle(vo)-2*pi*angle(va),[],3); % Upper confidence band
pvoL=max(2*pi*angle(vo)-2*pi*angle(va),[],3); % Lower confidence band

pvs=mean(abs(2*pi*angle(vs)-2*pi*angle(va)),3); % p stands for phase
pvsU=max(2*pi*angle(vs)-2*pi*angle(va),[],3); % Upper confidence band
pvsL=max(2*pi*angle(vs)-2*pi*angle(va),[],3); % Lower confidence band

pvt=mean(abs(2*pi*angle(vt)-2*pi*angle(va)),3); % p stands for phase
pvtU=max(2*pi*angle(vo)-2*pi*angle(va),[],3); % Upper confidence band
pvtL=max(2*pi*angle(vo)-2*pi*angle(va),[],3); % Lower confidence band

if p~=0

    figure
    maxfig
    hold on
    H=plot(ka,evs(1,:),'m--d',ka,evs(2,:),'m--s',ka,evs(3,:),'m--v'); % y axis is a square
    set(H,'linewidth',2);
    H=plot(ka,evt(1,:),'r-.d',ka,evt(2,:),'r-.s',ka,evt(3,:),'r-.v'); % z axis is a v
    set(H,'linewidth',2);
    H=plot(ka,evo(1,:),'b:d',ka,evo(2,:),'b:s',ka,evo(3,:),'b:v'); % x axis is a diamond
    set(H,'linewidth',2);
    if iterations==1
        H=title(['Velocity Error vs. ka, Point Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
    else
        H=title(['Velocity Error vs. ka, Point Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
    end
    xlabel('ka','fontsize',16);
    ylabel('Error in dB','fontsize',16);
    H=legend('Six Mic x','Six Mic y','Six Mic z',...
        'Tetra x','Tetra y','Tetra z',...
        'Ortho x','Ortho y','Ortho z',0);
    set(H,'fontsize',14);

    figure % Plot Phase Error
    maxfig
    hold on

    H=plot(ka,pvs(1,:,:),'m--d',ka,pvs(2,:,:),'m--s',ka,pvs(3,:,:),'m--v'); % y axis is a square
    set(H,'linewidth',2);

    % H=errorbar(ka,pvo(2,:,:),pvoU(2,:,:),pvoL(2,:,:),'b');
    % set(H,'linewidth',2);
    % H=errorbar(ka,pvs(2,:,:),pvsU(2,:,:),pvsL(2,:,:),'m');
    % set(H,'linewidth',2);
    % H=errorbar(ka,pvt(2,:,:),pvtU(2,:,:),pvtL(2,:,:),'r');
```

```
% set(H,'linewidth',2);

H=plot(ka,pvt(1,:,:),'r-.d',ka,pvt(2,:,:),'r-.s',ka,pvt(3,:,:),'r-.v'); % z axis is a v
set(H,'linewidth',2);

% H=errorbar(ka,pvo(3,:,:),pvoU(3,:,:),pvoL(3,:,:),'b');
% set(H,'linewidth',2);
% H=errorbar(ka,pvs(3,:,:),pvsU(3,:,:),pvsL(3,:,:),'m');
% set(H,'linewidth',2);
% H=errorbar(ka,pvt(3,:,:),pvtU(3,:,:),pvtL(3,:,:),'r');
% set(H,'linewidth',2);

H=plot(ka,pvo(1,:,:),'b:d',ka,pvo(2,:,:),'b:s',ka,pvo(3,:,:),'b:v'); % x axis is a diamond
set(H,'linewidth',2);

% H=errorbar(ka,pvo(1,:,:),pvoU(1,:,:),pvoL(1,:,:),'b');
% set(H,'linewidth',2);
% H=errorbar(ka,pvs(1,:,:),pvsU(1,:,:),pvsL(1,:,:),'m');
% set(H,'linewidth',2);
% H=errorbar(ka,pvt(1,:,:),pvtU(1,:,:),pvtL(1,:,:),'r');
% set(H,'linewidth',2);

if iterations==1
   H=title(['Velocity Phase Error vs. ka, Point Sensors, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Velocity Phase Error vs. ka, Point Sensors, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in Degrees','fontsize',16);
H=legend('Six Mic x','Six Mic y','Six Mic z',...
   'Tetra x','Tetra y','Tetra z',...
   'Ortho x','Ortho y','Ortho z',0);
set(H,'fontsize',14);    hold off

%%%%%%%%%%%%%%% velocity magnitude error, in the r direction,
set(0,'DefaultAxesColorOrder',[1 0 0;],'DefaultAxesLineStyleOrder','-|:|--|-.')

Vmagsix=20*log10(sqrt(dot(vs,vs)./dot(va,va)));
Vmagsix=mean(Vmagsix,3);
Vmagtetra=20*log10(sqrt(dot(vt,vt)./dot(va,va)));
Vmagtetra=mean(Vmagtetra,3);
Vmagortho=20*log10(sqrt(dot(vo,vo)./dot(va,va)));
Vmagortho=mean(Vmagortho,3);

figure
H=plot(ka,Vmagsix,'m',ka,Vmagtetra,'r',ka,Vmagortho,'b');
set(H,'linewidth',2);
if iterations==1
   H=title(['Velocity Magnitude Error vs. ka, Point Sensors, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Velocity Magnitude Error vs. ka, Point Sensors, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
```

```
   H=legend('Six Mic','Tetra','Ortho',0);
   set(H,'fontsize',14);
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);

end
nao=nargout; % number arguments outputed
if nao~=0
   varargout={evs,evt,evo};
end
```

```
% FVSIX    free field velocity estimate for six microphone probe
%     FVSIX(r,k,radius,A) estimates the velocity at the center of the
%     sphere based on Euler's Equation. It outputs velocity in x,y,z.
%
% Six microphone technique for estimated 3d velocity. This script accepts
% inputs of r,f,radius- r being the location of the probe, f being the
% frequency of interest, and radius being the radius of the probe.

function output=fvsix(r,k,radius,A)

c=343; % speed of sound in air
rho=1.21; % density of air in kg/m^3
omega=k*c; % angular velocity

% determine the locations of the six microphones and the vectors that point
% to them
r1=r+[ radius;0;0;];
r2=r+[-radius;0;0;];
r3=r+[0; radius;0;];
r4=r+[0;-radius;0;];
r5=r+[0;0; radius;];
r6=r+[0;0;-radius;];

% Determine the pressure at each microphone
p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
p5=pressure(r5,k,A);
p6=pressure(r6,k,A);

% determine the particle velocity at the center of the probe
Ux=(p2-p1)./(j*rho.*omega*(r1(1)-r2(1))); % This vector points in the - x direction
Uy=(p4-p3)./(j*rho.*omega*(r3(2)-r4(2))); % This vector points in the - y direction
Uz=(p6-p5)./(j*rho.*omega*(r5(3)-r6(3))); % This vector points in the - z direction

output=[Ux;Uy;Uz];
```

```
% Heather Smith code developed for Dr. Leishman's phsyics 565 class.
% problem 4.2.1 plane wave impinging on a rigid sphere of radius a. Code
% modified by Lance Locey.

function Ptotal=hmodscatt(radius,ka,phi,theta,eta,A)

psc=0;
mmax=13; %number of terms in summation

c=343;
a=radius;  %radius of sphere
r=radius; %on the surface of the sphere r=a
k=ka/a;
kr=r*k;
% N=99; %angular resolution
% theta=0:2*pi/N:2*pi;

%%%%%%%%%%%%%%taken from spsixtemp
k=ka/radius;
micloc=[+radius,0,0;-radius,0,0;0,+radius,0;0,-radius,0;0,0,+radius;0,0,-radius;]; % mic location matrix,
assuming no rotations in phi or theta

% rotate the microphones about the x axis
micloc=micloc*[cos(eta)*cos(theta)+sin(eta)*sin(theta)*sin(phi), cos(eta)*sin(theta)-
sin(eta)*cos(theta)*sin(phi), sin(eta)*cos(phi);
   -sin(theta)*cos(phi), cos(phi)*cos(theta), sin(phi);
   -sin(eta)*cos(theta)+cos(eta)*sin(theta)*sin(phi), -sin(eta)*sin(theta)-cos(eta)*cos(theta)*sin(phi),
cos(eta)*cos(phi)];

beta=atan2(sqrt(micloc(:,1).^2+micloc(:,2).^2),(micloc(:,3))); % picks off the polar angle  in the ?? plane
theta=beta; % BEWARE theta is used in two different contexts!~!
%%%%%%%%%%%%%%%%%%%%taken from spsixtemp

for m=0:mmax
   deltam=atan(((m+1)*sphB(m+1,ka)-m*sphB(m-1,ka))./(m*sphN(m-1,ka)-(m+1)*sphN(m+1,ka)));
   Pm=legendre(m,cos(theta)); % note the addition of the phi term, assuming no change in eta or theta
   pterm=A*(2*m+1)*i^(m+1)*exp(-i*deltam)*sin(deltam)*Pm(1,:)*(sphB(m,kr)+i*sphN(m,kr));
   psc=psc+pterm;
end


%%%%%%%%%%%%%%%%%%%%taken from spsixtemp

Pplane=A*exp(j*k*micloc(:,3))'; % note, no exp(-jkz) because the plane wave is propogating in the
negative z
Ptotal=psc+Pplane;
%%%%%%%%%%%%%%%%%%%%taken from spsixtemp

figure
polar(theta',abs(psc));
title(['|psc(theta)| for ka=' num2str(ka,3)])


phi=0:2*pi/N:2*pi;
theta2=0:2*pi/N:pi;
[Theta,Phi]=meshgrid(theta2,phi);
```

```
newpsc=zeros(length(phi),N/2+1);

b=max(abs(psc(1:(N/2+1))));
for n=1:length(phi)
   newpsc(n,:)=abs(psc(1:(N/2+1)))/b;
end

   x=newpsc.*cos(Phi).*sin(Theta);
   y=newpsc.*sin(Phi).*sin(Theta);
   z=newpsc.*cos(Theta);
   figure
   surf(x,y,z,newpsc);
   caxis([min(min(min(newpsc))),max(max(max(newpsc)))]);
   view(3);
   colorbar('vert')
   shading interp
   axis equal
   xlabel('X');ylabel('Y');zlabel('Z');
   title(['|psc(theta,phi)| for ' int2str(f) ' Hz'])
```

```
%This file reads in a text file from wavefront.  It converts it to fft and
%than calculates energy densities based on if it is the six mic, ortho, or
%tetra configuration.  Create by Brady Woolford and Lance Locey on April
%19, 2004


clc
close all
clear
d=dir; % setup an array of structs that has info on each item in the directory
numfiles=length(d); % each item in the directory must be checked
for k=3:numfiles % the first 2 elements are '.' and '..', which is directory stuff, so skip it
   tempname=d(k).name; % get the name of the first item in the directory
   p=1;
   tempsize=length(tempname);
   while (tempname(p)~='.' & p<tempsize ) % search through the string and figure out where the .txt starts
      p=p+1;
   end % end while

   if tempsize==p+3
      if strcmpi(tempname(p+1:p+3),'txt')
       file=tempname;
      end % end if
   end
  % The below code needs modification to do a recersive search in other directories
%     if (d(k).isdir==1)
%        if (k~=2) % if it's not the 1st or 2nd item in the directory and it is a directory, then process it
%           ecfilenamesearch(d(k).name,filelist); % if the item is a directory, start the search over again
%           cd('..') % return to the previous directory
%        end % end '..'
%     end % end isdir

end % end numfiles for loop



data_file=textread(file,'%s');  %This reads in the desired wavefront file
I=strmatch('(V)', data_file); %following this value is where the data begins for each mic

D=size(I);  %determines the number of rows and columns for the I matrix based on the imported data

if D(1)==6;       %Calculates value for six mic probe, D(1) tells the number of rows
                  %in the I matrix which correspondes to the number of mics on probe
   %pressure mic 1
   p1=data_file(I(1)+1:(I(2)-I(1)));
   p1=str2double(p1);
   p1_fft=fft(p1/length(p1));  %fft takes the pressures from time base to frequency base
   %pressure mic 2
   p2=data_file(I(2)+1:(I(3)-I(1)));
   p2=str2double(p2);
   p2_fft=fft(p2/length(p2));
   %pressure mic 3
   p3=data_file(I(3)+1:(I(4)-I(1)));
   p3=str2double(p3);
   p3_fft=fft(p3/length(p3));
   %pressure mic 4
```

```
    p4=data_file(I(4)+1:(I(5)-I(1)));
    p4=str2double(p4);
    p4_fft=fft(p4/length(p4));
    %pressure mic 5
    p5=data_file(I(5)+1:(I(6)-I(1)));
    p5=str2double(p5);
    p5_fft=fft(p5/length(p5));
    %pressure mic 6
    p6=data_file(I(6)+1:length(data_file));
    p6=str2double(p6);
    p6_fft=fft(p6/length(p6));

    %plotting routines for six mic probe
    figure
    subplot(3,2,1)
    plot(p1_fft)

    subplot(3,2,2)
    plot(p2_fft)

    subplot(3,2,3)
    plot(p3_fft)

    subplot(3,2,4)
    plot(p4_fft)

    subplot(3,2,5)
    plot(p5_fft)

    subplot(3,2,6)
    plot(p6_fft)

else %This else statement will calculate the values for the ortho and tetra probe
    %pressure mic 1
    p1=data_file(I(1)+1:(I(2)-I(1)));
    p1=str2double(p1);
    p1_fft=fft(p1/length(p1));
    %pressure mic 2
    p2=data_file(I(2)+1:(I(3)-I(1)));
    p2=str2double(p2);
    p2_fft=fft(p2/length(p2));
    %pressure mic 3
    p3=data_file(I(3)+1:(I(4)-I(1)));
    p3=str2double(p3);
    p3_fft=fft(p3/length(p3));
    %pressure mic 4
    p4=data_file(I(4)+1:length(data_file));
    p4=str2double(p4);
    p4_fft=fft(p4/length(p4));

    %plotting routines for tetra and ortho
    figure
    subplot(2,2,1)
    plot(p1_fft)

    subplot(2,2,2)
```

```
    plot(p2_fft)

    subplot(2,2,3)
    plot(p3_fft)

    subplot(2,2,4)
    plot(p4_fft)

end
```

```
% INTENSITY     compute the intensity
%     INTENSITY(r,k,A) calls the pressure function to determine the
%     pressure at a given location. It then computes the intensity at that
%     location by multiplying p*uconjugate/2. It returns complex intensity
%     values.
%
% Uc=intensity(r,k,A) returns a vector value corresponding to the complex
% intensity associated with the vector r. A is the source strength

function Ic=intensity(r,k,A)

p=pressure(r,k,A);
uconj=conj(velocity(r,k,A));
Icx= (1/2)*p.*uconj(1,:);
Icy= (1/2)*p.*uconj(2,:);
Icz= (1/2)*p.*uconj(3,:);

Ic=[Icx;Icy;Icz;];
```

```
% IORTHO intensity estimate based on the orthogonal probe geometry
%       IORTHO(r,k,radius,A)
% iortho(r,k,radius,A) determines the active intensity based on the
% orthogonal probe geometry
function Ia=iortho(r,k,radius,A)

r1=[r(1)+radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic1 position
r2=[r(1)-radius/sqrt(3);r(2)+radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic2 position
r3=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic3 position
r4=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)+radius/sqrt(3)]; % mic 4 position

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

p=(p1+p2+p3+p4)/4;

for temp=1:length(k)
u(:,temp)=vortho(r,k(temp),radius,A);
end

% uactual=velocity(r,k,A);
Ix=real((1/2)*p.*conj(u(1,:)));
Iy=real((1/2)*p.*conj(u(2,:)));
Iz=real((1/2)*p.*conj(u(3,:)));
Ia=[Ix;Iy;Iz];
```

```
% ISIX      intensity estimated by the six mic probe
%     ISIX(r,k,radius,A) estimates the intensity at the center of a sphere
%     based on the six mic probe configuration.
function Ic=isix(r,k,radius,A)

c=343;
rho=1.21;
omega=k*c;

r1=r+[ radius;0;0;];
r2=r+[-radius;0;0;];
r3=r+[0; radius;0;];
r4=r+[0;-radius;0;];
r5=r+[0;0; radius;];
r6=r+[0;0;-radius;];

% Determine the pressure at each microphone
p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
p5=pressure(r5,k,A);
p6=pressure(r6,k,A);

G12=p1.*conj(p2)/2;
G34=p3.*conj(p4)/2;
G56=p5.*conj(p6)/2;

Icx=-imag(G12)./(omega*rho*2*radius);
Icy=-imag(G34)./(omega*rho*2*radius);
Icz=-imag(G56)./(omega*rho*2*radius);



Ic=[Icx;Icy;Icz;];
```

```
function Ic=itetra(r,k,radius,A)
% ITETRA      Intensity based on Ono Sokki Method
%     ITETRA(r,k,radius,A) computes the intensity at the center of the
%     sphere based on the Ono Sokki cross spectrum method at a location r.
%     ka is nondimensional frequency given a monopole source of strength
%     A.
% function determines the active intensity based on the ono sokki
% tetrahedron probe. (r,k,radius,A)

c=343;
rho=1.21;
omega=k*c;

%%%%%%%%%%%%%%%%%% Tetrahedron mic positioning scheme
r1=[r(1)                ; r(2)+2/3*radius*2^(1/2)       ; r(3)-1/3*radius];
r2=[r(1)-1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r3=[r(1)+1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r4=[r(1)                ; r(2)                      ; r(3)+radius];

% seperation distance between microphones
d=sqrt(dot(r1-r2,r1-r2));

if abs(d-(2*sqrt(6)*radius/3))>1e-6 % This checks to make sure the seperation distance is accurate
   disp('error computing seperation distance between microphones')
   return
end

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

G12=p1.*conj(p2)/2;
G13=p1.*conj(p3)/2;
G14=p1.*conj(p4)/2;
G23=p2.*conj(p3)/2;
G24=p2.*conj(p4)/2;
G34=p3.*conj(p4)/2;
%%%%%%%%% Ono Sokki Stuff
Ix=-3./(omega*rho*d).*imag( (1/6)*(G34/2 - G24/2 - G23 +G12/2 - G13/2 ) );
Iy=-3./(omega*rho*d).*imag( (1/6)*(sqrt(3)*G34/6 +sqrt(3)*G24/6 - 2*sqrt(3)*G14/6 ...
   -3*sqrt(3)*G12/6 -3*sqrt(3)*G12/6 ));
Iz=-3./(omega*rho*d).*imag( (1/6)*( -sqrt(2)*G34/sqrt(3) -sqrt(2)*G24/sqrt(3) - sqrt(2)*G14/sqrt(3) ));
C1=-1/2; % sign diference comes from taking the Hermitian transpose (in sitetra...)
C2=-1/2;
C3=-1/2;
% note- don't divide by 1.5 because of over estimating the velocity (no
% sphere present)
Vx=C1*(-2*p2+2*p3)./(j*omega*rho*d);
Vy=C2*(4*p1/sqrt(3)-2*p2/sqrt(3)-2*p3/sqrt(3))./(j*omega*rho*d);
Vz=C3*(-sqrt(2/3)*p1-sqrt(2/3)*p2-sqrt(2/3)*p3+3*sqrt(2/3)*p4)./(j*omega*rho*d);
Ix=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vx));
Iy=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vy));
Iz=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vz));

Ic=[Ix;Iy;Iz];
```

```
% MAXFIG    maximize the figure
%     MAXFIG maximizes the current figure to fill the entire screen.

function maxfig
H=gcf;

set(H,'units','normalized','outerposition',[0 0 1 1]);
```

```
function H=oplot
% OPLOT    plot the orthogonal probe
%    OPLOT plots the orthogonal probe microphones on the surface
%    of a sphere. It is helpful when trying to understand the geometry of
%    the orthogonal probe. The vector r points to the center of the probe.
%    The function returns the handle to the figure. OPLOT the 3
%    stands for 3 dimensional figure. The default is to leave
%    hold on, so that other figures can be drawn on the same sphere.

close all
r=[0;0;0;];

radius=1;

colors=jet(5);

micloc=[r(1)+radius/sqrt(3),r(2)-radius/sqrt(3),r(3)-radius/sqrt(3);
r(1)-radius/sqrt(3),r(2)+radius/sqrt(3),r(3)-radius/sqrt(3); % mic2 position
r(1)-radius/sqrt(3),r(2)-radius/sqrt(3),r(3)-radius/sqrt(3); % mic3 position
r(1)-radius/sqrt(3),r(2)-radius/sqrt(3),r(3)+radius/sqrt(3);];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);


[X,Y,Z] = SPHERE(20);
X=X*radius;
X=X+r(1);
Y=Y*radius;
Y=Y+r(2);
Z=Z*radius;
Z=Z+r(3);

SURF(X,Y,Z)
alpha('clear')
hold on

for i=1:4
   H=plot3(x(i),y(i),z(i),'color',colors(i+1,:))
   set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','MarkerEdgeColor',colors(i+1,:));
   set(H,'MarkerFaceColor',colors(i+1,:))

end
H=legend('Mic 1','Mic 2','Mic 3','Mic 4');
set(H,'FontSize',18);

H=gca;
set(H,'FontSize',14);
H=xlabel('X Axis');
set(H,'FontSize',16);
H=ylabel('Y Axis');
set(H,'FontSize',16);
H=zlabel('Z Axis');
set(H,'FontSize',16);
H=title('Orthogonal Probe','fontsize',16)
```

```
set(gca,'PlotBoxAspectRatioMode','manual','DataAspectRatioMode','manual','CameraViewAngleMode','ma
nual')

CAMERATOOLBAR('Show')
maxfig
axis square
```

```
% OPECHECK    orthogonal probe phase magnitude error check
%     OPECHECK(r,f,radius) checks the phase and magnitude of the orthogonal
%     probe method for estimating velocity.


function opecheck(r,f,radius)

rho=1.21; % density of air
c=343;
omega=2*pi*f;
k=omega/c;


format compact % don't print a lot of space when displaying data....

disp('actual velocity as computed in opmecheck'); % display what the actual particle velocity in X should
be
actual=velocity(r,k) % determine what the actual particle velocity should be for comparison


% use code from vest.m to determine the different components.



r1=[r(1)+radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic1 position
r2=[r(1)-radius/sqrt(3);r(2)+radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic2 position
r3=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic3 position
r4=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)+radius/sqrt(3)]; % mic 4 position

p1=pressure(r1,k);
p2=pressure(r2,k);
p3=pressure(r3,k);
p4=pressure(r4,k);

%On Axis Velocity Values

   Ux13=(p1-p3)./(j*rho.*omega*(r1(1)-r3(1))  ); % This vector points in the - x direction
   Uy23=(p2-p3)./(j*rho.*omega*(r2(2)-r3(2))  ); % This vector points in the - y direction
   Uz43=(p4-p3)./(j*rho.*omega*(r4(3)-r3(3))  ); % This vector points in the - z direction
   Ux31=-Ux13; % This vector points in the + x direction
   Uy32=-Uy23; % This vector points in the + y direction
   Uz34=-Uz43; % This vector points in the + z direction

 % In the XY plane

   if( real(Ux31(1))>0 & real(Uy32(1))>0)
      theta=atan2(abs(Uy23),abs(Ux13));  % first quadrant
   elseif( real(Ux31(1))>0 & real(Uy32(1))<0)
      theta=atan2(-abs(Uy23),abs(Ux13)); % fourth quadrant
   elseif( real(Ux31(1))<0 & real(Uy32(1)) <0)
      theta=atan2(-abs(Uy23),-abs(Ux13)); % third quadrant
   elseif( real(Ux31(1))<0 & real(Uy32(1)) >0)
      theta=atan2(abs(Uy23),-abs(Ux13)); % second quadrant
   else
      disp('Error with Theta- no value in any quadrant')
      disp('computer paused - Ux31==0')
```

```
    theta=zeros(1,5000);
    pause
  end


  U12=(p1-p2)./(j*rho.*omega*sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2)); % This vector points in the + y - x
direction
  U21=-U12;
  Uy12=-U12.*sin(theta)./cos(theta+pi/4);
  Ux21=U21.*cos(theta)./cos(theta+pi/4);

% In the XZ Plane

  U41=(p4-p1)./(j*rho.*omega*sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2)); % This vector points in the - z + x
direction
  U14=-U41;

  if( real(Ux31(1))>0 & real(Uz34(1))>0)
    betaxz=atan2(abs(Uz34),abs(Ux13));  % first quadrant
  elseif( real(Ux31(1))>0 & real(Uz34(1))<0)
    betaxz=atan2(-abs(Uz34),abs(Ux13)); % fourth quadrant
  elseif( real(Ux31(1))<0 & real(Uz34(1)) <0)
    betaxz=atan2(-abs(Uz34),-abs(Ux13)); % third quadrant
  elseif( real(Ux31(1))<0 & real(Uz34(1)) >0)
    betaxz=atan2(abs(Uz34),-abs(Ux13)); % second quadrant
  else
    disp('Error with Theta- no value in any quadrant')
    disp('computer paused - Ux31==0')
    betaxz=zeros(1,5000);
    pause
  end

  Ux14=U14.*cos(betaxz)./cos(betaxz+pi/4);
  Ux41=-Ux14;
  Uz14=-U14.*sin(betaxz)./cos(betaxz+pi/4);

% In the YZ Plane

  U24=(p2-p4)./(j*rho.*omega*sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2)); % This vector points in the - y + z
direction
  U42=-U24;

  if( real(Uy32(1))>0 & real(Uz34(1))>0)
    alphayz=atan2(abs(Uz34),abs(Uy32));  % first quadrant
  elseif( real(Uy32(1))>0 & real(Uz34(1))<0)
    alphayz=atan2(-abs(Uz34),abs(Uy32)); % fourth quadrant
  elseif( real(Uy32(1))<0 & real(Uz34(1)) <0)
    alphayz=atan2(-abs(Uz34),-abs(Uy32)); % third quadrant
  elseif( real(Uy32(1))<0 & real(Uz34(1)) >0)
    alphayz=atan2(abs(Uz34),-abs(Uy32)); % second quadrant
  else
    disp('Error with Theta- no value in any quadrant')
    disp('computer paused - Ux31==0')
    alphayz=zeros(1,5000);
    pause
  end
```

```
    Uz24=U42.*sin(alphayz)./cos(alphayz+pi/4);
    Uy42=U42.*cos(alphayz)./cos(alphayz+pi/4);

    Ux=Ux21-Ux31+Ux41;
    Uy=Uy12-Uy32+Uy42;
    Uz=Uz14+Uz24-Uz34;

    alphayz
    betaxz
    theta

%     disp('Ux=Ux21-Ux31+Ux41')
%     disp(Ux21)
%     disp(Ux31)
%     disp(Ux41)
%     orthoplot3(r)
    dberror_ortho(1)=20*abs(log( abs(Ux)./abs(actual(1))))
    dberror_ortho(2)=20*abs(log( abs(Uy)./abs(actual(2))))
    dberror_ortho(3)=20*abs(log( abs(Uz)./abs(actual(3))))
    deltaphase_ortho(1)=abs(180/pi*(angle(actual(1))-angle(Ux)))% difference in phase in degrees
    deltaphase_ortho(2)=abs(180/pi*(angle(actual(2))-angle(Uy)))% difference in phase in degrees
    deltaphase_ortho(3)=abs(180/pi*(angle(actual(3))-angle(Uz)))% difference in phase in degrees
```

```
function H=orthoplot3(r)
% ORTHOPLOT3    plot the orthogonal probe

%    ORTHOPLOT3(r) plots the orthogonal probe microphones on the surface
%    of a sphere. It is helpful when trying to understand the geometry of
%    the orthogonal probe. The vector r points to the center of the probe.
%    The function returns the handle to the figure. ORTHOPLOT3 the 3
%    stands for 3 dimensional figure. The default is to leave
%    hold on, so that other figures can be drawn on the same sphere.
n=nargin; % n represents the number of arguments passed into the m file.

if n==0
   r=[0;0;0;];
end

radius=.00825;

micloc=[r(1)+radius/sqrt(3),r(2)-radius/sqrt(3),r(3)-radius/sqrt(3);
r(1)-radius/sqrt(3),r(2)+radius/sqrt(3),r(3)-radius/sqrt(3); % mic2 position
r(1)-radius/sqrt(3),r(2)-radius/sqrt(3),r(3)-radius/sqrt(3); % mic3 position
r(1)-radius/sqrt(3),r(2)-radius/sqrt(3),r(3)+radius/sqrt(3);];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);

set(gcf,'DefaultLineEraseMode','xor')

[X,Y,Z] = SPHERE(20);
X=X*.00825;
X=X+r(1);
Y=Y*.00825;
Y=Y+r(2);
Z=Z*.00825;
Z=Z+r(3);

SURF(X,Y,Z)
alpha('clear')
hold on

H=line(x,y,z);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','b')
H=line([x(1),x(4)],[y(1),y(4)],[z(1),z(4)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','b');
H=line([x(1),x(3)],[y(1),y(3)],[z(1),z(3)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','b');
H=line([x(2),x(4)],[y(2),y(4)],[z(2),z(4)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','b');


%%% draw a yellow circle at the center of the sphere
H=line(r(1),r(2),r(3));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','h','Color','y')
H=title('Orthogonal Probe');
set(H,'FontSize',16);
```

```
% hold off
H=gca;
set(H,'FontSize',14);
H=xlabel('X Axis')
set(H,'FontSize',16);
H=ylabel('Y Axis')
set(H,'FontSize',16);
H=zlabel('Z Axis')
set(H,'FontSize',16);
H=gcf;
```

```
function orthoplot3r(phi,theta,eta)
% This function plots a sphere that has microphones on it like the
% orthogonal probe. it accepts the arguments phi, theta, eta which which
% are angles of rotation about x, z, and y.
% this new function, based on orthoplot3, is
% designed to look at rotations of the orthogonal sphere. It takes as
% arguments phi, theta, and eta. Phi is the angle of rotation about the x
% axis. Theta is the angle of rotation about the z axis, and eta is the
% angle of rotation about the y axis. Orthoplot3r the last r at the end of
% the function name stands for rotation

radius=.00825;

micloc=[  +radius/sqrt(3),-radius/sqrt(3),-radius/sqrt(3);

   -radius/sqrt(3),radius/sqrt(3),-radius/sqrt(3); % mic2 position
   -radius/sqrt(3),-radius/sqrt(3),-radius/sqrt(3); % mic3 position
   -radius/sqrt(3),-radius/sqrt(3),+radius/sqrt(3);] % mic 4 position

micloc=micloc*[cos(eta)*cos(theta)+sin(eta)*sin(theta)*sin(phi), cos(eta)*sin(theta)-
sin(eta)*cos(theta)*sin(phi), sin(eta)*cos(phi);
   -sin(theta)*cos(phi), cos(phi)*cos(theta), sin(phi);
   -sin(eta)*cos(theta)+cos(eta)*sin(theta)*sin(phi), -sin(eta)*sin(theta)-cos(eta)*cos(theta)*sin(phi),
cos(eta)*cos(phi)];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);

[X,Y,Z] = SPHERE(20);
X=X*.00825;
Y=Y*.00825;
Z=Z*.00825;
figure
SURF(X,Y,Z)
alpha('clear');
hold on

h=line(x,y,z);
set(h,'Marker','o','MarkerSize',24,'LineWidth',5);
title('Orthogonal Probe');
hold off
```

```
clc
close all
clear

% first plot the triangle with circles representing microphones
r=.00825

hold on
H=line([-r/sqrt(3),r/sqrt(3)],[-r/sqrt(3),-r/sqrt(3)]);
set(H,'linewidth',2);
H=line([-r/sqrt(3),-r/sqrt(3)],[r/sqrt(3),-r/sqrt(3)]);
set(H,'linewidth',2);
H=line([-r/sqrt(3),r/sqrt(3)],[r/sqrt(3),-r/sqrt(3)]);
set(H,'linewidth',2);
% Plot the microphones

plot(r/sqrt(3),-r/sqrt(3),'MarkerSize',24,'Marker','o')
plot(-r/sqrt(3),r/sqrt(3),'MarkerSize',24,'Marker','o')
plot(-r/sqrt(3),-r/sqrt(3),'MarkerSize',24,'Marker','o')
%Plot the straight lines
H=line([-r/sqrt(3),1.5*r/sqrt(3)],[-r/sqrt(3),1.5*r/sqrt(3)]/3);
set(H,'linewidth',2,'LineStyle','--');
H=line([0,1.5*r/sqrt(3)],[0,0]);
set(H,'linewidth',2);
H=line([1.5*r/sqrt(3),1.5*r/sqrt(3)],[0,1.5*r/sqrt(3)/3]);
set(H,'linewidth',2);

% Plot the angles

x=r/sqrt(10):.000001:r/3;
y=sqrt((r^2)/9-x.^2);
H=plot(x,y,'b-','linewidth',2);

x=-2.916e-3:.0000001:3.9133e-3;
y=sqrt((r^2)/4-x.^2);
H=plot(x,y,'b-','linewidth',2);

title('X-Y Plane')
```

```
close all
clc
clear

rho=1.21;
c=343;
ka=[.05,.1,.3,.5];
theta=0:.01:pi;
R=[0,.97,-0.97];
figure;
S={'k-','k:','k--','k-.'};
for Rindex=1:length(R)
   for kaindex=1:length(ka)
      Pc=1;  % eqn 4.10a
      Vt=1*cos(theta)/(rho*c); % eqn 4.10b
      et=1/(12*rho*c^2)*(1+3*cos(theta).*cos(theta)); % eqn 4.10c
      It=cos(theta)/(2*rho*c); % eqn 4.10d

      Pm2= 1; % Pm2 is the pressure at microphone 2
      dM=1; % sensitivity mismatch
      dP=.25; % phase mismatch

      dMdb=20*log10(dM); % equation 4.3
      Pm1=dM*exp(j*dP)*Pm2; % equation 4.1

      Ut=Pc*conj(Pc)/(12*rho*c^2); % Equation 4.4a, Pc is the complex pressure, Ut stands for U as a
function of theta
      Tt=rho*Vt.*conj(Vt)/4; % equation 4.4b
      et= Ut+Tt; % equation 4.4c
      It= 1/2*real(Pc*conj(Vt)); % eqn 4.4d

      Pec= (Pm2+Pm1)/2; % equation 4.7
      Vet(kaindex)= (Pm2-Pm1)./(j*rho*c*2*ka(kaindex)*3/2); % eqn 4.8 (j*rho*c*2*k*d)
********** Here may be a problem when comparing sphere with 2 mics
      Pf=exp(-j*ka(kaindex)*cos(theta))+R(Rindex)*exp(j*ka(kaindex)*cos(theta)); % eqn 4.14
      Ps_hat=ps(ka(kaindex),theta,R(Rindex)); % eqn 4.13
      Pex=Ps_hat.'./Pf;
      subplot(3,2,2*Rindex-1)
      hold on
      plot(180/pi*theta,20*log10(abs(Pex)),char(S(kaindex)))
      title(['Parkins pg 44 R=',num2str(R(Rindex))])
      hold off
      subplot(3,2,2*Rindex)
      hold on
      plot(180/pi*theta,180/pi*angle(Pex),char(S(kaindex)))
      title(['Parkins pg 44 R=',num2str(R(Rindex))])
      hold off
   end % end ka
   hold off
end % end R loop
%
%

% %%%%%%%%%%%%% Bias Errors for Two Point Sensor Page 47... Graph Comparison
% on Page 48
% clc
```

```
clear
% close all
rho=1.21;
c=343;
theta=0;
kbcostheta=.001:.001:.75;

R=0;

dM=1; % sensitivity mismatch
dP=0; % phase mismatch
Pm1=dM*exp(j*dP)*(exp(-j*kbcostheta)+R*exp(j*kbcostheta)); % eqn 4.24a
Pm2=(exp(j*kbcostheta)+R*exp(-j*kbcostheta)); % eqn 4.24b

Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kbcostheta); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
title('Perfect matched two microphones')
subplot(2,2,1)
plot(kbcostheta,UbdB);title('U bias compare pg 48');
subplot(2,2,2)
plot(kbcostheta,TbdB);title('T bias');
subplot(2,2,3)
plot(kbcostheta,ebdB);title('e bias');
subplot(2,2,4)
plot(kbcostheta,IbdB);title('I bias');

%%%%%Bias errors for spherical sensor Parkins PG 48 Graph comparison on
%%%%%page 49
% clc
% close all
clear
dM=1;
dP=0;
rho=1.21;
c=343;
R=0;
ka=.01:.01:.5;
theta=0;
Pm1=dM*exp(j*dP)*Ps(ka,theta,R);
Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance- I
think he is wrong
```

```
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Pc=1+R;

Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U bias compare pg 49');
subplot(2,2,2)
plot(ka,TbdB);title('T bias');
subplot(2,2,3)
plot(ka,ebdB);title('e bias');
subplot(2,2,4)
plot(ka,IbdB);title('I bias');


%%%%%%%%%%%%%%%%%%%%%%%% Bias errors for mismatched microphones pg 49
%%%%%%%%%%%%%%%%%%%%%%%% compare graph on pg 50
% close all
clear
clc
clear('UbdB','TbdB','ebdB','IbdB')
dP=pi/180; % delta Phase
dM=10^(-1/80); % delta Magnitude
rho=1.21;
c=343;
theta=0;
kb=.01:.001:.75;
ka=2/3*kb; % see bottom of page 45!!!!
R=.97;

% first determine pressure given two point sensor
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta))+R*exp(j*kb*cos(theta))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta))+R*exp(-j*kb*cos(theta))); % eqn 4.24b
% Exact quantities at center of sensor
Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d
% Estimates at center

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a problem
when comparing sphere with 2 mics
```

```
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));

UbdB(1,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(1,:)= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB(1,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(1,:)= 10*log10(abs(Iet)./It); % ean 4.9d

% now determine quantities given spherical sensor
Pm1=dM*exp(j*dP)*Ps(ka,theta,R);
Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance- I
think he is wrong
% exact acoustical values at center
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d


% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*ka); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a problem
when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB(2,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(2,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB(2,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(2,:)= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(kb,UbdB);title('U Bias compare pg 50');
ylim([-3 3])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(kb,TbdB);title('T Bias');
ylim([0 50])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(kb,ebdB);title('e Bias');
ylim([-3 3])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')
subplot(2,2,4)
plot(kb,IbdB);title('I Bias');
ylim([-30 20])
xlabel('kb')
```

```
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')


%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 54
clc
% close all
clear

ka=.01:.01:.5;
R=0;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

for index=1:length(dP)
   Pm1=dM(index)*exp(j*dP(index))*Ps(ka,theta,R);
   Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance-
I think he is wrong


   % Estimate of Pressure, velocity at center
   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*3/2*ka); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

   % Estimate of NRG, Intensity at center
   eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
   Iet=1/2*real(Pec.*conj(Vet));

   UbdB(index,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
   TbdB(index,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
   ebdB(index,:)= 10*log10(abs(eet)./et); % eqn 4.9c
   IbdB(index,:)= 10*log10(abs(Iet)./It);
end

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias compare pg 54');
ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
```

```
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 55
clc
% close all
clear

ka=.01:.01:.5;
R=.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

for index=1:length(dP)
   Pm1=dM(index)*exp(j*dP(index))*Ps(ka,theta,R);
   Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance-
I think he is wrong


   % Estimate of Pressure, velocity at center
   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*3/2*ka); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

   % Estimate of NRG, Intensity at center
   eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
   Iet=1/2*real(Pec.*conj(Vet));

   UbdB(index,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
   TbdB(index,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
   ebdB(index,:)= 10*log10(abs(eet)./et); % eqn 4.9c
   IbdB(index,:)= 10*log10(abs(Iet)./It);
end

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias compare pg 55');
ylim([-3 3])
```

```
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
ylim([0 40])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
ylim([-30 20])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 56
clc
% close all
clear

ka=.01:.01:.5;
R=-.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

for index=1:length(dP)
   Pm1=dM(index)*exp(j*dP(index))*Ps(ka,theta,R);
   Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance-
I think he is wrong


   % Estimate of Pressure, velocity at center
   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*3/2*ka); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

   % Estimate of NRG, Intensity at center
   eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
   Iet=1/2*real(Pec.*conj(Vet));

   UbdB(index,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
```

```
    TbdB(index,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
    ebdB(index,:)= 10*log10(abs(eet)./et); % eqn 4.9c
    IbdB(index,:)= 10*log10(abs(Iet)./It);
end

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias compare pg 56');
ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
```

```
close all
clc
clear

rho=1.21;
c=343;
ka=[.5,1,1.5,2];
theta=0:.01:pi;
R=[0,.97,-0.97];
figure;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

for Rindex=1:length(R)
   Pc=1;  % eqn 4.10a
   Vt=1*cos(theta)/(rho*c); % eqn 4.10b
   et=1/(12*rho*c^2)*(1+3*cos(theta).*cos(theta)); % eqn 4.10c
   It=cos(theta)/(2*rho*c); % eqn 4.10d

   Pm2= 1; % Pm2 is the pressure at microphone 2
   dM=1; % sensitivity mismatch
   dP=.25; % phase mismatch

   dMdb=20*log10(dM); % equation 4.3
   Pm1=dM*exp(j*dP)*Pm2; % equation 4.1

   Ut=Pc*conj(Pc)/(12*rho*c^2); % Equation 4.4a, Pc is the complex pressure, Ut stands for U as a
function of theta
   Tt=rho*Vt.*conj(Vt)/4; % equation 4.4b
   et= Ut+Tt; % equation 4.4c
   It= 1/2*real(Pc*conj(Vt)); % eqn 4.4d

   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
   Pf=exp(-j*ka'*cos(theta))+R(Rindex)*exp(j*ka'*cos(theta)); % eqn 4.14
   Ps_hat=ps(ka,theta,R(Rindex)); % eqn 4.13
   Pex=Ps_hat./Pf.';
   subplot(3,2,2*Rindex-1)
   hold on
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})
   plot(180/pi*theta,20*log10(abs(Pex)));xlabel('\theta (deg)');
   %        plot(180/pi*theta,20*log10(abs(Pex)),char(S(kaindex)));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})

   hold off
   subplot(3,2,2*Rindex)
   hold on
   %        plot(180/pi*theta,180/pi*angle(Pex),char(S(kaindex)));xlabel('\theta (deg)');
   plot(180/pi*theta,180/pi*angle(Pex));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   hold off
   legend('ka=.5','ka=1','ka=1.5','ka=2')
   hold off
end % end R loop
```

```
% %%%%%%%%%%%%%% Bias Errors for Two Point Sensor Page 47... Graph Comparison
% on Page 48
% clc
clear
% close all
rho=1.21;
c=343;
theta=0;
kb=.01:.01:2;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

R=0;

dM=1; % sensitivity mismatch
dP=0; % phase mismatch
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta))+R*exp(j*kb*cos(theta))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta))+R*exp(-j*kb*cos(theta))); % eqn 4.24b

Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb*cos(theta)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
title('Perfect matched two microphones')
subplot(2,2,1)
plot(kb*cos(theta),UbdB);title('U Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,2)
plot(kb*cos(theta),TbdB);title('T Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,3)
plot(kb*cos(theta),ebdB);title('e Bias \theta = 0');xlabel('kb');ylabel('Bias(dB)');
subplot(2,2,4)
plot(kb*cos(theta),IbdB);title('I Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');

%%%%%Bias errors for spherical sensor Parkins PG 48 Graph comparison on
%%%%%page 49
% clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
```

```
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dM=1;
dP=0;
rho=1.21;
c=343;
R=0;
ka=.01:.01:2;
theta=0;
Pm1=dM*exp(j*dP)*Ps(ka,theta,R);
Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance- I
think he is wrong

Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Pc=1+R;

Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,2)
plot(ka,TbdB);title('T Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,3)
plot(ka,ebdB);title('e Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,4)
plot(ka,IbdB);title('I Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');


%%%%%%%%%%%%%%%%%%%%%%%%% Bias errors for mismatched microphones pg 49
%%%%%%%%%%%%%%%%%%%%%%%%% compare graph on pg 50
% close all
clear
clc
clear('UbdB','TbdB','ebdB','IbdB')

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dP=pi/180; % delta Phase
dM=10^(-1/80); % delta Magnitude
rho=1.21;
c=343;
theta=0;
kb=.01:.01:2;
ka=2/3*kb; % see bottom of page 45!!!!
R=.97;
```

```
% first determine pressure given two point sensor
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta))+R*exp(j*kb*cos(theta))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta))+R*exp(-j*kb*cos(theta))); % eqn 4.24b
% Exact quantities at center of sensor
Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d
% Estimates at center

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a problem
when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));

UbdB(1,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(1,:)= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB(1,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(1,:)= 10*log10(abs(Iet)./It); % ean 4.9d

% now determine quantities given spherical sensor
Pm1=dM*exp(j*dP)*Ps(ka,theta,R);
Pm2=Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance- I
think he is wrong
% exact acoustical values at center
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*ka); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a problem
when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB(2,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(2,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB(2,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(2,:)= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(kb,UbdB);title('U Bias');
% ylim([-3 3])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(kb,TbdB);title('T Bias');
% ylim([0 50])
```

```
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(kb,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')
subplot(2,2,4)
plot(kb,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')


%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 54
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=0;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta,R);
Pm2=ones(4,1)*Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here
may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
```

```
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 55
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

ka=.01:.01:2;
R=.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta,R);

Pm2=ones(4,1)*Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here
may be a problem when comparing sphere with 2 mics
```

```
% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([0 40])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 56
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=-.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta,R);
```

```
Pm2=ones(4,1)*Ps(ka,theta+pi,R); % Note- Parkins says it should be inputed kd where d is the seperation distance- I think he is wrong


% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
```

```
close all
clc
clear
% Note- This first section of code doesn't deviate from Parkinska2 because this is only
% looking at the excess pressure due to the sphere as a function of theta
% and ka
rho=1.21;
c=343;
ka=[.5,1,1.5,2];
theta=0:.01:pi;
R=[0,.97,-0.97];
figure;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

for Rindex=1:length(R)
   Pc=1;  % eqn 4.10a
   Vt=1*cos(theta)/(rho*c); % eqn 4.10b
   et=1/(12*rho*c^2)*(1+3*cos(theta).*cos(theta)); % eqn 4.10c
   It=cos(theta)/(2*rho*c); % eqn 4.10d

   Pm2= 1; % Pm2 is the pressure at microphone 2
   dM=1; % sensitivity mismatch
   dP=.25; % phase mismatch

   dMdb=20*log10(dM); % equation 4.3
   Pm1=dM*exp(j*dP)*Pm2; % equation 4.1

   Ut=Pc*conj(Pc)/(12*rho*c^2); % Equation 4.4a, Pc is the complex pressure, Ut stands for U as a
function of theta
   Tt=rho*Vt.*conj(Vt)/4; % equation 4.4b
   et= Ut+Tt; % equation 4.4c
   It= 1/2*real(Pc*conj(Vt)); % eqn 4.4d

   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
   Pf=exp(-j*ka'*cos(theta))+R(Rindex)*exp(j*ka'*cos(theta)); % eqn 4.14
   Ps_hat=ps(ka,theta,R(Rindex)); % eqn 4.13
   Pex=Ps_hat./Pf.';
   subplot(3,2,2*Rindex-1)
   hold on
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})
   plot(180/pi*theta,20*log10(abs(Pex)));xlabel('\theta (deg)');
   %        plot(180/pi*theta,20*log10(abs(Pex)),char(S(kaindex)));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})

   hold off
   subplot(3,2,2*Rindex)
   hold on
   %        plot(180/pi*theta,180/pi*angle(Pex),char(S(kaindex)));xlabel('\theta (deg)');
   plot(180/pi*theta,180/pi*angle(Pex));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   hold off
   legend('ka=.5','ka=1','ka=1.5','ka=2')
```

```
   hold off
end % end R loop



% %%%%%%%%%%%%% Bias Errors for Two Point Sensor Page 47... Graph Comparison
% on Page 48
% clc
clear
% close all
rho=1.21;
c=343;
theta=0;
kb=.01:.01:2;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

R=0;

dM=1; % sensitivity mismatch
dP=0; % phase mismatch    %%%%%%%%% note- wierd +.9553 comes from ortho geometry- see notes
18/jun/04
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta+.9553))+R*exp(j*kb*cos(theta+.9553))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta-.9553))+R*exp(-j*kb*cos(theta-.9553))); % eqn 4.24b

Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb*cos(theta)*1/sqrt(3)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here
may be a problem when comparing sphere with 2 mics
% added 2sqrt(6)/3 because that is the long seperation distance on the
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
title('Perfect matched two microphones')
subplot(2,2,1)
plot(kb*cos(theta),UbdB);title('U Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,2)
plot(kb*cos(theta),TbdB);title('T Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,3)
plot(kb*cos(theta),ebdB);title('e Bias \theta = 0');xlabel('kb');ylabel('Bias(dB)');
subplot(2,2,4)
plot(kb*cos(theta),IbdB);title('I Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');

%%%%%Bias errors for spherical sensor Parkins PG 48 Graph comparison on
%%%%%page 49
% clc
% close all
```

```
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dM=1;
dP=0;
rho=1.21;
c=343;
R=0;
ka=.01:.01:2;
theta=0;
Pm1=dM*exp(j*dP)*Ps(ka,theta+.9553,R);
Pm2=Ps(ka,theta+2.1863,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong

Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Pc=1+R;

Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2*1/sqrt(3)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may
be a problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,2)
plot(ka,TbdB);title('T Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,3)
plot(ka,ebdB);title('e Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,4)
plot(ka,IbdB);title('I Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');


%%%%%%%%%%%%%%%%%%%%%%%%% Bias errors for mismatched microphones pg 49
%%%%%%%%%%%%%%%%%%%%%%%%% compare graph on pg 50
% close all
clear
clc
clear('UbdB','TbdB','ebdB','IbdB')

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dP=pi/180; % delta Phase
dM=10^(-1/80); % delta Magnitude
rho=1.21;
c=343;
theta=0;
```

```
kb=.01:.01:2;
ka=2/3*kb; % see bottom of page 45!!!!
R=.97;

% first determine pressure given two point sensor
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta+.9553))+R*exp(j*kb*cos(theta+.9553))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta-.9553))+R*exp(-j*kb*cos(theta-.9553))); % eqn 4.24b
% Exact quantities at center of sensor
Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d
% Estimates at center

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb*1/sqrt(3)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));

UbdB(1,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(1,:)= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB(1,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(1,:)= 10*log10(abs(Iet)./It); % ean 4.9d

% now determine quantities given spherical sensor
Pm1=dM*exp(j*dP)*Ps(ka,theta+.9553,R);
Pm2=Ps(ka,theta+2.1863,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong
% exact acoustical values at center
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*ka*1/sqrt(3)); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB(2,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(2,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB(2,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(2,:)= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(kb,UbdB);title('U Bias');
% ylim([-3 3])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
```

```
subplot(2,2,2)
plot(kb,TbdB);title('T Bias');
% ylim([0 50])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(kb,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')
subplot(2,2,4)
plot(kb,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')


%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 54
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=0;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+.9553,R);
Pm2=ones(4,1)*Ps(ka,theta+2.1863,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*1/sqrt(3))); % eqn 4.8 (j*rho*c*2*k*d) **************
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
```

```
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 55
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

ka=.01:.01:2;
R=.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+.9553,R);
Pm2=ones(4,1)*Ps(ka,theta+2.1863,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
```

```
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*1/sqrt(3))); % eqn 4.8 (j*rho*c*2*k*d) **************
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([0 40])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 56
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=-.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
```

```
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+.9553,R);
Pm2=ones(4,1)*Ps(ka,theta+2.1863,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*1/sqrt(3))); % eqn 4.8 (j*rho*c*2*k*d) **********
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
```

```
close all
clc
clear
% Note- This first section of code doesn't deviate from Parkinska2 because this is only
% looking at the excess pressure due to the sphere as a function of theta
% and ka
rho=1.21;
c=343;
ka=[.5,1,1.5,2];

theta=0:.01:pi;
R=[0,.97,-0.97];
figure;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

for Rindex=1:length(R)
   Pc=1;  % eqn 4.10a
   Vt=1*cos(theta)/(rho*c); % eqn 4.10b
   et=1/(12*rho*c^2)*(1+3*cos(theta).*cos(theta)); % eqn 4.10c
   It=cos(theta)/(2*rho*c); % eqn 4.10d

   Pm2= 1; % Pm2 is the pressure at microphone 2
   dM=1; % sensitivity mismatch
   dP=.25; % phase mismatch

   dMdb=20*log10(dM); % equation 4.3
   Pm1=dM*exp(j*dP)*Pm2; % equation 4.1

   Ut=Pc*conj(Pc)/(12*rho*c^2); % Equation 4.4a, Pc is the complex pressure, Ut stands for U as a
function of theta
   Tt=rho*Vt.*conj(Vt)/4; % equation 4.4b
   et= Ut+Tt; % equation 4.4c
   It= 1/2*real(Pc*conj(Vt)); % eqn 4.4d

   Pec= (Pm2+Pm1)/2; % equation 4.7
   Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
   Pf=exp(-j*ka'*cos(theta))+R(Rindex)*exp(j*ka'*cos(theta)); % eqn 4.14
   Ps_hat=ps(ka,theta,R(Rindex)); % eqn 4.13
   Pex=Ps_hat./Pf.';
   subplot(3,2,2*Rindex-1)
   hold on
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})
   plot(180/pi*theta,20*log10(abs(Pex)));xlabel('\theta (deg)');
   %        plot(180/pi*theta,20*log10(abs(Pex)),char(S(kaindex)));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   %        set(gca,'LineStyleOrder', {'-',':','--','-.'})

   hold off
   subplot(3,2,2*Rindex)
   hold on
   %        plot(180/pi*theta,180/pi*angle(Pex),char(S(kaindex)));xlabel('\theta (deg)');
   plot(180/pi*theta,180/pi*angle(Pex));xlabel('\theta (deg)');
   title(['R=',num2str(R(Rindex))])
   hold off
```

```
   legend('ka=.5','ka=1','ka=1.5','ka=2')

   hold off
end % end R loop



% %%%%%%%%%%%%% Bias Errors for Two Point Sensor Page 47... Graph Comparison
% on Page 48
% clc
clear
% close all
rho=1.21;
c=343;
theta=0;
kb=.01:.01:2;
set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

R=0;

dM=1; % sensitivity mismatch
dP=0; % phase mismatch
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta+pi/6))+R*exp(j*kb*cos(theta+pi/6))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta-pi/6))+R*exp(-j*kb*cos(theta-pi/6))); % eqn 4.24b

Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb*cos(theta)*sqrt(3)/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here
may be a problem when comparing sphere with 2 mics
% added 2sqrt(6)/3 because that is the long seperation distance on the
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
title('Perfect matched two microphones')
subplot(2,2,1)
plot(kb*cos(theta),UbdB);title('U Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,2)
plot(kb*cos(theta),TbdB);title('T Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');
subplot(2,2,3)
plot(kb*cos(theta),ebdB);title('e Bias \theta = 0');xlabel('kb');ylabel('Bias(dB)');
subplot(2,2,4)
plot(kb*cos(theta),IbdB);title('I Bias');xlabel('kb cos(\theta)');ylabel('Bias(dB)');

%%%%%Bias errors for spherical sensor Parkins PG 48 Graph comparison on
%%%%%page 49
% clc
```

```
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dM=1;
dP=0;
rho=1.21;
c=343;
R=0;
ka=.01:.01:2;
theta=0;
Pm1=dM*exp(j*dP)*Ps(ka,theta+pi/6,R);
Pm2=Ps(ka,theta+5*pi/6,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong

Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pec= (Pm2+Pm1)/2; % equation 4.7
Pc=1+R;

Vet= (Pm2-Pm1)./(j*rho*c*2*ka*3/2*sqrt(3)/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may
be a problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));
UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,2)
plot(ka,TbdB);title('T Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,3)
plot(ka,ebdB);title('e Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');
subplot(2,2,4)
plot(ka,IbdB);title('I Bias, \theta = 0');xlabel('ka');ylabel('Bias(dB)');


%%%%%%%%%%%%%%%%%%%%%%%% Bias errors for mismatched microphones pg 49
%%%%%%%%%%%%%%%%%%%%%%%% compare graph on pg 50
% close all
clear
clc
clear('UbdB','TbdB','ebdB','IbdB')

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
dP=pi/180; % delta Phase
dM=10^(-1/80); % delta Magnitude
rho=1.21;
c=343;
```

```
theta=0;
kb=.01:.01:2;
ka=2/3*kb; % see bottom of page 45!!!!
R=.97;

% first determine pressure given two point sensor
Pm1=dM*exp(j*dP)*(exp(-j*kb*cos(theta+pi/6))+R*exp(j*kb*cos(theta+pi/6))); % eqn 4.24a
Pm2=(exp(j*kb*cos(theta-pi/6))+R*exp(-j*kb*cos(theta-pi/6))); % eqn 4.24b
% Exact quantities at center of sensor
Pc=1+R;  % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d
% Estimates at center

Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*kb*sqrt(3)/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4; % eqn 4.4c
Iet=1/2*real(Pec.*conj(Vet));

UbdB(1,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(1,:)= 20*log10(abs(Vet./Vt)); % eqn 4.9b
ebdB(1,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(1,:)= 10*log10(abs(Iet)./It); % ean 4.9d

% now determine quantities given spherical sensor
Pm1=dM*exp(j*dP)*Ps(ka,theta+pi/6,R);
Pm2=Ps(ka,theta+5*pi/6,R); % Note- Parkins says it should be inputed kd where d is the seperation
distance- I think he is wrong
% exact acoustical values at center
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(j*rho*c*2*ka*sqrt(3)/2); % eqn 4.8 (j*rho*c*2*k*d) ********** Here may be a
problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB(2,:)= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB(2,:)= 20*log10(abs(Vet)./Vt); % eqn 4.9b

ebdB(2,:)= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB(2,:)= 10*log10(abs(Iet)./It); % ean 4.9d

figure;
subplot(2,2,1)
plot(kb,UbdB);title('U Bias');
% ylim([-3 3])
legend('Two-Point', 'Spherical')
```

```
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(kb,TbdB);title('T Bias');
% ylim([0 50])
legend('Two-Point', 'Spherical')
xlabel('kb')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(kb,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')
subplot(2,2,4)
plot(kb,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('kb')
ylabel('Bias(dB)')
legend('Two-Point', 'Spherical')


%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 54
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=0;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+pi/6,R);
Pm2=ones(4,1)*Ps(ka,theta+5*pi/6,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*sqrt(3)/2)); % eqn 4.8 (j*rho*c*2*k*d) **************
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
```

```
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 55
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')

ka=.01:.01:2;
R=.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+pi/6,R);
Pm2=ones(4,1)*Ps(ka,theta+5*pi/6,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
```

```
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*sqrt(3)/2)); % eqn 4.8 (j*rho*c*2*k*d) **************
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([0 40])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-30 20])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')

%%%%%%%%%%%%%%%%%%%%%%%%%%%% Spherical Sensor Bias errors pg 53 compare with
%%%%%%%%%%%%%%%%%%%%%%%%%%%% plot on page 56
clc
% close all
clear

set(0,'DefaultAxesColorOrder',[0 0 0;],...
   'DefaultAxesLineStyleOrder','-|:|--|-.')
ka=.01:.01:2;
R=-.97;
theta=0;
rho=1.21;
c=343;
dP=[pi/180 -pi/180 pi/180 -pi/180]; % delta phase
dM=[10^(1/80) 10^(1/80) 10^(-1/80) 10^(-1/80)]; % delta magnitude
Pc=1+R; % eqn 4.10a
Vt=(1-R)*cos(theta)/(rho*c); % eqn 4.10b
```

```
et=1/(12*rho*c^2)*((1+R)^2+3*(1-R)^2*cos(theta).*cos(theta)); % eqn 4.10c
It=(1-(R)^2)*cos(theta)/(2*rho*c); % eqn 4.10d

Pm1=(dM.*exp(j*dP))'*Ps(ka,theta+pi/6,R);
Pm2=ones(4,1)*Ps(ka,theta+5*pi/6,R); % Note- Parkins says it should be inputed kd where d is the
seperation distance- I think he is wrong

% Estimate of Pressure, velocity at center
Pec= (Pm2+Pm1)/2; % equation 4.7
Vet= (Pm2-Pm1)./(ones(4,1)*(j*rho*c*2*3/2*ka*sqrt(3)/2)); % eqn 4.8 (j*rho*c*2*k*d) **********
Here may be a problem when comparing sphere with 2 mics

% Estimate of NRG, Intensity at center
eet=Pec.*conj(Pec)/(12*rho*c^2)+rho*Vet.*conj(Vet)/4;
Iet=1/2*real(Pec.*conj(Vet));

UbdB= 20*log10(abs(Pec)./Pc); % eqn 4.9a
TbdB= 20*log10(abs(Vet)./Vt); % eqn 4.9b
ebdB= 10*log10(abs(eet)./et); % eqn 4.9c
IbdB= 10*log10(abs(Iet)./It);

figure;
subplot(2,2,1)
plot(ka,UbdB);title('U Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,2)
plot(ka,TbdB);title('T Bias');
% ylim([-3 3])
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
xlabel('ka')
ylabel('Bias(dB)')
subplot(2,2,3)
plot(ka,ebdB);title('e Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
subplot(2,2,4)
plot(ka,IbdB);title('I Bias');
% ylim([-3 3])
xlabel('ka')
ylabel('Bias(dB)')
legend('+.25dB,+1deg', '+.25dB,-1deg', '-.25dB,+1deg', '-.25dB,-1deg')
```

```
function [probe,p1,p2,p3,p4,p5,p6,k,radius]=Pextract(filename)
% PEXTRACT  Pressure Extraction
%    [probe,p1,p2,p3,p4,p5,p6,radius] = PEXTRACT(filename) reads a text
%    file and extracts pressure data from the text file. PEXTRACT also
%    returns a string which indicates the type of probe: six mic,
%    tetrahedron, or orthogonal. PEXTRACT also returns the radius of the
%    probe.

c=343; % speed of sound

warning off MATLAB:conversionToLogical
dssdata=textread(filename,'%s');  %This reads in the desired wavefront txt file

% determine header length
H=strmatch('Units',dssdata); % H will have the indexes into dssdata where 'Units' occurs
hlength=H(2)+2; % The last element of the header is 2 elements after the second occurance of 'Units'

% determine data length
D=strmatch('size',dssdata);
dlength=str2num(char(dssdata(D(1)+1)));
hdlength=hlength+dlength; % length of one header + one data

% determine the type of probe
if any(strcmpi(dssdata(1:hlength),'six')) % determine if six microphone probe
   probe='Six';

   U=strmatch('Units',dssdata(1:hlength)); % U will have the indexs into dssdata where 'Units' occurs
   switch char(dssdata(U(2)+1)) % this switch determines how that data has been imported... as Pa, or Volts
      case 'Pascal'
         %do nothing to convert data into Pascals
         p1=str2num(char(dssdata(hlength+1:hdlength)));
         p2=str2num(char(dssdata(hlength+hdlength+1:2*hdlength)));
         p3=str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength)));
         p4=str2num(char(dssdata(hlength+3*hdlength+1:4*hdlength)));
         p5=str2num(char(dssdata(hlength+4*hdlength+1:5*hdlength)));
         p6=str2num(char(dssdata(hlength+5*hdlength+1:length(dssdata))));
      case 'Volts'
         S=strmatch('Sensitivity',dssdata);
         sensitivity=str2num(char(dssdata(S+1))); % get the sensitivity in V/Pa
         p1=(str2num(char(dssdata(hlength+1:hdlength))))/sensitivity(1);
         p2=(str2num(char(dssdata(hlength+hdlength+1:2*hdlength))))/sensitivity(2);
         p3=(str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength))))/sensitivity(3);
         p4=(str2num(char(dssdata(hlength+3*hdlength+1:4*hdlength))))/sensitivity(4);
         p5=(str2num(char(dssdata(hlength+4*hdlength+1:5*hdlength))))/sensitivity(5);
         p6=(str2num(char(dssdata(hlength+5*hdlength+1:length(dssdata)))))/sensitivity(6);
      otherwise
         disp('Houston, we have a problem!')
   end
   % perform fft and appropriate conversion into frequency domain
   p1=edfft(p1);
   p2=edfft(p2);
   p3=edfft(p3);
   p4=edfft(p4);
   p5=edfft(p5);
   p6=edfft(p6);
```

```
    % determine size of probe
    S=any(strcmpi(dssdata(1:hlength),'small')); % look for any occurance of the word small
    if S==1
       radius=.0106;
    else
       radius=.0254;
    end

elseif any(strcmpi(dssdata(1:78),'tetra')) % determine if tetrahedron probe
    probe='Tetra';
    U=strmatch('Units',dssdata(1:hlength)); % U will have the indexs into dssdata where 'Units' occurs
    switch char(dssdata(U(2)+1)) % this switch determines how that data has been imported... as Pa, or Volts
       case 'Pascal'
          %do nothing to convert data into Pascals
          p4=str2num(char(dssdata(hlength+1:hdlength)));
          p2=str2num(char(dssdata(hlength+hdlength+1:2*hdlength)));
          p3=str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength)));
          p1=str2num(char(dssdata(hlength+3*hdlength+1:4*hdlength)));
          p5=0; % must pass p5 and p6 back....
          p6=0;
       case 'Volts'
          S=strmatch('Sensitivity',dssdata);
          sensitivity=str2num(char(dssdata(S+1))); % get the sensitivity in V/Pa
          p4=(str2num(char(dssdata(hlength+1:hdlength))))/sensitivity(1);
          p2=(str2num(char(dssdata(hlength+hdlength+1:2*hdlength))))/sensitivity(2);
          p3=(str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength))))/sensitivity(3);
          p1=(str2num(char(dssdata(hlength+3*hdlength+1:length(dssdata)))))/sensitivity(4);
          p5=0; % must pass p5 and p6 back....
          p6=0;
       otherwise
          disp('Houston, we have a problem!')
    end

    % perform edfft   and appropriate conversion into frequency domain
    p1=edfft(p1);
    p2=edfft(p2);
    p3=edfft(p3);
    p4=edfft(p4);

    % determine size of probe
    S=any(strcmpi(dssdata(1:hlength),'small')); % look for any occurance of the word small
    if S==1
       radius=.00825;
    else
       radius=.0254;
    end
elseif any(strcmpi(dssdata(1:78),'ortho')) % determine if orthogonal probe
    probe='Ortho';
    U=strmatch('Units',dssdata(1:hlength)); % U will have the indexs into dssdata where 'Units' occurs
    switch char(dssdata(U(2)+1)) % this switch determines how that data has been imported... as Pa, or Volts
       case 'Pascal'
          %do nothing to convert data into Pascals
          p3=str2num(char(dssdata(hlength+1:hdlength)));
          p2=str2num(char(dssdata(hlength+hdlength+1:2*hdlength)));
          p1=str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength)));
          p4=str2num(char(dssdata(hlength+3*hdlength+1:4*hdlength)));
```

```
            p5=0; % must pass p5 and p6 back....
            p6=0;
        case 'Volts'
            S=strmatch('Sensitivity',dssdata);
            sensitivity=str2num(char(dssdata(S+1))); % get the sensitivity in V/Pa
            p3=(str2num(char(dssdata(hlength+1:hdlength))))/sensitivity(1);
            p2=(str2num(char(dssdata(hlength+hdlength+1:2*hdlength))))/sensitivity(2);
            p1=(str2num(char(dssdata(hlength+2*hdlength+1:3*hdlength))))/sensitivity(3);
            p4=(str2num(char(dssdata(hlength+3*hdlength+1:4*hdlength))))/sensitivity(4);
            p5=0; % must pass p5 and p6 back....
            p6=0;
        otherwise
            disp('Houston, we have a problem!')
    end

    % perform fft and appropriate conversion into frequency domain
    p1=edfft(p1);
    p2=edfft(p2);
    p3=edfft(p3);
    p4=edfft(p4);

    % determine size of probe
    S=any(strcmpi(dssdata(1:hlength),'small')); % look for any occurance of the word small
    if S==1
        radius=.00825;
    else
        radius=.0254;
    end
else
    disp('Error! probe type was not determined from .txt file!')
    return
end

% determine k
S=strmatch('rate',dssdata(1:hlength));
foldingfreq=str2num(char(dssdata(S+2)))/2;

f=0:foldingfreq/length(p1):foldingfreq;
f=f(1:length(f)-1);
k=2*pi*f/c;
k=k';
```

```
function [p1,p2,p3,p4]=vortho(r,k,radius,A)
% PORTHO      estimate the pressure of orthogonal probe
%     [p1,p2,p3,p4] = PORTHO(r,k,radius,A) estimates the pressure at a
%     position r, given k, radius, and source strength A. Assumes a
%     monopole source radiating into the free field. r must be a vector
%     with in [x;y;z]. Output is the pressure as measured by the four microphones

r1=[r(1)+radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic1 position
r2=[r(1)-radius/sqrt(3);r(2)+radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic2 position
r3=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic3 position
r4=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)+radius/sqrt(3)]; % mic 4 position

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
```

```
function pplot
% PPLOT     plot all three probes on the surface of one sphere
%     PPLOT Compares the geometry of all three probes visually. PPLOT calls
%     sixplot3, tetraplot3, orthoplot3.

clc
clear
close all

r=[0;0;0;];
frames=120; % number of frames per probe type

%%%% set the background color to white, invert hardcopy is to ensure matlab
%%%% exports the figure properly if exported to a different format.
set(gcf, 'color', 'white');
set(gcf, 'InvertHardCopy', 'off');
maxfig
set(gca,'Projection','orthographic')
h=sixplot3(r);
axis off
set(gca,'PlotBoxAspectRatioMode','manual','DataAspectRatioMode','manual','CameraViewAngleMode','ma
nual')

random=100*rand(1,1);

aviobj = avifile(['myavi',num2str(random),'.avi'],'fps',32);


for i=1:frames
   view(3*i,20);
%     h=gca;

%     pause(.1)
   frame=getframe(gca);
   aviobj=addframe(aviobj,frame);
end

tetraplot3(r);
axis off

for i=1:frames
   view(3*i,20)
%     pause(.1)
end

orthoplot3(r);
axis off

for i=1:frames
   view(3*i,20)
%     pause(.1)
end


for k=1:25
   h = plot(fft(eye(k+16)));
```

```
    set(h,'EraseMode','xor');
    axis equal;
    frame = getframe(gca);
    aviobj = addframe(aviobj,frame);
end

aviobj = close(aviobj);

cameratoolbar('show')
del('myavi.avi')
```

```
% PRESSURE      determine the pressure
%     PRESSURE(r,k,A) returns the complex pressure at location r given a
%     source strength of A and wavenumber k. P=A*exp(-j*k*r)/r from Kinsler
%     + Frey.

function output=pressure(r,k,A) % r should be a vector [x;y;z]
rho=1.21;
c=343;
output= (A./sqrt(dot(r,r))).*exp(-j.*k.*sqrt(dot(r,r)));
```

```
function [pcmd,dev] = printopt
%PRINTOPT Printer defaults.
%  PRINTOPT is an M-file that you or your system manager can
%  edit to indicate your default printer type and destination.
%
%  [PCMD,DEV] = PRINTOPT returns two strings, PCMD and DEV.
%  PCMD is a string containing the print command that
%  PRINT uses to spool a file to the printer. Its default is:
%
%    Unix:      lpr -s -r
%    Windows:   COPY /B LPT1:
%    Macintosh: Print -Mps
%    VMS:       PRINT/DELETE
%
%   Note: SGI and Solaris 2 users who do not use BSD printing,
%   i.e. lpr, need to edit this file and uncomment the line
%   to specify 'lp -c'.
%
%  DEV is a string containing the default device option for
%  the PRINT command. Its default is:
%
%    Unix & VMS: -dps2
%    Windows:    -dwin
%    Macintosh:  -dps2
%
%  See also PRINT.


%   Copyright 1984-2000 The MathWorks, Inc.
%   The MathWorks, Inc. grants permission for Licensee to modify
%   this file.  Licensee's use of such modified file is subject
%   to the terms and conditions of The MathWorks, Inc. Software License
%   Booklet.
% $Revision: 1.46 $  $Date: 2002/05/17 17:18:32 $

% Intialize options to empty matrices
pcmd = []; dev = [];

% This file automatically defaults to the dev and pcmd shown above
% in the online help text. If you would like to set the dev or pcmd
% default to be different from those shown above, enter it after this
% paragraph.  For example, pcmd = 'lpr -s -r -Pfred' would change the
% default for Unix systems to send output to a printer named fred.
% Note that pcmd is ignored by the Windows drivers.
% See PRINT.M for a complete list of available devices.

%---> Put your own changes to the defaults here (if needed)

orient LANDSCAPE % always print the default in Landscape orientation

% ----------- Do not modify anything below this line ----------------
% The code below this line automatically computes the defaults
% promised in the table above unless they have been overridden.

cname = computer;
```

```
if isempty(pcmd)

  % For Unix
  pcmd = 'lpr -s -r';

  % For Windows
  if strcmp(cname(1:2),'PC')
     pcmd = 'COPY /B $filename$ $portname$';
  end

  % For Macintosh
  if strcmp(cname(1:3), 'MAC'), pcmd = 'Print -Mps'; end

  % For SGI
  %if strcmp(cname(1:3),'SGI'), pcmd = 'lp -c'; end

  % For Solaris
  %if strcmp(cname,'SOL2'), pcmd = 'lp -c'; end
end

if isempty(dev)

  % For Unix, VAX/VMS, and Macintosh
  dev = '-dps2';

  % For Windows
  if strcmp(cname(1:2),'PC'), dev = '-dwin'; end
end
```

```
function Ptotal=Ps(ka,theta,R)

nmax=13; % maximum number of terms

temp=0;
for n=0:nmax
   L=legendre(n,cos(theta));
   Hderivative=-1./(2*sqrt(pi./ka).*ka.^2).*( sqrt(2)*pi.*((besselj(n+1/2,ka)) +n.*besselj(n+1/2,ka)- ...
      ka.*besselj(n-1/2,ka)- j*bessely(n+1/2,ka)-j*n.*bessely(n+1/2,ka)+j.*ka.*bessely(n-1/2,ka)));
   temp=j^n*(2*n+1)*(R+(-1)^n)*L(1,:)'*(1./(Hderivative))+temp;
end
junk=1./(j*(ka).^2);
for index=1:length(ka)
Ptotal(:,index)=junk(index)*temp(:,index);
end
```

```
function [p1,p2,p3,p4,p5,p6]=vsix(r,k,radius,A)
% PSIX    free field pressure estimate for six microphone probe
%     [p1,p2,p3,p4,p5,p6] = PSIX(r,k,radius,A) estimates the pressure
%     at the center of the probe. Six pressures are outputed.

% determine the locations of the six microphones and the vectors that point
% to them
r1=r+[ radius;0;0;];
r2=r+[-radius;0;0;];
r3=r+[0; radius;0;];
r4=r+[0;-radius;0;];
r5=r+[0;0; radius;];
r6=r+[0;0;-radius;];

% Determine the pressure at each microphone
p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
p5=pressure(r5,k,A);
p6=pressure(r6,k,A);
```

```
function [p1,p2,p3,p4]=vtetra(r,k,radius,A)
% PTETRA      estimate the pressure of tetrahedron probe
%     [p1,p2,p3,p4] = PTETRA(r,k,radius,A) estimates the pressure at the
%     microphone locations a position r, given k, radius, and source
%     strength A. Assumes a monopole source radiating into the free field.
%     r must be a vector with in [x;y;z]. Output is the complex pressure.

r1=[r(1)                ; r(2)+2/3*radius*2^(1/2)       ; r(3)-1/3*radius];
r2=[r(1)-1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r3=[r(1)+1/3*radius*6^(1/2); r(2)-1/9*3^(1/2)*radius*6^(1/2); r(3)-1/3*radius];
r4=[r(1)                ; r(2)                      ; r(3)+radius];

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
```

```
function output=seactual(k,A)
%SEACTUAL   Sphere energy actual.
%   NRG = SEACTUAL(k,A) returns the scalar energy of the plane wave
%   with source strength A.  SEACTUAL assumes a plane wave propagating
%   in the negative z direction, impinging upon a sphere.
%
%   See also SEORTHO, SESIX, SETETRA.


c=343;
rho=1.21; % density of air in kg/m^3

U=A*conj(A)/(2*rho*c^2); % estimate potential energy

T=A*conj(A)/(2*rho*c^2);

output= (T+U)*ones(1,length(k));
```

```
function [varargout]=SEe(iterations)
% SEE       Spherical Effects Error
%     [varargout]=SEE(iterations) collects the pressure, velocity,
%     intensity and energy error for the various probe designs. SSE with no argument will estimate the
errors of only one location
%     SSE(10) will estimate the error of 10 random locations with the default of radius .00825 meters. It
calls the various SE error functions
%     and plots the results. It does not take into account the difference
%     in radius between the small six microphone probe and the other small
%     radius probes. The default probe radius is .00825.
clc
close all
n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      phi=2*pi*randn(1); % generate random vector
      theta=2*pi*randn(1);
      eta=2*pi*randn(1);
      iterations=1;
      radius=.00825;
      ka=.01:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=0;
   case 1 % one argument passed in, number of iterations
      phi=2*pi*randn(1,iterations); % generate random vector
      theta=2*pi*randn(1,iterations);
      eta=2*pi*randn(1,iterations);
      radius=.00825;
      ka=.01:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=0;
   otherwise
      disp('Error-.- wrong number of input arguments')
      return
end

for index=1:iterations
   [Es(:,:,index),Et(:,:,index),Eo(:,:,index)] =
SEEERROR(phi(index),theta(index),eta(index),ka,radius,A(index),p); % free field energy error
   [Ps(:,:,index),Pt(:,:,index),Po(:,:,index)] =
SEPERROR(phi(index),theta(index),eta(index),ka,radius,A(index),p); % free field pressure error
   [Vs(:,:,index),Vt(:,:,index),Vo(:,:,index)] =
SEVERROR(phi(index),theta(index),eta(index),ka,radius,A(index),p); % free field velocity error
   [Is(:,:,index),It(:,:,index),Io(:,:,index)] =
SEIERROR(phi(index),theta(index),eta(index),ka,radius,A(index),p); % free field intensity error
end
Es=mean(Es,3);Et=mean(Et,3);Eo=mean(Eo,3); % energy mean
Ps=mean(Ps,3);Pt=mean(Pt,3);Po=mean(Po,3); % pressure mean
Vs=mean(Vs,3);Vt=mean(Vt,3);Vo=mean(Vo,3); % velocity mean
Is=mean(Is,3);It=mean(It,3);Io=mean(Io,3); % intensity mean

figure % plot Energy and Pressure Error
maxfig
```

```
H=plot(ka,Es,'m--d',ka,Ps,'m--s',ka,Et,'r-.d',ka,Pt,'r-.s',ka,Eo,'b:d',ka,Po,'b:s');
set(H,'linewidth',2);
H=legend('Energy Six Mic','Pressure Six Mic','Energy Tetra','Pressure Tetra','Energy Ortho','Pressure
Ortho',0);
set(H,'fontsize',14);
if iterations==1
   H=title(['Energy and Pressure Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Energy and Pressure Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe
radius ',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);

figure % Plot velocity Error
maxfig
hold on
H=plot(ka,Vs(1,:),'m--d',ka,Vs(2,:),'m--s',ka,Vs(3,:),'m--v');
set(H,'linewidth',2);
H=plot(ka,Vt(1,:),'r-.d',ka,Vt(2,:),'r-.s',ka,Vt(3,:),'r-.v');
set(H,'linewidth',2);
H=plot(ka,Vo(1,:),'b:d',ka,Vo(2,:),'b:s',ka,Vo(3,:),'b:v');
set(H,'linewidth',2);

H=legend('Six Mic x','Six Mic y','Six Mic z',...
   'Tetra x','Tetra y','Tetra z',...
   'Ortho x','Ortho y','Ortho z',0);
set(H,'fontsize',14);
if iterations==1
   H=title(['Velocity Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
   H=title(['Velocity Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);

figure % plot intensity error
maxfig
hold on
H=plot(ka,(Is(1,:)),'m--d',ka,(Is(2,:)),'m--s',ka,(Is(3,:)),'m--v');
set(H,'linewidth',2);
H=plot(ka,(It(1,:)),'r-.d',ka,(It(2,:)),'r-.s',ka,(It(3,:)),'r-.v');
set(H,'linewidth',2);
H=plot(ka,(Io(1,:)),'b:d',ka,(Io(2,:)),'b:s',ka,(Io(3,:)),'b:v');
set(H,'linewidth',2);

H=legend('Six Mic x','Six Mic y','Six Mic z',...
   'Tetra x','Tetra y','Tetra z',...
   'Ortho x','Ortho y','Ortho z',0);
set(H,'fontsize',14);
if iterations==1
   H=title(['Intensity Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
```

```
else
   H=title(['Intensity Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);
```

```
%  Spherical Effects Energy Density Function Help
%
%  Energy Functions
%    seactual    - Determine actual NRG at the center of probe.
%    seortho     - Determine NRG estimate w/ scattering ortho probe.
%    sesix       - Determine NRG estimate w/ scattering six mic probe.
%    setetra     - Determine NRG estimate w/ scattering tetrahedron probe.
%
%  Error Functions.
%    SEe         - Spherical Effects Error
%    SEeerror    - Plot the energy error vs. ka, where a is the radius.
%    SEierror    - Plot the intensity error vs. ka, where a is the radius.
%    SEperror    - Plot the pressure error vs. ka, where a is the radius.
%    SEverror    - Plot the velocity error vs. ka, where a is the radius.
%
%  Plotting Functions.
%    maxfig      - Maximize the current figure.
%    orthoplot3  - Plot a 3-d sphere with microphones according to the orthogonal probe.
%    orthoplot3r - Same as orthoplot3, but accepts rotations about x, y, and z axis.
%    pplot       - Plot all three probes in the same figure, animate.
%    sixplot3    - Plot a 3-d sphere with 6 microphones on the surface.
%    tetraplot3  - Plot a 3-d sphere with microphones at locations given by Ono Sokki.
%
%  Pressure Functions.
%    sportho     - Estimate the total pressure measured by orthogonal probe.
%    spsix       - Estimate the total pressure measured by six mic probe.
%    sptetra     - Estimate the total pressure measured by tetrahedron probe.
%
%  Scattering Functions
%    spherescatt - Spherical scattering on a hard sphere (Heather Smith).
%    hmodscatt   - Spherical scattering on a hard sphere (Heather Smith).
%
%  Spherical Bessel Functions
%    sphB        - Spherical Bessel Function (Heather Smith).
%    sphN        - Spherical Neuman Function (Heather Smith).
%    spherbessely- Spherical Bessel Y function (Heather Smith).
%    spherbesselj- Spherical Bessel J function (Heather Smith).
%
%  Velocity Functions.
%    svactual    - Compute actual velocity at center of sphere.
%    svortho     - Estimate velocity at the center of sphere ortho.
%    svsix       - Estimate velocity at the center of sphere w/ 6 mics.
%    svtetra     - Estimate velocity at the center of sphere tetrahderon.
%
%  Misc Functions.
%    printopt    - Set printer default to print Landscape Orientation.
%    scthree     - Compare the three probe velocity estimates w/ spherical effects.
%    sitetra     - Calculate intensity based on Ono Sokki method
%
%    See also FFedfun.

help('seedfun');
```

```
function [varargout] = seeerror(phi,theta,eta,ka,radius,A,p)
% SEEERROR  Spherical Effects Energy Error.
%    [varargout] = SEEERROR(phi,theta,eta,ka,radius,A,p) collects the energy
%    error for the three probe designs. SEEERROR then takes averages of the various
%    errors. It also plots the results.
%
%    SEEERROR with no argument will estimate the Energy error at one random location.
%    If one argument is passed in, example: SEEERROR(10), will estimate the
%    error of 10 random locations based on the default radius of .00825 meters. If six arguments are passed
in,
%    SEEERROR will estimate the energy error given phi, theta, eta
%    ka, radius, and source strength A. The seventh
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      phi=2*pi*randn(1); % generate random vector
      theta=2*pi*randn(1);
      eta=2*pi*randn(1);
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, which corresponds to the number of iterations
      iterations=phi;
      phi=2*pi*randn(iterations); % generate random vector
      theta=2*pi*randn(iterations);
      eta=2*pi*randn(iterations);
      radius=.00825; % default is the small sphere
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 5
      A= 10*randn(1)+10*randn(1)*j;
      k=ka/radius;
      iterations=1;
      p=1;
   case 6
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   case 7
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
   otherwise
      disp('Error-- too many input arguments!')
      return
end
```

```
rho=1.21; % density of air
c= 343; % speed of sound in air
for index=1:iterations

   ea(index,:)=seactual(k,A(index));
   eo(index,:)=seortho(phi(index),theta(index),eta(index),k,radius,A(index));
   es(index,:)=sesix(phi(index),theta(index),eta(index),k,radius,A(index));
   et(index,:)=setetra(phi(index),theta(index),eta(index),k,radius,A(index));
end
ees=10*log10(mean(es./ea,1)); % energy error six
eet=10*log10(mean(et./ea,1)); % energy error tetra
eeo=10*log10(mean(eo./ea,1)); % energy error ortho

if p~=0
   figure
   maxfig
   H=plot(ka,ees,'m--s',ka,eet,'r-.v',ka,eeo,'b:d');
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Energy Magnitude Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Energy Magnitude Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe
radius ',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);
   H=legend('Six Mic','Tetra','Ortho',3);
   set(H,'fontsize',14);
end

nao=nargout; % number arguments outputed
if nao~=0
   varargout={ees;eet;eeo};
end
```

```
function [varargout] = SEierror(phi,theta,eta,ka,radius,A,p)
% SEIERROR  Spherical Effects Intensity Error.
%    [varargout] = SEIERROR(phi,theta,eta,ka,radius,A,p) collects the intensity
%    error for the three probe designs. SEIERROR then takes averages of the various
%    errors. It also plots the results.
%
%    SEIERROR with no argument will estimate the intensity error at one random location.
%    If one argument is passed in, example: SEIERROR(10), will estimate the
%    error of 10 random locations based on the radius of .00825 meters. If six arguments are passed in,
%    SEIERROR will estimate the intensity error given phi, theta, eta
%    ka, radius, and source strength A. The seventh
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      phi=2*pi*randn(1); % generate random vector
      theta=2*pi*randn(1);
      eta=2*pi*randn(1);
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, which corresponds to the number of iterations
      iterations=phi;
      phi=2*pi*randn(iterations); % generate random vector
      theta=2*pi*randn(iterations);
      eta=2*pi*randn(iterations);
      radius=.00825; % default is the small sphere
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 6
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   case 7
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
   otherwise
      disp('Error-- too many input arguments!')
      return
end

rho=1.21; % density of air
c= 343; % speed of sound in air

for index=1:iterations
   [ax,ay,az]=svactual(phi(index),theta(index),eta(index),k,radius,A(index));
```

```
    for temp=1:length(k)
        [ox(temp),oy(temp),oz(temp)]=svortho(phi(index),theta(index),eta(index),k(temp),radius,A(index));
        % average pressure for each probe
        op(temp)=mean(sportho(phi(index),theta(index),eta(index),k(temp),radius,A(index)));
        sp(temp)=mean(spsix(phi(index),theta(index),eta(index),k(temp),radius,A(index)));
    end
    [sx,sy,sz]=svsix(phi(index),theta(index),eta(index),k,radius,A(index));

    % Intensity Actual
    Iax(index,:)=real(.5*A(index).*conj(ax));
    Iay(index,:)=real(.5*A(index).*conj(ay));
    Iaz(index,:)=real(.5*A(index).*conj(az));

    % Intensity Orthogonal
    Iox(index,:)=real(.5*op.*conj(ox));
    Ioy(index,:)=real(.5*op.*conj(oy));
    Ioz(index,:)=real(.5*op.*conj(oz));

    % Intensity Six
    Isx(index,:)=real(.5*sp.*conj(sx));
    Isy(index,:)=real(.5*sp.*conj(sy));
    Isz(index,:)=real(.5*sp.*conj(sz));

    % Intensity Tetra
    [Itx(index,:),Ity(index,:),Itz(index,:)]=sitetra(phi(index),theta(index),eta(index),k,radius,A(index));

end

Iesx=10*log10(mean(abs(Isx./Iax),1)); % intensity error ortho x
Iesy=10*log10(mean(abs(Isy./Iay),1)); % intensity error ortho y
Iesz=10*log10(mean(abs(Isz./Iaz),1)); % intensity error ortho z

Ietx=10*log10(mean(abs(Itx./Iax),1)); % intensity error tetra x
Iety=10*log10(mean(abs(Ity./Iay),1)); % intensity error tetra y
Ietz=10*log10(mean(abs(Itz./Iaz),1)); % intensity error tetra z

Ieox=10*log10(mean(abs(Iox./Iax),1)); % intensity error ortho x
Ieoy=10*log10(mean(abs(Ioy./Iay),1)); % intensity error ortho y
Ieoz=10*log10(mean(abs(Ioz./Iaz),1)); % intensity error ortho z

if p~=0

    figure
    maxfig
    hold on
    H=plot(ka,Iesx,'m--d',ka,Iesy,'m--s',ka,Iesz,'m--v');
    set(H,'linewidth',2);
    H=plot(ka,Ietx,'r-.d',ka,Iety,'r-.s',ka,Ietz,'r-.v');
    set(H,'linewidth',2);
    H=plot(ka,Ieox,'b:d',ka,Ieoy,'b:s',ka,Ieoz,'b:v');
    set(H,'linewidth',2);
    if iterations==1
        H=title(['Intensity Magnitude Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius),
' meters'],'fontsize',16);
    else
```

```
      H=title(['Intensity Magnitude Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe
radius ',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',0);
   set(H,'fontsize',14);
   hold off

   figure
   hold on
   H=plot(ka,180/pi*mean((angle(Iax)-angle(Isx)),1),'m--d',ka,180/pi*mean((angle(Iay)-angle(Isy)),1),'m--
s',ka,180/pi*mean((angle(Iaz)-angle(Isz)),1),'m--v');
   set(H,'linewidth',2);
   H=plot(ka,180/pi*mean((angle(Iax)-angle(Itx)),1),'r-.d',ka,180/pi*mean((angle(Iay)-angle(Ity)),1),'r-
.s',ka,180/pi*mean((angle(Iaz)-angle(Itz)),1),'r-.v');
   set(H,'linewidth',2);
   H=plot(ka,180/pi*mean((angle(Iax)-angle(Iox)),1),'b:d',ka,180/pi*mean((angle(Iay)-
angle(Ioy)),1),'b:s',ka,180/pi*mean((angle(Iaz)-angle(Ioz)),1),'b:v');
   set(H,'linewidth',2);
   xlabel('ka','fontsize',16);
   ylabel('Error in degrees','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',...
      0);
   set(H,'fontsize',14);
   if iterations==1
      H=title(['Intensity Phase Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Intensity Phase Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end

end

nao=nargout; % number arguments outputed
if nao~=0
   varargout={[Iesx;Iesy;Iesz],[Ietx;Iety;Ietz],[Ieox;Ieoy;Ieoz]};
end
```

```
function output=seortho(phi,theta,eta,k,radius,A)
%SEORTHO    Sphere Energy Orthogonal Probe.
%   NRG = SEORTHO(phi,theta,eta,k,A) returns the scalar NRG of a plane wave
%   with source strength A, rotated about x, y, and z by angles phi, theta, and eta.
%   Velocity is predicted based on a taylor series expansion in conjunction with Euler's equation.
%   Pressure is averaged to estimate the pressure at the center of the sphere.
%   k is the acoustic wave number. Velocity and pressure are used to
%   determine potential and kinetic energy.
%
%   See also SEACTUAL, SESIX, SETETRA.
c=343;
rho=1.21; % density of air in kg/m^3

for index=1:length(k)
   p(index,:)=sportho(phi,theta,eta,k(index),radius,A);
   [vx(:,index),vy(:,index),vz(:,index)]=svortho(phi,theta,eta,k(index),radius,A);
end

U=mean(p,2).*conj(mean(p,2))/(2*rho*c^2); % estimate potential energy

Tx=rho*(vx.*conj(vx))/2;
Ty=rho*(vy.*conj(vy))/2;
Tz=rho*(vz.*conj(vz))/2;
T=Tx+Ty+Tz;

output= (T'+U)';
```

```
function [varargout] = seperror(phi,theta,eta,ka,radius,A,p)
% SEPERROR  Spherical Effects Pressure Error.
%    [varargout] = SEPERROR(phi,theta,eta,ka,radius,A,p) collects the pressure
%    error for the three probe designs. SEPERROR then takes averages of the various
%    errors. It also plots the results.
%
%    SEPERROR with no argument will estimate the pressure error at one random location.
%    If one argument is passed in, example: SEPERROR(10), will estimate the
%    error of 10 random locations based on the default radius of .00825 meters.
%    If five arguments are passed in, A will be chosen at random.  If six arguments are passed in,
%    SEPERROR will estimate the pressure error given phi, theta, eta
%    ka, radius, and source strength A. The seventh
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      phi=2*pi*randn(1); % generate random vector
      theta=2*pi*randn(1);
      eta=2*pi*randn(1);
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      A=A'*ones(1,length(k));
      p=1;
   case 1 % one argument passed in, which corresponds to the number of iterations
      iterations=phi;
      phi=2*pi*randn(1,iterations); % generate random vector
      theta=2*pi*randn(1,iterations);
      eta=2*pi*randn(1,iterations);
      radius=.00825; % default is the small sphere
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      A=A'*ones(1,length(k));
      p=1;
   case 5
      k=ka/radius;
      A= 10*randn(1)+10*randn(1)*j;
      A=A'*ones(1,length(k));
      iterations=1;
      p=1;
   case 6
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      A=A'*ones(1,length(k));
      iterations=1;
      p=1;
   case 7
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      A=A'*ones(1,length(k));
```

```
        iterations=1;
    otherwise
        disp('Error-- incorrect number of input arguments!')
        return
end

rho=1.21; % density of air
c= 343; % speed of sound in air

for index=1:iterations % index is associated with the number of random iterations
    for kindex=1:length(k) % note- kindex refers to the index associated with wavenumber k
        po(index,kindex)=mean(sportho(phi(index),theta(index),eta(index),k(kindex),radius,A(index)));
        ps(index,kindex)=mean(spsix(phi(index),theta(index),eta(index),k(kindex),radius,A(index)));
        pt(index,kindex)=mean(sptetra(phi(index),theta(index),eta(index),k(kindex),radius,A(index)));
    end
end
pes=20*log10(mean(abs(ps./A),1)); % pressure error six
pet=20*log10(mean(abs(pt./A),1)); % pressure error tetra
peo=20*log10(mean(abs(po./A),1)); % pressure error ortho

if p~=0
    figure
    maxfig
    hold on
    H=plot(ka,pes,'m--d',ka,pet,'r-.d',ka,peo,'b:d');
    set(H,'linewidth',2);
    if iterations==1
        H=title(['Pressure Magnitude Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius),
' meters'],'fontsize',16);
    else
        H=title(['Pressure Magnitude Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe
radius ',num2str(radius), ' meters'],'fontsize',16);
    end
    xlabel('ka','fontsize',16);
    ylabel('Error in dB','fontsize',16);
    H=legend('Six Mic','Tetra','Ortho',3);
    set(H,'fontsize',14);
end

nao=nargout; % number arguments outputed
if nao~=0
    varargout={pes;pet;peo};
end
```

```
function output=sesix(phi,theta,eta,k,radius,A)
%SESIX    Sphere Energy Six Microphone Probe.
%   NRG = SESIX(phi,theta,eta,k,A) returns the scalar NRG of a plane wave
%   with source strength A, rotated about x, y, and z by angles phi, theta, and eta.
%   Velocity is predicted based on Euler's Equation. Pressure is
%   averaged to estimate the pressure at the center of the sphere. k is the
%   acoustic wave number. Velocity and pressure are used to determine
%   potential and kinetic energy.
%
%   See also SEACTUAL, SEORTHO, SETETRA.
c=343;
rho=1.21; % density of air in kg/m^3

for index=1:length(k)
   p(index,:)=spsix(phi,theta,eta,k(index),radius,A);
   [vx(:,index),vy(:,index),vz(:,index)]=svsix(phi,theta,eta,k(index),radius,A);
end

U=mean(p,2).*conj(mean(p,2))/(2*rho*c^2); % estimate potential energy

Tx=rho*(vx.*conj(vx))/2;
Ty=rho*(vy.*conj(vy))/2;
Tz=rho*(vz.*conj(vz))/2;
T=Tx+Ty+Tz;

output= (T'+U)';
```

```
function output=setetra(phi,theta,eta,k,radius,A)
%SETETRA    Sphere Energy Six Microphone Probe.
%   NRG = SETETRA(phi,theta,eta,k,A) returns the scalar NRG
%   of a plane wave with source strength A, rotated about x,
%   y, and z by angles phi, theta, and eta. Velocity is
%   predicted based on Euler's Equation and the Ono Sokki
%   method. Pressure is averaged to estimate the pressure at
%   the center of the sphere. k is the acoustic wave number.
%   Velocity and pressure are used to determine potential and
%   kinetic energy.
%
%   See also SEACTUAL, SEORTHO, SESIX.
c=343;
rho=1.21; % density of air in kg/m^3

for index=1:length(k)
   p(index,:)=sptetra(phi,theta,eta,k(index),radius,A);
   [vx(:,index),vy(:,index),vz(:,index)]=svtetra(phi,theta,eta,k(index),radius,A);
end

U=mean(p,2).*conj(mean(p,2))/(2*rho*c^2); % estimate potential energy

Tx=rho*(vx.*conj(vx))/2;
Ty=rho*(vy.*conj(vy))/2;
Tz=rho*(vz.*conj(vz))/2;
T=Tx+Ty+Tz;

output= (T'+U)';
```

```
function [varargout] = severror(phi,theta,eta,ka,radius,A,p)
% SEVERROR  Spherical Effects Velocity Error.
%    [varargout] = SEVERROR(phi,theta,eta,ka,radius,A,p) collects the velocity
%    error for the three probe designs. SEVERROR then takes averages of the various
%    errors. It also plots the results.
%
%    SEVERROR with no argument will estimate the velocity error at one random location.
%    If one argument is passed in, example: SEVERROR(10), will estimate the
%    error of 10 random locations based on the default radius of .00825 meters.
%    If five arguments are passed in, A will be chosen at random.  If six arguments are passed in,
%    SEVERROR will estimate the velocity error given phi, theta, eta
%    ka, radius, and source strength A. The seventh
%    variable, p, is a flag to plot. The default is to display the plots.
%    If you don't want to see plots, set p=0

n=nargin; % n represents the number of arguments passed into the m file.

switch n
   case 0 % no arguments passed in, just estimate error at one random location
      phi=2*pi*randn(1); % generate random vector
      theta=2*pi*randn(1);
      eta=2*pi*randn(1);
      iterations=1;
      radius=.00825;
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1)+j*10*randn(1);
      p=1;
   case 1 % one argument passed in, which corresponds to the number of iterations
      iterations=phi;
      phi=2*pi*randn(1,iterations); % generate random vector
      theta=2*pi*randn(1,iterations);
      eta=2*pi*randn(1,iterations);
      radius=.00825; % default is the small sphere
      ka=.1:.1:2;
      k=ka/radius;
      A=10*randn(1,iterations)+j*10*randn(1,iterations);
      p=1;
   case 5
      k=ka/radius;
      A= 10*randn(1)+10*randn(1)*j;
      iterations=1;
      p=1;
   case 6
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
      p=1;
   case 7
      % use phi, theta, eta, k, radius, A as passed in
      k=ka/radius;
      iterations=1;
   otherwise
      disp('Error-- incorrect number of input arguments!')
      return
end
```

```
rho=1.21; % density of air
c= 343; % speed of sound in air

for index=1:iterations
   [ax(index,:),ay(index,:),az(index,:)]=svactual(phi(index),theta(index),eta(index),k,radius,A(index));
   for temp=1:length(k)

[ox(index,temp),oy(index,temp),oz(index,temp)]=svortho(phi(index),theta(index),eta(index),k(temp),radius
,A(index));
   end
   [sx(index,:),sy(index,:),sz(index,:)]=svsix(phi(index),theta(index),eta(index),k,radius,A(index));
   [tx(index,:),ty(index,:),tz(index,:)]=svtetra(phi(index),theta(index),eta(index),k,radius,A(index));
end
veox=20*log10(mean(abs(ox./ax),1)); % velocity error ortho x
veoy=20*log10(mean(abs(oy./ay),1)); % velocity error ortho x
veoz=20*log10(mean(abs(oz./az),1)); % velocity error ortho x

vetx=20*log10(mean(abs(tx./ax),1)); % velocity error tetra x
vety=20*log10(mean(abs(ty./ay),1)); % velocity error tetra x
vetz=20*log10(mean(abs(tz./az),1)); % velocity error tetra x

vesx=20*log10(mean(abs(sx./ax),1)); % velocity error ortho x
vesy=20*log10(mean(abs(sy./ay),1)); % velocity error ortho x
vesz=20*log10(mean(abs(sz./az),1)); % velocity error ortho x

if p~=0

   figure
   maxfig
   hold on
   H=plot(ka,180/pi*mean((angle(ox)-angle(ax)),1),'m--d',ka,180/pi*mean((angle(oy)-angle(ay)),1),'m--
s',ka,180/pi*mean((angle(oz)-angle(az)),1),'m--v'); % y axis is a square
   set(H,'linewidth',2);
   H=plot(ka,180/pi*mean((angle(tx)-angle(ax)),1),'r-.d',ka,180/pi*mean((angle(ty)-angle(ay)),1),'r-
.s',ka,180/pi*mean((angle(tz)-angle(az)),1),'r-.v'); % z axis is a v
   set(H,'linewidth',2);
   H=plot(ka,180/pi*mean((angle(sx)-angle(ax)),1),'b:d',ka,180/pi*mean((angle(sy)-
angle(ay)),1),'b:s',ka,180/pi*mean((angle(sz)-angle(az)),1),'b:v'); % x axis is a diamond
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Velocity Phase Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
   else
      H=title(['Velocity Phase Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in degrees','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',3);
   set(H,'fontsize',14);

   figure
   maxfig
```

```
   hold on
   H=plot(ka,vesx,'m--d',ka,vesy,'m--s',ka,vesz,'m--v'); % y axis is a square
   set(H,'linewidth',2);
   H=plot(ka,vetx,'r-.d',ka,vety,'r-.s',ka,vetz,'r-.v'); % z axis is a v
   set(H,'linewidth',2);
   H=plot(ka,veox,'b:d',ka,veoy,'b:s',ka,veoz,'b:v'); % x axis is a diamond
   set(H,'linewidth',2);
   if iterations==1
      H=title(['Velocity Magnitude Error vs. ka, Spherical Sensor, 1 average, Probe radius ',num2str(radius),
' meters'],'fontsize',16);
   else
      H=title(['Velocity Magnitude Error vs. ka, Spherical Sensor, ',num2str(iterations) ,' averages, Probe
radius ',num2str(radius), ' meters'],'fontsize',16);
   end
   xlabel('ka','fontsize',16);
   ylabel('Error in dB','fontsize',16);
   H=legend('Six Mic x','Six Mic y','Six Mic z',...
      'Tetra x','Tetra y','Tetra z',...
      'Ortho x','Ortho y','Ortho z',3);
   set(H,'fontsize',14);
end
% actual velocity in x,y,z similar to FFverror
va(1,:)=mean(ax,1);
va(2,:)=mean(ay,1);
va(3,:)=mean(az,1);

vs(1,:)=mean(sx,1);
vs(2,:)=mean(sy,1);
vs(3,:)=mean(sz,1);

vt(1,:)=mean(tx,1);
vt(2,:)=mean(ty,1);
vt(3,:)=mean(tz,1);

vo(1,:)=mean(ox,1);
vo(2,:)=mean(oy,1);
vo(3,:)=mean(oz,1);

%%%%%%%%%%%%%%% velocity magnitude error, in the r direction,
set(0,'DefaultAxesColorOrder',[1 0 0;],'DefaultAxesLineStyleOrder','-|:|--|-.')

Vmagsix=20*log10(sqrt(dot(vs,vs)./dot(va,va)));
Vmagsix=mean(Vmagsix,3);
Vmagtetra=20*log10(sqrt(dot(vt,vt)./dot(va,va)));
Vmagtetra=mean(Vmagtetra,3);
Vmagortho=20*log10(sqrt(dot(vo,vo)./dot(va,va)));
Vmagortho=mean(Vmagortho,3);

figure
H=plot(ka,Vmagsix,'m',ka,Vmagtetra,'r',ka,Vmagortho,'b');
set(H,'linewidth',2);
if iterations==1
   H=title(['Velocity Magnitude Error vs. ka, Point Sensors, 1 average, Probe radius ',num2str(radius), '
meters'],'fontsize',16);
else
```

```
   H=title(['Velocity Magnitude Error vs. ka, Point Sensors, ',num2str(iterations) ,' averages, Probe radius
',num2str(radius), ' meters'],'fontsize',16);
end
H=legend('Six Mic','Tetra','Ortho',0);
set(H,'fontsize',14);
xlabel('ka','fontsize',16);
ylabel('Error in dB','fontsize',16);

nao=nargout; % number arguments outputed
if nao~=0
   varargout={[vesx;vesy;vesz],[vetx;vety;vetz],[veox;veoy;veoz]};
end
```

```
%SIERROR    Sphere Intensity Error.
%   SIERROR plots the error associated with estimating the intensity based
%   on the three probe designs: six mic, orthogonal, and tetrahedron. This
%   function is not yet complete.

clc
close all
clear
warning off MATLAB:divideByZero

rho=1.21; % density of air
c= 343; % speed of sound in air
A= sqrt(rho*c); % source strength A
radius=.00825;
ka=.1:.1:1;
k=ka/radius;
omega=k*c;
iteration=1;
for iteration=1:iteration
   phi=pi*rand(1);
   %     phi=0;
   theta=pi*rand(1);
   %     theta=0;
   eta=pi*rand(1);
   %     eta=0;

   for tempindex=1:length(ka)
      % determine pressure for six mic, ortho, tetra probe
      sixp(:,tempindex,iteration)=spsix(radius,k(tempindex),phi,theta,eta,A); % get the 6 pressures for each
ka value
      fouro(:,tempindex,iteration)=sportho(radius,k(tempindex),phi,theta,eta,A); % get the 4 pressures for
each ka value
      tetrap(:,tempindex,iteration)=sptetra(radius,k(tempindex),phi,theta,eta,A);
      % determine velocity for ortho probe

vo(1:3,tempindex,iteration)=svortho(radius,k(tempindex),fouro(1,tempindex,iteration),fouro(2,tempindex,it
eration),fouro(3,tempindex,iteration),fouro(4,tempindex,iteration));

vt(:,tempindex,iteration)=svtetra(radius,k(tempindex),tetrap(1,tempindex,iteration),tetrap(2,tempindex,itera
tion),tetrap(3,tempindex,iteration),tetrap(4,tempindex,iteration));
   end
   vs(:,:,iteration)=svsix(radius,ka,sixp(1,:,iteration),sixp(2,:,iteration),sixp(3,:,iteration),...
      sixp(4,:,iteration),sixp(5,:,iteration),sixp(6,:,iteration)); % determine velocity for the 6 mic probe


end

disp('end')

slog=mean(sixlog,3);
olog=mean(ortholog,3);
slog=10*log10(slog);
olog=10*log10(olog);

figure
```

```
subplot(2,1,1)
plot(ka,slog(1,:),'-v',ka,slog(2,:),'-v',ka,slog(3,:),'-v')
title('dB Error, Active Intensity, Six Mic')
xlabel('ka')
ylabel('dB')
legend('x','y','z')

subplot(2,1,2)
plot(ka,olog(1,:),'-v',ka,olog(2,:),'-v',ka,olog(3,:),'-v')
title('dB Error, Active Intensity, Ortho')
xlabel('ka')
ylabel('dB')
legend('x','y','z')

figure

subplot(2,1,1)
plot(ka,perEsix(1,:),'-v',ka,perEsix(2,:),'-v',ka,perEsix(3,:),'-v')
title('Percent Error, Active Intensity, Six Mic')
legend('x','y','z')
subplot(2,1,2)
plot(ka,perEortho(1,:),'-v',ka,perEortho(2,:),'-v',ka,perEortho(3,:),'-v')
title('Percent Error, Active Intensity, Ortho')
legend('x','y','z')

% subplot(3,1,3)
% plot(ka,perEono(1,:),'-v',ka,perEono(2,:),'-v',ka,perEono(3,:),'-v')
% title('Percent Error, Active Intensity, Ono')
% legend('x','y','z')
```

```
function [Ix,Iy,Iz]=sitetra(phi,theta,eta,k,radius,A)
% SITETRA      Intensity based on Ono Sokki Method
%    SITETRA(phi,theta,eta,k,radius,A) computes the intensity at the center of the
%    sphere based on the Ono Sokki cross spectrum method at a location r.
%    k is the wave number vector given a monopole source of strength
%    A.

c=343;
rho=1.21;
omega=k*c;

% seperation distance between microphones
d=2*sqrt(6)*radius/3; % Notebook # 3, page dated 13/april/04

for temp=1:length(k)
   % pressure for each microphone
   p=sptetra(phi,theta,eta,k(temp),radius,A);
   p1(temp)=p(1);
   p2(temp)=p(2);
   p3(temp)=p(3);
   p4(temp)=p(4);
end

G12=p1.*conj(p2)/2;
G13=p1.*conj(p3)/2;
G14=p1.*conj(p4)/2;
G23=p2.*conj(p3)/2;
G24=p2.*conj(p4)/2;
G34=p3.*conj(p4)/2;

%%%%%%%%%% Ono Sokki Stuff
Ix=-3./(omega*rho*d).*imag( (1/6)*(G34/2 - G24/2 - G23 +G12/2 - G13/2 ) );
Iy=-3./(omega*rho*d).*imag( (1/6)*(sqrt(3)*G34/6 +sqrt(3)*G24/6 - 2*sqrt(3)*G14/6 ...
   -3*sqrt(3)*G12/6 -3*sqrt(3)*G12/6 ));
Iz=-3./(omega*rho*d).*imag( (1/6)*( -sqrt(2)*G34/sqrt(3) -sqrt(2)*G24/sqrt(3) - sqrt(2)*G14/sqrt(3) ));


C1=1/2;
C2=1/2;
C3=1/2;
Vx=C1*(-2*p2+2*p3)./(j*omega*rho*d)/1.5;
Vy=C2*(4*p1/sqrt(3)-2*p2/sqrt(3)-2*p3/sqrt(3))./(j*omega*rho*d)/1.5;
Vz=C3*(-sqrt(2/3)*p1-sqrt(2/3)*p2-sqrt(2/3)*p3+3*sqrt(2/3)*p4)./(j*omega*rho*d)/1.5;


Ix=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vx));
Iy=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vy));
Iz=1/2*real( ((p1+p2+p3+p4)/4).*conj(Vz));
```

```
function H=sixplot3(r)
% SIXPLOT3      plot the six microphone probe on a sphere
%     SIXPLOT3(r) plots the six microphone probe on the surface of a sphere,
%     the center of the sphere is marked as a red circle. The solid circles
%     represent the positive direction in x, y, and z, repectively. The default
%     is to leave hold on, so that other figures can be drawn on the same sphere.


n=nargin; % n represents the number of arguments passed into the m file.

if n==0
   r=[0;0;0;];
end

radius=.00825;

micloc= [r(1)+radius,r(2),r(3);
         r(1)-radius,r(2),r(3);
         r(1),r(2)+radius,r(3);

         r(1),r(2)-radius,r(3);
         r(1),r(2),r(3)+radius;
         r(1),r(2),r(3)-radius;];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);

set(gcf,'DefaultLineEraseMode','xor')

[X,Y,Z] = SPHERE(20);
X=X*.00825;
X=X+r(1);
Y=Y*.00825;
Y=Y+r(2);
Z=Z*.00825;
Z=Z+r(3);
SURF(X,Y,Z)
alpha(0);
hold on

H=line(x(1:2),y(1:2),z(1:2));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','m','EraseMode','xor')
H=line(x(3:4),y(3:4),z(3:4));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','k')
H=line(x(5:6),y(5:6),z(5:6));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','g')
H=line(r(1),r(2),r(3));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','r')
H=title('Six Microphone Probe');
set(H,'fontsize',14)
H=legend('X axis','Y axis','Z axis');
set(H,'fontsize',14)

%%% draw a yellow circle at the center of the sphere
H=line(r(1),r(2),r(3));
```

```
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','y')
H=title('Six Microphone Probe');
set(H,'fontsize',14);

%%% draw a blue circle to indicate the positve x direction
H=line(x(1),y(1),z(1));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','m','MarkerFaceColor','m')

%%% draw a black circle to indicate the positve x direction
H=line(x(3),y(3),z(3));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','k','MarkerFaceColor','k')

%%% draw a green circle to indicate the positve x direction
H=line(x(5),y(5),z(5));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','g','MarkerFaceColor','g')

% hold off
H=gcf;
```

```
function B=sphB(m,x)
%SPHB   Spherical Bessel Function.
%   B = SPHB(m,k)
%
%   See also SPHN, SPHERBESSELJ, SPHERBESSELJ.

for n=1:length(x)
   B(n)=sqrt(pi/2/x(n))*besselj(m+.5,x(n));
end

return

function f=spherbesselj(n,x)
%SPHERBESSELJ    Spherical Bessel J Function.
%   F = SPHERBESSELJ(n,x)
%
%   See also SPHB, SPHB, SPHERBESSELY.

f=sqrt(pi./2./x).*besselj(n+1/2,x);

function f=spherbessely(n,x)
%SPHERBESSELY   Spherical Bessel Y Function.
%   F = SPHERBESSELY(n,x)
%
%   See also SPHB, SPHB, SPHERBESSELJ.

f=sqrt(pi./2./x).*bessely(n+1/2,x);
```

```
% Heather Smith code developed for Dr. Leishman's phsyics 565 class.
% problem 4.2.1 plane wave impinging on a rigid sphere of radius a
clear; close all;
psc=0;
mmax=100; %number of terms in summation
f=4000; %frequency
c=343;
a=.11;  %radius of sphere
r=2.134; %on the surface of the sphere r=a
k=2*pi*f/c;
ka=k*a;
kr=r*k;
N=100; %angular resolution
theta=0:2*pi/N:2*pi;

for m=0:mmax
   deltam=atan(((m+1)*sphB(m+1,ka)-m*sphB(m-1,ka))./(m*sphN(m-1,ka)-(m+1)*sphN(m+1,ka)));
   Pm=legendre(m,cos(theta));
   pterm=(2*m+1)*i^(m+1)*exp(-i*deltam)*sin(deltam)*Pm(1,:)*(sphB(m,kr)+i*sphN(m,kr));
   psc=psc+pterm;
end

polar(theta,abs(psc)/max(abs(psc)));
title(['|psc(theta)| for ' int2str(f) ' Hz'])

phi=0:2*pi/N:2*pi;
theta2=0:2*pi/N:pi;
[Theta,Phi]=meshgrid(theta2,phi);
newpsc=zeros(length(phi),N/2+1);

b=max(abs(psc(1:(N/2+1))));
for n=1:length(phi)
   newpsc(n,:)=abs(psc(1:(N/2+1)))/b;
end

   x=newpsc.*cos(Phi).*sin(Theta);
   y=newpsc.*sin(Phi).*sin(Theta);
   z=newpsc.*cos(Theta);
   figure
   surf(x,y,z,newpsc);
   caxis([min(min(min(newpsc))),max(max(max(newpsc)))]);
   view(3);
   colorbar('vert')
   shading interp
   axis equal
   xlabel('X');ylabel('Y');zlabel('Z');
   title(['|psc(theta,phi)| for ' int2str(f) ' Hz'])
```

```
% Heather Smith code developed for Dr. Leishman's phsyics 565 class.
% problem 4.2.1 plane wave impinging on a rigid sphere of radius a
clear; close all;
psc=0;
mmax=14; %number of terms in summation
f=4000; %frequency
c=343;
a=.11;  %radius of sphere
r=2.134; %on the surface of the sphere r=a
r=a;
k=2*pi*f/c;
radius=a;
ka=k*a;
kr=r*k;
N=100; %angular resolution
theta=0:2*pi/N:2*pi;
ka=6;
k=ka/radius
kr=ka;
f=k*c/2/pi;
for m=0:mmax
   deltam=atan(((m+1)*sphB(m+1,ka)-m*sphB(m-1,ka))./(m*sphN(m-1,ka)-(m+1)*sphN(m+1,ka)));
   Pm=legendre(m,cos(theta));
   pterm=(2*m+1)*i^(m+1)*exp(-i*deltam)*sin(deltam)*Pm(1,:)*(sphB(m,kr)+i*sphN(m,kr));
   psc=psc+pterm;
end

polar(theta,abs(psc)/max(abs(psc)));
title(['|psc(theta)| for ' int2str(f) ' Hz'])

phi=0:2*pi/N:2*pi;
theta2=0:2*pi/N:pi;
[Theta,Phi]=meshgrid(theta2,phi);
newpsc=zeros(length(phi),N/2+1);

b=max(abs(psc(1:(N/2+1))));
for n=1:length(phi)
   newpsc(n,:)=abs(psc(1:(N/2+1)))/b;
end

   x=newpsc.*cos(Phi).*sin(Theta);
   y=newpsc.*sin(Phi).*sin(Theta);
   z=newpsc.*cos(Theta);
   figure
   surf(x,y,z,newpsc);
   caxis([min(min(min(newpsc))),max(max(max(newpsc)))]);
   view(3);
   colorbar('vert')
   shading interp
   axis equal
   xlabel('X');ylabel('Y');zlabel('Z');
   title(['|psc(theta,phi)| for ' int2str(f) ' Hz'])
```

```
function N=sphN(m,x)
%SPHB   Spherical Bessel Function.
%   N = SPHN(m,k)
%
%   See also SPHB, SPHERBESSELJ, SPHERBESSELJ.

for n=1:length(x)

   N(n)=sqrt(pi/x(n)/2)*bessely(m+.5,x(n));

end

return
```

```
function Ptotal=sptetra(phi,theta,eta,k,radius,A);
%SPTETRA    Sphere Pressure Tetrahedron Probe.
%   P = SPTETRA(radius,k,phi,theta,eta,A) returns a 1x4 vector that contains
%   the complex pressure, due to scattering and a plane wave, that would be
%   measured at the locations of the four microphones of the orthogonal
%   probe.
%
%   See also SPACTUAL, SPORTHO, SPSIX.

psc=0;
mmax=13;
c=343; % speed of sound in m/s^2
a=radius;  %radius of sphere
r=radius; %on the surface of the sphere r=a
kr=r*k;
ka=k*radius;
micloc=[0                , +2/3*radius*2^(1/2)       ,-1/3*radius;
       -1/3*radius*6^(1/2), -1/9*3^(1/2)*radius*6^(1/2),-1/3*radius;
       +1/3*radius*6^(1/2), -1/9*3^(1/2)*radius*6^(1/2),-1/3*radius;
       0                , 0                        ,+radius];


micloc=micloc*[cos(eta)*cos(theta)+sin(eta)*sin(theta)*sin(phi), cos(eta)*sin(theta)-
sin(eta)*cos(theta)*sin(phi), sin(eta)*cos(phi);
   -sin(theta)*cos(phi), cos(phi)*cos(theta), sin(phi);
   -sin(eta)*cos(theta)+cos(eta)*sin(theta)*sin(phi), -sin(eta)*sin(theta)-cos(eta)*cos(theta)*sin(phi),
cos(eta)*cos(phi)];

beta=atan2(sqrt(micloc(:,1).^2+micloc(:,2).^2),(micloc(:,3)));
theta=beta; % BEWARE theta is used in two different contexts!~!

for m=0:mmax
   deltam=atan(((m+1)*sphB(m+1,ka)-m*sphB(m-1,ka))./(m*sphN(m-1,ka)-(m+1)*sphN(m+1,ka)));
   Pm=legendre(m,cos(theta)); % note the addition of the phi term, assuming no change in eta or theta
   pterm=A*(2*m+1)*i^(m+1)*exp(-i*deltam)*sin(deltam)*Pm(1,:)*(sphB(m,kr)+i*sphN(m,kr));
   psc=psc+pterm;
end

Pplane=A*exp(j*k*micloc(:,3))'; % note, no exp(-jkz) because the plane wave is propogating in the
negative z
Ptotal=psc+Pplane;
```

```
function [Ux, Uy, Uz]=svactual(phi,theta,eta,k,radius,A)
%SVACTUAL   Sphere Velocity Actual.
%   [Ux, Uy, Uz] = SVACTUAL(phi,theta,eta,k,radius,A) returns the velocity of the plane wave
%   with source strength A.  SVACTUAL assumes a plane wave propagating
%   in the negative z direction, impinging upon a sphere. Phi, Theta, and
%   Eta represent rotations about the x, y, and z axes.
%
%   See also SVORTHO, SVSIX, SVTETRA.

rho=1.21; % density of air
c=343;
omega=k*c;

va=-A/(rho*c);
vactual=([0,0,va]*[ cos(eta)*cos(theta)-sin(eta)*sin(phi)*sin(theta),
cos(eta)*sin(theta)+sin(eta)*sin(phi)*cos(theta), sin(eta)*cos(phi);
   -cos(phi)*sin(theta), cos(phi)*cos(theta), -sin(phi);
   -sin(eta)*cos(theta)-cos(eta)*sin(phi)*sin(theta), -sin(eta)*sin(theta)+cos(eta)*sin(phi)*cos(theta),
cos(eta)*cos(phi) ]');


Ux=vactual(1)*ones(1,length(k));
Uy=vactual(2)*ones(1,length(k));
Uz=vactual(3)*ones(1,length(k));
% v.a=v; % return the velocity as a struct
```

```
function [Ux, Uy, Uz]=svortho(phi,theta,eta,k,radius,A)
%SVORTHO   Sphere Velocity Ortho.
%   [Ux, Uy, Uz] = SVORTHO(phi,theta,eta,k,radius,A) returns the velocity of the plane wave
%   with source strength A.  SVORTHO assumes a plane wave propagating
%   in the negative z direction, impinging upon a sphere. Phi, Theta, and
%   Eta represent rotations about the x, y, and z axes.
%
%   See also SVACTUAL, SVSIX, SVTETRA.

%%%%%%%% Error checking... see if k is greater than a 1x1 matrix, if so,
%%%%%%%% break

if length(k)>1
   disp('Error! ka values in vest must be 1x1 only. Use a for loop.')
   disp('for temp=1:length(temp)')
   disp('     svtetra(r,k(temp),radius,A)')
   disp('end')
   output=('ERROR! k must be a scalar!');
   return;
end;

for index=1:length(k)
   p(index,:)=sportho(phi,theta,eta,k(index),radius,A);
end
p1=p(:,1).';
p2=p(:,2).';
p3=p(:,3).';
p4=p(:,4).';

ka=k*radius;
r1=[radius/sqrt(3);-radius/sqrt(3);-radius/sqrt(3)]; % mic1 position
r2=[-radius/sqrt(3);radius/sqrt(3);-radius/sqrt(3)]; % mic2 position
r3=[-radius/sqrt(3);-radius/sqrt(3);-radius/sqrt(3)]; % mic3 position
r4=[-radius/sqrt(3);-radius/sqrt(3);radius/sqrt(3)]; % mic 4 position

rho=1.21; % density of air
c=343;
omega=k*c;

%On Axis Velocity Values

Ux13=(p1-p3)./(j*rho.*omega*(r1(1)-r3(1))  ); % This vector points in the - x direction
Uy23=(p2-p3)./(j*rho.*omega*(r2(2)-r3(2))  ); % This vector points in the - y direction
Uz43=(p4-p3)./(j*rho.*omega*(r4(3)-r3(3))  ); % This vector points in the - z direction
Ux31=-Ux13; % This vector points in the + x direction
Uy32=-Uy23; % This vector points in the + y direction
Uz34=-Uz43; % This vector points in the + z direction

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

% In the 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) > 85 & abs(phadiffxy) < 275)
   theta=-theta;
end

U12=(p1-p2)./(j*rho.*omega*sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2)); % This vector points in the + y - x
direction

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

% In the XZ Plane
U41=(p4-p1)./(j*rho.*omega*sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2)); % This vector points in the - z + x
direction
U14=-U41;

% 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) > 85 & abs(phadiffxz) < 275)
   betaxz=-betaxz;
end
betaa=3*pi/4-betaxz;
U14p=U14./cos(betaa);
Ux14=U14p.*cos(betaxz);
Uz14=U14p.*sin(betaxz);

% In the YZ Plane
U24=(p2-p4)./(j*rho.*omega*sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2)); % This vector points in the - y + z
direction

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

% if the phase difference is between 90 and 270 degrees, change the sign of
% alphayz

if ( abs(phadiffyz) > 85 & abs(phadiffyz) < 275)
   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)/1.5;
Uy=-(Uy12-Uy32+Uy24)/1.5;
Uz=-(Uz14-Uz34+Uz24)/1.5;
```

```
function [Ux, Uy, Uz]=svsix(phi,theta,eta,k,radius,A)
%SVSIX   Sphere Velocity Six.
%   [Ux, Uy, Uz] = SVSIX(phi,theta,eta,k,radius,A) returns the velocity of the plane wave
%   with source strength A.  SVSIX assumes a plane wave propagating
%   in the negative z direction, impinging upon a sphere. Phi, Theta, and
%   Eta represent rotations about the x, y, and z axes.
%
%   See also SVACTUAL, SVORTHO, SVTETRA.

rho=1.21; % density of air in kg/m^3
c=343;
omega=k*c; % angular velocity

for index=1:length(k)
   p(index,:)=spsix(phi,theta,eta,k(index),radius,A);
end
p1=p(:,1).';
p2=p(:,2).';
p3=p(:,3).';
p4=p(:,4).';
p5=p(:,5).';
p6=p(:,6).';

% determine the particle velocity at the center of the probe
Ux=-(p2-p1)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - x direction
Uy=-(p4-p3)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - y direction
Uz=-(p6-p5)./(j*rho.*omega*(2*radius))/1.5; % This vector points in the - z direction
```

```
function [Ux, Uy, Uz]=svtetra(phi,theta,eta,k,radius,A)
%SVTETRA   Sphere Velocity Tetrahedron.
%   [Ux, Uy, Uz] = SVTETRA(phi,theta,eta,k,radius,A) returns the velocity of the plane wave
%   with source strength A.  SVTETRA assumes a plane wave propagating
%   in the negative z direction, impinging upon a sphere. Phi, Theta, and
%   Eta represent rotations about the x, y, and z axes.
%
%   See also SVACTUAL, SVORTHO, SVSIX.

% Vector notation- vector starts at second digit, ends at first digit. So,
% U12 increases from mic 2 to mic 1.   Written

for index=1:length(k)
   p(index,:)=sptetra(phi,theta,eta,k(index),radius,A);
end
p1=p(:,1).';
p2=p(:,2).';
p3=p(:,3).';
p4=p(:,4).';

rho=1.21; % density of air
c=343;
omega=k*c;

d=2*sqrt(6)*radius/3; % seperation distance between microphones

C1=1/2; % difference in sign (when compared with vtetra) is due to the hermitian transpose
C2=1/2;
C3=1/2;

Ux=C1*(-2*p2+2*p3)./(j*omega*rho*d)/1.5;
Uy=C2*(4*p1/sqrt(3)-2*p2/sqrt(3)-2*p3/sqrt(3))./(j*omega*rho*d)/1.5;
Uz=C3*(-sqrt(2/3)*p1-sqrt(2/3)*p2-sqrt(2/3)*p3+3*sqrt(2/3)*p4)./(j*omega*rho*d)/1.5;
```

```
function H=tetraplot3(r)
% TETRAPLOT3        plot the tetrahedron probe on a sphere
%     TETRAPLOT3(r) plots the tetrahedron probe on the surface of a sphere,
%     the center of the sphere is marked as a red circle. TETRAPLOT3
%     returns a handle to the tetrahedron figure. The default is to leave
%     hold on, so that other figures can be drawn on the same sphere.

n=nargin; % n represents the number of arguments passed into the m file.

if n==0
   r=[0;0;0;];
end


radius=.00825;

micloc=[r(1)                  , r(2)+2/3*radius*2^(1/2)        , r(3)-1/3*radius;
       r(1)-1/3*radius*6^(1/2), r(2)-1/9*3^(1/2)*radius*6^(1/2), r(3)-1/3*radius;
       r(1)+1/3*radius*6^(1/2), r(2)-1/9*3^(1/2)*radius*6^(1/2), r(3)-1/3*radius;
       r(1)                  , r(2)                           , r(3)+radius;];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);

set(gcf,'DefaultLineEraseMode','xor')

[X,Y,Z] = SPHERE(20);
X=X*.00825;
X=X+r(1);
Y=Y*.00825;
Y=Y+r(2);
Z=Z*.00825;
Z=Z+r(3);
SURF(X,Y,Z)
alpha(0)
hold on

%%% draw a line connecting all the mics
H=line(x,y,z);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
H=line([x(1),x(4)],[y(1),y(4)],[z(1),z(4)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
H=line([x(1),x(3)],[y(1),y(3)],[z(1),z(3)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
H=line([x(2),x(4)],[y(2),y(4)],[z(2),z(4)]);
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');

%%% draw a yellow circle at the center of the sphere
H=line(r(1),r(2),r(3));
set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','y')
H=title('Tetrahedron Probe');
set(H,'fontsize',14);
% hold off
H=gcf;
```

```
function tplot
% TPLOT       plot the tetrahedron probe on a sphere
%    TPLOT plots the tetrahedron probe on the surface of a sphere,
%    the center of the sphere is marked as a red circle. TPLOT
%    returns a handle to the tetrahedron figure. The default is to leave
%    hold on, so that other figures can be drawn on the same sphere.

close all
r=[0;0;0;];

radius=1;

colors=jet(5);


micloc=[r(1)                  , r(2)+2/3*radius*2^(1/2)       , r(3)-1/3*radius;
      r(1)-1/3*radius*6^(1/2), r(2)-1/9*3^(1/2)*radius*6^(1/2), r(3)-1/3*radius;
      r(1)+1/3*radius*6^(1/2), r(2)-1/9*3^(1/2)*radius*6^(1/2), r(3)-1/3*radius;
      r(1)                  , r(2)                        , r(3)+radius;];

x=micloc(:,1);
y=micloc(:,2);
z=micloc(:,3);

[X,Y,Z] = SPHERE(20);
X=X*radius;
X=X+r(1);
Y=Y*radius;
Y=Y+r(2);
Z=Z*radius;
Z=Z+r(3);
SURF(X,Y,Z)
alpha(0)
hold on
for i=1:4
   H=plot3(x(i),y(i),z(i),'color',colors(i+1,:));
   set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','MarkerEdgeColor',colors(i+1,:));
   set(H,'MarkerFaceColor',colors(i+1,:));

end
H=legend('Mic 1','Mic 2','Mic 3','Mic 4')
set(H,'FontSize',18);

H=gca;
set(H,'FontSize',14);
H=xlabel('X Axis');
set(H,'FontSize',16);
H=ylabel('Y Axis');
set(H,'FontSize',16);
H=zlabel('Z Axis');
set(H,'FontSize',16);
H=title('Tetrahedron Probe','fontsize',16)
set(gca,'PlotBoxAspectRatioMode','manual','DataAspectRatioMode','manual','CameraViewAngleMode','ma
nual')

CAMERATOOLBAR('Show')
```

```
maxfig
axis square

% hold off
% H=gcf;
% %%% draw a line connecting all the mics
% H=line(x,y,z);
% set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
% H=line([x(1),x(4)],[y(1),y(4)],[z(1),z(4)]);
% set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
% H=line([x(1),x(3)],[y(1),y(3)],[z(1),z(3)]);
% set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
% H=line([x(2),x(4)],[y(2),y(4)],[z(2),z(4)]);
% set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','color','r');
%
% %%% draw a yellow circle at the center of the sphere
% H=line(r(1),r(2),r(3));
% set(H,'LineWidth',6,'MarkerSize',24,'Marker','o','Color','y')
% H=title('Tetrahedron Probe');
% set(H,'fontsize',14);
% % hold off
% H=gcf;
```

```
% VELOCITY      compute the velocity
%     VELOCITY(r,k,A) computes the velocity at a position r, given wave
%     number k and source strength A. Assumes a monopole source radiating
%     into the free field. r must be a vector with in [x;y;z]. Output is
%     the complex velocity in x, y, and z directions. See Kinsler + Frey,
%     eqn 5.11.8.


% inputs are vector r in [x;y;z] format and k as the wave number
% output is the velocity components in [x;y;z] form velocity(r,k,A)

function output=velocity(r,k,A) % r should be a vector [x;y;z]

pvirtual=pressure(r,k,A); % pressure at the virtual point


rho=1.21; % density of air in kg/m^3
c=343; % speed of sound in m/s
%%%%%%% Velocity in the r direction
Vr=(1-j./(k.*sqrt(dot(r,r)))).*pvirtual./(rho*c);% see Kinsler + Frey, eqn 5.11.8

%output=
%[Vr*r(1)/sqrt(dot(r,r));Vr*r(2)/sqrt(dot(r,r));Vr*r(3)/sqrt(dot(r,r));];
output= [Vr*r(1)/sqrt(dot(r,r));Vr*r(2)/sqrt(dot(r,r));Vr*r(3)/sqrt(dot(r,r))];
```

```
% VORTHO      estimate the velocity of orthogonal probe
%     VORTHO(r,k,radius,A) estimates the velocity at a position r, given ka,
%     radius, and source strength A. Assumes a monopole source radiating
%     into the free field. r must be a vector with in [x;y;z]. Output is
%     the complex velocity in x, y, and z directions. NOTE- ka must be a
%     scalar value. It cannot be a vector. If ka is a vector, VEST will
%     return an error.


% vest(r,f,radius) This function estimates the velocity between all four microphones and returns the
velocity components between all four mics
% Inputs are the vector r, radius of sphere, and the wave number k. This function calls the pressure
function.
% Output is a vector length equal to the vector wave number k.
% Vector notation- vector starts at first digit, ends at second digit. So,
% V12 increases from mic 1 to mic 2 REMEBER, vest must accept only one ka
% value at a time. It does not work when ka is a vector

function output=vortho(r,k,radius,A)

warning off MATLAB:divideByZero

%%%%%%%% Error checking... see if ka is greater than a 1x1 matrix, if so,
%%%%%%%% break

if length(k)>1
   disp('Error! ka values in vest must be 1x1 only. Vse a for loop.')
   disp('for temp=1:length(temp)')
   disp('     vest(r,ka(temp),radius,A)')
   disp('end')
   output=('ERROR! ka must be a scalar!');
   return;
end;

c=343;
omega=k*c;
rho=1.21; % density of air
format compact
format short

r1=[r(1)+radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic1 position
r2=[r(1)-radius/sqrt(3);r(2)+radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic2 position
r3=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)-radius/sqrt(3)]; % mic3 position
r4=[r(1)-radius/sqrt(3);r(2)-radius/sqrt(3);r(3)+radius/sqrt(3)]; % mic 4 position

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

%On Axis Velocity Values

Vx13=(p1-p3)./(j*rho.*omega*abs(r1(1)-r3(1))  ); % This vector points in the - x direction
Vy23=(p2-p3)./(j*rho.*omega*abs(r2(2)-r3(2))  ); % This vector points in the - y direction
Vz43=(p4-p3)./(j*rho.*omega*abs(r4(3)-r3(3))  );% This vector points in the - z direction
```

```
Vx31=-Vx13; % This vector points in the + x direction
Vy32=-Vy23; % This vector points in the + y direction
Vz34=-Vz43; % This vector points in the + z direction

% determine the phase angle of the on axis velocities
Vx31pha=angle(Vx31);
Vy32pha=angle(Vy32);
Vz34pha=angle(Vz34);

% In the XY plane

% difference in phase of Vx and Vy
phadiffxy=180/pi*(Vx31pha-Vy32pha);
theta=atan(abs(Vy32)./abs(Vx31));
% if the phase difference is between 90 and 270 degrees, change the sign of
% theta
if ( abs(phadiffxy) > 85 & abs(phadiffxy) < 275)
   theta=-theta;
end

V12=(p1-p2)./(j*rho.*omega*sqrt((r2(1)-r1(1) )^2 + (r2(2)-r1(2))^2)); % This vector points in the + y - x
direction

thetaa=3*pi/4-theta; % thetaa is the angle as measured from the diagonal. This gives the correct sign on the
diagonal
V12p=V12./cos(thetaa); % this line may not need the ./
Vy12=V12p.*sin(theta);
Vx12=V12p.*cos(theta);

% In the XZ Plane
V41=(p4-p1)./(j*rho.*omega*sqrt((r4(1)-r1(1) )^2 + (r4(3)-r1(3))^2)); % This vector points in the - z + x
direction
V14=-V41;

% difference in phase of Vx and Vz
phadiffxz=180/pi*(Vx31pha-Vz34pha); % difference in phase of Vx and Vz
betaxz=atan(abs(Vz34)./abs(Vx31)); % Vx31 points in the positive x direction, Vz34 points in the positive
z direction

% if the phase difference is between 90 and 270 degrees, change the sign of
% betaxz

if ( abs(phadiffxz) > 85 & abs(phadiffxz) < 275)
   betaxz=-betaxz;
end
betaa=3*pi/4-betaxz;
V14p=V14./cos(betaa);
Vx14=V14p.*cos(betaxz);
Vz14=V14p.*sin(betaxz);

% In the YZ Plane
V24=(p2-p4)./(j*rho.*omega*sqrt((r2(2)-r4(2) )^2 + (r2(3)-r4(3))^2)); % This vector points in the - y + z
direction

% difference in phase of Vy and Vz
phadiffyz=180/pi*(Vy32pha-Vz34pha); % difference in phase of Vy and Vz
```

```
alphayz=atan(abs(Vz34)./abs(Vy32));

% if the phase difference is between 90 and 270 degrees, change the sign of
% alphayz

% if ( abs(phadiffyz) > 90 & abs(phadiffyz) < 270)
if ( abs(phadiffyz) >= 85 & abs(phadiffyz) <= 275)

   alphayz=-alphayz;
end
alphaa=3*pi/4-alphayz;
V24p=V24./cos(alphaa);
Vz24=V24p.*sin(alphayz);
Vy24=V24p.*cos(alphayz);

%Taylor Series expansion
Vx=Vx12-Vx31+Vx14;
Vy=Vy12-Vy32+Vy24;
Vz=Vz14-Vz34+Vz24;
if isnan(Vx)
   Vx=0;
end
if isnan(Vy)
   Vy=0;
end
if isnan(Vz)
   Vz=0;
end
output=[Vx;Vy;Vz];
```

```
% VSIX    free field velocity estimate for six microphone probe
%     VSIX(r,k,radius,A) estimates the velocity at the center of the
%     sphere based on Euler's Equation. It outputs velocity in x,y,z.
%
% Six microphone technique for estimated 3d velocity. This script accepts
% inputs of r,f,radius- r being the location of the probe, f being the
% frequency of interest, and radius being the radius of the probe.

function output=vsix(r,k,radius,A)

c=343; % speed of sound in air
rho=1.21; % density of air in kg/m^3
omega=k*c; % angular velocity

% determine the locations of the six microphones and the vectors that point
% to them
r1=r+[ radius;0;0;];
r2=r+[-radius;0;0;];
r3=r+[0; radius;0;];
r4=r+[0;-radius;0;];
r5=r+[0;0; radius;];
r6=r+[0;0;-radius;];

% Determine the pressure at each microphone
p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);
p5=pressure(r5,k,A);
p6=pressure(r6,k,A);

% determine the particle velocity at the center of the probe
Ux=(p2-p1)./(j*rho.*omega*(r1(1)-r2(1))); % This vector points in the - x direction
Uy=(p4-p3)./(j*rho.*omega*(r3(2)-r4(2))); % This vector points in the - y direction
Uz=(p6-p5)./(j*rho.*omega*(r5(3)-r6(3))); % This vector points in the - z direction

output=[Ux;Uy;Uz];
```

```
% VTETRA      estimate the velocity of tetrahedron probe
%    VTETRA(r,k,radius,A) estimates the velocity at a position r, given ka,
%    radius, and source strength A. Assumes a monopole source radiating
%    into the free field. r must be a vector with in [x;y;z]. Output is
%    the complex velocity in x, y, and z directions. Velocity estimate
%    based on Ono Sokki tetrahedron intensity probe.

% This function estimates the velocity between all four microphones and returns the velocity components between all four mics
% Inputs are the vector r, radius of sphere, and the wave number k. This function calls the pressure function.
% Output is a vector length equal to the vector wave number k.
% Vector notation- vector starts at first digit, ends at second digit. So,
% U12 increases from mic 1 to mic 2

function output=vtetra(r,k,radius,A)

rho=1.21; % density of air in kg/m^3
c=343;
omega=k*c;

%%%%%%%% Error checking... see if ka is greater than a 1x1 matrix, if so,
%%%%%%%% break

if length(k)>1
   disp('Error! ka values in vest must be 1x1 only. Use a for loop.')
   disp('for temp=1:length(temp)')
   disp('     vest(r,ka(temp),radius,A)')
   disp('end')
   output=('ERROR! ka must be a scalar!');
   return;
end;

r1=[r(1)                ; r(2)+2/3*radius*2^(1/2)       ; r(3)-1/3*radius];
r2=[r(1)-(6^(1/2)/3)*radius; r(2)-(2^(1/2)/3)*radius        ; r(3)-1/3*radius];
r3=[r(1)+(6^(1/2)/3)*radius; r(2)-(2^(1/2)/3)*radius        ; r(3)-1/3*radius];
r4=[r(1)                ; r(2)                      ; r(3)+radius];

d=sqrt(sum((r1-r2).^2)); % seperation distance between microphones

p1=pressure(r1,k,A);
p2=pressure(r2,k,A);
p3=pressure(r3,k,A);
p4=pressure(r4,k,A);

C1=-1/2;
C2=-1/2;
C3=-1/2;
Vx=C1*(-2*p2+2*p3)./(j*omega*rho*d);
Vy=C2*(4*p1/sqrt(3)-2*p2/sqrt(3)-2*p3/sqrt(3))./(j*omega*rho*d);
Vz=C3*(-sqrt(2/3)*p1-sqrt(2/3)*p2-sqrt(2/3)*p3+3*sqrt(2/3)*p4)./(j*omega*rho*d);
abs(Vy);
Vy12=abs((p1-p2)*cos(pi/6)./(j*omega*d*rho));
Vy13=abs((p1-p3)*cos(pi/6)./(j*omega*d*rho));
Vy14=abs((p1-p4)./(sqrt(3)*j*omega*d*rho));
Vy42=abs((p4-p2)./(sqrt(3)*j*omega*d*rho));
```

```
Vy43=abs((p4-p3)./(sqrt(3)*j*omega*d*rho)*sin(pi/6));
Vyy=abs((Vy12+Vy13+Vy14+Vy42+Vy43)*(-1/2));
Vy-Vyy;
output=[Vx;Vy;Vz;];
```