2014-03-17

Estimating the Acoustic Power of Sources in Nonideal Enclosures Using Generalized Acoustic Energy Density

Daniel Ryan Marquez
Brigham Young University - Provo

Estimating the Acoustic Power of Sources in Nonideal Enclosures

Using Generalized Acoustic Energy Density

Daniel R. Marquez

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

Master of Science

Scott D. Sommerfeldt, Chair
Timothy W. Leishman
Jonathan D. Blotter

Department of Physics and Astronomy

Brigham Young University

March 2014

ABSTRACT

Estimating the Acoustic Power of Sources in Nonideal Enclosures
Using Generalized Acoustic Energy Density

Daniel R. Marquez
Department of Physics and Astronomy, BYU
Master of Science

Sound power measurements of acoustic sources are generally made in reverberation or anechoic chambers using acoustic pressure measurements as outlined in specific ISO or other standards. A reverberation chamber produces an approximate diffuse-field condition, wherein the sound power is determined from the spatially averaged squared pressure. An anechoic chamber produces an approximate free-field condition, wherein the sound power is estimated from squared pressure over an enveloping measurement surface. However, in many cases it is desirable to estimate sound power within nonideal semi-reverberant spaces. In these environments, both direct and reverberant energies may contribute significantly to the total acoustic field. This paper introduces two measurement methods that utilize a weighted combination of potential and kinetic energy densities, known as generalized acoustic energy density, to estimate sound power in nonideal semi-reverberant rooms. The first method employs a generalized sound power formulation, which is an adaptation to an equation developed in 1948 for semi-reverberant spaces. The second, called the two-point in situ method, is a technique based on the generalized sound power formulation for quick and accurate in situ sound power estimates. Since the generalized acoustic energy density is more spatially uniform than the squared acoustic pressure in an enclosed field, these methods have the advantage of achieving the same accuracy in sound power determination with fewer measurement positions. This thesis explores the possibility of using these new methods in place of methods outlined in current ISO standards by describing analytical, numerical, and experimental results.

Keywords: sound power, generalized energy density, semi-reverberant enclosures

ACKNOWLEDGMENTS

I would like to express gratitude to all those who have helped me complete this research and thesis. First and foremost, I appreciate my wife Janneke and son Carson for their constant support and patience throughout the entire process. I thank my advisor, Dr. Scott Sommerfeldt, for his continued guidance and support despite his very busy schedule. I appreciate Dr. Jonathan Blotter for his helpful input as part of my graduate committee. In addition, I extend special gratitude toward Dr. Timothy Leishman, of my committee, who gave me constant guidance, support, and references almost daily. I appreciate those who assisted me with the numerous measurements conducted throughout campus, namely Matt Calton, Zac Jensen, Jay Eyring, Michael Denison, Travis Hoyt, and Tim Nysetvold. Thanks to Buye Xu and Michael Muhlestein for their contributions to my work in numerical modeling and animation. I recognize Microflown Technologies and their willingness to lend me their Ultimate Sound Probe to conduct much of my work. Finally, thanks to the members of the Acoustics Research Group for their support, and to Caterpillar Inc. for funding this project.

Table of Contents

List of Tables

List of Figures

List of Acronyms and Symbols

GED……time-averaged discrete-point generalized energy density [Eq. (1.10)]
GSPF……..generalized sound power formulation [Eq. (2.3)]
KED……time-averaged discrete-point kinetic energy density [Eq. (1.8c)]
MHSE…….modified Hopkins-Stryker equation [Eq. (1.9)]
OAL……unweighted overall sound power level
PED……time-averaged discrete-point potential energy density [Eq. (1.8b)]
RMSD…… arithmetic, 1/3-octave band, root-mean-square deviation [Eq. (3.6)]
TED……time-averaged discrete-point total energy density [Eq. (1.8a)]
USP…… Microflown Ultimate Sound Probe
c…… adiabatic speed of sound
C_v……coefficient of variation [Eq. (2.2)]
d_{min}……specified minimum distance from a source beyond which the free-field component is negligible [Eq. (3.1)]
f…… Frequency (Hz)
$H_{m,n}(f)$……..frequency response function between the excitation signal and microphone output at angle (m, n)
ΔL_p……change in sound pressure level
$L_{p,r}$…… sound pressure level measured in the reverberant field
L_w……discrete-point energy density level [Eq. (2.1a)]
L_Π……sound power level
M…… power attenuation coefficient
$<p^2>_{t,V_l}$…….. temporally and locally averaged squared pressure
$<p^2(r)>_t$……time-averaged discrete-point squared pressure
r……..distance from the acoustic center of a source to a field point
r_c……critical distance [Eq. (4.12)]
R……room constant [Eq. (1.1b)]
S…… surface area of the room
$S_{m,n}$…… angle-dependent area weighting factor
T_{60}……...frequency-dependent reverberation time

$<|\vec{u}|^2>_{t,V_l}$……temporally and locally averaged squared particle velocity magnitude
$<|\vec{u}(r)|^2>_t$……..time-averaged discrete-point squared particle velocity magnitude
V……volume of the enclosure
$<w_k>_{t,V_l}$……. temporally and locally averaged kinetic energy density [Eq. (1.3c)]
$<w_p>_{t,V_l}$……..temporally and locally averaged potential energy density [Eq. (1.3b)]
$<w_t>_{t,V_l}$……..temporally and locally averaged total energy density [Eq. (1.3a)]
$<w_{g,\beta}(r)>_t$……time-averaged discrete-point generalized energy density (GED)
$<w_k(r)>_t$…….. time-averaged discrete-point kinetic energy density (KED)
$<w_p(r)>_t$……time-averaged discrete-point potential energy density (PED)
$<w_t(r)>_t$……time-averaged discrete-point total energy density (TED)
$<w_i(r)>_t$…… time-averaged *i*th type discrete-point energy density
$w_{i,ref}$……..*i*th type reference energy density [Eqs. (2.1b) and (2.1c)]
$<\alpha>_s$……... spatially averaged absorption coefficient
β……generalized energy density weighting factor
$\gamma(\theta_0, \phi_0)$…… directivity factor of the source in a specified direction
θ…… polar angle
λ…… acoustic wavelength
μ…… mean value
$<\Pi_s>_t$…… time-averaged sound power of the source
ρ_0……density of air
σ…… standard deviation
ϕ……azimuthal angle

Chapter 1

Introduction

Over the past several decades, it has become quite common for acousticians to conduct sound power measurements on acoustic sources as a way of quantifying their performance or noise characteristics. These measurements are generally made in reverberation or anechoic chambers using acoustic pressure measurements as outlined in specific ISO or other standards.[1-4] A reverberation chamber produces an approximate diffuse-field condition, wherein the sound power is determined from the spatially averaged squared pressure. An anechoic chamber produces an approximate free-field condition, wherein the sound power is estimated from squared pressure, which is proportional to acoustic intensity, over an enveloping measurement surface. However, in many cases it is desirable to estimate sound power within nonideal semi-reverberant spaces. In these environments, both direct and reverberant energies can contribute significantly to the total acoustic field.

This thesis outlines the basic concept of how energy-based acoustics can be applied to improve and simplify sound power measurements in nonideal conditions. It also provides the details of analytical, numerical, and experimental research efforts implemented to explore

determination of the sound power radiation of sources within nonideal spaces using acoustic energy density.

1.1 General Background

To help the significance of this research, this section contains a description of the basic theory of sound within semi-reverberant enclosures, the current accepted methods for determining sound power in ideal enclosures, and previous research related to energy-based acoustics.

1.1.1 Sound Power of Industrial Products

As part of the development of industrial products, many manufacturers measure and publish the radiated sound powers of their products. This is motivated by both the demands of legislation and customers. For an increasing number of customers, it is important to evaluate the noise specifications of products before a purchase is made. For most manufacturers, it is important that measurements be made according to international standards.

In the past, the radiated sound pressure level was a metric sometimes reported, but sound power has become more common. Sound power is the time-averaged total acoustic energy flux through a Gaussian surface produced by a source, per unit time. Sound pressure is the effect caused by a sound source radiating power. We hear sound pressure, but it is caused by the sound power emitted from the source.

1.1.2 Sound Pressure in Semi-Reverberant Enclosures

The sound power of a source is most commonly determined from sound pressure measurements. Unfortunately, the room has an influence on the pressure response, which affects

the measurement. When a source radiates into a room, the direct energy spreads according to the inverse-square law until it encounters a surface. Depending on the absorption, a particular amount of the energy is absorbed while the remainder is reflected. All reflected energy is generally termed reverberant energy. For rooms of high absorption, the direct energy dominates and the acoustic characteristics of the source are similar to that in free-air. As the absorption decreases, greater portions of the direct energy are reflected at the surfaces. At low enough absorption levels, the reverberant energy dominates and the acoustic characteristics of the source are more dependent on the qualities of the room.[5]

When the rate of absorption at the various surfaces equals the rate at which the source radiates direct energy, the energy density in the room reaches a steady-state value. In this condition, the energy density at any point consists of both direct and reverberant energies. The direct energy is equal to that which would be established at the point if the room walls were removed and the source were radiating into free space.[5] For simplicity, the reverberant field may be considered diffuse. This means the flow in all directions is equally probable and the time-averaged energy density, spatially averaged over a volume large compared to wavelength and small compared to room dimensions, is uniform. Thus, the total energy in the room consists of both direct and reverberant energy[5] and the pressure response depends on the distance from the source and the specific characteristics of the room. This principle is depicted in Fig. 1.1, where the change in sound pressure level ΔL_p for the direct, reverberant, and total energy fields are plotted as a function of distance r from the source.[6]

It can be quite challenging to determine the sound power of a source within this type of environment because both energy fields must be considered simultaneously. For this purpose, more direct and universally accepted methods have been established in ISO and other standards

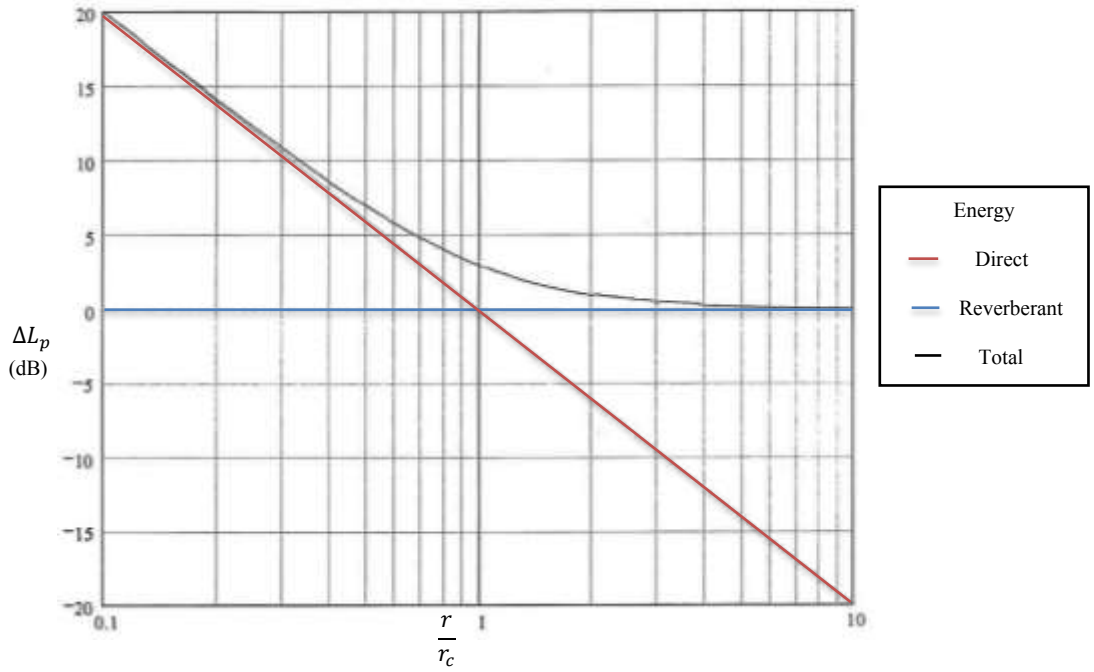

Figure 1.1 The change in sound pressure level ΔL_p relative to that at the critical distance r_c for the direct, reverberant, and total energy fields as a function of distance r from the source. The critical distance is the radial distance for a given angle where the direct and reverberant energy values are equal.

which eliminate one of the two energy fields by placing the source under test in specialized rooms. For example, anechoic rooms are designed to nearly eliminate acoustic reflections and approximate a free-field or direct-field condition. Reverberation rooms maximize acoustic reflections and produce an approximate diffuse-field condition with dominant reverberant energy at most positions in the room. It is in these specialized environments that most methods of sound power determination are established. The in situ method outlined in ISO 3747[7] is a current standard applicable in semi-reverberant spaces, and is considered in Ch. 4. Petersen[8] gives an overview of associated standards and when they are applicable.

1.1.3 Free-Field Sound Power Measurements

In free-field conditions, the sound power output of the source is directly proportional to the mean-square sound pressure averaged over an enveloping Gaussian-like surface. This is

because the latter, when properly weighted, approximates the acoustic energy flux through that surface, which in a lossless medium is proportional to the sound power output of the source. Current standards[3,4] outline the specific procedures of performing point-by-point surveys of sound pressure over measurement spheres or hemispheres, similar to that depicted in Fig. 1.2.

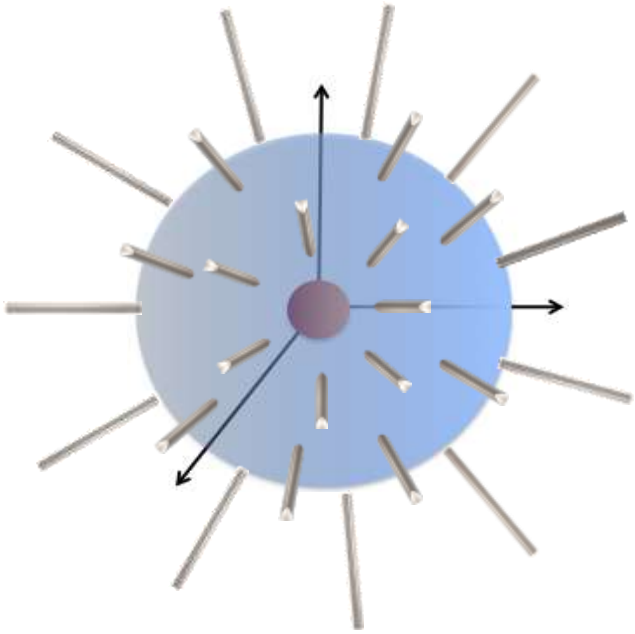

Figure 1.2 Several microphones radially positioned on a spherical measurement surface enveloping a spherical sound source.

The number of points chosen depends upon the directivity of the source. For accuracy within the higher octave bands, the standards require a minimum of twenty points and an even greater number for highly directional sources.

It is important to reiterate that these procedures require a qualified anechoic chamber to approximate a free-field condition as closely as possible. Incorporating this method in other environments will introduce error.

1.1.4 Reverberant-Field Sound Power Measurements

In a reverberation chamber, the sound field is intended to be diffuse – meaning the locally averaged energy density is distributed equally throughout the room and its flow in all directions is equally probable. In such conditions, the sound power is proportional to the spatially averaged energy density. Hence, assuming that the room is properly qualified, the sound power produced by the source may be obtained from discrete sound pressure measurements. In practice, a reverberation chamber only approximates a diffuse field, so there exists an even greater degree of spatial variation in the pressure response than would exist for the ideal case. For this reason, the standards[1,2] outline methods in which many pressure measurements can be utilized to estimate the average squared pressure response, which corresponds to that of the diffuse field condition from which the sound power can be estimated.

These methods are very specific and often require six or more microphones to be placed simultaneously within the enclosure. In addition, each microphone must be placed a minimum distance away from each other, from the source, and from the walls of the chamber. In some cases, it can be a challenge to find sufficient measurement locations that satisfy all criteria. Furthermore, this technique should be implemented in a qualified reverberation chamber to approximate a diffuse field as closely as possible.

1.1.5 The Hopkins-Stryker Equation

The principles and limitations of determining the radiated acoustic power of a source in specialized environments such as reverberation or anechoic chambers are reasonably understood. However, such ideal test facilities are rather expensive and not often readily accessible, making it something of a challenge to implement the techniques mentioned. In practice, many people

interested in sound power of sources are forced to make their measurements in factory or laboratory areas where the walls are neither acoustically dead nor completely reflecting, and would be considered nonideal for the standardized methods of determining sound power. In these enclosures, the pressure response behaves as described in Sec. 1.1.2 where the sound level does not follow the inverse-square law near the microphone positions as it should in a free field. Neither is the sound level essentially independent of location as it should be in a truly reverberant field. However, since measurements must often be made under such conditions, acousticians have worked to outline procedures by which the data may be obtained as accurately as possible.[9]

In 1948, Hopkins and Stryker[5] published a derivation of an equation predicting the temporally and locally averaged[10] total energy density, accounting for both free and reverberant contributions. The result may be expressed as

$$<w_t>_{t,V_l} = \frac{<\Pi_s>_t}{c}\left[\frac{\gamma}{4\pi r^2} + \frac{4}{R}\right], \tag{1.1a}$$

where

$$R = \frac{S<\alpha>_s}{1 - <\alpha>_s} \tag{1.1b}$$

is the room constant, $<w_t>_{t,V_l}$ is the temporally and locally averaged total energy density, $<\Pi_s>_t$ is the time-averaged sound power of the source, c is the adiabatic speed of sound, γ is the directivity factor of the source in a specified direction, r is the distance from the acoustic center of the source to the field point, S is the surface area of the room, and $<\alpha>_s$ is the spatially averaged absorption coefficient. This formulation is often termed the Hopkins-Stryker equation. While Eq. (1.1b) is a commonly used formulation for the room constant, investigators in this

field are in disagreement as to the most appropriate analytical expression. This topic is further addressed in Sec. 4.1.2.

The first term within the brackets of Eq. (1.1a) represents the direct energy component, wherein the inverse-square law is observed. Beranek[11] defines the directivity factor as the intensity radiated in the defined angle (θ, ϕ) divided by the intensity at the same point due to an omnidirectional source radiating the same sound power, which can be expressed as

$$\gamma(\theta, \phi) = \frac{<I(\theta, \phi)>_t}{<I_{omni}>_t} \ . \tag{1.2}$$

Many definitions for γ, like that used in Eq. (1.1a), are concerned with radiation in the direction of maximum response. However, Beranek defines it more generally for any arbitrary direction. In its simplest form, it can be considered a factor that weights the level of intensity radiated in the direction of the measurement angle relative to the average intensity of all angles for the particular source under test. In part of their work, Hopkins and Stryker assumed constant values of 1, 2, or 4 for the directivity factor, corresponding to a source located away from walls, near one wall, or in a dihedral corner, respectively. These values are independent of angle and frequency, but this is generally not the case.

The second term within the brackets of Eq. (1.1a) represents the reverberant energy component of the acoustic field and is a constant value dependent only upon the characteristics of the room.

The temporally and locally averaged total energy density is defined as

$$<w_t>_{t,V_l} = <w_p>_{t,V_l} + \ <w_k>_{t,V_l} \ , \tag{1.3a}$$

where

$$<w_p>_{t,V_l} = \frac{<p^2>_{t,V_l}}{2\rho_0 c^2} \tag{1.3b}$$

is the averaged potential energy density,

$$<w_k>_{t,V_l} = \frac{\rho_0}{2}<|\vec{u}|^2>_{t,V_l} \tag{1.3c}$$

is the averaged kinetic energy density, $<p^2>_{t,V_l}$ is the averaged squared pressure, ρ_0 is the density of air, and $<|\vec{u}|^2>_{t,V_l}$ is the averaged squared particle velocity magnitude.

At the introduction of the Hopkins-Stryker equation, techniques for measuring the particle velocity were not common. Acousticians worked around this issue by making a plane wave assumption within diffuse fields. In such cases, when spatially averaged over a local region sufficiently large compared to the wavelength, the temporally and locally averaged total energy density [see Eq. (1.3a)] is approximated as[12]

$$<w_t>_{t,V_l} \approx \frac{<p^2>_{t,V_l}}{\rho_0 c^2} = 2<w_p>_{t,V_l}. \tag{1.4}$$

Substituting Eq. (1.4) into Eq. (1.1a) yields a formulation that perhaps better highlights the significance of Hopkins' and Stryker's work:

$$<p^2>_{t,V_l} = \rho_0 c <\Pi_s>_t \left[\frac{\gamma}{4\pi r^2} + \frac{4}{R}\right]. \tag{1.5}$$

Their work accordingly outlines a method to determine the temporally and locally averaged squared pressure within a semi-reverberant enclosure if the sound power and directivity factor of the source are known, as well as the geometry of the room, its average absorption coefficient, and the distance from the source. However, if the goal is to determine the sound power of the source, one would rearrange Eq. (1.5) as follows:

$$<\Pi_s>_t = \frac{<p^2>_{t,V_l}}{\rho_0 c \left[\frac{\gamma}{4\pi r^2} + \frac{4}{R}\right]}. \tag{1.6}$$

In this manner, one can determine sound power radiation by measuring the locally averaged squared pressure in the room.

In practice, it can be quite challenging to make locally averaged field measurements. It has thus become common practice among acousticians to replace the ideal locally averaged quantities of the Hopkins-Stryker formula by averaging discrete-point measurements distributed throughout reverberant rooms. In this case, Eq. (1.6) becomes

$$<\Pi_s>_t = \frac{1}{I}\sum_{i=1}^{I}\frac{<p^2(r_i)>_t}{\rho_0 c\left[\frac{\gamma}{4\pi r_i^2}+\frac{4}{R}\right]}\ , \tag{1.7}$$

where $<p^2(r_i)>_t$ represents the *i*th time-averaged, *discrete-point* squared pressure value.

Wells[13] reported successful results in implementing the formulation represented in Eq. (1.7) to calculate the sound power radiation of three sources located in nonideal rooms within 3 dB, a level of accuracy adequate for many engineering applications. However, the experimental efforts essential to achieve those results were quite cumbersome, requiring several discrete measurement positions.

1.1.6 Discrete-Point Acoustic Energy Density

Most of the challenges associated with the implementation of current sound power standards and the experimental methods reported by Wells are associated with the difficulty in accurately measuring the averaged squared pressure in the room by means of distributed point measurements. Because the squared pressure within both ideal and nonideal environments is nonuniformly distributed in general, these methods require the cumbersome process of spatially averaging over a large number of discrete measurement positions. In theory, if the acoustic field in a space were distributed more uniformly, it would be possible to accurately estimate the sound power with fewer point measurements, thereby simplifying the measurement. This concept has led researchers[14-17] to investigate other methods of estimating sound power by utilizing *discrete-*

point total energy density (TED) measurements, where both the pressure and particle velocity magnitude are simultaneously determined at a point in space. Because the TED is a more spatially uniform energetic value in the acoustic field, these discrete-point measurements better approximate local averaging when compared to point measurements of the squared pressure field.

The TED is the sum of two *discrete-point* energy density quantities: potential energy density (PED) and kinetic energy density (KED). It is expressed as

$$<w_t(r)>_t = <w_p(r)>_t + <w_k(r)>_t \ , \tag{1.8a}$$

where

$$<w_p(r)>_t \ = \frac{<p^2(r)>_t}{\rho_0 c^2} \ , \tag{1.8b}$$

is the PED,

$$<w_k(r)>_t \ = \ \rho_0 <|\vec{u}(r)|^2>_t \ \ , \tag{1.8c}$$

is the KED, and $<p^2(r)>_t$ and $<|\vec{u}(r)|^2>_t$ are the time-averaged squared pressure and particle velocity magnitude at a given field point, respectively.[12]

By comparing Eqs. (1.4) and (1.8b), it is apparent that the PED approximately equals the temporally and locally averaged total energy density quantity. Thus, the original Hopkins-Stryker equation [Eq. (1.1a)] determines the temporally and locally averaged total energy density while also approximating the discrete-point PED. In this light, Eq. (1.7) can be simplified to

$$<\Pi_s>_t = \frac{1}{I}\sum_{i=1}^{I} \frac{<w_p(r_i)>_t}{\left[\frac{\gamma}{4\pi r_i^2 c} + \frac{4}{Rc}\right]} \ , \tag{1.9}$$

which demonstrates that one can solve for sound power in terms of the PED. As a result of conversion to discrete-point energy density values and spatial averaging, Eq. (1.9) will be

referred to as "the modified Hopkins-Stryker equation" (MHSE) moving forward. Since the PED is proportional to squared pressure, it exhibits the same nonuniform spatial distribution as the squared pressure field. This is because it constitutes only a portion of the total energetic information [see Eq. (1.8a)].

Since the TED constitutes all of the acoustic energy, Tichy and Baade[14] suggested in 1974 that it would have less spatial variation within enclosed sound fields than the PED. It would thus be a more advantages field point quantity to use for the determination of sound power. Cook and Schade[15] agreed with this assumption and postulated that the spatial variance of TED in a diffuse field should be approximately one-half that of the PED. In 1979, Jacobsen[16] reported normalized spatial standard deviations of 1, 0.58, and 0.58 for the PED, KED, and TED fields, respectively, based on experimental measurements in a reverberant field. In 2007, Nutter *et al*.[17] reported both the KED and TED to have demonstrated significantly lower spatial variation than the PED within multiple reverberation chambers. Their work suggests that one might benefit by using TED over PED because TED is expected to reduce measurement error and simplify measurement processes by requiring fewer measurement positions.

While both the TED and KED fields have a spatial variance that is approximately half that of PED, there still exists a degree of variation within ideal reverberant spaces. In 2011, Xu *et al*.[18] introduced an additional degree of freedom to the equation for calculating the TED in an attempt to further improve spatial uniformity. They reported that by weighting the potential and kinetic energies differently, one can establish what they refer to as a generalized energy density (GED) that can be optimized for different applications:

$$<w_{g,\beta}(r)>_t = \beta<w_p(r)>_t + (1-\beta)<w_k(r)>_t \;; \qquad (0 \le \beta \le 1)\,. \tag{1.10}$$

Their work has shown that for a diffuse field, weighting the PED by 1/4 and the KED by 3/4 (i.e., $\beta = 1/4$), the GED has a minimum spatial variance that is even less than the TED. This suggests that utilizing the GED field could facilitate even more accurate field measurements with fewer points.

1.2 General Motivations for Research

Because sound power measurements have become common for research labs and industry, research in both ideal and nonideal enclosures continues to expand. The principles and limitations of determining radiated sound power in anechoic rooms, outdoors, or in reverberation chambers are reasonably understood. By necessity, however, one must frequently carry out measurements in ordinary spaces where the limitations of current methods in these environments are less understood. The identification of shortfalls associated with previous research may lead to effective use of new technology to meet the needs of research and industry.

1.2.1 Problems or Shortfalls of Past Work

While the techniques for estimating sound power within reverberation or anechoic rooms are widely accepted, they pose significant challenges. In each case, the methods require several measurement positions, with stringent regulations on placement and quantity. They are often quite cumbersome and time consuming. In addition, expensive anechoic or reverberation rooms are required for accurate sound power determination.

The past research and experimentation of estimating sound power within semi-reverberant enclosures has been limited to sampling the acoustic pressure field, which is known

to be complex and spatially variant. For this reason, the methods required a large spatial sampling of the acoustic field in order to obtain reasonable accuracy.

Efforts to utilize the more spatially uniform acoustic fields of TED and GED have improved the accuracy of sound power estimates with fewer measurement positions, but they have been confined to highly reverberant fields only.[17,18]

1.2.2 Needs of Researchers and Industry

Due to the rigorous and time-consuming nature of current sound power measurement methods, both academic and industry groups would benefit from methods that effectively minimize time and cost. Most qualified anechoic and reverberant rooms are not readily accessible to the common researcher or professional, so in many cases, the acoustic source under test must be transported to these ideal spaces. The prominent need among those undertaking sound power estimates are methods that can not only determine sound power accurately with relatively few measurements, but also limit costs.

1.2.3 Uniqueness of Research Required to Address the Needs

The application of an energy density sensor to measure both pressure and particle velocity simultaneously is perhaps the most efficient way to measure the GED. In most cases, the limitations of energy-based acoustics are associated with the accuracy of the probe in measuring the particle velocity, especially at higher frequencies. Measurements taken in several ideal and nonideal environments will be necessary to determine the possibility and limitations of utilizing the GED field when estimating sound power within arbitrary spaces.

1.3 General Objectives of Research

The underlying question that motivates this research can be expressed as follows: is it possible to accurately and conveniently measure the radiated sound power of a source within semi-reverberant enclosures using generalized acoustic energy density? The general aims of this research are to answer this question through the following objectives:

1. Numerically and experimentally explore, develop, and validate a new method of measuring sound power within semi-reverberant enclosures using generalized acoustic energy density.
2. Determine the possibility of reducing the number of measurement positions required to accurately measure the energy field in semi-reverberant enclosures.
3. Determine the limitations of generalized energy density when it is applied to sound power estimates within semi-reverberant enclosures.

A more detailed explanation of the research objectives will be explained in subsequent chapters related to specific research methods.

1.4 Scope of Research

The method of utilizing generalized energy density to estimate the radiated power of sources within semi-reverberant enclosures has never been explored. Thus, it is natural for many questions to arise pertaining to the efficiency, accuracy, and implementation of the method. The scope of the proposed research is to investigate whether this approach is accurate and practical to any degree and is not meant to be a comprehensive study exploring all potential research questions. The numerical simulations serve to verify proof-of-concept while the experimental work is meant to verify real-world application. Both the numerical and experimental phases

explore a limited number of semi-reverberant enclosures as well as acoustic source locations. The experiments use only three small, well-behaved sources. This limits the ability to explore how the method is affected by a wide variation in source size and directivity. Hence, the results of the research do not cover a large survey of acoustic configurations. These require further investigation.

1.5 Plan of Development

This chapter has explained the general background and motivations for the research, and the general objectives and scope of the thesis. Chs. 2 and 3 provide the details of the numerical and experimental work, respectively, while Ch. 4 introduces a new technique termed the two-point in situ method. Each is structured in a format similar to the thesis as a whole. They serve as self-contained reports of individual methods, with their own insights pertaining to specific background, motivations, and objectives. In addition, each chapter explains the respective method employed, details the measured or simulated data, provides analysis of the results, and states specific conclusions. Finally, Ch. 5 restates the significant conclusions from the methods detailed in Chs. 2 through 4 and explains the impact of the work as a whole on current methods. Ch. 5 also provides recommendations for future work.

Chapter 2

A Sound Power Formulation for Semi-Reverberant Enclosures Using Generalized Acoustic Energy Density: Numerical Results

It is often useful to analytically and/or numerically explore and verify physical principles of research before beginning experimental work. This chapter outlines the numerical modeling tools developed to characterize the GED field within nonideal spaces and to investigate its usefulness in sound power estimates. It also includes a derivation of the generalized sound power formulation, which plays the central role in each of the specific methods presented in this thesis.

2.1 Specific Background

In 2010, Xu and Sommerfeldt[19] developed a hybrid modal expansion technique to model sound fields within semi-reverberant enclosures. Xu created a MATLAB script that combines the free-field Green's function and modal expansion theory in order to separate the direct and reverberant fields within the enclosure.[20] In particular, the code has the capability of modeling the global pressure and particle velocity response due to a source located within a variety of rectangular enclosures. The user has the option of choosing the position of the source, the frequency of radiation, the dimensions of the room, and the complex acoustic impedance of

parallel surfaces. His work serves as a valuable tool for modeling enclosed sound fields in an accurate and efficient manner.

2.2 Specific Motivations

The newly developed GED ($\beta = 1/4$) quantity has proven to be the most uniform energy field within reverberation chambers; however, it is hypothesized that these results are not confined to such ideal spaces. It is possible to explore this idea by characterizing the GED field within semi-reverberant spaces using the hybrid modal expansion script. It would simply require one to convert the pressure and particle velocity fields into the PED and KED fields and sum the two according to Eq. (1.10). If the theory is correct, the benefits of using the GED to estimate sound power could be extended beyond the reverberation chamber to a wide variety of more common semi-reverberant volumes. Because the GED ($\beta = 1/4$) in the enclosure is the most uniform energy quantity, this approach would allow one to more accurately measure sound power while requiring fewer measurement positions.

2.3 Specific Objectives

The following important questions will be answered by the numerical methods discussed in this chapter:

1. Is the GED field still the most uniform energy field within semi-reverberant enclosures of uniform and non-uniform absorption?
2. When compared to the number of microphone positions required to accurately estimate the mean PED within a diffuse field, can the number of measurement positions be reduced when determining the mean GED?
3. How do these results differ within various semi-reverberant enclosures?

4. Do the benefits of using GED cease once the field becomes mostly free-field in nature, and if so, what are the limits?
5. Can the GED field be measured in place of the PED field to accurately estimate sound power within semi-reverberant enclosures?

The answers to these questions could have a significant impact on the current methods of measuring sound power.

The aims of the numerical methods discussed in this chapter are to answer the above questions by completing the following objectives:

1. Develop appropriate modeling tools by adapting the MATLAB script established by Xu to generate acoustic energy density fields within rectangular rooms of varying absorption.
2. Investigate and compare the spatial variance of the four types of energy density fields in semi-reverberant enclosures, specifically PED, KED, TED, and GED.
3. Determine how the spatial variance of the GED field is affected within semi-reverberant enclosures when the value of the weighting factor β is altered.
4. Statistically determine the number of spatially averaged measurement positions that are required to adequately estimate the mean energy within enclosures of varying absorption.
5. Repeat the process and compare results for each of the four energy density fields.
6. Develop an analytical formulation that utilizes the GED in place of the PED within the MHSE equation [refer to Eq. (1.9)] to estimate sound power.
7. Determine if using the GED field to estimate sound power gives accurate results within diffuse fields.
8. If this method proves valid, explore its performance in semi-reverberant fields.

9. Characterize the accuracy, associated benefits, and limitations of utilizing the GED for sound power determination within diffuse and semi-reverberant spaces.

2.4 Methods

2.4.1 Modeling the Energy Density Fields

The hybrid modal analysis script developed by Xu[20] served as the underlying simulation tool for modeling in this work. The pressure and particle velocity fields were converted into PED and KED fields according to Eqs. (1.8b) and (1.8c), respectively. At that point, it was a relatively straight-forward process to model and explore the four types of energy density fields within numerous enclosures of varying characteristics. Differences of the PED, KED, TED, and GED fields are demonstrated in the examples of Figs. 2.1 through 2.4, which depict a plan view of the horizontal plane intersecting the sound source. The colors represent the respective energy levels on a horizontal plane through the room; thus, a more spatially uniform energy field has less variation in color.

For reference, energy density levels are calculated as follows:

$$L_{w_i} = 10\, log_{10}\left[\frac{<w_i(r)>_t}{w_{i,ref}}\right], \tag{2.1a}$$

where $<w_i(r)>_t$ is the *i*th type discrete-point energy density value (i.e., PED, KED, TED, or GED), and $w_{i,ref}$ is the *i*th type reference energy density. The reference values are expressed as

$$w_{i,ref} = \frac{(20\ \mu Pa)^2}{2\rho c^2} \qquad (PED, KED, GED), \tag{2.1b}$$

and

$$w_{i,ref} = \frac{(20\ \mu Pa)^2}{\rho c^2} \qquad (TED). \tag{2.1c}$$

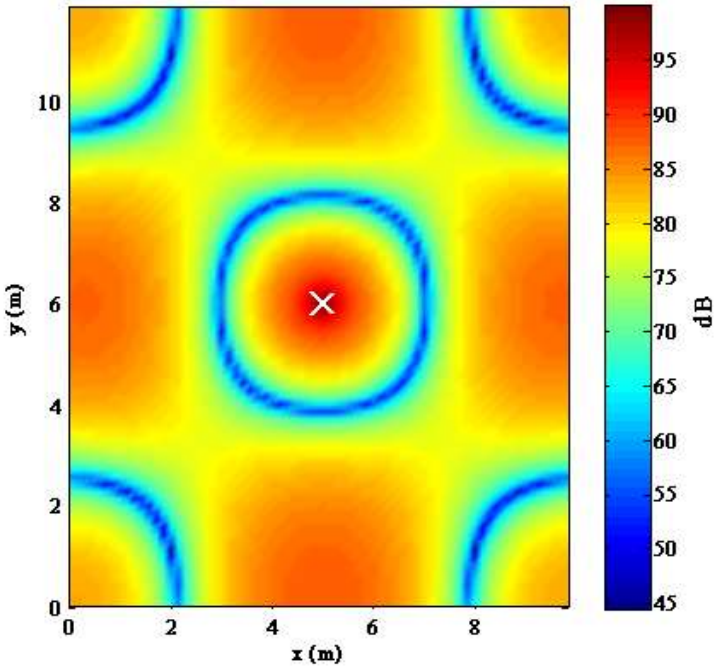

Figure 2.1 Plan view of the PED field produced by a point source radiating at 50 Hz in a rectangular enclosure of uniform absorption and dimensions 10 x 12 x 14 m. The absorption coefficient is 0.01 and the position of the source is indicated by the "X" symbol. The horizontal plane intersects the source located at x = 5 m, y = 6 m, z = 7 m. The complex impedance value of the boundaries is 200 + 340i.

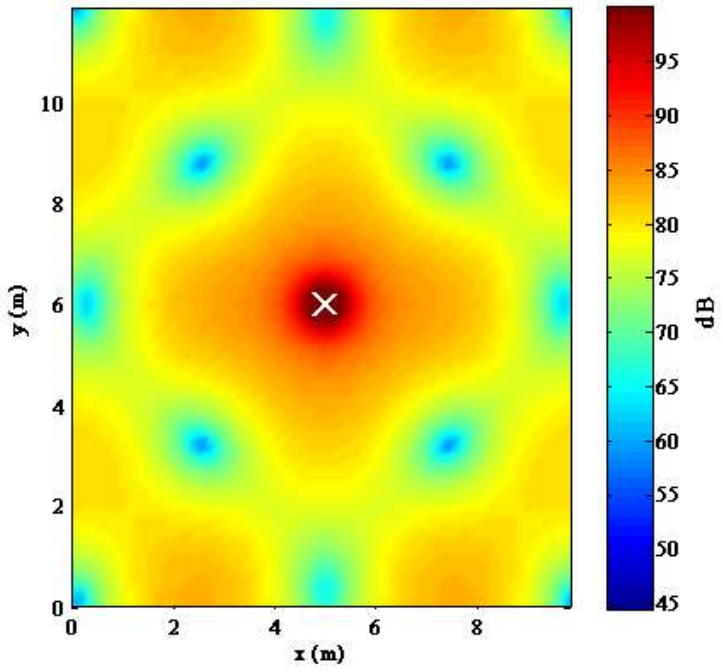

Figure 2.2 Plan view of the KED field produced by a point source radiating at 50 Hz in a rectangular enclosure of uniform absorption and dimensions 10 x 12 x 14 m. The absorption coefficient is 0.01 and the position of the source is indicated by the "X" symbol. The horizontal plane intersects the source located at x = 5 m, y = 6 m, z = 7 m. The complex impedance value of the boundaries is 200 + 340i.

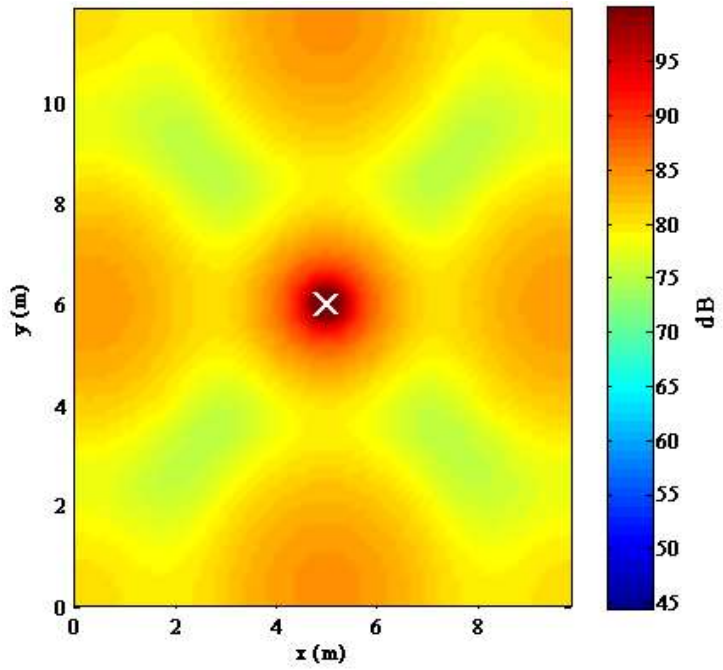

Figure 2.3 Plan view of the TED field produced by a point source radiating at 50 Hz in a rectangular enclosure of uniform absorption and dimensions 10 x 12 x 14 m. The absorption coefficient is 0.01 and the position of the source is indicated by the "X" symbol. The horizontal plane intersects the source located at x = 5 m, y = 6 m, z = 7 m. The complex impedance value of the boundaries is 200 + 340i.

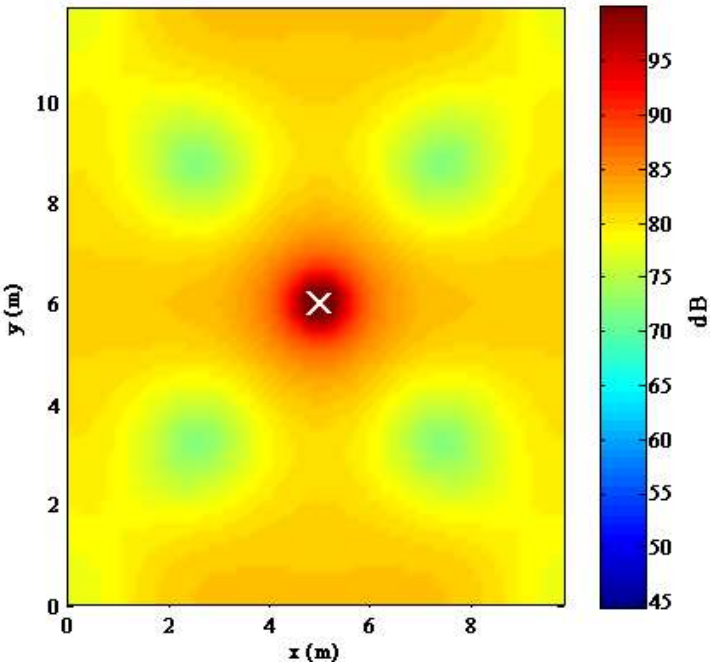

Figure 2.4 Plan view of the GED field ($\beta = 1/4$) produced by a point source radiating at 50 Hz in a rectangular enclosure of uniform absorption and dimensions 10 x 12 x 14 m. The absorption coefficient is 0.01 and the position of the source is indicated by the "X" symbol. The horizontal plane intersects the source located at x = 5 m, y = 6 m, z = 7 m. The complex impedance of the boundaries is 200 + 340i.

2.4.2 Statistical Analysis

Some statistical analysis tools were required in order to quantify and compare the spatial variation between the energy fields within several enclosure types, as well as determine the number of required measurements for accurate mean energy estimates. The script used to model the enclosed sound fields was further adapted to calculate the mean energy density within the volume. The algorithm considered the volume as a whole except for any grid point located within one meter from the walls and within a designated radius d_{min} from the source defined by ISO 3741.[1] This was implemented in order to exclude boundary effects from the walls and to reduce the direct-field energy from the source, respectively. The following sections give a brief explanation of the tools implemented, once the mean energy density in each enclosure was known.

2.4.2.1 Coefficient of Variation and Box-and-Whisker Plots

The coefficient of variation is a normalized measure of dispersion of a probability distribution,[21] defined as the ratio of the standard deviation σ to the mean μ:

$$C_v = \frac{\sigma}{\mu} \ . \tag{2.2}$$

It can be considered simply as the mean-normalized standard deviation of the field. This particular metric was chosen to quantify the spatial variation of each energy field under consideration. It is a more useful metric in this study of energy density than the traditional standard deviation because the latter must always be understood in the context of the mean of the data. The actual value of the coefficient of variation is thus independent of the unit in which the measurement has been taken. For each unique numerical simulation (i.e., involving a unique frequency of radiation, source position, source strength, room size, and absorption) the spatial

variance across the three-dimensional point grid was characterized for each energy density field using its value and saved for later comparison.

The spatial variation among the energy density fields was also analyzed by using box-and-whisker plots. A box-and-whisker plot is a convenient way of graphically depicting groups of numerical data through their quartiles. In addition, these plots generally include lines extending vertically from the boxes (whiskers) indicating variability of the upper and lower quartiles. Outliers are then plotted as individual points (see Fig. 2.5). In this manner, the spacings between the different parts of the plot help indicate the degree of dispersion and skewness in the data.[22]

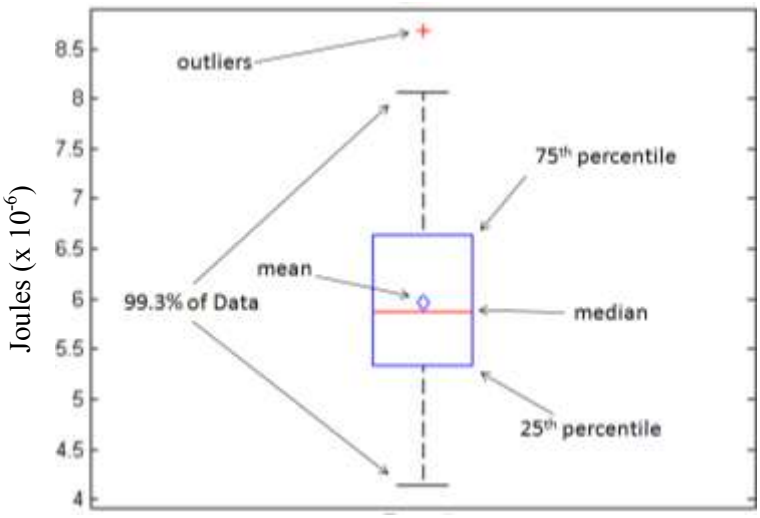

Figure 2.5 Instructive diagram of a box-and-whisker plot used for visually inspecting the degree of dispersion among a data set.

2.4.2.2 The GED Weighting Factor β in Semi-Reverberant Enclosures

Xu *et al*[18] demonstrated how a value of $\beta = 1/4$ optimizes the spatial uniformity of the GED within diffuse fields, but does that value hold for nonideal spaces? Once the tools for modeling and determining the spatial variance of the energy density fields were established, it was possible to explore how the coefficient of variation of the GED field is affected within semi-

reverberant enclosures when the value of the weighting factor β is altered. An algorithm was created to determine and plot the coefficient of variation of the GED field as the value of β was varied systematically between 0 and 1 [refer to Eq. (1.10)]. The results in Fig. 2.6 are typical, where the frequency of radiation is 500 Hz and the absorption coefficient of the room is 0.01.

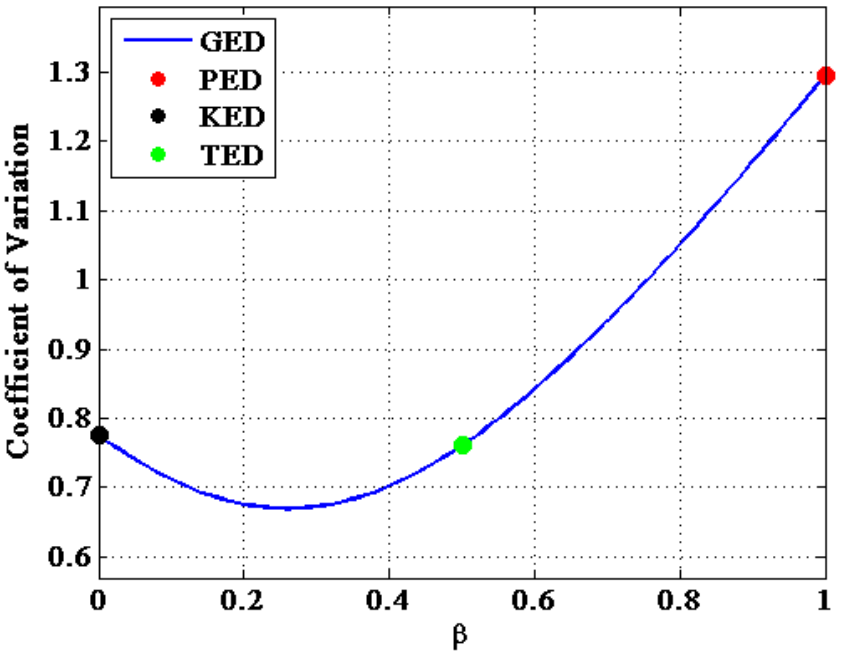

Figure 2.6 The change in coefficient of variation of the GED field as a function of the weighting factor β. The coefficient of variation values for the PED, KED, and TED fields are included for comparison. The value of β which corresponds to the lowest coefficient of variation is 0.26. The excitation frequency is 500 Hz and the room absorption coefficient is 0.01. The room has dimensions 5 x 6 x 7 m. The point source is located at x = 2.2 m, y = 4 m, z = 3.7 m.

This process was repeated for several frequencies and absorption coefficients while the model reported the value of β which corresponded to the lowest coefficient of variation of the GED field. In this manner, it was possible to determine the value of β that minimized the spatial variation of the GED field as a function of both frequency and absorption coefficient. The key results are discussed in Sec. 2.5.1.2.

2.4.2.3 Required Number of Measurement Positions

In order to determine the number of measurement positions required to accurately estimate the mean energy density level in each field, the MATLAB script was further adapted to calculate the mean energy density levels of the simulated energy fields from a series of automated and random configurations of sensor locations. For the first set of configurations, the source was randomly placed within a given volume, the frequency of radiation and characteristics of the room were chosen, and the number of sensors was set to six. The algorithm then randomly placed the sensors throughout the volume while adhering to the sensor location constraints described in ISO 3741.[1] In particular, each sensor needed to be at least one meter from the boundaries, separated from each other by at least half the wavelength of the corresponding frequency, and further than the minimum distance d_{min} from the source. This process is depicted in Fig. 2.7 (note that the sensors, represented by the "O" symbols, do not necessarily lie within the horizontal plane).

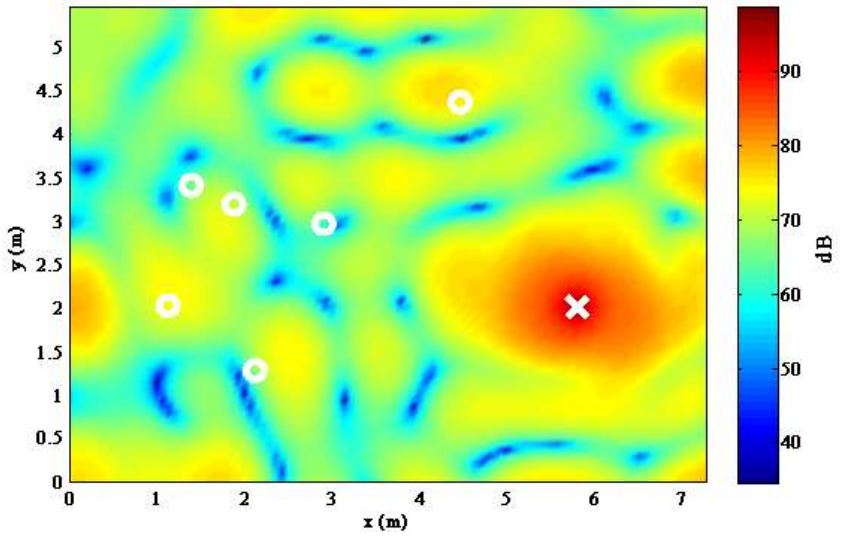

Figure 2.7 Plan view of six randomly placed energy density sensors in a simulated PED field in an enclosure of dimensions 7.3 x 5.5 x 6.4 m. The "X" and "O" symbols represent the source and sensor locations, respectively. The uniform absorption coefficient of the room boundaries is 0.51 and the radiation frequency of the source is 200 Hz. The mean energy density level within the volume is 68.4 dB, and the mean energy density and standard deviation of the receivers are 68.0 dB and 4.3 dB, respectively. The plane intersects the source, but not all receivers. The source is located at x = 5.8 m, y = 2 m, z = 4.3 m.

From these six locations, the standard deviation and mean energy density levels for the PED, KED, TED, and GED fields were calculated and stored. Six sensor locations were randomly selected again and the process was repeated 5,000 times, resulting in 5,000 mean energy levels and standard deviations for each energy density field.

For the same source location, excitation frequency, and room characteristics, the above-mentioned process was repeated 5,000 times for five sensor locations, then four, and so on, until only one sensor location at a time was considered. In this way, one could gain further insight into the number of measurement positions required to accurately estimate the mean energy density of the four energy types within the given enclosure. The entire process was repeated several times, while varying the absorption coefficient of the room, the excitation frequency, and source location.

2.4.3 The Generalized Sound Power Formulation

Most of the research performed on determining the sound power of acoustic sources has relied on sound power formulations similar to the MHSE [see Eqs. (1.7) and (1.9)], where either discrete squared pressure or PED were the measureable field quantities. However, it was hypothesized that sound power estimates could become more accurate and the process made simpler if the less variant GED ($\beta = 1/4$) field is measured instead. In addition, estimates would most likely improve if the directivity factor γ was made to be a function of angle, as in Eq. (1.2). In this manner, the sound power formulation was altered and is called "the generalized sound power formulation" or GSPF. It is expressed as follows:

$$<\Pi_s>_t = \frac{1}{I}\sum_{i=1}^{I} \frac{<w_{g,\beta}(r_i)>_t}{\frac{\gamma(\theta_i,\phi_i)}{4\pi r_i^2 c} + \frac{4}{Rc}} \ , \tag{2.3}$$

where $<w_{g,\beta}(r_i)>_t$ is the *i*th generalized energy density quantity for a chosen β value, and $\gamma(\theta_i, \phi_i)$ is *i*th angle-dependent directivity factor of the source along the axis to the *i*th receiver. In this manner, the GSPF has the ability to incorporate each of the energy density types by varying the value of β:

$$<w_{g,1}(r_i)>_t = (1)PED + (1-1)KED = PED\,, \tag{2.4a}$$

$$<w_{g,0}(r_i)>_t = (0)PED + (1-0)KED = KED\,, \tag{2.4b}$$

$$<w_{g,1/2}(r_i)>_t = \left(\frac{1}{2}\right)PED + \left(1-\frac{1}{2}\right)KED = \frac{1}{2}TED, \tag{2.4c}$$

$$<w_{g,1/4}(r_i)>_t = \left(\frac{1}{4}\right)PED + \left(1-\frac{1}{4}\right)KED = GED\,. \tag{2.4d}$$

2.4.4 Implementation of the GSPF

A process very similar to that outlined in Sec. 2.4.2.3 was implemented to explore the GSPF by way of simulation. The MATLAB script randomly placed a number of energy density sensors, ranging from one to six, throughout a given GED ($\beta = 1/4$) field generated by a point source of known free-field sound power. However, instead of averaging over the GED measurements, the values were inserted into the GSPF [Eq. (2.3)], along with their associated distances r_i. The sound power estimates were then averaged over the measurement positions. The source directivity γ was assigned a fixed value of one, since a point source was used. The script repeated this process 5,000 times for each collective number of measurement positions and for a variety of room and excitation frequency combinations. Thus, the sound power estimates for a range of measurement positions could be studied and compared to the known sound power output of the source in many ideal and nonideal environments.

The simulations were repeated in the same manner for the additional energy density fields where PED, KED, and TED values were inserted into the GSPF. This served as a way to compare the four energy density field types in terms of accuracy and number of required measurement positions.

2.5 Results and Discussion

2.5.1 Energy Density in Semi-Reverberant Enclosures

The following section includes the results of the methods discussed for exploring energy density within semi-reverberant enclosures. The MATLAB model was used to generate the PED, KED, TED, and GED ($\beta = 1/4$) fields for a variety of room dimensions, absorption coefficients, and source locations. In each simulation, the source was omnidirectional. The results are presented in the following order: (1) the spatial variance of each energy field, (2) the optimal value of the GED weighting factor β and (3) the required number of measurement positions for accurate mean energy density estimation.

2.5.1.1 Spatial Variance

When the four energy density fields and their corresponding box-and-whisker plots are displayed side-by-side for various configurations, as in Figs. 2.8-2.10, it is quite easy to visually confirm that the GED is more spatially uniform than the PED. There is a wider range of colors among the PED models and a wider spread of energy values in the box-and-whisker plots. These traits are most apparent with lower room absorption coefficients (see Fig. 2.8). However, this is not the case between the TED and GED. It is therefore more useful to compare the calculated spatial variations of the four energy density fields by their specific coefficient of variation.

In Fig. 2.8, the frequency of radiation is 400 Hz and the absorption coefficient α = 0.01 enables simulation of a reverberation chamber. In this environment, one sees a significant improvement in spatial uniformity between the PED and GED. The coefficient of variation of the GED field is 0.51, approximately half the value of the PED field at 1.055. Those of the KED and TED are 0.603 and 0.591, respectively, falling between those of the PED and GED. However, this is to be expected as the advantages of using the GED field in such diffuse environments have already been explored and verified by others.[16-18]

In Fig. 2.9, α = 0.32 for the same room conditions. From visual inspection, it appears that the GED field is still more spatially uniform than that of the PED, but very similar to those of the KED and TED. The coefficient of variation of the GED field is 0.728, about two-thirds the value of the PED field, which is 1.092. This suggests that there is not as great an improvement in spatial uniformity of the GED field when compared to that produced by the lower absorption value. The coefficient of variation values for the KED and TED are very similar to that of the GED, at 0.769 and 0.781 respectively. However, even in this semi-reverberant enclosure, the GED maintains the lowest value.

For high absorption coefficients of 0.5 and above, there does not appear to be much improvement in spatial uniformity by utilizing KED, TED, or GED in place of PED. Visual differences are minimal when α = 0.5 as in Fig. 2.10, and nearly negligible when α = 0.95 (see Fig. A.1). While the GED has the lowest coefficient of variation of the four fields in Fig. 2.10, it is not much better when compared to the other fields. Fig. A.1 demonstrates that the spatial distributions of the four fields are very similar when the energy is characterized as free-field.

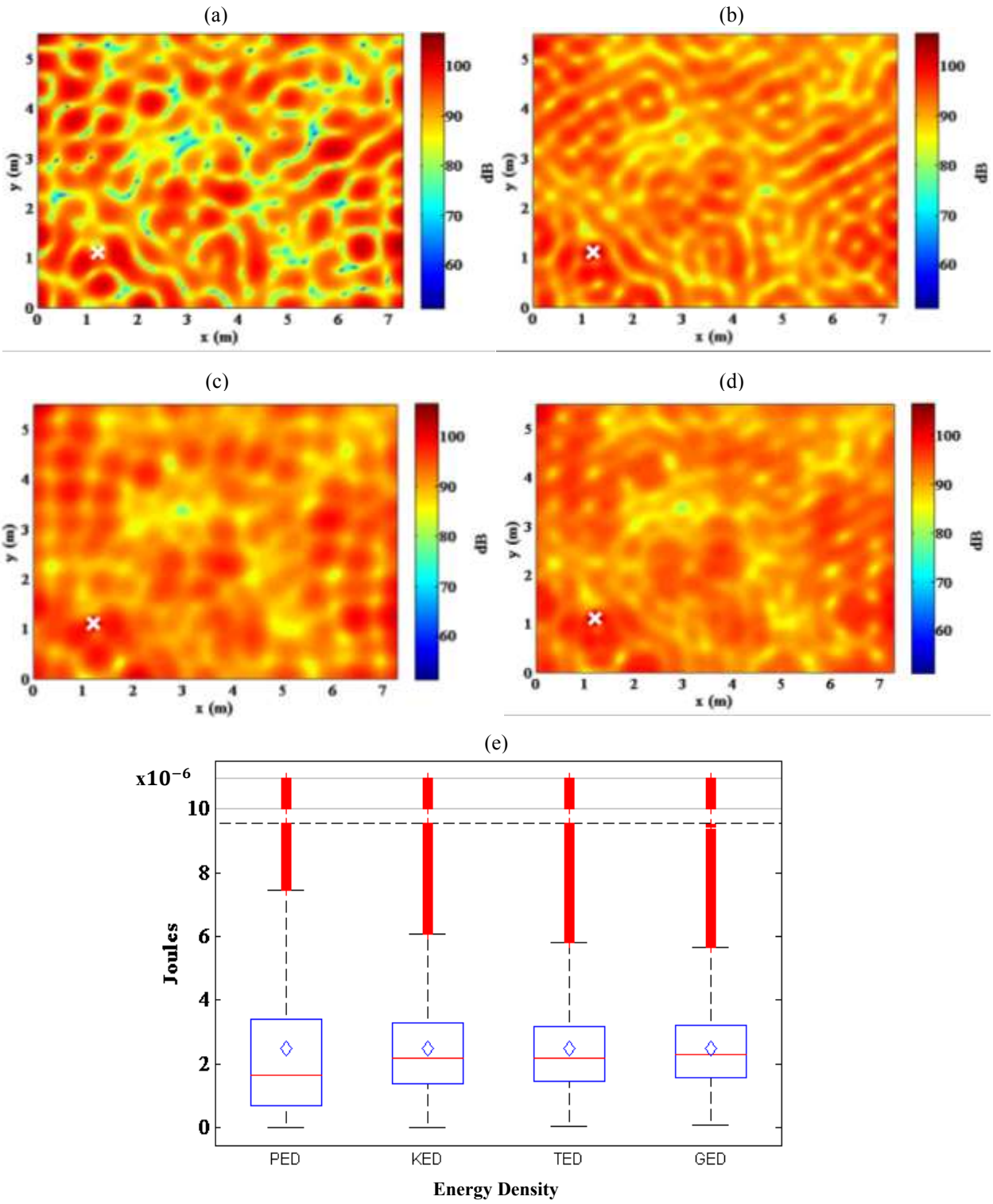

Figure 2.8 Plan views of the energy density fields in a room with a uniform absorption coefficient of 0.01 and dimensions 5 x 6 x 7 m. The source is located at the "x" symbol (x = 1.2 m, y = 1.1 m, z = 4.4 m) and radiates at 400 Hz. The horizontal plane intersects the source. (a) PED, $C_v = 1.055$. (b) KED, $C_v = 0.603$. (c) TED, $C_v = 0.591$. (d) GED, $C_v = 0.510$. (e) Box-and-whisker plots for the four energy density fields. The outliers have been compressed above the dashed horizontal line.

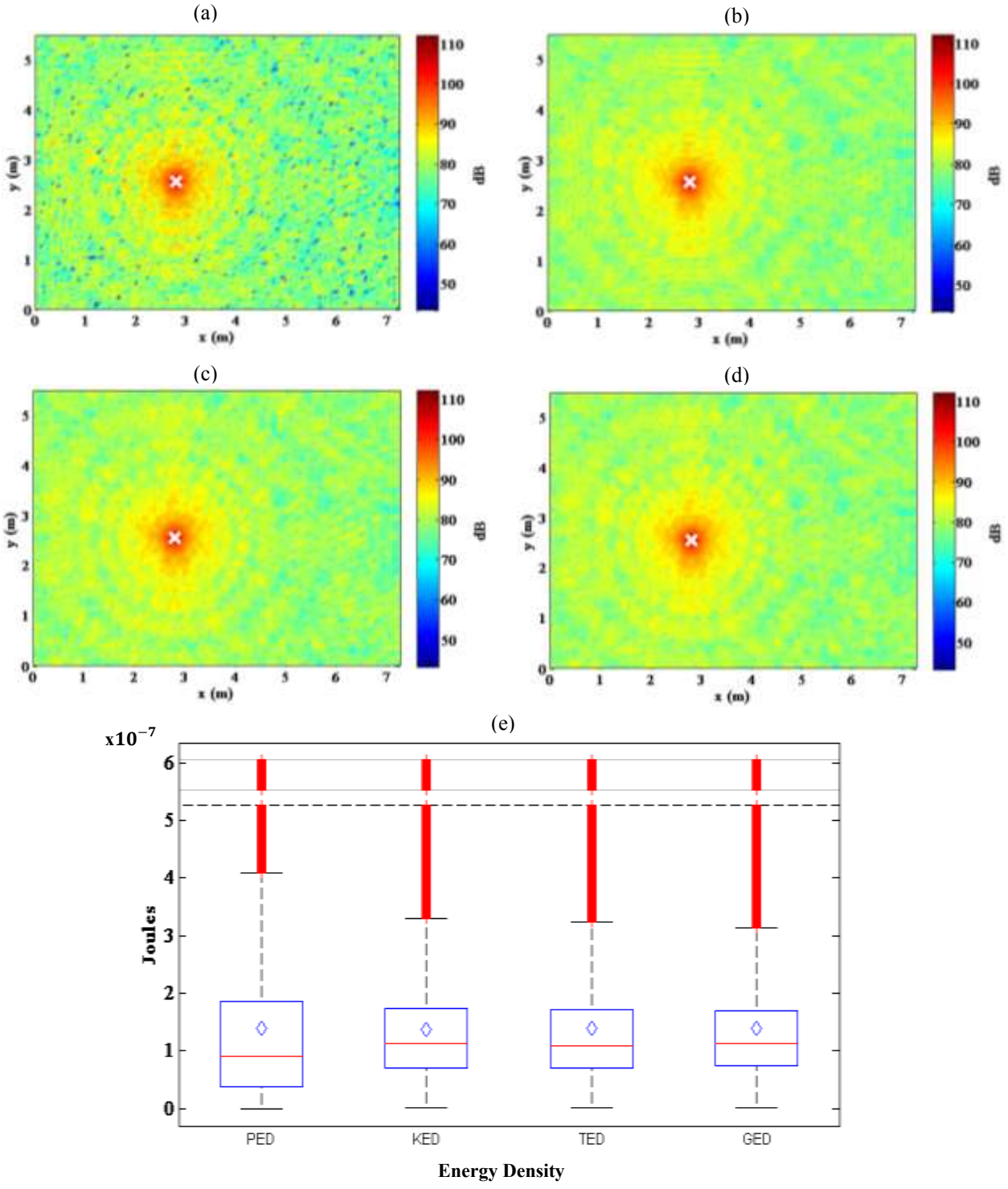

Figure 2.9 Plan views of the energy density fields in a room with a uniform absorption coefficient of 0.32 and dimensions 5 x 6 x 7 m. The source is located at the "x" symbol (x = 2.8, y = 2.6, z = 4.1) and radiates at 2 kHz. The horizontal plane intersects the source. (a) PED, $C_v = 1.092$. (b) KED, $C_v = 0.769$. (c) TED, $C_v = 0.781$. (d) GED, $C_v = 0.728$. (e) Box-and-whisker plots for the four energy density fields. The outliers have been compressed above the dashed horizontal line.

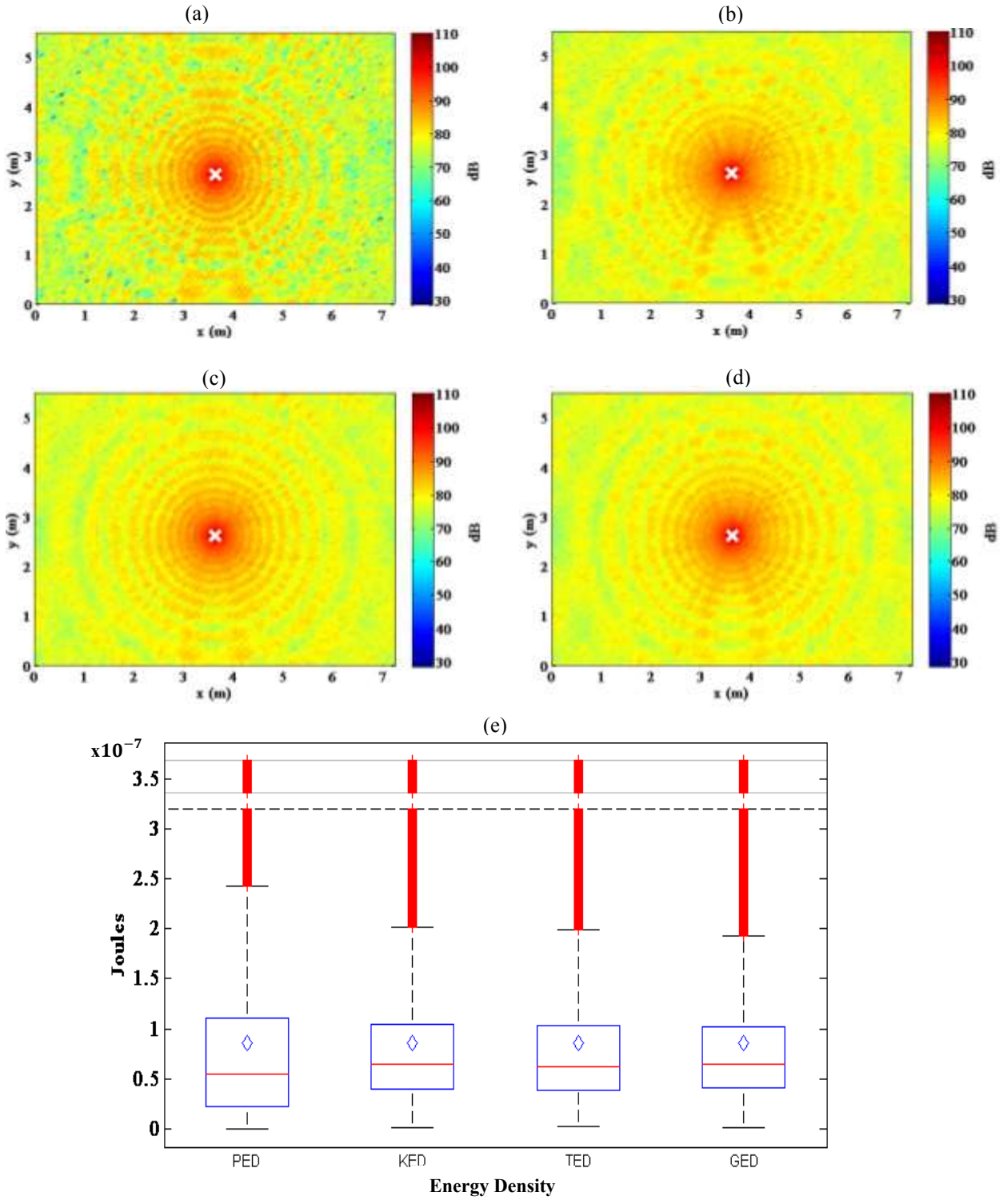

Figure 2.10 Plan views of the energy density fields in a room with a uniform absorption coefficient of 0.5 and dimensions 5 x 6 x 7 m. The source is located at the "x" symbol (x = 3.6 m, y = 2.8 m, z = 3.2 m) and radiates at 3 kHz. The horizontal plane intersects the source. (a) PED, $C_v = 1.189$. (b) KED, $C_v = 0.956$. (c) TED, $C_v = 0.968$. (d) GED, $C_v = 0.933$. (e) Box-and-whisker plots for the four energy density fields. The outliers have been compressed above the dashed horizontal line.

2.5.1.2 Optimized β in Semi-Reverberant Enclosures

The plots in Fig. 2.11 demonstrate that for a frequency range between 100 Hz and 3 kHz, and for many absorption coefficients, a value of 0.25 for the GED weighting factor β usually comes close to minimizing the coefficient of variation within the field. This number varies somewhat at the lower frequencies near and below the Schroeder frequency of the room, and for larger coefficients. This may be seen more distinctly in the surface plot of Fig. 2.12. In this plot, a significant portion of the surface maintains a value at or near 0.25, but varies for lower frequencies and/or higher absorption coefficients. This is most likely due to the lack of acoustic diffusivity caused by low modal densities near or below the Schroeder frequency and/or by the dominant free-field energy component with higher absorption. Frequencies above 3 kHz were not explored due to much larger computation time requirements.

2.5.1.3 Adequate Number of Measurement Positions

An effective way to quantify the benefits of using the GED ($\beta = 1/4$) in enclosed fields is demonstrated in Figs. 2.13-2.15. For example, the results of Fig. 2.13a show that, for a nearly diffuse field, spatial averaging of four randomly placed pressure microphones is required to determine the mean PED within the volume as well as one randomly placed energy density sensor determines the mean GED. In addition, averaging with just two GED sensors can more accurately determine the mean than with six microphones. There does not appear to be as great of an incremental advantage in using three or more GED sensors. The frequency for these results is 400 Hz, but similar results hold for other frequencies as well (see Fig. A.2). It is also apparent that the KED and TED fields introduce significant advantages.

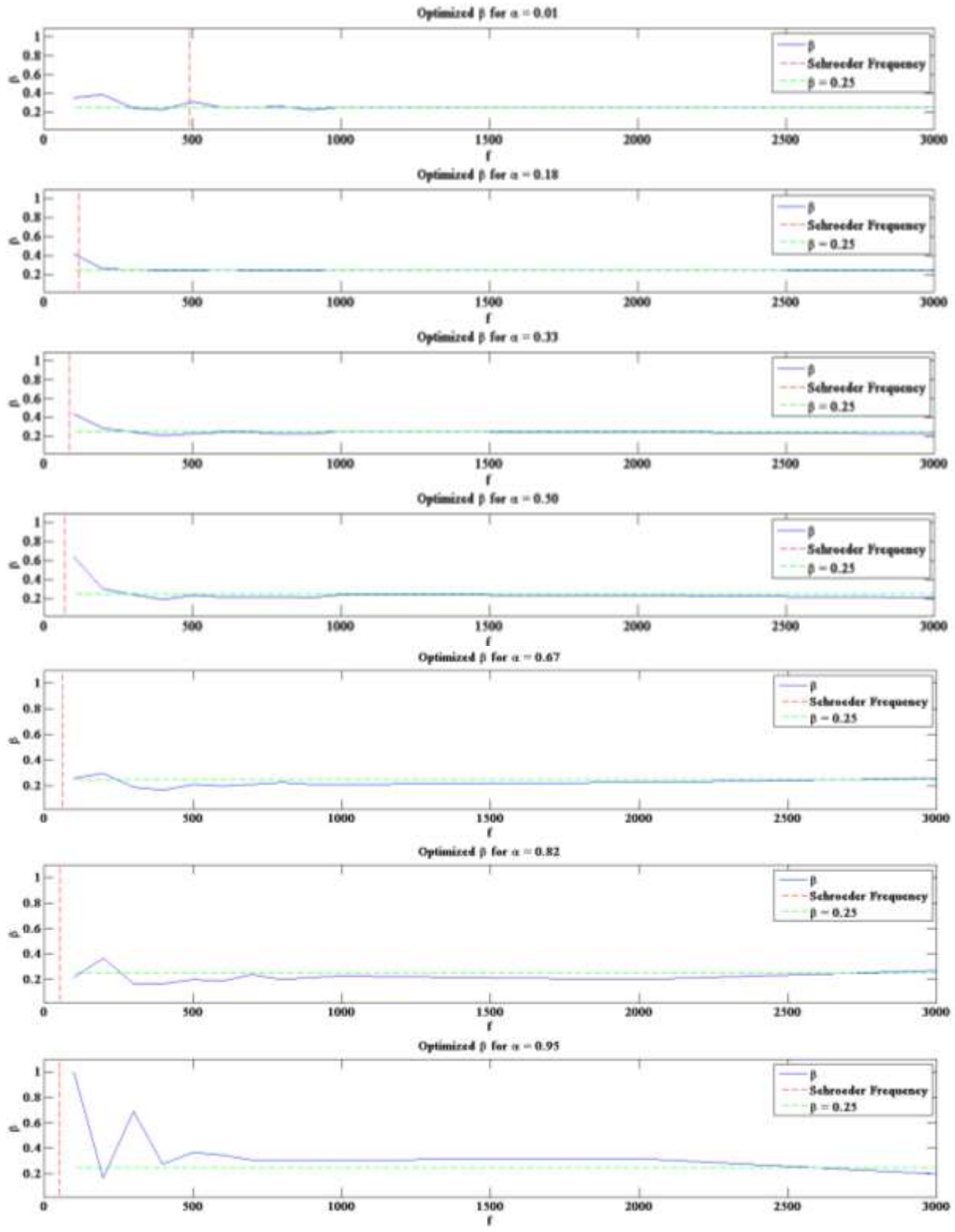

Figure 2.11 The value of the GED weighting factor β as a function of frequency, which minimizes the coefficient of variation of the GED field generated within a 5 x 6 x 7 m space of varying absorption coefficient α. The source is located at x = 3.5 m, y = 2.9 m, z = 1.8m.

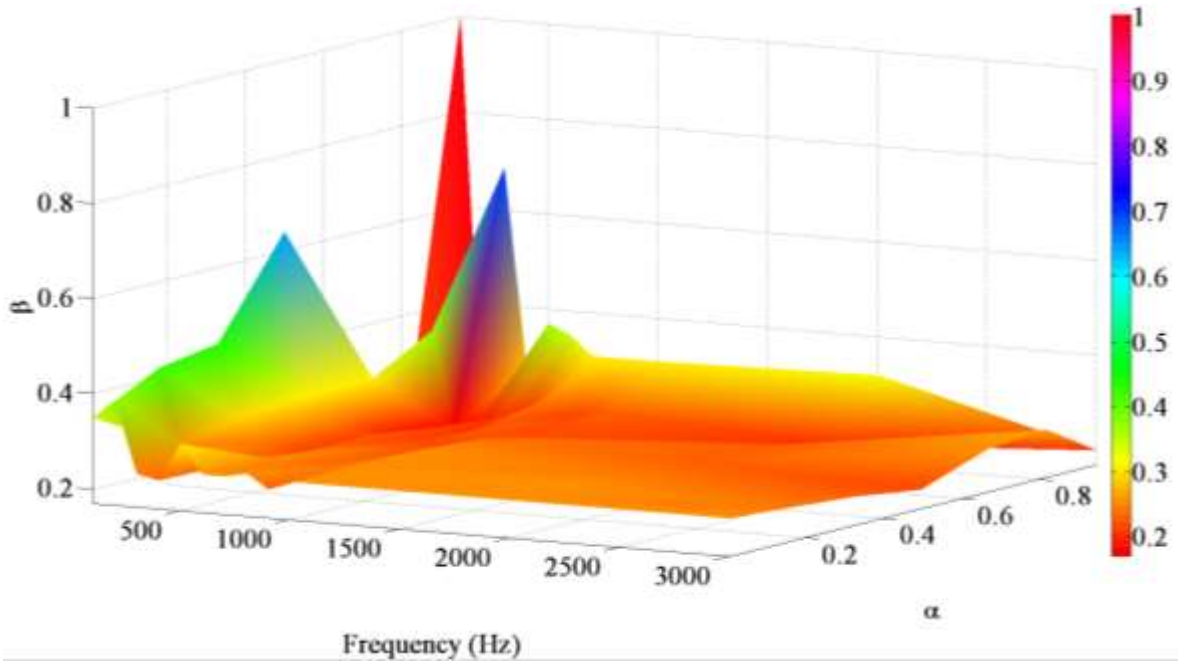

Figure 2.12 Surface plot of the GED weighting factor β as a function of frequency and room absorption coefficient α, which minimizes the coefficient of variation of the GED field within an enclosure of dimensions 5 x 6 x 7 m.

The bars in Fig 2.13b demonstrate the standard deviation between the 5,000 estimated mean energy densities of each field measured by the various collections of sensors. For this particular enclosure, the least variation for any sized sensor collection occurred for the GED.

The advantages of measuring the GED are also substantial when the absorption coefficient of the room is 0.33 (see Fig. 2.14) and 0.5 (see Fig. 2.15). However, it appears that these benefits are greatest within diffuse fields and reduce with increasing α until the absorption is so high that the free-field energy dominates and the advantage becomes negligible (see Fig. A.5). However, even with an absorption coefficient as high as 0.5, the adequate number of sensors required to accurately estimate the mean energy density to within less than 0.5 dB is half that of the PED. Refer to Figs. A.2-A.5 in Appendix A to see the results for other frequency and absorption combinations.

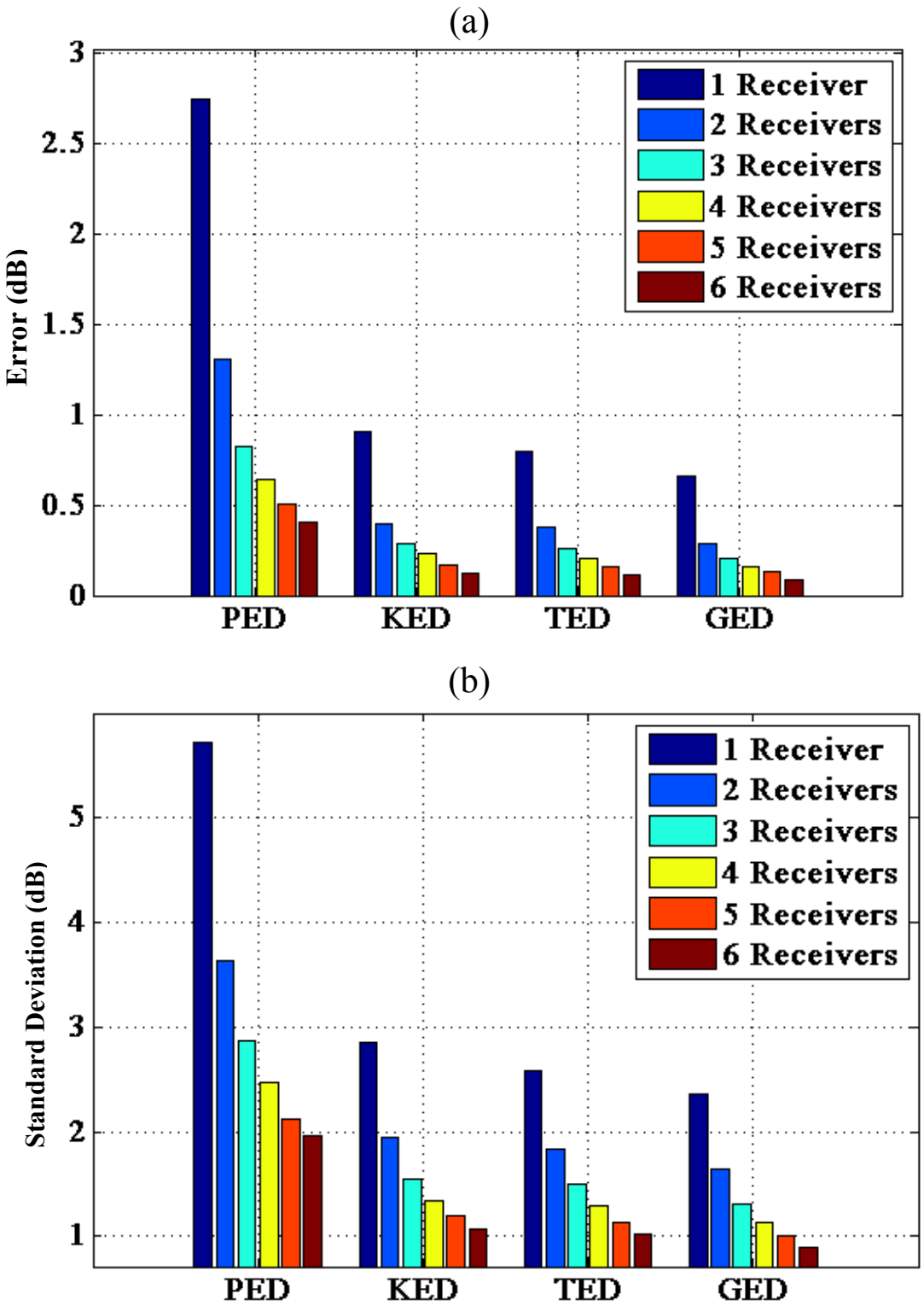

Figure 2.13 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.01. Each simulation was repeated and averaged 5,000 times at a frequency of 400 Hz. (a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined from all field points. (b) Standard deviation of the estimated mean energy densities made by the various sensors.

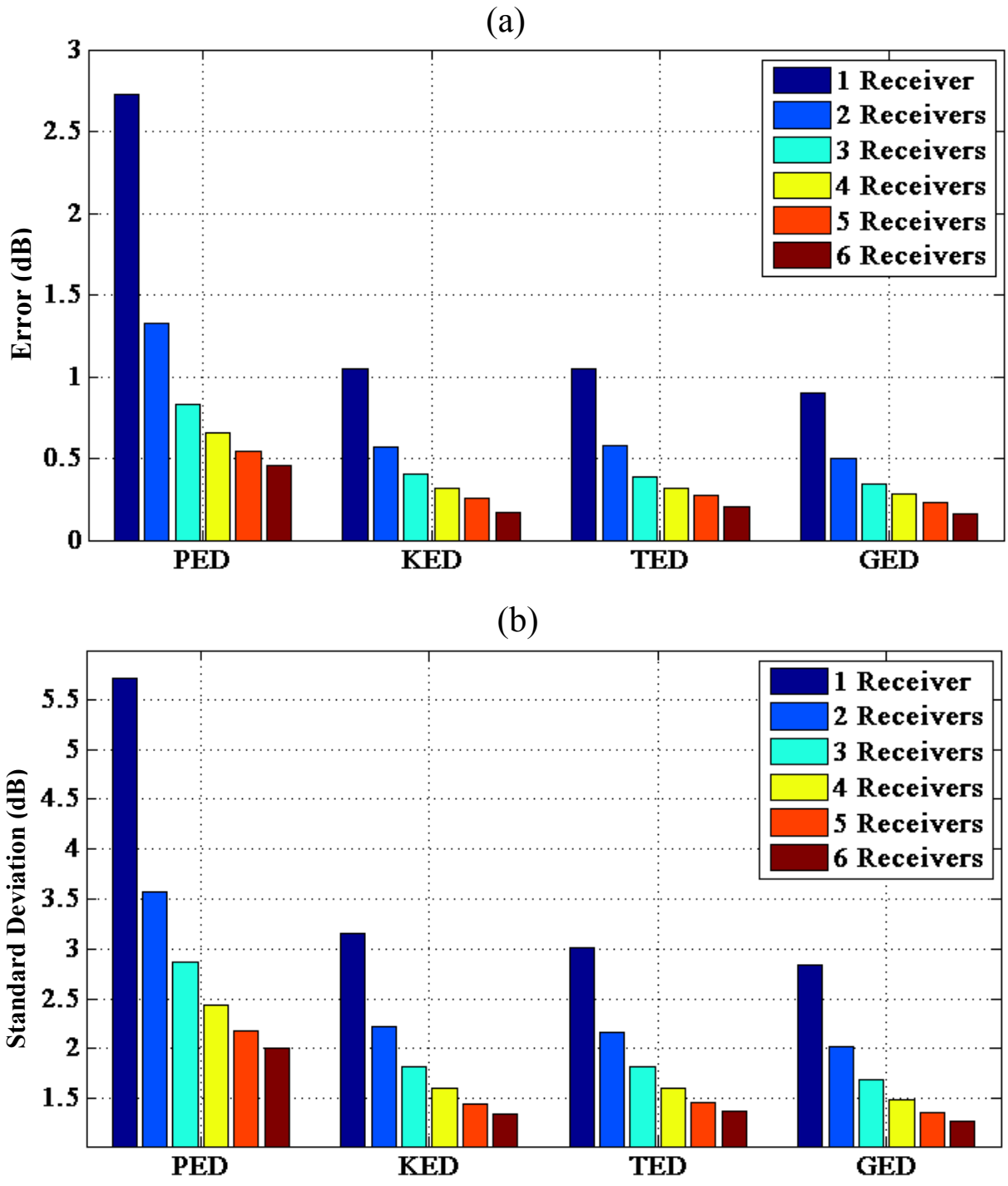

Figure 2.14 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.33. Each simulation was repeated and averaged 5,000 times at a frequency of 3 kHz. (a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined by all field points. (b) Standard deviation of the estimated mean energy densities made by the various sensors.

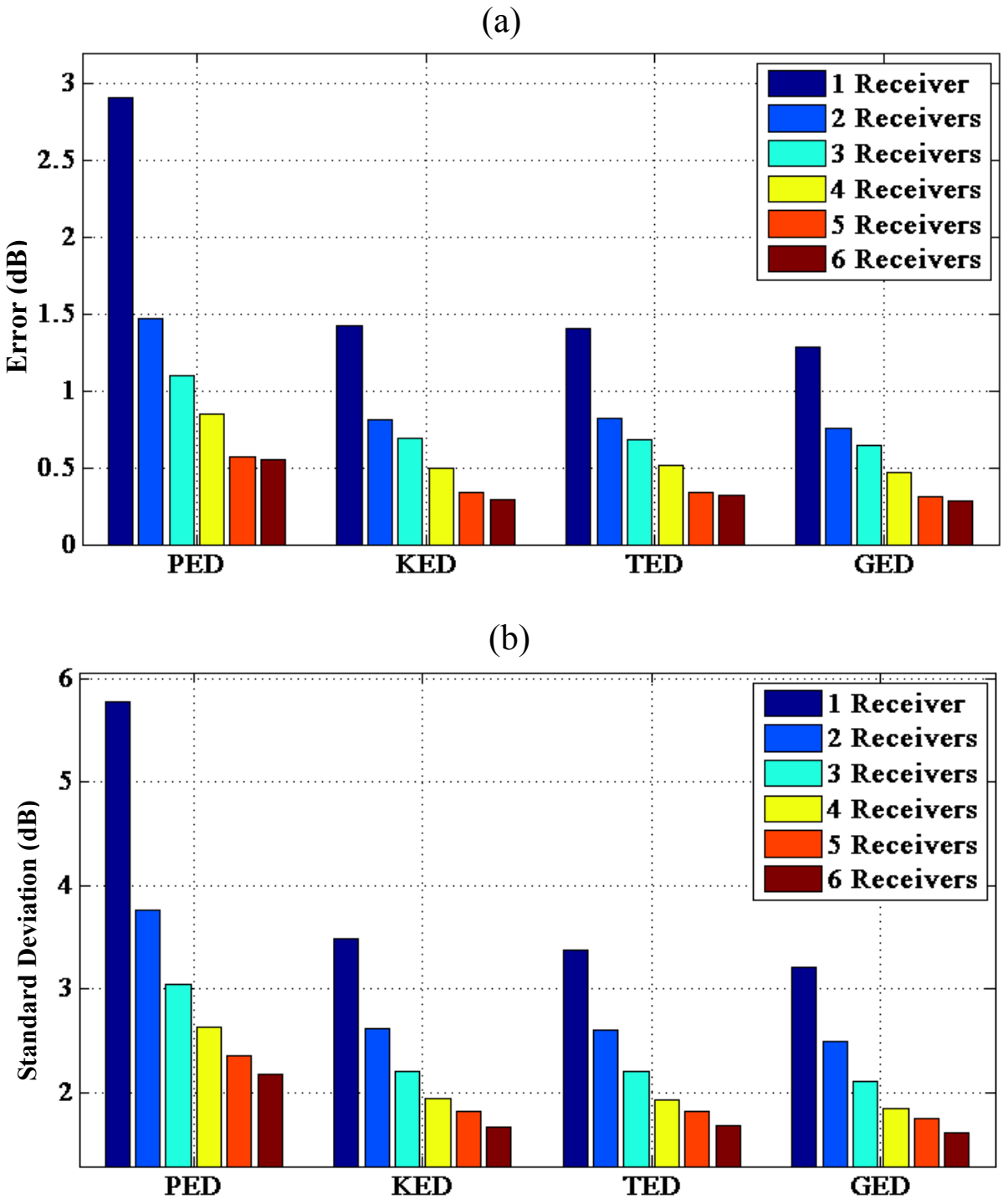

Figure 2.15 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.5. Each simulation was repeated and averaged 5,000 times at a frequency of 1 kHz. (a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined from all field points. (b) Standard deviation of the estimated mean energy densities made by the various sensors.

2.5.2 Sound Power in Semi-Reverberant Enclosures

The accuracy of the GSPF in a very reverberant space is somewhat erratic at lower frequencies near or below the Schroeder frequency, with errors as high as $\pm$ 4 dB (see Fig. 2.16). This trend is common for all four energy density fields. However, the method performs better above 1 kHz where the field is naturally more diffuse. The sound power estimates using the PED are consistently lower than the levels estimated by the other energy density fields, but improve with increasing sensor numbers. There are less significant differences in estimated sound power when varying the number of sensors used for the other energy fields.

For higher absorption values, the GSPF estimates the sound power more accurately. For α = 0.33 and 0.5, errors are less than 2 dB across the indicated frequency band and within $\pm$ 0.5 dB between 1 kHz and 3 kHz (see Figs. 2.17 and 2.18). Some of the same trends seen in the case of very low absorption are also present for these higher absorptions. The PED estimate errors are highest when only one sensor is used, but decreases with an increase in number of sensors. In general, one GED sensor performs as well as multiple PED sensors within these semi-reverberant spaces. (See Fig. A.6 to view the estimated sound power when α = 0.82).

The plots in Fig. 2.19 show the sound power errors within a semi-reverberant enclosure of non-uniform absorption. In this case, the average absorption coefficient of 0.35 is the resultant of three coefficients associated with parallel surfaces. In particular, α = 0.18, 0.33, and 0.5 for two pairs of walls, and for the floor and ceiling, respectively. The results are similar to the case of uniform absorption. The GSPF estimates sound power to within 3 dB at lower frequencies and within 1 dB at higher frequencies. It is quite apparent that it is more accurate when using only a single field sensor when compared to the traditional method of measuring the PED.

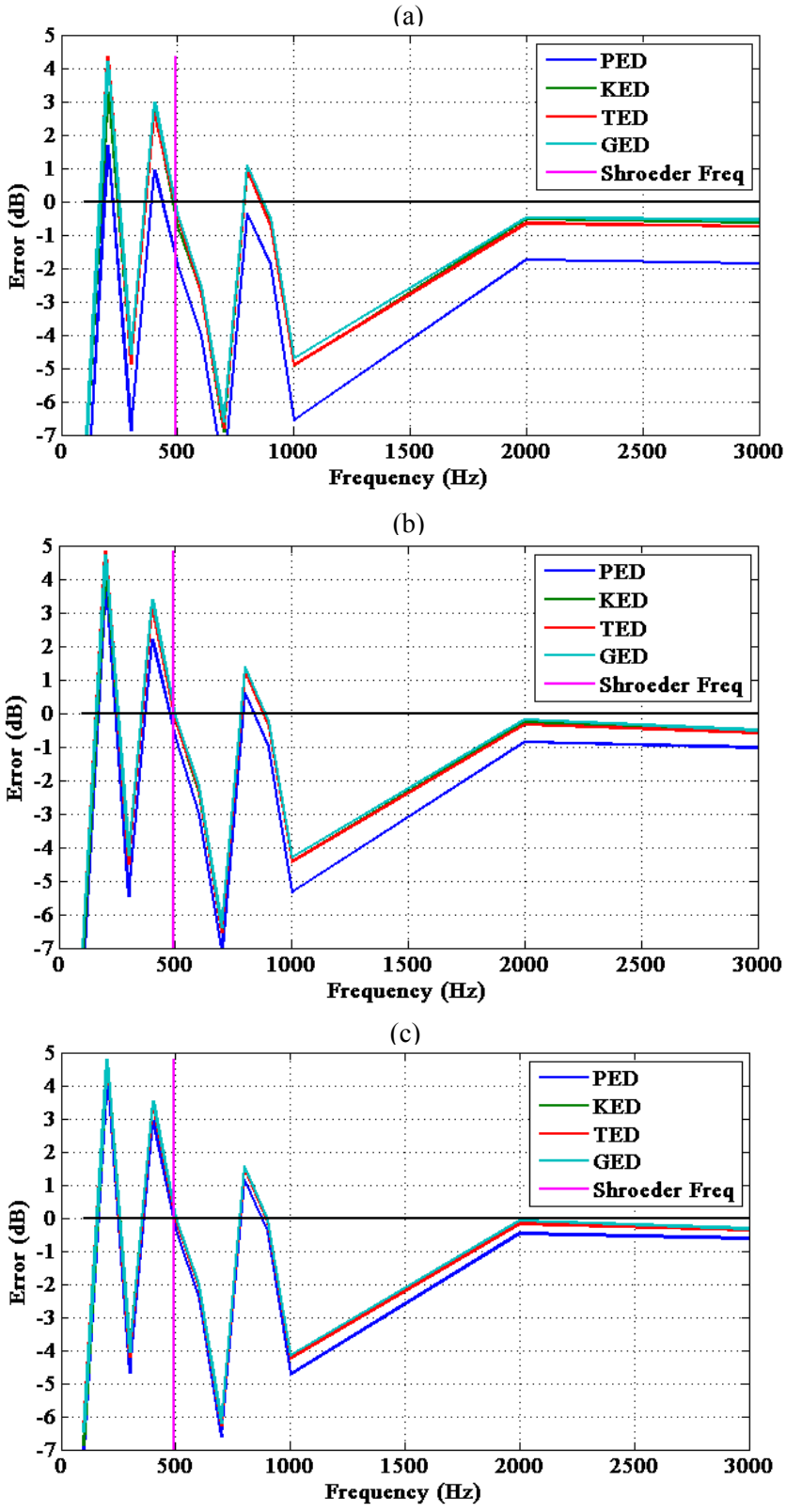

Figure 2.16 The mean error in sound power estimation over 5,000 simulations of the GSPF, where in each case, a collection of randomly placed sensors is used within the four energy density fields. The room has dimensions 7.3 x 5.4 x 4.5 m and an absorption coefficient of 0.01. (a) 1 sensor. (b) 2 sensors. (c) 4 sensors.

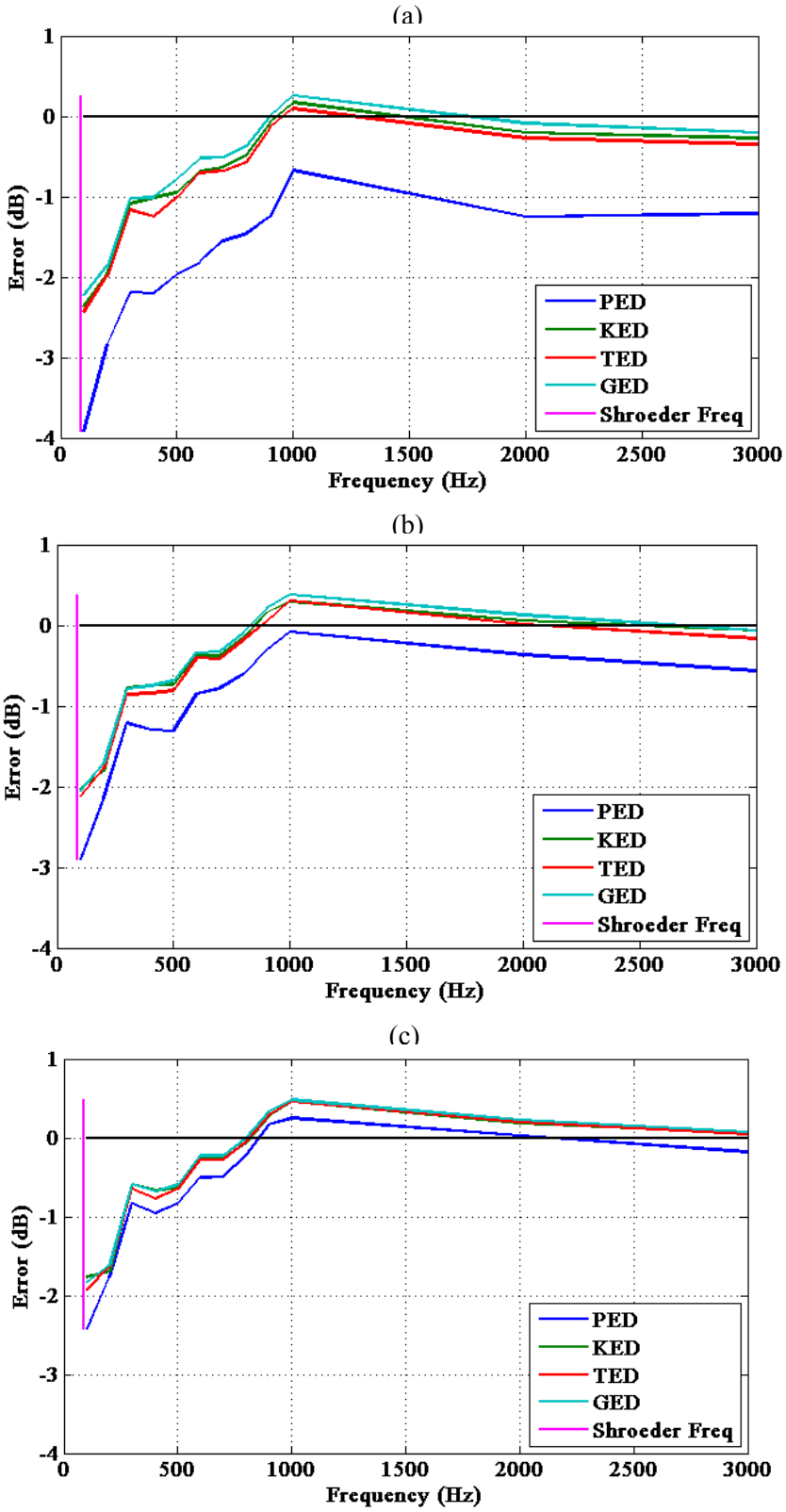

Figure 2.17 The mean error in sound power estimation over 5,000 simulations of the GSPF, where in each case, a collection of randomly placed sensors is used within the four energy density fields. The room has dimensions 7.3 x 5.4 x 4.5 m and an absorption coefficient of 0.33. (a) 1 sensor. (b) 2 sensors. (c) 4 sensors.

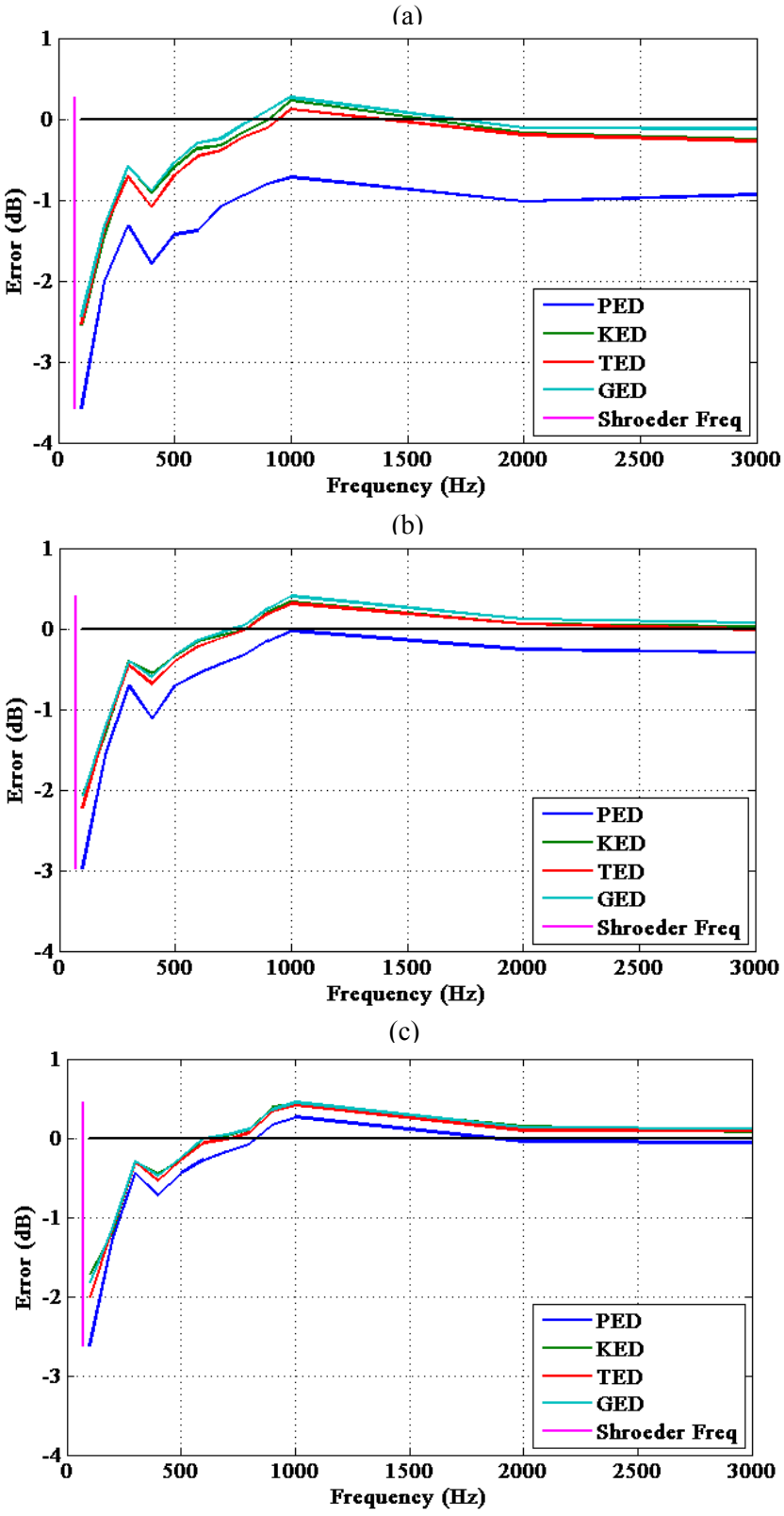

Figure 2.18 The mean error in sound power estimation over 5,000 simulations of the GSPF, where in each case, a collection of randomly placed sensors is used within the four energy density fields. The room has dimensions 7.3 x 5.4 x 4.5 m and an absorption coefficient of 0.5. (a) 1 sensor. (b) 2 sensors. (c) 4 sensors.

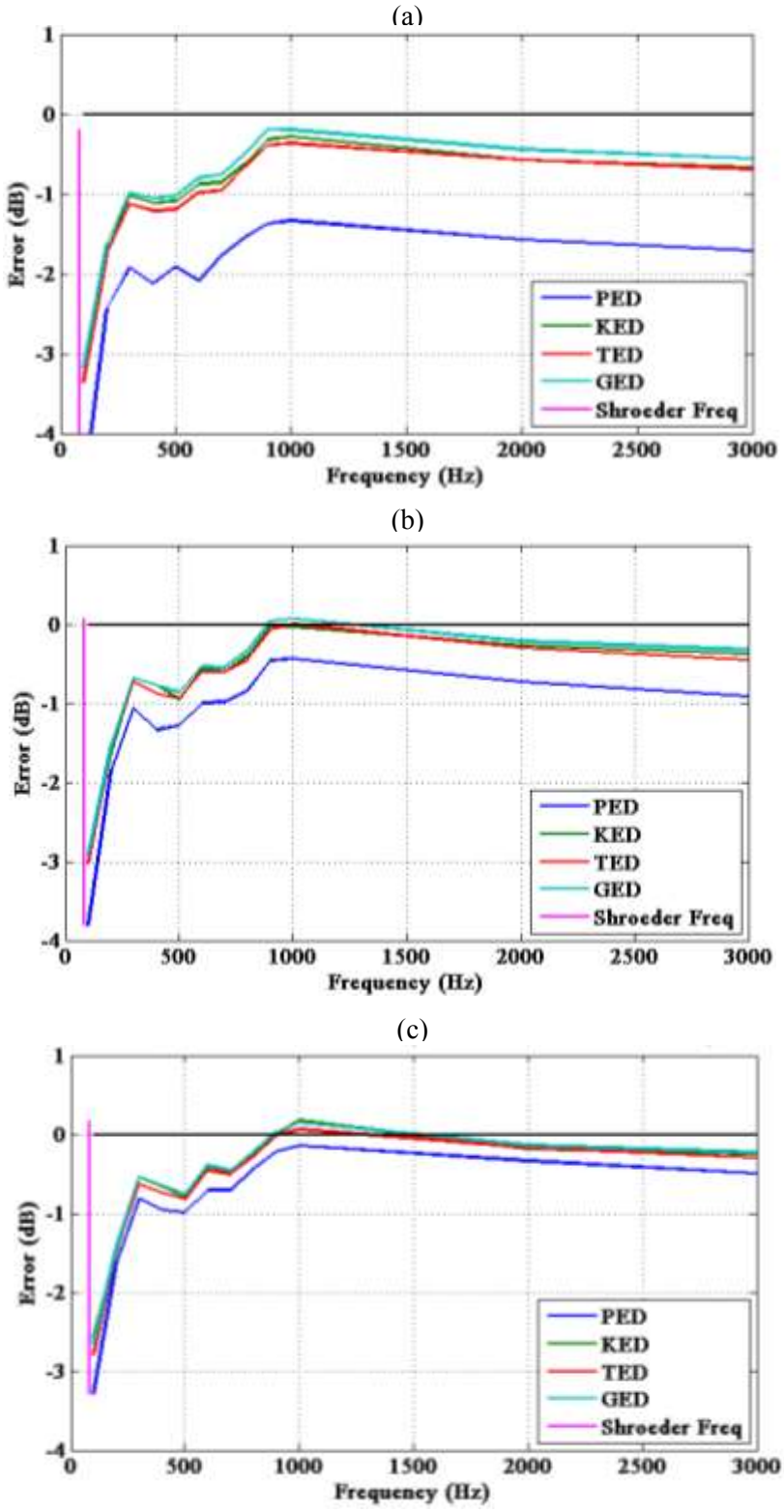

Figure 2.19 The mean error in sound power estimation over 5,000 simulations of the GSPF, where in each case, a collection of randomly placed sensors is used within the four energy density fields. The room has dimensions 7.3 x 5.4 x 4.5 m and non-uniform absorption (α = 0.18, 0.33, 0.5 for parallel surfaces). The average absorption coefficient of the enclosure is 0.35. (a) 1 sensor. (b) 2 sensors. (c) 4 sensors.

2.6 Specific Conclusions

The numerical modeling and statistical exploration of the GED field in both diffuse and semi-reverberant fields has demonstrated its effectiveness in comparison to the PED. The fundamental characteristic that causes its improved performance is its greater spatial uniformity over an enclosed sound field. The specific conclusions associated with the methods performed in this chapter are as follows:

1. The GED field demonstrates the lowest spatial variation within enclosures having absorption coefficient values between 0 and 0.5, and tends to match that of the PED at higher levels of absorption.
2. The nominal value of the weighting factor β that optimizes the spatial uniformity of the GED is near 0.25 for many absorption coefficients and frequencies of interest. Most discrepancies are minor and occur at very low frequencies below the Schroeder frequency of the room.
3. GED measurements yield the most accurate mean value estimates of the energy field with the fewest sensors. The number of GED sensors compared to microphones can be reduced by approximately ¼ in diffuse fields and by ½ in semi-reverberant enclosures. However, there are no significant benefits for very high absorption (i.e., $\alpha > 0.5$).
4. For diffuse and semi-reverberant rectangular enclosures with $\alpha < 0.5$, utilizing the GED ($\beta = 1/4$) in the GSPF is a method that can generally estimate the sound power of an acoustic source within 1-3 dB for the frequency range between 100 Hz and 3 kHz. The same results seem to hold for cases of low uniform absorption.
5. The estimated power errors are generally the lowest when the GED is measured instead of PED, KED, or TED.

While these conclusions are strictly based on the numerical simulations implemented for this chapter, it is believed that the advantages associated with GED in semi-reverberant enclosures exist in practice. Experimental efforts have been conducted to verify real-world application of the concepts and are explained in Ch. 3.

Chapter 3

A Sound Power Formulation for Semi-Reverberant Enclosures Using Generalized Acoustic Energy Density: Experimental Methods and Results

This chapter outlines the experimental techniques used to explore the efficiency and practicality of estimating the sound power of acoustic sources within diffuse and semi-reverberant fields using the GSPF described in Ch. 2. It also describes how using the formulation overcomes some distinct challenges associated with standards for determining sound power within reverberation chambers.

3.1 Specific Background

The equation that Hopkins and Stryker developed in 1948 to estimate the total energy density within semi-reverberant enclosures [see Eq. (1.1)] has been used by many for different applications. In their paper,[5] they recommended using a value of either one, two, or four for the directivity factor γ. These values correspond to an omnidirectional source radiating in a free-space, half-space, or quarter space, respectively, and are not dependent on frequency. In their opinion, values of γ greater than four were not likely to be encountered in practice.

Thompson, *et al*.[23] understood that the directivity of the source would not hold constant and in 1976 they published a paper that addressed this concern. They introduced a computational technique that modified the value of γ for each receiver position. They reported excellent agreement with measured data, but the method was still limited by the need to know both the sound power of the source and the absorption in the room.

Recall from Sec. 1.1.4 that a few standards[1,2] avoid using the directivity factor entirely by requiring the source to be placed in a qualified reverberation room and placing all receivers beyond a specified distance from the source wherein the free-field component of the energy is negligible. This distance is given as[1]

$$d_{min} = D\sqrt{\frac{V}{T_{60}}}\,, \tag{3.1}$$

where D = 0.08 m for $f \geq 5$ kHz and 0.16 m for $f < 5$ kHz, V is the volume of the enclosure, and T_{60} is the frequency-dependent reverberation time. One is then required to calculate the standard deviation of sound pressure levels among several positions (at least six). The assumption is that if the value of the standard deviation is higher than a certain threshold, the directivity of the source is too high for the given number of spatial measurements to average appropriately. In this case, one must include more spatial sampling positions in the field until the standard deviation is reduced to an acceptable value.

3.2 Specific Motivations

The directivity factors of most sources are functions of frequency, demonstrating recognizable mid-frequency lobing and significant directionality at high frequencies ($\gamma > 4$). Thus, in addition to changing the value of γ according to measurement angle and position,

similar to the work of Thompson *et al*, one should consider how γ changes as a function of frequency. This would allow the MHSE represented in Eq. (1.9) to better estimate the sound power radiation for more directional sources and for a wider frequency range.

Thompson *et al* demonstrated that sound pressure level estimates improve by varying the directivity factor appropriately. However, their technique does not explore KED or TED because probes that could measure the acoustic particle velocity were not yet common. In Ch. 2, the benefits of using the GED ($\beta = 1/4$) to estimate sound power were verified numerically, but for omnidirectional sources only. It has yet to be explored experimentally whether utilizing the GED within diffuse fields and/or semi-reverberant enclosures can accurately estimate the sound power of directional sources.

It can also become quite cumbersome to avoid the directivity factor of the source by following the sound power standards for reverberant rooms. Because the goal is to eliminate the need to consider the direct field, the standards are not suited for highly directional sources whose direct energy fields reach beyond the minimum distance d_{min}. For such cases, the standard requires many more measurement positions. This could become either highly cumbersome or nearly impossible to achieve. The GSPF accounts for the direct energy and could overcome this specific challenge, allowing one to more conveniently measure the sound power of highly directional sources within reverberant rooms.

3.3 Specific Objectives

The following important questions will be answered by the experimental methods discussed in this chapter:

1. Because the GSPF uses the more spatially uniform GED and incorporates an angle and frequency-dependent γ, does it more accurately and conveniently estimate sound power within semi-reverberant enclosures?
2. Could the ISO 3741 measurement constraints on source-receiver proximity be removed by employing the GSPF for reverberation rooms because it accounts for the direct-field component of the energy?

The answers to these questions could have a significant impact on current methods of measuring sound power.

The aims of the experimental work discussed in this chapter were to answer the above questions through the following objectives:

1. Measure the frequency-dependent sound powers of three sources according to precision standards ISO 3741 and ISO 3745 and compare them to those predicted from the GSPF for the same sources located within three semi-reverberant enclosures and a reverberation chamber.
2. Determine if the method can accurately and conveniently estimate sound power within semi-reverberant enclosures.
3. Determine if using the method in a reverberation chamber could outperform the results of ISO 3741 when sensors are placed near the source under test to deliberately violate the intended reverberant-field requirements of the standard.

3.4 Methods

3.4.1 Test Sources and Enclosures

In theory, the GSPF is flexible enough to determine the sound power of a wide range of acoustic sources in a variety of environments. This section describes the different sources and rooms that were used to explore the accuracy of the technique in a variety of real-world settings.

3.4.1.1 Sources

Apart from physical size and power output, the directivity of a source is one of its most significant attributes. For this reason, three sources of varying directivity were used throughout the experimental work outlined in this chapter. The first was a dodecahedron loudspeaker mounted on a stand (see Fig. 3.1). This configuration was chosen because it approximates an omnidirectional source up to approximately 1 kHz. The second source was the same

Figure 3.1 The dodecahedron loudspeaker used in the investigation. It is composed of 3-inch drivers mounted on 12 regular pentagonal faces. Ideally, all drivers radiate in phase and with equal amplitude.

loudspeaker, but with only one driver active. In this manner , the configuration approximated a driver mounted in a spherical enclosure. The loudspeaker could alternate between each configuration by a switch located near the bottom of the polyhedron. The third source was a horn-loaded compression driver (see Fig. 3.2) mounted on a stand. This particular configuration was chosen for its high directivity at mid-to-high frequencies and its high sound power output.

Figure 3.2 A compression driver attached to a 35° x 45° constant directivity horn.

3.4.1.2 Enclosures

Four enclosures of various volumes and average absorption coefficients were used in the experiments. The first was a qualified reverberation chamber with dimensions 5 x 6 x 7 m and volume 210 m^3 (see Fig 3.3). Experiments were performed in this room for two reasons. First, the benefits associated with the GED are most prevalent in a diffuse field, so the sound power estimates within the reverberation chamber would serve as accuracy benchmarks for the GSPF method. Second, the room satisfied the test room qualifications required for ISO 3741, where the

Figure 3.3 Qualified reverberation chamber with dimensions 5 x 6 x 7 m and volume 210 m^3.

receiver placement violation tests could be conducted.

The other three enclosures were considered more semi-reverberant in nature and varied in size and acoustic absorption to explore the possible implementation of more practical field measurements. The smallest was a classroom with dimensions 4.8 x 6.5 x 2.9 m and volume 90.5 m^3. The concrete floor was covered with carpet, the walls were made of drywall, and the ceiling was composed of suspended acoustic tiling (see Fig. 3.4). The intermediate-sized enclosure was a conference room with dimensions 9.3 x 9.2 x 2.7 m and volume 231 m^3. The concrete floor was carpeted, the walls were made of wood paneling and drywall, and the ceiling was composed of suspended acoustic tiling (see Fig. 3.5). The largest room was a dance studio with dimensions 9.6 x 15.4 x 3.6 m and volume 532 m^3. The floor consisted of wood on concrete, the walls

Figure 3.4 A university classroom with dimensions 4.8 x 6.5 x 2.9 m and volume 90.5 m^3. The floor is carpeted cement, the walls are made of drywall, and the ceiling is composed of suspended acoustic tiling.

Figure 3.5 A conference room with dimensions 9.3 x 9.2 x 2.7 and volume 231 m^3. The floor is carpeted cement, the walls are made of wood paneling and drywall, and the ceiling is composed of suspended acoustic tiling.

were made of ceramic tiling covered in part with absorptive treatment, and ceiling was composed of acoustic tiling with additional spaced absorptive treatments (see Fig. 3.6).

Figure 3.6 A dance studio with dimensions 9.6 x 15.4 x 3.6 m and volume 532 m^3. The floor is wood, the walls consist of ceramic tile with some absorptive treatment, and the ceiling is composed of acoustic tiling with added absorptive treatments.

3.4.2 Data Acquisition System

In order to maintain consistency between measurements, it was necessary to build a data acquisition system that could be transported and used in all of the enclosures detailed above. The system consisted of a portable rack unit with an HP/Agilent VXI analyzer of 54 input channels and 3 output channels (HPE8491B/HPE1432A/HPE1433B cards), a Presonus Firepod interface with 8 preamplified inputs and 8 outputs, a Crown XLS1000 power amplifier, a Bose model loudspeaker for monitoring, and the M+P International Smart Office Analyzer software on a laptop to process and export acoustic data (see Fig. 3.7). The analyzer was used to capture data

from various microphone and energy density measurements and the power amplifier was used to drive the three test sources. Precautions were taken to ensure identical analyzer and amplifier settings for measurements of each test source in all test spaces.

Figure 3.7 A portable dynamic signal analysis system with several input and output channels, a power amplifier, a loudspeaker for monitoring, and a software package.

3.4.3 The Generalized Sound Power Formulation

The GSPF requires one to know three frequency-dependent variables: the directivity factor of the source for multiple radiation angles, the absorption of the enclosure, and the GED within the room at several positions [see Eq. (2.3)]. The methods used to acquire these values are outlined in the following sections.

3.4.3.1 Determination of the Directivity Factor

Because it was necessary to know the frequency-dependent directivity factor for multiple polar and azimuthal angles, it was decided to determine the value of γ with high angular resolution and thus provide flexibility for subsequent experimental procedures.

As depicted in Fig. 3.8, the source directivities (and sound powers) were measured by mounting each source on a narrow stand in an anechoic chamber, then rotating it under a semi-circular microphone array with a computer-controlled turntable. Absorptive materials were positioned on top of the turntable and stand to reduce scattering. The turntable was programmed

Figure 3.8 The experimental setup for determining the directivity factor and sound power of the source under test in an anechoic chamber. A dodecahedron loudspeaker is shown mounted on a turntable stand. The turntable is covered with absorptive treatment to maintain anechoic conditions. A 180° arc array of 37 microphones senses the radiated field in 5° increments. The source was also rotated in 5° increments in the azimuthal angle.

to rotate in 5° steps in the azimuth angle ϕ. Thirty-seven G.R.A.S. 40AE 1/2-inch precision microphones were also positioned on the measurement arc at 5° increments in the polar angle θ. Each was relatively calibrated to a reference microphone. The arc was positioned in the vertical plane of the source and had a radius of 1.8 m. The axis of the top microphone (θ = 0°) was directed along a vertical line running through the approximate acoustic center of the source. For the dodecahedron and single driver sources, the acoustic center was assumed to be the geometric center of the polyhedron and the loudspeaker driver, respectively. Finding the acoustic center of the horn-loaded driver was more challenging. Efforts were made to follow the suggestions offered by Ureda[24-27] on the acoustic center of horns and how their apparent apexes move as a function of frequency. In particular, he discusses where to place the axis of rotation for directivity and polar-plot measurements to minimize average error. The axis of the microphone located at the center of the array (θ = 90°) was also directed along a horizontal line running through the approximate acoustic center of each source.

In the measurement configuration, averaged frequency response and coherence functions between the excitation signal and microphone outputs were generated between 20 Hz and 20 kHz. This was considered a good choice for precision directivity measurements because they provided smooth frequency dependence in the measured data. A formulation following from Eq. (1.2) demonstrates how one calculates the angle and frequency-dependent directivity factor using frequency response functions:

$$\gamma_{m,n}(f) = \frac{<I_{m,n}(f)>_t}{<I_{omni,m,n}(f)>_t} = \frac{4\pi r^2 \left|H_{m,n}(f)\right|^2}{\sum_m \sum_n S_{m,n} \left|H_{m,n}(f)\right|^2} \ , \tag{3.2}$$

where the subscripts m and n represent discrete polar and azimuth angles on the measurement surface with $\Delta\theta = \Delta\phi = 5°$, $H_{m,n}(f)$ is the frequency response function between the excitation

signal and microphone output, and $S_{m,n}$ is the area weighting factor (i.e., the effective sampling area per microphone on the measurement sphere of radius 1.8 m). The latter is determined by surface integration over appropriate sections of the measurement sphere.[28] Appendix B provides a derivation of this formulation.

For each source rotation position in ϕ, the 37 frequency response functions in θ were exported for post-processing. This procedure began at $\phi = 0°$ and was repeated for each 5° increment until each source had been rotated through $\phi = 355°$. This created a 2,664-point set of complex frequency response functions over the measurement sphere to characterize the directional behavior of each source. For reference, the initial axes of the sources were oriented toward the microphone located at $\theta = 90°$.

3.4.3.2 Determination of the Room Constant

The room constant R is a frequency dependent value that depends on the average absorption of the room and its surface area [see Eq. (1.1b)]. The approximate surface area of each rectangular test room was easily determined by use of a laser distance finder to measure its dimensions. The average absorption coefficient $<\alpha>_s$ was determined indirectly by reverberation time (T_{60}) measurements and the Eyring equation with air absorption correction:[29]

$$<\alpha>_s = 1 - e^{\frac{1}{S}\left(4MV - \frac{0.16V}{T_{60}}\right)}, \tag{3.3}$$

where M is the power attenuation coefficient, in reciprocal meters, calculated according to ISO 9613-1.[30] This particular formulation was used because past research[29] has shown that the Eyring formulation is better to use than the traditional Sabine formulation for cases of higher absorption. At lower absorptions it also converges to the Sabine formulation.

For all T_{60} measurements, the dodecahedron loudspeaker was excited by a maximum length sequence signal and the integrated impulse responses from twelve source-receiver positions were used for averaging, as required by the ISO 354 standard[31] (see Fig. 3.9). The average T_{60} values were determined from the integrated impulse response method performed by the EASERA software package from the Ahnert Feistel Media Group.

Figure 3.9 The test configuration for determining the frequency-dependent reverberation time of the medium-sized test room. The dodecahedron loudspeaker was excited with a maximum length sequence signal and the impulse response from twelve source-receiver positions were used for averaging, as required by the ISO 354 standard. The average reverberation time was calculated by the integrated impulse response method performed by the EASERA software package.

3.4.3.3 Energy Density Measurements

The GSPF requires measurements of the energy density field at multiple angles and distances from the source. For each measurement, the distance from the acoustic center of the source to the energy density probe was determined using a laser distance finder, which was

accurate to 1 mm. The polar and azimuth angles were measured using a laser pointer attached to an altazimuth mount (see Fig. 3.10). The mount was placed at the position of the acoustic center of the source and the relative angles θ and ϕ of the sensor were determined with the aid of the laser pointer.

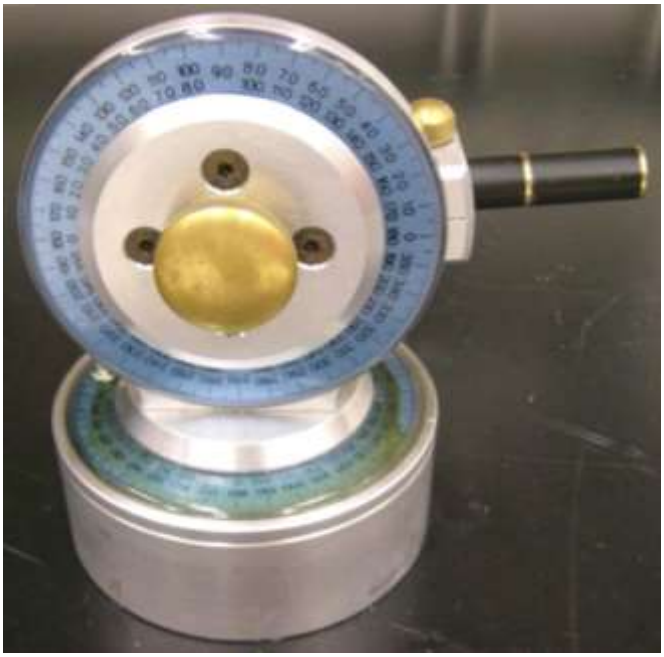

Figure 3.10 A laser pointer attached to an altazimuth mount.

Two three-dimensional energy density probes were used throughout the experiments detailed in this chapter. The first was the Type 50VI vector intensity probe manufactured by G.R.A.S. Sound and Vibration. The second was the Ultimate Sound Probe (USP), manufactured by Microflown Technologies. Each utilized very different techniques to determine the particle velocity, which is often where accuracy suffers. Every experiment was repeated with each probe as a way of comparing results based on the two technologies.

The G.R.A.S. Type 50VI (see Fig. 3.11) incorporates the pressure microphone gradient technique for measuring the particle velocity. The specific details of this method are prevalent in

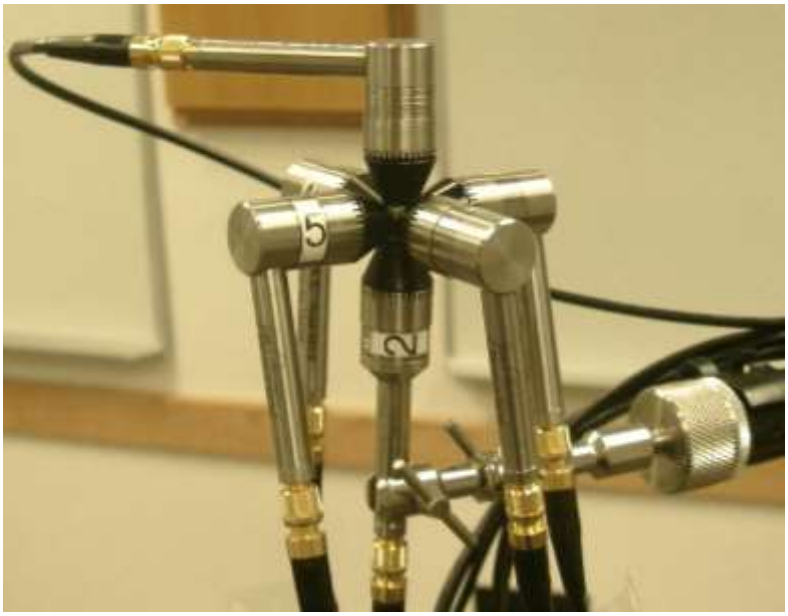

Figure 3.11 View of the G.R.A.S. Type 50VI vector intensity probe.

the literature.[32-34] The technique estimates the particle velocity from the spatial gradient of the sound pressure field, which is approximated from the difference between the signals of closely spaced microphones. The probe consists of three spaced pairs of phase-matched 1/2-inch microphones mounted perpendicular to each other, making it possible to estimate three orthogonal particle velocity components based on the pressure gradients. Each microphone was calibrated at 1 kHz with a piston phone. According to G.R.A.S., the RA004 25 mm spacer placed between each microphone pair permits good accuracy in the bandwidth of approximately 80 Hz to 6.3 kHz. The acoustic pressure at the probe position was estimated by averaging the pressure detected by all six microphones.

The Microflown USP (see Fig. 3.12) uses three pairs of very thin heated platinum wires, mounted orthogonally, to detect the acoustic particle velocity around the wires for each Cartesian component. A microphone is mounted near the velocity sensors to measure the acoustic pressure. The voltage response of each transducer is output to a signal conditioning system, which corrects

Figure 3.12 View of the Microflown Technologies USP (www.microflown.com).

for amplitude and phase mismatch, and provides power to the sensors. The measured responses from each sensor are then calibrated by applying frequency-dependent correction curves unique to each sensor. These are calculated values included with the probe and are derived from modeling the pressure and particle velocity fields due to a piston on a sphere.

Once the pressure and particle velocity magnitude were determined by each probe, the TED, PED, KED, and GED ($\beta = 1/4$) were calculated according to Eqs. (1.8a)-(1.8c) and (1.10), respectively, for a variety of source and test room configurations (see Fig. 3.13). The probes were moved after each source-receiver measurement and repeated until six spatial energy density measurements were taken with each probe.

3.4.3.4 Determining the Sound Power

Once the procedures for acquiring the angle-dependent directivity factor, room constant, and four energy density types had been conducted, the data were used to calculate and compare the sound power estimates of each source in the various test enclosures. The PED values were

(a)

(b)

Figure 3.13 Experimental configurations for testing the GSPF in semi-reverberant spaces. The probes were moved after each source-receiver measurement and repeated until six spatial energy density measurements were taken with each probe. (a) Classroom (small room). (b) Dance studio (large room).

inserted into Eq. (1.9) to estimate the power according to the MHSE. Note that although the directivity factors of each source were known, γ in this formulation is a constant and was therefore set to a chosen value of 1. The KED, TED, and GED values were inserted into the GSPF with $\beta = 0$, $1/2$, and $1/4$, respectively, but also included energy field "correction" terms as outlined in the standards.[1,4] In particular, the low-frequency Waterhouse correction and high-frequency Vorlander corrections were applied to the reverberant field energy:[35-37]

$$<\Pi_s>_t = \frac{1}{I}\sum_{i=1}^{I} \frac{<w_{g,\beta}(r_i)>_t}{\frac{\gamma(\theta_i,\phi_i)}{4\pi r_i^2 c} + \frac{4}{Rc\Omega}} \ , \tag{3.4a}$$

where $\gamma(\theta_i, \phi_i) = \gamma_{m,n}$ from the measured directivitites (rounded to the nearest 5° in each angle), and

$$\Omega = 10^{\frac{23.98V}{cT_{60}}}\left(1 + \frac{S\lambda}{8V}\right) , \tag{3.4b}$$

where λ is the acoustic wavelength. Note that near field correction terms are not included in this formulation, but are discussed in Sec. 4.4.1.

In addition, the changes in the characteristic impedance of air and radiation impedance for actual meteorological conditions were accounted for when converting to power levels:[1,4]

$$L_\Pi = 10log_{10}\left[\frac{<\Pi_s>_t}{\Pi_{ref}}\right] + C_1 + C_2 \ , \tag{3.5a}$$

where

$$C_1 = -10log_{10}\left[\frac{p_s}{p_{s,0}}\right] + 5log_{10}\left[\frac{273.15+\delta}{\delta_0}\right], \tag{3.5b}$$

$$C_2 = -10log_{10}\left[\frac{p_s}{p_{s,0}}\right] + 15log_{10}\left[\frac{273.15+\delta}{\delta_1}\right], \tag{3.5c}$$

where Π_{ref} = 10^{-12} Watts, p_s is the static test-room pressure in kilopascals, $p_{s,0}$ is the reference static pressure of 101.325 kPa, δ is the test-room air temperature in degrees Celsius, δ_0 =314 K, and δ_1 = 296 K. It should be noted that the energy correction terms were also applied to the MHSE [Eq. (1.9)] in the same manner for equitable comparison when using the PED values.

3.4.4 Reference Sound Power

So that the sound power predictions could be compared for accuracy, the three test sources were placed in both the reverberation and anechoic chambers where the precision sound power standards were conducted.[1,4] The ISO 3745 standard for anechoic chambers generally requires only twenty equally-spaced microphone positions on the measurement sphere. However, the 2,664-point set of measurements taken from the directivity factor procedure described in Sec. 3.4.3.1 could also be used according to the standard as long as the proper area weightings were applied. Thus, the sound power curves of the sources were determined in the anechoic chamber with much greater resolution than the precision standard generally requires.

In theory, the frequency-dependent sound power curve obtained for each source should not differ much between the two standards. However, where there are discrepancies, the power curves generated from the 2,664 microphone positions are assumed to be more accurate. Thus, these curves serve as the reference sound power to which all experimental results are compared.

Overlaying the predicted sound power curves with the reference curve provides a beneficial visual comparison. In addition, an arithmetic, 1/3-octave band, root-mean-square deviation (RMSD) was performed between the reference power and each prediction to provide a quantitative comparison in dB. The specific formulation for the RMSD is

$$RMSD = \sqrt{\frac{1}{J}\sum_{j=1}^{J}\left(L_{\Pi_j} - L_{\Pi,ref_j}\right)^2}\ , \tag{3.6}$$

where the index j refers to the jth 1/3-octave band, L_{Π} is the sound power level estimate under comparison, and $L_{\Pi,ref}$ is the sound power level determined from ISO 3745 using all 2,664 spatial measurements.

3.4.5 ISO 3741 Violation Tests

In the reverberation chamber, each source was tested at multiple positions and experiments were conducted for each case by systematically varying microphone positions. In particular, from one to all microphones were placed closer to the source than the ISO 3741 standard permits – some positions chosen to be within less than 1 meter from the source (see Fig. 3.14). In this way, the direct energy component was significant enough to alter the sound power estimates defined by the official calculation procedure. These same tests were repeated by placing the sources and energy density probes in the same positions. However, in this case, the GSPF was implemented in place of the standard to compensate for any non-negligible direct-energy components (see Fig. 3.15). The sound power estimates from the violated standard configurations and the GSPF were compared to each other and to the reference curves.

3.5 Results and Discussion

3.5.1 Directivity Factors

The directivity factors (not directivity functions) of the dodecahedron source at six discrete frequencies are represented in balloon-plot form in Fig. 3.16. Both the distance of the pattern and the color show the value of the directivity factor at each angle. At 103 Hz, the source

Figure 3.14 Test configuration for ISO 3741 in a reverberation chamber where some source-receiver distances have been deliberately set to less than the minimum distance d_{min} designated in the standard.

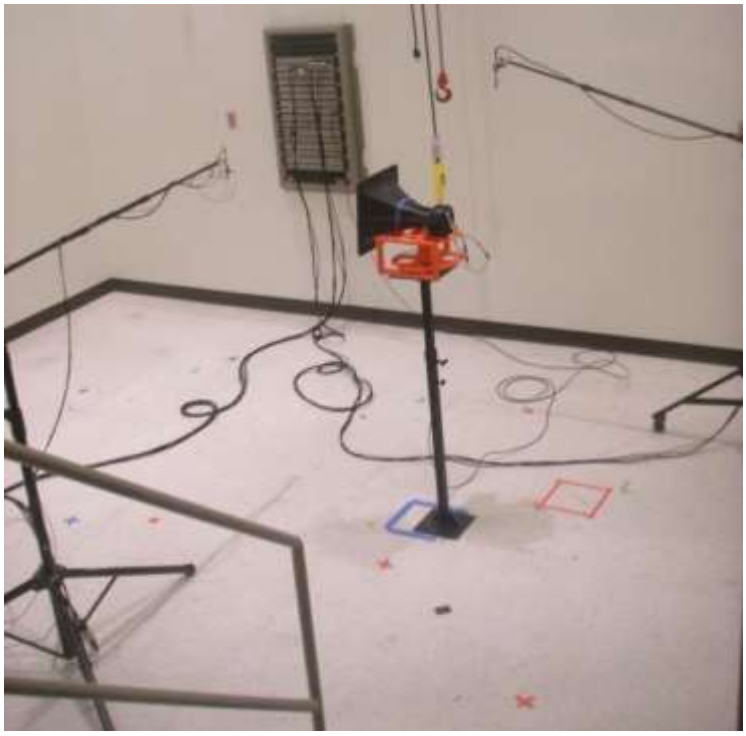

Figure 3.15 ISO 3741 violation test configuration using energy density probes close to the source.

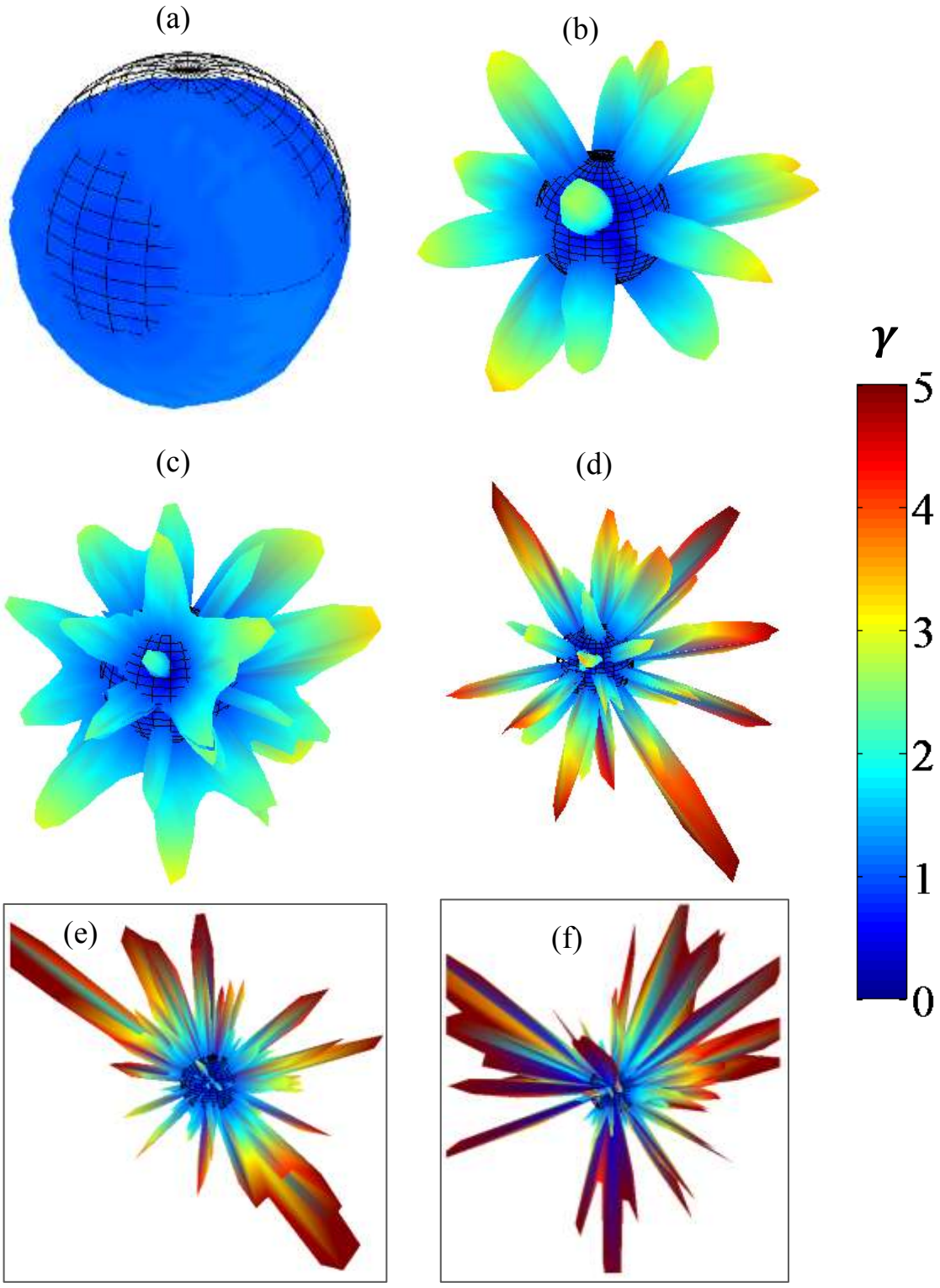

Figure 3.16 The directivity factors of the dodecahedron source depicted in balloon-plot form. The mesh surface represents an omnidirectional source ($\gamma = 1$) radiating the same power. Both the distance of the pattern and the color show the value of the directivity factor at each angle. (a) 103 Hz. (b) 1.9 kHz. (c) 3kz. (d) 5.4 kHz. (e) 7.8 kHz. (f) 9.8 kHz.

radiates nearly omnidirectionally, and continues to do so until approximately 1.2 kHz. At higher frequencies, diffraction, interference, and individual driver directivities interact to form interesting patterns. At 1.9 kHz, distinct lobes of magnitude 2.5 are visible. With increasing frequency, they gradually evolve into the pattern seen at 3 kHz. It is even more complex at 5.4 kHz, many with directivity factor lobes reaching a magnitude of 5. Above 6 kHz, the patterns appear erratic, demonstrating how complex the direct radiation field can become when twelve drivers interact at higher frequencies (e.g., above cone break up). This suggests likelihood for error in defining γ for the dodecahedron when implementing the GSPF at frequencies higher than 1.2 kHz; a small error in measurement angle could correspond to a much different value for γ than expected.

The directivity factor plots differ considerably for the single-driver dodecahedron source, as seen in Fig. 3.17. Like the full dodecahedron source, the single driver nominally radiates an omnidirectional field at 103 Hz. However, it begins to noticeably beam at frequencies as low as 150 Hz. By 1.9 kHz, a large main lobe has developed along the principal axis and by 6.1 kHz, the lobe has reached a magnitude of approximately 15. The main lobe splits into strong multiple lobes as the frequency approaches 8.7 kHz. The source becomes very directional, with γ values as high as 50 along the principal axis, for frequencies of 9.8 kHz and higher. While lobing clearly does exist, the directivity factors of the single-driver dodecahedron source vary more smoothly over angle and frequency than those of the source when all drivers are active.

As expected, the horn-loaded compression driver demonstrates the most extreme directivity in the mid-to-high frequency range (see Fig. 3.18). At 103 Hz, the source radiates nearly omnidirectionally, but signs of lobing occur at frequencies as low as 119 Hz. (Poor measurement coherence resulting from weak output results in the ragged appearance.) A distinct

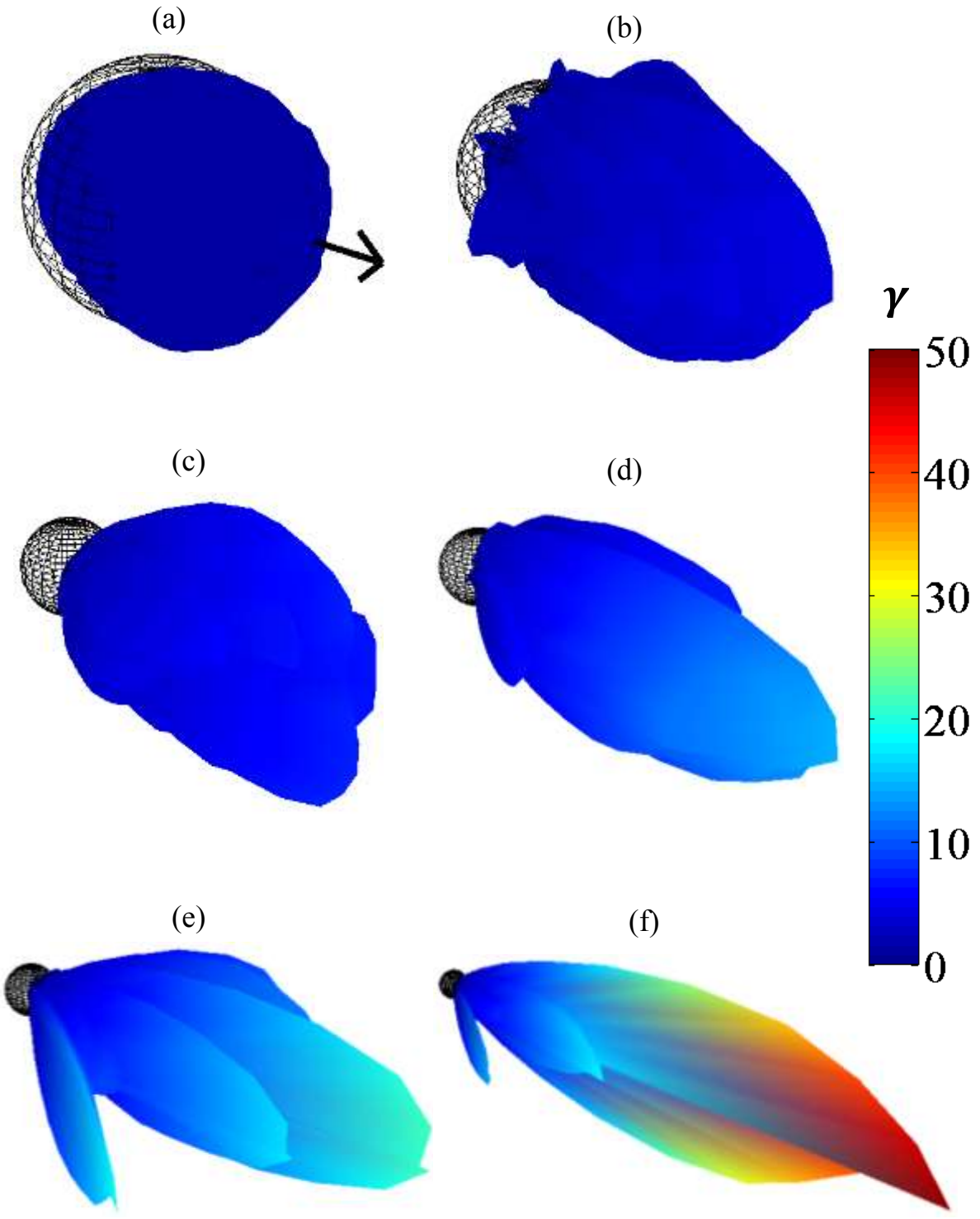

Figure 3.17 The directivity factors of the single-driver dodecahedron source depicted in balloon-plot form. The mesh surface represents an omnidirectional source ($\gamma = 1$) radiating the same power. Both the distance of the pattern and the color show the value of the directivity factor at each angle. The arrow depicts the principal axis. (a) 103 Hz. (b) 1.9 kHz. (c) 4.3 kHz. (d) 6.1 kHz. (e) 8.7 kHz. (f) 9.8 kHz.

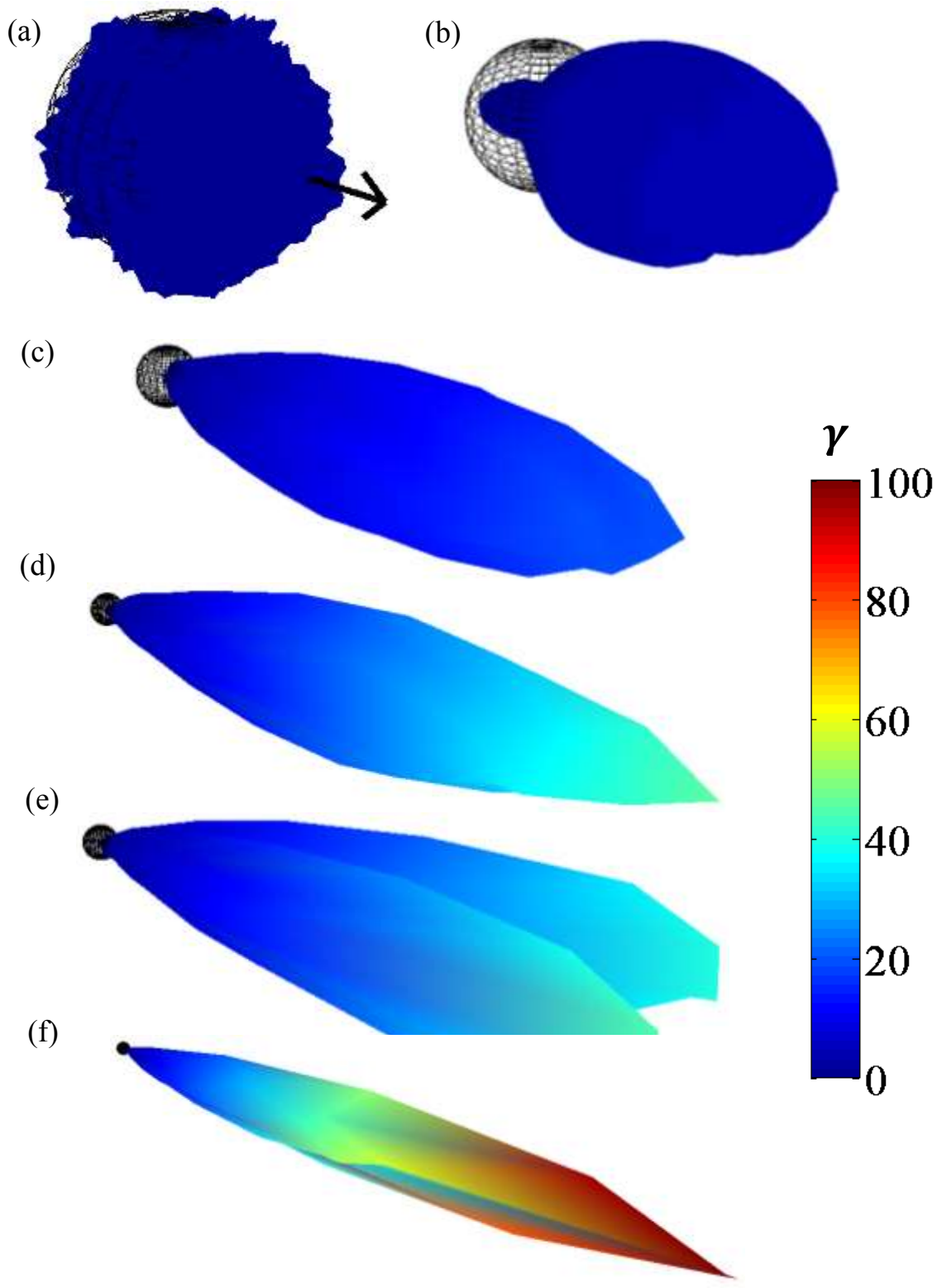

Figure 3.18 The directivity factors of the horn-loaded compression driver depicted in balloon-plot form. The mesh surface represents an omnidirectional source ($\gamma = 1$) radiating the same power. Both the distance of the pattern and the color show the value of the directivity factor at each angle. The arrow depicts the principal axis. (a) 103 Hz. (b) 568 Hz. (c) 1.9 kHz. (d) 5.4 kHz. (e) 7.8 kHz. (f) 15.8 kHz.

front lobe and smaller back lobe are visible at 568 Hz. By 1 kHz, the main lobe has a value of 20. Above 1 kHz, the source demonstrates extreme directivity with values reaching as high as 100 along the principal axis at 15.8 kHz. The directivity of this source emphasizes how important it is to consider γ as a function of both angle and frequency when estimating sound power. The MHSE would introduce significant errors when estimating the frequency-dependent sound power of this source if the value of γ were set to a constant.

3.5.2 Room Constants

Listed in Table 3.1 are the 1/3-octave band absorption coefficients of the four test enclosures. These values were determined as outlined in Sec. 3.4.3.2 and inserted into Eq. (1.1b) to obtain 1/3-octave band room constant values (see Fig. 3.19). With a few exceptions, the rooms exhibited lower absorption at low frequencies and increasing absorption with increasing frequency. The medium-sized room has a small bump in its room-constant curve, while the large room shows substantial absorption at both low and high frequencies. As expected, the reverberation chamber has the lowest absorption at all frequencies. Fig. 3.19 reveals that at any given 1/3-octave band center frequency, there is usually a significant difference in the room constants of the four rooms, thus allowing the GSPF to be tested and explored in a diverse set of reverberant and semi-reverberant environments.

3.5.3 Sound Power Estimates

3.5.3.1 General Discussion

As shown in Figs. 3.20-3.23, the sound power estimates of ISO 3741 and ISO 3745 agree for most of the frequency range of interest. For each test source, the greatest discrepancies lie in

Table 3.1 The 1/3-octave band average absorption coefficients of the four rooms used in the experiments.

Reverb. Chamber (210 m^3)		**Small Room (90.5 m^3)**		**Medium Room (231 m^3)**		**Large Room (532 m^3)**	
Freq. (Hz)	α	Freq. (Hz)	α	Freq. (Hz)	α	Freq. (Hz)	α
100	0.021	100	0.158	100	0.146	100	0.132
125	0.021	125	0.167	125	0.118	125	0.206
160	0.023	160	0.146	160	0.098	160	0.222
200	0.023	200	0.128	200	0.107	200	0.239
250	0.021	250	0.123	250	0.109	250	0.242
315	0.019	315	0.132	315	0.107	315	0.209
400	0.019	400	0.167	400	0.116	400	0.185
500	0.018	500	0.199	500	0.125	500	0.158
630	0.018	630	0.237	630	0.140	630	0.161
800	0.017	800	0.247	800	0.173	800	0.149
1000	0.019	1000	0.259	1000	0.210	1000	0.153
1250	0.022	1250	0.248	1250	0.231	1250	0.185
1600	0.025	1600	0.266	1600	0.163	1600	0.209
2000	0.029	2000	0.250	2000	0.133	2000	0.254
2500	0.032	2500	0.263	2500	0.134	2500	0.268
3150	0.037	3150	0.249	3150	0.143	3150	0.289
4000	0.042	4000	0.263	4000	0.169	4000	0.319
5000	0.055	5000	0.280	5000	0.190	5000	0.351
6300	0.082	6300	0.294	6300	0.213	6300	0.362
8000	0.086	8000	0.353	8000	0.278	8000	0.429
10000	0.103	10000	0.396	10000	0.349	10000	0.494

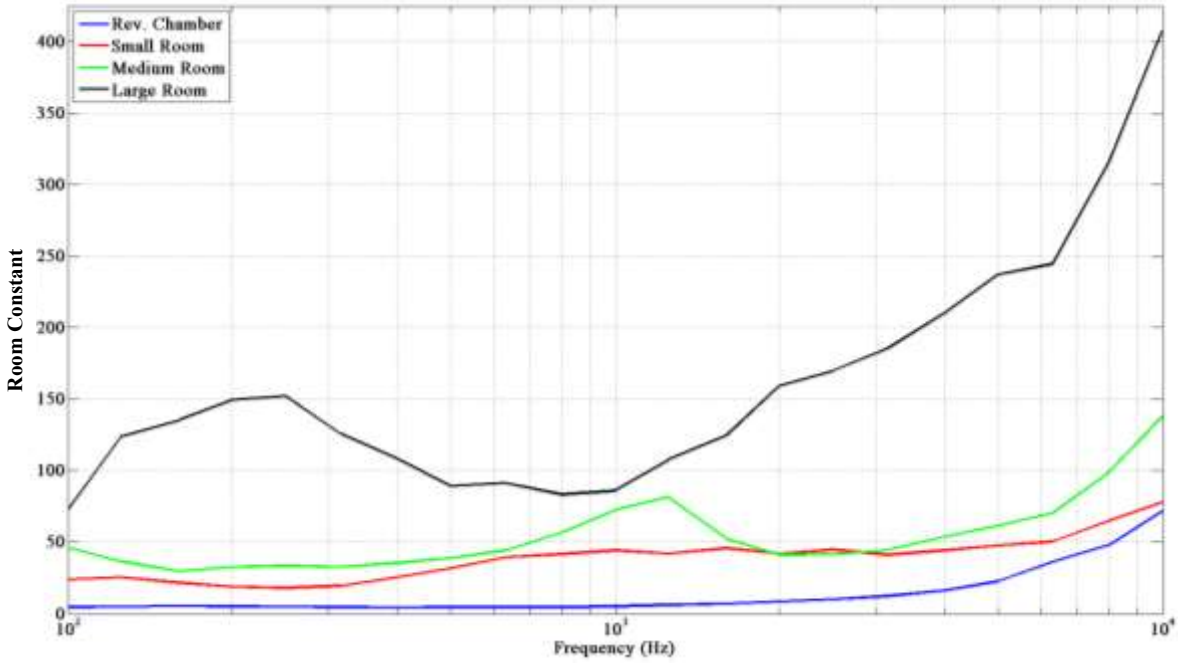

Figure 3.19 The 1/3-octave band room constant R for each of the four test environments.

the ranges below 300 Hz and above 5 kHz. However, as mentioned previously the ISO 3745 anechoic sound power estimates are considered to be the most accurate.

It is not feasible to include in this thesis the sound power estimates of every source-enclosure combination. However, the curves depicted in Figs. 3.20-3.23 do include all sourcesand test rooms in some fashion, demonstrating the general results of the MHSE and GSPF methods including energy field corrections. Note that the two ISO curves in each of these figures are results from the reverberation and anechoic chambers, not from the semi-reverberant enclosures. The PED curve (MHSE) and KED, TED, and GED curves (GSPF) were generated by averaging over two randomly-placed measurements of the respective energy density types. This number was chosen to demonstrate the level of accuracy each method achieves with few measurements. Thus, the results depicted in the figures suggest both the accuracy and convenience of each approach.

The top and bottom plots in each figure include similar results using the G.R.A.S. Type 50VI and Microflown USP, respectively. The reader will quickly notice the differences between the power estimates associated with each sensor. Compared to the Type 50VI, the curves generated by the USP tend to be more widely spread and erratic. As explained previously, much of the discrepancies between the two sensors are attributed to the differing technologies in measuring particle velocity. It appears that the Type 50VI performed more consistently over the wide set of experiments conducted.

The dashed vertical lines represent the lower and upper usable frequencies of the measurements. The lower limit is represented by the Schroeder frequency of the test room where diffuse-field assumptions begin to break down. The upper limit is held constant at 6.3 kHz for all figures and represents the published frequency limitation of the Type 50VI which is based on the

set distance between microphones when mounted. The RMSD values are calculated over the usable bandwidth. Even though the USP is not necessarily limited at 6.3 kHz, the same upper frequency limit was set for both probes for equitable comparison. It should be noted, however, that estimated overall sound power levels or overall levels (OALs) for each method are computed over the entire bandwidth of interest (100 Hz < f < 10 kHz).

In general, the MHSE (which uses PED and a γ value set to 1) performs well at low frequencies where the sources radiate omnidirectionally. However, the method struggles to maintain accuracy in the mid-to-high frequency range where it often strongly overestimates the power. This is not the case for the GSPF as it quite consistently matches the reference power curves for each source-enclosure configuration.

At the lower frequencies, where the sources struggle to maintain high sound power output, the background noise of the HVAC system in the semi-reverberant enclosures could become significant, introducing significant estimation errors below 200 Hz. Appropriate measurement and subtraction of ambient noise could help alleviate this problem, but it was not carried out for this phase of the work.

3.5.3.2 Specific Source-Enclosure Configurations

The sound power estimates for the single-driver dodecahedron source located in the reverberation chamber are shown in Fig. 3.20. The estimates generated from each method and energy density probe are very similar and follow the reference curve closely. According to the reference, the OAL of the source is 88.5 dB for the frequency range of interest. The power estimates from the MHSE and GSPF fall within $\pm$ 0.4 dB of this value for both probes. It is interesting to note that each of these curves more closely follows the reference curve generated from the method in the anechoic chamber (ISO 3745), even though the measurements were

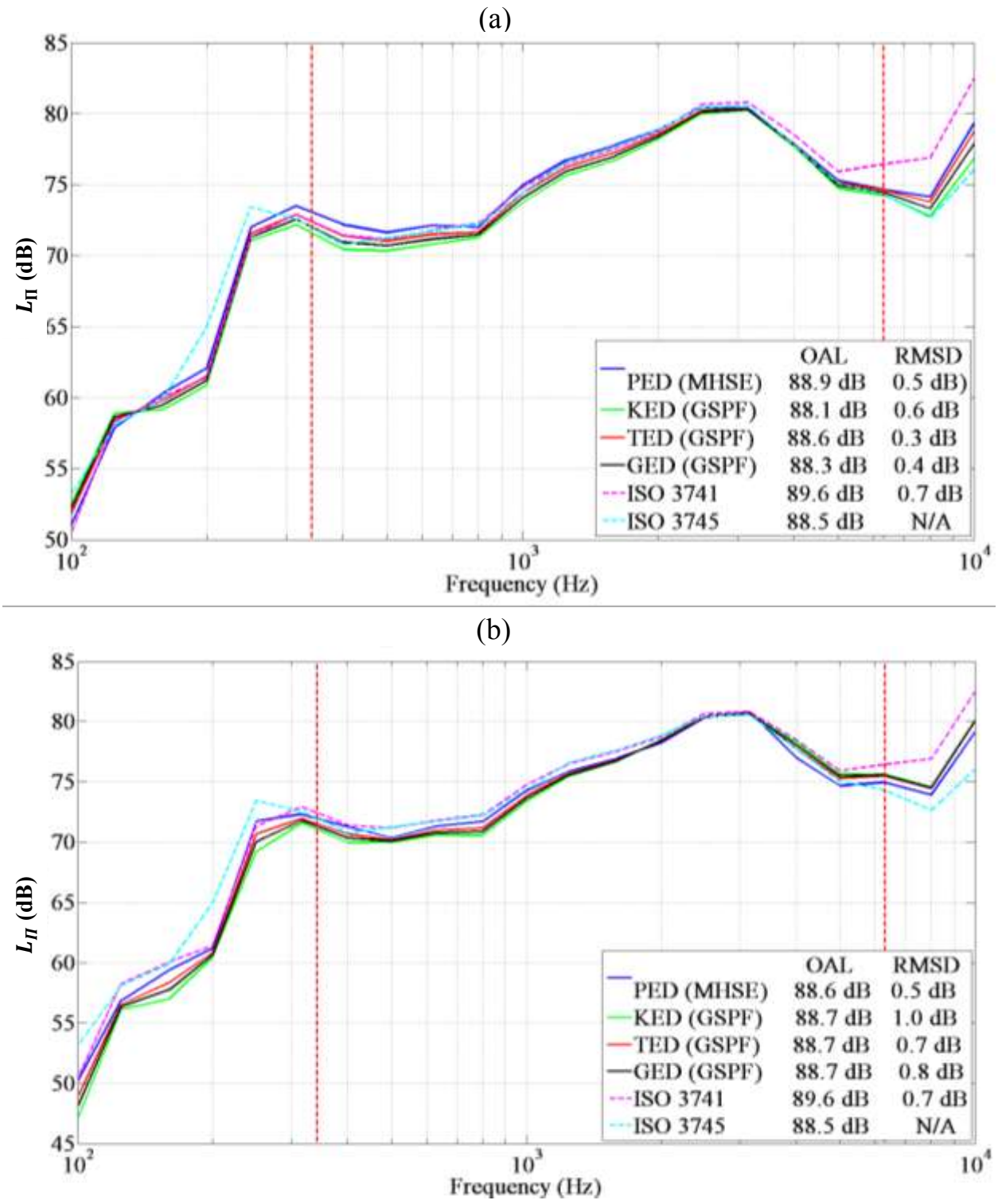

Figure 3.20 The 1/3-octave band sound power estimates of the single-driver dodecahedron source in the reverberation chamber determined by the MHSE or GSPF. Two randomly placed energy density measurements of each type were taken and used in the respective equation. The dashed vertical lines represent the lower and upper usable frequencies of the measurements. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

conducted inside the reverberation room. It is believed that the MHSE equation performs relatively well in this setting because the measurement positions were far enough away that the direct energy field component was negligible and the erroneous value of $\gamma = 1$ did not introduce much error. It would also appear that the two spatial PED measurements were able to adequately estimate the mean energy within the room, which may not always be the case in fields known to demonstrate high spatial variance.

The sound power estimates of the horn-loaded compression driver inside the small room are demonstrated in Fig. 3.21. The MHSE dramatically overestimates the power (up to 10 dB) at approximately 500 Hz and above, which is where the source begins to demonstrate significant beaming (refer to Fig. 3.18b). In this small semi-reverberant environment, the direct energy field component is quite significant, especially for such a highly directional source. The set value of $\gamma = 1$ introduces significant error in the power estimation. Depending on the probe, the MHSE estimates OALs at least 6 to 7 dB higher than the reference OAL of 101.5 dB. For the GSPF, the results from the USP have a much greater spread than those from the Type 50VI. When using the GED ($\beta = 1/4$), the overall level of the GSPF from the USP is 3.5 dB lower, while the Type 50VI estimates an overall level within 0.2 dB. The GSPF demonstrates the same trend as seen in Fig. 3.20, where the estimates match the reference curve nicely at the very high frequencies (see Fig. 3.21a). These results exhibit the superiority of the GSPF method.

In Fig. 3.22, the results are displayed for the single-driver dodecahedron in the medium-sized enclosure, where the MHSE again struggles at higher frequencies. When implementing the GED ($\beta = 1/4$), the overall level of the GSPF by the USP is 2.4 dB lower than the reference value of 88.5 dB, while that of the Type 50VI is only 1 dB lower. The power estimates for both probes fall between the two reference curves above 7 kHz.

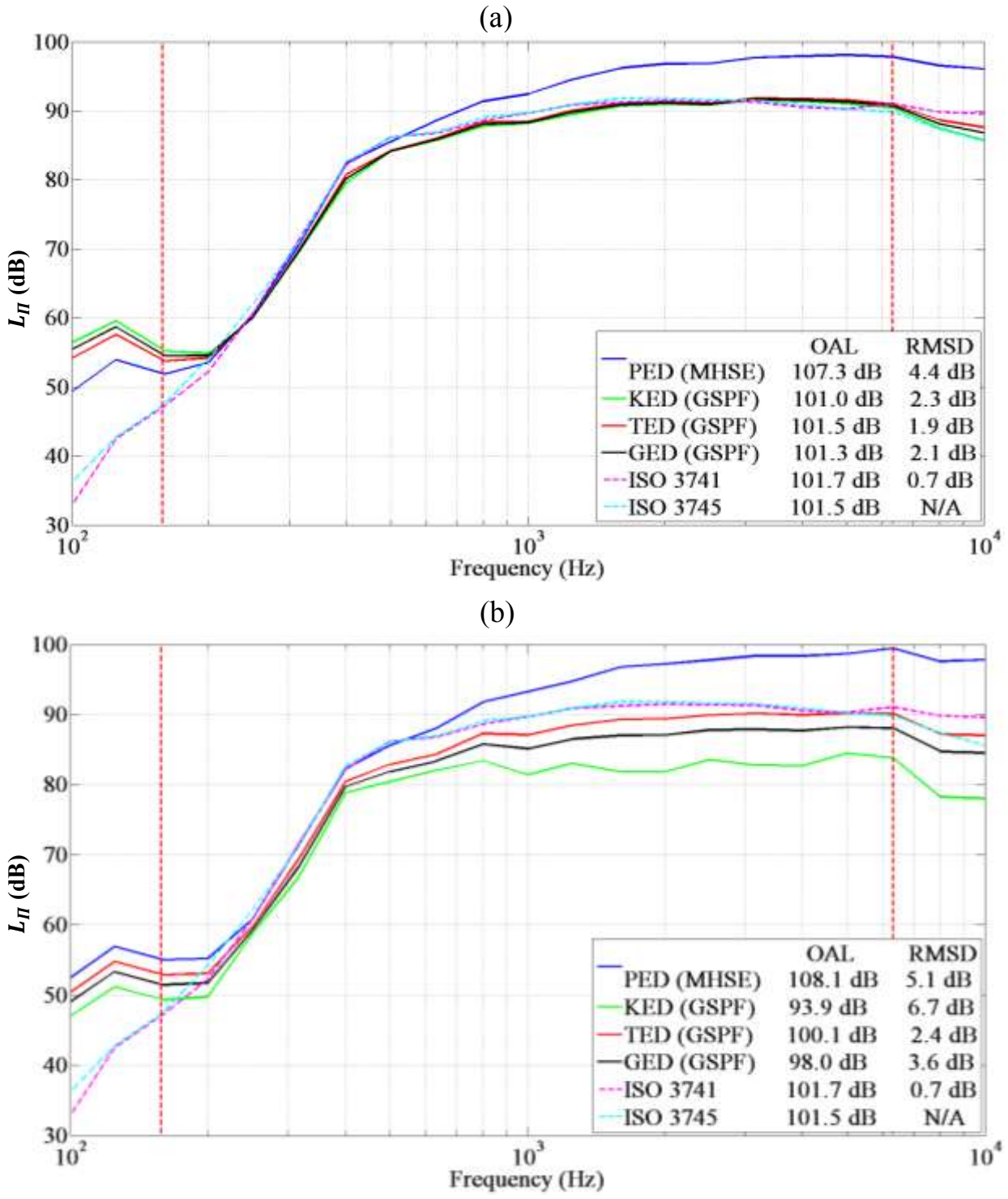

Figure 3.21 The 1/3-octave band sound power estimates of the horn-loaded compression driver in the small room determined by the MHSE or GSPF. Two randomly placed energy density measurements of each type were taken and used in the respective equation. The dashed vertical lines represent the lower and upper usable frequencies of the measurements. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

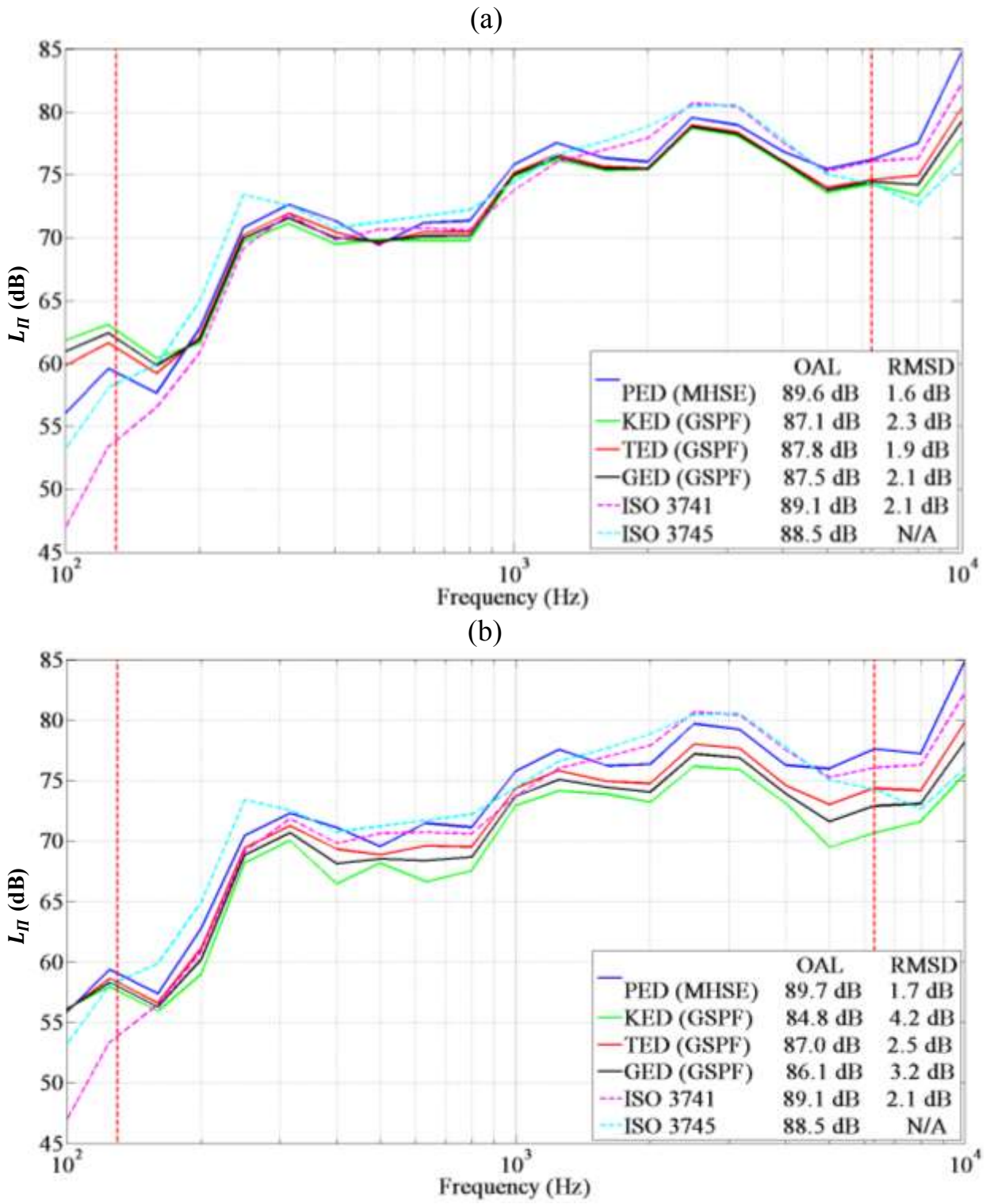

Figure 3.22 The 1/3-octave band sound power estimates of the single-driver dodecahedron source in the medium-sized room determined by the MHSE or GSPF. Two randomly placed energy density measurements of each type were taken and used in the respective equation. The dashed vertical lines represent the lower and upper usable frequencies of the measurements. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

The results of the full dodecahedron in the large room are shown in Fig. 3.23. All of the power estimates from the USP probe overestimate the power above 3 kHz. However, using GED, the overall level from the GSPF is only 0.7 dB higher than the reference value of 91.5 dB. The Type 50VI performs better at the higher frequencies and estimates the OAL within 0.1 dB. The power estimates for both probes again fall between the two reference curves above 7 kHz.

The results from the RMSD analysis are summarized in Table 3.2, where the lowest values for each test configuration and probe are highlighted. In general, the GSPF estimates the sound power with the least amount of deviation from the reference power when the GED ($\beta = 1/4$) or TED is the measured field quantity. At times, these estimates also outperform the ISO 3741 standard in terms of RMSD. The MHSE equation estimates the power with the lowest RMSD within the medium-sized room. The range in RMSD between the MHSE and the GSPF are greater in the semi-reverberant enclosures than in the reverberation room.

Table 3.2 The RMSD of the sound power estimates determined from the PED, KED, TED, GED ($\beta = 1/4$), and ISO 3741 standard over the usable bandwidth. The MHSE was implemented for PED values, while the GSPF was implemented for the KED, TED, and GED values. Energy-field corrections were applied to all estimates. The reference sound power curve to which each method was compared was that obtained from the 2,664-measurement point ISO 3745 method. The lowest RMSD values for the given test configuration and probe are highlighted.

RMSD (dB)								
	Rev. Chamber (Single Driver)		**Small Room (Horn)**		**Medium Room (Single Driver)**		**Large Room (Dodecahedron)**	
	Type 50VI	USP	Type 50VI	USP	Type 50VI	USP	Type 50VI	USP
PED	0.5	**0.5**	4.4	5.1	**1.6**	**1.7**	1.9	2.2
KED	0.6	1.0	2.3	6.7	2.3	4.2	1.6	1.6
TED	**0.3**	0.7	1.9	2.4	1.9	2.5	**1.1**	**1.4**
GED ($\beta = 1/4$)	0.4	0.8	2.1	3.6	2.1	3.2	1.3	**1.4**
ISO 3741	0.7	0.7	**0.7**	**0.7**	2.1	2.1	1.6	1.6

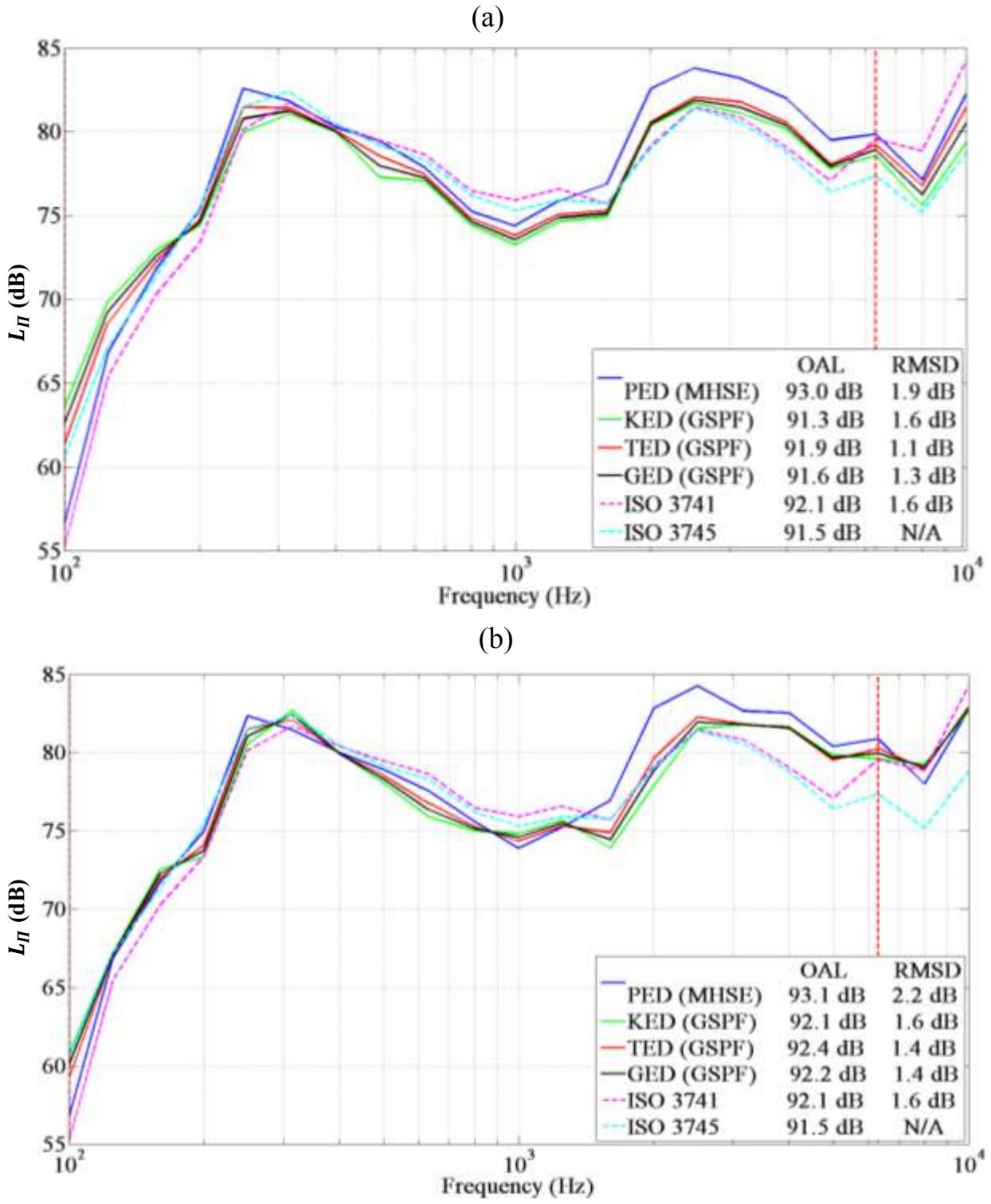

Figure 3.23 The 1/3-octave band sound power estimates of the dodecahedron source in the large room determined by the MHSE or GSPF. Two randomly placed energy density measurements of each type were taken and used in the respective equation. The dashed vertical lines represent the lower and upper usable frequencies of the measurements. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

3.5.4 ISO 3741 Violation Tests

Figures 3.24-3.26 depict the estimated sound power from the reverberation chamber ISO 3741 violation tests for all three sources. These particular results correspond to a violation of all six measurement positions in terms of proximity to the source under test. In every case, calculating the sound power according to standard when the positions are in violation introduces very large errors, especially in the higher frequency range.

For the dodecahedron and both probes, the OAL estimated by the violated standard is 2.4 dB higher than that estimated if the measurement positions are not in violation, and 3.2 dB higher than the reference OAL from the ISO 3745 standard (see Fig. 3.24). Note that despite the close proximity of all six measurement positions, the GSPF matches the reference curve very nicely throughout the entire bandwidth, estimating an OAL within 0.6 dB of the reference level with the USP and within 0.2 dB with the Type 50VI.

The effects of the violating source-receiver distances are more prevalent with the more directional sources. This is demonstrated in Fig. 3.25, which corresponds to the single-driver dodecahedron source and Fig. 3.26, which corresponds to the horn-loaded compression driver. In both cases, the errors are significant, introducing OAL estimates that are as much as 6.4 dB higher than those resulting from the correctly controlled procedure. However, because the GSPF accounts for the direct energy field component, it is able to estimate the sound power more accurately over the bandwidth. It estimates the OAL within 0.4 dB of the reference level for the single-driver source and within 1.3 dB for the horn-loaded compression driver.

The RMSD values of these violation tests are summarized in Table 3.3, where the lowest values for each source and probe are highlighted. Note that the GSPF estimates the sound power

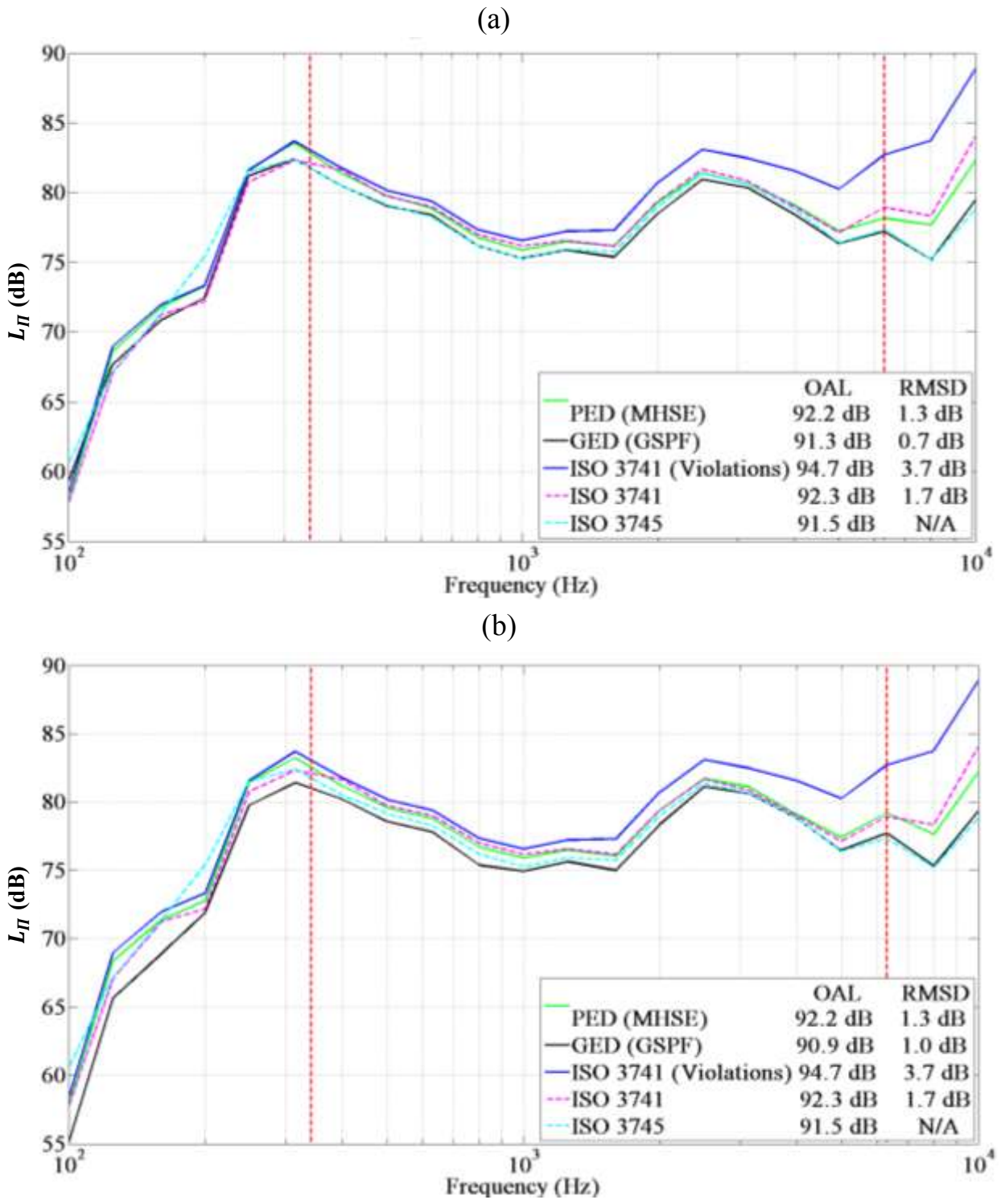

Figure 3.24 The averaged 1/3-octave band sound power estimates from six varying positions placed close to the dodecahedron source, in violation of ISO 3741 in the reverberation chamber. The three "ISO" curves utilize microphones as sensors while the PED and GED curves use energy density probes, and the MHSE and GSPF, respectively. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

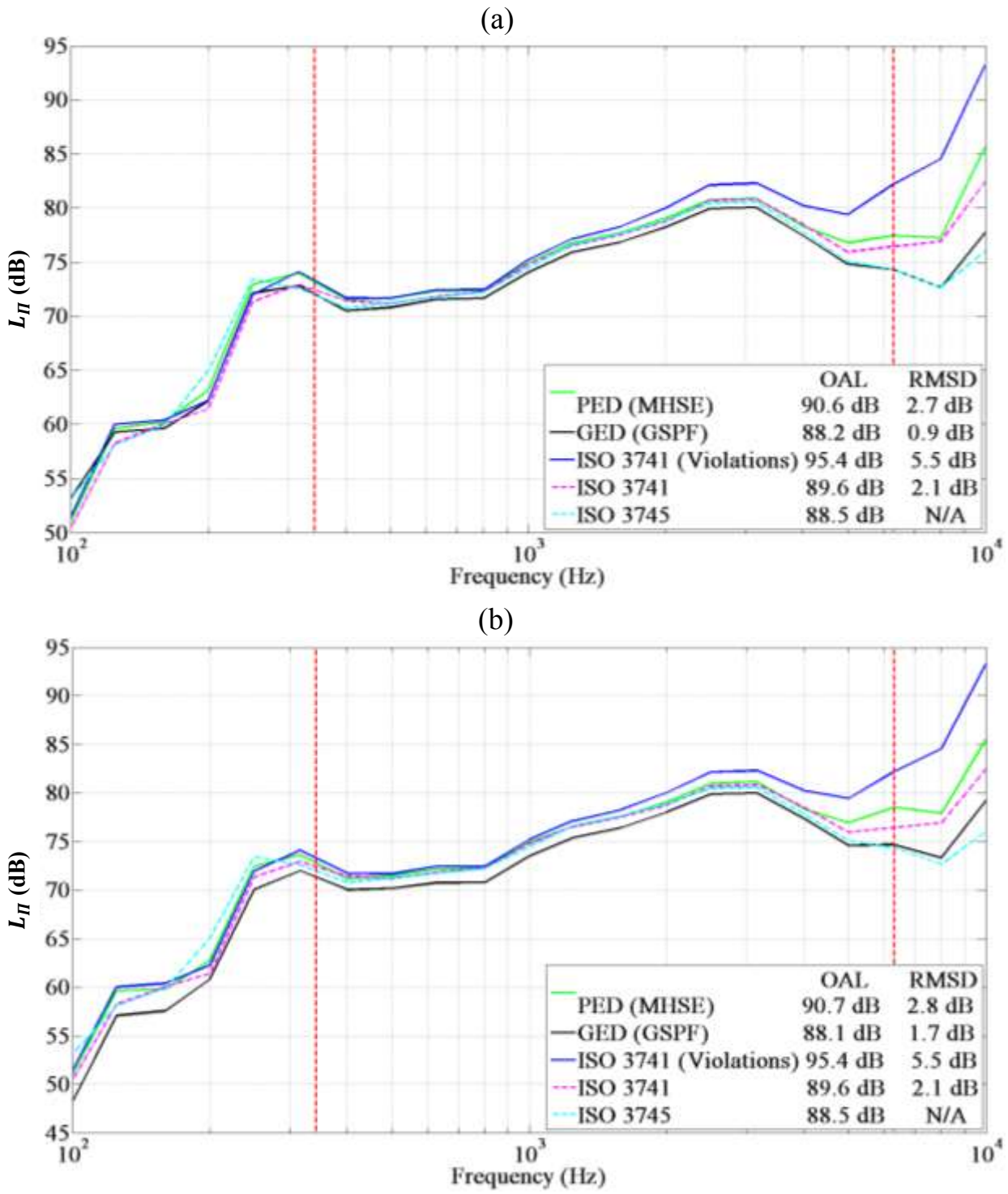

Figure 3.25 The averaged 1/3-octave band sound power estimates from six varying positions placed close to the single-driver dodecahedron source, in violation of ISO 3741 in the reverberation chamber. The three "ISO" curves utilize microphones as sensors while the PED and GED curves use energy density probes, and the MHSE and GSPF, respectively. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

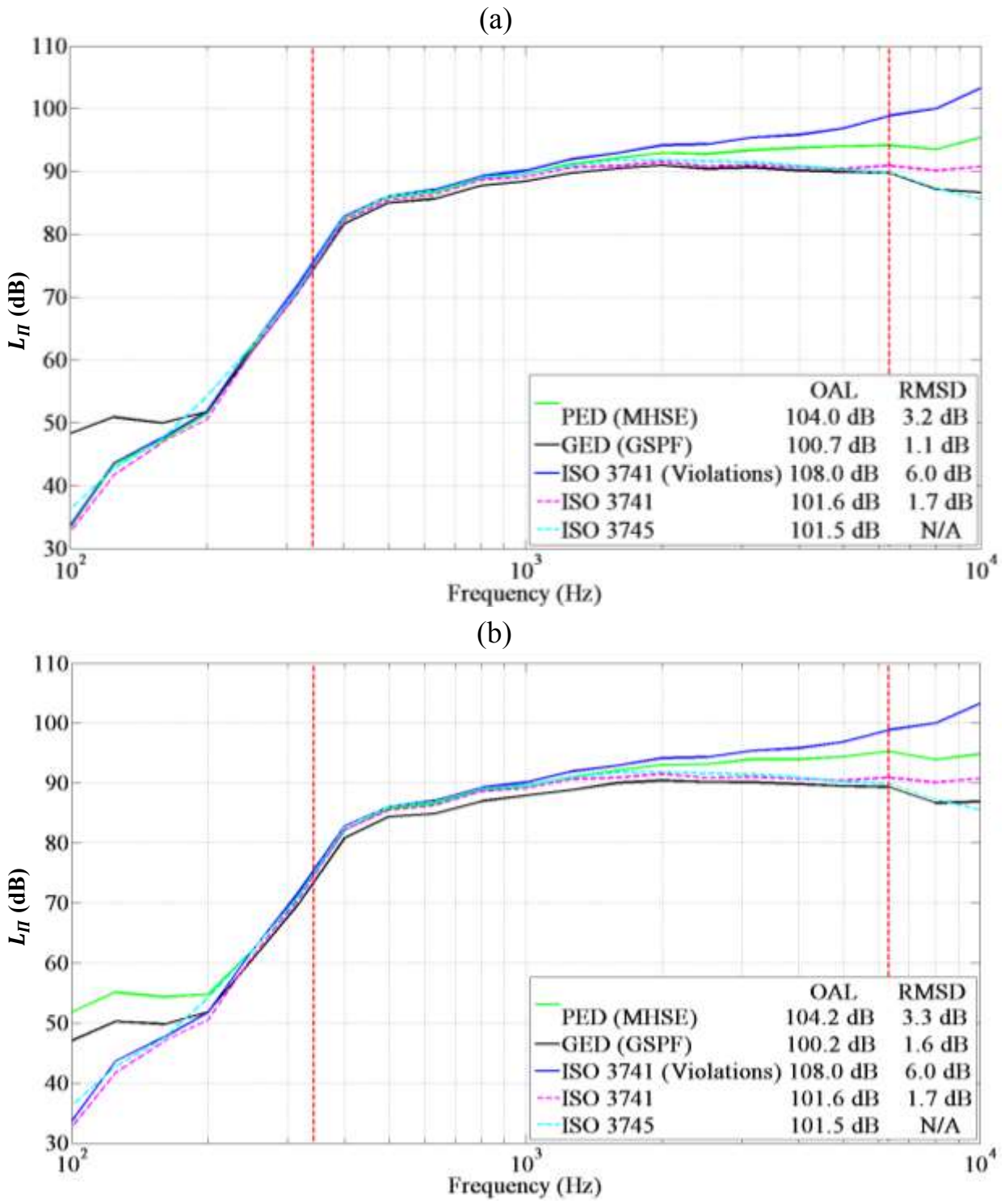

Figure 3.26 The averaged 1/3-octave band sound power estimates from six varying positions placed close to the horn-loaded compression driver, in violation of ISO 3741 in the reverberation chamber. The three "ISO" curves utilize microphones as sensors while the PED and GED curves use energy density probes, and the MHSE and GSPF, respectively. The power curve determined by ISO 3745 serves as the reference in the RMSD analysis. The energy density fields were measured by (a) the G.R.A.S. Type 50VI, and (b) the Microflown USP.

Table 3.3 The RMSD of the sound power estimates determined from the ISO 3741 violation tests. The reference sound power curve to which each method was compared was that obtained from the 2,664-measurement point ISO 3745 method. The lowest RMSD values for the given test configuration and probe are highlighted.

RMSD (dB)						
	Dodecahedron		**Single Driver**		**Horn**	
	Type 50VI	USP	Type 50VI	USP	Type 50VI	USP
PED	1.3	1.3	2.7	2.8	3.2	3.3
GED(β = 1/4)	**0.7**	**1.0**	**0.9**	**1.7**	**1.1**	**1.6**
ISO 3741 (Violations)	3.7	3.7	5.5	5.5	6.0	6.0
ISO 3741	1.7	1.7	2.1	2.1	1.7	1.7

with the least amount of deviation from the reference power in every case, including that of the ISO 3741 standard, which is specifically designed for reverberant environments.

3.6 Specific Conclusions

The experimental exploration of the GSPF in both diffuse and semi-reverberant fields has demonstrated its effectiveness in comparison to the MHSE. The fundamental characteristics thatproduce its improved performance are its use of the more spatially uniform GED field and its implementation of a spatially varying and frequency-dependent directivity factor. The specific conclusions associated with the methods performed in this chapter are as follows:

1. In general, the MHSE performs well at estimating the sound power in the low-frequency range, were each source radiates omnidirectionally. However, it struggles to maintain accuracy in the mid-to-high frequency range where it tends to overestimate the power. These errors become more prevalent with increasing source directivity, introducing OAL errors as high as 6.6 dB when compared to the ISO 3745 reference.
2. For all sources in each test room, the GSPF tends to follow the reference power curves performed by the high-resolution ISO 3745 standard quite nicely, producing OAL

estimates within 1 dB or less. The results are generally better with the G.R.A.S. Type 50VI probe.

3. When compared to ISO 3745, the RMSD values are generally the lowest when the GSPF is employed to estimate the sound power of the sources, even outperforming the ISO 3741 standard in some configurations.
4. The differences in RMSD between the MHSE and the GSPF are generally greater with increasing direct-field energy in semi-reverberant enclosures.
5. For each source located in the reverberation room, the sound power calculated according to ISO 3741 when the minimum source-receiver distances are violated involved large errors, especially in the higher frequency range. For the dodecahedron and single driver sources, the overall levels were estimated 1-2 dB higher than the case of no violations. For the horn-loaded compression driver, they were as much as 6.4 dB higher.
6. Despite close proximity of all six measurement positions, the GSPF closely followed the reference curve throughout the entire frequency band of interest. It estimated OALs within 1.3 dB for the horn-loaded compression driver and within 0.5 dB for the other sources.
7. The GSPF estimates the sound power with the least amount of deviation (RMSD) from the reference power in every case, even outperforming the ISO 3741standard in a reverberation room.

These conclusions suggest that the GSPF more accurately estimates sound power within semi-reverberant enclosures than the MHSE equation. In addition, the ISO 3741 measurement constraints on the proximity of receiver to the source can be removed when using the GSPF in

reverberation rooms because it accounts for the direct energy field component. This allows for more convenient power measurements for highly directional sources.

Chapter 4

The Two-Point In Situ Method Using Generalized Acoustic Energy Density

The results of Ch. 3 demonstrate that the GSPF accurately estimates the sound power radiation of a variety of sources in multiple semi-reverberant enclosures. However, the accuracy of the method is dependent on how well the directivity factor of the source and room constant are known [refer to Eq. (3.4a)]. This chapter outlines some significant challenges in determining both quantities and how these challenges reduce the practicality and convenience of the method. It also introduces a new technique called the two-point in situ method, based on the same formulation, that was developed specifically to overcome these challenges.

4.1 Specific Background: Improving Estimates of the Room Constant

Significant effort has been expended over the last fifty years to improve techniques of measuring the absorption in enclosures and in calculating the room constant R. This section briefly summarizes this work.

4.1.1 Room Absorption

A few commonly prescribed techniques for determining the average absorption coefficient of an enclosure are the Sabine coefficient method and the reverberation time method. The average absorption coefficient determined by the Sabine method is expressed in the following equation:

$$<\alpha>_s = \frac{\sum S_i \alpha_i}{\sum S_i} \ , \tag{4.1}$$

where $<\alpha>_s$ is the spatially-averaged absorption coefficient of the enclosure, S_i is the surface area of the *i*th boundary of the room, and α_i is the corresponding absorption coefficient of that boundary surface. The values of each α_i are generally taken from published values for the specific building materials and are assumed to be independent of boundary location for large, live rooms.

An alternate way of determining the average absorption is to measure the reverberation time. Since this is an in situ measurement, the technique also includes the effects of the room contents and air absorption. While Sabine introduced a reverberation time method as well, past research[29] has shown that the Eyring formulation is better for cases of higher surface absorption. It is expressed as

$$T_{60} = \frac{0.161V}{-Sln(1 - <\alpha>_s)} \ . \tag{4.2}$$

Neubauer[38] introduced a method in 2001 that estimates the average absorption coefficient of rooms with non-uniformly distributed absorption by using a modified Fitzroy equation.[39]

4.1.2 Calculation Methods for the Room Constant

Investigators in this field are in disagreement as to the most appropriate analytical expression for the room constant R. Hopkins and Stryker[5] and Beranek[40] suggest the formulation

$$R = \frac{S<\alpha>_s}{1 - <\alpha>_s}, \tag{4.3}$$

as given in Eq. (1.1b). Others[41] equate the room constant to the number of absorption units in the room:

$$R = S<\alpha>_s \ . \tag{4.4}$$

In addition, many[23, 35-37, 41-44] suggest modifying the room constant to account for air absorption and energy field anomalies.

In situ methods can avoid analytical expressions and correction factors for the room constant by using the MHSE [refer to Eq. (1.9)] and direct measurement. For example, the walk-away method[45,46] indirectly measures the acoustical characteristics of a room by determining the increase in sound pressure level of a source within the room compared to the sound pressure level of the same source in a free field. Thus, the room constant R can be found through the use of a sound source whose directivity factor γ is known for a given direction, by placing the source in the test room, and then obtaining, at a given distance r in the specified direction, the difference in sound pressure level in the room and the sound pressure level that would exist in free space. The equation for the walk-away method room constant prediction is[46]

$$R = \frac{16\pi r^2}{\gamma(10^{\frac{\Delta L_p}{10}} - 1)} \ , \tag{4.5}$$

where ΔL_p is the associated difference in sound pressure level. Note the pitfalls of this formulation, such as the need to predict the free-field 6 dB per doubling line from spaced measurements and the fact that the reverberant field is completely neglected in this process.

Another in situ technique, called the loss due to distance method,[47] is somewhat similar to the walk-away method. A source is assigned a directivity factor of 1 to 8, depending on the space, and placed in the test room where the difference in sound pressure level at several pairs of

distances r and $2r$ are measured for multiple angles. The average difference in sound pressure level is then considered to be representative of the room. The room constant formulation for the loss due to distance follows as[47]

$$R = \frac{64\pi r^2(1 - 10^{\frac{\Delta L_p}{10}})}{\gamma(10^{\frac{\Delta L_p}{10}} - 4)}. \tag{4.6}$$

There also exists a reference sound power method where a sound source of known sound power is placed in the test room. By subtracting sound pressure levels $L_{p,r}$ measured in the reverberant field from the known sound power level of the reference source L_{Π}, the value of the room constant can be predicted using the following equation:[47]

$$R = 10^{\frac{(L_{\Pi} - L_{p,r} + 6)}{10}}. \tag{4.7}$$

4.2 Specific Motivations

It should be noted that a current ISO standard [7] exists for determining sound power levels by way of in situ measurements. However, the test room must be sufficiently reverberant to cause the directivity of the source under test to have an insignificant influence on the measured sound pressure levels. A standardized in situ method does not currently exist to simultaneously account for the direct and reverberant energy fields to determine the sound power of a source in semi-reverberant enclosures. The GSPF equation, explored in Chs. 2 and 3, demonstrates potential to solve this issue, but it too has significant challenges.

As previously stated, the accuracy of the method is dependent on how well the directivity factor and room constant are defined. The results of Ch. 3 reveal that inclusion of a spatially and frequency-dependent directivity factor consistently yields more accurate sound power estimates. However, in order to obtain such data, it was necessary to place each source in a qualified

anechoic chamber and perform the lengthy 2,664-measurement point procedure described in Sec. 3.4.3.1. If the goal is to determine the sound power radiation of each source, it would not make sense for one to spend the time and resources necessary to complete this procedure as only one step in the GSPF. It would be simpler to follow the ISO 3745 standard, where this same procedure allows one to estimate the power immediately. Thus, in this regard, the GSPF is not very practical.

The GSPF also requires an accurate knowledge of the room constant R. Past work on this topic has proven to be quite complex, where each approach has its own unique set of challenges. For example, both the Sabine method [Eq. (4.1)] and reverberation time method [Eq. (4.2)] for determining the average absorption coefficient assume simple rectangular rooms with a diffuse sound field. These are often not the characteristics of industrial environments or other enclosures where in situ measurements are performed. In fact, actual volume and surface areas of common rooms are often unknown. In addition, once the average absorption coefficient is obtained, there are disagreements on which formulation one should use to calculate R and how to apply correction terms. The techniques developed to overcome these challenges, like the walk-away method, loss due to distance method, and reference sound power method, measure the room constant directly. However, both the walk-away method and loss due to distance method are cumbersome, requiring a large number of measurement positions and a prior knowledge of the directivity factor of the source under test. The reference sound power method does not require knowledge of the directivity factor, though it does require measurements to be taken at distances wherein the direct energy component is negligible. This may not always be feasible. Since the GSPF relies on accurate values for the room constant, its practicality also suffers from the same challenges discussed earlier.

The GSPF performs well in semi-reverberant enclosures because it utilizes a more spatially uniform energy quantity for field measurements. It simultaneously considers both the direct and reverberant energy fields by including a spatial and frequency-dependent directivity factor and absorption effects of the room. Nevertheless, the method is less practical due to the lengthy and complex procedures required to obtain the directivity factor and room constant. Perhaps these challenges could be solved by developing a method similar to the GSPF that determines both the directivity factor and room constant in situ. This would significantly improve the convenience and practicality of the GSPF, which has already proven to be accurate in determining sound power.

4.3 Specific Objectives

The following important questions will be answered by the experimental methods discussed in this chapter:

1. By way of in situ measurements in semi-reverberant enclosures, is it possible to determine both the frequency-dependent room constant of the enclosure and the frequency-dependent directivity factor of the source under test in one experimental procedure?
2. Can the GSPF be implemented with these in situ measurements to calculate the sound power within semi-reverberant enclosures?

The answers to these questions could have a significant impact on the current methods of measuring sound power.

The aims of the experimental methods discussed in this chapter are to answer the above questions through the following objectives:

1. Investigate and determine a way to measure both γ and R in situ within a variety of semi-reverberant enclosures.
2. Determine if it is feasible to implement the GSPF with γ and R values determined from in situ measurements.
3. Characterize the accuracy, associated benefits, and limitations of utilizing the in situ approach for determining sound power within semi-reverberant enclosures.

4.4 Methods

This section introduces a technique called the two-point in situ method. Its primary purpose is to provide an efficient procedure for determining the sound power output of a source located within semi-reverberant enclosures. The method implements the GSPF in an iterative approach to systematically estimate the directivity factor of the source under test, the room constant, and the energy density field. The details of the two-point in situ method are outlined first. The efforts taken to experimentally verify the method are then explained.

4.4.1 The Two-Point In Situ Method

The governing equation associated with the two-point in situ method is similar to that used in the loss due to distance method, by which one calculates the room constant in terms of field measurements and the directivity factor of the source [refer to Eq. (4.6)]. However, it utilizes a variation of the GSPF to improve both the accuracy and versatility of determining the room constant, as well as estimating the directivity factor of the source. The procedure then uses the field measurements required to obtain γ and R, by inserting them back into the usual GSPF to determine the sound power of the source. In all, the two-point in situ method requires only four GED measurements, and a single reference directivity source (a broadband source in which the

frequency-dependent directivity factor for a given angle is known). This is unlike other commonly used methods that employ a reference sound power source (e.g., ISO 3747 and the reference sound power method).

The two-point in situ method is conducted as followed:

1. Place both the source under test and the reference directivity source within the enclosure. Both should be placed at least 1 m from any reflecting surface, including the floor, and each other.
2. Using a broadband signal, excite only the reference directivity source.
3. Take two consecutive GED measurements at varying distances along a single angle relative to the reference directivity source, for which γ is known (see Fig. 4.1). Let r_1' and r_2' be the distances from the reference directivity source to positions 1 and 2, respectively (the primes indicating field positions associated with the reference directivity source measurements). The reference directivity source must operate at the same amplitude and with the same excitation signal for each measurement.

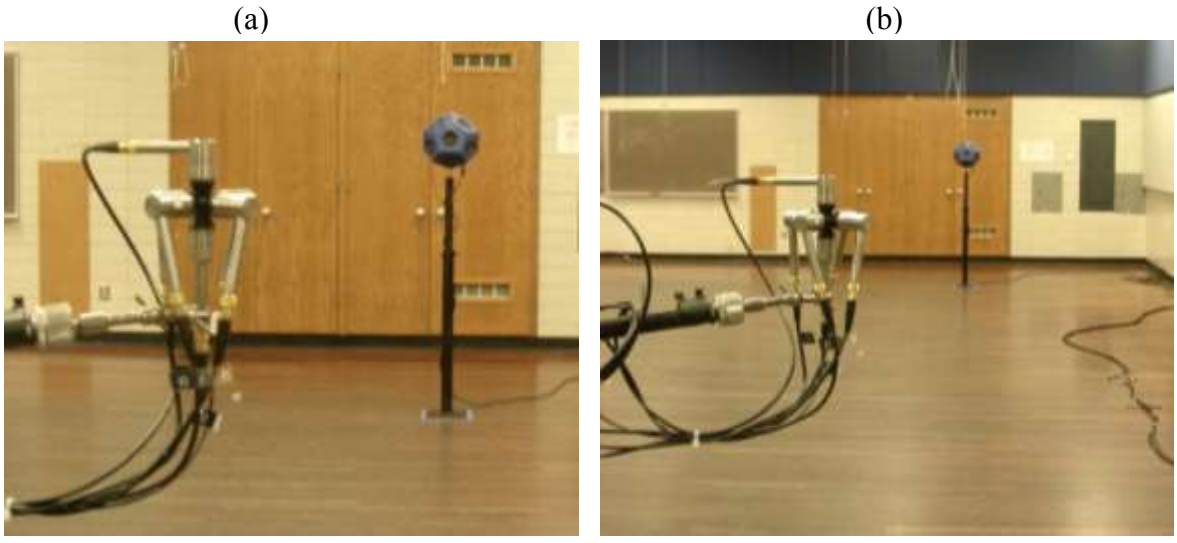

Figure 4.1 Example configuration for step 3. The dodecahedron is the reference directivity source, and the energy density probe is located at position 1 (a), and position 2 (b). Both positions are along the same angle relative to the reference directivity source.

4. Calculate the room constant, R, from the following equation (see Appendix C.1):

$$R_{in\ situ} = \frac{16\pi\left[1 - \frac{<w_{g,1/4}(r_1')>_t}{<w_{g,1/4}(r_2')>_t}\right]}{\gamma_{ref}(\theta_0, \phi_0)\left[\frac{<w_{g,1/4}(r_1')>_t}{<w_{g,1/4}(r_2')>_t}\left[\frac{\delta(r_2')}{r_2'^2}\right] - \frac{\delta(r_1')}{r_1'^2}\right]}, \tag{4.8}$$

where $<w_{g,1/4}(r_1')>_t$ and $<w_{g,1/4}(r_2')>_t$ are the time-averaged $\beta = 1/4$ GED measurements at positions r_1' and r_2', respectively, $\gamma_{ref}(\theta_0, \phi_0)$ is the directivity factor of the reference directivity source at the given angle, and $\delta(r_1')$ and $\delta(r_2')$ are the losses due to air absorption over the respective distances r_1' and r_2' (see Appendix C.1 for a full derivation of this equation). The air absorption $\delta(r)$, defined in ISO 3745[4], can be expressed in energetic form as:

$$\delta(r) = 10^{\frac{A_0(1.0053 - 0.0012A_0)^{1.6}}{10}}, \tag{4.9}$$

where A_0 is equal to the numerical value of $a(f)r$ and where $a(f)$ is the attenuation coefficient for the specific temperature, humidity, and static pressure as a function of frequency according to Eqs. (3) - (5) of ISO 9613.[30]

5. Turn off the reference directivity source and excite only the source under test.
6. Repeat step 3, measuring at an arbitrary angle and with the source under test as the source (see Fig. 4.2). Let r_1 and r_2 be the distances from the source under test to positions 1 and 2, respectively. The source under test must operate at the same amplitude for each measurement.
7. Calculate the unique directivity factor of the source under test $\gamma_s(\theta_0, \phi_0)$ for the given angle, using the following equation and the in situ room constant calculated in Eq. (4.8):

$$\gamma_s(\theta_0,\phi_0) = \frac{16\pi\left[1-\frac{<w_{g,1/4}(r_1)>_t}{<w_{g,1/4}(r_2)>_t}\right]}{R_{in\ situ}\left[\frac{<w_{g,1/4}(r_1)>_t}{<w_{g,1/4}(r_2)>_t}\left[\frac{\delta(r_2)}{r_2^2}\right]-\frac{\delta(r_1)}{r_1^2}\right]}\ . \tag{4.10}$$

(a)

(b)

Figure 4.2 Example configuration for step 6. The horn-loaded compression driver is the source under test. The energy density probe is located at position 1 (a) and position 2 (b). Both positions are along the same arbitrary angle relative to the source under test.

8. Calculate the sound power of the source under test according to Eq. (4.11), using the same GED field measurements from step 7, and the values for $R_{in\ situ}$ and $\gamma_s(\theta_0,\phi_0)$ determined from Eqs. (4.8) and (4.10), respectively:

$$<\Pi_s>_t = \frac{1}{2}\sum_{i=1}^{2}\frac{<w_{g,1/4}(r_i)>_t}{\frac{\gamma_s(\theta_0,\phi_0)\delta(r_i)}{4\pi r_i^2 c}+\frac{4}{R_{in\ situ}c}}\ , \tag{4.11}$$

where $\delta(r)$ is defined in Eq. (4.9).

In order to improve accuracy, it is recommended to measure and subtract background noise levels from each field measurement when implementing Eqs. (4.8), (4.10), and (4.11).

4.4.2 Experimental Verification Procedure

The results from Ch. 3 demonstrate how the GSPF is flexible enough to accurately determine the sound power of a wide range of acoustic sources in a variety of environments. Hence, the same sources, enclosures, data acquisition system, and energy density probes described in Secs. 3.4.1-3.4.3, were used to explore the performance of the two-point in situ method under similar conditions. Although the GED is the recommended energy field quantity in the two-point in situ method, the PED was also determined at each measurement point and substituted into Eqs. (4.8), (4.10), and (4.11) for comparison to the GED power estimates.

The 2,664-point measurement procedures described in Sec. 3.4.3.1 provided directivity factor data at 5° intervals in both θ and ϕ (refer to Figs. 3.8 and 3.16-3.18). This qualified each loudspeaker source as a potential reference directivity source for the two-point in situ method. Thus, for each configuration where a particular loudspeaker was the source under test, the experiment was repeated twice, where each additional source served as the reference directivity source.

Each of the sound power estimates were compared to the reference sound power curves acquired by following the ISO 3741 and 3745 standards (refer to Sec. 3.4.4). The same 1/3-octave band root-mean-square deviation (RMSD) analysis was also performed on each estimation [refer to Eq. (3.6)].

4.5 Results and Discussion

It is not feasible to include the sound power estimates of every source-enclosure combination in this thesis. However, the curves depicted in Figs. 4.3-4.6 collectively include all sources and test rooms, demonstrating the general results of the two-point in situ method. In each figure, the sound power estimates from the two-point in situ method are overlaid with that of the ISO 3741 and ISO 3745 methods for comparison. The room constant and directivity factor estimates, determined from steps 4 and 7 of the method, are included in Appendix C.2 for each configuration.

Similar to the plots in Ch. 3, the dashed vertical lines represent the lower and upper usable frequencies of the measurements. The RMSD values were calculated over the usable bandwidth, while estimated OALs were computed over the entire bandwidth of interest (100 Hz $< f <$ 10 kHz).

In general, the two-point in situ method generates sound power curves that follow the general shape of the reference curve quite well, but with some erratic behavior in the low and high frequency range. However, these discrepancies are minimized to values of 0.6 dB or less when the GED is the measured field quantity.

It is interesting to note that the in situ room constant and directivity factor estimates, calculated from steps 4 and 7, tend to be quite erratic when compared to expected values. However, the apparent errors tend to have a compensatory effect when inserted into the GSPF in step 8, generating relatively accurate sound power results (see Appendix C.2).

4.5.1 Specific Source-Enclosure Configurations

The sound power estimates of the dodecahedron source located in the reverberation chamber are shown in Fig. 4.3, where the single-driver dodecahedron was used as the reference directivity source. Other than a 4 dB discrepancy at approximately 300 Hz, the GED estimate matches the reference curve well. According to the reference, the OAL of the source, within the frequency range of interest, is 91.5 dB. The power estimates from the two-point in situ method for both energy density values fall within $\pm$ 0.5 dB of this value. See Fig. C.1 for additional insight.

Note that in the high-frequency range, both estimates more closely follow the reference curve generated from the method in the anechoic chamber (ISO 3745) even though the measurements were conducted inside the reverberation room. This same trend was witnessed with implementation of the GSPF in Ch. 3.

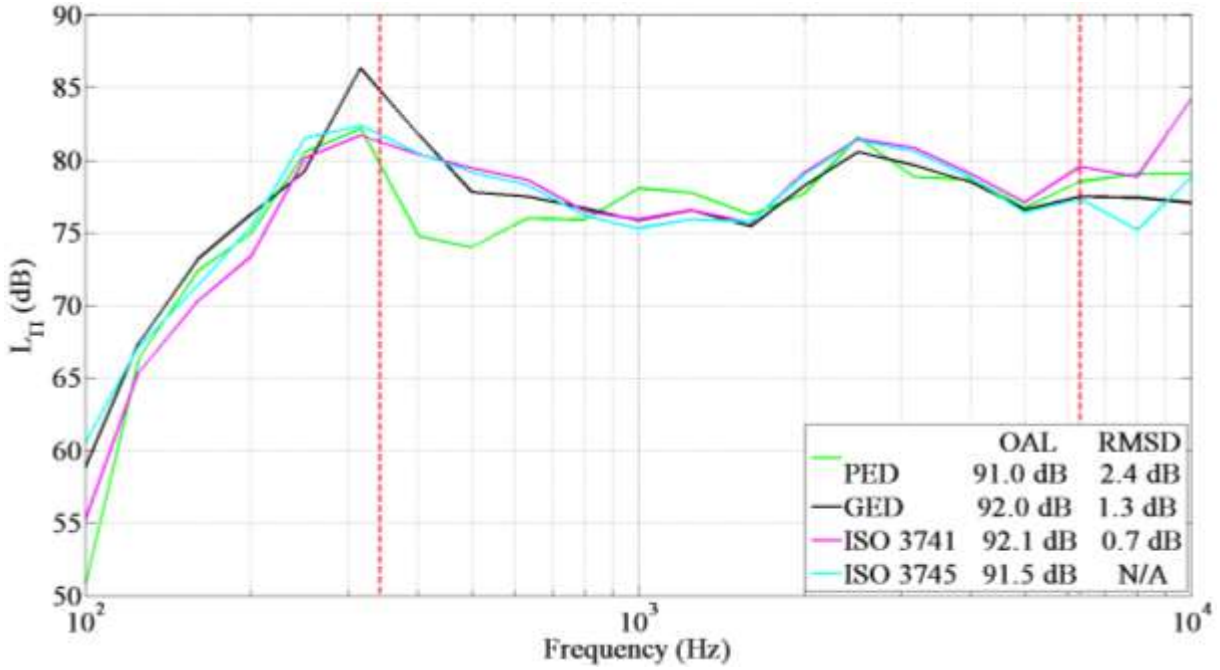

Figure 4.3 1/3-octave band sound power estimates of the dodecahedron source calculated from the two-point in situ method in the reverberation chamber. The single-driver dodecahedron served as the reference directivity source and the energy density fields were measured by the G.R.A.S. Type 50VI. The RMSD results are compared to the reference power curve measured by ISO 3745.

The sound power estimates of the horn-loaded compression driver inside the small room are presented in Fig. 4.4. In this configuration, the probes were positioned 45° off axis and the single-driver dodecahedron served as the reference directivity source. This is a scenario where the validity of the estimated room constant is dubious (refer to Fig. C.2); nevertheless, the GED matches the reference curve quite nicely over most of the bandwidth. The PED curve is quite erratic in the lower frequency range. Both curves overestimate the power above 7 kHz. This is most likely due to the diffuse-field assumption breaking down at low frequencies and the probe assumptions breaking down at high frequencies. The OAL of the GED curve is within 0.6 dB compared to the reference, despite the high-frequency these discrepancies.

In Fig. 4.5, the results are displayed for the single-driver dodecahedron in the medium-sized enclosure, where the probes were positioned on-axis and the dodecahedron served as the

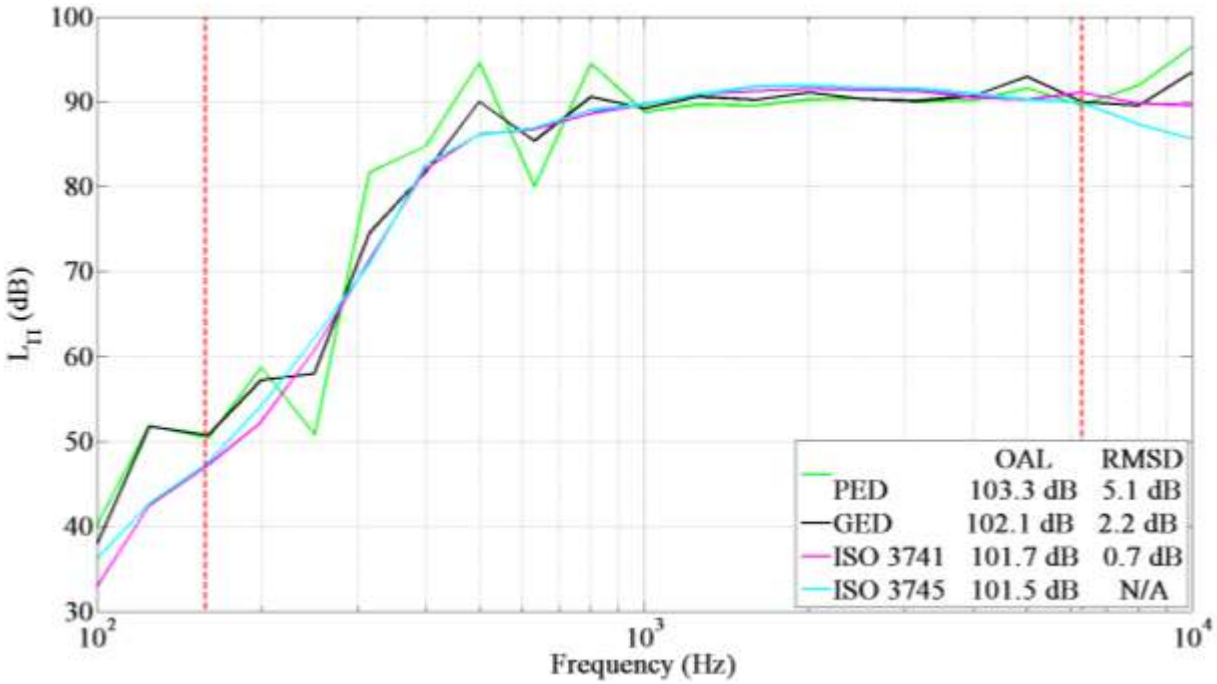

Figure 4.4 1/3-octave band sound power estimates of the horn-loaded compression driver calculated from the two-point in situ method, at 45° off axis, in the small room. The single-driver dodecahedron served as the reference directivity source and the energy density fields were measured by the G.R.A.S. Type 50VI. The RMSD results are compared to the reference power curve measured by ISO 3745.

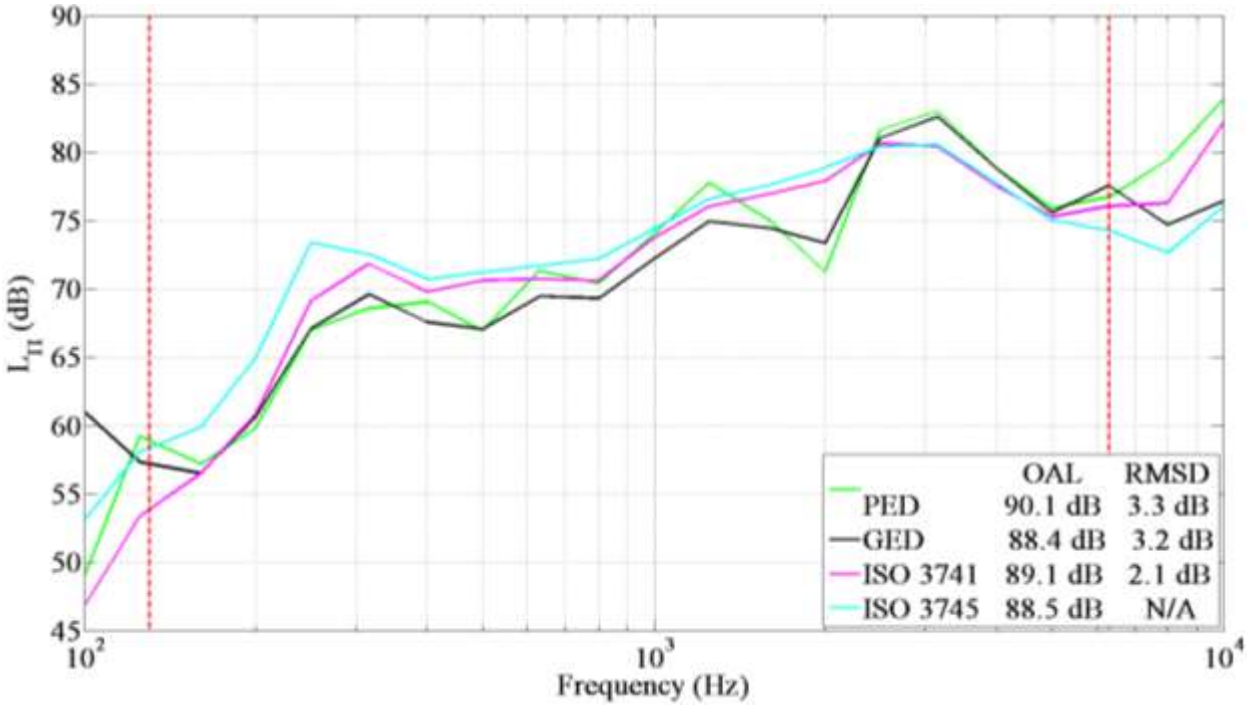

Figure 4.5 1/3-octave band sound power estimates of the single-driver dodecahedron source calculated from the two-point in situ method, on-axis, in the medium-sized room. The dodecahedron served as the reference directivity source and the energy density fields were measured by the Microflown USP. The RMSD results are compared to the reference power curve measured by ISO 3745.

reference directivity source. The OAL of the GED estimate is only 0.1 dB lower than the reference value of 88.5 dB, while that of the PED is 1.6 dB higher. It is interesting to observe that when the dodecahedron serves as the reference directivity source, both curves track the reference curve less accurately. See Fig. C.3 for additional insight.

The results of the horn-loaded compression driver in the large room are shown in Fig. 4.6, where the probes were positioned 45° off axis and the single-driver dodecahedron was the reference directivity source. Both of the energy density types overestimate the OAL, the PED by 2.1 dB and the GED by 0.9 dB. The GED performs better above 5 kHz. The results of Fig. C.4 demonstrate the same tendency for the two-point in situ method to compensate for seemingly erroneous estimates of the room constant.

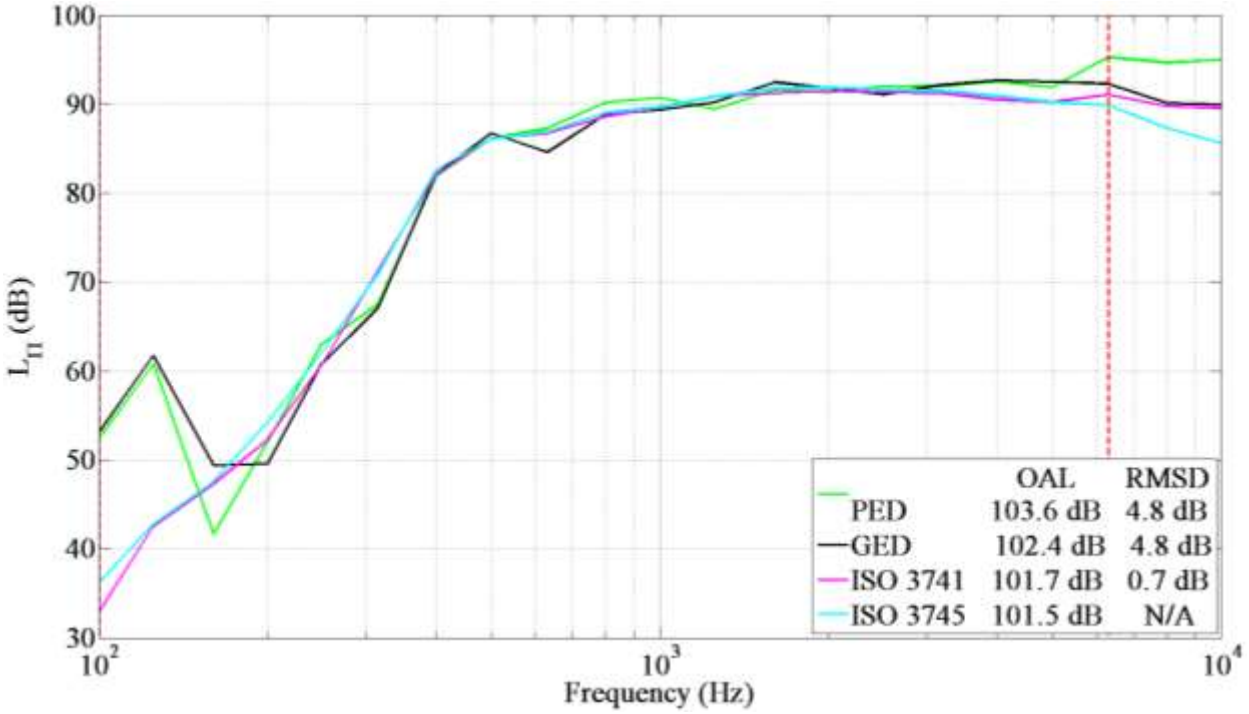

Figure 4.6 1/3-octave band sound power estimates of the horn-loaded compression driver calculated from the two-point in situ method, at 45° off axis, in the large room. The single-driver dodecahedron served as the reference directivity source and the energy density fields were measured by the G.R.A.S. Type 50VI. The RMSD results are compared to the reference power curve measured by ISO 3745.

The results from the RMSD analysis are summarized in Table 4.1. In general, the two-point in situ method estimates the sound power with less deviation from the reference power when the GED is the measured field quantity instead of the PED, with values as low as 1.3 dB in the reverberation chamber. According to the analysis, the two-point in situ method performs best in enclosures with low absorption, and decreases in accuracy with increasing absorption. However, the OAL sound power estimates, where GED was employed, were within 0.9 dB for all four enclosures.

4.5.2 Reference Directivity Sources

The implementation of a reference directivity source to calculate the sound power of the source under test can be more convenient than the use of a reference sound power source.[47]

Table 4.1 The RMSD of the sound power estimates determined from the PED, GED, and ISO 3741 standard. The two-point in situ method was implemented for the PED and GED values. The reference sound power curve was that obtained from the 2,664-measurement point ISO 3745 method.

	RMSD (dB)			
	Rev. Chamber Dodecahedron (G.R.A.S. Type 50VI)	**Small Room Horn** (G.R.A.S. Type 50VI)	**Medium Room Single Driver** (Microflown USP)	**Large Room Horn** (G.R.A.S. Type 50VI)
PED	2.4	5.1	3.3	4.8
GED	1.3	2.2	3.2	4.8
ISO 3741	0.7	0.7	2.1	0.7

However, there are a few details to consider. For example, how does the directivity pattern or maximum directivity factor of the reference directivity source affect the results of the two-point in situ method? The results from this study have given some insights into these questions.

Perhaps the most significant factor when considering a reference directivity source is how smoothly the directivity factors vary over frequency and angle. The purpose of the reference directivity source is to define a directivity factor for a given angle throughout the frequency band of interest and thus reduce the number of unknowns in the GSPF. In practice, the technician must place the reference directivity source in a room and position the probes at the designated angle or angles for which the directivity factor is known. It is reasonable to expect a few degrees of experimental error during this process. These errors are less likely to propagate if a well-behaving reference directivity source were in place, where the directivity factor does not vary much over angle and slowly evolves over frequency. An example of this is seen in the single-driver dodecahedron source, where a wide and smoothly shaped main lobe slowly forms with increasing frequency (refer to Fig. 3.17). Contrast this to the directivity factors of the dodecahedron and horn-loaded compression driver, which behave nicely at the low frequencies, but become either exceptionally complex or focused in the mid-to-high frequency range (refer to

Figs. 3.16 and 3.18). A small angular error in probe placement in these radiation fields is more likely to introduce error in the two-point in situ method.

To further illustrate this point, the change in the directivity factor for each reference directivity source was analyzed over 5° increments along the azimuthal angle relative to the principal axis (see Figs. 4.7-4.9). Each curve represents the amount of error that would be introduced if the probes were positioned 5° from the intended angle. The results of Fig. 4.7 demonstrate that, below 1 kHz, a 5° placement error at any angle between 0° and 90° does not translate into an appreciable directivity factor value error for the dodecahedron source. This is because the source radiates nearly omnidirectionally within that range. However, due to the complex radiation of the source above 1 kHz, this type of placement error, at any angle, would measure the energy field where the directivity factor differs by as much as 1.8 from the intended value. That corresponds to a percent error as high as 36 in γ.

The same trend is demonstrated with the horn-loaded compression driver (Fig 4.8), where the greatest potential for error is along or close to the principal axis where the directivity factor is high. For example, if the target angle were 10° off axis, but the probes were instead placed at 15°, in the higher frequency range, they would measure the energy field where the directivity factor differs by as much as 12 from the defined value. This type of error would surely propagate through the two-point in situ method and affect sound power estimates.

In comparison to the other reference directivity sources, the single-driver dodecahedron is more ideal (see Fig. 4.9). The difference between intended and actual directivity factors of the main lobe does not change by more than 0.5 until approximately 5 kHz for all angles of interest. There is a greater potential for error at higher frequencies near the principal axis, but it is less prominent than the other two sources. Hence, a source similar to the single-driver dodecahedron

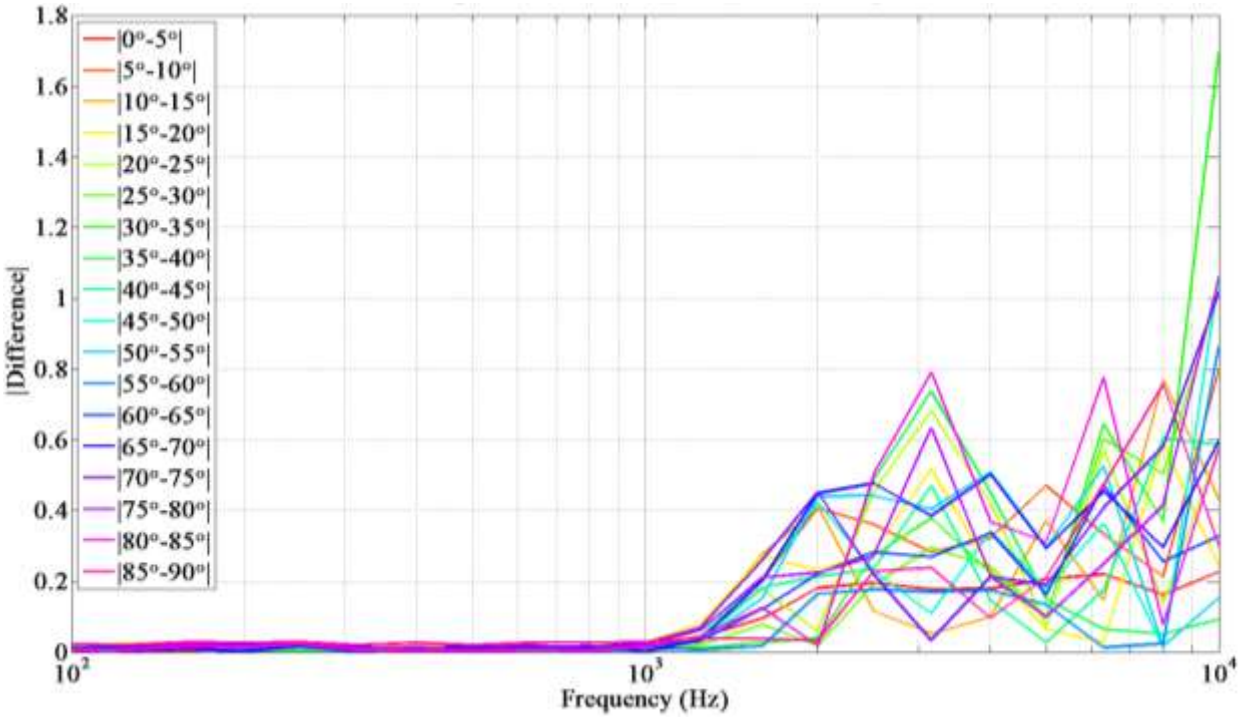

Figure 4.7 The absolute value of the difference between the directivity factors for each respective pair of angles [$|Intended - Actual|$], relative to an arbitrary chosen principal axis. These results are for the dodecahedron source.

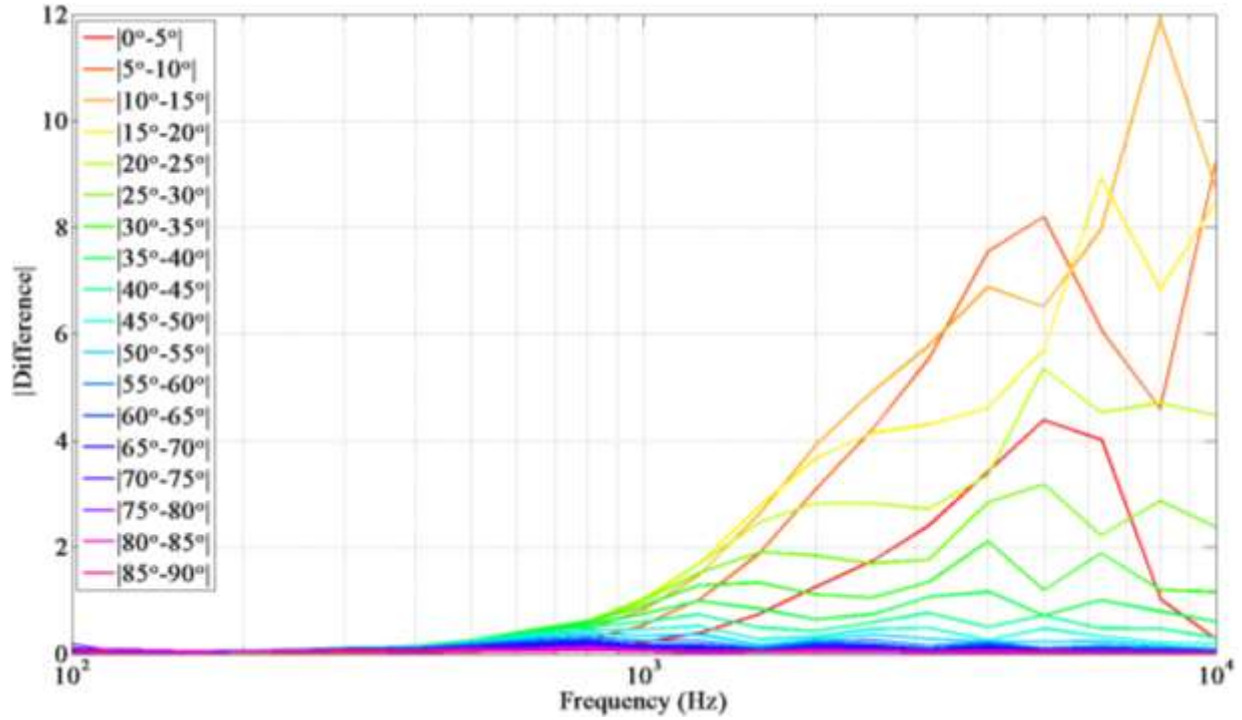

Figure 4.8 The absolute value of the difference between the directivity factors for each respective pair of angles [$|Intended - Actual|$], relative to the principal axis. These results are for the horn-loaded compression driver.

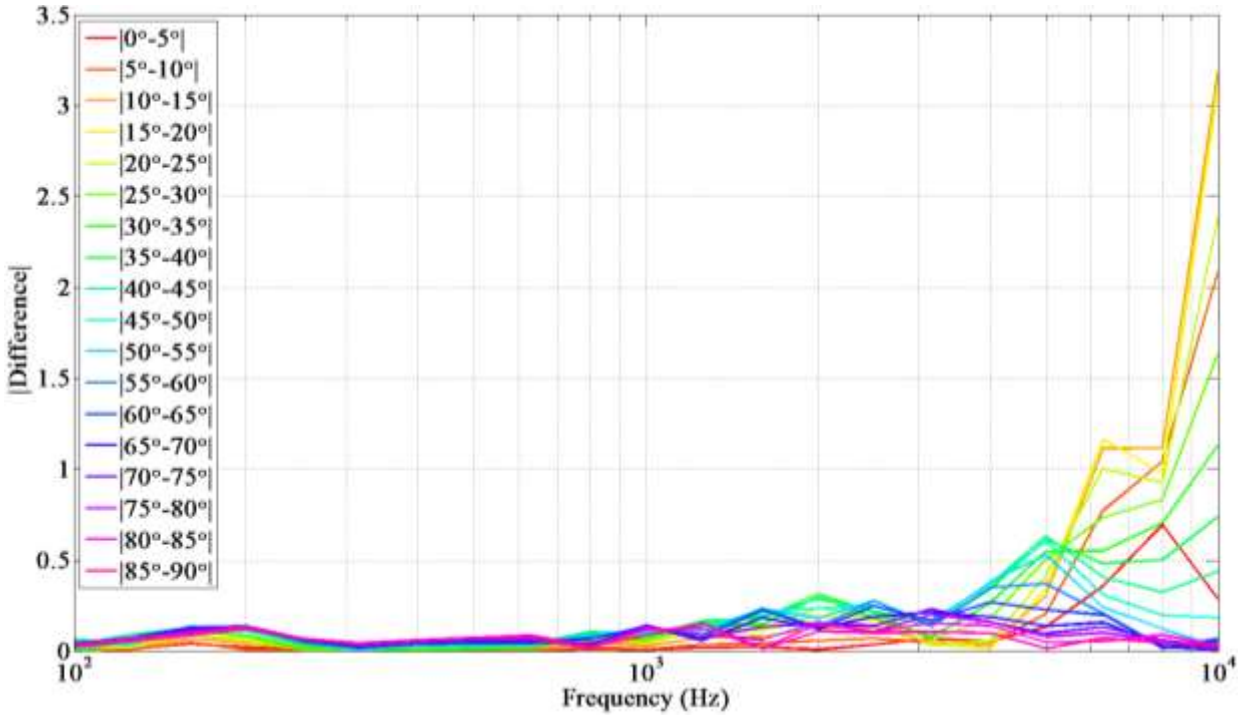

Figure 4.9 The absolute value of the difference between the directivity factors for each respective pair of angles [$|Intended - Actual|$], relative to the principal axis. These results are for the single-driver dodecahedron source.

is the recommended reference directivity source for use with the two-point in situ method because it is less prone to experimental error for angles between 0° and 90° relative to the principal axis.

Note that the technician can reduce the potential for error among highly-directional reference directivity sources by placing the energy density probes at greater off-axis angles. The plots in Fig. 4.10 demonstrate this effect efficiently. The difference in directivity factor of the horn-loaded compression driver between 85° and 90° is significantly less than between 0° and 5°. Of course, this may introduce inadequate signal-to-noise ratios.

4.5.3 Limitations of the Two-Point In Situ Method

The formulations represented in Eqs. (4.8) and (4.10) calculate the room constant and directivity factor of the source by simultaneously considering the direct and reverberant energies

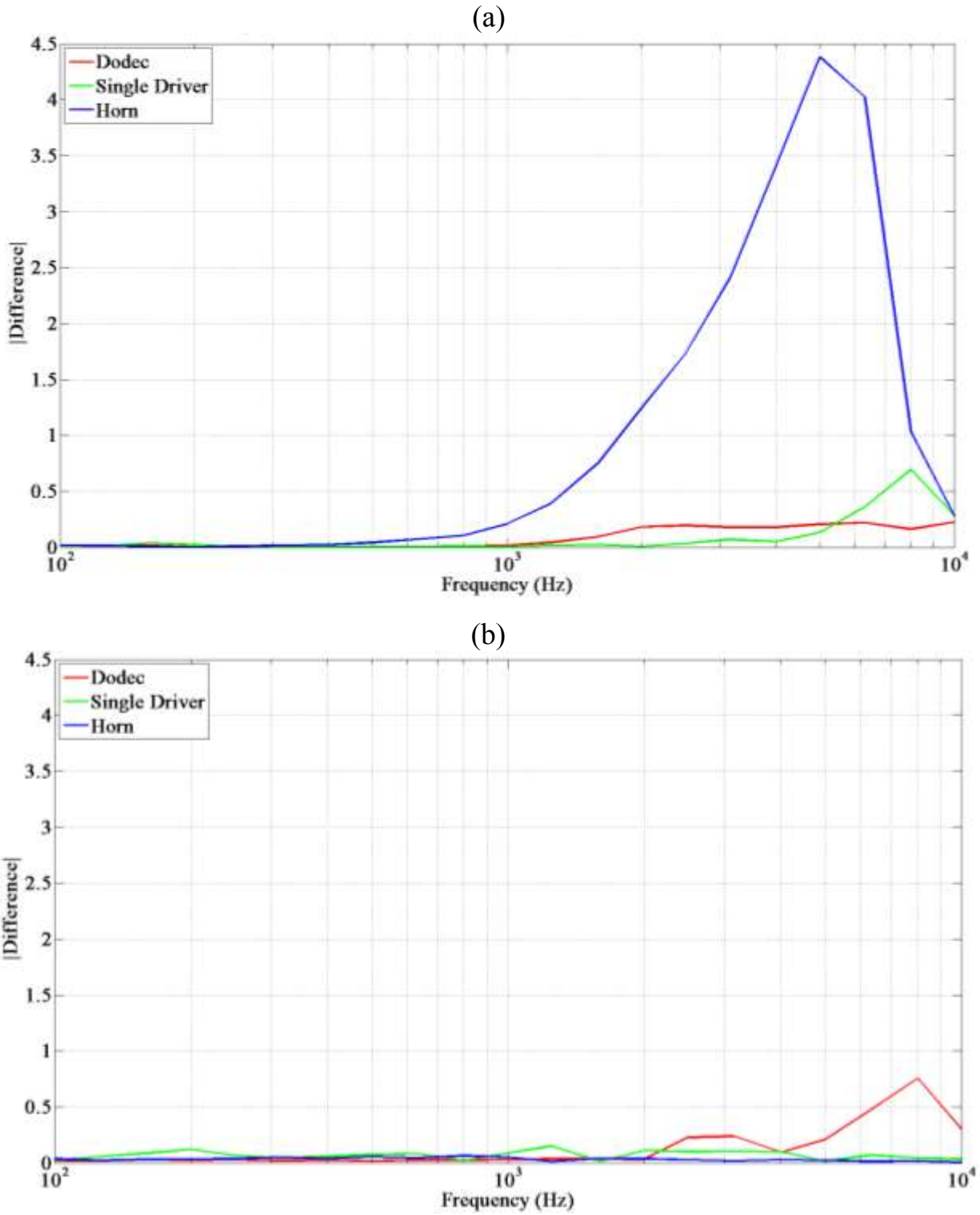

Figure 4.10 The absolute value of the difference between the directivity factors for the pair of angles (a) 0° and 5°, and (b) 85° and 90°, relative to the principal axis. The results for the three reference directivity sources are included for comparison.

within the enclosure. However, if both measurement positions where located in a region of the room where either energy component was negligible, the method would have trouble resolving both quantities. Hence, the two-point in situ method is most accurate when at least one measurement probe is positioned in a region where both energy components contribute significantly. In other words, one probe should be reasonably close to the critical distance r_c associated with the particular measurement angle, defined as the distance at which the average direct and reverberant field contributions are equal:

$$r_c(\theta, \phi) = \sqrt{\frac{\gamma(\theta_0, \phi_0)R}{16\pi}}. \tag{4.12}$$

In many situations, placing a measurement probe close to the estimated critical distance is not problematic. However, in particular configurations where the source under test is very directional and/or the average room absorption in the room is very high, the values of γ and/or R could be large enough to push the critical distance beyond the physical constraints of the enclosure, thus limiting the accuracy of the two-point in situ method. It may be possible to avoid this issue by choosing a measurement angle where the value of γ is assumed to be less. In general, the two-point in situ method produces more accurate sound power estimates for sources of moderate-to-low directivity placed in rooms of reasonably low absorption.

4.6 Specific Conclusions

The experimental exploration of the two-point in situ method in both diffuse and semi-reverberant fields has demonstrated its effectiveness. Unlike the procedures discussed in Ch. 3, the two-point in situ method has the ability to accurately estimate the sound power of a source without foreknowledge of its directivity factor and the absorption characteristics of the room. Instead, these quantities are obtained by in situ energy density measurements, which greatly

simplify the process. Because the room constant is measured directly, it is not necessary to determine the geometry and average absorption of the room. The specific conclusions associated with the method are as follows:

1. In general, the two-point in situ method generates sound power curves that consistently approach the reference power curves performed by the high-resolution ISO 3745 standard, producing OAL estimates within 0.9 dB. The erratic behaviors in the low and high-frequency range are minimized to acceptable values when the GED is the measured field quantity.
2. The in situ room constant and directivity factor estimates tend to be quite erratic when compared to expected values. However, these apparent errors tend to have a compensatory effect when inserted into the GSPF to estimate the power.
3. When compared to ISO 3745, the RMSD values of the estimated sound power are generally lower when the GED field is utilized in the two-point in situ method in place of the PED.
4. For the reference directivity source, it is better to use a well-behaving source, where the directivity factor does not vary much over angle and slowly evolves over frequency.
5. The two-point in situ method is most accurate when at least one measurement probe is positioned in a region where both the direct and reverberant energies contribute significantly.

Chapter 5

General Conclusions

This thesis has discussed the concept of how energy-based acoustics can be applied to improve and simplify sound power measurements. It has also presented several analytical, numerical, and experimental research efforts to explore these measurements in nonideal spaces using acoustic energy density.

5.1 Numerical Exploration

The numerical modeling and statistical exploration of the GED field in both diffuse and semi-reverberant fields has demonstrated its effectiveness in comparison to the PED. The fundamental characteristic that causes its improved performance is its greater spatial uniformity over an enclosed sound field. As a result, the number of GED sensors compared to microphones can be reduced by approximately one fourth in diffuse fields and by one half in semi-reverberant enclosures.

The GSPF was developed as a way of determining the sound power radiation of sources located in semi-reverberant enclosures. Through numerical simulations involving several rectangular enclosures of uniform and nonuniform absorption coefficients with values of up to

0.5, it was determined that use of the GED in the GSPF enables estimates of the sound power within 1 to 3 dB for the frequency range between 100 Hz and 3 kHz.

5.2 Experimental Verification of the GSPF

The experimental exploration of the GSPF in both diffuse and semi-reverberant fields has demonstrated its superiority over the original Hopkins-Stryker equation introduced in 1948. The fundamental characteristics that cause its improved performance are its use of the more spatially uniform GED field and implementation of an angularly varying and frequency-dependent directivity factor.

For three unique sound sources and four diverse environments, the GSPF consistently generated sound power estimates similar to those determined by the ISO 3745 standard, producing overall power level estimates within 1 dB or less. In addition, the ISO 3741 measurement constraints on the proximity to the source can be removed when using the GSPF within reverberation rooms because it accounts for the direct energy field component.

5.3 The Two-Point In Situ Method

The two-point in situ method was introduced as a way to overcome some of the practical challenges of implementing the GSPF for sound power determination in semi-reverberant enclosures. It is a quick and accurate method that does not require foreknowledge of the directivity factor and absorption characteristics of the room. Instead, these quantities are obtained by in situ energy density measurements and inserted into the GSPF.

In general, the two-point in situ method generates sound power curves that are consistently very close to the reference power curves determined by the ISO 3745 standard, producing OAL estimates within 0.6 dB. The method is most accurate when at least one

measurement probe is positioned in a region where both the direct and reverberant energies contribute significantly.

5.4 Significance of Results

Due to the rigorous and time-consuming nature of current sound power measurement methods, both academic and professional groups would benefit from techniques that effectively minimize time and cost. The two-point in situ method is a potential option for many because it eliminates the need of specialized acoustic environments, which many nationally accepted standards require. In addition, the implementation of generalized energy density allows one to determine sound power accurately and with relatively few measurements, due to its greater spatial uniformity when compared to that of the squared pressure field.

Because the two-point in situ method accounts for the direct energy component, it can overcome the measurement constraints involving proximity of sensors to the source when following the ISO 3741 standard in reverberation chambers. This allows for more convenient power estimates of highly directional sources.

5.5 Recommendations for Future Work

This study was limited to small and well behaving sources within rectangular enclosures. A thorough investigation of the GSPF and two-point in situ method using a more diverse set of sources and enclosures would be beneficial to further characterize their limitations. For example, how well does the two-point in situ method perform with very large complex sources of odd geometries? Is the method still valid in nonrectangular rooms or rooms coupled to adjacent spaces?

It would be advantageous to further explore how the measurement angles of each source under test and distances from the source to the energy density sensors affect the accuracy of the two-point in situ method. Are there general angles or distances that tend to optimize results? How vital is it for at least one sensor to be near the critical distance, and is there a way to overcome this constraint if necessary?

A thorough exploration of how various sources acting as the reference directivity source affect the potential for error when positioning the energy density probes for the two-point in situ method would be very useful. Is there a universal type of reference directivity source for most applications, or are some types better prescribed for specific configurations?

Continued exploration of correction terms to the GSPF and two-point in situ method power estimates would be beneficial. Does the in situ measurement of the room constant truly eliminate the need for corrections? Is the air absorption correction term applied in the most accurate way?

There is a possibility that the two-point in situ method could be simplified even more by reducing the number of required measurements positions from four to two. This could be done by positioning the source under test and reference directivity source along the same measurement axis, such that the two probe positions aligned to the reference directivity source would also be aligned to the source under test. It would be very beneficial to explore this configuration as a way to further simplify the two-point in situ method.

The scope of the proposed research was to investigate whether this approach is accurate and practical. It has demonstrated the potential to overcome many significant challenges that exist among current methods of determining sound power, but there are still more questions to explore. Further research into the questions proposed in this section could significantly improve

the two-point in situ method, which perhaps might reform the way scientists and engineers undertake sound power estimates in the future.

References

[1] ISO 3741:2010. "Acoustics – Determination of sound power levels and sound energy levels of noise sources using sound pressure – Precision methods for reverberation test rooms" (International Organization for Standardization, Geneva, 2010).

[2] ISO 3743-2:1994. "Acoustics – Determination of sound power levels of noise sources using sound pressure – Engineering methods for small, movable sources in reverberant fields – Part 2: Methods for special reverberation test rooms" (International Organization for Standardization, Geneva, 1994).

[3] ISO 3744:2010. "Acoustics – Determination of sound power levels and sound energy levels of noise sources using sound pressure – Engineering methods for an essentially free field over a reflecting plane" (International Organization for Standardization, Geneva, 2010).

[4] ISO 3745:2012. "Acoustics – Determination of sound power levels and sound energy levels of noise sources using sound pressure – Precision methods for anechoic rooms and hemi-anechoic rooms" (International Organization for Standardization, Geneva, 2012).

[5] H. F. Hopkins and N. R. Stryker, "A proposed loudness-efficiency rating for loudspeakers and the determination of system power requirements for enclosures," Proceedings of the I.R.E., Chicago, 315-335 (March 1948).

[6] J. Eargle, "Sound System Design Reference Manual," JBL Professional (Jan. 1999).

[7] ISO 3747:2010. "Acoustics –Determination of sound power levels and sound energy levels of noise sources using sound pressure – Engineering/survey methods for use *in situ* in a reverberant environment" (International Organization for Standardization, Geneva, 2010).

[8] E. C. Petersen, "An overview of standards for sound power determination," Application Note, Bruel & Kjaer Instruments, Inc.

[9] R. J. Wells, "Apparatus Noise Measurement," Proceedings of the AIEE, Chicago, 1170-1173 (Dec. 1955).

[10] A. D. Pierce, *Acoustics: An Introduction to its Physical Principles and Applications* (American Institute of Physics, Inc., New York, 1991), pp. 268.

[11] L. L. Beranek, *Acoustics* (American Institute of Physics, Inc., New York, 1996), pp. 109-112.

[12] L. L. Beranek, *Acoustics* (American Institute of Physics, Inc., New York, 1996), pp. 41, 298.

[13] R. J. Wells and F. M. Wiener, "On the determination of the acoustic power of a source of sound in semi-reverberant spaces," NOISE Control, 21-29 (Jan./Feb, 1961).

[14] J. Tichy and P. Baade, "Effect of rotating diffusors and sampling techniques on sound-pressure averaging in reverberation rooms," J. Acoust. Soc. Am. **56**, 137-143 (1974).

[15] R. K. Cook and P. A. Schade, "New method for measurement of the total energy density of sound waves," Proceedings of Inter-Noise 74, Washington DC, 1974, pp. 101-106.

[16] F. Jacobsen, "The diffuse sound field: Statistical considerations concerning the reverberant field in the steady state," The Acoustics Laboratory, Technical University of Denmark, Report No. 27 (1979).

[17] D. Nutter, T. Leishman, S. Sommerfeldt, and J. Blotter, "Measurement of sound power and absorption in reverberation chambers using energy density," J. Acoust. Soc. Am. **121**, 2700-2710 (2007).

[18] B. Xu, S. Sommerfeldt, and T. Leishman, "Generalized acoustic energy density," J. Acoust. Soc. Am. **130**, 1370-1380 (2011).

[19] B. Xu, S. Sommerfeldt, "A hybrid modal analysis for enclosed sound fields," J. Acoust. Soc. Am. **128**, 2857-2867 (2010).

[20] B. Xu, "Generalized Acoustic Energy Density and Its Applications," Ph. D dissertation, Brigham Young University, 2010

[21] http://en.wikipedia.org/wiki/Coefficient_of_variation

[22] http://en.wikipedia.org/wiki/Box_and_whisker_plot

[23] J. Thompson, L. Mitchell, and C. Hurst, "A modified room acoustics approach to determine sound pressure levels in irregularly proportioned factory spaces," Proceedings of Inter-Noise 76, Washington DC, 1976, pp. 465-468.

[24] M. S. Ureda, "Apparent apex, part III: the three-dimensional case," Presented at the 91st Convention of the Audio Engineering Society, October 1991, New York.

[25] M. S. Ureda, "Apparent apex," Presented at the 102nd Convention of the Audio Engineering Society, March 1997, Munich.

[26] M. S. Ureda, "The effects of horn acoustic center on array directivity," Presented at the 104th Convention of the Audio Engineering Society, May 1998, Amsterdam.

[27] M. S. Ureda, "On the movement of a horn's acoustic center," Presented at the 106th Convention of the Audio Engineering Society, May 1999, Munich.

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

[29] M. Hodgson, "Experimental evaluation of the accuracy of the Sabine and Eyring theories in the case of non-low surface absorption," J. Acoust. Soc. Am. **94**, 835-840 (1993).

[30] ISO 9613-1:1993. "Acoustics –Attenuation of sound during propagation outdoors – Part 1: Calculation of the absorption of sound by the atmosphere " (International Organization for Standardization, Geneva, 1993).

[31] ISO 354:2003. "Acoustics –Measurement of sound absorption in a reverberation room" (International Organization for Standardization, Geneva, 2003).

[32] M. Waser and M. Crocker, "Introduction to the two-microphone cross-spectral method of determining sound intensity," Noise Control Engineering Journal, **22**, 76-85 (1984).

[33] J. Tichy, "Acoustic Intensity Measurements – A Review," Presented at the 9th AIAA/NASA Aeroacoustics Conference, October 1984, Williamsburg.

[34] P. Bruel, "Characteristics of intensity measuring equipment," Noise Control Engineering Journal, **32**, 55-65 (1989).

[35] R. Waterhouse, “Interference patterns in reverberant sound fields,” J. Acoust. Soc. Am. **27**, 247-258 (1955).

[36] R. Waterhouse and R. Cook, “Interference patterns in reverberant sound fields. II,” J. Acoust. Soc. Am. **37**, 424-428 (1965).

[37] M. Vorlander, “Revised relation between the sound power and the average sound pressure level in rooms and the consequences for acoustic measurements,” Acoustica, **81**, 332-343 (1995).

[38] R. Neubauer, “Estimation of reverberation time in rectangular rooms with non-uniformly distributed absorption using a modified Fitzroy equation,” Acoustica, **8**, 115-137 (2001).

[39] D. Fitzroy, “Reverberation formulae which seems to be more accurate with non-uniform distribution of absorption,” J. Acoust. Soc. Am. **31**, 893-897 (1959).

[40] L. L. Beranek, *Acoustics* (American Institute of Physics, Inc., New York, 1996), pp. 312.

[41] R. Young, “Sabine reverberation equation and sound power calculations,” J. Acoust. Soc. Am. **31**, 912-921 (1959).

[42] F. Agerkvist and F. Jacobsen, “Sound power determination in reverberation rooms at low frequencies,” Journal of Sound and Vibration, **166**, 179-190 (1993).

[43] A. Schaffner, “Accurate estimation of the mean sound pressure level in enclosures,” J. Acoust. Soc. Am. **106**, 823-827 (1999).

[44] T. Jo and M Koyasu, “Measurement of reverberation time based on the direct-reverberant sound energy ratio in steady state,” Proceedings of Inter-Noise 75, Sendai, 1975.

[45] J. B. Moreland, “Measurement of sound absorption in rooms,” J. Acoust. Soc. Am. **61**, 476-483 (1977).

[46] C. Ianniello, “Walk away method versus reverberation time method in determining the room constant,” Applied Acoustics, **14**, 83-92 (1980).

[47] P. Terrell, W. Hanson, and M. Ramsey, “Predicting noise reduction from absorptive treatments in industrial spaces: Alternatives to the ‘Sabine’ method,” Am. Ind. Hyg. Assoc. J. **44**, 809-813 (1983).

Appendix A

Additional Figures for Chapter 2

This section includes additional figures related to the numerical results included in Ch. 2, that otherwise would lengthen the presentation of the data if included in the main body of the text.

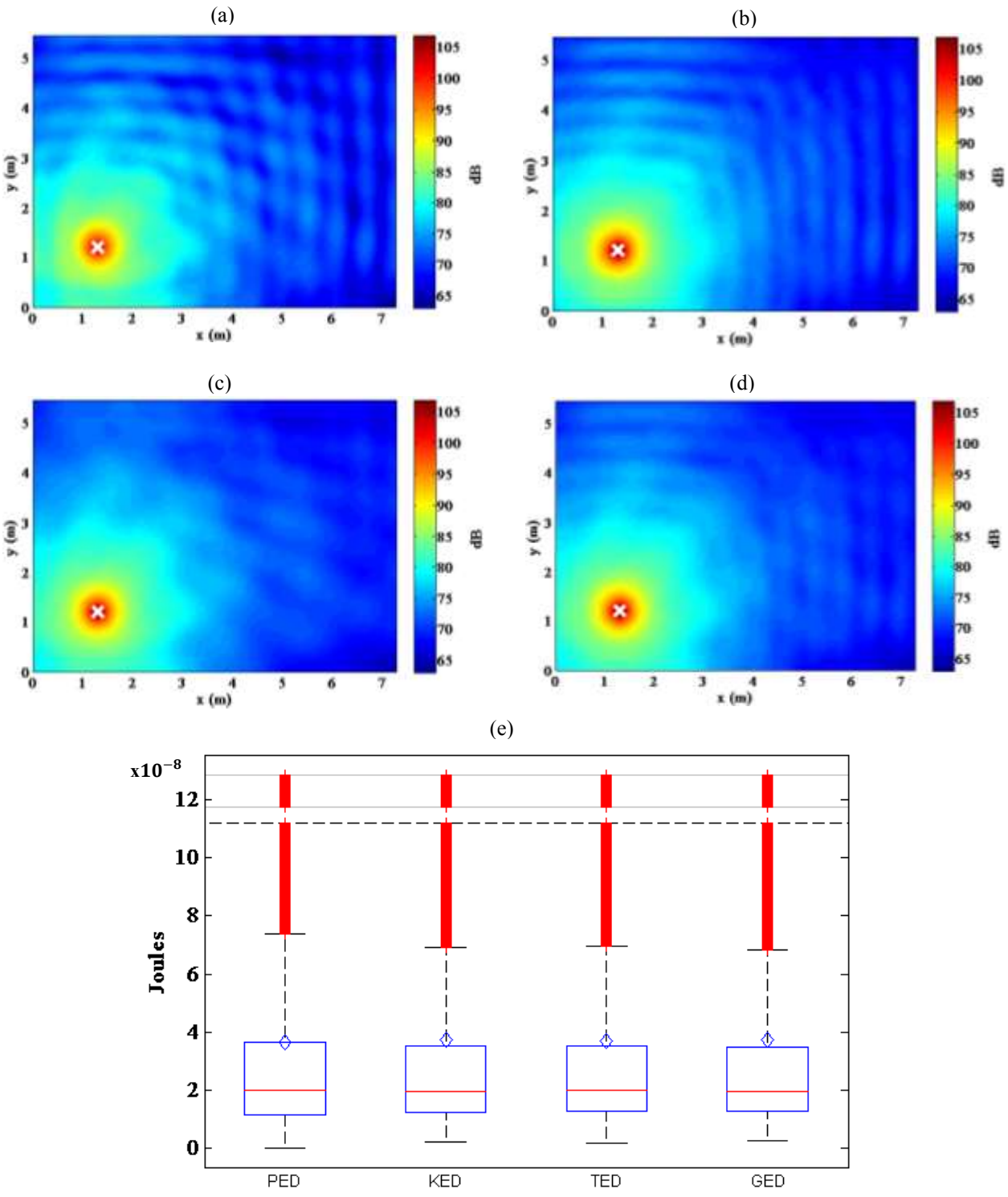

Figure A.1 Plan views of the energy density fields in a room with a uniform absorption coefficient of 0.95 and dimensions 5 x 6 x 7 m. The source is located at the "x" symbol (x = 1.3 m, y = 1.2 m, z = 4.7 m) and radiates at 300 Hz. The horizontal plane intersects the source. a) PED, $C_v = 1.817$. b) KED, $C_v = 1.943$. c) TED, $C_v = 1.869$. d) GED, $C_v = 1.903$. e) Box-and-whisker plots of the four energy density fields. The outliers have been compressed above the dashed horizontal line.

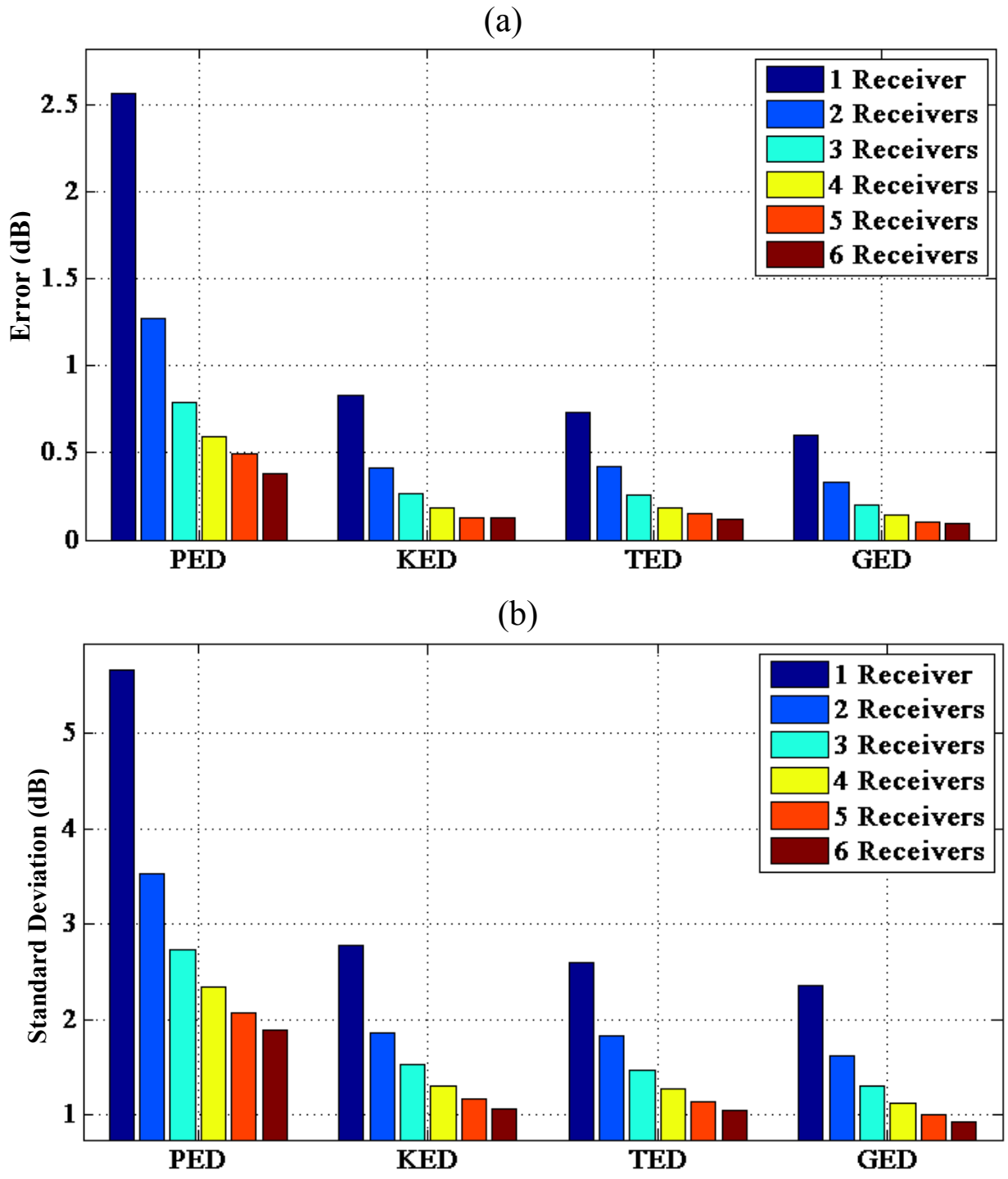

Figure A.2 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.01. Each simulation was repeated and averaged 5,000 times at a frequency of 2 kHz. a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined from all field points. b) Standard deviation of the estimated mean energy densities made by the various sensors.

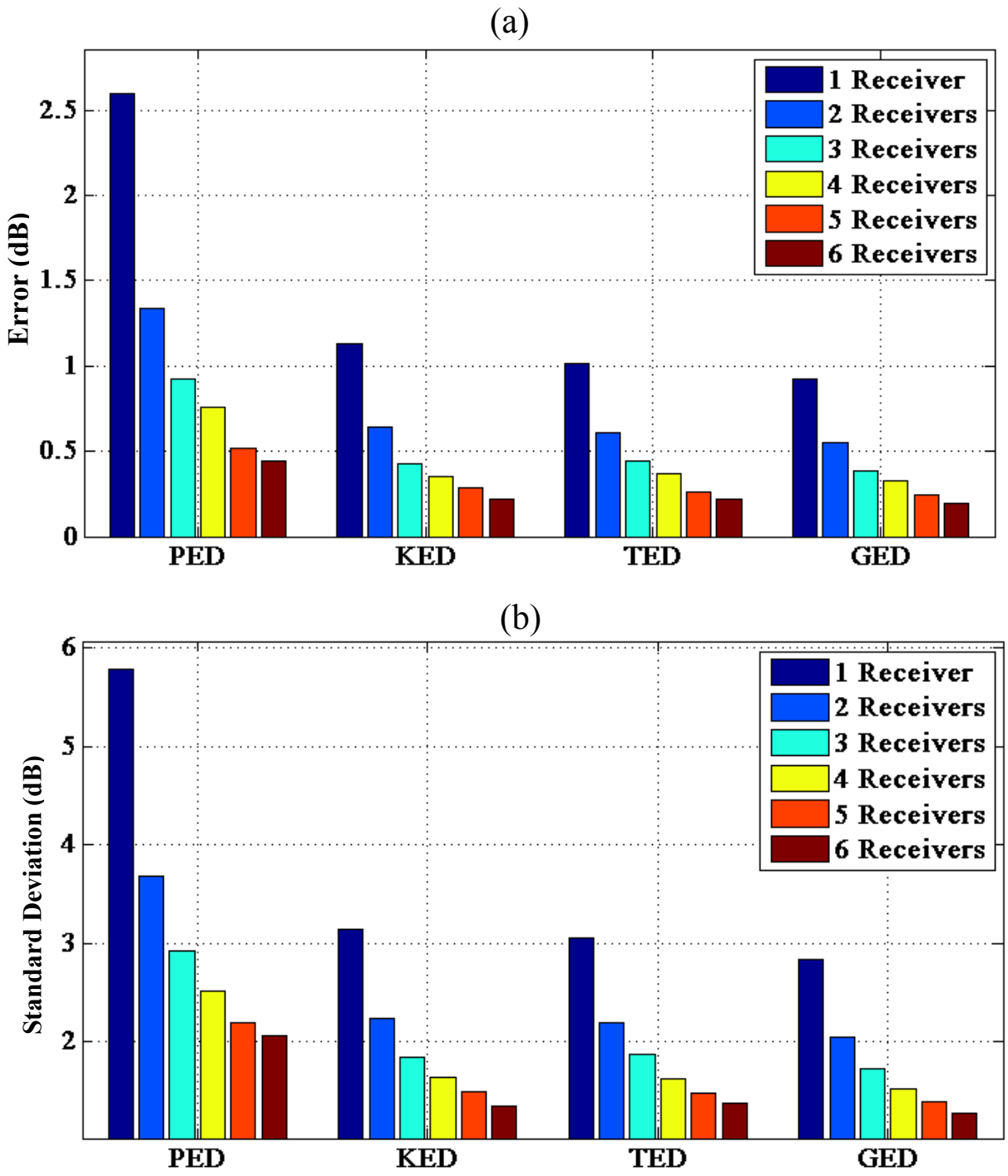

Figure A.3 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.33. Each simulation was repeated and averaged 5,000 times at a frequency of 1 kHz. a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined from all field points. b) Standard deviation of the estimated mean energy densities made by the various sensors.

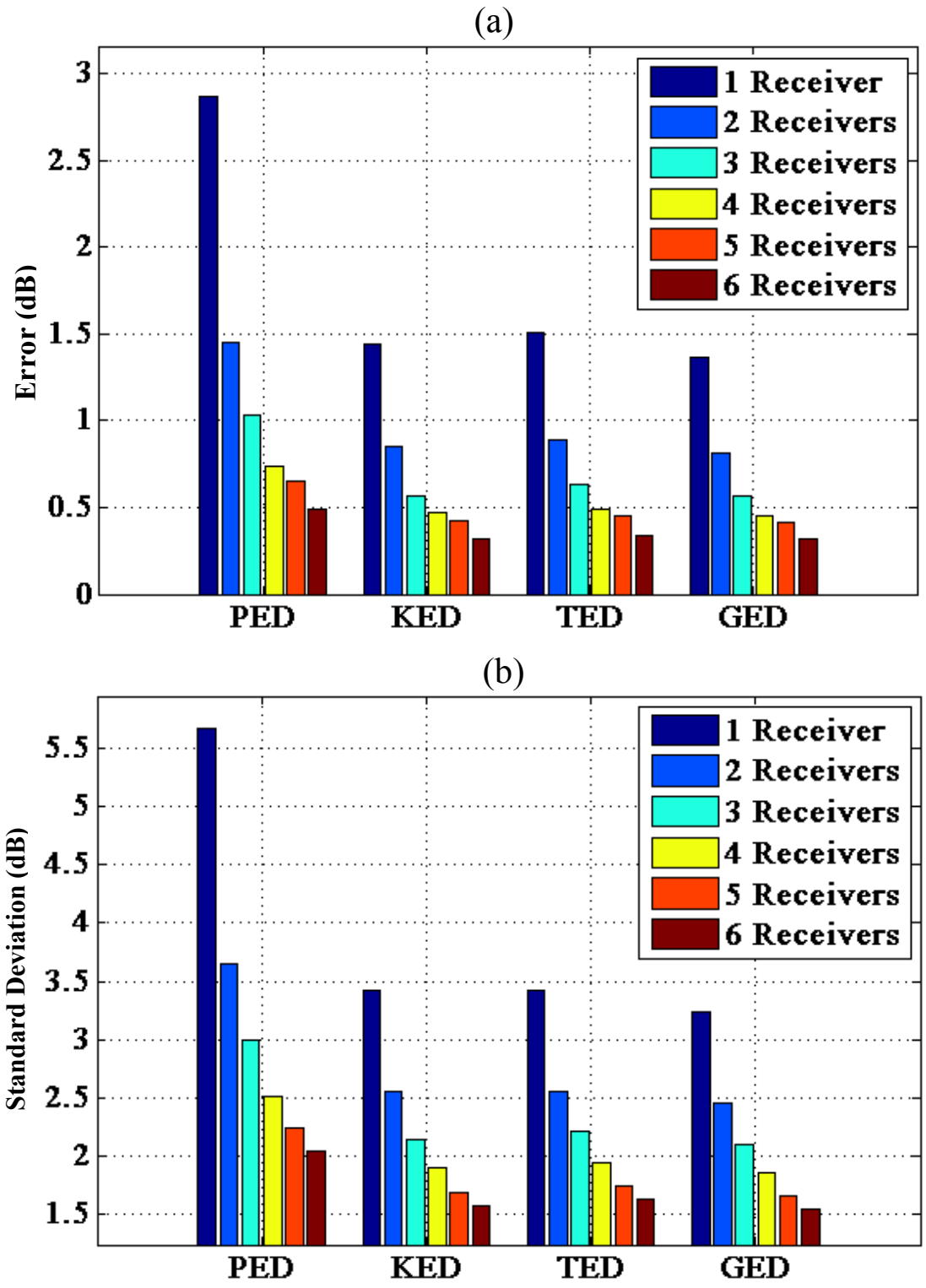

Figure A.4 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7 m and an absorption coefficient of 0.5. Each simulation was repeated and averaged 5,000 times at a frequency of 400 Hz. a) Error of the estimated mean energy density by the various sensors compared to the actual mean energy density as determined from all field points. b) Standard deviation of the estimated mean energy densities made by the various sensors.

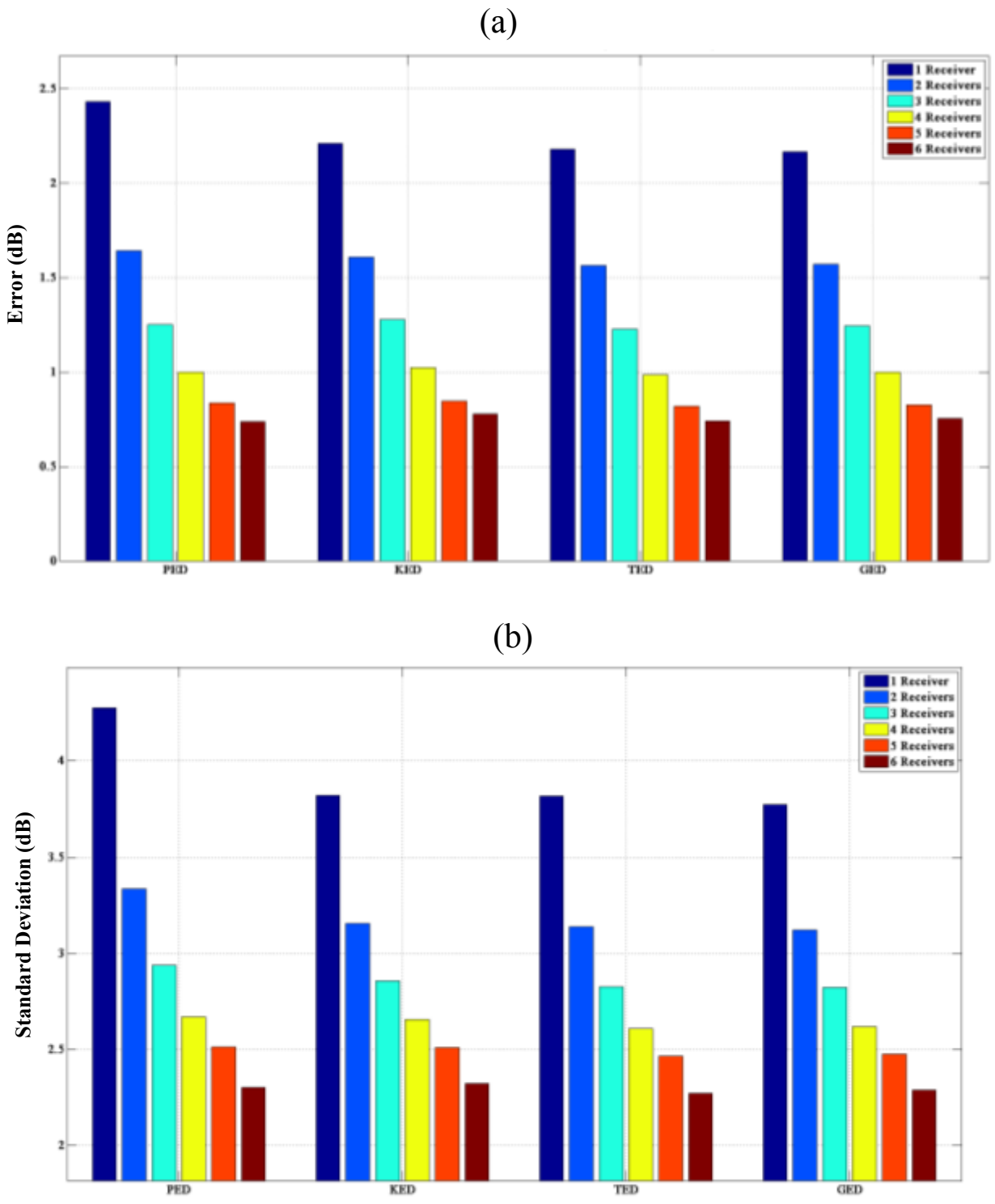

Figure A.5 Statistical results of randomly placing one through six sensors within four energy density fields generated in a room of dimensions 5 x 6 x 7m and an absorption coefficient of 0.95. Each simulation was repeated and averaged 5,000 times at a frequency of 300 Hz. a) Error of the estimated mean energy density by various sensors compared to the actual mean energy density as determined from all field points.
b) Standard deviation of the estimated mean energy densities made by the various sensors.

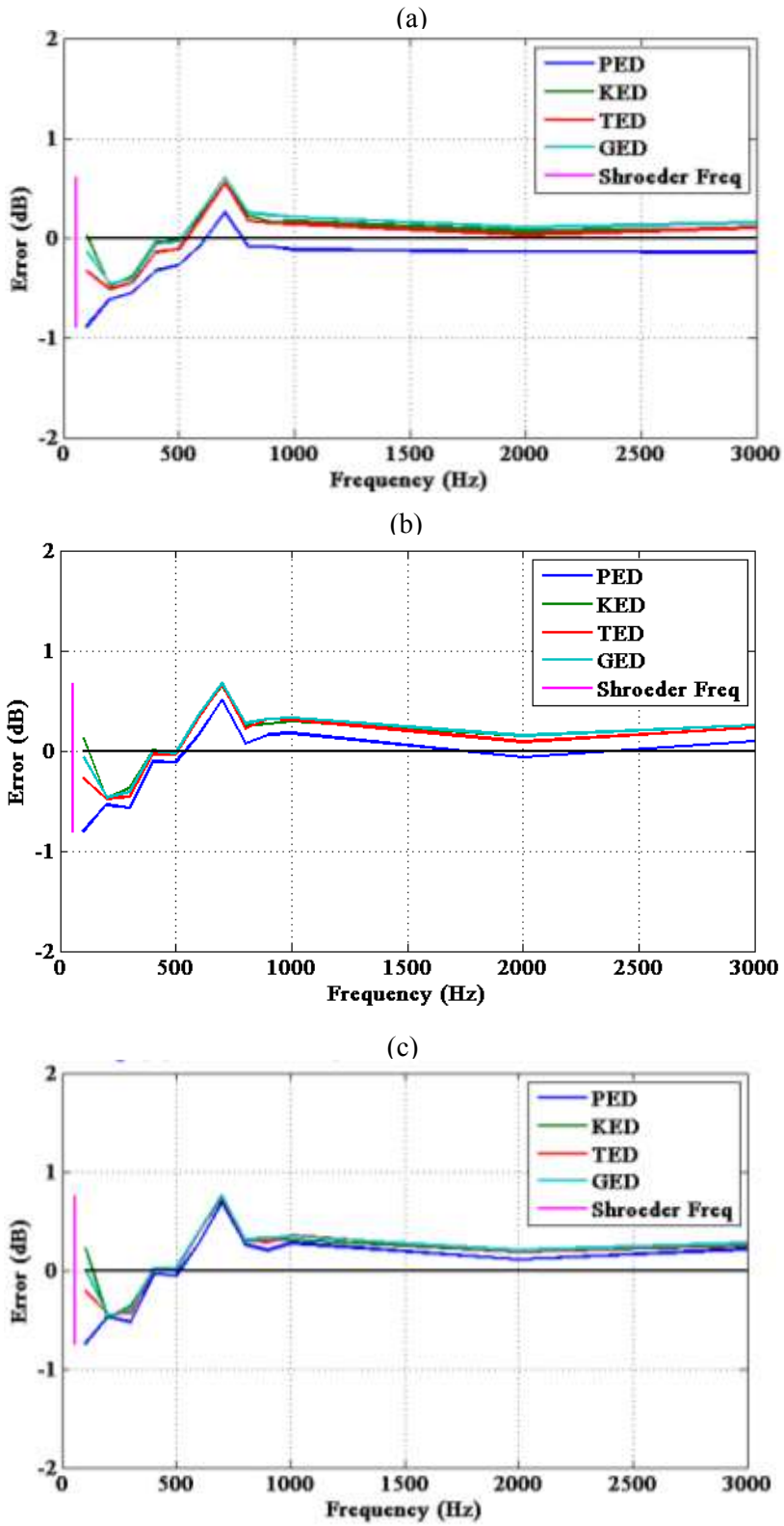

Figure A.6 The mean error in sound power estimation over 5,000 simulations of the GSPF, where in each case, a collection of randomly placed sensors is used within four energy density fields. The room has dimensions 7.3 x 5.4 x 4.5 m and an absorption coefficient of 0.82. a) 1 sensor. b) 2 sensors. c) 4 sensors.

Appendix B

Derivation for the Experimental Determination of the Directivity Factor

The time-averaged radial intensity at any point on a far-field measurement surface is

$$<I_{m,n}(f)>_t = \frac{\left|p_{m,n}(f)\right|^2}{2\rho_0 c} = \frac{|e(f)|^2\left|H_{m,n}(f)\right|^2}{2\rho_0 c}, \tag{B.1}$$

where the subscripts m and n represent discrete polar and azimuthal angles on the surface, $e(f)$ is the frequency-dependent voltage of the excitation signal, and $H_{m,n}(f)$ is the ratio

$$H_{m,n}(f) = \frac{p_{m,n}(f)}{e(f)}. \tag{B.2}$$

The total sound power is

$$<\Pi(f)>_t \approx \frac{|e(f)|^2}{2\rho_0 c}\sum_m\sum_n S_{m,n}\left|H_{m,n}(f)\right|^2, \tag{B.3}$$

where $S_{m,n}$ is an area weighting factor explained by Leishman *et al.*[28]

The intensity radiated by an omnidirectional source with this same power is

$$<I_{omni,m,n}(f)>_t = \frac{<\Pi(f)>_t}{4\pi r^2} \approx \frac{1}{4\pi r^2}\left(\frac{|e(f)|^2}{2\rho_0 c}\sum_m\sum_n S_{m,n}\left|H_{m,n}(f)\right|^2\right). \tag{B.4}$$

From Eq. (1.3), the directivity factor for any measurement point is then

$$\gamma_{m,n}(f) = \frac{<I_{m,n}(f)>_t}{<I_{omni,m,n}(f)>_t} = \frac{4\pi r^2 \left|H_{m,n}(f)\right|^2}{\sum_m \sum_n S_{m,n} \left|H_{m,n}(f)\right|^2} \ . \tag{B.5}$$

Appendix C

Derivations and Additional Figures for Chapter 4

C.1 Derivation of the R and γ Terms in the Two-Point In Situ Method

The governing formulation of the two-point in situ method is the generalized sound power formulation (GSPF), including air absorption. It is expressed as

$$<\Pi_s>_t = \frac{1}{I}\sum_{i=1}^{I} \frac{<w_{g,1/4}(r_i)>_t}{\frac{\gamma_i(\theta_0,\phi_0)\delta(r_i)}{4\pi r_i^2 c} + \frac{4}{Rc}} \ . \tag{C.1}$$

Air absorption $\delta(r)$, defined in ISO 3745,[4] is included in the direct energy term. However, since the two-point in situ method determines the room constant R by in situ measurements, the analytical reverberant field corrections discussed in Sec. 4.1.2 are not needed.

For a measurement at a single point, the $\beta = 1/4$ GED is

$$<w_{g,1/4}(r)>_t = \frac{<\Pi_s>_t}{c}\left[\frac{\gamma(\theta_0,\phi_0)\delta(r)}{4\pi r^2} + \frac{4}{R}\right]. \tag{C.2}$$

If two measurements were taken along the same angle relative to the source, the two formulations would be

$$<w_{g,1/4}(r_1')>_t = \frac{<\Pi_s>_t}{c}\left[\frac{\gamma_{ref}(\theta_0,\phi_0)\delta(r_1')}{4\pi r_1'^2}+\frac{4}{R}\right], \tag{C.3a}$$

$$<w_{g,1/4}(r_2')>_t = \frac{<\Pi_s>_t}{c}\left[\frac{\gamma_{ref}(\theta_0,\phi_0)\delta(r_2')}{4\pi r_2'^2}+\frac{4}{R}\right], \tag{C.3b}$$

where r_1' and r_2' are field positions 1 and 2, respectively (the primes indicating field positions associated with the reference directivity source measurements), and $\gamma_{ref}(\theta_0,\phi_0)$ is the known directivity factor of the reference source along the specified angle.

Rearranging Eq. (C.3b) to solve for power in terms of GED and then substituting it into Eq. (C.3a) yields:

$$<w_{g,1/4}(r_1')>_t = \frac{<w_{g,1/4}(r_2')>_t}{\left[\frac{\gamma_{ref}(\theta_0,\phi_0)\delta(r_2')}{4\pi r_2'^2}+\frac{4}{R}\right]}\left[\frac{\gamma_{ref}(\theta_0,\phi_0)\delta(r_1')}{4\pi r_1'^2}+\frac{4}{R}\right]. \tag{C.4}$$

In this form, it is possible to solve for R in terms of the GED quantities and $\gamma_{ref}(\theta_0,\phi_0)$:

$$R_{in\ situ} = \frac{16\pi\left[1-\frac{<w_{g,1/4}(r_1')>_t}{<w_{g,1/4}(r_2')>_t}\right]}{\gamma_{ref}(\theta_0,\phi_0)\left[\frac{<w_{g,1/4}(r_1')>_t}{<w_{g,1/4}(r_2')>_t}\left[\frac{\delta(r_2')}{r_2'^2}\right]-\frac{\delta(r_1')}{r_1'^2}\right]}. \tag{C.5}$$

One can also solve for the directivity factor of the source under test $\gamma_s(\theta_0,\phi_0)$ in terms of the GED quantities and $R_{in\ situ}$:

$$\gamma_s(\theta_0,\phi_0) = \frac{16\pi\left[1-\frac{<w_{g,\frac{1}{4}}(r_1)>_t}{<w_{g,\frac{1}{4}}(r_2)>_t}\right]}{R_{in\ situ}\left[\frac{<w_{g,\frac{1}{4}}(r_1)>_t}{<w_{g,\frac{1}{4}}(r_2)>_t}\left[\frac{\delta(r_2)}{r_2^2}\right]-\frac{\delta(r_1)}{r_1^2}\right]}, \tag{C.6}$$

where r_1 and r_2 represent the distances from the source under test to field positions 1 and 2, respectively.

C.2 Additional Figures

This section includes additional figures related to the two point in situ method, detailed in Ch. 4, that otherwise would lengthen the presentation of the data if included in the main body of the text. Each figure includes estimates of the room constant R and directivity factor γ determined by in situ measurements. Note that the T_{60} results were based on room volumes and surface areas that neglected the presence of stationary diffusers located in the reverberation chamber and irregularities in the other test rooms. From this standpoint, they involved some errors as well.

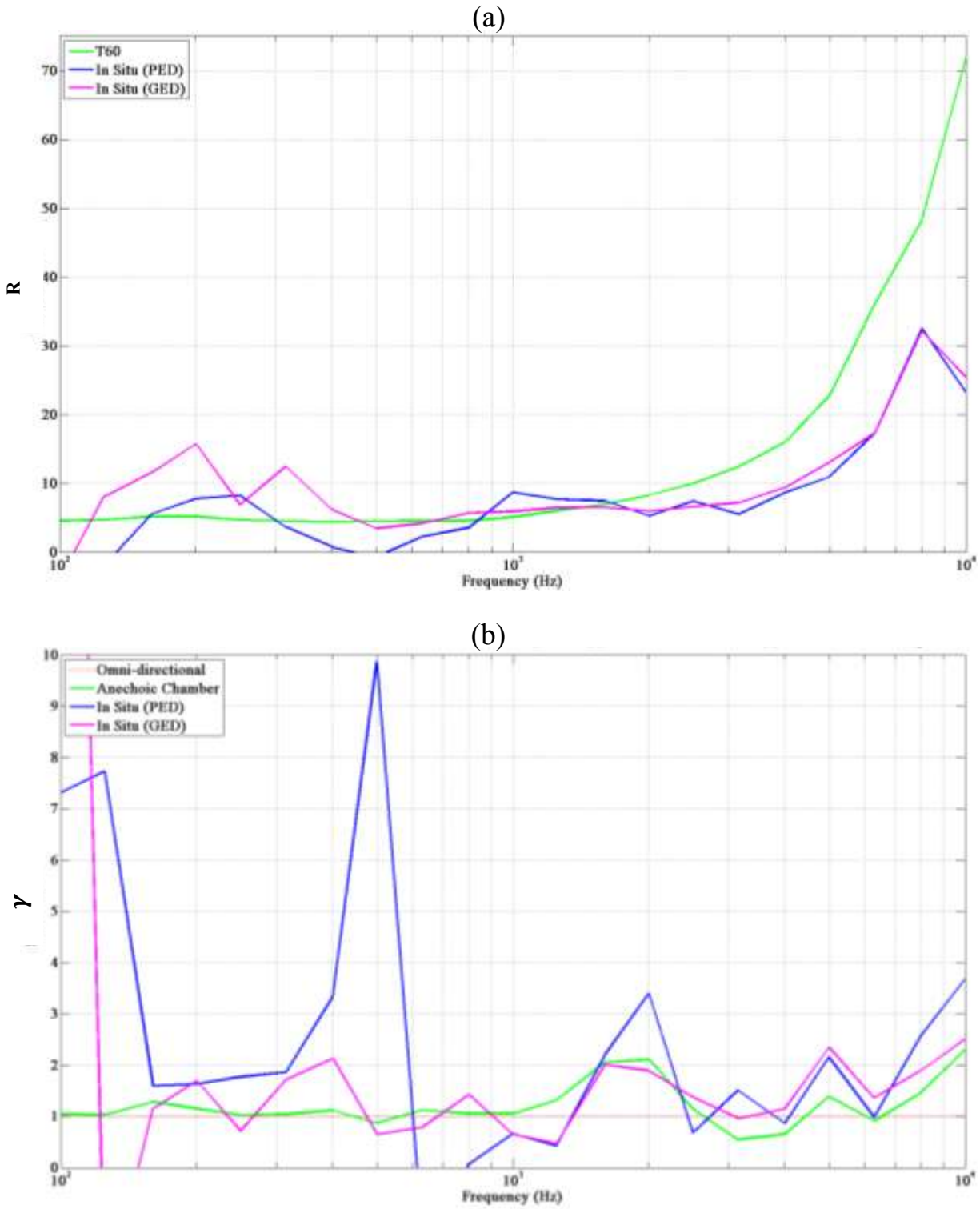

Figure C.1 Results from the two-point in situ method using the G.R.A.S. Type 50VI and the single-driver dodecahedron as the reference directivity source. a) Estimates of the room constant R for the reverberation chamber. The green curve is the estimate by the traditional reverberation time method. b) Estimates of the on axis directivity factor γ for the dodecahedron source. The green curve is the estimate from the 2,664-point method in the anechoic chamber. The dotted red line serves as a visual reference for an omnidirectional source.

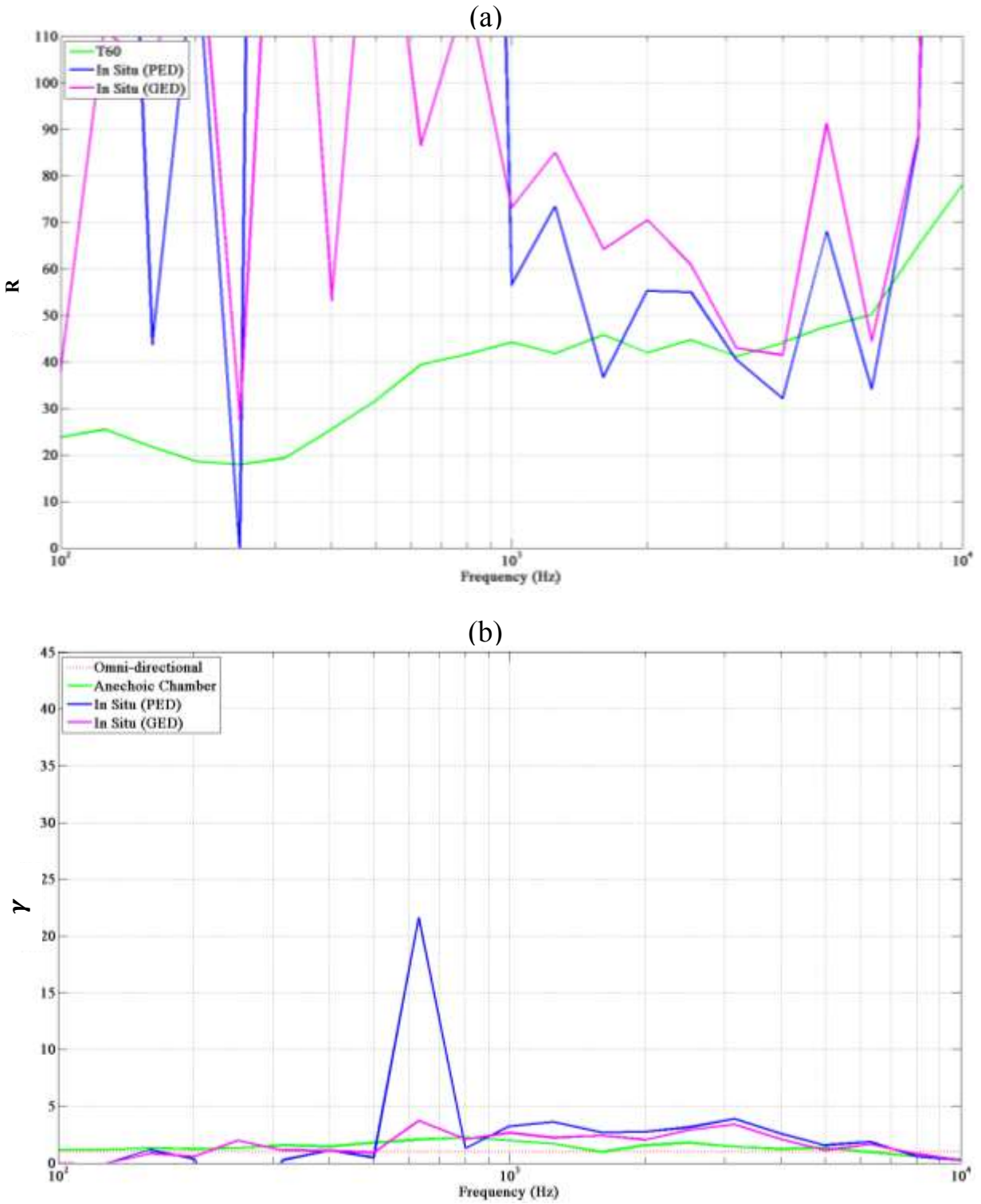

Figure C. 2 Results from the two-point in situ method using the G.R.A.S. Type 50VI and the single-driver dodecahedron as the reference directivity source. a) Estimates of the room constant R for the small room. The green curve is the estimate by the traditional reverberation time method. b) Estimates of the directivity factor γ for the horn-loaded compression driver at 45° off axis. The green curve is the estimate from the 2,664-point method in the anechoic chamber. The dotted red line serves as a visual reference for an omnidirectional source.

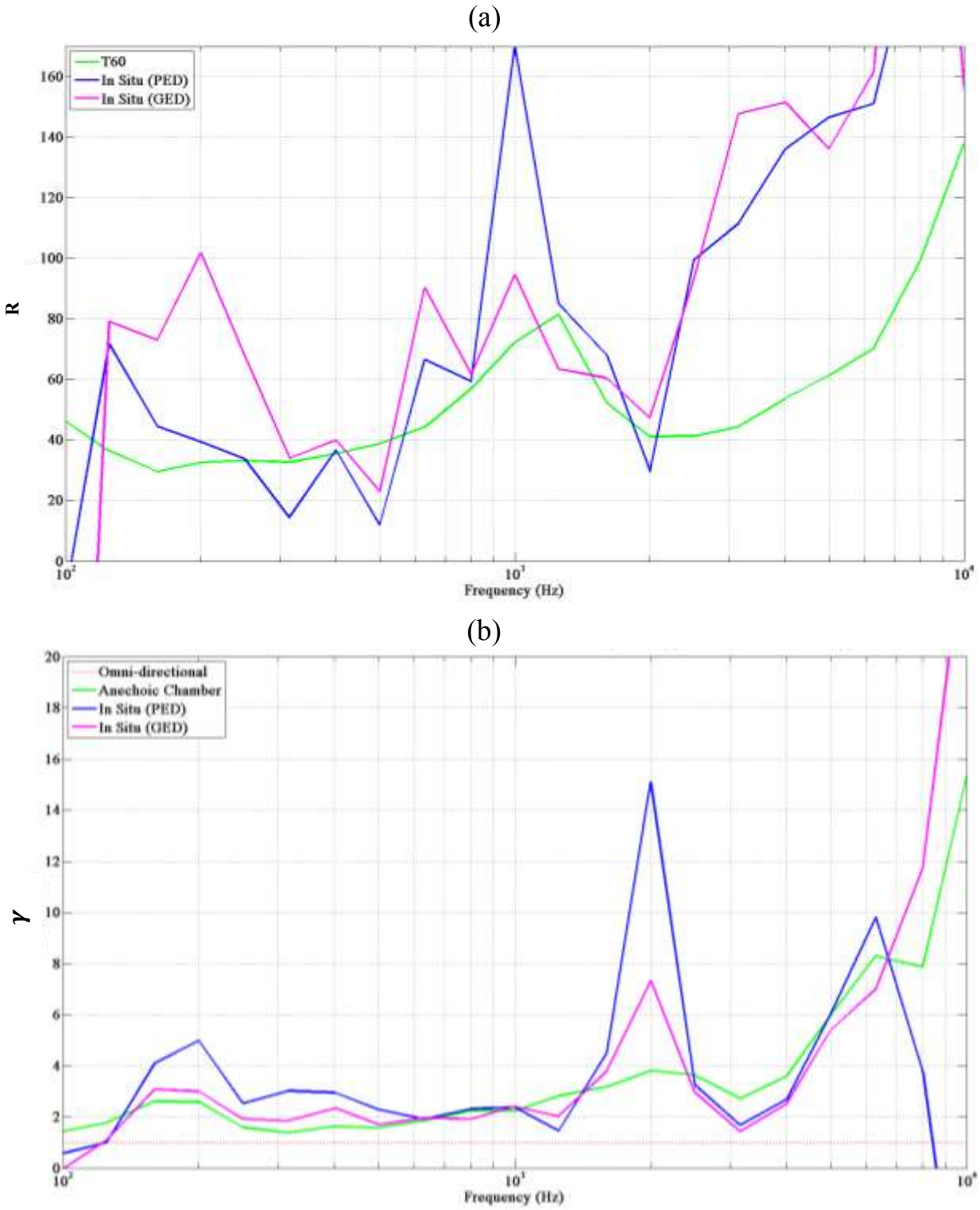

Figure C. 3 Results from the two-point in situ method using the Microflown USP and the dodecahedron as the reference directivity source. a) Estimates of the room constant R for the medium-sized room. The green curve is the estimate by the traditional reverberation time method. b) Estimates of the on axis directivity factor γ for the single-driver dodecahedron source. The green curve is the estimate from the 2,664-point method in the anechoic chamber. The dotted red line serves as a visual reference for an omnidirectional source.

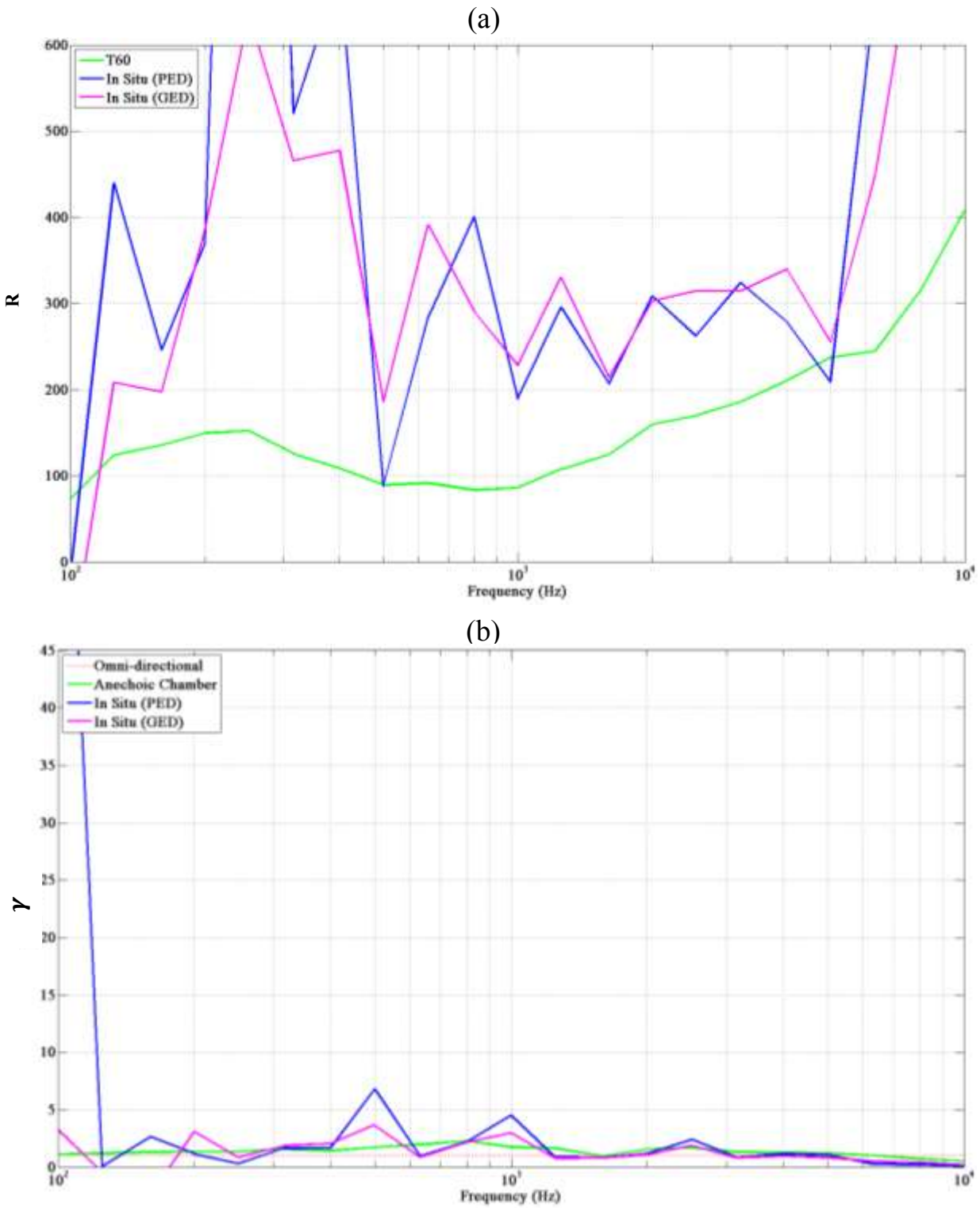

Figure C. 4 Results from the two-point in situ method using the G.R.A.S. Type 50VI and the single-driver dodecahedron as the reference directivity source. a) Estimates of the room constant R for the large room. The green curve is the estimate by the traditional reverberation time method. b) Estimates of the directivity factor γ for the horn-loaded compression driver at 45° off axis. The green curve is the estimate from the 2,664-point method in the anechoic chamber. The dotted red line serves as a visual reference for an omnidirectional source.

Appendix D

Matlab Code for Chapter 2

HybridModal_modified_Thesis.m

```
%% Hybrid Modal Analysis Code for Enclosed Sound Fields
%Code originally developed by Buye Xu, modified and expanded by Daniel Marquez

clc; clear all; close all;

%%
set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',14.4);
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1);
scsz = get(0,'ScreenSize');
rootpath_workspaces = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
Research\Research Presentations\8_2_Research_Presentation\High_Uniform_Abs_Workspaces\';
rootpath_stats = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
Research\Research
Presentations\8_2_Research_Presentation\High_Unif_Abs_Beta_Tests_Power_Est_Power_Err_Data\';
rho = 1.21;                             %density of air
rho_w = 2600;                           %density of the wall (concrete)
h_w = 0.1524;                           %thickness of the wall (6 inches)
S = 1;                                  %area of wall used to compute the wall impedance
C = 343;                                %speed of sound in air
BETA = 1/4;                             %GED weighting factor (1 = Ep; 0 = Ek; 1/2 = (1/2)*Et; 1/4
= Optimized Eg)
pref = 20e-6;                           %reference pressure
vref = pref/(rho*C);                    %reference velocity
Powref = 10e-12;                        %reference power
fmax = 3000;
Epref = pref^2/(2*rho*C^2);             %reference potential energy density
Ekref = (rho/2)*vref^2;                 %reference kinetic energy density
Etref = Epref + Ekref;                  %reference total energy density
Egref = BETA*Epref + (1-BETA)*Ekref;    %reference generalized energy density
dmin_src_wall = 1.5;                    %minimum distance the source must be from the wall (ISO
Standard Value)

%%
knflag = 1;                         %0: classical modes, 1: modified modes (more accurate)
knflagtext = ['Classical';          %text labeling for output
              'Modified '];
soundField.knflag = knflag;         %includes knflag in the soundfield output data file
```

```
greenflag = 1;                      %0: no hybrid, 1: hybrid (includes also the free field green
function)(more accurate)
greenflagtext = ['    ';            %text labeling for output
                 '(FG)'];
soundField.greenflag = greenflag;  %includes greenflag in the soundfield output data file
coupleflag = 1;                     %0: uncoupled, 1: coupled (more accurate, but slows down the
computation)
coupleflagtext = ['Simplified';    %text labeling for output
                  '  Full    '];
soundField.coupleflag = coupleflag;%includes coupleflag in the soundfield output data file
sumflag = 1;                        %-1: don't calculate p or u,  0: KX*X (faster but requires
more memory), 1: sum(KX(i)*X) (slower)
vflag = 1;                          %0: pressure only, 1: p and u
random_source_location = 2;         %0: user designates source location; 1: random source location
in x-y, z=close to the floor (meets ISO standards); 2: random source location in x,y, and z
(meets ISO standards)
random_receiver_locations = 1;      %-1: no receivers; 0: user designates receiver locations; 1:
random receiver locations (meets ISO standards)
stats = 1;                          %0: one run of receiver placement; 1: statistics of 5000 runs
for receiver placements (each from 0 to 6 receivers)
alpha_test = 0;                     %0: no testing of alpha (the absorption coefficient); 1: alpha
test on (determines how the overall sound power estimates using the GED differ when inputing
erroneous values of alpha)
beta_test = 0;                      %0: no testing of BETA; 1:BETA test on (the GED weighting
coefficient)
saveflag = 0;                       %automatically saves plots
save_workspace_flag = 0;            %automatically saves the workspace

%% Directivity factor ^2 should be integrated to 4pi.
lambdaF = @(theta,phi) 1;% sqrt(2)*cos(theta/2); %directivity factors for the two sources

%%
Dim = [4.96 5.88 6.99];
Lx = Dim(1);                    %x dimension
Ly = Dim(2);                    %y dimension
Lz = Dim(3);                    %z dimension
f0 = 100                        %Hz - Driving frequency
w0 = 2*pi*f0;                   %angular frequency
k0 = w0/C;                      %wave number
dB_Power = 80;                          %free-field source sound power (in dB)
Power = Powref*10^(dB_Power/10);        %free-field source sound power
Qs = sqrt((2*Power*C)/(rho*pi*f0^2)); %source strength
u0 = Qs*1i*rho*C*k0;                    %complex velocity at the surface of the source
[x0 y0 z0] = getSrcLoc(random_source_location,Lx,Ly,Lz,dmin_src_wall); %generates a source
location according to which type is selected

%%
% zw =  (200 + 340*1i);             %manually assign a complex value for the impedance of the
wall (assuming uniform absorption in the room)
% Zw = ones(1,3)*zw;                %uniform specific acoustic impedance for all the walls
% alpha = ones(1,3)*alpha;          %uniform normal-incidence absorption coefficients for all
the walls
zw1 = getzw1(w0,rho_w,h_w,S);       %impedance of the two Ly*Lz boundaries
zw2 = getzw2(w0,rho_w,h_w,S);       %impedance of the two Lx*Lz boundaries
zw3 = getzw3(w0,rho_w,h_w,S);       %impedance of the two Lx*Ly boundaries
Zw = [zw1 zw2 zw3];                 %specific acoustic impedance for all the walls
alpha1 = getAlpha1(zw1);            %normal-incidence absorption coefficient (largest wall)
alpha2 = getAlpha2(zw2);            %normal-incidence absorption coefficient (mid-sized wall)
alpha3 = getAlpha3(zw3);            %normal-incidence absorption coefficient (smallest wall)
alpha = [alpha1 alpha2 alpha3]      %uniform normal-incidence absorption coefficients for all
the walls
% soundField.Zw = Zw;               %includes the value of Zw in the soundfield output data file
% soundField.alpha = mean_alpha;    %includes the value of alpha in the soundfield output data
file
beta0 = 1./Zw;
beta = -1i*beta0*k0;                %specific acoustic admittance of the boundary
                                    %(might need to be multiplied by rho*C though)

%%
V = Lx*Ly*Lz;                       %Volume of the enclosure
S = (2*Lx*Ly)+(2*Lx*Lz)+(2*Ly*Lz); %Surface area of the enclosure
```

```
mean_alpha = (2*Ly*Lz*alpha(1) + 2*Lx*Lz*alpha(2) + 2* Lx*Ly*alpha(3))/S  %average normal-
incidence absorption coefficient of the room
A = S*mean_alpha;                   %total absorption in the room
T60 = (0.161*V)/(S*mean_alpha(1));  %reverberation time of the room
% D1 = 0.08;               %for f > 5000 (ISO standard 3741 (8.3))
D1 = 0.16;                  %for f < 5000 (ISO standard 3741 (8.3))
% dmin_mic_src = D1*sqrt(V/T60);    %the minimum distance a receiver can be to the source
% set_dmin_mic_src = D1*sqrt(V/set_T60);
set_dmin_mic_src = 1.64;       %set value for the minimun distance between mic and source
(corresponds to kr=3 for lowest frequency)
dmin_mic_mic = (1/2).*(C/f0);  %minimum distance between mics
dmin_mic_wall = 1;             %minimum distance from mic to any boundaries
f_natural = getf_natural(C,Lx,Ly,Lz); %calculates and creates a [] x 4 matrix: sorted eigen
frequencies, x dim mode index, y dim mode index,z dim mode index
f_nat = f_natural(1:50,:)             %truncated matrix of natural frequencies
mode = getmode(f0,f_natural);  %dictates the closes mode excited with the given driving frequency
fshroeder = C*4/sqrt(pi*2*sum(mean_alpha.*[Ly*Lz, Lx*Lz, Lx*Ly]))-
2*(Ly*Lz+Lx*Lz+Lx*Ly)/16/Lx/Ly/Lz %the shroeder frequency of the room
if f0 > fshroeder
    playsound
end

%%
% addreso = (13.1234*pi)/(2*k0);   %set the resolution of the modeling grid according to
frequency
if f0 < 200                         %automatically adjust the resolution according to frequency
    addreso = 7;
elseif 200 <= f0 && f0 < 400
    addreso = 4;
elseif 400 <= f0 && f0 < 600
    addreso = 2;
elseif 600 <= f0 && f0 < 800
    addreso = 1;
else
    addreso = .5;
end

%%
approach = input(' For summation of modes, do you want to sum only those modes with high
amplitudes, define a radius from k0,\n or use an approximate number of modes? 1:amplitudes,
2:radius, 3:modes ');
switch approach
    case 1
        mode_optflag = 2;
        kr = 3;
    case 2
        radius = input('Do you want to define the radius in terms of modal width or by a set
frequency? 1: modal width, 2: set frequency ');
        switch radius
            case 1
                mode_width = 6.91/(pi*T60);               %equation that calculates the modal
width at half-amplitude
                display(['Modal width = ' num2str(mode_width,'%.2f') ' Hz'])
                optimize = input('Do you want to use the code to help determine how many modal
widths are adequate or do you want to just specify a number? 1:optimize, 2:specify ');
                  switch optimize
                      case 1
                          mode_optflag = 1;
                      case 2
                          mode_optflag = 0;
                          num_mode_widths = input('How many modal widths do you want the radius
to be? ');
                          del_freq = num_mode_widths*mode_width  %the radius of integration in Hz
                          kr = (2*pi*del_freq)/C        %the radius of integration in terms of k
                  end
            case 2
                mode_optflag = 0;
                del_freq = input('What do you want the radius to be? (in Hz) ');
                kr = (2*pi*del_freq)/C                  %the radius of integration in terms of k
        end
    case 3
```

```
        mode_optflag = 0;
        num_modes = input('Approximately how many modes do you want to sum over? ');
        del_freq = (num_modes*100*C)/(2*pi*f0^2) - .001*num_modes  %an empirical equation that
approximates how many modes the code will sum over
        kr = (2*pi*del_freq)/C                    %the radius of integration in terms of k
end

%%
if mode_optflag == 1
   a = real(beta);                    %real part of beta
   b = imag(beta);                    %imaginary part of beta
   if knflag == 1                     %modified modes are used.
            ax = a(1);                %this represents beta prime; if beta prime is real, the
            ay = a(2);                %eigenvalues are real and the resulting modal functions
            az = a(3);                %are guaranteed to be complete and orthogonal
            bx = 1i*b(1);
            by = 1i*b(2);
            bz = 1i*b(3);
   else                               %classical modes are used
            ax = 0;                   %this represents beta; the boundary is usually assumed
            ay = 0;                   %locally reacting and is purely imaginary
            az = 0;
            bx = beta(1);
            by = beta(2);
            bz = beta(3);
   end                                %end of modified/classical modes programming
   if sumflag >= 0      %when either p or p and u are wanting to be calculated
           % Define the field grids, x,y,z.
           Xresolution = getReso(Lx,k0,addreso);
           x = (0:Lx/Xresolution:Lx);
           Nx = length(x);
           Yresolution = getReso(Ly,k0,addreso);
           y = (0:Ly/Yresolution:Ly);
           Ny = length(y);
           Zresolution = getReso(Lz,k0,addreso);
           z = (0:Lz/Zresolution:Lz);
           Nz = length(z);
           [X,Y,Z] = vectorlize(x,y,z);
           r2 = (X-x0).^2 + (Y-y0).^2 + (Z-z0).^2;
           r = r2.^.5;
           theta = acos((Z-z0)./r);
           phi = 0;
   end
    skip = 2;
    beg = 1;
    finish = 50;
    tolerance = 0.1; %dB
    kr = zeros(finish-beg+2,1);
    kxN = zeros(finish-beg+2,1);
    kyN = zeros(finish-beg+2,1);
    kzN = zeros(finish-beg+2,1);
    tot_modes = zeros(finish-beg+1,1);
    inc_modes = zeros(finish-beg+2,1);
    PM_runs = zeros(1,Nx*Ny*Nz,finish-beg+2);
    VMx_runs = zeros(1,Nx*Ny*Nz,finish-beg+2);
    VMy_runs = zeros(1,Nx*Ny*Nz,finish-beg+2);
    VMz_runs = zeros(1,Nx*Ny*Nz,finish-beg+2);
    full_PM = zeros(1,Nx*Ny*Nz,finish-beg+2);
    full_VMx = zeros(1,Nx*Ny*Nz,finish-beg+2);
    full_VMy = zeros(1,Nx*Ny*Nz,finish-beg+2);
    full_VMz = zeros(1,Nx*Ny*Nz,finish-beg+2);
    mean_dB_Ep = zeros(finish-beg+1,1);
    ctr = 1;

    for g = beg:finish
        resumflag = 0;
        ctr = ctr + 1;
        del_freq = g*skip*mode_width;
        kr(ctr) = (2*pi*del_freq)/C;
        kxN(ctr) = ceil((k0+kr(ctr))*Lx/pi/2)*2;    %the mth mode in the x direction with the
given k value (k0+kr)
```

```
        kyN(ctr) = ceil((k0+kr(ctr))*Ly/pi/2)*2;    %the nth mode in the y direction with the
given k value (k0+kr)
        kzN(ctr) = ceil((k0+kr(ctr))*Lz/pi/2)*2;    %the lth mode in the z direction with the
given k value (k0+kr)

        if kxN(ctr) ~= kxN(ctr-1)
            resumflag = 1;
            Kx = getKn(kxN(ctr),beta(1),k0,Lx,knflag);
            Bx = -ax./Kx;
            Bx(isinf(Bx))=0;
            Bx(isnan(Bx))=0;
        end
        if kyN(ctr) ~= kyN(ctr-1)
            resumflag = 1;
            Ky = getKn(kyN(ctr),beta(2),k0,Ly,knflag);
            By = -ay./Ky;
            By(isinf(By))=0;
            By(isnan(By))=0;
        end
        if kzN(ctr) ~= kzN(ctr-1)
            resumflag = 1;
            Kz = getKn(kzN(ctr),beta(3),k0,Lz,knflag);
            Bz = -az./Kz;
            Bz(isinf(Bz))=0;
            Bz(isnan(Bz))=0;
        end
        if resumflag == 1
            [MA_full,CNN_full] =
            getMA(k0,Kx,Ky,Kz,kr,Bx,By,Bz,Lx,Ly,Lz,ax,ay,az,bx,by,bz,coupleflag);
            [MB_full,A0,B0] =
            getMB(Kx,Ky,Kz,Bx,By,Bz,Lx,Ly,Lz,beta,k0,u0,x0,y0,z0,greenflag,lambdaF);
            [KX_full,KY_full,KZ_full] = vectorlize(Kx,Ky,Kz);
            [BX_full,BY_full,BZ_full] = vectorlize(Bx,By,Bz);
            K2_full_sqr = real(KX_full).^2 + real(KY_full).^2 + real(KZ_full).^2;
            K2_full = sqrt(K2_full_sqr);
         end
        tot_modes(ctr-1) = length(KX_full);
        total_modes = tot_modes(ctr-1)
        ind1 = find(abs(K2_full-k0) > kr(ctr));
        ind2 = find(abs(K2_full-k0) <= kr(ctr-1));
        ind = cat(2,ind1,ind2);
        K2 = K2_full;
        K2(ind) = [];
        MA = MA_full;
        MA(ind,:) = [];
        MA(:,ind) = [];
        KX = KX_full;
        KX(ind) = [];
        KY = KY_full;
        KY(ind) = [];
        KZ = KZ_full;
        KZ(ind) = [];
        BX = BX_full;
        BX(ind) = [];
        BY = BY_full;
        BY(ind) = [];
        BZ = BZ_full;
        BZ(ind) = [];
        CNN = CNN_full;
        CNN(ind) = [];
        MB = MB_full;
        MB(ind) = [];
        PN = (MA\MB).';
        inc_modes(ctr) = inc_modes(ctr-1) + length(KX);
        included_modes = inc_modes(ctr)

        %%
        if sumflag >= 0      %when either p or p and u are wanting to be calculated
            [PM,VMx,VMy,VMz] = sumPsiV(k0,X,Y,Z,KX,KY,KZ,PN,BX,BY,BZ,sumflag,vflag);
            PM_runs(1,:,ctr) = PM(1,:);
            VMx_runs(1,:,ctr) = VMx(1,:);
```

```
        VMy_runs(1,:,ctr) = VMy(1,:);
        VMz_runs(1,:,ctr) = VMz(1,:);
        full_PM(1,:,ctr) = full_PM(1,:,ctr-1) + PM_runs(1,:,ctr);
        full_VMx(1,:,ctr) = full_VMx(1,:,ctr-1) + VMx_runs(1,:,ctr);
        full_VMy(1,:,ctr) = full_VMy(1,:,ctr-1) + VMy_runs(1,:,ctr);
        full_VMz(1,:,ctr) = full_VMz(1,:,ctr-1) + VMz_runs(1,:,ctr);
        P01 = u0/4/pi./r.*(A0.*exp(-1i*k0*r)).*lambdaF(theta,phi);
        P02 = u0/4/pi./r.*(B0.*exp(1i*k0*r));
        P0 = P01 + P02;
        PL = full_PM(1,:,ctr) + P0;
        P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the same
        dimensions as the rooom
    end                             %end of p/p and u programming

    %%
    %Generate the Potential Energy Density Matrix
    Ep = getEp(P,rho,C);        %potential energy density field
    %Find the energetic (linear) mean value of the entire valid field in the room
    (requirements: at least 1 m from 1)any boundary and 2)set_dmin_mic_src from the source
    (which corresponds to a distance when alpha = 0.01))
    mean_Ep = getmean_Ep(Nx,Ny,Nz,Lx,Ly,Lz,Ep,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0);
    %Put the energetic mean value of the entire valid field into dB
    mean_dB_Ep(ctr-1) = 10*log10(mean_Ep/Epref);
    progress = ((ctr-1)/(finish-beg+1))*100*2
    n = skip*beg:skip:skip*(ctr-1);
    x_line = [0 n(ctr-1)];
    y_line_top = [mean_dB_Ep(ctr-1)+(tolerance/2) mean_dB_Ep(ctr-1)+(tolerance/2)];
    y_line_bot = [mean_dB_Ep(ctr-1)-(tolerance/2) mean_dB_Ep(ctr-1)-(tolerance/2)];

    figure(19)
    set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
    plot(n(1:ctr-1),mean_dB_Ep(1:ctr-1))
    title(['Converging Potential Energy Density Field, f=' num2str(f0,'%.1f') ' Hz, \alpha='
    num2str(mean_alpha,'%.2f') ', \Delta f=' num2str(mode_width,'%.2f') ' Hz'])
    xlabel('# of Modal Widths')
    ylabel('Field Mean (dB)')
    line(x_line,y_line_top,'Color','r','linestyle','--')
    line(x_line,y_line_bot,'Color','r','linestyle','--')
    xlim([0 n(ctr-1)])
    ylim([min(mean_dB_Ep(1:ctr-1))-0.1 max(mean_dB_Ep)+0.1])
    drawnow

    if ctr >= ceil(((finish-beg+1)/2)+1)
        playsound
        toc
        convergence = input('\n Has the energy field converged? 1:Yes, 2:No (keep going) ');
        switch convergence
            case 1
                break
            case 2
                continue
        end
    end
end

%%
num_mode_widths = input('\n How many modal widths do you want the radius to be? ');
dist = abs(num_mode_widths - n);
[pot_dist,dist_ind] = min(dist);
sort_dist = sort(dist);

if sort_dist(1) == 0
    del_freq = n(dist_ind)*mode_width        %the radius of integration in Hz
    kr = (2*pi*del_freq)/C                   %the radius of integration in terms of k
    chosen_iter = n(dist_ind)/skip;
    total_modes = tot_modes(chosen_iter)
    included_modes = inc_modes(chosen_iter+1)
else
    greater = n(ceil(num_mode_widths/skip));
    less = n(floor(num_mode_widths/skip));
    greater_or_less = input(['\nThe energy field was not calculated for that exact # of modal
```

```
        widths.\n Do you want to use the # of modal widths just greater (' num2str(greater,'%.f')
        ') or just less (' num2str(less,'%.f') ') than your intial request of '
        num2str(num_mode_widths,'%.f') '? 1: greater, 2: less ']);
        switch greater_or_less
            case 1
                del_freq = greater*mode_width        %the radius of integration in Hz
                kr = (2*pi*del_freq)/C               %the radius of integration in terms of k
                chosen_iter = greater/skip;
                total_modes = tot_modes(chosen_iter)
                included_modes = inc_modes(chosen_iter+1)
            case 2
                del_freq = less*mode_width           %the radius of integration in Hz
                kr = (2*pi*del_freq)/C               %the radius of integration in terms of k
                chosen_iter = less/skip;
                total_modes = tot_modes(chosen_iter)
                included_modes = inc_modes(chosen_iter+1)
        end
    end

    %% NOW USE THE CHOSEN FIELD AND PROCEEED
    if sumflag >= 0      %when either p or p and u are wanting to be calculated
        PM = full_PM(1,:,chosen_iter+1);
        VMx = full_VMx(1,:,chosen_iter+1);
        VMy = full_VMy(1,:,chosen_iter+1);
        VMz = full_VMz(1,:,chosen_iter+1);
        PL = PM + P0;
        P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the same
        dimensions as the rooom
        V01x = -1i*(X-x0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V01y = -1i*(Y-y0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V01z = -1i*(Z-z0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V02x = 1i*(X-x0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        V02y = 1i*(Y-y0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        V02z = 1i*(Z-z0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        VxL = VMx + V01x+V02x;
        VyL = VMy + V01y+V02y;
        VzL = VMz + V01z+V02z;
        Vx = (1/(rho*C))*reshape(VxL,Nx,Ny,Nz); %reshapes the VxL, VyL, and VzL vectors into
        matrices that have the same dimensions as the rooom
        Vy = (1/(rho*C))*reshape(VyL,Nx,Ny,Nz); %they are all multiplied by 1/(rho*C) because the
        original code scaled Vx, Vy, and Vz by rho*C
        Vz = (1/(rho*C))*reshape(VzL,Nx,Ny,Nz);
        soundField.P = P;   %store P, Vx, Vy, and Vz into the soundfield output data file
        soundField.Vx = Vx;
        soundField.Vy = Vy;
        soundField.Vz = Vz;
    %     save([rootpath,filename1,'Field',filename2,'.mat'], 'soundField');
    end                              %end of p/p and u programming
end                %end of mode_optflag calculations

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if mode_optflag ==0
    kxN = ceil((k0+kr)*Lx/pi/2)*2;     %the mth mode in the x direction with the given k value
(k0+kr)
    kyN = ceil((k0+kr)*Ly/pi/2)*2;     %the nth mode in the y direction with the given k value
(k0+kr)
    kzN = ceil((k0+kr)*Lz/pi/2)*2;     %the lth mode in the z direction with the given k value
(k0+kr)
    a = real(beta);                    %real part of beta
    b = imag(beta);                    %imaginary part of beta

    if knflag == 1                     %modified modes are used.
        ax = a(1);                     %this represents beta prime; if beta prime is real, the
        ay = a(2);                     %eigenvalues are real and the resulting modal functions
        az = a(3);                     %are guaranteed to be complete and orthogonal
        bx = 1i*b(1);
        by = 1i*b(2);
        bz = 1i*b(3);
```

```
else                                %classical modes are used
    ax = 0;                         %this represents beta; the boundary is usually assumed
    ay = 0;                         %locally reacting and is purely imaginary
    az = 0;
    bx = beta(1);
    by = beta(2);
    bz = beta(3);
end                                 %end of modified/classical modes programming
Kx = getKn(kxN,beta(1),k0,Lx,knflag);
kxN = length(Kx);
Ky = getKn(kyN,beta(2),k0,Ly,knflag);
kyN = length(Ky);
Kz = getKn(kzN,beta(3),k0,Lz,knflag);
kzN = length(Kz);
Bx = -ax./Kx;
By = -ay./Ky;
Bz = -az./Kz;
Bx(isinf(Bx))=0;
By(isinf(By))=0;
Bz(isinf(Bz))=0;
Bx(isnan(Bx))=0;
By(isnan(By))=0;
Bz(isnan(Bz))=0;
[KX,KY,KZ] = vectorlize(Kx,Ky,Kz);       %calls the function that reshapes
[BX,BY,BZ] = vectorlize(Bx,By,Bz);       %calls the function that reshapes
tot_modes = length(KX)

%%
[MA,CNN] = getMA(k0,Kx,Ky,Kz,kr,Bx,By,Bz,Lx,Ly,Lz,ax,ay,az,bx,by,bz,coupleflag);
[MB,A0,B0] = getMB(Kx,Ky,Kz,Bx,By,Bz,Lx,Ly,Lz,beta,k0,u0,x0,y0,z0,greenflag,lambdaF);
K2 = real(KX).^2 + real(KY).^2 + real(KZ).^2;
ind = find(abs(sqrt(K2)-k0)>(kr));
K2(ind) = [];
MA(ind,:) = [];%
MA(:,ind) = [];
KX(ind) = [];
KY(ind) = [];
KZ(ind) = [];
BX(ind) = [];
BY(ind) = [];
BZ(ind) = [];
CNN(ind) = [];
MB(ind) = [];
PN = (MA\MB).';
included_modes = length(KX)

%%
if sumflag >= 0      %when either p or p and u are wanting to be calculated
    %Define the field grids, x,y,z.
    Xresolution = getReso(Lx,k0,addreso);
    x = (0:Lx/Xresolution:Lx);
    Nx = length(x);
    Yresolution = getReso(Ly,k0,addreso);
    y = (0:Ly/Yresolution:Ly);
    Ny = length(y);
    Zresolution = getReso(Lz,k0,addreso);
    z = (0:Lz/Zresolution:Lz);
    Nz = length(z);
    soundField.x = x;
    soundField.y = y;
    soundField.z = z;
    [X,Y,Z] = vectorlize(x,y,z);

%%
    [PM,VMx,VMy,VMz] = sumPsiV(k0,X,Y,Z,KX,KY,KZ,PN,BX,BY,BZ,sumflag,vflag);

%%
    r2 = (X-x0).^2 + (Y-y0).^2 + (Z-z0).^2;
    r = r2.^.5;
    theta = acos((Z-z0)./r);
    phi = 0;
```

```
        P01 = u0/4/pi./r.*(A0.*exp(-1i*k0*r)).*lambdaF(theta,phi);
        P02 = u0/4/pi./r.*(B0.*exp(1i*k0*r));
        P0 = P01 + P02;
        PL = PM + P0;
        P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the same
        dimensions as the rooom
        V01x = -1i*(X-x0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V01y = -1i*(Y-y0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V01z = -1i*(Z-z0).*(1 + 1i*k0*r)./r2.*P01/k0;
        V02x = 1i*(X-x0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        V02y = 1i*(Y-y0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        V02z = 1i*(Z-z0).*(-1 + 1i*k0*r)./r2.*P02/k0;
        VxL = VMx + V01x+V02x;
        VyL = VMy + V01y+V02y;
        VzL = VMz + V01z+V02z;
        Vx = (1/(rho*C))*reshape(VxL,Nx,Ny,Nz); %reshapes the VxL, VyL, and VzL vectors into
        matrices that have the same dimensions as the rooom
        Vy = (1/(rho*C))*reshape(VyL,Nx,Ny,Nz); %they are all multiplied by 1/(rho*C) because the
        original code scaled Vx, Vy, and Vz by rho*C
        Vz = (1/(rho*C))*reshape(VzL,Nx,Ny,Nz);
        soundField.P = P;   %store P, Vx, Vy, and Vz into the soundfield output data file
        soundField.Vx = Vx;
        soundField.Vy = Vy;
        soundField.Vz = Vz;
    %     save([rootpath,filename1,'Field',filename2,'.mat'], 'soundField');
    end                             %end of p/p and u programming
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if mode_optflag == 2
    clock1 = tic;
    kxN = ceil((k0+kr)*Lx/pi/2)*2;     %the mth mode in the x direction with the given k value
                                       (k0+kr)
    kyN = ceil((k0+kr)*Ly/pi/2)*2;     %the nth mode in the y direction with the given k value
                                       (k0+kr)
    kzN = ceil((k0+kr)*Lz/pi/2)*2;     %the lth mode in the z direction with the given k value
                                       (k0+kr)
    a = real(beta);                    %real part of beta
    b = imag(beta);                    %imaginary part of beta
    if knflag == 1                     %modified modes are used.
        ax = a(1);                     %this represents beta prime; if beta prime is real, the
        ay = a(2);                     %eigenvalues are real and the resulting modal functions
        az = a(3);                     %are guaranteed to be complete and orthogonal
        bx = 1i*b(1);
        by = 1i*b(2);
        bz = 1i*b(3);
    else                               %classical modes are used
        ax = 0;                        %this represents beta; the boundary is usually assumed
        ay = 0;                        %locally reacting and is purely imaginary
        az = 0;
        bx = beta(1);
        by = beta(2);
        bz = beta(3);
    end                                %end of modified/classical modes programming
    if sumflag >= 0      %when either p or p and u are wanting to be calculated
        % Define the field grids, x,y,z.
        Xresolution = getResolutionCap(fmax,Lx,C);
        x = (0:Lx/Xresolution:Lx);
        Nx = length(x);
        Yresolution = getResolutionCap(fmax,Ly,C);
        y = (0:Ly/Yresolution:Ly);
        Ny = length(y);
        Zresolution = getResolutionCap(fmax,Lz,C);
        z = (0:Lz/Zresolution:Lz);
        Nz = length(z);
        [X,Y,Z] = vectorlize(x,y,z);
        r2 = (X-x0).^2 + (Y-y0).^2 + (Z-z0).^2;
        r = r2.^.5;
        theta = acos((Z-z0)./r);
```

```
    phi = 0;
end
Kx = getKn(kxN,beta(1),k0,Lx,knflag);
kxN = length(Kx);
Ky = getKn(kyN,beta(2),k0,Ly,knflag);
kyN = length(Ky);
Kz = getKn(kzN,beta(3),k0,Lz,knflag);
kzN = length(Kz);
Bx = -ax./Kx;
By = -ay./Ky;
Bz = -az./Kz;
Bx(isinf(Bx))=0;
By(isinf(By))=0;
Bz(isinf(Bz))=0;
Bx(isnan(Bx))=0;
By(isnan(By))=0;
Bz(isnan(Bz))=0;
[KX_full,KY_full,KZ_full] = vectorlize(Kx,Ky,Kz);
[BX_full,BY_full,BZ_full] = vectorlize(Bx,By,Bz);
tot_modes = length(KX_full)

%%
[MA_full,CNN_full] = getMA(k0,Kx,Ky,Kz,kr,Bx,By,Bz,Lx,Ly,Lz,ax,ay,az,bx,by,bz,coupleflag);
[MB_full,A0,B0] = getMB(Kx,Ky,Kz,Bx,By,Bz,Lx,Ly,Lz,beta,k0,u0,x0,y0,z0,greenflag,lambdaF);
amplitudes = MA_full\MB_full;
norm_amps = amplitudes./max(amplitudes);
sort_norm_amps = sort(norm_amps,'descend');
playsound

%%
figure(25)
set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
subplot(3,1,1)
plot(abs(amplitudes/max(amplitudes)).*100)
xlabel('Modes')
ylabel('Normalized Modal Amplitude (%)')
title('Modal Amplitudes')
subplot(3,1,2)
plot(abs(sort_norm_amps).*100)
xlabel('Modes')
ylabel('Normalized Modal Amplitude (%)')
title('Sorted Modal Amplitudes')
grid on
subplot(3,1,3)
semilogx(abs(sort_norm_amps).*100)
title('Sorted Modal Amplitudes')
xlabel('Modes')
ylabel('Normalized Modal Amplitude (%)')
grid on
drawnow

total_time = toc(clock1)
method = input('Do you want to choose a percentile for the summation cutoff or have the code
show you the converging PED field to help you decide?  1:specify, 2:optimize ');
switch method
    case 1
        percentage = input('Based on the plot of normalized modal amplitudes, at what
        percentile do you want to make the cutoff for modal summation? (in percent) ');
        clock2 = tic;
        x_line = [1 length(KX_full)];
        y_line = [percentage percentage];

        figure(26)
        set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
        subplot(3,1,1)
        plot(abs(amplitudes/max(amplitudes)).*100)
        xlabel('Modes')
        ylabel('Normalized Modal Amplitude (%)')
        title('Modal Amplitudes')
        line(x_line,y_line,'Color','r')
        subplot(3,1,2)
```

```
            plot(abs(sort_norm_amps).*100)
            xlabel('Modes')
            ylabel('Normalized Modal Amplitude (%)')
            title('Sorted Modal Amplitudes')
            grid on
            subplot(3,1,3)
            semilogx(abs(sort_norm_amps).*100)
            title('Sorted Modal Amplitudes')
            xlabel('Modes')
            ylabel('Normalized Modal Amplitude (%)')
            line(x_line,y_line,'Color','r');
            grid on
            drawnow

            low_amps = find(abs(norm_amps) < percentage/100);
            MA = MA_full;
            MA(low_amps,:) = [];
            MA(:,low_amps) = [];
            KX = KX_full;
            KX(low_amps) = [];
            KY = KY_full;
            KY(low_amps) = [];
            KZ = KZ_full;
            KZ(low_amps) = [];
            BX = BX_full;
            BX(low_amps) = [];
            BY = BY_full;
            BY(low_amps) = [];
            BZ = BZ_full;
            BZ(low_amps) = [];
            CNN = CNN_full;
            CNN(low_amps) = [];
            MB = MB_full;
            MB(low_amps) = [];
            PN = (MA\MB).';
            included_modes = length(KX)
            ISO_ind = 1;

            %%
            [PM,VMx,VMy,VMz] = sumPsiV(k0,X,Y,Z,KX,KY,KZ,PN,BX,BY,BZ,sumflag,vflag);
            P01 = u0/4/pi./r.*(A0.*exp(-1i*k0*r)).*lambdaF(theta,phi);
            P02 = u0/4/pi./r.*(B0.*exp(1i*k0*r));
            P0 = P01 + P02;
            PL = PM + P0;
            P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the same
            dimensions as the rooom
            V01x = -1i*(X-x0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V01y = -1i*(Y-y0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V01z = -1i*(Z-z0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V02x = 1i*(X-x0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            V02y = 1i*(Y-y0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            V02z = 1i*(Z-z0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            VxL = VMx + V01x+V02x;
            VyL = VMy + V01y+V02y;
            VzL = VMz + V01z+V02z;
            Vx = (1/(rho*C))*reshape(VxL,Nx,Ny,Nz); %reshapes the VxL, VyL, and VzL vectors into
            matrices that have the same dimensions as the rooom
            Vy = (1/(rho*C))*reshape(VyL,Nx,Ny,Nz); %they are all multiplied by 1/(rho*C) because
            the original code scaled Vx, Vy, and Vz by rho*C
            Vz = (1/(rho*C))*reshape(VzL,Nx,Ny,Nz);
            soundField.P = P;   %store P, Vx, Vy, and Vz into the soundfield output data file
            soundField.Vx = Vx;
            soundField.Vy = Vy;
            soundField.Vz = Vz;
%             save([rootpath,filename1,'Field',filename2,'.mat'], 'soundField');
            total_time = total_time + toc(clock2)
        case 2
                beg = input('At what percentile do you want to begin for modal summation? (in
                percent)' );
                step = input('What do you want the step increments to be? (in percent)' );
                finish = input('At what percentile do you wnat to finish for modal summation? (in
```

```
    percent)' );
    clock2 = tic;
    tolerance = 0.1; %dB
    percentage = zeros(ceil(abs(finish-beg)+2),1);
    inc_modes = zeros(ceil(abs(finish-beg)+2),1);
    full_PM = zeros(1,Nx*Ny*Nz,ceil(abs(finish-beg)+2));
    full_VMx = zeros(1,Nx*Ny*Nz,ceil(abs(finish-beg)+2));
    full_VMy = zeros(1,Nx*Ny*Nz,ceil(abs(finish-beg)+2));
    full_VMz = zeros(1,Nx*Ny*Nz,ceil(abs(finish-beg)+2));
    mean_dB_Ep = zeros(ceil(abs(finish-beg)+1),1);
    ctr = 1;
    percentage(1) = 100;
    for g = beg:-step:finish
        ctr = ctr + 1;
        percentage(ctr) = g;
        y_line = [percentage(ctr) percentage(ctr)];
        low_amps = find(abs(norm_amps) < percentage(ctr)/100);
        used_amps = find(abs(norm_amps) > percentage(ctr-1)/100);
        if isempty(used_amps)
            ind = low_amps;
        else
            ind = cat(1,low_amps,used_amps);
        end
        MA = MA_full;
        MA(ind,:) = [];
        MA(:,ind) = [];
        KX = KX_full;
        KX(ind) = [];
        KY = KY_full;
        KY(ind) = [];
        KZ = KZ_full;
        KZ(ind) = [];
        BX = BX_full;
        BX(ind) = [];
        BY = BY_full;
        BY(ind) = [];
        BZ = BZ_full;
        BZ(ind) = [];
        CNN = CNN_full;
        CNN(ind) = [];
        MB = MB_full;
        MB(ind) = [];
        PN = (MA\MB).';
        inc_modes(ctr) = inc_modes(ctr-1) + length(KX);
        included_modes = inc_modes(ctr)
        [PM,VMx,VMy,VMz] = sumPsiV(k0,X,Y,Z,KX,KY,KZ,PN,BX,BY,BZ,sumflag,vflag);
        full_PM(1,:,ctr) = full_PM(1,:,ctr-1) + PM;
        full_VMx(1,:,ctr) = full_VMx(1,:,ctr-1) + VMx;
        full_VMy(1,:,ctr) = full_VMy(1,:,ctr-1) + VMy;
        full_VMz(1,:,ctr) = full_VMz(1,:,ctr-1) + VMz;
        P01 = u0/4/pi./r.*(A0.*exp(-1i*k0*r)).*lambdaF(theta,phi);
        P02 = u0/4/pi./r.*(B0.*exp(1i*k0*r));
        P0 = P01 + P02;
        PL = full_PM(1,:,ctr) + P0;
        P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the
        same dimensions as the rooom

       %%
        %Generate the Potential Energy Density Matrix
        Ep = getEp(P,rho,C);        %potential energy density field
        %Find the energetic (linear) mean value of the entire valid field in the room
        (requirements: at least 1 m from 1)any boundary and 2)set_dmin_mic_src from
        the source (which corresponds to a distance when mean_alpha = 0.01))
        if ctr == 2
           ISO_ind = 1;
        end
        [ISO_ind,mean_Ep] =
getMean_PE(Nx,Ny,Nz,Lx,Ly,Lz,Ep,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0,ctr,ISO_ind);
        %Put the energetic mean value of the entire valid field into dB
        mean_dB_Ep(ctr-1) = 10*log10(mean_Ep/Epref);
        progress = ((ctr-1)/(length(beg:-step:finish)))*100
```

```
    n = beg:-step:beg-(step*(ctr-2));
    x_line1 = [0 length(KX_full)];
    x_line2 = [beg n(ctr-1)];
    y_line_top = [mean_dB_Ep(ctr-1)+ tolerance mean_dB_Ep(ctr-1)+ tolerance];
    y_line_bot = [mean_dB_Ep(ctr-1)- tolerance mean_dB_Ep(ctr-1)- tolerance];

    %%
    figure(19)
    set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
    subplot(2,1,1)
    plot(abs(amplitudes/max(amplitudes)).*100)
    xlabel('Modes')
    ylabel('Normalized Modal Amplitude (%)')
    title('Modal Amplitudes')
    line(x_line1,y_line,'Color','r')
    subplot(2,1,2)
    plot(n(1:ctr-1),mean_dB_Ep(1:ctr-1))
    set(gca,'xdir','reverse')
    title(['Converging Potential Energy Density Field, f=' num2str(f0,'%.1f') '
    Hz, \alpha=' num2str(mean_alpha,'%.2f')])
    xlabel('Percentile of Normalized Amplitudes Included (%)')
    ylabel('Field Mean (dB)')
    line(x_line2,y_line_top,'Color','r','linestyle','--')
    line(x_line2,y_line_bot,'Color','r','linestyle','--')
    ylim([min(mean_dB_Ep(1:ctr-1))-0.15 max(mean_dB_Ep)+0.15])
    drawnow
    playwaitsound
    ch = getkeywait(11);
    if ch == -1
      continue
    else
      break
    end
end
playsound
total_time = total_time + toc(clock2)

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
percentile_input = input('\n Which percentile do you want to use? ');
clock3 = tic;
dist = abs(percentile_input - n);
[pot_dist,dist_ind] = min(dist);
sort_dist = sort(dist);
if sort_dist(1) == 0
    percentile = n(dist_ind)                %the percentile of inlcuded
    normalized modal amplitudes
    chosen_iter = find(n == percentile);
    total_modes = tot_modes
    included_modes = inc_modes(dist_ind + 1)
else
    temp_n = fliplr(n);
    greater = temp_n(ceil(percentile_input/step));
    less = temp_n(floor(percentile_input/step));
    less_or_greater = input(['\nThe energy field was not calculated for that
    exact percentile.\n Do you want to use the percentile just less ('
    num2str(less,'%.2f') ') or just greater (' num2str(greater,'%.2f') ') than
    your intial request of ' num2str(percentile_input,'%.2f') '? 1: less, 2:
    greater ']);
    switch less_or_greater
        case 1
            percentile = less                %the percentile of inlcuded
            normalized modal amplitudes
            total_modes = tot_modes
            chosen_iter = find(n == less);
            included_modes = inc_modes(chosen_iter + 1)
        case 2
            percentile = greater                %the percentile of inlcuded
            normalized modal amplitudes
            total_modes = tot_modes
            chosen_iter = find(n == greater);
            included_modes = inc_modes(chosen_iter + 1)
```

```
                end
            end

            % NOW USE THE CHOSEN FIELD AND PROCEEED
            PM = full_PM(1,:,chosen_iter+1);
            VMx = full_VMx(1,:,chosen_iter+1);
            VMy = full_VMy(1,:,chosen_iter+1);
            VMz = full_VMz(1,:,chosen_iter+1);
            r = r2.^.5;
            P01 = u0/4/pi./r.*(A0.*exp(-1i*k0*r)).*lambdaF(theta,phi);
            P02 = u0/4/pi./r.*(B0.*exp(1i*k0*r));
            P0 = P01 + P02;
            PL = PM + P0;
            P = reshape(PL,Nx,Ny,Nz); %reshapes the P vector into a matrix that has the same
            dimensions as the rooom
            V01x = -1i*(X-x0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V01y = -1i*(Y-y0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V01z = -1i*(Z-z0).*(1 + 1i*k0*r)./r2.*P01/k0;
            V02x = 1i*(X-x0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            V02y = 1i*(Y-y0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            V02z = 1i*(Z-z0).*(-1 + 1i*k0*r)./r2.*P02/k0;
            VxL = VMx + V01x+V02x;
            VyL = VMy + V01y+V02y;
            VzL = VMz + V01z+V02z;
            Vx = (1/(rho*C))*reshape(VxL,Nx,Ny,Nz); %reshapes the VxL, VyL, and VzL vectors
            into matrices that have the same dimensions as the rooom
            Vy = (1/(rho*C))*reshape(VyL,Nx,Ny,Nz); %they are all multiplied by 1/(rho*C)
            because the original code scaled Vx, Vy, and Vz by rho*C
            Vz = (1/(rho*C))*reshape(VzL,Nx,Ny,Nz);
            soundField.P = P;   %store P, Vx, Vy, and Vz into the soundfield output data file
            soundField.Vx = Vx;
            soundField.Vy = Vy;
            soundField.Vz = Vz;
        %    save([rootpath,filename1,'Field',filename2,'.mat'], 'soundField');
            total_time = total_time + toc(clock3)
    end
end            %end of mode_optflag == 2 programming

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
clock4 = tic;
%Generate the Energy Density Matrices
Ep = getEp(P,rho,C);        %potential energy density field
Ek = getEk(Vx,Vy,Vz,rho);   %kinetic energy density field
Et = getEt(Ep,Ek);          %total Energy Density field
Eg = getEg(Ep,Ek,BETA);     %generalized Energy Density field
%Generate dB quantities
dB_P = 20*log10(abs(P)/pref);    %SPL field
dB_Ep = 10*log10(abs(Ep)/Epref); %potential energy density field in dB
dB_Ek = 10*log10(abs(Ek)/Ekref); %kinetic energy density field in dB
dB_Et = 10*log10(abs(Et)/Etref); %total energy density field in dB
dB_Eg = 10*log10(abs(Eg)/Egref); %generalized energy density field in dB
%Find the energetic (linear) mean value of the entire valid field in the room (requirements: at
least 1 m from 1)any boundary and 2)set_dmin_mic_src from the source (which corresponds to a
distance when alpha = 0.01))
[mean_Ep,mean_Ek,mean_Et,mean_Eg,Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_fie
ld_points_lin,Eg_valid_field_points_lin] =
getFieldMeans(Nx,Ny,Nz,Lx,Ly,Lz,Ep,Ek,Et,Eg,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0);
%Put the energetic mean value of the entire valid field into dB
mean_dB_Ep = 10*log10(mean_Ep/Epref);
mean_dB_Ek = 10*log10(mean_Ek/Ekref);
mean_dB_Et = 10*log10(mean_Et/Etref);
mean_dB_Eg = 10*log10(mean_Eg/Egref);
%Find the standard deviation of the valid energetic (linear) field in the room
sd_Ep = std(Ep_valid_field_points_lin);
sd_Ek = std(Ek_valid_field_points_lin);
sd_Et = std(Et_valid_field_points_lin);
sd_Eg = std(Eg_valid_field_points_lin);
%Find the coefficient of variation (mean-normalized standard deviation) of
```

```
%the valid energetic (linear) field in the room
cv_Ep = sd_Ep/mean_Ep;
cv_Ek = sd_Ek/mean_Ek;
cv_Et = sd_Et/mean_Et;
cv_Eg = sd_Eg/mean_Eg;
if save_workspace_flag == 1
    save([rootpath_workspaces num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
'_Uniform_Absorp_Workspace.mat'])
end

%%
% BETA tests (GED weighting factor)
if beta_test == 1
    var_BETA = 0:0.01:1;
    temp_BETA = zeros(1,length(var_BETA));
    cv_Eg_temp = zeros(1,length(var_BETA));
        for b = 1:length(var_BETA)
            temp_BETA(b) = var_BETA(b);
            Eg_temp = getEg(Ep,Ek,var_BETA(b));
            if length(ISO_ind) < 2
                ISO_ind = 1;
            end
            [ISO_ind,mean_Eg,Eg_valid_field_points_lin] =
           getMean_GE(Nx,Ny,Nz,Lx,Ly,Lz,Eg_temp,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0,ISO_ind);
            sd_Eg = std(Eg_valid_field_points_lin);
            cv_Eg_temp(b) = sd_Eg/mean_Eg;
        end
[min_cv_Eg,min_ind] = min(cv_Eg_temp);
line_PED = [cv_Ep cv_Ep];
line_KED = [cv_Ek cv_Ek];
line_TED = [cv_Et cv_Et];
line_var_BETA = [var_BETA(1) var_BETA(end)];
line_min_y = [0 min_cv_Eg];
line_min_x = [var_BETA(min_ind) var_BETA(min_ind)];

figure(46)
plot(temp_BETA,cv_Eg_temp)
xlabel('\beta')
ylabel('Coefficient of Variance')
title(['Optimized \beta for GED = ' num2str(var_BETA(min_ind),'%.2f') ', f = ' num2str(f0,'%.1f')
'Hz, \alpha = ' num2str(mean_alpha,'%.2f')])
line(line_var_BETA,line_PED,'Color','r','linestyle','-.')
line(line_var_BETA,line_KED,'Color','k','linestyle',':')
line(line_var_BETA,line_TED,'Color','g','linestyle','--')
% line(line_min_x, line_min_y,'Color','r')
legend('GED','PED','KED','TED','Location','NorthWest')
ylim([min_cv_Eg - 0.1 cv_Ep + 0.1])

% saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
'_Uniform_Abs_Beta_Test.fig']);
end

%%
% Receiver placement and values
switch stats
    case 0                          %statistics are not chosen
        r = 6;  %number of receivers to use
        [rec_num receivers] =
getRec_Loc(random_receiver_locations,x0,y0,z0,Lx,Ly,Lz,set_dmin_mic_src,dmin_mic_mic,dmin_mic_wal
l,r); %Generates receiver location(s) according to which type is selected
        %Find the energetic (linear) values of each receiver in the room (receiver number varies
from 1-6)
        [PED_receivers_val,KED_receivers_val,TED_receivers_val,GED_receivers_val] =
getRecsVal(rec_num,receivers,Lx,Ly,Lz,Ep,Ek,Et,Eg);
        %Find the energetic (linear) mean value of the combination of receiver(s) in the room
        mean_PED_rec = mean(PED_receivers_val);
        mean_KED_rec = mean(KED_receivers_val);
        mean_TED_rec = mean(TED_receivers_val);
        mean_GED_rec = mean(GED_receivers_val);
        %Put the energetic mean values in dB
        dB_mean_PED_rec = 10*log10(mean_PED_rec/Epref);
```

```
        dB_mean_KED_rec = 10*log10(mean_KED_rec/Ekref);
        dB_mean_TED_rec = 10*log10(mean_TED_rec/Etref);
        dB_mean_GED_rec = 10*log10(mean_GED_rec/Egref);
        %Find the dB values of each receiver in the room (receiver number varies from 1-6)
        [PED_receivers_dB_val,KED_receivers_dB_val,TED_receivers_dB_val,GED_receivers_dB_val] =
        getRecsVal(rec_num,receivers,Lx,Ly,Lz,dB_Ep,dB_Ek,dB_Et,dB_Eg);
        %Find the dB mean value of the combination of receiver(s) in the room
        dBfield_mean_PED_rec = mean(PED_receivers_dB_val);
        dBfield_mean_KED_rec = mean(KED_receivers_dB_val);
        dBfield_mean_TED_rec = mean(TED_receivers_dB_val);
        dBfield_mean_GED_rec = mean(GED_receivers_dB_val);
        %Find the dB standard deviation of the combination of receiver(s) in the
        %room (this is in agreement with ISO standard 3741 Section 8.4.2.2)
        dB_sd_PED_rec = std(PED_receivers_dB_val);
        dB_sd_KED_rec = std(KED_receivers_dB_val);
        dB_sd_TED_rec = std(TED_receivers_dB_val);
        dB_sd_GED_rec = std(GED_receivers_dB_val);
    case 1                              %statistics are chosen
        set_dmin_mic_src = 1.64; %Guarantees to always place the receivers in the far field for
        all frequencies of 100 Hz and above (far field: kr=3 or above)
        alpha_formulation = 1; %(1: Eyring Formulation (accurate for all absorption values); 2:
        Sabine Formulation (only accurate for low absorption values))
        mean_alpha = (2*Ly*Lz*alpha(1) + 2*Lx*Lz*alpha(2) + 2* Lx*Ly*alpha(3))/S;
        if alpha_formulation == 1
            R = (-S*log(1 - mean_alpha))/(1-mean_alpha);    %Eyring Formulation (accurate for all
        absorption values)
        elseif alpha_formulation == 2
            R = (mean_alpha*S)/(1-mean_alpha);    %Sabine Formulation (only accurate for low
        absorption values)
        end
        tao = (4*V)/(C*A);
        insitu_factor = sqrt((2*pi*C*tao)/(k0^2*V));
        Power_insitu = Power*(1 - insitu_factor);
        Power_insitu_dB = 10*log10(Power_insitu/Powref);
        Power_insitu_dB = abs(Power_insitu_dB);
        % Find the energetic (linear) values of each receiver in the room (receiver number varies
        from 1-6)
        num_runs = 5000;                %number of runs
        max_rec = 6;                    %maximum number of receivers used in the analysis
        dB_mean_PED_rec = zeros(num_runs, max_rec);
        dB_mean_KED_rec = zeros(num_runs, max_rec);
        dB_mean_TED_rec = zeros(num_runs, max_rec);
        dB_mean_GED_rec = zeros(num_runs, max_rec);
        dB_sd_PED_rec = zeros(num_runs, max_rec);
        dB_sd_KED_rec = zeros(num_runs, max_rec);
        dB_sd_TED_rec = zeros(num_runs, max_rec);
        dB_sd_GED_rec = zeros(num_runs, max_rec);
        mean_Power_PED_est = zeros(num_runs, max_rec);
        mean_Power_KED_est = zeros(num_runs, max_rec);
        mean_Power_TED_est = zeros(num_runs, max_rec);
        mean_Power_GED_est = zeros(num_runs, max_rec);
        if alpha_test == 1
            alpha_start = 0.01;
            alpha_step = 0.01;
            alpha_finish = 0.99;
            alpha_test_PED_matrix =
            zeros(num_runs,max_rec,length(alpha_start:alpha_step:alpha_finish));
            alpha_test_KED_matrix =
            zeros(num_runs,max_rec,length(alpha_start:alpha_step:alpha_finish));
            alpha_test_TED_matrix =
            zeros(num_runs,max_rec,length(alpha_start:alpha_step:alpha_finish));
            alpha_test_GED_matrix =
            zeros(num_runs,max_rec,length(alpha_start:alpha_step:alpha_finish));
        end
        for h = 1:num_runs          %the number of runs
          for r = 1:max_rec         %the range of receivers used
            [rec_num receivers] =
getRec_Loc(random_receiver_locations,x0,y0,z0,Lx,Ly,Lz,set_dmin_mic_src,dmin_mic_mic,dmin_mic_wal
            l,r); %Generates receiver location(s) according to which type is selected
            [PED_receivers_val,KED_receivers_val,TED_receivers_val,GED_receivers_val] =
            getRecsVal(rec_num,receivers,Lx,Ly,Lz,Ep,Ek,Et,Eg);
```

```
    %Find the energetic (linear) mean value of the combination of receiver(s) in the room
    mean_PED_rec = mean(PED_receivers_val);
    mean_KED_rec = mean(KED_receivers_val);
    mean_TED_rec = mean(TED_receivers_val);
    mean_GED_rec = mean(GED_receivers_val);
    %Put the energetic mean values in dB
    dB_mean_PED_rec(h,r) = 10*log10(mean_PED_rec/Epref);
    dB_mean_KED_rec(h,r) = 10*log10(mean_KED_rec/Ekref);
    dB_mean_TED_rec(h,r) = 10*log10(mean_TED_rec/Etref);
    dB_mean_GED_rec(h,r) = 10*log10(mean_GED_rec/Egref);
    %Find the dB values of each receiver in the room (receiver number varies from 1-6)
    [PED_receivers_dB_val,KED_receivers_dB_val,TED_receivers_dB_val,GED_receivers_dB_val]
    = getRecsdBVal(rec_num,receivers,Lx,Ly,Lz,dB_Ep,dB_Ek,dB_Et,dB_Eg);
    %Find the dB standard deviation of the combination of receiver(s) in the
    %room (this is in agreement with ISO standard 3741 Section 8.4.2.2)
    dB_sd_PED_rec(h,r) = std(PED_receivers_dB_val);
    dB_sd_KED_rec(h,r) = std(KED_receivers_dB_val);
    dB_sd_TED_rec(h,r) = std(TED_receivers_dB_val);
    dB_sd_GED_rec(h,r) = std(GED_receivers_dB_val);
    % Sound Power Estimates
    r_receivers = get_r_receivers(receivers,x0,y0,z0);
    mean_Power_PED_est(h,r) =
    mean(PED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
    (4/(R*C))));
    mean_Power_KED_est(h,r) =
    mean(KED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
    (4/(R*C))));
    mean_Power_TED_est(h,r) =
    0.5.*mean(TED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
    (4/(R*C))));
    mean_Power_GED_est(h,r) =
    mean(GED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
   (4/(R*C))));
    if alpha_test == 1
        ctr_alpha = 0;
        for alpha_temp = alpha_start:alpha_step:alpha_finish
            ctr_alpha = ctr_alpha+1;
            if alpha_formulation == 1
                R_temp = (-S*log(1 - alpha_temp))/(1-alpha_temp);    %Eyring Formulation
                (accurate for all absorption values)
            elseif alpha_formulation == 2
                R_temp = (alpha_temp*S)/(1-alpha_temp);    %Sabine Formulation (only
                accurate for low absorption values)
            end
            alpha_test_PED_matrix(h,r,ctr_alpha) =
            mean(PED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
            (4/(R_temp*C))),2);
            alpha_test_KED_matrix(h,r,ctr_alpha) =
            mean(KED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
           (4/(R_temp*C))),2);
            alpha_test_TED_matrix(h,r,ctr_alpha) =
        mean((0.5.*TED_receivers_val)./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C))
            + (4/(R_temp*C))),2);
            alpha_test_GED_matrix(h,r,ctr_alpha) =
            mean(GED_receivers_val./((lambdaF(theta,phi)./(4*pi*(r_receivers.').^2*C)) +
            (4/(R_temp*C))),2);
        end
    end
  end

 end                %end of the two for loops
% Take each energy field's Standard Deviation of all the receiver means recorded for the
many runs
std_of_PED_means = std(dB_mean_PED_rec,0,1);
std_of_KED_means = std(dB_mean_KED_rec,0,1);
std_of_TED_means = std(dB_mean_TED_rec,0,1);
std_of_GED_means = std(dB_mean_GED_rec,0,1);
std_of_means_matrix =
[std_of_PED_means(1,1),std_of_PED_means(1,2),std_of_PED_means(1,3),std_of_PED_means(1,4),
std_of_PED_means(1,5),std_of_PED_means(1,6);
```

```
std_of_KED_means(1,1),std_of_KED_means(1,2),std_of_KED_means(1,3),std_of_KED_means(1,4),s
td_of_KED_means(1,5),std_of_KED_means(1,6);

std_of_TED_means(1,1),std_of_TED_means(1,2),std_of_TED_means(1,3),std_of_TED_means(1,4),s
td_of_TED_means(1,5),std_of_TED_means(1,6);
std_of_GED_means(1,1),std_of_GED_means(1,2),std_of_GED_means(1,3),std_of_GED_means(1,4),s
td_of_GED_means(1,5),std_of_GED_means(1,6)];
 % Take the each energy field's standard deviation of all the standard deviations recorded
for the many runs
 std_of_PED_stds = std(dB_sd_PED_rec,0,1);
 std_of_KED_stds = std(dB_sd_KED_rec,0,1);
 std_of_TED_stds = std(dB_sd_TED_rec,0,1);
 std_of_GED_stds = std(dB_sd_GED_rec,0,1);
 std_of_stds_matrix =
[std_of_PED_stds(1,1),std_of_PED_stds(1,2),std_of_PED_stds(1,3),std_of_PED_stds(1,4),std_
of_PED_stds(1,5),std_of_PED_stds(1,6);
std_of_KED_stds(1,1),std_of_KED_stds(1,2),std_of_KED_stds(1,3),std_of_KED_stds(1,4),std_o
f_KED_stds(1,5),std_of_KED_stds(1,6);
std_of_TED_stds(1,1),std_of_TED_stds(1,2),std_of_TED_stds(1,3),std_of_TED_stds(1,4),std_o
f_TED_stds(1,5),std_of_TED_stds(1,6);
std_of_GED_stds(1,1),std_of_GED_stds(1,2),std_of_GED_stds(1,3),std_of_GED_stds(1,4),std_o
f_GED_stds(1,5),std_of_GED_stds(1,6)];
 % Take the each energy field's mean of all the standard deviations recorded for the many
runs
 mean_of_PED_stds = mean(dB_sd_PED_rec,1);
 mean_of_KED_stds = mean(dB_sd_KED_rec,1);
 mean_of_TED_stds = mean(dB_sd_TED_rec,1);
 mean_of_GED_stds = mean(dB_sd_GED_rec,1);
 mean_of_stds_matrix =
[mean_of_PED_stds(1,1),mean_of_PED_stds(1,2),mean_of_PED_stds(1,3),mean_of_PED_stds(1,4),
mean_of_PED_stds(1,5),mean_of_PED_stds(1,6);
mean_of_KED_stds(1,1),mean_of_KED_stds(1,2),mean_of_KED_stds(1,3),mean_of_KED_stds(1,4),m
ean_of_KED_stds(1,5),mean_of_KED_stds(1,6);
mean_of_TED_stds(1,1),mean_of_TED_stds(1,2),mean_of_TED_stds(1,3),mean_of_TED_stds(1,4),m
ean_of_TED_stds(1,5),mean_of_TED_stds(1,6);
mean_of_GED_stds(1,1),mean_of_GED_stds(1,2),mean_of_GED_stds(1,3),mean_of_GED_stds(1,4),m
ean_of_GED_stds(1,5),mean_of_GED_stds(1,6)];
% Take the mean of all the means recorded for the many runs
 mean_of_PED_means = mean(dB_mean_PED_rec,1);
 mean_of_KED_means = mean(dB_mean_KED_rec,1);
 mean_of_TED_means = mean(dB_mean_TED_rec,1);
 mean_of_GED_means = mean(dB_mean_GED_rec,1);
rec_est =
 [mean_dB_Ep,mean_of_PED_means(1,1),mean_of_PED_means(1,2),mean_of_PED_means(1,3),mean_of_
 PED_means(1,4),mean_of_PED_means(1,5),mean_of_PED_means(1,6);
 mean_dB_Ek,mean_of_KED_means(1,1),mean_of_KED_means(1,2),mean_of_KED_means(1,3),mean_of_K
 ED_means(1,4),mean_of_KED_means(1,5),mean_of_KED_means(1,6);
 mean_dB_Et,mean_of_TED_means(1,1),mean_of_TED_means(1,2),mean_of_TED_means(1,3),mean_of_T
 ED_means(1,4),mean_of_TED_means(1,5),mean_of_TED_means(1,6);
 mean_dB_Eg,mean_of_GED_means(1,1),mean_of_GED_means(1,2),mean_of_GED_means(1,3),mean_of_G
 ED_means(1,4),mean_of_GED_means(1,5),mean_of_GED_means(1,6)];
rec_est_err = [abs(mean_dB_Ep - mean_of_PED_means(1,1)),abs(mean_dB_Ep -
 mean_of_PED_means(1,2)),abs(mean_dB_Ep - mean_of_PED_means(1,3)),abs(mean_dB_Ep -
 mean_of_PED_means(1,4)),abs(mean_dB_Ep - mean_of_PED_means(1,5)),abs(mean_dB_Ep -
 mean_of_PED_means(1,6));
abs(mean_dB_Ek - mean_of_KED_means(1,1)),abs(mean_dB_Ek -
 mean_of_KED_means(1,2)),abs(mean_dB_Ek - mean_of_KED_means(1,3)),abs(mean_dB_Ek -
 mean_of_KED_means(1,4)),abs(mean_dB_Ek - mean_of_KED_means(1,5)),abs(mean_dB_Ek -
 mean_of_KED_means(1,6));
abs(mean_dB_Et - mean_of_TED_means(1,1)),abs(mean_dB_Et -
 mean_of_TED_means(1,2)),abs(mean_dB_Et - mean_of_TED_means(1,3)),abs(mean_dB_Et -
 mean_of_TED_means(1,4)),abs(mean_dB_Et - mean_of_TED_means(1,5)),abs(mean_dB_Et -
 mean_of_TED_means(1,6));
abs(mean_dB_Eg - mean_of_GED_means(1,1)),abs(mean_dB_Eg -
 mean_of_GED_means(1,2)),abs(mean_dB_Eg - mean_of_GED_means(1,3)),abs(mean_dB_Eg -
 mean_of_GED_means(1,4)),abs(mean_dB_Eg - mean_of_GED_means(1,5)),abs(mean_dB_Eg -
 mean_of_GED_means(1,6))];
% Sound Power Results
% Put all Power estimates from each energy field into dB (for 1 - 6 sensors)
 dB_Power_PED_est = 10*log10(mean_Power_PED_est./Powref);
 dB_Power_KED_est = 10*log10(mean_Power_KED_est./Powref);
```

```
dB_Power_TED_est = 10*log10(mean_Power_TED_est./Powref);
dB_Power_GED_est = 10*log10(mean_Power_GED_est./Powref);
% Find the median of each energy field's dB power estimates (for 1 - 6 sensors)
median_of_dB_Power_PED_est = median(dB_Power_PED_est,1);
median_of_dB_Power_KED_est = median(dB_Power_KED_est,1);
median_of_dB_Power_TED_est = median(dB_Power_TED_est,1);
median_of_dB_Power_GED_est = median(dB_Power_GED_est,1);
All_medians_of_dB_Power_est =
cat(1,median_of_dB_Power_PED_est,median_of_dB_Power_KED_est,median_of_dB_Power_TED_est,me
dian_of_dB_Power_GED_est);
% Find the standard deviation of each energy field's dB power estimates (for 1 - 6
sensors)
std_of_dB_Power_PED_est_Powers = std(dB_Power_PED_est,0,1);
std_of_dB_Power_KED_est_Powers = std(dB_Power_KED_est,0,1);
std_of_dB_Power_TED_est_Powers = std(dB_Power_TED_est,0,1);
std_of_dB_Power_GED_est_Powers = std(dB_Power_GED_est,0,1);
All_std_of_dB_Power_est =
cat(1,std_of_dB_Power_PED_est_Powers,std_of_dB_Power_KED_est_Powers,std_of_dB_Power_TED_e
st_Powers,std_of_dB_Power_GED_est_Powers);

set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',14);
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',1);
set(0,'DefaultLineLineWidth',1);
scsz = get(0,'ScreenSize');
%   saveflag = 0;

%%
if alpha_test == 1
   median_alpha_test_PED_matrix = median(alpha_test_PED_matrix,1);
   median_alpha_test_KED_matrix = median(alpha_test_KED_matrix,1);
   median_alpha_test_TED_matrix = median(alpha_test_TED_matrix,1);
   median_alpha_test_GED_matrix = median(alpha_test_GED_matrix,1);
   median_alpha_test_PED_matrix = squeeze(median_alpha_test_PED_matrix);
   median_alpha_test_KED_matrix = squeeze(median_alpha_test_KED_matrix);
   median_alpha_test_TED_matrix = squeeze(median_alpha_test_TED_matrix);
   median_alpha_test_GED_matrix = squeeze(median_alpha_test_GED_matrix);
   dB_median_alpha_test_PED_matrix = 10*log10(median_alpha_test_PED_matrix./Powref);
   dB_median_alpha_test_KED_matrix = 10*log10(median_alpha_test_KED_matrix./Powref);
   dB_median_alpha_test_TED_matrix = 10*log10(median_alpha_test_TED_matrix./Powref);
   dB_median_alpha_test_GED_matrix = 10*log10(median_alpha_test_GED_matrix./Powref);
   alphas_for_plot = alpha_start:alpha_step:alpha_finish;
   actual_alpha_index = find(abs(alphas_for_plot - mean_alpha) == min(abs(alphas_for_plot
   - mean_alpha)));
   error_dB_median_alpha_test_PED_matrix_1 = dB_median_alpha_test_PED_matrix(1,:) -
    median_of_dB_Power_PED_est(1,1);
   error_dB_median_alpha_test_PED_matrix_2 = dB_median_alpha_test_PED_matrix(2,:) -
   median_of_dB_Power_PED_est(1,2);
   error_dB_median_alpha_test_PED_matrix_3 = dB_median_alpha_test_PED_matrix(3,:) -
   median_of_dB_Power_PED_est(1,3);
   error_dB_median_alpha_test_PED_matrix_4 = dB_median_alpha_test_PED_matrix(4,:) -
   median_of_dB_Power_PED_est(1,4);
   error_dB_median_alpha_test_PED_matrix_5 = dB_median_alpha_test_PED_matrix(5,:) -
   median_of_dB_Power_PED_est(1,5);
   error_dB_median_alpha_test_PED_matrix_6 = dB_median_alpha_test_PED_matrix(6,:) -
   median_of_dB_Power_PED_est(1,6);
   error_dB_median_alpha_test_KED_matrix_1 = dB_median_alpha_test_KED_matrix(1,:) -
   median_of_dB_Power_KED_est(1,1);
   error_dB_median_alpha_test_KED_matrix_2 = dB_median_alpha_test_KED_matrix(2,:) -
   median_of_dB_Power_KED_est(1,2);
   error_dB_median_alpha_test_KED_matrix_3 = dB_median_alpha_test_KED_matrix(3,:) -
   median_of_dB_Power_KED_est(1,3);
   error_dB_median_alpha_test_KED_matrix_4 = dB_median_alpha_test_KED_matrix(4,:) -
   median_of_dB_Power_KED_est(1,4);
   error_dB_median_alpha_test_KED_matrix_5 = dB_median_alpha_test_KED_matrix(5,:) -
   median_of_dB_Power_KED_est(1,5);
   error_dB_median_alpha_test_KED_matrix_6 = dB_median_alpha_test_KED_matrix(6,:) -
   median_of_dB_Power_KED_est(1,6);
   error_dB_median_alpha_test_TED_matrix_1 = dB_median_alpha_test_TED_matrix(1,:) -
   median_of_dB_Power_TED_est(1,1);
```

```
    error_dB_median_alpha_test_TED_matrix_2 = dB_median_alpha_test_TED_matrix(2,:) -
     median_of_dB_Power_TED_est(1,2);
    error_dB_median_alpha_test_TED_matrix_3 = dB_median_alpha_test_TED_matrix(3,:) -
     median_of_dB_Power_TED_est(1,3);
    error_dB_median_alpha_test_TED_matrix_4 = dB_median_alpha_test_TED_matrix(4,:) -
     median_of_dB_Power_TED_est(1,4);
    error_dB_median_alpha_test_TED_matrix_5 = dB_median_alpha_test_TED_matrix(5,:) -
     median_of_dB_Power_TED_est(1,5);
    error_dB_median_alpha_test_TED_matrix_6 = dB_median_alpha_test_TED_matrix(6,:) -
     median_of_dB_Power_TED_est(1,6);
    error_dB_median_alpha_test_GED_matrix_1 = dB_median_alpha_test_GED_matrix(1,:) -
     median_of_dB_Power_GED_est(1,1);
    error_dB_median_alpha_test_GED_matrix_2 = dB_median_alpha_test_GED_matrix(2,:) -
     median_of_dB_Power_GED_est(1,2);
    error_dB_median_alpha_test_GED_matrix_3 = dB_median_alpha_test_GED_matrix(3,:) -
     median_of_dB_Power_GED_est(1,3);
    error_dB_median_alpha_test_GED_matrix_4 = dB_median_alpha_test_GED_matrix(4,:) -
     median_of_dB_Power_GED_est(1,4);
    error_dB_median_alpha_test_GED_matrix_5 = dB_median_alpha_test_GED_matrix(5,:) -
     median_of_dB_Power_GED_est(1,5);
    error_dB_median_alpha_test_GED_matrix_6 = dB_median_alpha_test_GED_matrix(6,:) -
     median_of_dB_Power_GED_est(1,6);
    x_line_actual_alpha = [alphas_for_plot(actual_alpha_index)
     alphas_for_plot(actual_alpha_index)];
    y_line_actual_alpha = [min(error_dB_median_alpha_test_PED_matrix_1)-5
     max(error_dB_median_alpha_test_PED_matrix_1)+1];
    playsound

    figure(100)
    subplot(2,2,1)
    set(gcf,'Position',[0, 0, scsz(3), scsz(4)])

plot(alphas_for_plot,error_dB_median_alpha_test_PED_matrix_1,alphas_for_plot,error_dB_med
 ian_alpha_test_PED_matrix_2,alphas_for_plot,error_dB_median_alpha_test_PED_matrix_3,alpha
 s_for_plot,error_dB_median_alpha_test_PED_matrix_4,alphas_for_plot,error_dB_median_alpha_
 test_PED_matrix_5,alphas_for_plot,error_dB_median_alpha_test_PED_matrix_6,'LineWidth',1.5
 )
    xlabel('\alpha')
    ylabel('Power (dB)')
    title(['PED, Error In Estimated Power (f=' num2str(f0,'%.1f') 'Hz)'])
    line(x_line_actual_alpha,y_line_actual_alpha,'Color','r','LineWidth',1.5)
    legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
     Receivers','Actual \alpha','Location','South')
    grid on

    subplot(2,2,2)

 plot(alphas_for_plot,error_dB_median_alpha_test_KED_matrix_1,alphas_for_plot,error_dB_med
 ian_alpha_test_KED_matrix_2,alphas_for_plot,error_dB_median_alpha_test_KED_matrix_3,alpha
 s_for_plot,error_dB_median_alpha_test_KED_matrix_4,alphas_for_plot,error_dB_median_alpha_
 test_KED_matrix_5,alphas_for_plot,error_dB_median_alpha_test_KED_matrix_6,'LineWidth',1.5
 )
    xlabel('\alpha')
    ylabel('Power (dB)')
    title(['KED, Error In Estimated Power (f=' num2str(f0,'%.1f') 'Hz)'])
    line(x_line_actual_alpha,y_line_actual_alpha,'Color','r','LineWidth',1.5)
    legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
 Receivers','Actual \alpha','Location','South')
    grid on

    subplot(2,2,3)

 plot(alphas_for_plot,error_dB_median_alpha_test_TED_matrix_1,alphas_for_plot,error_dB_med
 ian_alpha_test_TED_matrix_2,alphas_for_plot,error_dB_median_alpha_test_TED_matrix_3,alpha
 s_for_plot,error_dB_median_alpha_test_TED_matrix_4,alphas_for_plot,error_dB_median_alpha_
 test_TED_matrix_5,alphas_for_plot,error_dB_median_alpha_test_TED_matrix_6,'LineWidth',1.5
 )
    xlabel('\alpha')
    ylabel('Power (dB)')
    title(['TED, Error In Estimated Power (f=' num2str(f0,'%.1f') 'Hz)'])
    line(x_line_actual_alpha,y_line_actual_alpha,'Color','r','LineWidth',1.5)
```

```
    legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
Receivers','Actual \alpha','Location','South')
    grid on

    subplot(2,2,4)
plot(alphas_for_plot,error_dB_median_alpha_test_GED_matrix_1,alphas_for_plot,error_dB_med
ian_alpha_test_GED_matrix_2,alphas_for_plot,error_dB_median_alpha_test_GED_matrix_3,alpha
s_for_plot,error_dB_median_alpha_test_GED_matrix_4,alphas_for_plot,error_dB_median_alpha_
test_GED_matrix_5,alphas_for_plot,error_dB_median_alpha_test_GED_matrix_6,'LineWidth',1.5
)
    xlabel('\alpha')
    ylabel('Power (dB)')
    title(['GED, Error In Estimated Power (f=' num2str(f0,'%.1f') 'Hz)'])
    line(x_line_actual_alpha,y_line_actual_alpha,'Color','r','LineWidth',1.5)
    legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
Receivers','Actual \alpha','Location','South')
    grid on

    if saveflag == 1
      saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
'_Uniform_Abs_alpha_error_test.fig']);
    end
end
for k = 1:2
     Power_assumption = k;
     if Power_assumption == 1
         Actual_dB_Power = dB_Power;
     end
     if Power_assumption == 2
         Actual_dB_Power = Power_insitu_dB;
     end
     % Find the error (in dB) of each energy field's power estimates (for 1 - 6 sensors)
     All_mean_dB_errors_of_est_Powers = All_medians_of_dB_Power_est - Actual_dB_Power;
     % Plots of power estimate results
     low_est = min(min(All_medians_of_dB_Power_est))-0.5;
     high_est = max(max(All_medians_of_dB_Power_est))+0.5;
     x_line_power = [0.5 4.5];
     y_line_power_actual = [Actual_dB_Power Actual_dB_Power];
     if low_est > Actual_dB_Power
          low_est = Actual_dB_Power - 0.5;
     end
     if high_est < Actual_dB_Power
        high_est = Actual_dB_Power + 0.5;
     end
     figure(70+k)
     subplot(2,1,1)
     set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
     bar(All_medians_of_dB_Power_est)
     set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',18)
     ylabel('Power (dB)')
     ylim([low_est high_est])
     grid on
     if k == 2
         title(['Estimated Sound Power With Adjusted Input Power (Uniform
     Absorption)(Median Over 5k Runs) (\alpha=' num2str(mean_alpha,'%.2f') ', f='
     num2str(f0,'%.1f') 'Hz)'])
     else
         title(['Estimated Sound Power (Uniform Absorption)(Median Over 5k Runs) (\alpha='
     num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])
     end
     line(x_line_power,y_line_power_actual,'Color','r')
     legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
      Receivers','Actual Power','Location','SouthEastOutside')

     figure(70+k)
     subplot(2,2,3)
     bar(All_mean_dB_errors_of_est_Powers)
     set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',18)
     ylabel('Error (dB)')
     grid on
     title('Error of Estimated Power (Over 5k Runs)')
```

```
            subplot(2,2,4)
            bar(All_std_of_dB_Power_est)
            set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',18)
            ylabel('dB')
            grid on
            title('Standard Deviation of Estimated Power (Over 5k Runs)')

           if saveflag == 1
               if k == 1
                   saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
               '_Uniform_Abs_Power_est.fig']);
               end
               if k == 2
                   saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
               '_Uniform_Abs_Power_est_adjusted.fig']);
               end
           end
           if saveflag == 1
              if k == 1
                  dlmwrite([rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
                  '_Uniform_Abs_Power_est_error.txt']',All_mean_dB_errors_of_est_Powers,
                  'delimiter', '\t','newline', 'pc')
              end
              if k == 2
                  dlmwrite([rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
                  '_Uniform_Abs_Power_est_error_adjusted.txt']',All_mean_dB_errors_of_est_Powers,
                  'delimiter', '\t','newline', 'pc')
              end
           end
      end
end          %end of the no stats/stats programming
playsound

%%
lim =
getlim(Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_field_points_lin,Eg_valid_fie
ld_points_lin,Epref,Ekref,Etref,Egref);
if stats == 0                %plots that do not include stats
   plots =
getplots(scsz,x,y,z,lim,BETA,dB_Ep,dB_Ek,dB_Et,dB_Eg,x0,y0,z0,Lx,Ly,Lz,rec_num,receivers,mean_alp
ha,f0,mean_dB_Ep,mean_dB_Ek,mean_dB_Et,mean_dB_Eg,cv_Ep,cv_Ek,cv_Et,cv_Eg,dB_mean_PED_rec,dB_mean
_KED_rec,dB_mean_TED_rec,dB_mean_GED_rec,dB_sd_PED_rec,dB_sd_KED_rec,dB_sd_TED_rec,dB_sd_GED_rec,
Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_field_points_lin,Eg_valid_field_poin
ts_lin,Epref,Ekref,Etref,Egref);
end
if stats == 1                 %plots that include stats
   plots =
getplots_stats(scsz,x,y,z,lim,BETA,dB_Ep,dB_Ek,dB_Et,dB_Eg,x0,y0,z0,Lx,Ly,Lz,mean_alpha,f0,mean_d
B_Ep,mean_dB_Ek,mean_dB_Et,mean_dB_Eg,cv_Ep,cv_Ek,cv_Et,cv_Eg,Ep_valid_field_points_lin,Ek_valid_
field_points_lin,Et_valid_field_points_lin,Eg_valid_field_points_lin,Epref,Ekref,Etref,Egref,rec_
est,rec_est_err,std_of_means_matrix,std_of_stds_matrix,mean_of_stds_matrix);
end
```

get_r_receivers.m

```
function r_receivers = get_r_receivers(receivers,x0,y0,z0)
    r_receivers = zeros(length(receivers(:,1)),1);
    for i = 1:length(receivers(:,1))
    r_receivers(i) = sqrt((receivers(i,1)-x0)^2 + (receivers(i,2)-y0)^2 + (receivers(i,3)-z0)^2);
    end
end
```

getAlpha1.m

```
function alpha1 = getAlpha1(zw1)
```

```
phi = angle(zw1);
w = abs(zw1);
ab = 1./w;
gamma = cos(phi)./w;
sigma = -sin(phi)./w;
alpha1 = 8*gamma.*(1-gamma.*log(1+(2*gamma+1)./(ab.^2))+((gamma.^2 -
sigma.^2)./sigma).*atan(sigma./(ab.^2 + gamma)));
end
```

getAlpha2.m

```
function alpha2 = getAlpha2(zw2)
phi = angle(zw2);
w = abs(zw2);
ab = 1./w;
gamma = cos(phi)./w;
sigma = -sin(phi)./w;
alpha2 = 8*gamma.*(1-gamma.*log(1+(2*gamma+1)./(ab.^2))+((gamma.^2 -
sigma.^2)./sigma).*atan(sigma./(ab.^2 + gamma)));
end
```

getAlpha3.m

```
function alpha3 = getAlpha3(zw3)
phi = angle(zw3);
w = abs(zw3);
ab = 1./w;
gamma = cos(phi)./w;
sigma = -sin(phi)./w;
alpha3 = 8*gamma.*(1-gamma.*log(1+(2*gamma+1)./(ab.^2))+((gamma.^2 -
sigma.^2)./sigma).*atan(sigma./(ab.^2 + gamma)));
end
```

getEg.m

```
function Eg = getEg(Ep,Ek,BETA)
Eg = BETA.*Ep + (1-BETA).*Ek;
end
```

getEk.m

```
function Ek = getEk(Vx,Vy,Vz,rho)
Ek = (rho/2).*(Vx.*conj(Vx) + Vy.*conj(Vy) + Vz.*conj(Vz));
end
```

getEp.m

```
function Ep = getEp(P,rho,C)
Ep = (P.*conj(P))./(2*rho*C^2);
End
```

getEt.m

```
function Et = getEt(Ep,Ek)
Et = Ep + Ek;
end
```

getf_natural.m

```
function f_natural = getf_natural(C,Lx,Ly,Lz)
lx = Lx;
my = Ly;
nz = Lz;
f_eig = zeros(round(lx+1),round(my+1),round(nz+1));
f_eig_col = zeros(round(lx*my*nz),1);
f_natural = zeros(round(lx*my*nz),4);
for l = 0:round(lx)
    for m = 0:round(my)
        for n = 0:round(nz)
            f_eig(l+1,m+1,n+1) = (C/(2*pi)).*sqrt((l.*pi./Lx).^2 + (m.*pi./Ly).^2 +
           (n.*pi./Lz).^2); %calculates all the eigen fr3quencies for the given modes
            f_eig_col = reshape(f_eig,[],1);     %converts the matrix of eigen frequencies into a
             single column vector
            [f_natural,index] = sort(f_eig_col); %Sorts all frequencies in ascending order and
             gives a corresponding vector of indices for those frequencies
            [f_natural(:,2),f_natural(:,3),f_natural(:,4)] = ind2sub([l+1,m+1,n+1],index);
            %builds a matrix with column vectors: sorted frequencies, l index, m index, n index
            f_natural(:,2) = f_natural(:,2)-1;   %subtract 1 from each column to make the 0,0,0
             mode first
            f_natural(:,3) = f_natural(:,3)-1;
            f_natural(:,4) = f_natural(:,4)-1;
        end
    end
end
end
```

getFieldMeans.m

```
function
[mean_Ep,mean_Ek,mean_Et,mean_Eg,Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_fie
ld_points_lin,Eg_valid_field_points_lin] =
getFieldMeans(Nx,Ny,Nz,Lx,Ly,Lz,Ep,Ek,Et,Eg,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0)
red_x = ceil(Nx/Lx);
red_y = ceil(Ny/Ly);
red_z = ceil(Nz/Lz);
Ep_red_field = Ep(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
Ek_red_field = Ek(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
Et_red_field = Et(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
Eg_red_field = Eg(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
Ep_valid_field =
zeros(length(Ep_red_field(:,1,1)),length(Ep_red_field(1,:,1)),length(Ep_red_field(1,1,:)));
Ek_valid_field = Ep_valid_field;
Et_valid_field = Ep_valid_field;
Eg_valid_field = Ep_valid_field;
   for i = 1:length(Ep_red_field(:,1,1))
       for j = 1:length(Ep_red_field(1,:,1))
           for k = 1:length(Ep_red_field(1,1,:))
               if sqrt((((i/length(Ep_red_field(:,1,1)))*(Lx-2*dmin_mic_wall) + dmin_mic_wall) -
               x0).^2 + (((j/length(Ep_red_field(1,:,1)))*(Ly-2*dmin_mic_wall) + dmin_mic_wall) -
              y0).^2 + (((k/length(Ep_red_field(1,1,:)))*(Lz-2*dmin_mic_wall) + dmin_mic_wall) -
               z0).^2) < set_dmin_mic_src
                  Ep_valid_field(i,j,k) = 0;
                  Ek_valid_field(i,j,k) = 0;
                  Et_valid_field(i,j,k) = 0;
                  Eg_valid_field(i,j,k) = 0;
               else
                  Ep_valid_field(i,j,k) = Ep_red_field(i,j,k);
                  Ek_valid_field(i,j,k) = Ek_red_field(i,j,k);
                  Et_valid_field(i,j,k) = Et_red_field(i,j,k);
                  Eg_valid_field(i,j,k) = Eg_red_field(i,j,k);
               end
           end
       end
   end
```

```
Ep_valid_field_lin = reshape(Ep_valid_field,1,[]);
Ek_valid_field_lin = reshape(Ek_valid_field,1,[]);
Et_valid_field_lin = reshape(Et_valid_field,1,[]);
Eg_valid_field_lin = reshape(Eg_valid_field,1,[]);
[~,~,Ep_valid_field_points_lin] = find(Ep_valid_field_lin);
[~,~,Ek_valid_field_points_lin] = find(Ek_valid_field_lin);
[~,~,Et_valid_field_points_lin] = find(Et_valid_field_lin);
[~,~,Eg_valid_field_points_lin] = find(Eg_valid_field_lin);
mean_Ep = mean(Ep_valid_field_points_lin);
mean_Ek = mean(Ek_valid_field_points_lin);
mean_Et = mean(Et_valid_field_points_lin);
mean_Eg = mean(Eg_valid_field_points_lin);
end
```

getINGB.m

```
function Kn = getINGB(N, B, iteration)
% N = 20
% B = -9.152
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Solve N roots for xtanx = -B and xcotx = B
% x* = x - f(x)/f'(ksi) \epsilon x- f(x)/f'(X) = N(x,X),
% x* \epsilon X and N(x,X)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N1 = ceil(N/2);
N2 = floor(N/2);
% if B >= 0
%     B = -B;
% end
if B <= 0
    X1start = [0:1:N1-1]*pi;
    X1end = X1start + min(sqrt(-B),pi/2-0.00001);
    X2start = [0:1:N2-1]*pi + pi/2;
    X2end = X2start + min(sqrt(-B),pi/2-0.00001);
    X1start(1) = min(1,sqrt(-B/1.56)); % to avoid f'(X1start) = 0;
    orderflag = 1;
    X1 = (X1start+X1end)/2;
    X2 = (X2start+X2end)/2;
    for i = 1: iteration
        [X1start, X1end] = getN1Range(X1,X1start,X1end,B, orderflag);
        [X2start, X2end] = getN2Range(X2,X2start,X2end,B, orderflag);
        X1 = (X1start+X1end)/2;
        X2 = (X2start+X2end)/2;
    end
else
    X3start = B+1;
    X3end = B;
    if B<=1
        X2start = [0:1:N1-1]*pi + pi/2;
    else
        X2start = [1:1:N1]*pi + pi/2;
        X4start = B;
        X4end = B-1;
        X4 = (X4start+X4end)/2;
    end
    X2end = X2start - min(sqrt(B),pi/2-0.00001);
    X1start = [1:1:N2]*pi;
    X1end = X1start - min(sqrt(B),pi/2-0.00001);
    orderflag = -1;
    X1 = (X1start+X1end)/2;
    X2 = (X2start+X2end)/2;
    X3 = (X3start+X3end)/2;
    for i = 1: iteration
        [X1start, X1end] = getN1Range(X1,X1start,X1end,B, orderflag);
        [X2start, X2end] = getN2Range(X2,X2start,X2end,B, orderflag);
        [X3start, X3end] = getN3Range(X3,X3start,X3end,B, orderflag);

        X1 = (X1start+X1end)/2;
        X2 = (X2start+X2end)/2;
```

```
        X3 = (X3start+X3end)/2;
        if B>1
        [X4start, X4end] = getN4Range(X4,X4start,X4end,B, orderflag);
        X4 = (X4start+X4end)/2;
        end
    end
    X1 = [1i*X3,X1];
    if B>1
    X2 = [1i*X4,X2];
    end
end
Kn = sort([X1,X2]);
Kn = Kn(1:N);
% Kn = [0,Kn(1:N-1)];
% keyboard
return
function [Xstart,Xend] = getN1Range(x,x1,x2,B,orderflag)
%%%%%%%%%%%%
% orderflag = 1 if B<0 and -1 if B>0
%%%%%%%%%%%%%%%%
    nn1 = x - getf1(x,B)./getfp1(x1);
    nn2 = x - getf1(x,B)./getfp1(x2);
    nn = [nn1;nn2;x;x1;x2];
    f1nn = orderflag*getf1(nn,B);
    [f1nn,ix] = sort(f1nn,1);
    for i = 1:length(x)
        ix1 = find(f1nn(:,i)>0,1);
        nni = nn(ix(:,i),i);
        Xstart(i) = nni(ix1-1);
        Xend(i) = nni(ix1);
    end
%     keyboard
return
function [Xstart,Xend] = getN2Range(x,x1,x2,B,orderflag)
%%%%%%%%%%%%
% orderflag = 1 if B<0 and -1 if B>0
%%%%%%%%%%%%%%%%
    nn1 = x - getf2(x,B)./getfp2(x1);
    nn2 = x - getf2(x,B)./getfp2(x2);
    nn = [nn1;nn2;x;x1;x2];
    f2nn = orderflag*getf2(nn,B);
    [f2nn,ix] = sort(f2nn,1);
    for i = 1:length(x)
        ix1 = find(f2nn(:,i)>0,1);
        nni = nn(ix(:,i),i);
        Xstart(i) = nni(ix1);
        Xend(i) = nni(ix1-1);
    end
return
function [Xstart,Xend] = getN3Range(x,x1,x2,B,orderflag)
%%%%%%%%%%%%
% orderflag = 1 if B<0 and -1 if B>0
%%%%%%%%%%%%%%%%
    nn1 = x - getf3(x,B)./getfp3(x1);
    nn2 = x - getf3(x,B)./getfp3(x2);
    nn = [nn1;nn2;x;x1;x2];
    f2nn = orderflag*getf3(nn,B);
    [f2nn,ix] = sort(f2nn,1);
    for i = 1:length(x)
        ix1 = find(f2nn(:,i)>0,1);
        nni = nn(ix(:,i),i);
        Xstart(i) = nni(ix1);
        Xend(i) = nni(ix1-1);
    end
return
function [Xstart,Xend] = getN4Range(x,x1,x2,B,orderflag)
%%%%%%%%%%%%
% orderflag = 1 if B<0 and -1 if B>0
%%%%%%%%%%%%%%%%
    nn1 = x - getf2(x,B)./getfp4(x1);
    nn2 = x - getf2(x,B)./getfp4(x2);
```

```
    nn = [nn1;nn2;x;x1;x2];
    f2nn = orderflag*getf4(nn,B);
    [f2nn,ix] = sort(f2nn,1);
    for i = 1:length(x)
        ix1 = find(f2nn(:,i)>0,1);
        nni = nn(ix(:,i),i);
        Xstart(i) = nni(ix1);
        Xend(i) = nni(ix1-1);
    end
return
function f1 = getf1(x,B)
    f1 = x.*tan(x)+B;
return

function f2 = getf2(x,B)
    f2 = x.*cot(x)-B;
return
function f3 = getf3(x,B)
    f3 = x.*tanh(x)-B;
return
function f4 = getf4(x,B)
    f4 = x.*coth(x)-B;
return
function f1p = getfp1(x)
f1p = x.*sec(x).^2+tan(x);
return
function f2p = getfp2(x)
f2p = cot(x) - x.*csc(x).^2;
return
function f3p = getfp3(x)
f3p = x.*sech(x).^2+tanh(x);
return
function f4p = getfp4(x)
f4p = coth(x) - x.*csch(x).^2;
return
```

getkeywait.m

```
function ch = getkeywait(m)
% GETKEYWAIT - get a key within a time limit
%   CH = GETKEYWAIT(P) waits for a keypress for a maximum of P seconds. P
%   should be a positive number. CH is a double representing the key
%   pressed key as an ascii number, including backspace (8), space (32),
%   enter (13), etc. If a non-ascii key (Ctrl, Alt, etc.) is pressed, CH
%   will be NaN.
%   If no key is pressed within P seconds, -1 is returned, and if something
%   went wrong during excution 0 is returned.
%
%  See also INPUT,
%           GETKEY (FileExchange)
% tested for Matlab 6.5 and higher
% version 2.1 (jan 2012)
% author : Jos van der Geest
% email  : jos@jasen.nl
% History
% 1.0 (2005) creation
% 2.0 (apr 2009) - expanded error check on input argument, changed return
% values when a non-ascii was pressed (now NaN), or when something went
% wrong (now 0); added comments ; slight change in coding
% 2.1 (jan 2012) - modified a few properties, included check is figure
%                  still exists (after comment on GETKEY on FEX by Andrew).
% check input argument
error(nargchk(1,1,nargin)) ;
if numel(m)~=1 || ~isnumeric(m) || ~isfinite(m) || m <= 0,
    error('Argument should be a single positive number.') ;
end
% set up the timer
tt = timer ;
tt.timerfcn = 'uiresume' ;
```

```
tt.startdelay = m ;
% Set up the figure
% May be the position property should be individually tweaked to avoid visibility
callstr = 'set(gcbf,''Userdata'',double(get(gcbf,''Currentcharacter''))) ; uiresume ' ;
fh = figure(...
    'name','Press a key', ...
    'keypressfcn',callstr, ...
    'windowstyle','modal',...
    'numbertitle','off', ...
    'position',[0 0  1 1],...
    'userdata',-1) ;
try
    % Wait for something to happen or the timer to run out
    start(tt) ;
    uiwait ;
    ch = get(fh,'Userdata') ;
    if isempty(ch), % a non-ascii key was pressed, return a NaN
        ch = NaN ;
    end
catch
    % Something went wrong, return zero.
    ch = 0 ;
end
% clean up the timer ...
stop(tt) ;
delete(tt) ;
% ... and figure
if ishandle(fh)
    delete(fh) ;
end
```

getKn.m

```
function kn = getKn(N,beta,k0,L,type)
    if (nargin < 5)
        type = 1;
    end
    switch type
        case 0
            n = (0:N-1);
            kn = n *pi / L;
        case 1
            a = real(beta)*L/2;
            kn = getINGB(N,a,30)*2/L;
        case 2
            ns = (0:N-1);
            q(1) = sqrt(1i*2*beta*k0*L)/pi/1i;
            q(2:length(ns)) = ns(2:length(ns))-1i*k0*L/pi^2./ns(2:length(ns))*2*beta;
            kn = q*pi/L;
        case 3
            aa = real(beta);
            a = -abs(aa);
            n1 = (0:N/2-1)*pi;
            n2 = (0:N/2-1)*pi;
            A = ones(1,N/2) * (a)*4*L/2;
            maxkL = 1/2*(n1 + ((n1.^2 - A)).^.5);
%            maxkL = n1+ones(1,N/2)*pi/2;
            maxkL2 = 1/4*(pi*ones(1,N/2)+2*n2+(-A*4+(2*n2+ones(1,N/2)*pi).^2).^.5);
%            maxkL2 = n2+ones(1,N/2)*pi;
            Nx = 100000;
            for l = 1:N/2
                if maxkL(l)>n1(l)+pi/2
                    maxl = n1(l)+pi/2;
                else
                    maxl = maxkL(l);
                end
                if maxkL2(l)>n1(l)+pi
                    maxl2 = n1(l)+pi;
                else
```

```
                maxl2 = maxkL2(l);
            end
            minl = (l-1)*pi;
            minl2 = minl+pi/2;
            maxthe = maxl - minl;
            maxthe2 = maxl2 - minl2;
            dth = maxthe/Nx;
            dth2 = maxthe2/Nx;
            theta1(l,:) = (maxl:-dth:minl);
            theta2(l,:) = (maxl2:-dth2:minl2);
        end
        dimThe = size(theta1);
        xtanx = theta1 .* tan(theta1);
        xcotx = theta2 .* cot(theta2);
        hl = ones(dimThe(1),dimThe(2))* (-a)*L/2;
        diff1 = -xtanx + hl;
        diff2 = xcotx + hl;
        n = 1;
        for l = 1 : dimThe(1)
            for m = 2 : dimThe(2)
                if diff1(l,m) >= 0 || m == dimThe(2)
                    if abs(diff1(l,m)) < abs(diff1(l,m-1))
                        kn(n) = theta1(l,m);
                    else
                        kn(n) = theta1(l,m-1);
                    end
                    xtan((n+1)/2) = kn(n)*tan(kn(n));
                    n = n+1;
                    break;
                end
            end
            for m = 2 : dimThe(2)
                if diff2(l,m) >= 0 || m == dimThe(2)
                    if abs(diff2(l,m)) < abs(diff2(l,m-1))
                        kn(n) = theta2(l,m);
                    else
                        kn(n) = theta2(l,m-1);
                    end
                    xcot(n/2) = kn(n)*cot(kn(n));
                    n = n+1;
                    break;
                end
            end
        end
        if aa>0
            kn = (1:N)*pi-kn;
            if aa<1
                kn = [sqrt(3/2*(-5+sqrt(5)*sqrt(9 - 4*aa*L/2))),kn(1:N-1)];
            end
        end
        kn = kn/L*2;
%        kn = horzcat([-kn(1)],kn);
    end
return
```

getlim.m

```
function lim =
getlim(Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_field_points_lin,Eg_valid_fie
ld_points_lin,Epref,Ekref,Etref,Egref)
   offset = 0.9;
   sorted_Ep = sort(Ep_valid_field_points_lin);
   sorted_Ek = sort(Ek_valid_field_points_lin);
   sorted_Et = sort(Et_valid_field_points_lin);
   sorted_Eg = sort(Eg_valid_field_points_lin);
   perc_75_Ep = sorted_Ep(ceil(.75*length(sorted_Ep)));
   perc_25_Ep = sorted_Ep(ceil(.25*length(sorted_Ep)));
   perc_75_Ek = sorted_Ek(ceil(.75*length(sorted_Ek)));
   perc_25_Ek = sorted_Ek(ceil(.25*length(sorted_Ek)));
```

```
   perc_75_Et = sorted_Et(ceil(.75*length(sorted_Et)));
   perc_25_Et = sorted_Et(ceil(.25*length(sorted_Et)));
   perc_75_Eg = sorted_Eg(ceil(.75*length(sorted_Eg)));
   perc_25_Eg = sorted_Eg(ceil(.25*length(sorted_Eg)));
%    lim = max([(1/Epref)*(perc_75_Ep + offset*(perc_75_Ep - perc_25_Ep)) (1/Ekref)*(perc_75_Ek +
offset*(perc_75_Ek - perc_25_Ek)) (1/Etref)*(perc_75_Et + offset*(perc_75_Et - perc_25_Et))
(1/Egref)*(perc_75_Eg + offset*(perc_75_Eg - perc_25_Eg))]);
   lim = max([(perc_75_Ep + offset*(perc_75_Ep - perc_25_Ep)) (perc_75_Ek + offset*(perc_75_Ek -
perc_25_Ek)) (perc_75_Et + offset*(perc_75_Et - perc_25_Et)) (perc_75_Eg + offset*(perc_75_Eg -
perc_25_Eg))]);
end
```

getMA.m

```
function [MA,CNN] = getMA(k0,Kx,Ky,Kz,kr,Bx,By,Bz,Lx,Ly,Lz,ax,ay,az,bx,by,bz,coupleflag)
kxN = length(Kx);
kyN = length(Ky);
kzN = length(Kz);
TN = kxN*kyN*kzN;
psix20 = ones(1,kxN).^2;
psiy20 = ones(1,kyN).^2;
psiz20 = ones(1,kzN).^2;
psix2L = (cos(Kx*Lx) + Bx.*sin(Kx*Lx)).^2;
psiy2L = (cos(Ky*Ly) + By.*sin(Ky*Ly)).^2;
psiz2L = (cos(Kz*Lz) + Bz.*sin(Kz*Lz)).^2;
cos2kxL = cos(2*Kx*+Lx);
cos2kyL = cos(2*Ky*Ly);
cos2kzL = cos(2*Kz*Lz);
sin2kxL = sin(2*Kx*Lx);
sin2kyL = sin(2*Ky*Ly);
sin2kzL = sin(2*Kz*Lz);
psix0 = 1;
psiy0 = 1;
psiz0 = 1;
PsixL = cos(Kx*Lx) + Bx.*sin(Kx*Lx);
PsiyL = cos(Ky*Ly) + By.*sin(Ky*Ly);
PsizL = cos(Kz*Lz) + Bz.*sin(Kz*Lz);
Cnnx = (2*Kx.*(-ax+(ax^2+Kx.^2).*Lx + ax*cos2kxL)+(-ax^2+Kx.^2).*sin2kxL)./Kx.^3/4;
Cnny = (2*Ky.*(-ay+(ay^2+Ky.^2).*Ly + ay*cos2kyL)+(-ay^2+Ky.^2).*sin2kyL)./Ky.^3/4;
Cnnz = (2*Kz.*(-az+(az^2+Kz.^2).*Lz + az*cos2kzL)+(-az^2+Kz.^2).*sin2kzL)./Kz.^3/4;
Cnnx(isnan(Cnnx)) = Lx;
Cnny(isnan(Cnny)) = Ly;
Cnnz(isnan(Cnnz)) = Lz;
k2xy = transpose(Kx.^2)*ones(1,kyN)+ones(kxN,1)*Ky.^2;
dkN2 = zeros(kxN,kyN,kzN);
if coupleflag == 1
    [KX,KY,KZ] = vectorK(Kx,Ky,Kz);
    MA = sparse(1:TN,1:TN,(k0^2 - KX.^2 - KY.^2 - KZ.^2),TN,TN);
%     DMN1 = zeros(1,TN);
%     DMN2 = zeros(1,TN);
%     DMN3 = zeros(1,TN);
    Cnnxy = Cnnx.'*Cnny;
    Cnnyz = Cnny.'*Cnnz;
    Cnnzx = Cnnz.'*Cnnx;
    for l = 1:kxN
        for m = 1:kyN
            cnnxy = Cnnxy(l,m);
            indexbase = l+kxN*(m-1);
            for n1 = 1:kzN
                ind1 = indexbase+kxN*kyN*(n1-1);
                psiL1 = PsizL(n1);
                psi01 = psiz0;
                for n2 = 1:n1-1
                    psiL2 = PsizL(n2);
                    psi02 = psiz0;
                    dmn = cnnxy*(conj(psi01)*psi02 + conj(psiL1)*conj(psiL2));
                    ind2 = indexbase+kxN*kyN*(n2-1);
                    MA(ind1,ind2) = bz*dmn;
                    MA(ind2,ind1) = bz*conj(dmn);
```

```
                end
                dmn = bz*cnnxy*(psiz2L(n1) + psiz20(n1));
%                 DMN1(ind1) = dmn;
                MA(ind1,ind1) = MA(ind1,ind1)*cnnxy*Cnnz(n1)+dmn;
            end
        end
    end
    for n = 1:kzN
        for m = 1:kyN
            cnnyz = Cnnyz(m,n);
            indexbase = kxN*kyN*(n-1)+kxN*(m-1);
            for l1 = 1:kxN
                ind1 = indexbase+l1;
                psiL1 = PsixL(l1);
                psi01 = psix0;
                for l2 = 1:l1-1
                    psiL2 = PsixL(l2);
                    psi02 = psix0;
                    dmn = cnnyz*(conj(psi01)*psi02 + conj(psiL1)*conj(psiL2));
                    ind2 = indexbase+l2;
                    MA(ind1,ind2) = bx*dmn;
                    MA(ind2,ind1) = bx*conj(dmn);
                end
                dmn = bx*cnnyz*(psix2L(l1) + psix20(l1));
%                 DMN2(ind1) = dmn;
                MA(ind1,ind1) = MA(ind1,ind1) + dmn;
            end
        end
    end
    for n = 1:kzN
        for l = 1:kxN
            cnnzx = Cnnzx(n,l);
            indexbase = kxN*kyN*(n-1)+l;
            for m1 = 1:kyN
                ind1 = indexbase+kxN*(m1-1);
                psiL1 = PsiyL(m1);
                psi01 = psiy0;
                for m2 = 1:m1-1
                    psiL2 = PsiyL(m2);
                    psi02 = psiy0;
                    dmn = cnnzx*(conj(psi01)*psi02 + conj(psiL1)*conj(psiL2));
                    ind2 = indexbase+(m2-1)*kxN;
                    MA(ind1,ind2) = by*dmn;
                    MA(ind2,ind1) = by*conj(dmn);
                end
                dmn = by*cnnzx*(psiy2L(m1) + psiy20(m1));
%                 DMN3(ind1) = dmn;
                MA(ind1,ind1) = MA(ind1,ind1) + dmn;
            end
        end
    end
else   %if coupleflag == 0
    MA1xy = transpose(Cnnx)*Cnny;
    MA2xyib = by*transpose(Cnnx)*(psiy2L + psiy20) + bx*transpose(psix2L + psix20)*Cnny; %should
be I not FI

    DNN = zeros(kxN,kyN,kzN);
    CNN = zeros(kxN,kyN,kzN);
    for n = 1:kzN
        dkN2(:,:,n) = k0^2 - k2xy - Kz(n)^2;
        CNN(:,:,n) = MA1xy*Cnnz(n);
        DNN(:,:,n) = bz*MA1xy*(psiz2L(n) + psiz20(n)) + MA2xyib*Cnnz(n);
    end
    MAd = reshape(DNN + CNN.*dkN2,1,[]);
    MA = sparse(1:TN,1:TN,MAd,TN,TN);
end
[CNNX,CNNY,CNNZ] = vectorlize(Cnnx,Cnny,Cnnz);
CNN = CNNX.*CNNY.*CNNZ;
end
```

getMB.m

```
function [MB,A0,B0] = getMB(Kx,Ky,Kz,Bx,By,Bz,Lx,Ly,Lz,beta,k0,u0,x0,y0,z0,greenflag,lambdaF)
if nargin < 17, lambdaF=@(theta,phi) 1;end
kxN = length(Kx);
kyN = length(Ky);
kzN = length(Kz);
I = complex(0,1);
kxm = max(Kx);
kym = max(Ky);
kzm = max(Kz);
if greenflag > 0
    Xresolution = getReso(Lx,kxm,4);
    dx = Lx/Xresolution;
    x = (dx/2:dx:Lx-dx/2);
    Nx = length(x);
    Yresolution = getReso(Ly,kym,4);
    dy = Ly/Yresolution;
    y = (dy/2:dy:Ly-dy/2);
    Ny = length(y);
    Zresolution = getReso(Lz,kzm,4);
    dz = Lz/Zresolution;
    z = (dz/2:dz:Lz-dz/2);
    Nz = length(z);
    rx2 = ((z-z0)'*ones(1,Ny)).^2 + (ones(Nz,1)*(y-y0)).^2;
    ry2 = ((x-x0)'*ones(1,Nz)).^2 + (ones(Nx,1)*(z-z0)).^2;
    rz2 = ((y-y0)'*ones(1,Nx)).^2 + (ones(Ny,1)*(x-x0)).^2;
    rx0 = (x0^2 * ones(Nz,Ny) + rx2).^.5;
    rxL = ((Lx-x0)^2 * ones(Nz,Ny) + rx2).^.5;
    ry0 = (y0^2 * ones(Nx,Nz) + ry2).^.5;
    ryL = ((Ly-y0)^2 * ones(Nx,Nz) + ry2).^.5;
    rz0 = (z0^2 * ones(Ny,Nx) + rz2).^.5;
    rzL = ((Lz-z0)^2 * ones(Ny,Nx) + rz2).^.5;
    thx0 = acos((z-z0)'*ones(1,Ny)./rx0);
    thxL = acos((z-z0)'*ones(1,Ny)./rxL);
    thy0 = acos(ones(Nx,1)*(z-z0)./ry0);
    thyL = acos(ones(Nx,1)*(z-z0)./ryL);
    thz0 = acos(-ones(Ny,Nx)*z0./rz0);
    thzL = acos(ones(Ny,Nx)*(Lz-z0)./rzL);
%%%% not verified!!!!!
    phx0 = pi - atan(-ones(Nz,1)*(y-y0)/x0);
    phxL = atan(ones(Nz,1)*(y-y0)/(Lx-x0));
    phy0 = 2*pi-acos((x-x0)'*ones(1,Nz)./sqrt(y0^2+((x-x0)'*ones(1,Nz)).^2));
    phyL = acos((x-x0)'*ones(1,Nz)./sqrt((Ly-y0)^2+((x-x0)'*ones(1,Nz)).^2));
    phz0 = 0;
    phzL = 0;
    P10x0 = (u0./(rx0).*exp(-I*k0*rx0)/4/pi);
    P10xL = (u0./(rxL).*exp(-I*k0*rxL)/4/pi);
    P10y0 = (u0./(ry0).*exp(-I*k0*ry0)/4/pi);
    P10yL = (u0./(ryL).*exp(-I*k0*ryL)/4/pi);
    P10z0 = (u0./(rz0).*exp(-I*k0*rz0)/4/pi);
    P10zL = (u0./(rzL).*exp(-I*k0*rzL)/4/pi);
    P20x0 = (u0./(rx0).*exp(I*k0*rx0)/4/pi);
    P20xL = (u0./(rxL).*exp(I*k0*rxL)/4/pi);
    P20y0 = (u0./(ry0).*exp(I*k0*ry0)/4/pi);
    P20yL = (u0./(ryL).*exp(I*k0*ryL)/4/pi);
    P20z0 = (u0./(rz0).*exp(I*k0*rz0)/4/pi);
    P20zL = (u0./(rzL).*exp(I*k0*rzL)/4/pi);
    dP10x0 = -P10x0 .*((x0 + I*k0*x0*rx0)./rx0.^2);
    dP10xL = -P10xL .*(((Lx-x0) + I*k0*(Lx-x0)*rxL)./rxL.^2);
    dP10y0 = -P10y0 .*((y0 + I*k0*y0*ry0)./ry0.^2);
    dP10yL = -P10yL .*(((Ly-y0) + I*k0*(Ly-y0)*ryL)./ryL.^2);
    dP10z0 = -P10z0 .*((z0 + I*k0*z0*rz0)./rz0.^2);
    dP10zL = -P10zL .*(((Lz-z0) + I*k0*(Lz-z0)*rzL)./rzL.^2);
    dP20x0 = -P20x0 .*((x0 - I*k0*x0*rx0)./rx0.^2);
    dP20xL = -P20xL .*(((Lx-x0) - I*k0*(Lx-x0)*rxL)./rxL.^2);
    dP20y0 = -P20y0 .*((y0 - I*k0*y0*ry0)./ry0.^2);
    dP20yL = -P20yL .*(((Ly-y0) - I*k0*(Ly-y0)*ryL)./ryL.^2);
    dP20z0 = -P20z0 .*((z0 - I*k0*z0*rz0)./rz0.^2);
```

```
    dP20zL = -P20zL .*(((Lz-z0) - I*k0*(Lz-z0)*rzL)./rzL.^2);
    if greenflag == 2
        sumbetap10 =
       beta(1)*sum(sum(P10x0+P10xL))+beta(2)*sum(sum(P10y0+P10yL))+beta(3)*sum(sum(P10z0+P10zL))
       ;
        sumbetap20 =
       beta(1)*sum(sum(P20x0+P20xL))+beta(2)*sum(sum(P20y0+P20yL))+beta(3)*sum(sum(P20z0+P20zL))
       ;
        sumdp10 = sum(sum(dP10x0+dP10xL))+sum(sum(dP10y0+dP10yL))+sum(sum(dP10z0+dP10zL));
        sumdp20 = sum(sum(dP20x0+dP20xL))+sum(sum(dP20y0+dP20yL))+sum(sum(dP20z0+dP20zL));
        A0 = (sumdp20 - sumbetap20)/(sumbetap10 - sumbetap20 - sumdp10 + sumdp20);
    else
        A0 = 1;
    end
  B0 = 1-A0;
% size(thx0)
% figure();
% plot(thx0);
% hold on;
% plot(thxL);
% figure();
% plot(thy0);
% hold on;
% plot(thyL);
% figure();
% plot(thz0);
% hold on;
% plot(thzL);
    dP0x0 =((A0*P10x0 + B0*P20x0)*beta(1) - A0*dP10x0 - B0*dP20x0).*lambdaF(thx0,phx0);
    dP0xL =((A0*P10xL + B0*P20xL)*beta(1) - A0*dP10xL - B0*dP20xL).*lambdaF(thxL,phxL);
    dP0y0 =((A0*P10y0 + B0*P20y0)*beta(2) - A0*dP10y0 - B0*dP20y0).*lambdaF(thy0,phy0);
    dP0yL =((A0*P10yL + B0*P20yL)*beta(2) - A0*dP10yL - B0*dP20yL).*lambdaF(thyL,phyL);
    dP0z0 =((A0*P10z0 + B0*P20z0)*beta(3) - A0*dP10z0 - B0*dP20z0).*lambdaF(thz0,phz0);
    dP0zL =((A0*P10zL + B0*P20zL)*beta(3) - A0*dP10zL - B0*dP20zL).*lambdaF(thzL,phzL);
    dSxy = Lx*Ly/Nx/Ny;
    dSyz = Ly*Lz/Ny/Nz;
    dSxz = Lx*Lz/Nx/Nz;
    Emz0 = zeros(kxN,kyN);
    EmzL = zeros(kxN,kyN);
    Emx0 = zeros(kyN,kzN);
    EmxL = zeros(kyN,kzN);
    Emy0 = zeros(kzN,kxN);
    EmyL = zeros(kzN,kxN);
    for l = 1:kxN
        for m = 1: kyN
            psixy = (cos(Ky(m)*y) + By(m)*sin(Ky(m)*y)).'*(cos(Kx(l)*x) + Bx(l)*sin(Kx(l)*x));
            Emz0(l,m) = -sum(sum(conj(psixy).*dP0z0))*dSxy;
            EmzL(l,m) = -sum(sum(conj(psixy).*dP0zL))*dSxy;
        end
    end
    for l = 1:kyN
        for m = 1: kzN
            psiyz = (cos(Kz(m)*z) + Bz(m)*sin(Kz(m)*z)).'*(cos(Ky(l)*y) + By(l)*sin(Ky(l)*y));
            Emx0(l,m) = -sum(sum(conj(psiyz).*dP0x0))*dSyz;
            EmxL(l,m) = -sum(sum(conj(psiyz).*dP0xL))*dSyz;
        end
    end
    for l = 1:kzN
        for m = 1: kxN
            psizx = (cos(Kx(m)*x) + Bx(m)*sin(Kx(m)*x)).'*(cos(Kz(l)*z) + Bz(l)*sin(Kz(l)*z));
            Emy0(l,m) = -sum(sum(conj(psizx).*dP0y0))*dSxz;
            EmyL(l,m) = -sum(sum(conj(psizx).*dP0yL))*dSxz;
        end
    end
    psix0 = 1;
    psiy0 = 1;
    psiz0 = 1;
    PsixL = cos(Kx*Lx) + Bx.*sin(Kx*Lx);
    PsiyL = cos(Ky*Ly) + By.*sin(Ky*Ly);
    PsizL = cos(Kz*Lz) + Bz.*sin(Kz*Lz);
    MB = zeros(kxN,kyN,kzN);
```

```
    for l = 1:kxN
        psixL = conj(PsixL(l));
        for m = 1:kyN
            psiyL = conj(PsiyL(m));
            for n = 1:kzN
                psizL = conj(PsizL(n));
                MB(l,m,n) = psix0*Emx0(m,n) + psixL*EmxL(m,n) + psiy0*Emy0(n,l) +
                psiyL*EmyL(n,l)+ psiz0*Emz0(l,m) + psizL*EmzL(l,m);
            end
        end
    end
MB = reshape(MB,[],1);
else
    psixx0 = cos(Kx*x0) + Bx.*sin(Kx*x0);
    psiyy0 = cos(Ky*y0) + By.*sin(Ky*y0);
    psizz0 = cos(Kz*z0) + Bz.*sin(Kz*z0);
%     [pxL,pyL,pzL]= vectorlize(psixx0,psiyy0,psizz0);
%     MB = u0*(pxL.*pyL.*pzL).';
    MB = zeros(kxN,kyN,kzN);
    MBxy = u0*transpose(psixx0)*psiyy0;
    for n = 1:kzN
        MB(:,:,n) = MBxy*psizz0(n);
    end
  MB = reshape(MB,[],1);
    A0 = 0;
    B0 = 0;
end
end
```

getmean_Ep.m

```
function mean_Ep = getmean_Ep(Nx,Ny,Nz,Lx,Ly,Lz,Ep,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0)
red_x = ceil(Nx/Lx);
red_y = ceil(Ny/Ly);
red_z = ceil(Nz/Lz);
Ep_red_field = Ep(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
Ep_valid_field =
zeros(length(Ep_red_field(:,1,1)),length(Ep_red_field(1,:,1)),length(Ep_red_field(1,1,:)));
   for i = 1:length(Ep_red_field(:,1,1))
       for j = 1:length(Ep_red_field(1,:,1))
           for k = 1:length(Ep_red_field(1,1,:))
               if sqrt((((i/length(Ep_red_field(:,1,1)))*(Lx-2*dmin_mic_wall) + dmin_mic_wall) -
               x0).^2 + (((j/length(Ep_red_field(1,:,1)))*(Ly-2*dmin_mic_wall) + dmin_mic_wall) -
               y0).^2 + (((k/length(Ep_red_field(1,1,:)))*(Lz-2*dmin_mic_wall) + dmin_mic_wall) -
               z0).^2) < set_dmin_mic_src
                  Ep_valid_field(i,j,k) = 0;
               else
                  Ep_valid_field(i,j,k) = Ep_red_field(i,j,k);
               end
           end
       end
   end
Ep_valid_field_lin = reshape(Ep_valid_field,1,[]);
[~,~,Ep_valid_field_points_lin] = find(Ep_valid_field_lin);
mean_Ep = mean(Ep_valid_field_points_lin);
end
```

getMean_GE.m

```
function [ISO_ind,mean_Eg,Eg_valid_field_points_lin] =
getMean_GE(Nx,Ny,Nz,Lx,Ly,Lz,Eg,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0,ISO_ind)
red_x = ceil(Nx/Lx);
red_y = ceil(Ny/Ly);
red_z = ceil(Nz/Lz);
Eg_red_field = Eg(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
if ISO_ind == 1
```

```
   Eg_valid_field =
zeros(length(Eg_red_field(:,1,1)),length(Eg_red_field(1,:,1)),length(Eg_red_field(1,1,:)));
   for i = 1:length(Eg_red_field(:,1,1))
       for j = 1:length(Eg_red_field(1,:,1))
           for k = 1:length(Eg_red_field(1,1,:))
               if sqrt((((i/length(Eg_red_field(:,1,1)))*(Lx-2*dmin_mic_wall) + dmin_mic_wall) -
               x0).^2 + (((j/length(Eg_red_field(1,:,1)))*(Ly-2*dmin_mic_wall) + dmin_mic_wall) -
               y0).^2 + (((k/length(Eg_red_field(1,1,:)))*(Lz-2*dmin_mic_wall) + dmin_mic_wall) -
               z0).^2) < set_dmin_mic_src
                  Eg_valid_field(i,j,k) = 0;
               else
                  Eg_valid_field(i,j,k) = Eg_red_field(i,j,k);
               end
           end
       end
   end
   ISO_ind = find(Eg_valid_field == 0);
   Eg_valid_field(ISO_ind) = [];
   Eg_valid_field_points_lin = reshape(Eg_valid_field,1,[]);
   mean_Eg = mean(Eg_valid_field);
else
   Eg_valid_field = Eg_red_field;
   Eg_valid_field(ISO_ind) = [];
   Eg_valid_field_points_lin = reshape(Eg_valid_field,1,[]);
   mean_Eg = mean(Eg_valid_field);
end
end
```

getMean_PE.m

```
function [ISO_ind,mean_Ep] =
getMean_PE(Nx,Ny,Nz,Lx,Ly,Lz,Ep,dmin_mic_wall,set_dmin_mic_src,x0,y0,z0,ctr,ISO_ind)
red_x = ceil(Nx/Lx);
red_y = ceil(Ny/Ly);
red_z = ceil(Nz/Lz);
Ep_red_field = Ep(red_x:Nx-red_x,red_y:Ny-red_y,red_z:Nz-red_z);
if ctr == 2
   Ep_valid_field =
zeros(length(Ep_red_field(:,1,1)),length(Ep_red_field(1,:,1)),length(Ep_red_field(1,1,:)));
   for i = 1:length(Ep_red_field(:,1,1))
       for j = 1:length(Ep_red_field(1,:,1))
           for k = 1:length(Ep_red_field(1,1,:))
               if sqrt((((i/length(Ep_red_field(:,1,1)))*(Lx-2*dmin_mic_wall) + dmin_mic_wall) -
               x0).^2 + (((j/length(Ep_red_field(1,:,1)))*(Ly-2*dmin_mic_wall) + dmin_mic_wall) -
               y0).^2 + (((k/length(Ep_red_field(1,1,:)))*(Lz-2*dmin_mic_wall) + dmin_mic_wall) -
               z0).^2) < set_dmin_mic_src
                  Ep_valid_field(i,j,k) = 0;
               else
                  Ep_valid_field(i,j,k) = Ep_red_field(i,j,k);
               end
           end
       end
   end
   ISO_ind = find(Ep_valid_field == 0);
   Ep_valid_field(ISO_ind) = [];
%    Ep_valid_field_lin = reshape(Ep_valid_field,1,[]);
   mean_Ep = mean(Ep_valid_field);
else
   Ep_valid_field = Ep_red_field;
   Ep_valid_field(ISO_ind) = [];
%    Ep_valid_field_lin = reshape(Ep_valid_field,1,[]);
   mean_Ep = mean(Ep_valid_field);
end
end
```

getmode.m

```
function mode = getmode(f0,f_natural)
    [~,in] = min((abs(f_natural(:,1)-f0)));
    mode = f_natural(in,2:end);
end
```

getplots.m

```
function plots =
getplots(scsz,x,y,z,lim,BETA,dB_Ep,dB_Ek,dB_Et,dB_Eg,x0,y0,z0,Lx,Ly,Lz,rec_num,receivers,mean_alp
ha,f0,mean_dB_Ep,mean_dB_Ek,mean_dB_Et,mean_dB_Eg,cv_Ep,cv_Ek,cv_Et,cv_Eg,dB_mean_PED_rec,dB_mean
_KED_rec,dB_mean_TED_rec,dB_mean_GED_rec,dB_sd_PED_rec,dB_sd_KED_rec,dB_sd_TED_rec,dB_sd_GED_rec,
Ep_valid_field_points_lin,Ek_valid_field_points_lin,Et_valid_field_points_lin,Eg_valid_field_poin
ts_lin,Epref,Ekref,Etref,Egref)
    planes = input('Which planes do you want to graph? [0=none, 1=xy, 2=xz, 3=yz, 4=all] ');
    set(0,'DefaultAxesFontName','Times New Roman');
    set(0,'DefaultAxesFontSize',12);
    set(0,'DefaultAxesFontWeight','demi');
    set(0,'DefaultAxesLineWidth',1);
    set(0,'DefaultLineLineWidth',1);
    set(0,'DefaultLineMarkerSize',8);
    left = round(scsz(4)/27);%how far left to start the left-most graphs (0 = all the way to
                             left)
                             %27 in denominator for regular and widescreen, 8
                             %for Gateway, 0.8 for regular double screen, 0.565 for wide/regular
                             double screen (set left2 to 0.84)
    left2 = scsz(4)-200;     %0.835 if using a double screen where the left one is wide screen
                             and the right is regular (set to 1 in all other cases)
    up_height = 25;          %how high up to start the upper graphs (0 = the bottom of the
                             screen)
    low_height = 55;         %how high up to start the lower graphs (0 = the bottom of the
                             screen)
    w_comp = 1.2;            %plot size width compression (higher values make it more
                             compressed)
                             %1.6 for regular and widescreen, 1.2 for Gateway
    h_comp = 2.6;            %plot size height compression (higher values make it more
                             compressed)
                             %2.5 for regular and widescreen, 2.8 for Gateway
    src_color = 'w';         %source marker color
    src_size = 14;           %source marker size
    src_line_width = 2.5;    %source marker line width
    rec_color = 'w';         %receiver marker color
    rec_size = 10;           %receiver marker size
    rec_line_width = 2.5;    %receiver marker line width
    caxis_font_size = 15;    %font size of color bar label
            switch planes
                case 0
                case 1                          %xy plane
                    figure(1)
                    set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
                    subplot(2,2,1)
%                     set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                    scsz(4)/h_comp]);
                    pcolor(x,y,squeeze(dB_Ep(:,:,round(length(z).*z0/Lz))).');
                    colorbar
                    caxis auto
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
                       h(s) = plot3(receivers(s,1),receivers(s,2),0.003,'o');

                    set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                       hold on
                    end
```

```
                hold on
                h_src = plot3(x0,y0,0.003,'x');

               set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
                title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
                num2str(cv_Ep,'%.3f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
                STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB'])
                xlabel('x (m)')
                ylabel('y (m)')
                h = colorbar;
%                 set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);
                color_1 = caxis;

                subplot(2,2,2)
%                 set(gcf,'Position',[left2, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                pcolor(x,y,squeeze(dB_Ek(:,:,round(length(z).*z0/Lz))).');
                colorbar
                caxis([color_1(1) color_1(2)]);
                shading interp
                axis image
                hold on
                h = zeros(rec_num,1);
                for s = 1:rec_num
                   h(s) = plot3(receivers(s,1),receivers(s,2),0.003,'o');

                set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                   hold on
                end
                hold on
                h_src = plot3(x0,y0,0.003,'x');

               set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
                title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                'Hz, F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance='
                num2str(cv_Ek,'%.3f') ' R. Mean=' num2str(dB_mean_KED_rec,'%.1f') 'dB, R.
                STD=' num2str(dB_sd_KED_rec,'%.1f') 'dB'])
                xlabel('x (m)')
                ylabel('y (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);

%                 set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
                subplot(2,2,3)
                pcolor(x,y,squeeze(dB_Et(:,:,round(length(z).*z0/Lz))).');
                colorbar
                caxis([color_1(1) color_1(2)])
                shading interp
                axis image
                hold on
                h = zeros(rec_num,1);
                for s = 1:rec_num
                   h(s) = plot3(receivers(s,1),receivers(s,2),0.003,'o');

                set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                   hold on
                end
                hold on
                h_src = plot3(x0,y0,0.003,'x');

               set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
                title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                'Hz, F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance='
                num2str(cv_Et,'%.3f') ' R. Mean=' num2str(dB_mean_TED_rec,'%.1f') 'dB, R.
                STD=' num2str(dB_sd_TED_rec,'%.1f') 'dB']);
```

```
                    xlabel('x (m)')
                    ylabel('y (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',caxis_font_size);

%                     set(gcf,'Position',[left2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
                    subplot(2,2,4)
                    pcolor(x,y,squeeze(dB_Eg(:,:,round(length(z).*z0/Lz))).');
                    colorbar
                    caxis([color_1(1) color_1(2)])
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
                       h(s) = plot3(receivers(s,1),receivers(s,2),0.003,'o');

                    set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                       hold on
                    end
                    hold on
                    h_src = plot3(x0,y0,0.003,'x');

                   set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                    hold off
                    title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                    'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance='
                    num2str(cv_Eg,'%.3f') ' R. Mean=' num2str(dB_mean_GED_rec,'%.1f') 'dB, R.
                    STD=' num2str(dB_sd_GED_rec,'%.1f') 'dB'])
                    xlabel('x (m)')
                    ylabel('y (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',caxis_font_size);

                    figure(2)
%                     set(gcf,'Position',[scsz(3)/2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                     hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                    ,[0 lim],'extrememode','compress');
%                     hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                    points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                    set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
                    hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                    ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                    lim],'extrememode','compress');
                    hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                    in),mean(Eg_valid_field_points_lin)];
                    hold on
                    plot(hm,'d')
                    title('Box Plots of the Normalized Energetic Field Values')
                    ylabel('Joules')
                case 2                          %xz plane
                    figure(1)
                    set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                    scsz(4)/h_comp]);
                    pcolor(x,z,squeeze(dB_Ep(:,round(length(y).*y0/Ly),:)).');
                    colorbar
                    caxis auto
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
                       h(s) = plot3(receivers(s,1),receivers(s,3),0.003,'o');
```

```
 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(x0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
 hold off
 title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
 'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
 num2str(cv_Ep,'%.2f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
 STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB']);
 xlabel('x (m)')
 ylabel('z (m)')
 h = colorbar;
 set(get(h,'ylabel'),'String', 'dB','FontName','Times New
 Roman','FontWeight','demi','FontSize',caxis_font_size);
 color_2 = caxis;

 figure(2)
 set(gcf,'Position',[left2, scsz(4)/2+up_height, scsz(4)/w_comp,
 scsz(4)/h_comp]);
 pcolor(x,z,squeeze(dB_Ek(:,round(length(y).*y0/Ly),:)).');
 colorbar
 caxis([color_2(1) color_2(2)])
 shading interp
 axis image
 hold on
 h = zeros(rec_num,1);
 for s = 1:rec_num
    h(s) = plot3(receivers(s,1),receivers(s,3),0.003,'o');

 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(x0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
 hold off
 title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
 'Hz, F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance='
 num2str(cv_Ek,'%.2f') ' R. Mean=' num2str(dB_mean_KED_rec,'%.1f') 'dB, R.
 STD=' num2str(dB_sd_KED_rec,'%.1f') 'dB']);
 xlabel('x (m)')
 ylabel('z (m)')
 h = colorbar;
 set(get(h,'ylabel'),'String', 'dB','FontName','Times New
 Roman','FontWeight','demi','FontSize',caxis_font_size);

 figure(3)
 set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
 pcolor(x,z,squeeze(dB_Et(:,round(length(y).*y0/Ly),:)).');
 colorbar
 caxis([color_2(1) color_2(2)])
 shading interp
 axis image
 hold on
 h = zeros(rec_num,1);
 for s = 1:rec_num
    h(s) = plot3(receivers(s,1),receivers(s,3),0.003,'o');

 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(x0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
 hold off
 title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
```

```
                    'Hz, F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance='
                    num2str(cv_Et,'%.2f') ' R. Mean=' num2str(dB_mean_TED_rec,'%.1f') 'dB, R.
                    STD=' num2str(dB_sd_TED_rec,'%.1f') 'dB']);
                    xlabel('x (m)')
                    ylabel('z (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',caxis_font_size);

                    figure(4)
                    set(gcf,'Position',[left2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
                    pcolor(x,z,squeeze(dB_Eg(:,round(length(y).*y0/Ly),:)).');
                    colorbar
                    caxis([color_2(1) color_2(2)])
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
                       h(s) = plot3(receivers(s,1),receivers(s,3),0.003,'o');

                    set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                       hold on
                    end
                    hold on
                    h_src = plot3(x0,z0,0.003,'x');

                   set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                    hold off
                    title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                    'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance='
                    num2str(cv_Eg,'%.2f') ' R. Mean=' num2str(dB_mean_GED_rec,'%.1f') 'dB, R.
                    STD=' num2str(dB_sd_GED_rec,'%.1f') 'dB']);
                    xlabel('x (m)')
                    ylabel('z (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',12);

                    figure(5)
%                    set(gcf,'Position',[scsz(3)/2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                    hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                    ,[0 lim],'extrememode','compress');
%                    hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                    points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                    hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                    ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                    lim],'extrememode','compress');
                    hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                    in),mean(Eg_valid_field_points_lin)];
                    hold on
                    plot(hm,'d')
                    title('Box Plots of the Normalized Energetic Field Values')
                    ylabel('Joules')
                case 3                          %yz plane
                    figure(1)
                    set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                    scsz(4)/h_comp]);
                    pcolor(y,z,squeeze(dB_Ep(round(length(x).*x0/Lx),:,:)).');
                    colorbar
                    caxis auto
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
```

```
    h(s) = plot3(receivers(s,2),receivers(s,3),0.003,'o');

 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(y0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
 hold off
 title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
 'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
 num2str(cv_Ep,'%.2f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
 STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB']);
 xlabel('y (m)')
 ylabel('z (m)')
 h = colorbar;
 set(get(h,'ylabel'),'String', 'dB','FontName','Times New
 Roman','FontWeight','demi','FontSize',caxis_font_size);
 color_3 = caxis;

 figure(2)
 set(gcf,'Position',[left2, scsz(4)/2+up_height, scsz(4)/w_comp,
 scsz(4)/h_comp]);
 pcolor(y,z,squeeze(dB_Ek(round(length(x).*x0/Lx),:,:)).');
 colorbar
 caxis([color_3(1) color_3(2)])
 shading interp
 axis image
 hold on
 h = zeros(rec_num,1);
 for s = 1:rec_num
    h(s) = plot3(receivers(s,2),receivers(s,3),0.003,'o');

 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(y0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
 hold off
 title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
 'Hz, F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance='
 num2str(cv_Ek,'%.2f') ' R. Mean=' num2str(dB_mean_KED_rec,'%.1f') 'dB, R.
 STD=' num2str(dB_sd_KED_rec,'%.1f') 'dB']);
 xlabel('y (m)')
 ylabel('z (m)')
 h = colorbar;
 set(get(h,'ylabel'),'String', 'dB','FontName','Times New
 Roman','FontWeight','demi','FontSize',caxis_font_size);

 figure(3)
 set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
 pcolor(y,z,squeeze(dB_Et(round(length(x).*x0/Lx),:,:)).');
 colorbar
 caxis([color_3(1) color_3(2)])
 shading interp
 axis image
 hold on
 h = zeros(rec_num,1);
 for s = 1:rec_num
    h(s) = plot3(receivers(s,2),receivers(s,3),0.003,'o');

 set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
    hold on
 end
 hold on
 h_src = plot3(y0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
```

```
                    hold off
                    title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                    'Hz, F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance='
                    num2str(cv_Et,'%.2f') ' R. Mean=' num2str(dB_mean_TED_rec,'%.1f') 'dB, R.
                    STD=' num2str(dB_sd_TED_rec,'%.1f') 'dB']);
                    xlabel('y (m)')
                    ylabel('z (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',caxis_font_size);

                    figure(4)
                    set(gcf,'Position',[left2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
                    pcolor(y,z,squeeze(dB_Eg(round(length(x).*x0/Lx),:,:)).');
                    colorbar
                    caxis([color_3(1) color_3(2)])
                    shading interp
                    axis image
                    hold on
                    h = zeros(rec_num,1);
                    for s = 1:rec_num
                       h(s) = plot3(receivers(s,2),receivers(s,3),0.003,'o');

                    set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
                       hold on
                    end
                    hold on
                    h_src = plot3(y0,z0,0.003,'x');

                  set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                    hold off
                    title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                    'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance='
                    num2str(cv_Eg,'%.2f') ' R. Mean=' num2str(dB_mean_GED_rec,'%.1f') 'dB, R.
                    STD=' num2str(dB_sd_GED_rec,'%.1f') 'dB']);
                    xlabel('y (m)')
                    ylabel('z (m)')
                    h = colorbar;
                    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                    Roman','FontWeight','demi','FontSize',caxis_font_size);

                    figure(5)
%                    set(gcf,'Position',[scsz(3)/2, low_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                    hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                    ,[0 lim],'extrememode','compress');
%                    hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                    points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                    hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                    ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                    lim],'extrememode','compress');
                    hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                    in),mean(Eg_valid_field_points_lin)];
                    hold on
                    plot(hm,'d')
                    title('Box Plots of the Normalized Energetic Field Values')
                    ylabel('Joules')
                case 4                          %all planes
                    subplot(2,2,1)
                    set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                    scsz(4)/h_comp]);
%                     set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                    scsz(4)/h_comp]);
                    pcolor(x,y,squeeze(dB_Ep(:,:,round(length(z).*z0/Lz))).');
                    colorbar
                    caxis auto
                    shading interp
```

```
axis image
hold on
h = zeros(rec_num,1);
for s = 1:rec_num
   h(s) = plot3(receivers(s,1),receivers(s,2),0.003,'o');

set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
   hold on
end
hold on
h_src = plot3(x0,y0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
num2str(cv_Ep,'%.2f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB']);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

subplot(2,2,2)
set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
%   set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(x,z,squeeze(dB_Ep(:,round(length(y).*y0/Ly),:)).');
colorbar
caxis auto
shading interp
axis image
hold on
h = zeros(rec_num,1);
for s = 1:rec_num
   h(s) = plot3(receivers(s,1),receivers(s,3),0.003,'o');

set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
   hold on
end
hold on
h_src = plot3(x0,z0,0.003,'x');

set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
num2str(cv_Ep,'%.2f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB']);
xlabel('x (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

subplot(2,2,3)
set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
%   set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Ep(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis auto
shading interp
axis image
hold on
h = zeros(rec_num,1);
for s = 1:rec_num
   h(s) = plot3(receivers(s,2),receivers(s,3),0.003,'o');

set(h(s),'Color',rec_color,'MarkerSize',rec_size,'LineWidth',rec_line_width);
```

```
                    hold on
                end
                hold on
                h_src = plot3(y0,z0,0.003,'x');

               set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
                title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f')
                'Hz, F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance='
                num2str(cv_Ep,'%.2f') ' R. Mean=' num2str(dB_mean_PED_rec,'%.1f') 'dB, R.
                STD=' num2str(dB_sd_PED_rec,'%.1f') 'dB']);
                xlabel('y (m)')
                ylabel('z (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);
            otherwise
                warning('Unexpected plot type. No plot created.');
        end                 %end plane programming
        plots = 1;
end                     %end function
```

getplots_stats.m

```
function plots =
getplots_stats(scsz,x,y,z,lim,BETA,dB_Ep,dB_Ek,dB_Et,dB_Eg,x0,y0,z0,Lx,Ly,Lz,mean_alpha,f0,mean_d
B_Ep,mean_dB_Ek,mean_dB_Et,mean_dB_Eg,cv_Ep,cv_Ek,cv_Et,cv_Eg,Ep_valid_field_points_lin,Ek_valid_
field_points_lin,Et_valid_field_points_lin,Eg_valid_field_points_lin,Epref,Ekref,Etref,Egref,rec_
est,rec_est_err,std_of_means_matrix,std_of_stds_matrix,mean_of_stds_matrix,rootpath_stats);
%     planes = input('Which planes do you want to graph? [0=none, 1=xy, 2=xz, 3=yz, 4=all] ');
    planes = 1;

    set(0,'DefaultAxesFontName','Times New Roman');
    set(0,'DefaultAxesFontSize',14.4);
    set(0,'DefaultAxesFontWeight','demi');
    set(0,'DefaultAxesLineWidth',1);
    set(0,'DefaultLineLineWidth',1);
    set(0,'DefaultLineMarkerSize',8);

    left = round(scsz(4)/0.565);%how far left to start the left-most graphs (0 = all the way to
                            left)
                            %27 in denominator for regular and widescreen, 8
                            %for Gateway, 0.8 for regular double screen, 0.565 for wide/regular
                            double screen (set left2 to 0.84)
    left2 = 0.84;           %0.835 if using a double screen where the left one is wide screen
                            and the right is regular (set to 1 in all other cases)
    up_height = 25;         %how high up to start the upper graphs (0 = the bottom of the
                            screen)
    low_height = 55;        %how high up to start the lower graphs (0 = the bottom of the
                            screen)
    w_comp = 1.65;          %plot size width compression (higher values make it more
                            compressed)
                            %1.6 for regular and widescreen, 1.2 for Gateway
    h_comp = 2.5;           %plot size height compression (higher values make it more
                            compressed)
                            %2.5 for regular and widescreen, 2.8 for Gateway
    src_color = 'w';        %source marker color
    src_size = 14;          %source marker size
    src_line_width = 2.5;   %source marker line width
    caxis_font_size = 15;   %font size of color bar label
        switch planes
            case 0
            case 1                          %xy plane
                figure(1)
                set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
                subplot(2,2,1)
%                 set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                pcolor(x,y,squeeze(dB_Ep(:,:,round(length(z).*z0/Lz))).');
```

```
colorbar
caxis auto
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.3f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

color_1 = caxis;

%   figure(2)
subplot(2,2,2)
%   set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height,
scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(x,y,squeeze(dB_Ek(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)]);
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.3f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

%   figure(3)
subplot(2,2,3)
%   set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(x,y,squeeze(dB_Et(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)])
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.3f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

%   figure(4)
subplot(2,2,4)
%   set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(x,y,squeeze(dB_Eg(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)])
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
```

```
                hold off
%                 title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
                f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
                Variance=' num2str(cv_Eg,'%.3f')]);
                title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
                F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.3f')]);
                xlabel('x (m)')
                ylabel('y (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);

%                 saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
                '_Uniform_Abs_Energy_Fields.fig']);

                figure(5)
                subplot(2,2,1)
                set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
%                 set(gcf,'Position',[left, 0, scsz(3), scsz(4)])
%                 set(gcf,'Position',[1.01*left, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
%                 hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                 ,[0 lim],'extrememode','compress');
%                 hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                lim],'extrememode','compress');
                hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                in),mean(Eg_valid_field_points_lin)];
                hold on
                plot(hm,'d')
                title(['Normalized Energetic Field Values (\alpha=' num2str(mean_alpha,'%.2f') ',
                f=' num2str(f0,'%.1f') 'Hz)'])
                ylabel('Joules')

%                 figure(5)
                subplot(2,2,2)
%                 set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
%                 set(gcf,'Position',[scsz(3)/3 + left, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                bar(rec_est_err)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0 1.1*max(max(rec_est_err))])
                grid on
                title(['Error of Receiver Mean Estimates Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])

%                 figure(5)
                subplot(2,2,3)
%                 set(gcf,'Position',[0, 0, scsz(3), scsz(4)])
%                 set(gcf,'Position',[1.01*left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
                bar(std_of_means_matrix)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0.8*min(min(std_of_means_matrix)) 1.05*max(max(std_of_means_matrix))])
                grid on
                title(['Standard Deviation of Means Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])

%                 saveas(gcf, [rootpath_stats num2str(f0) 'Hz_alpha_' num2str(mean_alpha,'%.2f')
```

```
    '_Uniform_Abs_Energy_Fields_stats.fig']);

case 2                          %xz plane
    figure(1)
    set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
    pcolor(x,z,squeeze(dB_Ep(:,round(length(y).*y0/Ly),:)).');
    colorbar
    caxis auto
    shading interp
    axis image
    hold on
    h_src = plot3(x0,z0,0.003,'x');
    set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
    hold off
    title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
    F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.2f')]);
    xlabel('x (m)')
    ylabel('z (m)')
    h = colorbar;
    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
    Roman','FontWeight','demi','FontSize',caxis_font_size);
    color_2 = caxis;

    figure(2)
    set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height, scsz(4)/w_comp,
    scsz(4)/h_comp]);
    pcolor(x,z,squeeze(dB_Ek(:,round(length(y).*y0/Ly),:)).');
    colorbar
    caxis([color_2(1) color_2(2)])
    shading interp
    axis image
    hold on
    h_src = plot3(x0,z0,0.003,'x');
    set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
    hold off
    title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
    F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.2f')]);
    xlabel('x (m)')
    ylabel('z (m)')
    h = colorbar;
    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
    Roman','FontWeight','demi','FontSize',caxis_font_size);

    figure(3)
    set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
    pcolor(x,z,squeeze(dB_Et(:,round(length(y).*y0/Ly),:)).');
    colorbar
    caxis([color_2(1) color_2(2)])
    shading interp
    axis image
    hold on
    h_src = plot3(x0,z0,0.003,'x');
    set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
    hold off
    title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
    F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.2f')]);
    xlabel('x (m)')
    ylabel('z (m)')
    h = colorbar;
    set(get(h,'ylabel'),'String', 'dB','FontName','Times New
    Roman','FontWeight','demi','FontSize',caxis_font_size);

    figure(4)
    set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
    scsz(4)/h_comp]);
    pcolor(x,z,squeeze(dB_Eg(:,round(length(y).*y0/Ly),:)).');
    colorbar
    caxis([color_2(1) color_2(2)])
    shading interp
    axis image
    hold on
```

```
                h_src = plot3(x0,z0,0.003,'x');
                set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
%                 title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
                f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
                Variance=' num2str(cv_Eg,'%.2f')]);
                title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
                F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.2f')]);
                xlabel('x (m)')
                ylabel('z (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);

                figure(5)
                set(gcf,'Position',[0, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                 hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                ,[0 lim],'extrememode','compress');
%                 hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                lim],'extrememode','compress');
                hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                in),mean(Eg_valid_field_points_lin)];
                hold on
                plot(hm,'d')
                title(['Normalized Energetic Field Values (\alpha=' num2str(mean_alpha,'%.2f') ',
                f=' num2str(f0,'%.1f') 'Hz)'])
                ylabel('Joules')

                figure(6)
                set(gcf,'Position',[scsz(3)/2, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                bar(rec_est_err)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0 1.1*max(max(rec_est_err))])
                grid on
                title(['Error of Receiver Mean Estimates Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])

                figure(7)
                set(gcf,'Position',[0, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
                bar(std_of_means_matrix)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0.8*min(min(std_of_means_matrix)) 1.05*max(max(std_of_means_matrix))])
                grid on
                title(['Standard Deviation of Means Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])
            case 3                          %yz plane
                figure(1)
                set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
                pcolor(y,z,squeeze(dB_Ep(round(length(x).*x0/Lx),:,:)).');
                colorbar
                caxis auto
                shading interp
                axis image
                hold on
                h_src = plot3(y0,z0,0.003,'x');
                set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
```

```
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);
color_3 = caxis;

figure(2)
set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Ek(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(3)
set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Et(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(4)
set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Eg(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
%   title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
Variance=' num2str(cv_Eg,'%.2f')]);
title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);
```

```
                figure(5)
                set(gcf,'Position',[0, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                 hb =
boxplot([Ep_valid_field_points_lin.'./Epref,Ek_valid_field_points_lin.'./Ekref,Et_valid_field_poi
nts_lin.'./Etref,Eg_valid_field_points_lin.'./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                ,[0 lim],'extrememode','compress');
%                 hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                lim],'extrememode','compress');
                hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                in),mean(Eg_valid_field_points_lin)];
                hold on
                plot(hm,'d')
                title(['Normalized Energetic Field Values (\alpha=' num2str(mean_alpha,'%.2f') ',
                f=' num2str(f0,'%.1f') 'Hz)'])
                ylabel('Joules')

                figure(6)
                set(gcf,'Position',[scsz(3)/2, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                bar(rec_est_err)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0 1.1*max(max(rec_est_err))])
                grid on
                title(['Error of Receiver Mean Estimates Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])

                figure(7)
                set(gcf,'Position',[0, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
                bar(std_of_means_matrix)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0.8*min(min(std_of_means_matrix)) 1.05*max(max(std_of_means_matrix))])
                grid on
                title(['Standard Deviation of Means Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])
            case 4                          %all planes
                figure(1)
                set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
                pcolor(x,y,squeeze(dB_Ep(:,:,round(length(z).*z0/Lz))).');
                colorbar
                caxis auto
                shading interp
                axis image
                hold on
                h_src = plot3(x0,y0,0.003,'x');
                set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
                hold off
                title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
                F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.2f')]);
                xlabel('x (m)')
                ylabel('y (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);
                color_1 = caxis;

                figure(2)
                set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
```

```
pcolor(x,y,squeeze(dB_Ek(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)])
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.2f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(3)
set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(x,y,squeeze(dB_Et(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)])
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.2f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(4)
set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(x,y,squeeze(dB_Eg(:,:,round(length(z).*z0/Lz))).');
colorbar
caxis([color_1(1) color_1(2)])
shading interp
axis image
hold on
h_src = plot3(x0,y0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
%   title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
Variance=' num2str(cv_Eg,'%.2f')]);
title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.2f')]);
xlabel('x (m)')
ylabel('y (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(5)
set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(x,z,squeeze(dB_Ep(:,round(length(y).*y0/Ly),:)).');
colorbar
caxis auto
shading interp
axis image
hold on
h_src = plot3(x0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
```

```
F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.2f')]);
xlabel('x (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

color_2 = caxis;

figure(6)
set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(x,z,squeeze(dB_Ek(:,round(length(y).*y0/Ly),:)).');
colorbar
caxis([color_2(1) color_2(2)])
shading interp
axis image
hold on
h_src = plot3(x0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.2f')]);
xlabel('x (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(7)
set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(x,z,squeeze(dB_Et(:,round(length(y).*y0/Ly),:)).');
colorbar
caxis([color_2(1) color_2(2)])
shading interp
axis image
hold on
h_src = plot3(x0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.2f')]);
xlabel('x (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(8)
set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(x,z,squeeze(dB_Eg(:,round(length(y).*y0/Ly),:)).');
colorbar
caxis([color_2(1) color_2(2)])
shading interp
axis image
hold on
h_src = plot3(x0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
%   title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
Variance=' num2str(cv_Eg,'%.2f')]);
title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.2f')]);
xlabel('x (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);
```

```
figure(9)
set(gcf,'Position',[left, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Ep(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis auto
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['PED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ep,'%.1f') 'dB, F. Variance=' num2str(cv_Ep,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

color_3 = caxis;

figure(10)
set(gcf,'Position',[scsz(3)/2 + left2*left, scsz(4)/2+up_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Ek(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['KED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Ek,'%.1f') 'dB, F. Variance=' num2str(cv_Ek,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',caxis_font_size);

figure(11)
set(gcf,'Position',[left, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Et(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
hold off
title(['TED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
F. Mean=' num2str(mean_dB_Et,'%.1f') 'dB, F. Variance=' num2str(cv_Et,'%.2f')]);
xlabel('y (m)')
ylabel('z (m)')
h = colorbar;
set(get(h,'ylabel'),'String', 'dB','FontName','Times New
Roman','FontWeight','demi','FontSize',12);

figure(12)
set(gcf,'Position',[scsz(3)/2 + left2*left, low_height, scsz(4)/w_comp,
scsz(4)/h_comp]);
pcolor(y,z,squeeze(dB_Eg(round(length(x).*x0/Lx),:,:)).');
colorbar
caxis([color_3(1) color_3(2)])
shading interp
axis image
hold on
h_src = plot3(y0,z0,0.003,'x');
set(h_src,'Color',src_color,'MarkerSize',src_size,'LineWidth',src_line_width);
```

```
                hold off
%                 title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', \beta=' num2str(BETA) ',
                f=' num2str(f0,'%.1f') 'Hz, F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F.
                Variance=' num2str(cv_Eg,'%.2f')]);
                title(['GED, \alpha=' num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz,
                F. Mean=' num2str(mean_dB_Eg,'%.1f') 'dB, F. Variance=' num2str(cv_Eg,'%.2f')]);
                xlabel('y (m)')
                ylabel('z (m)')
                h = colorbar;
                set(get(h,'ylabel'),'String', 'dB','FontName','Times New
                Roman','FontWeight','demi','FontSize',caxis_font_size);

                figure(13)
                set(gcf,'Position',[0, scsz(4)/2+up_height, scsz(4)/w_comp, scsz(4)/h_comp]);
%                 hb =
boxplot([Ep_valid_field_points_lin./Epref,Ek_valid_field_points_lin./Ekref,Et_valid_field_poi
nts_lin./Etref,Eg_valid_field_points_lin./Egref],'labels',{'PED','KED','TED','GED'},'datalim'
                ,[0 lim],'extrememode','compress');
%                 hm =
[mean(Ep_valid_field_points_lin)/Epref,mean(Ek_valid_field_points_lin)/Ekref,mean(Et_valid_field_
                points_lin)/Etref,mean(Eg_valid_field_points_lin)/Egref];
                hb =
boxplot([Ep_valid_field_points_lin.',Ek_valid_field_points_lin.',0.5.*Et_valid_field_points_lin.'
                ,Eg_valid_field_points_lin.'],'labels',{'PED','KED','TED','GED'},'datalim',[0
                lim],'extrememode','compress');
                hm =
[mean(Ep_valid_field_points_lin),mean(Ek_valid_field_points_lin),0.5*mean(Et_valid_field_points_l
                in),mean(Eg_valid_field_points_lin)];
                hold on
                plot(hm,'d')
                title(['Normalized Energetic Field Values (\alpha=' num2str(mean_alpha,'%.2f') ',
                f=' num2str(f0,'%.1f') 'Hz)'])
                ylabel('Joules')

                figure(14)
                set(gcf,'Position',[scsz(3)/2, scsz(4)/2+up_height, scsz(4)/w_comp,
                scsz(4)/h_comp]);
                bar(rec_est_err)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0 1.1*max(max(rec_est_err))])
                grid on
                title(['Error of Receiver Mean Estimates Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])

                figure(15)
                set(gcf,'Position',[0, low_height scsz(4)/w_comp, scsz(4)/h_comp]);
                bar(std_of_means_matrix)
                set(gca,'XTickLabel',{'PED','KED','TED','GED'},'FontSize',15)
                legend('1 Receiver','2 Receivers','3 Receivers','4 Receivers','5 Receivers','6
                Receivers')
                ylabel('dB')
                ylim([0.8*min(min(std_of_means_matrix)) 1.05*max(max(std_of_means_matrix))])
                grid on
                title(['Standard Deviation of Means Over 5k Runs (\alpha='
                num2str(mean_alpha,'%.2f') ', f=' num2str(f0,'%.1f') 'Hz)'])
            otherwise
                warning('Unexpected plot type. No plot created.');
        end                 %end of plane programming
    plots = 1;
end                 %end function
```

getRec_Loc.m

```
function [rec_num receivers] =
getRec_Loc(random_receiver_locations,x0,y0,z0,Lx,Ly,Lz,set_dmin_mic_src,dmin_mic_mic,dmin_mic_wal
l,r)
```

```
if random_receiver_locations == -1      %no receivers are included
    receivers = [Lx Ly Lz];    %just removes the receiver marker from the plots
    rec_num = 1;
end
if random_receiver_locations == 0       %receiver locations are chosen by user
    receivers = [1,2,3;    %x, y, z location for each receiver defined by the user (each row is a
                           receiver)
                 2,3,6;
                 4,4,4;
                 3,5,5;
                 2,1,6];
    rec_num = length(receivers(:,1));
end
if random_receiver_locations == 1       %receiver locations are chosen randomly (in accordance
with ISO 3741)
    rec_num = r;                        %number of receivers used
    start_x = ones(rec_num,1).*x0;
    start_y = ones(rec_num,1).*y0;
    start_z = ones(rec_num,1).*z0;
    rec_mat = [start_x start_y start_z];
    switch rec_num
        case 1                          %1 receiver used
            while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
             set_dmin_mic_src
                rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            receivers = [rec_mat(1,1) rec_mat(1,2) rec_mat(1,3)];
        case 2                          %2 receivers used
            while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
            set_dmin_mic_src
                rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            while sqrt((x0-rec_mat(2,1))^2 + (y0-rec_mat(2,2))^2 + (z0-rec_mat(2,3))^2) <
            set_dmin_mic_src || sqrt((rec_mat(2,1)-rec_mat(1,1))^2 + (rec_mat(2,2)-
            rec_mat(1,2))^2 + (rec_mat(2,3)-rec_mat(1,3))^2) < dmin_mic_mic
                rec_mat(2,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(2,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(2,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            receivers = [rec_mat(1,1),rec_mat(1,2),rec_mat(1,3);
                         rec_mat(2,1),rec_mat(2,2),rec_mat(2,3)];
        case 3                          %3 receivers used
            while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
            set_dmin_mic_src
                rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            while sqrt((x0-rec_mat(2,1))^2 + (y0-rec_mat(2,2))^2 + (z0-rec_mat(2,3))^2) <
set_dmin_mic_src || sqrt((rec_mat(2,1)-rec_mat(1,1))^2 + (rec_mat(2,2)-rec_mat(1,2))^2 +
(rec_mat(2,3)-rec_mat(1,3))^2) < dmin_mic_mic
                rec_mat(2,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(2,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(2,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            while sqrt((x0-rec_mat(3,1))^2 + (y0-rec_mat(3,2))^2 + (z0-rec_mat(3,3))^2) <
            set_dmin_mic_src || sqrt((rec_mat(3,1)-rec_mat(2,1))^2 + (rec_mat(3,2)-
            rec_mat(2,2))^2 + (rec_mat(3,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
            sqrt((rec_mat(3,1)-rec_mat(1,1))^2 + (rec_mat(3,2)-rec_mat(1,2))^2 + (rec_mat(3,3)-
            rec_mat(1,3))^2) < dmin_mic_mic
                rec_mat(3,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(3,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(3,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            receivers = [rec_mat(1,1),rec_mat(1,2),rec_mat(1,3);
                         rec_mat(2,1),rec_mat(2,2),rec_mat(2,3);
                         rec_mat(3,1),rec_mat(3,2),rec_mat(3,3)];
```

```
 case 4                          %4 receivers used
    while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
    set_dmin_mic_src
        rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(2,1))^2 + (y0-rec_mat(2,2))^2 + (z0-rec_mat(2,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(2,1)-rec_mat(1,1))^2 + (rec_mat(2,2)-
    rec_mat(1,2))^2 + (rec_mat(2,3)-rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(2,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(2,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(2,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(3,1))^2 + (y0-rec_mat(3,2))^2 + (z0-rec_mat(3,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(3,1)-rec_mat(2,1))^2 + (rec_mat(3,2)-
    rec_mat(2,2))^2 + (rec_mat(3,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
    sqrt((rec_mat(3,1)-rec_mat(1,1))^2 + (rec_mat(3,2)-rec_mat(1,2))^2 + (rec_mat(3,3)-
    rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(3,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(3,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(3,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(4,1))^2 + (y0-rec_mat(4,2))^2 + (z0-rec_mat(4,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(4,1)-rec_mat(3,1))^2 + (rec_mat(4,2)-
    rec_mat(3,2))^2 + (rec_mat(4,3)-rec_mat(3,3))^2) < dmin_mic_mic ||
    sqrt((rec_mat(4,1)-rec_mat(2,1))^2 + (rec_mat(4,2)-rec_mat(2,2))^2 + (rec_mat(4,3)-
    rec_mat(2,3))^2) < dmin_mic_mic || sqrt((rec_mat(4,1)-rec_mat(1,1))^2 +
    (rec_mat(4,2)-rec_mat(1,2))^2 + (rec_mat(4,3)-rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(4,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(4,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(4,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    receivers = [rec_mat(1,1),rec_mat(1,2),rec_mat(1,3);
                 rec_mat(2,1),rec_mat(2,2),rec_mat(2,3);
                 rec_mat(3,1),rec_mat(3,2),rec_mat(3,3);
                 rec_mat(4,1),rec_mat(4,2),rec_mat(4,3)];
case 5                           %5 receivers used
    while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
    set_dmin_mic_src
        rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(2,1))^2 + (y0-rec_mat(2,2))^2 + (z0-rec_mat(2,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(2,1)-rec_mat(1,1))^2 + (rec_mat(2,2)-
    rec_mat(1,2))^2 + (rec_mat(2,3)-rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(2,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(2,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(2,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(3,1))^2 + (y0-rec_mat(3,2))^2 + (z0-rec_mat(3,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(3,1)-rec_mat(2,1))^2 + (rec_mat(3,2)-
    rec_mat(2,2))^2 + (rec_mat(3,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
    sqrt((rec_mat(3,1)-rec_mat(1,1))^2 + (rec_mat(3,2)-rec_mat(1,2))^2 + (rec_mat(3,3)-
    rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(3,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(3,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(3,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(4,1))^2 + (y0-rec_mat(4,2))^2 + (z0-rec_mat(4,3))^2) <
    set_dmin_mic_src || sqrt((rec_mat(4,1)-rec_mat(3,1))^2 + (rec_mat(4,2)-
    rec_mat(3,2))^2 + (rec_mat(4,3)-rec_mat(3,3))^2) < dmin_mic_mic ||
    sqrt((rec_mat(4,1)-rec_mat(2,1))^2 + (rec_mat(4,2)-rec_mat(2,2))^2 + (rec_mat(4,3)-
    rec_mat(2,3))^2) < dmin_mic_mic || sqrt((rec_mat(4,1)-rec_mat(1,1))^2 +
    (rec_mat(4,2)-rec_mat(1,2))^2 + (rec_mat(4,3)-rec_mat(1,3))^2) < dmin_mic_mic
        rec_mat(4,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(4,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
        rec_mat(4,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
    end
    while sqrt((x0-rec_mat(5,1))^2 + (y0-rec_mat(5,2))^2 + (z0-rec_mat(5,3))^2) <
```

```
        set_dmin_mic_src || sqrt((rec_mat(5,1)-rec_mat(4,1))^2 + (rec_mat(5,2)-
        rec_mat(4,2))^2 + (rec_mat(5,3)-rec_mat(4,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(5,1)-rec_mat(3,1))^2 + (rec_mat(5,2)-rec_mat(3,2))^2 + (rec_mat(5,3)-
        rec_mat(3,3))^2) < dmin_mic_mic || sqrt((rec_mat(5,1)-rec_mat(2,1))^2 +
        (rec_mat(5,2)-rec_mat(2,2))^2 + (rec_mat(5,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(5,1)-rec_mat(1,1))^2 + (rec_mat(5,2)-rec_mat(1,2))^2 + (rec_mat(5,3)-
        rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(5,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(5,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(5,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        receivers = [rec_mat(1,1),rec_mat(1,2),rec_mat(1,3);
                     rec_mat(2,1),rec_mat(2,2),rec_mat(2,3);
                     rec_mat(3,1),rec_mat(3,2),rec_mat(3,3);
                     rec_mat(4,1),rec_mat(4,2),rec_mat(4,3);
                     rec_mat(5,1),rec_mat(5,2),rec_mat(5,3)];
    case 6                          %6 receivers used
        while sqrt((x0-rec_mat(1,1))^2 + (y0-rec_mat(1,2))^2 + (z0-rec_mat(1,3))^2) <
        set_dmin_mic_src
            rec_mat(1,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(1,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(1,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        while sqrt((x0-rec_mat(2,1))^2 + (y0-rec_mat(2,2))^2 + (z0-rec_mat(2,3))^2) <
        set_dmin_mic_src || sqrt((rec_mat(2,1)-rec_mat(1,1))^2 + (rec_mat(2,2)-
        rec_mat(1,2))^2 + (rec_mat(2,3)-rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(2,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(2,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(2,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        while sqrt((x0-rec_mat(3,1))^2 + (y0-rec_mat(3,2))^2 + (z0-rec_mat(3,3))^2) <
        set_dmin_mic_src || sqrt((rec_mat(3,1)-rec_mat(2,1))^2 + (rec_mat(3,2)-
        rec_mat(2,2))^2 + (rec_mat(3,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(3,1)-rec_mat(1,1))^2 + (rec_mat(3,2)-rec_mat(1,2))^2 + (rec_mat(3,3)-
        rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(3,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(3,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(3,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        while sqrt((x0-rec_mat(4,1))^2 + (y0-rec_mat(4,2))^2 + (z0-rec_mat(4,3))^2) <
        set_dmin_mic_src || sqrt((rec_mat(4,1)-rec_mat(3,1))^2 + (rec_mat(4,2)-
        rec_mat(3,2))^2 + (rec_mat(4,3)-rec_mat(3,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(4,1)-rec_mat(2,1))^2 + (rec_mat(4,2)-rec_mat(2,2))^2 + (rec_mat(4,3)-
        rec_mat(2,3))^2) < dmin_mic_mic || sqrt((rec_mat(4,1)-rec_mat(1,1))^2 +
        (rec_mat(4,2)-rec_mat(1,2))^2 + (rec_mat(4,3)-rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(4,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(4,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(4,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        while sqrt((x0-rec_mat(5,1))^2 + (y0-rec_mat(5,2))^2 + (z0-rec_mat(5,3))^2) <
        set_dmin_mic_src || sqrt((rec_mat(5,1)-rec_mat(4,1))^2 + (rec_mat(5,2)-
        rec_mat(4,2))^2 + (rec_mat(5,3)-rec_mat(4,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(5,1)-rec_mat(3,1))^2 + (rec_mat(5,2)-rec_mat(3,2))^2 + (rec_mat(5,3)-
        rec_mat(3,3))^2) < dmin_mic_mic || sqrt((rec_mat(5,1)-rec_mat(2,1))^2 +
        (rec_mat(5,2)-rec_mat(2,2))^2 + (rec_mat(5,3)-rec_mat(2,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(5,1)-rec_mat(1,1))^2 + (rec_mat(5,2)-rec_mat(1,2))^2 + (rec_mat(5,3)-
        rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(5,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(5,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
            rec_mat(5,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
        end
        while sqrt((x0-rec_mat(6,1))^2 + (y0-rec_mat(6,2))^2 + (z0-rec_mat(6,3))^2) <
        set_dmin_mic_src || sqrt((rec_mat(6,1)-rec_mat(5,1))^2 + (rec_mat(6,2)-
        rec_mat(5,2))^2 + (rec_mat(6,3)-rec_mat(5,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(6,1)-rec_mat(4,1))^2 + (rec_mat(6,2)-rec_mat(4,2))^2 + (rec_mat(6,3)-
        rec_mat(4,3))^2) < dmin_mic_mic || sqrt((rec_mat(6,1)-rec_mat(3,1))^2 +
        (rec_mat(6,2)-rec_mat(3,2))^2 + (rec_mat(6,3)-rec_mat(3,3))^2) < dmin_mic_mic ||
        sqrt((rec_mat(6,1)-rec_mat(2,1))^2 + (rec_mat(6,2)-rec_mat(2,2))^2 + (rec_mat(6,3)-
        rec_mat(2,3))^2) < dmin_mic_mic ||  sqrt((rec_mat(6,1)-rec_mat(1,1))^2 +
        (rec_mat(6,2)-rec_mat(1,2))^2 + (rec_mat(6,3)-rec_mat(1,3))^2) < dmin_mic_mic
            rec_mat(6,1) = rand(1).*(Lx-2*dmin_mic_wall) + dmin_mic_wall;
```

```
                rec_mat(6,2) = rand(1).*(Ly-2*dmin_mic_wall) + dmin_mic_wall;
                rec_mat(6,3) = rand(1).*(Lz-2*dmin_mic_wall) + dmin_mic_wall;
            end
            receivers = [rec_mat(1,1),rec_mat(1,2),rec_mat(1,3);
                         rec_mat(2,1),rec_mat(2,2),rec_mat(2,3);
                         rec_mat(3,1),rec_mat(3,2),rec_mat(3,3);
                         rec_mat(4,1),rec_mat(4,2),rec_mat(4,3);
                         rec_mat(5,1),rec_mat(5,2),rec_mat(5,3);
                         rec_mat(6,1),rec_mat(6,2),rec_mat(6,3)];
    end       %end of receiver number programming
 end         %end of random receiver programming
end          %end of function
```

getRecsdBVal.m

```
function [PED_receivers_dB_val,KED_receivers_dB_val,TED_receivers_dB_val,GED_receivers_dB_val] =
getRecsdBVal(rec_num,receivers,Lx,Ly,Lz,dB_Ep,dB_Ek,dB_Et,dB_Eg)
  PED_receivers_dB_val = zeros(1,rec_num);
  KED_receivers_dB_val = PED_receivers_dB_val;
  TED_receivers_dB_val = PED_receivers_dB_val;
  GED_receivers_dB_val = PED_receivers_dB_val;
  for i = 1:rec_num
    PED_receivers_dB_val(i) =
dB_Ep(round((receivers(i,1)/Lx)*length(dB_Ep(:,1,1))),round((receivers(i,2)/Ly)*length(dB_Ep(1,:,
1))),round((receivers(i,3)/Lz)*length(dB_Ep(1,1,:))));
    KED_receivers_dB_val(i) =
dB_Ek(round((receivers(i,1)/Lx)*length(dB_Ek(:,1,1))),round((receivers(i,2)/Ly)*length(dB_Ek(1,:,
1))),round((receivers(i,3)/Lz)*length(dB_Ek(1,1,:))));
    TED_receivers_dB_val(i) =
dB_Et(round((receivers(i,1)/Lx)*length(dB_Et(:,1,1))),round((receivers(i,2)/Ly)*length(dB_Et(1,:,
1))),round((receivers(i,3)/Lz)*length(dB_Et(1,1,:))));
    GED_receivers_dB_val(i) =
dB_Eg(round((receivers(i,1)/Lx)*length(dB_Eg(:,1,1))),round((receivers(i,2)/Ly)*length(dB_Eg(1,:,
1))),round((receivers(i,3)/Lz)*length(dB_Eg(1,1,:))));
  end
end
```

getRecsVal.m

```
function [PED_receivers_val,KED_receivers_val,TED_receivers_val,GED_receivers_val] =
getRecsVal(rec_num,receivers,Lx,Ly,Lz,Ep,Ek,Et,Eg)
  PED_receivers_val = zeros(1,rec_num);
  KED_receivers_val = PED_receivers_val;
  TED_receivers_val = PED_receivers_val;
  GED_receivers_val = PED_receivers_val;
  for i = 1:rec_num
    PED_receivers_val(i) =
Ep(round((receivers(i,1)/Lx)*length(Ep(:,1,1))),round((receivers(i,2)/Ly)*length(Ep(1,:,1))),roun
d((receivers(i,3)/Lz)*length(Ep(1,1,:))));
    KED_receivers_val(i) =
Ek(round((receivers(i,1)/Lx)*length(Ek(:,1,1))),round((receivers(i,2)/Ly)*length(Ek(1,:,1))),roun
d((receivers(i,3)/Lz)*length(Ek(1,1,:))));
    TED_receivers_val(i) =
Et(round((receivers(i,1)/Lx)*length(Et(:,1,1))),round((receivers(i,2)/Ly)*length(Et(1,:,1))),roun
d((receivers(i,3)/Lz)*length(Et(1,1,:))));
    GED_receivers_val(i) =
Eg(round((receivers(i,1)/Lx)*length(Eg(:,1,1))),round((receivers(i,2)/Ly)*length(Eg(1,:,1))),roun
d((receivers(i,3)/Lz)*length(Eg(1,1,:))));
  end
end
```

getReso.m

```
function res = getReso(L,kmax,factor)
```

```
res = L*(kmax)/pi*2*factor;
```

getResolutionCap.m

```
function reso = getResolutionCap(fmax,L,C)
    reso = (6*fmax*L)/C/3;
end
```

getSrcLoc.m

```
function reso = getResolutionCap(fmax,L,C)
    reso = (6*fmax*L)/C/3;
end
```

getzw1.m

```
function zw1 = getzw1(w0,rho_w,h_w,S)
    X = w0*rho_w*h_w*S/1000000;
    zw1 = 37 + 1i*X;     %Test cell 2 - alpha @ 80 Hz (alpha = 0.1787)
end
```

getzw2.m

```
function zw2 = getzw2(w0,rho_w,h_w,S)
    X = w0*rho_w*h_w*S/1000000;
    zw2 = 37 + 1i*X;
end
```

getzw3.m

```
function zw3 = getzw3(w0,rho_w,h_w,S)
    X = w0*rho_w*h_w*S/1000000;
    zw3 = 37 + 1i*X;
end
```

playsound.m

```
function playsound
    Fs_temp = 2000;
    Ts_temp = 1/Fs_temp;
    f_temp = 300;
    t_temp = 0:600;
    y_temp = sin(2*pi*f_temp*t_temp*Ts_temp);
    sound(y_temp,Fs_temp)
end
```

playwaitsound.m

```
function playwaitsound
    Fs_temp = 2000;
    Ts_temp = 1/Fs_temp;
    f_temp = 800;
    t_temp = 0:1000;
    y_temp = sin(2*pi*f_temp*t_temp*Ts_temp);
```

```
    sound(y_temp,Fs_temp)
end
```

sumPsiV.m

```
function [PM,VMx,VMy,VMz,progress] = sumPsiV(k0,X,Y,Z,Kx,Ky,Kz,PN,BX,BY,BZ,sumflag,vflag)
I = complex(0,1);
Nx = length(X);
PM = zeros(1,Nx);
VMx = zeros(1,Nx);
VMy = zeros(1,Nx);
VMz = zeros(1,Nx);
if sumflag == 0
    onesx = ones(1,Nx);
    coskx = cos(Kx.'*X);
    cosky = cos(Ky.'*Y);
    coskz = cos(Kz.'*Z);
    sinkx = sin(Kx.'*X);
    sinky = sin(Ky.'*Y);
    sinkz = sin(Kz.'*Z);
    BX = BX.'*onesx;
    BY = BY.'*onesx;
    BZ = BZ.'*onesx;
    psixl = coskx + BX.*sinkx;
    psiym = cosky + BY.*sinky;
    psizn = coskz + BZ.*sinkz;
    ipnk0 = (I*PN/k0).';
%     PM =  PN.'*(psiym .* psixl .* psizn);
    PM =  PN*(psiym .* psixl .* psizn);
    if vflag == 1
%         VMx = ipnk0*((Kx.'*onesx) .* psiym .* (-sinkx + BX .* coskx) .* psizn);
        VMx = ipnk0.'*((Kx.'*onesx) .* psiym .* (-sinkx + BX .* coskx) .* psizn);
%         VMy = ipnk0*((Ky.'*onesx) .* (-sinky + BY .* cosky) .* psixl .* psizn);
        VMy = ipnk0.'*((Ky.'*onesx) .* (-sinky + BY .* cosky) .* psixl .* psizn);
%         VMz = ipnk0*((Kz.'*onesx) .* psiym .* psixl .* (-sinkz + BZ .* coskz));
        VMz = ipnk0.'*((Kz.'*onesx) .* psiym .* psixl .* (-sinkz + BZ .* coskz));
    end
else
    kxN = length(Kx);
    PNdk = PN/k0;
    parfor (l = 1 : kxN)
        kxl = Kx(l);
        kxlx = kxl*X;
        coskxlx = cos(kxlx);
        sinkxlx = sin(kxlx);
        psixl = coskxlx + BX(l)*sinkxlx;
        kym = Ky(l);
        kymy = kym*Y;
        coskymy = cos(kymy);
        sinkymy = sin(kymy);
        psiym = cos(kymy) + BY(l)*sinkymy;
        kzn = Kz(l);
        pn = PN(l);
        pndk = PNdk(l);
        kznz = kzn*Z;
        coskznz = cos(kznz);
        sinkznz = sin(kznz);
        psizn = coskznz + BZ(l)*sinkznz;
        PM = PM + psiym .* psixl .* psizn .* pn;
        if vflag == 1
            VMx = VMx + I*kxl .* psiym .* (-sinkxlx+ BX(l)*coskxlx) .* psizn  * pndk;
            VMy = VMy + I*kym .* (-sinkymy + BY(l)*coskymy) .* psixl .* psizn * pndk;
            VMz = VMz + I*kzn .* psiym .* psixl .* (-sinkznz + BZ(l)*coskznz) * pndk;
        end
     end
end
end
```

vectorK.m

```
function [KX,KY,KZ] = vectorK(kx,ky,kz)
kxN = length(kx);
kyN = length(ky);
kzN = length(kz);
KX = [];
KY = [];
kyk = reshape(ones(kxN,1)*ky,1,[]);
for (k = 1:kzN)
    for j = 1:kyN
        KX = cat(2,KX,kx);
    end
    KY = cat(2,KY,kyk);
end
KZ = reshape(ones(kxN*kyN,1)*kz,1,[]);
end
```

vectorlize.m

```
function [KX,KY,KZ] = vectorlize(kx,ky,kz)
kxN = length(kx);
kyN = length(ky);
kzN = length(kz);
KX = [];
KY = [];
kyk = reshape(ones(kxN,1)*ky,1,[]);
for (k = 1:kzN)
    for j = 1:kyN
        KX = cat(2,KX,kx);
    end
    KY = cat(2,KY,kyk);
end
KZ = reshape(ones(kxN*kyN,1)*kz,1,[]);
end
```

Appendix E

Matlab Code for Chapter 3

MHSE_Worksheet_in_SREs_and_Diffuse_Fields.m

```
%% CALCULATION OF SOUND POWER USING THE MODIFIED HOPKINS-STRYKER EQUATION
%This script processes ED measurements based on the data acquired from
%both the Microflown and G.R.A.S.Probes

clear all; close all; clc;
set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',18);
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',0.01);
set(0,'DefaultLineLineWidth',2.5);
scsz = get(0,'ScreenSize');
format shortG;

%% Define calculation parameters
Source = 1;       %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
absorp_form = 2;  %1:Sabine absorption formulation; 2:Eyring absorption formulation
air_absorp = 1;   %1:air absorption considered; 0:air absorption not considered
%Choose the sensor positions considered in this calculation
ED_pos = [2 6];

%% Define the measurement-specific data
loc_ED = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
Research\Official Thesis Research\OFFICIAL THESIS FILES\ESC C254\ESC C254 ED Tests (6_11_13)\ESC
C254 ED Data\';
num_pos = 6;     %number of measurement positions
%Input the ED receiver positions
r_ED_probes = [2.106  2.627  1.877 1.397  2.175  1.482]          %distance of ED probe to
acoustic center of source
alt_ED_probes = [20  15  0  5  -10  10]                          %altitude of probes (rounded to
nearest 5 degrees)
azimuth_ED_probes = [40 275 0 110 305 255];                       %azimuth angle (take 270 minus
value in the array)(rounded to the nearest 5 degrees)
theta_ED = 90 - alt_ED_probes;
phi_ED = 270 - azimuth_ED_probes

for ii  = 1:length(azimuth_ED_probes)
    if phi_ED(ii) < 0
        phi_ED(ii) = 360 + phi_ED(ii);
    end
end
```

```
theta_ED_ind = (theta_ED+5)/5;
phi_ED_ind = (phi_ED+5)/5;
%Atmospheric Conditions
hr = 16;        %relative humidity (%)
T_C = 23.9;     %ambient atmospheric temperature (C)
Pa_mb = 855;    %ambient atmospheric pressure (millibar)
c = 20.05*sqrt(273 + T_C);   %adiabatic speed of sound
Pa = Pa_mb*0.1;              %ambient atmospheric pressure (kPa)
T = T_C + 273.15;            %ambient atmospheric temperature (K)
[rho,rho_dry] = getRho(Pa_mb,T_C,hr); %density of air

%% Define the room-specific data
Lx = 6.45;                        %dimensions of the room (m)
Ly = 4.844;
Lz = 2.857;
S = 2*Lx*Ly + 2*Lx*Lz + 2*Ly*Lz;  %surface area of walls
V = Lx*Ly*Lz;                     %volume of the room
T_60 = [0.66 0.62 0.72 0.83 0.86 0.80 0.62 0.51 0.42 0.40 0.38 0.40 0.37 0.40 0.38 0.41 0.39 0.37
0.36 0.30 0.27];   %T_10 from EASERA (6/11/13)
A = (55.26/c)*(V./T_60);            %equivalent absorption area of the room (m^2)

%% Define constants
BETA = 1/4;         %the Generalized Energy Density weighting factor (1/4 is associated with
optimum spatial uniformity within the GED field)
pow_ref = 1e-12;    %reference sound power
p_ref = 20e-6;      %reference sound pressure
Pr = 101.325;       %reference ambient atmospheric pressure in kilopascals (1 atm)
T0 = 293.15;        %reference air temperature (K)
T01 = 273.16;       %triple-point isotherm temperature of water (K)
ENBW = 1.5;         %equal noise bandwidth for the Hanning window function
theta0 = 314;
theta1 = 296;
spacing = 0.025;   %spacing between the mics in the 6-mic probe (m)
manual_gamma = ones(1,21);

C1 = -10*log10(Pa/Pr) + 5*log10((273.15 + T_C)/theta0);   %temperature corrections according to
ISO 3741
C2 = -10*log10(Pa/Pr) + 15*log10((273.15 + T_C)/theta1);

%% Load in the data from the anechoic chamber and create 1/3 octave band directivity factors (for
all THREE sources)

theta = (0:5:180)'*pi/180;     %polar angles of the measurement sphere
phi = (0:5:355)*pi/180;        %azimuth angles of the measurement sphere
%Load in the FRF data from each source

if Source == 1    %Dodec
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\Dodec Directivity\Data\';
    load([loc_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 2    %Single Driver
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 3    %Horn
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\Horn Directivity\Data\';
    load([loc_Dir_Fact 'Horn_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
```

```
%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;              %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
[~,summed_weighted_Oct3_H2_sq] = MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h

%% Calculate the 1/3 octave band frequency-dependent room constant, R
%Generate the Total Absorption Coefficient (Eq 1-2.6.7 of Phscs 661 notes)
h = hr*(Pr/Pa)*10^(-6.8346*(T01/T)^(1.261)+4.6151); %the molar concentration of water vapor (%)
fro = (Pa/Pr)*(24+4.04e4*h*((0.02+h)/(0.391+h)));   %the relaxation frequency of oxygen
frN = (Pa/Pr)*(T0/T)^(1/2)*(9+280*h*exp(-4.170*((T0/T)^(1/3)-1))); %the relaxation frequency of
nitrogen
alpha_prop = f_Oct3.^2.*((1.84e-11*(Pr/Pa)*(T/T0)^(1/2)) + (T0/T)^(5/2).*(((0.01275*exp(-
2239.1/T))./(fro+(f_Oct3.^2/fro)))+((0.1068*exp(-3352.0/T))./(frN+(f_Oct3.^2/frN)))));  %the
total absorption due to wave propagation in air
ind_l = find(f_Oct3 == freq_model(1));                %narrowest frequency band indices (limited
by EASERA T_60 frequency range)
ind_h = find(f_Oct3 == freq_model(end));
alpha_prop = alpha_prop(ind_l:ind_h);
f_Oct3 = f_Oct3(ind_l:ind_h);
%Calculate the room constant, R

if absorp_form == 1                                     %Sabine absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = (0.161*V)./(S*T_60);               %the Sabine absorption coefficient of
                                                       %the wall (neglecting air absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S); %the total absorption coefficient in the
                                                        %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant
    end
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = ((0.161*V)./(S*T_60)).' - ((8*alpha_prop*V)/S); %the Sabine absorption
                                                                     %coefficient of the wall
                                                                     %(considering the effects of
                                                                     %air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);               %the total absorption in the
                                                                     %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                   %the 1/3 octave band
                                                                     %frequency-dependent room
                                                                     %constant (including air
                                                                     %absorption)
    end
end
if absorp_form == 2                                     %Eyring absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = 1-exp(-(0.161*V)./(S*T_60));       %The Eyring absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S); %the total absorption in the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant (no air absorption)
    end
```

```
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = 1-exp((1/S)*(8*alpha_prop.'*V-((0.161*V)./T_60))); %the Eyring absorption
                                                                        %coefficient of the wall
                                                                        %(considering air
                                                                        %absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S);                %the total absorption in
                                                                        %the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                      %the 1/3 octave band
                                                                        %frequency-dependent room
                                                                        %constant (including air
                                                                        %absorption)
    end
end

%% Calculate the four acoustic energy density quantities (PED, KED, TED, and GED) in 1/3 octave
bands (from each probe)
%Load in the data
ED_MF_MAT = zeros(4,21,num_pos);
ED_FD_MAT = zeros(4,21,num_pos);
pow_MF_MAT = zeros(4,21,num_pos);
pow_FD_MAT = zeros(4,21,num_pos);

for ii = 1:num_pos
    if Source == 1
        if ii <= 5
            load([loc_ED 'Dodec MF ' num2str(ii) ' FD ' num2str(ii+1) ' ESC C254.mat'])
        end
        if ii == 6
            load([loc_ED 'Dodec MF ' num2str(ii) ' FD ' num2str(ii-5) ' ESC C254.mat'])
        end
    end
    if Source == 2
        if ii <= 5
            load([loc_ED 'SD MF ' num2str(ii) ' FD ' num2str(ii+1) ' ESC C254.mat'])
        end
        if ii == 6
            load([loc_ED 'SD MF ' num2str(ii) ' FD ' num2str(ii-5) ' ESC C254.mat'])
        end
    end
    if Source == 3
        if ii <= 5
            load([loc_ED 'Horn MF ' num2str(ii) ' FD ' num2str(ii+1) ' ESC C254.mat'])
        end
        if ii == 6
            load([loc_ED 'Horn MF ' num2str(ii) ' FD ' num2str(ii-5) ' ESC C254.mat'])
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%MICROFLOWN PROBE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    eval('p_sq_V_MF = MAT_008_Autopower_Pressure(:,2)'';');
    eval('ux_sq_V_MF = MAT_009_Autopower_U_x(:,2)'';');
    eval('uy_sq_V_MF = MAT_010_Autopower_U_y(:,2)'';');
    eval('uz_sq_V_MF = MAT_011_Autopower_U_z(:,2)'';');

    if ii == 1
        eval('f_MF = MAT_008_Autopower_Pressure(:,1)'';');
    end

    clear  MAT_008_Autopower_Pressure  MAT_009_Autopower_U_x
    clear  MAT_010_Autopower_U_y  MAT_011_Autopower_U_z
    MF_Data_Mat = cat(1,p_sq_V_MF/ENBW,ux_sq_V_MF/ENBW,uy_sq_V_MF/ENBW,uz_sq_V_MF/ENBW);
    %Convert the squared voltages to squared pressure and particle velocity components
    [MF_p_sq,MF_U_mag_sq,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,MF_Data_Mat);
    %Convert the narrow band squared pressure to 1/3 octave band squared pressure

    if ii == 1
       [f_MF_Oct3,MF_p_sq_Oct3] = MyOct3Bands(f_MF,MF_p_sq);
       [~,MF_U_mag_sq_Oct3] = MyOct3Bands(f_MF,MF_U_mag_sq);
    else
       [~,MF_p_sq_Oct3] = MyOct3Bands(f_MF,MF_p_sq);
```

```
   [~,MF_U_mag_Oct3] = MyOct3Bands(f_MF,MF_U_mag_sq);
end

%Truncate the 1/3 octave band squared pressure data to go from 100 Hz to 10 kHz

if ii == 1
    ind_l = findnearest(freq_model(1),f_MF_Oct3,0);
    ind_h = findnearest(freq_model(end),f_MF_Oct3,0);
end
if ii == num_pos
    f_MF_Oct3 = f_MF_Oct3(ind_l:ind_h);
end

%Create a matrix of the PED, KED, TED, and GED vectors (1/3 octave bands)
PED_MF = (MF_p_sq_Oct3 / (2*rho*c^2)).';
KED_MF = ((rho/2) * MF_U_mag_sq_Oct3).';
TED_MF = (PED_MF + KED_MF);
GED_MF = (BETA*(PED_MF) + (1-BETA)*KED_MF);
ED_MF_MAT_full = cat(1,PED_MF,KED_MF,TED_MF,GED_MF);
%Truncate the 1/3 octave band ED matrix to the frequency range limited by the EASERA T60
%values
ED_MF_MAT(:,:,ii) = ED_MF_MAT_full(:,ind_l:ind_h);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%6-MIC ED PROBE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Load in the data
eval('p_1_1_FD = MAT_002_Autopower_Mic_1'';');
eval('p_2_2_FD = MAT_003_Autopower_Mic_2'';');
eval('p_3_3_FD = MAT_004_Autopower_Mic_3'';');
eval('p_4_4_FD = MAT_005_Autopower_Mic_4'';');
eval('p_5_5_FD = MAT_006_Autopower_Mic_5'';');
eval('p_6_6_FD = MAT_007_Autopower_Mic_6'';');
eval('p_2_1_FD = MAT_018_Crosspower_Mic_2'';');
eval('p_4_3_FD = MAT_031_Crosspower_Mic_4'';');
eval('p_6_5_FD = MAT_044_Crosspower_Mic_6'';');

if ii == 1
    f_FD = p_1_1_FD(1,:);
end

clear MAT_002_Autopower_Mic_1  MAT_003_Autopower_Mic_2  MAT_004_Autopower_Mic_3
clear MAT_005_Autopower_Mic_4  MAT_006_Autopower_Mic_5  MAT_007_Autopower_Mic_6
clear MAT_018_Crosspower_Mic_2  MAT_031_Crosspower_Mic_4  MAT_044_Crosspower_Mic_6
%Calculate PED
p_1_1_FD = p_1_1_FD(2,:)/ENBW;
p_2_2_FD = p_2_2_FD(2,:)/ENBW;
p_3_3_FD = p_3_3_FD(2,:)/ENBW;
p_4_4_FD = p_4_4_FD(2,:)/ENBW;
p_5_5_FD = p_5_5_FD(2,:)/ENBW;
p_6_6_FD = p_6_6_FD(2,:)/ENBW;
p_2_1_FD = p_2_1_FD(2,:)/ENBW;
p_4_3_FD = p_4_3_FD(2,:)/ENBW;
p_6_5_FD = p_6_5_FD(2,:)/ENBW;
PED_1_2_FD = (1/(4*rho*c^2)) * ( p_1_1_FD + p_2_2_FD + p_2_1_FD + conj(p_2_1_FD) )/2;
PED_3_4_FD = (1/(4*rho*c^2)) * ( p_3_3_FD + p_4_4_FD + p_4_3_FD + conj(p_4_3_FD) )/2;
PED_5_6_FD = (1/(4*rho*c^2)) * ( p_5_5_FD + p_6_6_FD + p_6_5_FD + conj(p_6_5_FD) )/2;
PED_FD = ( PED_1_2_FD + PED_3_4_FD + PED_5_6_FD )/3;
%Calculate KED
omega_FD = 2*pi*f_FD;
KED_Ux_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_1_1_FD + p_2_2_FD - p_2_1_FD -
conj(p_2_1_FD) );
KED_Uy_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_3_3_FD + p_4_4_FD - p_4_3_FD -
conj(p_4_3_FD) );
KED_Uz_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_5_5_FD + p_6_6_FD - p_6_5_FD -
conj(p_6_5_FD) );
KED_FD = (KED_Ux_FD + KED_Uy_FD + KED_Uz_FD);
TED_FD = PED_FD + KED_FD;
GED_FD = BETA*(PED_FD) + (1-BETA)*KED_FD;
%Convert the ED values into third octave bands and truncate it to the frequency range limited
%by the EASERA T60 values
[f_FD_Oct3,PED_FD_Oct3] = MyOct3Bands(f_FD,PED_FD);
```

```
    [~,KED_FD_Oct3] = MyOct3Bands(f_FD,KED_FD);
    [~,TED_FD_Oct3] = MyOct3Bands(f_FD,TED_FD);
    [~,GED_FD_Oct3] = MyOct3Bands(f_FD,GED_FD);
    ED_FD_MAT_full = cat(1,PED_FD_Oct3.',KED_FD_Oct3.',TED_FD_Oct3.',GED_FD_Oct3.');
    %Truncate the 1/3 octave band ED values to the frequency range limited by the EASERA T60
    %values
    ind_l = findnearest(freq_model(1),f_FD_Oct3,0);
    ind_h = findnearest(freq_model(end),f_FD_Oct3,0);

    if ii <= 5
        ED_FD_MAT(:,:,ii+1) = ED_FD_MAT_full(:,ind_l:ind_h);
    end
    if ii == 6
        ED_FD_MAT(:,:,ii-5) = ED_FD_MAT_full(:,ind_l:ind_h);
    end
    if ii == num_pos
        f_FD_Oct3 = f_FD_Oct3(ind_l:ind_h);
    end

    %% Calculate the 1/3 octave band frequency-dependent sound power output of the source (using
    %each of the 4 ED at each measurement position)

    for jj = 1:4
        if jj == 1
            pow_MF_MAT(jj,:,ii) = ED_MF_MAT(jj,:,ii) ./ (
             (manual_gamma/(4*pi*r_ED_probes(ii)^2*c)) + (4./(R_Oct3*c))  );
        elseif jj == 3
            pow_MF_MAT(jj,:,ii) = (ED_MF_MAT(jj,:,ii)/2) ./ (
         (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii),phi_ED_ind(ii),:)).'/(4*pi*r_ED_probes(ii)^2*c))
            + (4./(R_Oct3*c))  );
        else
            pow_MF_MAT(jj,:,ii) = ED_MF_MAT(jj,:,ii) ./ (
         (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii),phi_ED_ind(ii),:)).'/(4*pi*r_ED_probes(ii)^2*c))
            + (4./(R_Oct3*c))  );
        end
    end
    for jj= 1:4
        if ii <= 5
            if jj == 1
                pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
                 (manual_gamma/(4*pi*r_ED_probes(ii+1)^2*c)) + (4./(R_Oct3*c))  );
            elseif jj == 3
                pow_FD_MAT(jj,:,ii+1) = (ED_FD_MAT(jj,:,ii+1)/2) ./ (
   (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
                + (4./(R_Oct3*c))  );
            else
                pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
   (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
                + (4./(R_Oct3*c))  );
            end
        end
        if ii == 6
            if jj == 1
                pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
                 (manual_gamma/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
            elseif jj == 3
                pow_FD_MAT(jj,:,ii-5) = (ED_FD_MAT(jj,:,ii-5)/2) ./ (
                 (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
                 5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
            else
                pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
                 (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
                 5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
            end
        end
    end
end

%% Load in the sound power estimates from the ISO 3741 method (for comparison)
loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3741\Offical ISO 3741 Data\';
```

```
if Source == 1
    load([loc 'REPEAT Processed Dodec SP ISO 3741 Data']);
end
if Source == 2
    load([loc 'REPEAT Processed Single Driver SP ISO 3741 Data']);
end
if Source == 3
    load([loc 'REPEAT Processed Horn SP ISO 3741 Data']);
end

eval('L_w_3741_nv = L_w;');
eval('Overall_L_w_3741_nv = Overall_L_w;');
eval('f_3741 = f_Oct3;');
clear L_w   Overall_L_w   f_Oct3_cal

%% Load in the sound power estimates from the ISO 3745 method - Full Grid (for comparison)
loc2 = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full Grid)\Official ISO
3745 Data\Processed SP\';

if Source == 1
    load([loc2 'Dodec SP ISO 3745 Full Data']);
end
if Source == 2
    load([loc2 'Single Driver SP ISO 3745 Full Data']);
end
if Source == 3
    load([loc2 'Horn SP ISO 3745 Full Data']);
end

eval('L_w_3745_Full = L_w;');
eval('Overall_L_w_3745_Full = Overall_L_w;');
eval('f_3745_Full = f_Oct3_cal;');
clear L_w   A_weight_L_w   Overall_L_w   A_weight_Overall_L_w   f_Oct3_cal

%% Calculate the Mean Sound Power Estimates
mean_pow_MF = mean(pow_MF_MAT(:,:,ED_pos),3);
mean_pow_FD = mean(pow_FD_MAT(:,:,ED_pos),3);
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref);       %no correction terms
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref);       %no correction terms
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((C1 + C2).',4,length(f_Oct3));      %only
                                                                    %atmospheric corrections
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((C1 + C2).',4,length(f_Oct3));      %only
                                                                    %atmospheric corrections
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat(10*log10(1+((S*c)./(8*V.*f_Oct3))).',4,1);
                                                                   %only Waterhouse correction
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat(10*log10(1+((S*c)./(8*V.*f_Oct3))).',4,1);
                                                                   %only Waterhouse correction
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 +
C2).',4,1);                                          %only atmospheric and Waterhouse corrections
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 +
C2).',4,1);                                          %only atmospheric and Waterhouse corrections
mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((4.34*(A.'/S) +
10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 + C2).',4,1);                          %Full corrections
mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((4.34*(A.'/S) +
10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 + C2).',4,1);                          %Full corrections

%% Calculate the Overall Sound Power Level (1/3 Oct)
Overall_Lw_MF = 10*log10(sum(10.^(mean_Lw_MF./10),2));
Overall_Lw_FD = 10*log10(sum(10.^(mean_Lw_FD./10),2));

%% Calculate Statistics (Root-Mean-Square Deviation)
RMSD_MF = zeros(4,1);
RMSD_FD = zeros(4,1);

for ii = 1:4
    RMSD_MF(ii,:) = sqrt( mean((abs(mean_Lw_MF(ii,4:end)) - L_w_3745_Full(11:28).').^2) );
    RMSD_FD(ii,:) = sqrt( mean((abs(mean_Lw_FD(ii,4:end)) - L_w_3745_Full(11:28).').^2) );
end
```

```
RMSD_MF(5,1) = sqrt( mean((L_w_3741_nv(4:end).' - L_w_3745_Full(11:28).').^2) );
RMSD_FD(5,1) = sqrt( mean((L_w_3741_nv(4:end).' - L_w_3745_Full(11:28).').^2) );

%% Plot and Compare the Sound Power Estimates
colors = {'b','g','r','k'};

figure(1)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
for ii = 1:4
    semilogx(freq_model,mean_Lw_MF(ii,:),colors{ii})
    hold on
end
semilogx(f_3741,L_w_3741_nv,'m--',f_3745_Full(8:28),L_w_3745_Full(8:28),'c--')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Source == 1
    title('DODEC - L_w Estimates (Averaged) From the Microflown Probe')
end
if Source == 2
    title('SINGLE DRIVER - L_w Estimates (Averaged)From the Microflown Probe')
end
if Source == 3
    title('HORN - L_w Estimates (Averaged) From the Microflown Probe')
end
legend(['PED (HSE) (OAL = ' num2str(Overall_Lw_MF(1),'%.1f') ' dB, RMSD = '
num2str(RMSD_MF(1),'%.1f') ' dB)'], ['KED (MHSE)          ' num2str(Overall_Lw_MF(2),'%.1f') ' dB
' num2str(RMSD_MF(2),'%.1f') ' dB'], ['TED (MHSE)           ' num2str(Overall_Lw_MF(3),'%.1f') '
dB              ' num2str(RMSD_MF(3),'%.1f') ' dB'], ['GED (MHSE)          '
num2str(Overall_Lw_MF(4),'%.1f') ' dB               ' num2str(RMSD_MF(4),'%.1f') ' dB'], ['ISO
3741                ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB               '
num2str(RMSD_MF(5),'%.1f') ' dB'],['ISO 3745                '
num2str(Overall_L_w_3745_Full,'%.1f') ' dB                N/A'],'Location','SouthEast')
grid

figure(2)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
for ii = 1:4
    semilogx(freq_model,mean_Lw_FD(ii,:),colors{ii})
    hold on
 end
semilogx(f_3741,L_w_3741_nv,'m--',f_3745_Full(8:28),L_w_3745_Full(8:28),'c--')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Source == 1
    title('DODEC - 1/3 Octave Band L_w Estimates From the 6-Mic Probe')
end
if Source == 2
    title('SINGLE DRIVER - 1/3 Octave Band L_w Estimates From the 6-Mic Probe')
end
if Source == 3
    title('HORN - 1/3 Octave Band L_w Estimates From the 6-Mic Probe')
end
legend(['PED (HSE) (OAL = ' num2str(Overall_Lw_FD(1),'%.1f') ' dB, RMSD = '
num2str(RMSD_FD(1),'%.1f') ' dB)'], ['KED (MHSE)          ' num2str(Overall_Lw_FD(2),'%.1f') ' dB
' num2str(RMSD_FD(2),'%.1f') ' dB'], ['TED (MHSE)           ' num2str(Overall_Lw_FD(3),'%.1f') '
dB              ' num2str(RMSD_FD(3),'%.1f') ' dB'], ['GED (MHSE)          '
num2str(Overall_Lw_FD(4),'%.1f') ' dB               ' num2str(RMSD_FD(4),'%.1f') ' dB'], ['ISO
3741                ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB               '
num2str(RMSD_MF(5),'%.1f') ' dB'],['ISO 3745                '
num2str(Overall_L_w_3745_Full,'%.1f') ' dB                N/A'],'Location','SouthEast')
grid
```

ISO_3741_Violation_Tests_with_MHSE.m

```
%% Reverberation Chamber Violation Tests Comparisons for ISO 3741 and the MHSE

clear all; close all; clc;
set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',18);
```

```
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',0.01);
set(0,'DefaultLineLineWidth',2.5);
scsz = get(0,'ScreenSize');
format shortG;

%% Define calculation parameters
Source = 1;       %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
absorp_form = 2;  %1:Sabine absorption formulation; 2:Eyring absorption formulation
air_absorp = 1;   %1:air absorption considered; 0:air absorption not considered
%Assign each position a number (nv = no violation; v = violation)
nv1 = 1; nv2 = 2; nv3 = 3; nv4 = 4; nv5 = 5; nv6 = 6;
v1 = 7; v2 = 8; v3 = 9; v4 = 10; v5 = 11; v6 = 12;
%Choose the sensor positions considered in this calculation
ED_pos = [v1 v2 v3 v4 v5 v6];        %All in violation

%% Define the measurement-specific data
loc_ED = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
Research\Official Thesis Research\OFFICIAL THESIS FILES\Reverb Chamber\Reverb Chamber ED Tests
(5_31_13)\Reverb Chamber ED Data\';
%Input the ED receiver positions
r_ED_probes = [3.19 2.48 4.01 2.47 3.86 2.19 0.886 0.86 1.72 1.09 1.15 0.777]          %distance
                                                          %of ED probe to acoustic center of source
alt_ED_probes = [80 75 40 5 55 5 55 10 65 -15 85 5]
azimuth_ED_probes = [315 245 265 230 205 170 10 265 295 225 215 160];
theta_ED = 90 - alt_ED_probes;
phi_ED = 270 - azimuth_ED_probes

for ii  = 1:length(azimuth_ED_probes)
    if phi_ED(ii) < 0
        phi_ED(ii) = 360 + phi_ED(ii);
    end
end

theta_ED_ind = (theta_ED+5)/5;
phi_ED_ind = (phi_ED+5)/5;
%Atmospheric Conditions
hr = 17;        %relative humidity (%)
T_C = 22.5;     %ambient atmospheric temperature (C)
Pa_mb = 860;    %ambient atmospheric pressure (millibar)
c = 20.05*sqrt(273 + T_C);   %adiabatic speed of sound
Pa = Pa_mb*0.1;              %ambient atmospheric pressure (kPa)
T = T_C + 273.15;            %ambient atmospheric temperature (K)
[rho,rho_dry] = getRho(Pa_mb,T_C,hr); %density of air

%% Define the room-specific data
Lx = 4.96;                        %dimensions of the room (m)
Ly = 5.88;
Lz = 6.99;
S = 2*Lx*Ly + 2*Lx*Lz + 2*Ly*Lz;  %surface area of walls
V = Lx*Ly*Lz;                     %volume of the room
T_60 = [7.305 7.205 6.845 7.555 7.57 8.25 8.085 8.06 7.835 7.955 7.2 6.17 5.165 4.19 3.6 2.85
2.22 1.63 1.215 0.89 0.645];   %T_10 from EASERA (5/27/13)
A = (55.26/c)*(V./T_60);            %equivalent absorption area of the room (m^2)

%% Define constants
BETA = 1/4;         %the Generalized Energy Density weighting factor (1/4 is associated with
optimum spatial uniformity within the GED field)
pow_ref = 1e-12;    %reference sound power
Pr = 101.325;       %reference ambient atmospheric pressure in kilopascals (1 atm)
T0 = 293.15;        %reference air temperature (K)
T01 = 273.16;       %triple-point isotherm temperature of water (K)
ENBW = 1.5;           %equal noise bandwidth for the Hanning window function
theta0 = 314;
theta1 = 296;
spacing = 0.025;             %spacing between the mics in the 6-mic probe (m)
manual_gamma = ones(1,21);
C1 = -10*log10(Pa/Pr) + 5*log10((273.15 + T_C)/theta0);
C2 = -10*log10(Pa/Pr) + 15*log10((273.15 + T_C)/theta1);
```

```
%% Load in the data from the anechoic chamber and create 1/3 octave band directivity factors (for
%all THREE sources)
theta = (0:5:180)'*pi/180;     %polar angles of the measurement sphere
phi = (0:5:355)*pi/180;        %azimuth angles of the measurement sphere
%Load in the FRF data from each source

if Source == 1    %Dodec
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\Dodec Directivity\Data\';
    load([loc_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 2    %Single Driver
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 3    %Horn
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full
    Grid)\Official ISO 3745 Data\Horn Directivity\Data\';
    load([loc_Dir_Fact 'Horn_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end

%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;              %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
                                                %octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
[~,summed_weighted_Oct3_H2_sq] = MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h

%% Calculate the 1/3 octave band frequency-dependent room constant, R
%Generate the Total Absorption Coefficient (Eq 1-2.6.7 of Phscs 661 notes)
h = hr*(Pr/Pa)*10^(-6.8346*(T01/T)^(1.261)+4.6151); %the molar concentration of water vapor (%)
fro = (Pa/Pr)*(24+4.04e4*h*((0.02+h)/(0.391+h)));   %the relaxation frequency of oxygen
frN = (Pa/Pr)*(T0/T)^(1/2)*(9+280*h*exp(-4.170*((T0/T)^(1/3)-1))); %the relaxation frequency of
nitrogen
alpha_prop = f_Oct3.^2.*((1.84e-11*(Pr/Pa)*(T/T0)^(1/2)) + (T0/T)^(5/2).*(((0.01275*exp(-
2239.1/T))./(fro+(f_Oct3.^2/fro)))+((0.1068*exp(-3352.0/T))./(frN+(f_Oct3.^2/frN)))));  %the
total absorption due to wave propagation in air
ind_l = find(f_Oct3 == freq_model(1));                %narrowest frequency band indices (limited
by EASERA T_60 frequency range)
ind_h = find(f_Oct3 == freq_model(end));
alpha_prop = alpha_prop(ind_l:ind_h);
f_Oct3 = f_Oct3(ind_l:ind_h);
%Calculate the room constant, R
```

```
if absorp_form == 1                                     %Sabine absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = (0.161*V)./(S*T_60);               %the Sabine absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S); %the total absorption coefficient in the
                                                        %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant
    end
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = ((0.161*V)./(S*T_60)).' - ((8*alpha_prop*V)/S); %the Sabine absorption
                                                                    %coefficient of the wall
                                                                    %(considering the effects of
                                                                    %air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);              %the total absorption in the
                                                                    %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                  %the 1/3 octave band
                                                                    %frequency-dependent room
                                                                    %constant (including air
                                                                    %absorption)
    end
end
if absorp_form == 2                                     %Eyring absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = 1-exp(-(0.161*V)./(S*T_60));       %The Eyring absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S); %the total absorption in the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant (no air absorption)
    end
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = 1-exp((1/S)*(8*alpha_prop.'*V-((0.161*V)./T_60))); %the Eyring absorption
                                                                    %coefficient of the wall
                                                                    %(considering air
                                                                    %absorption)
        alpha_tot = alpha_wall.' + ((8*alpha_prop*V)/S);            %the total absorption in
                                                                    %the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                  %the 1/3 octave band
                                                                    %frequency-dependent room
                                                                    %constant (including air
                                                                    %absorption)
    end
end

%% Calculate the four acoustic energy density quantities (PED, KED, TED, and GED) in 1/3 octave
%bands (from each probe)
%Load in the data
ED_MF_MAT = zeros(4,21,12);
ED_FD_MAT = zeros(4,21,12);
pow_MF_MAT = zeros(4,21,12);
pow_FD_MAT = zeros(4,21,12);

for ii = 1:12
    if Source == 1
        if ii <= 5
            load([loc_ED 'Dodec MF ' num2str(ii) ' FD ' num2str(ii+1) ' Reverb No Violation (
            Official).mat'])
        end
        if ii == 6
            load([loc_ED 'Dodec MF ' num2str(ii) ' FD ' num2str(ii-5) ' Reverb No Violation
            (Official).mat'])
        end
        if ii > 6 && ii <= 11
             load([loc_ED 'Dodec MF ' num2str(ii-6) ' FD ' num2str(ii-5) ' Reverb Violation
             (Official).mat'])
        end
        if ii == 12
            load([loc_ED 'Dodec MF ' num2str(ii-6) ' FD ' num2str(ii-11) ' Reverb Violation
            (Official).mat'])
        end
```

```
end
if Source == 2
    if ii <= 5
        load([loc_ED 'SD MF ' num2str(ii) ' FD ' num2str(ii+1) ' Reverb No Violation
        (Official).mat'])
    end
    if ii == 6
        load([loc_ED 'SD MF ' num2str(ii) ' FD ' num2str(ii-5) ' Reverb No Violation
        (Official).mat'])
    end
    if ii > 6 && ii <= 11
         load([loc_ED 'SD MF ' num2str(ii-6) ' FD ' num2str(ii-5) ' Reverb Violation
         (Official).mat'])
    end
    if ii == 12
        load([loc_ED 'SD MF ' num2str(ii-6) ' FD ' num2str(ii-11) ' Reverb Violation
        (Official).mat'])
    end
end
if Source == 3
    if ii <= 5
        load([loc_ED 'Horn MF ' num2str(ii) ' FD ' num2str(ii+1) ' Reverb No Violation
        (Official).mat'])
    end
    if ii == 6
        load([loc_ED 'Horn MF ' num2str(ii) ' FD ' num2str(ii-5) ' Reverb No Violation
        (Official).mat'])
    end
    if ii > 6 && ii <= 11
         load([loc_ED 'Horn MF ' num2str(ii-6) ' FD ' num2str(ii-5) ' Reverb Violation
         (Official).mat'])
    end
    if ii == 12
        load([loc_ED 'Horn MF ' num2str(ii-6) ' FD ' num2str(ii-11) ' Reverb Violation
        (Official).mat'])
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%MICROFLOWN PROBE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
eval('p_sq_V_MF = MAT_008_Autopower_Pressure(:,2)'';');
eval('ux_sq_V_MF = MAT_009_Autopower_U_x(:,2)'';');
eval('uy_sq_V_MF = MAT_010_Autopower_U_y(:,2)'';');
eval('uz_sq_V_MF = MAT_011_Autopower_U_z(:,2)'';');

if ii == 1
    eval('f_MF = MAT_008_Autopower_Pressure(:,1)'';');
end

clear  MAT_008_Autopower_Pressure  MAT_009_Autopower_U_x
clear  MAT_010_Autopower_U_y  MAT_011_Autopower_U_z
MF_Data_Mat = cat(1,p_sq_V_MF/ENBW,ux_sq_V_MF/ENBW,uy_sq_V_MF/ENBW,uz_sq_V_MF/ENBW);
%Convert the squared voltages to squared pressure and particle velocity components
[MF_p_sq,MF_U_mag_sq,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,MF_Data_Mat);
%Convert the narrow band squared pressure to 1/3 octave band squared pressure

if ii == 1
   [f_MF_Oct3,MF_p_sq_Oct3] = MyOct3Bands(f_MF,MF_p_sq);
   [~,MF_U_mag_sq_Oct3] = MyOct3Bands(f_MF,MF_U_mag_sq);
else
   [~,MF_p_sq_Oct3] = MyOct3Bands(f_MF,MF_p_sq);
   [~,MF_U_mag_Oct3] = MyOct3Bands(f_MF,MF_U_mag_sq);
end

%Truncate the 1/3 octave band squared pressure data to go from 100 Hz to 10 kHz

if ii == 1
    ind_l = findnearest(freq_model(1),f_MF_Oct3,0);
    ind_h = findnearest(freq_model(end),f_MF_Oct3,0);
end
if ii == 12
    f_MF_Oct3 = f_MF_Oct3(ind_l:ind_h);
```

```
end

%Create a matrix of the PED, KED, TED, and GED vectors (1/3 octave bands)
PED_MF = (MF_p_sq_Oct3 / (2*rho*c^2)).';
KED_MF = ((rho/2) * MF_U_mag_sq_Oct3).';
TED_MF = (PED_MF + KED_MF);
GED_MF = (BETA*(PED_MF) + (1-BETA)*KED_MF);
ED_MF_MAT_full = cat(1,PED_MF,KED_MF,TED_MF,GED_MF);
%Truncate the 1/3 octave band ED matrix to the frequency range limited by the EASERA T60
%values
ED_MF_MAT(:,:,ii) = ED_MF_MAT_full(:,ind_l:ind_h);

if ii == 12
    f_MF = f_MF_Oct3;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%6-MIC ED PROBE%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Load in the data
eval('p_1_1_FD = MAT_002_Autopower_Mic_1'';');
eval('p_2_2_FD = MAT_003_Autopower_Mic_2'';');
eval('p_3_3_FD = MAT_004_Autopower_Mic_3'';');
eval('p_4_4_FD = MAT_005_Autopower_Mic_4'';');
eval('p_5_5_FD = MAT_006_Autopower_Mic_5'';');
eval('p_6_6_FD = MAT_007_Autopower_Mic_6'';');
eval('p_2_1_FD = MAT_018_Crosspower_Mic_2'';');
eval('p_4_3_FD = MAT_031_Crosspower_Mic_4'';');
eval('p_6_5_FD = MAT_044_Crosspower_Mic_6'';');

if ii == 1
    f_FD = p_1_1_FD(1,:);
end

clear MAT_002_Autopower_Mic_1  MAT_003_Autopower_Mic_2  MAT_004_Autopower_Mic_3
clear MAT_005_Autopower_Mic_4  MAT_006_Autopower_Mic_5  MAT_007_Autopower_Mic_6
clear MAT_018_Crosspower_Mic_2  MAT_031_Crosspower_Mic_4  MAT_044_Crosspower_Mic_6
%Calculate PED
p_1_1_FD = p_1_1_FD(2,:)/ENBW;
p_2_2_FD = p_2_2_FD(2,:)/ENBW;
p_3_3_FD = p_3_3_FD(2,:)/ENBW;
p_4_4_FD = p_4_4_FD(2,:)/ENBW;
p_5_5_FD = p_5_5_FD(2,:)/ENBW;
p_6_6_FD = p_6_6_FD(2,:)/ENBW;
p_2_1_FD = p_2_1_FD(2,:)/ENBW;
p_4_3_FD = p_4_3_FD(2,:)/ENBW;
p_6_5_FD = p_6_5_FD(2,:)/ENBW;
PED_1_2_FD = (1/(4*rho*c^2)) * ( p_1_1_FD + p_2_2_FD + p_2_1_FD + conj(p_2_1_FD) )/2;
PED_3_4_FD = (1/(4*rho*c^2)) * ( p_3_3_FD + p_4_4_FD + p_4_3_FD + conj(p_4_3_FD) )/2;
PED_5_6_FD = (1/(4*rho*c^2)) * ( p_5_5_FD + p_6_6_FD + p_6_5_FD + conj(p_6_5_FD) )/2;
PED_FD = ( PED_1_2_FD + PED_3_4_FD + PED_5_6_FD )/3;
%Calculate KED
omega_FD = 2*pi*f_FD;
KED_Ux_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_1_1_FD + p_2_2_FD - p_2_1_FD -
conj(p_2_1_FD) );
KED_Uy_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_3_3_FD + p_4_4_FD - p_4_3_FD -
conj(p_4_3_FD) );
KED_Uz_FD = ( 1./(4*omega_FD.^2*rho*spacing^2) ) .* ( p_5_5_FD + p_6_6_FD - p_6_5_FD -
conj(p_6_5_FD) );
KED_FD = (KED_Ux_FD + KED_Uy_FD + KED_Uz_FD);
TED_FD = PED_FD + KED_FD;
GED_FD = BETA*(PED_FD) + (1-BETA)*KED_FD;
%Convert the ED values into third octave bands and truncate it to the frequency range limited
%by the EASERA T60 values
[f_FD_Oct3,PED_FD_Oct3] = MyOct3Bands(f_FD,PED_FD);
[~,KED_FD_Oct3] = MyOct3Bands(f_FD,KED_FD);
[~,TED_FD_Oct3] = MyOct3Bands(f_FD,TED_FD);
[~,GED_FD_Oct3] = MyOct3Bands(f_FD,GED_FD);
ED_FD_MAT_full = cat(1,PED_FD_Oct3.',KED_FD_Oct3.',TED_FD_Oct3.',GED_FD_Oct3.');
%Truncate the 1/3 octave band ED values to the frequency range limited by the EASERA T60
%values
ind_l = findnearest(freq_model(1),f_FD_Oct3,0);
```

```
ind_h = findnearest(freq_model(end),f_FD_Oct3,0);

if ii <= 5
    ED_FD_MAT(:,:,ii+1) = ED_FD_MAT_full(:,ind_l:ind_h);
end
if ii == 6
    ED_FD_MAT(:,:,ii-5) = ED_FD_MAT_full(:,ind_l:ind_h);
end
if ii > 6 && ii <= 11
      ED_FD_MAT(:,:,ii+1) = ED_FD_MAT_full(:,ind_l:ind_h);
end
if ii == 12
    ED_FD_MAT(:,:,ii-5) = ED_FD_MAT_full(:,ind_l:ind_h);
end
if ii == 12
    f_FD_Oct3 = f_FD_Oct3(ind_l:ind_h);
end

%% Calculate the 1/3 octave band frequency-dependent sound power output of the source (using
%each of the 4 ED at each measurement position)

for jj = 1:4
    if jj == 1
        pow_MF_MAT(jj,:,ii) = ED_MF_MAT(jj,:,ii) ./ (
         (manual_gamma/(4*pi*r_ED_probes(ii)^2*c)) + (4./(R_Oct3*c))  );
    elseif jj == 3
        pow_MF_MAT(jj,:,ii) = (ED_MF_MAT(jj,:,ii)/2) ./ (
     (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii),phi_ED_ind(ii),:)).'/(4*pi*r_ED_probes(ii)^2*c))
        + (4./(R_Oct3*c))  );
    else
        pow_MF_MAT(jj,:,ii) = ED_MF_MAT(jj,:,ii) ./ (
     (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii),phi_ED_ind(ii),:)).'/(4*pi*r_ED_probes(ii)^2*c))
        + (4./(R_Oct3*c))  );
    end
end
for jj= 1:4
    if ii <= 5
        if jj == 1
            pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
             (manual_gamma/(4*pi*r_ED_probes(ii+1)^2*c)) + (4./(R_Oct3*c))  );
        elseif jj == 3
            pow_FD_MAT(jj,:,ii+1) = (ED_FD_MAT(jj,:,ii+1)/2) ./ (
(squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
            + (4./(R_Oct3*c))  );
        else
            pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
(squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
            + (4./(R_Oct3*c))  );
       end
    end
    if ii == 6
         if jj == 1
            pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
             (manual_gamma/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
         elseif jj == 3
            pow_FD_MAT(jj,:,ii-5) = (ED_FD_MAT(jj,:,ii-5)/2) ./ (
             (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
             5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
        else
            pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
             (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
             5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
        end
    end
    if ii > 6 && ii <= 11
        if jj == 1
            pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
             (manual_gamma/(4*pi*r_ED_probes(ii+1)^2*c)) + (4./(R_Oct3*c))  );
        elseif jj == 3
            pow_FD_MAT(jj,:,ii+1) = (ED_FD_MAT(jj,:,ii+1)/2) ./ (
(squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
```

```
                + (4./(R_Oct3*c))  );
            else
                pow_FD_MAT(jj,:,ii+1) = ED_FD_MAT(jj,:,ii+1) ./ (
   (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii+1),phi_ED_ind(ii+1),:)).'/(4*pi*r_ED_probes(ii+1)^2*c))
                + (4./(R_Oct3*c))  );
            end
        end
        if ii == 12
             if jj == 1
                pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
                (manual_gamma/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
             elseif jj == 3
                pow_FD_MAT(jj,:,ii-5) = (ED_FD_MAT(jj,:,ii-5)/2) ./ (
                (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
                 5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
            else
                pow_FD_MAT(jj,:,ii-5) = ED_FD_MAT(jj,:,ii-5) ./ (
                (squeeze(Dir_Fact_Oct3(theta_ED_ind(ii-5),phi_ED_ind(ii-
                5),:)).'/(4*pi*r_ED_probes(ii-5)^2*c)) + (4./(R_Oct3*c))  );
            end
        end
    end
end

%% Load in the sound power estimates from the VIOLATED ISO 3741 method (for comparison)
loc_violations = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Degree Documents\Thesis
Writing\Resources for Thesis Writing\Matlab Code\Chapter 3\Reverb ED and Violaiton Tests\';

if Source == 1
    load([loc_violations 'DODEC ALL VIOLATION DATA']);
end
if Source == 2
    load([loc_violations 'SD ALL VIOLATION DATA']);
end
if Source == 3
    load([loc_violations 'HORN ALL VIOLATION DATA']);
end

eval('L_w_Standard_violations = L_w;');
eval('Overall_L_w_Standard_violations = Overall_L_w;');
clear L_w   Overall_L_w

%% Load in the sound power estimates from the ISO 3741 method (for comparison)
loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3741\Offical ISO 3741 Data\';

if Source == 1
    load([loc 'Processed Dodec SP ISO 3741 Data']);
end
if Source == 2
    load([loc 'Processed Single Driver SP ISO 3741 Data']);
end
if Source == 3
    load([loc 'Processed Horn SP ISO 3741 Data']);
end

eval('L_w_3741_nv = L_w;');
eval('Overall_L_w_3741_nv = Overall_L_w;');
eval('f_3741 = f_Oct3;');
clear L_w   Overall_L_w   f_Oct3_cal

%% Load in the sound power estimates from the ISO 3745 method - Full Grid (for comparison)
loc2 = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full Grid)\Official ISO
3745 Data\Processed SP\';

if Source == 1
    load([loc2 'Dodec SP ISO 3745 Full Data']);
end
if Source == 2
    load([loc2 'Single Driver SP ISO 3745 Full Data']);
```

```
end
if Source == 3
    load([loc2 'Horn SP ISO 3745 Full Data']);
end

eval('L_w_3745_Full = L_w;');
eval('Overall_L_w_3745_Full = Overall_L_w;');
eval('f_3745_Full = f_Oct3_cal;');
clear L_w   A_weight_L_w   Overall_L_w   A_weight_Overall_L_w   f_Oct3_cal

%% Calculate the Mean Sound Power Estimates
mean_pow_MF = mean(pow_MF_MAT(:,:,ED_pos),3);
mean_pow_FD = mean(pow_FD_MAT(:,:,ED_pos),3);
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref);       %no correction terms
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref);       %no correction terms
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((C1 + C2).',4,length(f_Oct3));      %only
                                                                    %atmospheric corrections
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((C1 + C2).',4,length(f_Oct3));      %only
                                                                    %atmospheric corrections
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat(10*log10(1+((S*c)./(8*V.*f_Oct3))).',4,1);
                                                                  %only Waterhouse correction
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat(10*log10(1+((S*c)./(8*V.*f_Oct3))).',4,1);
                                                                   %only Waterhouse correction
% mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 +
C2).',4,1);                                             %only atmospheric and Waterhouse corrections
% mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 +
C2).',4,1);                                             %only atmospheric and Waterhouse corrections
mean_Lw_MF = 10*log10(mean_pow_MF/pow_ref) + repmat((4.34*(A.'/S) +
10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 + C2).',4,1);  %Full corrections
mean_Lw_FD = 10*log10(mean_pow_FD/pow_ref) + repmat((4.34*(A.'/S) +
10*log10(1+((S*c)./(8*V.*f_Oct3))) + C1 + C2).',4,1);  %Full corrections

%% Calculate the Overall Sound Power Level (1/3 Oct)
Overall_Lw_MF = 10*log10(sum(10.^(mean_Lw_MF./10),2));
Overall_Lw_FD = 10*log10(sum(10.^(mean_Lw_FD./10),2));

%% Calculate Statistics (Root-Mean-Square Deviation)
RMSD_MF = zeros(6,1);
RMSD_FD = zeros(6,1);

for ii = 1:4
    RMSD_MF(ii,:) = sqrt( mean((abs(mean_Lw_MF(ii,4:end)) - L_w_3745_Full(11:28).').^2) );
    RMSD_FD(ii,:) = sqrt( mean((abs(mean_Lw_FD(ii,4:end)) - L_w_3745_Full(11:28).').^2) );
end

RMSD_MF(5,:) = sqrt( mean((L_w_Standard_violations(4:end).' - L_w_3745_Full(11:28).').^2) );
RMSD_FD(5,:) = sqrt( mean((L_w_Standard_violations(4:end).' - L_w_3745_Full(11:28).').^2) );
RMSD_MF(6,1) = sqrt( mean((L_w_3741_nv(4:end).' - L_w_3745_Full(11:28).').^2) );
RMSD_FD(6,1) = sqrt( mean((L_w_3741_nv(4:end).' - L_w_3745_Full(11:28).').^2) );

%% Plot and Compare the Sound Power Estimates
colors = {'b','g','k'};

figure(1)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,mean_Lw_MF(1,:),'g',freq_model,mean_Lw_MF(4,:),'k',freq_model,L_w_Standard_vi
olations,'b',f_3741,L_w_3741_nv,'m--',f_3745_Full(8:28),L_w_3745_Full(8:28),'c--')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Source == 1
    title('DODEC - L_w Estimates From the Microflown Probe')
end
if Source == 2
    title('SINGLE DRIVER - L_w Estimates From the Microflown Probe')
end
if Source == 3
    title('HORN - L_w Estimates From the Microflown Probe')
end
legend(['PED (HSE)    (OAL = ' num2str(Overall_Lw_MF(1),'%.1f') ' dB, RMSD = '
num2str(RMSD_MF(1),'%.1f') ' dB)'], ['GED (MHSE)             ' num2str(Overall_Lw_MF(4),'%.1f') '
dB                ' num2str(RMSD_MF(4),'%.1f') ' dB'], ['ISO 3741 (Violations) '
```

```
num2str(Overall_L_w_Standard_violations,'%.1f') ' dB                ' num2str(RMSD_MF(5),'%.1f')
' dB'], ['ISO 3741                    ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB                '
num2str(RMSD_MF(6),'%.1f') ' dB'],['ISO 3745                          '
num2str(Overall_L_w_3745_Full,'%.1f') ' dB                  N/A'],'Location','SouthEast')
grid

figure(2)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,mean_Lw_FD(1,:),'g',freq_model,mean_Lw_FD(4,:),'k',freq_model,L_w_Standard_vi
olations,'b',f_3741,L_w_3741_nv,'m--',f_3745_Full(8:28),L_w_3745_Full(8:28),'c--')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Source == 1
    title('DODEC - L_w Estimates From the 6-Mic Probe')
end
if Source == 2
    title('SINGLE DRIVER - L_w Estimates From the 6-Mic Probe')
end
if Source == 3
    title('HORN - L_w Estimates From the 6-Mic Probe')
end
legend(['PED (HSE)    (OAL = ' num2str(Overall_Lw_FD(1),'%.1f') ' dB, RMSD = '
num2str(RMSD_FD(1),'%.1f') ' dB)'], ['GED (MHSE)              ' num2str(Overall_Lw_FD(4),'%.1f') '
dB                ' num2str(RMSD_FD(4),'%.1f') ' dB'], ['ISO 3741 (Violations) '
num2str(Overall_L_w_Standard_violations,'%.1f') ' dB                ' num2str(RMSD_FD(5),'%.1f')
' dB'], ['ISO 3741                    ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB                '
num2str(RMSD_FD(6),'%.1f') ' dB'],['ISO 3745                          '
num2str(Overall_L_w_3745_Full,'%.1f') ' dB                  N/A'],'Location','SouthEast')
grid
```

findnearest.m

```
function [r,c,V] = findnearest(srchvalue,srcharray,bias)

% Usage:
% Find the nearest numerical value in an array to a search value
% All occurances are returned as array subscripts
% Output:
% For 2D matrix subscripts (r,c) use:
%       [r,c] = findnearest(srchvalue,srcharray,gt_or_lt)
% To also output the found value (V) use:
%       [r,c,V] = findnearest(srchvalue,srcharray,gt_or_lt)
% For single subscript (i) use:
%         i   = findnearest(srchvalue,srcharray,gt_or_lt)
% Inputs:
%    srchvalue = a numerical search value
%    srcharray = the array to be searched
%    bias      = 0 (default) for no bias
%                -1 to bias the output to lower values
%                 1 to bias the search to higher values
%                (in the latter cases if no values are found
%                 an empty array is ouput)
% By Tom Benson (2002)
% University College London
% t.benson@ucl.ac.uk

if nargin<2
    error('Need two inputs: Search value and search array')
elseif nargin<3
    bias = 0;
end

% find the differences
srcharray = srcharray-srchvalue;

if bias == -1   % only choose values <= to the search value
    srcharray(srcharray>0) =inf;
elseif bias == 1  % only choose values >= to the search value
    srcharray(srcharray<0) =inf;
```

```
end

% give the correct output
if nargout==1 | nargout==0
    if all(isinf(srcharray(:)))
        r = [];
    else
        r = find(abs(srcharray)==min(abs(srcharray(:))));
    end
elseif nargout>1
    if all(isinf(srcharray(:)))
        r = [];c=[];
    else
        [r,c] = find(abs(srcharray)==min(abs(srcharray(:))));
    end
    if nargout==3
        V = srcharray(r,c)+srchvalue;
    end
end
```

get_MicroF_Vals_Uncorrected.m

```
%% Microflown Probe with Singal Conditioner set to UNCORRECTED MODE
%This function takes a matrix of Microflown data (in V^2) of size (4,length(f))
%(pressure,ux,uy,uz)' and applies the probe-specific calibration. The
%outputs are frequency-dependent squared pressure magnitude, squared
%particle velocity magnitude, and the three squared particle velocity components.

function [MicroF_p_sq,MicroF_U_mag_sq_uc,MicroF_Ux_sq_uc,MicroF_Uy_sq_uc,MicroF_Uz_sq_uc] =
get_MicroF_Vals_Uncorrected(f,DataMat)

 %% Input the Pressure Microphone Sensitivity Equations
 %Input the Pressure Sensitivity at 1 kHz
 S_p_at_1khz = 51.8;   %mV/Pa
 %Input the Sensititvity cornerfrequencies
 f_c1p = 21;            %Hz
 f_c2p = 182;           %Hz
 f_c3p = 16838;         %Hz
 %Input the Phase cornerfrequencies
 c1p = 20;             %Hz
 c2p = 164;            %Hz
 c3p = 18288;          %Hz
 %Input the Frequency-Dependent Sensitivity Equation
 S_p = S_p_at_1khz * (sqrt(1+(f./f_c3p).^2)./(sqrt(1+(f_c1p./f).^2).*sqrt(1+(f_c2p./f).^2)));
                                                                                %mV/Pa
 %Input the Frequency-Dependent Pressure Phase Response of the Probe
 theta_p = atand(c1p./f) + atand(c2p./f) + atand(f./c3p);   %deg

 %% Input the BLUE Particle Velocity Sensor Sensitivity Equations
 %(Blue = x Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_x = 24.284;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_x = 89;           %Hz
 f_c2u_x = 584;          %Hz
 f_c3u_x = 5110.44;      %Hz
 f_c4u_x = 196;          %Hz
 %Input the Phase cornerfrequencies
 c1u_x = 36;           %Hz
 c2u_x = 337;          %Hz
 c3u_x = 26278.11;     %Hz
 c4u_x = 325;          %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
 S_u_x_uc =  S_u_250Hz_x ./
(sqrt(1+(f_c1u_x./f).^2).*sqrt(1+(f/f_c2u_x).^2).*sqrt(1+(f/f_c3u_x).^2).*sqrt(1+(f_c4u_x./f).^2)
 );  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in Corrected Mode
 theta_u_x = atand(c1u_x./f) + atand(c4u_x./f);        %deg
```

```
 %% Input the RED Particle Velocity Sensor Sensitivity Equations
 %(Red = Y Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_y = 30.17;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_y = 142;         %Hz
 f_c2u_y = 692;         %Hz
 f_c3u_y = 4870;        %Hz
 f_c4u_y = 141;         %Hz
 %Input the Phase cornerfrequencies
 c1u_y = 44;            %Hz
 c2u_y = 362;           %Hz
 c3u_y = 20632;         %Hz
 c4u_y = 359;           %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
 S_u_y_uc =  S_u_250Hz_y ./
(sqrt(1+(f_c1u_y./f).^2).*sqrt(1+(f/f_c2u_y).^2).*sqrt(1+(f/f_c3u_y).^2).*sqrt(1+(f_c4u_y./f).^2)
 );  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in
 %Corrected Mode
 theta_u_y = atand(c1u_y./f) + atand(c4u_y./f);        %deg

 %% Input the GREEN Particle Velocity Sensor Sensitivity Equations
 %(Green = Z Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_z = 8.314;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_z = 66;          %Hz
 f_c2u_z = 571;         %Hz
 f_c3u_z = 7858;        %Hz
 f_c4u_z = 66;          %Hz
 %Input the Phase cornerfrequencies
 c1u_z = 21;            %Hz
 c2u_z = 555;           %Hz
 c3u_z = 15846;         %Hz
 c4u_z = 65;            %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
        S_u_z_uc =  S_u_250Hz_z ./
(sqrt(1+(f_c1u_z./f).^2).*sqrt(1+(f/f_c2u_z).^2).*sqrt(1+(f/f_c3u_z).^2).*sqrt(1+(f_c4u_z./f).^2)
 );  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in
 %Corrected Mode
 theta_u_z = atand(c1u_z./f) + atand(c4u_z./f);        %deg

 %% Convert the voltages to pressure and particle velocity magnitudes
 MicroF_p_sq = ( (sqrt((DataMat(1,:)))*1000) ./ S_p ).^2;  %Pa^2
 MicroF_Ux_sq_uc = ( sqrt(DataMat(2,:)) ./ S_u_x_uc ).^2;   %(m/s)^2
 MicroF_Uy_sq_uc = ( sqrt(DataMat(3,:)) ./ S_u_y_uc ).^2;   %(m/s)^2
 MicroF_Uz_sq_uc = ( sqrt(DataMat(4,:)) ./ S_u_z_uc ).^2;   %(m/s)^2
 MicroF_U_mag_sq_uc = MicroF_Ux_sq_uc + MicroF_Uy_sq_uc + MicroF_Uz_sq_uc;  %(m/s)^2

end
```

getRho.m

```
%This function calculates the density of air based on pressure (in mB),
%temperature (degrees Celsius), and relative humidity (in percent). The two
%outputs are calculated for humid air and dry air. The humid air
%calculation should be more accurate.

function [rho_humid,rho_dry] = getRho(pressure,temp,humidity)
    R = 287.05;
    P = pressure*100;
    R_d = 287.05;
    T = temp + 273.15;
    p_sat = 6.1078*10^( (7.5*temp)/(temp + 237.3) );
    P_v = humidity*p_sat;
    rho_humid = ( P/(R_d*T) ) * ( 1-((0.378*P_v)/P) );
    rho_dry = (P/(R*T));
```

```
end
```

MyOct3Bands.m

```
%FOR ARRAYS
%This function takes narrow band squared data magnitudes (proportional to
%intensity) and converts them into third-octave (squared summed energy)
%bands. It also outputs a new frequency array that corresponds to the
%third-octave bands to which the narrow band data were assigned.

function [f_Oct3,data_Oct3] = MyOct3Bands(f_narrow,data_narrow_sq)
    oct_c = [                          0.8   1     1.25  1.6   2 ...
        2.5   3.15  4     5     6.3   8     10    12.5  16    20 ...
        25    31.5  40    50    63    80    100   125   160   200 ...
        250   315   400   500   630   800   1000  1250  1600  2000 ...
        2500  3150  4000  5000  6300  8000  10000 12500 16000 20000 ...
        25000 31500 40000 50000 63000 80000 100000];

    if f_narrow(1) < 0.8
         f_narrow(1) = 0.8;
    end

    BandNum = round(10*log10(f_narrow));
    bands = unique(BandNum);
    data_Oct3 = zeros(length(bands),1);

    for i = 1:length(bands)
        inds = BandNum == bands(i);
        data_Oct3(i) = sum(data_narrow_sq(inds));
    end

    f_Oct3 = oct_c(bands+2).';
end
```

MyOct3Bands_MAT.m

```
%FOR 3-D MATRICES of size  (:,:,freqs)
%This function takes narrow band squared data magnitudes (proportional to
%intensity) and converts them into third-octave (squared summed energy)
%bands. It also outputs a new frequency array that corresponds to the
%third-octave bands to which the narrow band data were assigned.

function [f_Oct3,data_Oct3] = MyOct3Bands_MAT(f_narrow,data_narrow_sq)
    oct_c = [                          0.8   1     1.25  1.6   2 ...
        2.5   3.15  4     5     6.3   8     10    12.5  16    20 ...
        25    31.5  40    50    63    80    100   125   160   200 ...
        250   315   400   500   630   800   1000  1250  1600  2000 ...
        2500  3150  4000  5000  6300  8000  10000 12500 16000 20000 ...
        25000 31500 40000 50000 63000 80000 100000];

    if f_narrow(1) < 0.8
         f_narrow(1) = 0.8;
    end

    BandNum = round(10*log10(f_narrow));
    bands = unique(BandNum);
    data_Oct3 = zeros(length(data_narrow_sq(:,1,1)),length(data_narrow_sq(1,:,1)),length(bands));

    for i = 1:length(bands)
        inds = BandNum == bands(i);
        data_Oct3(:,:,i) = sum(data_narrow_sq(:,:,inds),3);
    end

    f_Oct3 = oct_c(bands+2).';
end
```

Appendix F

Matlab Code for Chapter 4

Two_Point_In_Situ_Method_GRAS.m

```
%% 2-Point In Situ Approach with A Reference Dirctivity Source (USING GENERALIZED ENERGY
DENSITY)(11/27/13)
%% Only the 6-mic Finite Difference Probe was used in these Experiments

clear all; close all; clc;
set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',18);
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',0.01);
set(0,'DefaultLineLineWidth',2.5);
scsz = get(0,'ScreenSize');
format shortG;

%% Define calculation parameters
Source = 1;          %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
Ref_Source = 2;      %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
Angle_num = 1;       %1, 2, or 3 only (1: on-axis; 2: 45 degrees off-axis; 3: 90 degrees off-
                     %axis)
FullCorrections = 1; %0: no correction terms condidered in the 2-point in situ method; 1: all
                     %corrections considered

%For Angles 1 and 2 Valid ED positions are: 1-3 only. For Angle 3 Valid ED positions are: 1 and 2
%only.
r_ED_close = 1;      %closer ED probe position to the SOURCE UNDER TEST for the in situ method
r_ED_far = 3;        %further ED probe position from the SOURCE UNDER TEST for the in situ method
r_ED_close_ref = 1;  %closer ED probe position to the REFERENCE SOURCE for the in situ method
r_ED_far_ref = 3;    %further ED probe position from the REFERECE SOURCE for the in situ method
absorp_form = 2; %1:Sabine absorption formulation; 2:Eyring absorption formulation
air_absorp = 1;  %1:air absorption considered, direct and reverberant energy corrections applied;
                 %0: all corrections not included

%% Define constants
Pr = 101.325;       %reference ambient atmospheric pressure in kilopascals (1 atm)
T0 = 293.15;        %reference air temperature (K)
T01 = 273.16;       %triple-point isotherm temperature of water (K)
freq_model = [100 125 160 200 250 315 400 500 630 800 1000 1250 1600 2000 2500 3150 4000 5000
              6300 8000 10000];
p_ref = 20e-6;
BETA = 1/4;         %the Generalized Energy Density weighting factor (1/4 is associated with
                    %optimum spatial uniformity within the GED field)
spacing = 0.025;    %spacing between the mics in the 6-mic probe (m)
```

```
pow_ref = 1e-12;    %reference sound power
ENBW = 1.5;         %equal noise bandwidth for the Hanning window function
theta0 = 314;
theta1 = 296;

%% Input the Measurement-Specific Data
hr = 42.5;        %relative humidity (%)
T_C = 21.65;     %ambient atmospheric temperature (C)
Pa_mb = 852.5;    %ambient atmospheric pressure (millibar)
c = 20.05*sqrt(273 + T_C);            %adiabatic speed of sound
Pa = Pa_mb*0.1;                       %ambient atmospheric pressure (kPa)
T = T_C + 273.15;                     %ambient atmospheric temperature (K)
[rho,rho_dry] = getRho(Pa_mb,T_C,hr); %density of air

%% Define the room-specific data
Lx = 4.96;                          %dimensions of the room (m)
Ly = 5.88;
Lz = 6.99;
S = 2*Lx*Ly + 2*Lx*Lz + 2*Ly*Lz;  %surface area of walls
V = Lx*Ly*Lz;                       %volume of the room
T_60 = [7.32 7.11 6.45 6.46 7.13 7.39 7.59 7.41 7.33 7.38 6.60 5.60 4.84 4.08 3.42 2.78 2.18 1.58
        1.04 0.82 0.60];
A = (55.26/c)*(V./T_60);          %equivalent absorption area of the room (m^2)

%% Load in the ED data
loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
      Thesis Research\OFFICIAL THESIS FILES\Reverb Chamber\Reverb GED 2-Point In Situ Tests
      (9_13_13)\Reverb GED In Situ Data (9_13_13)\';

if Angle_num == 1
    distances = [0.94  2.332  4.084]; %distances of each ED probe along the given angle
    r_ED_probe_1 = distances(1);
    r_ED_probe_2 = distances(2);
    r_ED_probe_3 = distances(3);
    alt = 0;                          %altitude of angle (rounded to nearest 5 degrees)
    azimuth = 270;                    %azimuth angle (take 270 minus value in the array)(rounded
                                      %to the nearest 5 degrees)
    if Source == 1
        data_b_1 = 'Dodec Angle 1 Pos 1 Background.mat';
        data_1 =   'Dodec Angle 1 Pos 1.mat';
        data_b_2 = 'Dodec Angle 1 Pos 2 Background.mat';
        data_2 =   'Dodec Angle 1 Pos 2.mat';
        data_b_3 = 'Dodec Angle 1 Pos 3 Background.mat';
        data_3 =   'Dodec Angle 1 Pos 3.mat';
    end
    if Source == 2
        data_b_1 = 'SD Angle 1 Pos 1 Background.mat';
        data_1 =   'SD Angle 1 Pos 1.mat';
        data_b_2 = 'SD Angle 1 Pos 2 Background.mat';
        data_2 =   'SD Angle 1 Pos 2.mat';
        data_b_3 = 'SD Angle 1 Pos 3 Background.mat';
        data_3 =   'SD Angle 1 Pos 3.mat';
    end
    if Source == 3
        data_b_1 = 'Horn Angle 1 Pos 1 Background.mat';
        data_1 =   'Horn Angle 1 Pos 1.mat';
        data_b_2 = 'Horn Angle 1 Pos 2 Background.mat';
        data_2 =   'Horn Angle 1 Pos 2.mat';
        data_b_3 = 'Horn Angle 1 Pos 3 Background.mat';
        data_3 =   'Horn Angle 1 Pos 3.mat';
    end
    if Ref_Source == 1
        ref_data_b_1 = 'Dodec Angle 1 Pos 1 Background.mat';
        ref_data_1 =   'Dodec Angle 1 Pos 1.mat';
        ref_data_b_2 = 'Dodec Angle 1 Pos 2 Background.mat';
        ref_data_2 =   'Dodec Angle 1 Pos 2.mat';
        ref_data_b_3 = 'Dodec Angle 1 Pos 3 Background.mat';
        ref_data_3 =   'Dodec Angle 1 Pos 3.mat';
    end
    if Ref_Source == 2
        ref_data_b_1 = 'SD Angle 1 Pos 1 Background.mat';
```

```
        ref_data_1 =   'SD Angle 1 Pos 1.mat';
        ref_data_b_2 = 'SD Angle 1 Pos 2 Background.mat';
        ref_data_2 =   'SD Angle 1 Pos 2.mat';
        ref_data_b_3 = 'SD Angle 1 Pos 3 Background.mat';
        ref_data_3 =   'SD Angle 1 Pos 3.mat';
    end
    if Ref_Source == 3
        ref_data_b_1 = 'Horn Angle 1 Pos 1 Background.mat';
        ref_data_1 =   'Horn Angle 1 Pos 1.mat';
        ref_data_b_2 = 'Horn Angle 1 Pos 2 Background.mat';
        ref_data_2 =   'Horn Angle 1 Pos 2.mat';
        ref_data_b_3 = 'Horn Angle 1 Pos 3 Background.mat';
        ref_data_3 =   'Horn Angle 1 Pos 3.mat';
    end
end

if Angle_num == 2
    distances = [1.05  2.205  3.525]; %distances of each ED probe along the given angle
    r_ED_probe_1 = distances(1);
    r_ED_probe_2 = distances(2);
    r_ED_probe_3 = distances(3);
    alt = 0;                          %altitude of angle (rounded to nearest 5 degrees)
    azimuth = -45;                    %azimuth angle (take 270 minus value in the array)(rounded
                                      %to the nearest 5 degrees)
    if Source == 1
        data_b_1 = 'Dodec Angle 2 Pos 1 Background.mat';
        data_1 =   'Dodec Angle 2 Pos 1.mat';
        data_b_2 = 'Dodec Angle 2 Pos 2 Background.mat';
        data_2 =   'Dodec Angle 2 Pos 2.mat';
        data_b_3 = 'Dodec Angle 2 Pos 3 Background.mat';
        data_3 =   'Dodec Angle 2 Pos 3.mat';
    end
    if Source == 2
        data_b_1 = 'SD Angle 2 Pos 1 Background.mat';
        data_1 =   'SD Angle 2 Pos 1.mat';
        data_b_2 = 'SD Angle 2 Pos 2 Background.mat';
        data_2 =   'SD Angle 2 Pos 2.mat';
        data_b_3 = 'SD Angle 2 Pos 3 Background.mat';
        data_3 =   'SD Angle 2 Pos 3.mat';
    end
    if Source == 3
        data_b_1 = 'Horn Angle 2 Pos 1 Background.mat';
        data_1 =   'Horn Angle 2 Pos 1.mat';
        data_b_2 = 'Horn Angle 2 Pos 2 Background.mat';
        data_2 =   'Horn Angle 2 Pos 2.mat';
        data_b_3 = 'Horn Angle 2 Pos 3 Background.mat';
        data_3 =   'Horn Angle 2 Pos 3.mat';
    end
    if Ref_Source == 1
        ref_data_b_1 = 'Dodec Angle 2 Pos 1 Background.mat';
        ref_data_1 =   'Dodec Angle 2 Pos 1.mat';
        ref_data_b_2 = 'Dodec Angle 2 Pos 2 Background.mat';
        ref_data_2 =   'Dodec Angle 2 Pos 2.mat';
        ref_data_b_3 = 'Dodec Angle 2 Pos 3 Background.mat';
        ref_data_3 =   'Dodec Angle 2 Pos 3.mat';
    end
    if Ref_Source == 2
        ref_data_b_1 = 'SD Angle 2 Pos 1 Background.mat';
        ref_data_1 =   'SD Angle 2 Pos 1.mat';
        ref_data_b_2 = 'SD Angle 2 Pos 2 Background.mat';
        ref_data_2 =   'SD Angle 2 Pos 2.mat';
        ref_data_b_3 = 'SD Angle 2 Pos 3 Background.mat';
        ref_data_3 =   'SD Angle 2 Pos 3.mat';
    end
    if Ref_Source == 3
        ref_data_b_1 = 'Horn Angle 2 Pos 1 Background.mat';
        ref_data_1 =   'Horn Angle 2 Pos 1.mat';
        ref_data_b_2 = 'Horn Angle 2 Pos 2 Background.mat';
        ref_data_2 =   'Horn Angle 2 Pos 2.mat';
        ref_data_b_3 = 'Horn Angle 2 Pos 3 Background.mat';
        ref_data_3 =   'Horn Angle 2 Pos 3.mat';
```

```
    end
end

if Angle_num == 3
    distances = [1.135  2.405]; %distances of each ED probe along the given angle
    r_ED_probe_1 = distances(1);
    r_ED_probe_2 = distances(2);
    alt = 0;                    %altitude of angle (rounded to nearest 5 degrees)
    azimuth = 0;                %azimuth angle (take 270 minus value in the array)(rounded to the
                                %nearest 5 degrees)
    if Source == 1
        data_b_1 = 'Dodec Angle 3 Pos 1 Background.mat';
        data_1 =   'Dodec Angle 3 Pos 1.mat';
        data_b_2 = 'Dodec Angle 3 Pos 2 Background.mat';
        data_2 =   'Dodec Angle 3 Pos 2.mat';
    end
    if Source == 2
        data_b_1 = 'SD Angle 3 Pos 1 Background.mat';
        data_1 =   'SD Angle 3 Pos 1.mat';
        data_b_2 = 'SD Angle 3 Pos 2 Background.mat';
        data_2 =   'SD Angle 3 Pos 2.mat';
    end
    if Source == 3
        data_b_1 = 'Horn Angle 3 Pos 1 Background.mat';
        data_1 =   'Horn Angle 3 Pos 1.mat';
        data_b_2 = 'Horn Angle 3 Pos 2 Background.mat';
        data_2 =   'Horn Angle 3 Pos 2.mat';
    end
    if Ref_Source == 1
        ref_data_b_1 = 'Dodec Angle 3 Pos 1 Background.mat';
        ref_data_1 =   'Dodec Angle 3 Pos 1.mat';
        ref_data_b_2 = 'Dodec Angle 3 Pos 2 Background.mat';
        ref_data_2 =   'Dodec Angle 3 Pos 2.mat';
    end
    if Ref_Source == 2
        ref_data_b_1 = 'SD Angle 3 Pos 1 Background.mat';
        ref_data_1 =   'SD Angle 3 Pos 1.mat';
        ref_data_b_2 = 'SD Angle 3 Pos 2 Background.mat';
        ref_data_2 =   'SD Angle 3 Pos 2.mat';
    end
    if Ref_Source == 3
        ref_data_b_1 = 'Horn Angle 3 Pos 1 Background.mat';
        ref_data_1 =   'Horn Angle 3 Pos 1.mat';
        ref_data_b_2 = 'Horn Angle 3 Pos 2 Background.mat';
        ref_data_2 =   'Horn Angle 3 Pos 2.mat';
    end
end

%% Calculate the 1/3 octave band frequency-dependent room constant, R (Based on T60 Data - Not In
%  Situ R)

%Generate the Total Absorption Coefficient (Eq 1-2.6.7 of Phscs 661 notes)
h = hr*(Pr/Pa)*10^(-6.8346*(T01/T)^(1.261)+4.6151); %the molar concentration of water vapor (%)
fro = (Pa/Pr)*(24+4.04e4*h*((0.02+h)/(0.391+h)));   %the relaxation frequency of oxygen
frN = (Pa/Pr)*(T0/T)^(1/2)*(9+280*h*exp(-4.170*((T0/T)^(1/3)-1))); %the relaxation frequency of
                                                                   %nitrogen
alpha_prop = freq_model.^2.*((1.84e-11*(Pr/Pa)*(T/T0)^(1/2)) + (T0/T)^(5/2).*(((0.01275*exp(-
             2239.1/T))./(fro+(freq_model.^2/fro)))+((0.1068*exp(-
             3352.0/T))./(frN+(freq_model.^2/frN)))));  %the total absorption due to wave
                                                        %propagation in air
%Calculate the room constant, R

if absorp_form == 1                                  %Sabine absorption formulation
    if air_absorp == 0                               %no air absorption
        alpha_wall = (0.161*V)./(S*T_60);            %the Sabine absorption coefficient of
                                                     %the wall (neglecting air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S); %the total absorption coefficient in the
                                                     %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';   %the 1/3 octave band frequency-dependent
                                                     %room constant
    end
```

```
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = ((0.161*V)./(S*T_60)) - ((8*alpha_prop*V)/S); %the Sabine absorption
                                                                        %coefficient of the wall
                                                                        %(considering the effects of
                                                                        %air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);                  %the total absorption in the
                                                                        %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                      %the 1/3 octave band
                                                                        %frequency-dependent room
                                                                        %constant (including air
                                                                        %absorption)
    end
end
if absorp_form == 2                                     %Eyring absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = 1-exp(-(0.161*V)./(S*T_60));       %The Eyring absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S); %the total absorption in the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant (no air absorption)
    end
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = 1-exp((1/S)*(8*alpha_prop*V-((0.161*V)./T_60))); %the Eyring absorption
coefficient of the wall (considering air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);                  %the total absorption in
                                                                        %the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                      %the 1/3 octave band
                                                                        %frequency-dependent room
                                                                        %constant (including air
                                                                        %absorption)
    end
end

%% Load in the data from the anechoic chamber and create 1/3 octave band directivity factors for
%  the Reference Source and Source Under Test (Not In Situ Directivity Factors)
theta = (0:5:180)'*pi/180;     %polar angles of the measurement sphere
phi = (0:5:355)*pi/180;        %azimuth angles of the measurement sphere
%Load in the FRF data from the SOURCE UNDER TEST

if Source == 1    %Dodec
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Dodec Directivity\Data\';
    load([loc_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 2    %Single Driver
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 3    %Horn
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Horn Directivity\Data\';
    load([loc_Dir_Fact 'Horn_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end

%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
```

```
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;                 %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
                                              %octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
                        [~,summed_weighted_Oct3_H2_sq] =
                        MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h
theta_insitu = 90 - alt;
phi_insitu = 270 - azimuth;

if phi_insitu < 0
    phi_insitu = 360 + phi_insitu;
end

theta_ind = (theta_insitu+5)/5;
phi_ind = (phi_insitu+5)/5;
gamma_Official = squeeze(Dir_Fact_Oct3(theta_ind,phi_ind,:)).';

%Load in the FRF data from the REFERECE SOURCE

if Ref_Source == 1    %Dodec
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Dodec
                        Directivity\Data\';
    load([loc_ref_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Ref_Source == 2    %Single Driver
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_ref_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Ref_Source == 3    %Horn
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Horn
                        Directivity\Data\';
    load([loc_ref_Dir_Fact 'Horn_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end

%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;                 %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
                                              %octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
```

```
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
                        [~,summed_weighted_Oct3_H2_sq] =
                        MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h
theta_insitu_ref = 90 - alt;
phi_insitu_ref = 270 - azimuth;

if phi_insitu_ref < 0
    phi_insitu_ref = 360 + phi_insitu_ref;
end

theta_ind = (theta_insitu_ref+5)/5;
phi_ind = (phi_insitu_ref+5)/5;
gamma_Official_ref = squeeze(Dir_Fact_Oct3(theta_ind,phi_ind,:)).';

%% Calculate the Schroeder Frequency of the Room
f_Sch = round( sqrt( (c^3)/(4*log(10)) )*sqrt( mean(T_60)/V ) );

%% Calculate the Angle-Dependent Critical Distance (Distance where the direct and reverberant
%  field energy are approximately equal)
r_c = sqrt((gamma_Official.*R_Oct3.')/(16*pi));
r_c_ref = sqrt((gamma_Official_ref.*R_Oct3.')/(16*pi));
r_c_Mat = [freq_model;
           r_c;
           r_c_ref]
r_Close = distances(r_ED_close);
r_Far = distances(r_ED_far);

%% CALUCULATE THE SOUND POWER USING THE IN SITU DIRECTIVITY FACTORS AND THE HOPKINS/STRYKER %
%  EQUATION

%% Calculate the four acoustic energy density quantities (PED, KED, TED, and GED) in 1/3 octave
%   bands (from the 6-Mic Probe)
%Load in the Background Noise data (Narrowband Squared Values)

if Angle_num == 3
    num_pos = 2;
else
    num_pos = 3;
end

%Source Under Test/Background
for ii = 1:num_pos
    temp = eval(['data_b_' num2str(ii)]);
    load([loc temp])
    eval(['p' num2str(ii) '_1_1_bg = MAT_001_Autopower_Mic_1(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_2_2_bg = MAT_002_Autopower_Mic_2(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_3_3_bg = MAT_003_Autopower_Mic_3(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_4_4_bg = MAT_004_Autopower_Mic_4(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_5_5_bg = MAT_005_Autopower_Mic_5(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_6_6_bg = MAT_006_Autopower_Mic_6(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_2_1_bg = MAT_011_Crosspower_Mic_2(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_4_3_bg = MAT_019_Crosspower_Mic_4(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_6_5_bg = MAT_027_Crosspower_Mic_6(:,2)/ENBW'';']);
    eval('f = MAT_027_Crosspower_Mic_6(:,1)'';');
    clear  MAT_001_Autopower_Mic_1  MAT_002_Autopower_Mic_2   MAT_003_Autopower_Mic_3
           MAT_004_Autopower_Mic_4  MAT_005_Autopower_Mic_5
    clear  MAT_006_Autopower_Mic_6  MAT_011_Crosspower_Mic_2  MAT_019_Crosspower_Mic_4
           MAT_027_Crosspower_Mic_6
end

%Reference Source/Background
for ii = 1:num_pos
```

```
    ref_temp = eval(['ref_data_b_' num2str(ii)]);
    load([loc ref_temp])
    eval(['ref_p' num2str(ii) '_1_1_bg = MAT_001_Autopower_Mic_1(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_2_2_bg = MAT_002_Autopower_Mic_2(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_3_3_bg = MAT_003_Autopower_Mic_3(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_4_4_bg = MAT_004_Autopower_Mic_4(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_5_5_bg = MAT_005_Autopower_Mic_5(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_6_6_bg = MAT_006_Autopower_Mic_6(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_2_1_bg = MAT_011_Crosspower_Mic_2(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_4_3_bg = MAT_019_Crosspower_Mic_4(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_6_5_bg = MAT_027_Crosspower_Mic_6(:,2)/ENBW'';']);
    clear  MAT_001_Autopower_Mic_1  MAT_002_Autopower_Mic_2   MAT_003_Autopower_Mic_3
           MAT_004_Autopower_Mic_4  MAT_005_Autopower_Mic_5
    clear  MAT_006_Autopower_Mic_6  MAT_011_Crosspower_Mic_2  MAT_019_Crosspower_Mic_4
           MAT_027_Crosspower_Mic_6
end

%Load in the Uncorrected Measurement data (Narrowband Squared Values)
%Source Under Test/Uncorrected
for ii = 1:num_pos
    temp = eval(['data_' num2str(ii)]);
    load([loc temp])
    eval(['p' num2str(ii) '_1_1_uc = MAT_001_Autopower_Mic_1(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_2_2_uc = MAT_002_Autopower_Mic_2(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_3_3_uc = MAT_003_Autopower_Mic_3(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_4_4_uc = MAT_004_Autopower_Mic_4(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_5_5_uc = MAT_005_Autopower_Mic_5(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_6_6_uc = MAT_006_Autopower_Mic_6(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_2_1_uc = MAT_011_Crosspower_Mic_2(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_4_3_uc = MAT_019_Crosspower_Mic_4(:,2)/ENBW'';']);
    eval(['p' num2str(ii) '_6_5_uc = MAT_027_Crosspower_Mic_6(:,2)/ENBW'';']);
    clear  MAT_001_Autopower_Mic_1  MAT_002_Autopower_Mic_2   MAT_003_Autopower_Mic_3
           MAT_004_Autopower_Mic_4  MAT_005_Autopower_Mic_5
    clear  MAT_006_Autopower_Mic_6  MAT_011_Crosspower_Mic_2  MAT_019_Crosspower_Mic_4
           MAT_027_Crosspower_Mic_6
end
%Reference Source/Uncorrected
for ii = 1:num_pos
    ref_temp = eval(['ref_data_' num2str(ii)]);
    load([loc ref_temp])
    eval(['ref_p' num2str(ii) '_1_1_uc = MAT_001_Autopower_Mic_1(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_2_2_uc = MAT_002_Autopower_Mic_2(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_3_3_uc = MAT_003_Autopower_Mic_3(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_4_4_uc = MAT_004_Autopower_Mic_4(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_5_5_uc = MAT_005_Autopower_Mic_5(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_6_6_uc = MAT_006_Autopower_Mic_6(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_2_1_uc = MAT_011_Crosspower_Mic_2(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_4_3_uc = MAT_019_Crosspower_Mic_4(:,2)/ENBW'';']);
    eval(['ref_p' num2str(ii) '_6_5_uc = MAT_027_Crosspower_Mic_6(:,2)/ENBW'';']);
    clear  MAT_001_Autopower_Mic_1  MAT_002_Autopower_Mic_2   MAT_003_Autopower_Mic_3
           MAT_004_Autopower_Mic_4  MAT_005_Autopower_Mic_5
    clear  MAT_006_Autopower_Mic_6  MAT_011_Crosspower_Mic_2  MAT_019_Crosspower_Mic_4
           MAT_027_Crosspower_Mic_6
end
if Angle_num == 3
    SUT_bg_data_pos_1 =
cat(1,p1_1_1_bg.',p1_2_2_bg.',p1_3_3_bg.',p1_4_4_bg.',p1_5_5_bg.',p1_6_6_bg.',p1_2_1_bg.',p1_4_3_
                        bg.',p1_6_5_bg.');
    SUT_bg_data_pos_2 =
cat(1,p2_1_1_bg.',p2_2_2_bg.',p2_3_3_bg.',p2_4_4_bg.',p2_5_5_bg.',p2_6_6_bg.',p2_2_1_bg.',p2_4_3_
                        bg.',p2_6_5_bg.');
    SUT_bg_data_mat = cat(3,SUT_bg_data_pos_1,SUT_bg_data_pos_2);
    SUT_uc_data_pos_1 =
cat(1,p1_1_1_uc.',p1_2_2_uc.',p1_3_3_uc.',p1_4_4_uc.',p1_5_5_uc.',p1_6_6_uc.',p1_2_1_uc.',p1_4_3_
                        uc.',p1_6_5_uc.');
    SUT_uc_data_pos_2 =
cat(1,p2_1_1_uc.',p2_2_2_uc.',p2_3_3_uc.',p2_4_4_uc.',p2_5_5_uc.',p2_6_6_uc.',p2_2_1_uc.',p2_4_3_
                        uc.',p2_6_5_uc.');
    SUT_uc_data_mat = cat(3,SUT_uc_data_pos_1,SUT_uc_data_pos_2);
```

```
    RS_bg_data_pos_1 =
cat(1,ref_p1_1_1_bg.',ref_p1_2_2_bg.',ref_p1_3_3_bg.',ref_p1_4_4_bg.',ref_p1_5_5_bg.',ref_p1_6_6_
                        bg.',ref_p1_2_1_bg.',ref_p1_4_3_bg.',ref_p1_6_5_bg.');
    RS_bg_data_pos_2 =
cat(1,ref_p2_1_1_bg.',ref_p2_2_2_bg.',ref_p2_3_3_bg.',ref_p2_4_4_bg.',ref_p2_5_5_bg.',ref_p2_6_6_
                        bg.',ref_p2_2_1_bg.',ref_p2_4_3_bg.',ref_p2_6_5_bg.');
    RS_bg_data_mat = cat(3,RS_bg_data_pos_1,RS_bg_data_pos_2);
    RS_uc_data_pos_1 =
cat(1,ref_p1_1_1_uc.',ref_p1_2_2_uc.',ref_p1_3_3_uc.',ref_p1_4_4_uc.',ref_p1_5_5_uc.',ref_p1_6_6_
                        uc.',ref_p1_2_1_uc.',ref_p1_4_3_uc.',ref_p1_6_5_uc.');
    RS_uc_data_pos_2 =
cat(1,ref_p2_1_1_uc.',ref_p2_2_2_uc.',ref_p2_3_3_uc.',ref_p2_4_4_uc.',ref_p2_5_5_uc.',ref_p2_6_6_
                        uc.',ref_p2_2_1_uc.',ref_p2_4_3_uc.',ref_p2_6_5_uc.');
    RS_uc_data_mat = cat(3,RS_uc_data_pos_1,RS_uc_data_pos_2);
else
    SUT_bg_data_pos_1 =
cat(1,p1_1_1_bg.',p1_2_2_bg.',p1_3_3_bg.',p1_4_4_bg.',p1_5_5_bg.',p1_6_6_bg.',p1_2_1_bg.',p1_4_3_
                        bg.',p1_6_5_bg.');
    SUT_bg_data_pos_2 =
cat(1,p2_1_1_bg.',p2_2_2_bg.',p2_3_3_bg.',p2_4_4_bg.',p2_5_5_bg.',p2_6_6_bg.',p2_2_1_bg.',p2_4_3_
                        bg.',p2_6_5_bg.');
    SUT_bg_data_pos_3 =
cat(1,p3_1_1_bg.',p3_2_2_bg.',p3_3_3_bg.',p3_4_4_bg.',p3_5_5_bg.',p3_6_6_bg.',p3_2_1_bg.',p3_4_3_
                        bg.',p3_6_5_bg.');
    SUT_bg_data_mat = cat(3,SUT_bg_data_pos_1,SUT_bg_data_pos_2,SUT_bg_data_pos_3);

    SUT_uc_data_pos_1 =
cat(1,p1_1_1_uc.',p1_2_2_uc.',p1_3_3_uc.',p1_4_4_uc.',p1_5_5_uc.',p1_6_6_uc.',p1_2_1_uc.',p1_4_3_
                        uc.',p1_6_5_uc.');
    SUT_uc_data_pos_2 =
cat(1,p2_1_1_uc.',p2_2_2_uc.',p2_3_3_uc.',p2_4_4_uc.',p2_5_5_uc.',p2_6_6_uc.',p2_2_1_uc.',p2_4_3_
                        uc.',p2_6_5_uc.');
    SUT_uc_data_pos_3 =
cat(1,p3_1_1_uc.',p3_2_2_uc.',p3_3_3_uc.',p3_4_4_uc.',p3_5_5_uc.',p3_6_6_uc.',p3_2_1_uc.',p3_4_3_
                        uc.',p3_6_5_uc.');
    SUT_uc_data_mat = cat(3,SUT_uc_data_pos_1,SUT_uc_data_pos_2,SUT_uc_data_pos_3);
    RS_bg_data_pos_1 =
cat(1,ref_p1_1_1_bg.',ref_p1_2_2_bg.',ref_p1_3_3_bg.',ref_p1_4_4_bg.',ref_p1_5_5_bg.',ref_p1_6_6_
                        bg.',ref_p1_2_1_bg.',ref_p1_4_3_bg.',ref_p1_6_5_bg.');
    RS_bg_data_pos_2 =
cat(1,ref_p2_1_1_bg.',ref_p2_2_2_bg.',ref_p2_3_3_bg.',ref_p2_4_4_bg.',ref_p2_5_5_bg.',ref_p2_6_6_
                        bg.',ref_p2_2_1_bg.',ref_p2_4_3_bg.',ref_p2_6_5_bg.');
    RS_bg_data_pos_3 =
cat(1,ref_p3_1_1_bg.',ref_p3_2_2_bg.',ref_p3_3_3_bg.',ref_p3_4_4_bg.',ref_p3_5_5_bg.',ref_p3_6_6_
                       bg.',ref_p3_2_1_bg.',ref_p3_4_3_bg.',ref_p3_6_5_bg.');
    RS_bg_data_mat = cat(3,RS_bg_data_pos_1,RS_bg_data_pos_2,RS_bg_data_pos_3);
    RS_uc_data_pos_1 =
cat(1,ref_p1_1_1_uc.',ref_p1_2_2_uc.',ref_p1_3_3_uc.',ref_p1_4_4_uc.',ref_p1_5_5_uc.',ref_p1_6_6_
                        uc.',ref_p1_2_1_uc.',ref_p1_4_3_uc.',ref_p1_6_5_uc.');
    RS_uc_data_pos_2 =
cat(1,ref_p2_1_1_uc.',ref_p2_2_2_uc.',ref_p2_3_3_uc.',ref_p2_4_4_uc.',ref_p2_5_5_uc.',ref_p2_6_6_
                        uc.',ref_p2_2_1_uc.',ref_p2_4_3_uc.',ref_p2_6_5_uc.');
    RS_uc_data_pos_3 =
cat(1,ref_p3_1_1_uc.',ref_p3_2_2_uc.',ref_p3_3_3_uc.',ref_p3_4_4_uc.',ref_p3_5_5_uc.',ref_p3_6_6_
                        uc.',ref_p3_2_1_uc.',ref_p3_4_3_uc.',ref_p3_6_5_uc.');
    RS_uc_data_mat = cat(3,RS_uc_data_pos_1,RS_uc_data_pos_2,RS_uc_data_pos_3);
end

%% Calculate PED (Narrowband Values)
%Source Under Test/Background
SUT_bg_PED_mat = zeros(1,length(SUT_bg_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    PED_1_2_bg = (1/(4*rho*c^2)) * ( SUT_bg_data_mat(1,:,ii) + SUT_bg_data_mat(2,:,ii) +
                 SUT_bg_data_mat(7,:,ii) + conj(SUT_bg_data_mat(7,:,ii)) )/2;
    PED_3_4_bg = (1/(4*rho*c^2)) * ( SUT_bg_data_mat(3,:,ii) + SUT_bg_data_mat(4,:,ii) +
                 SUT_bg_data_mat(8,:,ii) + conj(SUT_bg_data_mat(8,:,ii)) )/2;
    PED_5_6_bg = (1/(4*rho*c^2)) * ( SUT_bg_data_mat(5,:,ii) + SUT_bg_data_mat(6,:,ii) +
                 SUT_bg_data_mat(9,:,ii) + conj(SUT_bg_data_mat(9,:,ii)) )/2;
    SUT_bg_PED_mat(:,:,ii) = (( PED_1_2_bg + PED_3_4_bg + PED_5_6_bg )/3);
end
```

```
%Source Under Test/Uncorrected
SUT_uc_PED_mat = zeros(1,length(SUT_uc_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    PED_1_2_uc = (1/(4*rho*c^2)) * ( SUT_uc_data_mat(1,:,ii) + SUT_uc_data_mat(2,:,ii) +
                 SUT_uc_data_mat(7,:,ii) + conj(SUT_uc_data_mat(7,:,ii)) )/2;
    PED_3_4_uc = (1/(4*rho*c^2)) * ( SUT_uc_data_mat(3,:,ii) + SUT_uc_data_mat(4,:,ii) +
                 SUT_uc_data_mat(8,:,ii) + conj(SUT_uc_data_mat(8,:,ii)) )/2;
    PED_5_6_uc = (1/(4*rho*c^2)) * ( SUT_uc_data_mat(5,:,ii) + SUT_uc_data_mat(6,:,ii) +
                 SUT_uc_data_mat(9,:,ii) + conj(SUT_uc_data_mat(9,:,ii)) )/2;
    SUT_uc_PED_mat(:,:,ii) = (( PED_1_2_uc + PED_3_4_uc + PED_5_6_uc )/3);
end

%Reference Source/Background
RS_bg_PED_mat = zeros(1,length(RS_bg_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    PED_1_2_bg = (1/(4*rho*c^2)) * ( RS_bg_data_mat(1,:,ii) + RS_bg_data_mat(2,:,ii) +
                 RS_bg_data_mat(7,:,ii) + conj(RS_bg_data_mat(7,:,ii)) )/2;
    PED_3_4_bg = (1/(4*rho*c^2)) * ( RS_bg_data_mat(3,:,ii) + RS_bg_data_mat(4,:,ii) +
                 RS_bg_data_mat(8,:,ii) + conj(RS_bg_data_mat(8,:,ii)) )/2;
    PED_5_6_bg = (1/(4*rho*c^2)) * ( RS_bg_data_mat(5,:,ii) + RS_bg_data_mat(6,:,ii) +
                 RS_bg_data_mat(9,:,ii) + conj(RS_bg_data_mat(9,:,ii)) )/2;
    RS_bg_PED_mat(:,:,ii) = (( PED_1_2_bg + PED_3_4_bg + PED_5_6_bg )/3);
end

%Reference Source/Uncorrected
RS_uc_PED_mat = zeros(1,length(RS_uc_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    PED_1_2_uc = (1/(4*rho*c^2)) * ( RS_uc_data_mat(1,:,ii) + RS_uc_data_mat(2,:,ii) +
                 RS_uc_data_mat(7,:,ii) + conj(RS_uc_data_mat(7,:,ii)) )/2;
    PED_3_4_uc = (1/(4*rho*c^2)) * ( RS_uc_data_mat(3,:,ii) + RS_uc_data_mat(4,:,ii) +
                 RS_uc_data_mat(8,:,ii) + conj(RS_uc_data_mat(8,:,ii)) )/2;
    PED_5_6_uc = (1/(4*rho*c^2)) * ( RS_uc_data_mat(5,:,ii) + RS_uc_data_mat(6,:,ii) +
                 RS_uc_data_mat(9,:,ii) + conj(RS_uc_data_mat(9,:,ii)) )/2;
    RS_uc_PED_mat(:,:,ii) = (( PED_1_2_uc + PED_3_4_uc + PED_5_6_uc )/3);
end

%Create NOISE-CORRECTED PED data (Narrowband Squared Values)
%Source Under Test
SUT_PED_mat = SUT_uc_PED_mat - SUT_bg_PED_mat;
%Reference Source
RS_PED_mat = RS_uc_PED_mat - RS_bg_PED_mat;

%% Calculate KED (Narrowband Values)
omega = 2*pi*f;
%Source Under Test/Background
SUT_bg_KED_mat = zeros(1,length(SUT_bg_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    KED_Ux_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_bg_data_mat(1,:,ii) +
             SUT_bg_data_mat(2,:,ii) - SUT_bg_data_mat(7,:,ii) - conj(SUT_bg_data_mat(7,:,ii)) );
    KED_Uy_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_bg_data_mat(3,:,ii) +
             SUT_bg_data_mat(4,:,ii) - SUT_bg_data_mat(8,:,ii) - conj(SUT_bg_data_mat(8,:,ii)) );
    KED_Uz_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_bg_data_mat(5,:,ii) +
             SUT_bg_data_mat(6,:,ii) - SUT_bg_data_mat(9,:,ii) - conj(SUT_bg_data_mat(9,:,ii)) );
    SUT_bg_KED_mat(:,:,ii) = (KED_Ux_bg + KED_Uy_bg + KED_Uz_bg);
end

%Source Under Test/Uncorrected
SUT_uc_KED_mat = zeros(1,length(SUT_uc_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    KED_Ux_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_uc_data_mat(1,:,ii) +
             SUT_uc_data_mat(2,:,ii) - SUT_uc_data_mat(7,:,ii) - conj(SUT_uc_data_mat(7,:,ii)) );
    KED_Uy_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_uc_data_mat(3,:,ii) +
             SUT_uc_data_mat(4,:,ii) - SUT_uc_data_mat(8,:,ii) - conj(SUT_uc_data_mat(8,:,ii)) );
    KED_Uz_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( SUT_uc_data_mat(5,:,ii) +
             SUT_uc_data_mat(6,:,ii) - SUT_uc_data_mat(9,:,ii) - conj(SUT_uc_data_mat(9,:,ii)) );
    SUT_uc_KED_mat(:,:,ii) = (KED_Ux_uc + KED_Uy_uc + KED_Uz_uc);
end

%Reference Source/Background
RS_bg_KED_mat = zeros(1,length(RS_bg_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
```

```
    KED_Ux_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_bg_data_mat(1,:,ii) +
                RS_bg_data_mat(2,:,ii) - RS_bg_data_mat(7,:,ii) - conj(RS_bg_data_mat(7,:,ii)) );
    KED_Uy_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_bg_data_mat(3,:,ii) +
                RS_bg_data_mat(4,:,ii) - RS_bg_data_mat(8,:,ii) - conj(RS_bg_data_mat(8,:,ii)) );
    KED_Uz_bg = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_bg_data_mat(5,:,ii) +
                RS_bg_data_mat(6,:,ii) - RS_bg_data_mat(9,:,ii) - conj(RS_bg_data_mat(9,:,ii)) );
    RS_bg_KED_mat(:,:,ii) = (KED_Ux_bg + KED_Uy_bg + KED_Uz_bg);
end

%Reference Source/Uncorrected
RS_uc_KED_mat = zeros(1,length(RS_uc_data_mat(1,:,1)),num_pos);
for ii = 1:num_pos
    KED_Ux_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_uc_data_mat(1,:,ii) +
                RS_uc_data_mat(2,:,ii) - RS_uc_data_mat(7,:,ii) - conj(RS_uc_data_mat(7,:,ii)) );
    KED_Uy_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_uc_data_mat(3,:,ii) +
                RS_uc_data_mat(4,:,ii) - RS_uc_data_mat(8,:,ii) - conj(RS_uc_data_mat(8,:,ii)) );
    KED_Uz_uc = ( 1./(4*omega.^2*rho*spacing^2) ) .* ( RS_uc_data_mat(5,:,ii) +
                RS_uc_data_mat(6,:,ii) - RS_uc_data_mat(9,:,ii) - conj(RS_uc_data_mat(9,:,ii)) );
    RS_uc_KED_mat(:,:,ii) = (KED_Ux_uc + KED_Uy_uc + KED_Uz_uc);
end

%Create NOISE-CORRECTED KED data
%Source Under Test
SUT_KED_mat = SUT_uc_KED_mat - SUT_bg_KED_mat;
%Reference Source
RS_KED_mat = RS_uc_KED_mat - RS_bg_KED_mat;

%% Calculate TED (Narrowband Values)
%Source Under Test
SUT_TED_mat = SUT_PED_mat + SUT_KED_mat;
%Reference Source
RS_TED_mat = RS_PED_mat + RS_KED_mat;

%% Calculate GED (Narrowband Values)
%Source Under Test
SUT_GED_mat = (BETA*SUT_PED_mat) + (1-BETA)*SUT_KED_mat;
%Reference Source
RS_GED_mat = (BETA*RS_PED_mat) + (1-BETA)*RS_KED_mat;

%% Convert the narrow band data of all 4 ED values to 1/3 octave band values
if Angle_num == 3
    %Source Under Test
    [f_Oct3,SUT_PED_data_Oct3_1] = MyOct3Bands(f,SUT_PED_mat(1,:,1));
    [~,     SUT_PED_data_Oct3_2] = MyOct3Bands(f,SUT_PED_mat(1,:,2));
    [~,SUT_KED_data_Oct3_1] = MyOct3Bands(f,SUT_KED_mat(1,:,1));
    [~,SUT_KED_data_Oct3_2] = MyOct3Bands(f,SUT_KED_mat(1,:,2));
    [~,SUT_TED_data_Oct3_1] = MyOct3Bands(f,SUT_TED_mat(1,:,1));
    [~,SUT_TED_data_Oct3_2] = MyOct3Bands(f,SUT_TED_mat(1,:,2));
    [~,SUT_GED_data_Oct3_1] = MyOct3Bands(f,SUT_GED_mat(1,:,1));
    [~,SUT_GED_data_Oct3_2] = MyOct3Bands(f,SUT_GED_mat(1,:,2));
    SUT_ED_MAT_full_1 =
cat(1,SUT_PED_data_Oct3_1.',SUT_KED_data_Oct3_1.',SUT_TED_data_Oct3_1.',SUT_GED_data_Oct3_1.');
    SUT_ED_MAT_full_2 =
cat(1,SUT_PED_data_Oct3_2.',SUT_KED_data_Oct3_2.',SUT_TED_data_Oct3_2.',SUT_GED_data_Oct3_2.');
    %Reference Source
    [~,RS_PED_data_Oct3_1] = MyOct3Bands(f,RS_PED_mat(1,:,1));
    [~,RS_PED_data_Oct3_2] = MyOct3Bands(f,RS_PED_mat(1,:,2));
    [~,RS_KED_data_Oct3_1] = MyOct3Bands(f,RS_KED_mat(1,:,1));
    [~,RS_KED_data_Oct3_2] = MyOct3Bands(f,RS_KED_mat(1,:,2));
    [~,RS_TED_data_Oct3_1] = MyOct3Bands(f,RS_TED_mat(1,:,1));
    [~,RS_TED_data_Oct3_2] = MyOct3Bands(f,RS_TED_mat(1,:,2));
    [~,RS_GED_data_Oct3_1] = MyOct3Bands(f,RS_GED_mat(1,:,1));
    [~,RS_GED_data_Oct3_2] = MyOct3Bands(f,RS_GED_mat(1,:,2));
    RS_ED_MAT_full_1 =
cat(1,RS_PED_data_Oct3_1.',RS_KED_data_Oct3_1.',RS_TED_data_Oct3_1.',RS_GED_data_Oct3_1.');
    RS_ED_MAT_full_2 =
cat(1,RS_PED_data_Oct3_2.',RS_KED_data_Oct3_2.',RS_TED_data_Oct3_2.',RS_GED_data_Oct3_2.');
else
    %Source Under Test
    [f_Oct3,SUT_PED_data_Oct3_1] = MyOct3Bands(f,SUT_PED_mat(1,:,1));
    [~,     SUT_PED_data_Oct3_2] = MyOct3Bands(f,SUT_PED_mat(1,:,2));
```

```
    [~,     SUT_PED_data_Oct3_3] = MyOct3Bands(f,SUT_PED_mat(1,:,3));
    [~,SUT_KED_data_Oct3_1] = MyOct3Bands(f,SUT_KED_mat(1,:,1));
    [~,SUT_KED_data_Oct3_2] = MyOct3Bands(f,SUT_KED_mat(1,:,2));
    [~,SUT_KED_data_Oct3_3] = MyOct3Bands(f,SUT_KED_mat(1,:,3));
    [~,SUT_TED_data_Oct3_1] = MyOct3Bands(f,SUT_TED_mat(1,:,1));
    [~,SUT_TED_data_Oct3_2] = MyOct3Bands(f,SUT_TED_mat(1,:,2));
    [~,SUT_TED_data_Oct3_3] = MyOct3Bands(f,SUT_TED_mat(1,:,3));
    [~,SUT_GED_data_Oct3_1] = MyOct3Bands(f,SUT_GED_mat(1,:,1));
    [~,SUT_GED_data_Oct3_2] = MyOct3Bands(f,SUT_GED_mat(1,:,2));
    [~,SUT_GED_data_Oct3_3] = MyOct3Bands(f,SUT_GED_mat(1,:,3));
    SUT_ED_MAT_full_1 =
cat(1,SUT_PED_data_Oct3_1.',SUT_KED_data_Oct3_1.',SUT_TED_data_Oct3_1.',SUT_GED_data_Oct3_1.');
    SUT_ED_MAT_full_2 =
cat(1,SUT_PED_data_Oct3_2.',SUT_KED_data_Oct3_2.',SUT_TED_data_Oct3_2.',SUT_GED_data_Oct3_2.');
    SUT_ED_MAT_full_3 =
cat(1,SUT_PED_data_Oct3_3.',SUT_KED_data_Oct3_3.',SUT_TED_data_Oct3_3.',SUT_GED_data_Oct3_3.');
    %Reference Source
    [~,RS_PED_data_Oct3_1] = MyOct3Bands(f,RS_PED_mat(1,:,1));
    [~,RS_PED_data_Oct3_2] = MyOct3Bands(f,RS_PED_mat(1,:,2));
    [~,RS_PED_data_Oct3_3] = MyOct3Bands(f,RS_PED_mat(1,:,3));
    [~,RS_KED_data_Oct3_1] = MyOct3Bands(f,RS_KED_mat(1,:,1));
    [~,RS_KED_data_Oct3_2] = MyOct3Bands(f,RS_KED_mat(1,:,2));
    [~,RS_KED_data_Oct3_3] = MyOct3Bands(f,RS_KED_mat(1,:,3));
    [~,RS_TED_data_Oct3_1] = MyOct3Bands(f,RS_TED_mat(1,:,1));
    [~,RS_TED_data_Oct3_2] = MyOct3Bands(f,RS_TED_mat(1,:,2));
    [~,RS_TED_data_Oct3_3] = MyOct3Bands(f,RS_TED_mat(1,:,3));
    [~,RS_GED_data_Oct3_1] = MyOct3Bands(f,RS_GED_mat(1,:,1));
    [~,RS_GED_data_Oct3_2] = MyOct3Bands(f,RS_GED_mat(1,:,2));
    [~,RS_GED_data_Oct3_3] = MyOct3Bands(f,RS_GED_mat(1,:,3));
    RS_ED_MAT_full_1 =
cat(1,RS_PED_data_Oct3_1.',RS_KED_data_Oct3_1.',RS_TED_data_Oct3_1.',RS_GED_data_Oct3_1.');
    RS_ED_MAT_full_2 =
cat(1,RS_PED_data_Oct3_2.',RS_KED_data_Oct3_2.',RS_TED_data_Oct3_2.',RS_GED_data_Oct3_2.');
    RS_ED_MAT_full_3 =
cat(1,RS_PED_data_Oct3_3.',RS_KED_data_Oct3_3.',RS_TED_data_Oct3_3.',RS_GED_data_Oct3_3.');
end

%% Truncate the 1/3 octave band data to go from 100 Hz to 10 kHz
ind_l = findnearest(freq_model(1),f_Oct3,0);
ind_h = findnearest(freq_model(end),f_Oct3,0);
f_Oct3 = f_Oct3(ind_l:ind_h);

if Angle_num == 3
    %Source Under Test
    SUT_ED_MAT_1 = SUT_ED_MAT_full_1(:,ind_l:ind_h);
    SUT_ED_MAT_2 = SUT_ED_MAT_full_2(:,ind_l:ind_h);
    %Reference Source
    RS_ED_MAT_1 = RS_ED_MAT_full_1(:,ind_l:ind_h);
    RS_ED_MAT_2 = RS_ED_MAT_full_2(:,ind_l:ind_h);
else
    %Source Under Test
    SUT_ED_MAT_1 = SUT_ED_MAT_full_1(:,ind_l:ind_h);
    SUT_ED_MAT_2 = SUT_ED_MAT_full_2(:,ind_l:ind_h);
    SUT_ED_MAT_3 = SUT_ED_MAT_full_3(:,ind_l:ind_h);
    %Reference Source
    RS_ED_MAT_1 = RS_ED_MAT_full_1(:,ind_l:ind_h);
    RS_ED_MAT_2 = RS_ED_MAT_full_2(:,ind_l:ind_h);
    RS_ED_MAT_3 = RS_ED_MAT_full_3(:,ind_l:ind_h);
end

%% Calculate the Third-Octave Band IN SITU ROOM CONSTANT Using A Reference Source, Then The IN
SITU DIRECTIVITY FACTOR Of The Source, and From it The Sound Power (Using All Four EDs)!!!
%Calculate the constant b for each ED probe position
b = distances(r_ED_far)/distances(r_ED_close);
b_ref = distances(r_ED_far_ref)/distances(r_ED_close_ref);
R_in_situ = zeros(4,length(freq_model));
gamma = zeros(4,length(freq_model));
pow = zeros(4,length(freq_model));
L_w_In_Situ = zeros(4,length(freq_model));
Overall_Lw_In_Situ = zeros(4,1);
A_weight_L_w_In_Situ = zeros(4,length(freq_model));
```

```
Overall_A_weight_Lw_In_Situ = zeros(4,1);
C_weight_L_w_In_Situ = zeros(4,length(freq_model));
Overall_C_weight_Lw_In_Situ = zeros(4,1);

for ii = 1:4
    if FullCorrections == 0                %Does NOT Include Air Absorption
        %Calculate the Room Constant based on the Measurement Data from the Reference Source and
        %its Known Directivity Factor
        if r_ED_close_ref == 1
            if r_ED_far_ref == 2
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_1(ii,:)./RS_ED_MAT_2(ii,:))) ) ./ (
                            gamma_Official_ref.*((RS_ED_MAT_1(ii,:)./(RS_ED_MAT_2(ii,:)*b_ref^2))
                                - 1) );
            end
            if r_ED_far_ref == 3
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_1(ii,:)./RS_ED_MAT_3(ii,:))) ) ./ (
                            gamma_Official_ref.*((RS_ED_MAT_1(ii,:)./(RS_ED_MAT_3(ii,:)*b_ref^2))
                                - 1) );
            end
        end
        if r_ED_close_ref == 2
            if r_ED_far_ref == 3
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_2(ii,:)./RS_ED_MAT_3(ii,:))) ) ./ (
                            gamma_Official_ref.*((RS_ED_MAT_2(ii,:)./(RS_ED_MAT_3(ii,:)*b_ref^2))
                                - 1) );
            end
        end
        %Calculate the directivity factor of the source under test using in situ Room Constant
        %just calculated
        if r_ED_close == 1
            if r_ED_far == 2
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_1(ii,:)./SUT_ED_MAT_2(ii,:))) ) ./ (
                              R_in_situ(ii,:).*((SUT_ED_MAT_1(ii,:)./(SUT_ED_MAT_2(ii,:)*b^2)) -
                              1) );
            end
            if r_ED_far == 3
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_1(ii,:)./SUT_ED_MAT_3(ii,:))) ) ./ (
                              R_in_situ(ii,:).*((SUT_ED_MAT_1(ii,:)./(SUT_ED_MAT_3(ii,:)*b^2)) -
                              1) );
            end
        end
        if r_ED_close == 2
            if r_ED_far == 3
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_2(ii,:)./SUT_ED_MAT_3(ii,:))) ) ./ (
                              R_in_situ(ii,:).*((SUT_ED_MAT_2(ii,:)./(SUT_ED_MAT_3(ii,:)*b^2)) -
                              1) );
            end
        end

    end
    if FullCorrections == 1                %Includes Air Absorption
        DELTA_ref = 10.^( (alpha_prop*distances(r_ED_close_ref).*(1.0053-
                    0.0012*alpha_prop*distances(r_ED_close_ref)).^1.6)/10 );
        DELTAb_ref = 10.^( (alpha_prop*b_ref*distances(r_ED_close_ref).*(1.0053-
                     0.0012*alpha_prop*b_ref*distances(r_ED_close_ref)).^1.6)/10 );
        DELTA = 10.^( (alpha_prop*distances(r_ED_close).*(1.0053-
                0.0012*alpha_prop*distances(r_ED_close)).^1.6)/10 );
        DELTAb = 10.^( (alpha_prop*b*distances(r_ED_close).*(1.0053-
                 0.0012*alpha_prop*b*distances(r_ED_close)).^1.6)/10 );
        %Calculate the Room Constant based on the Measurement Data from the Reference Source and
        %its Known Directivity Factor
        if r_ED_close_ref == 1
            if r_ED_far_ref == 2
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_1(ii,:)./RS_ED_MAT_2(ii,:))) ) ./ (
```

```
            gamma_Official_ref.*(((RS_ED_MAT_1(ii,:).*DELTAb_ref)./(RS_ED_MAT_2(ii,:)*b_ref^2))
                                - DELTA_ref) );
            end
            if r_ED_far_ref == 3
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_1(ii,:)./RS_ED_MAT_3(ii,:))) ) ./ (
            gamma_Official_ref.*(((RS_ED_MAT_1(ii,:).*DELTAb_ref)./(RS_ED_MAT_3(ii,:)*b_ref^2))
                                - DELTA_ref) );
            end
        end
        if r_ED_close_ref == 2
            if r_ED_far_ref == 3
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (RS_ED_MAT_2(ii,:)./RS_ED_MAT_3(ii,:))) ) ./ (
            gamma_Official_ref.*(((RS_ED_MAT_2(ii,:).*DELTAb_ref)./(RS_ED_MAT_3(ii,:)*b_ref^2))
                                - DELTA_ref) );
            end
        end
        %Calculate the directivity factor of the source under test using in situ Room Constant
        %just calculated
        if r_ED_close == 1
            if r_ED_far == 2
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_1(ii,:)./SUT_ED_MAT_2(ii,:))) ) ./ (
                      R_in_situ(ii,:).*(((SUT_ED_MAT_1(ii,:).*DELTAb)./(SUT_ED_MAT_2(ii,:)*b^2))
                              - DELTA) );
            end
            if r_ED_far == 3
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_1(ii,:)./SUT_ED_MAT_3(ii,:))) ) ./ (
                      R_in_situ(ii,:).*(((SUT_ED_MAT_1(ii,:).*DELTAb)./(SUT_ED_MAT_3(ii,:)*b^2))
                              - DELTA) );
            end
        end
        if r_ED_close == 2
            if r_ED_far == 3
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (SUT_ED_MAT_2(ii,:)./SUT_ED_MAT_3(ii,:))) ) ./ (
                      R_in_situ(ii,:).*(((SUT_ED_MAT_2(ii,:).*DELTAb)./(SUT_ED_MAT_3(ii,:)*b^2))
                              - DELTA) );
            end
        end

    end
    % Calculate the 1/3 octave band frequency-dependent sound power output of the source (using
    % all 4 ED)
    if FullCorrections == 0       %Does NOT Include Air Absorption
        if r_ED_close == 1
            if r_ED_far == 2
                if ii == 3
                    pow_First =  (SUT_ED_MAT_1(ii,:)/2) ./ (
                                 (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                                 (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_2(ii,:)/2) ./ (
                                 (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                 (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_1(ii,:)) ./ (
                                 (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                                 (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_2(ii,:)) ./ (
                                 (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                 (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
            if r_ED_far == 3
                if ii == 3
                    pow_First =  (SUT_ED_MAT_1(ii,:)/2) ./ (
                                 (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
```

```
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_1(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
        end
        if r_ED_close == 2
            if r_ED_far == 3
                if ii == 3
                    pow_First =  (SUT_ED_MAT_2(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_2(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
        end

    end
    if FullCorrections == 1          %Includes Air Absorption
        if r_ED_close == 1
            if r_ED_far == 2
                if ii == 3
                    pow_First =  (SUT_ED_MAT_1(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_2(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_1(ii,:)) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_2(ii,:)) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
            if r_ED_far == 3
                if ii == 3
                    pow_First =  (SUT_ED_MAT_1(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_1(ii,:)) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
```

```
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
        end
        if r_ED_close == 2
            if r_ED_far == 3
                if ii == 3
                    pow_First =  (SUT_ED_MAT_2(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)/2) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_First =  (SUT_ED_MAT_2(ii,:)) ./ (
                                ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                    pow_Second = (SUT_ED_MAT_3(ii,:)) ./ (
                                ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
                end
                pow_Mat = cat(1,pow_First,pow_Second);
                pow(ii,:) = mean(pow_Mat,1);
            end
        end

    end
    if FullCorrections == 0    %Does NOT Include Reference Power Corrections or Acoustic
                               %Radiation Impedance Corrections
        % Convert Sound Power Estimates to Levels
        L_w_In_Situ(ii,:) = abs(10*log10(pow(ii,:)/pow_ref));
    end
    if FullCorrections == 1     %Includes Reference Power Corrections and Acoustic Radiation
                                %Impedance Corrections
        C1 = -10*log10(Pa/101.325) + 5*log10((273.15+T_C)/314);
        C2 = -10*log10(Pa/101.325) + 15*log10((273.15+T_C)/296);
        % Convert Sound Power Estimates to Levels
        L_w_In_Situ(ii,:) = abs(10*log10(pow(ii,:)/pow_ref)) + C1 + C2;
    end
    % Calculate the Overall Sound Power Level (1/3 Oct)
    Overall_Lw_In_Situ(ii) = 10*log10(sum(10.^(abs(L_w_In_Situ(ii,:))/10),2));

    %% Add Weightings
    %A-Weighting
    A_f = (12200^2*freq_model.^4)./(
(freq_model.^2+20.6^2).*sqrt((freq_model.^2+107.7^2).*(freq_model.^2+737.9^2)).*(freq_model.^2+12
        200^2) );
    A_weight = 2+20*log10(A_f);
    A_weight_L_w_In_Situ(ii,:) = L_w_In_Situ(ii,:) + A_weight;
    Overall_A_weight_Lw_In_Situ(ii) = 10*log10(sum(10.^(abs(A_weight_L_w_In_Situ(ii,:))/10),2));
    %C-Weighting
    C_f = (12200^2*freq_model.^2)./( (freq_model.^2+20.6^2).*(freq_model.^2+12200^2) );
    C_weight = 0.06+20*log10(C_f);
    C_weight_L_w_In_Situ(ii,:) = L_w_In_Situ(ii,:) + C_weight;
    Overall_C_weight_Lw_In_Situ(ii) = 10*log10(sum(10.^(abs(C_weight_L_w_In_Situ(ii,:))/10),2));
end

%% Load in the sound power estimates from the ISO 3741 method (for comparison)
loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
      Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3741\Offical ISO 3741
      Data\';

if Source == 1
    load([loc 'REPEAT Processed Dodec SP ISO 3741 Data']);
end
if Source == 2
```

```
    load([loc 'REPEAT Processed Single Driver SP ISO 3741 Data']);
end
if Source == 3
    load([loc 'REPEAT Processed Horn SP ISO 3741 Data']);
end

eval('L_w_3741_nv = L_w;');
eval('Overall_L_w_3741_nv = Overall_L_w;');
eval('f_3741 = f_Oct3;');
clear L_w   Overall_L_w   f_Oct3_cal
%Apply Weightings
%A-Weighting
A_weight_L_w_3741_nv = L_w_3741_nv + A_weight.';
Overall_A_weight_L_w_3741_nv = 10*log10(sum(10.^(abs(A_weight_L_w_3741_nv)/10),1));
%C-Weighting
C_weight_L_w_3741_nv = L_w_3741_nv + C_weight.';
Overall_C_weight_L_w_3741_nv = 10*log10(sum(10.^(abs(C_weight_L_w_3741_nv)/10),1));

%% Load in the sound power estimates from the ISO 3745 method - Full Grid (for comparison)
loc2 = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
       Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full Grid)\Official
       ISO 3745 Data\Processed SP\';

if Source == 1
    load([loc2 'Dodec SP ISO 3745 Full Data']);
end
if Source == 2
    load([loc2 'Single Driver SP ISO 3745 Full Data']);
end
if Source == 3
    load([loc2 'Horn SP ISO 3745 Full Data']);
end

eval('L_w_3745_Full = L_w;');
eval('Overall_L_w_3745_Full = Overall_L_w;');
eval('f_3745_Full = f_Oct3_cal;');
clear L_w   A_weight_L_w   Overall_L_w   A_weight_Overall_L_w   f_Oct3_cal
%Apply Weightings
%A-Weighting
A_weight_L_w_3745_Full = L_w_3745_Full(ind_l:ind_h) + A_weight.';
Overall_A_weight_L_w_3745_Full = 10*log10(sum(10.^(abs(A_weight_L_w_3745_Full)/10),1));
%C-Weighting
C_weight_L_w_3745_Full = L_w_3745_Full(ind_l:ind_h) + C_weight.';
Overall_C_weight_L_w_3745_Full = 10*log10(sum(10.^(abs(C_weight_L_w_3745_Full)/10),1));

%% Calculate Statistics (Root-Mean-Square Deviation)
RMSD = zeros(4,1);
RMSD_A_weight = zeros(4,1);
RMSD_C_weight = zeros(4,1);
for ii = 1:4
    RMSD(ii,:) = sqrt( mean((abs(L_w_In_Situ(ii,:)) - L_w_3745_Full(8:28).').^2) );
    RMSD_A_weight(ii,:) = sqrt( mean((abs(A_weight_L_w_In_Situ(ii,:)) -
                          A_weight_L_w_3745_Full.').^2) );
    RMSD_C_weight(ii,:) = sqrt( mean((abs(C_weight_L_w_In_Situ(ii,:)) -
                          C_weight_L_w_3745_Full.').^2) );
end
RMSD(5,1) = sqrt( mean((L_w_3741_nv.' - L_w_3745_Full(8:28).').^2) );
RMSD_A_weight(5,1) = sqrt( mean((A_weight_L_w_3741_nv.' - A_weight_L_w_3745_Full.').^2) );
RMSD_C_weight(5,1) = sqrt( mean((C_weight_L_w_3741_nv.' - C_weight_L_w_3745_Full.').^2) );

%% Plot and Compare the Sound Power Estimates
ref_source = ['Dodec','Single Driver','Horn'];

if Ref_Source == 1
    ref_num = 1:5;
end
if Ref_Source == 2
    ref_num = 6:18;
end
if Ref_Source == 3
    ref_num = 19:22;
```

```
end

%All EDs
figure(1)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(L_w_In_Situ(1,:)),'b',freq_model,abs(L_w_In_Situ(2,:)),'g',freq_model,abs
(L_w_In_Situ(3,:)),'r',freq_model,abs(L_w_In_Situ(4,:)),'k',f_3741,L_w_3741_nv,'m',f_3745_Full(8:
28),L_w_3745_Full(8:28),'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist.
              ' num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist. '
              num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 3
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_Lw_In_Situ(1),'%.1f') ' dB, RMSD = '
       num2str(RMSD(1),'%.1f') ' dB)'],['KED              ' num2str(Overall_Lw_In_Situ(2),'%.1f')
      ' dB                ' num2str(RMSD(2),'%.1f') ' dB'],['TED              '
```

```
      num2str(Overall_Lw_In_Situ(3),'%.1f') ' dB                ' num2str(RMSD(3),'%.1f') '
      dB'],['GED              ' num2str(Overall_Lw_In_Situ(4),'%.1f') ' dB                '
      num2str(RMSD(4),'%.1f') ' dB'],['ISO 3741       ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB
      ' num2str(RMSD(5),'%.1f') ' dB'],['ISO 3745       ' num2str(Overall_L_w_3745_Full,'%.1f') '
      dB                  N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid

%Just PED and GED
figure(2)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(L_w_In_Situ(1,:)),'g',freq_model,abs(L_w_In_Situ(4,:)),'k',f_3741,L_w_374
1_nv,'m',f_3745_Full(8:28),L_w_3745_Full(8:28),'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist.
              ' num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist. '
              num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 3
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
```

```
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_Lw_In_Situ(1),'%.1f') ' dB, RMSD = '
       num2str(RMSD(1),'%.1f') ' dB)'],['GED             ' num2str(Overall_Lw_In_Situ(4),'%.1f')
       ' dB               ' num2str(RMSD(4),'%.1f') ' dB'],['ISO 3741       '
       num2str(Overall_L_w_3741_nv,'%.1f') ' dB               ' num2str(RMSD(5),'%.1f') '
       dB'],['ISO 3745       ' num2str(Overall_L_w_3745_Full,'%.1f') ' dB
       N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid

%Just PED and GED (A-Weighted)
figure(3)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(A_weight_L_w_In_Situ(1,:)),'g',freq_model,abs(A_weight_L_w_In_Situ(4,:)),
'k',f_3741,A_weight_L_w_3741_nv,'m',f_3745_Full(ind_l:ind_h),A_weight_L_w_3745_Full,'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB_A)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist.
              ' num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-
             Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist. '
              num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
              'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 3
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
```

```
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_A_weight_Lw_In_Situ(1),'%.1f') ' dB_A, RMSD = '
       num2str(RMSD_A_weight(1),'%.1f') ' dB)'],['GED              '
       num2str(Overall_A_weight_Lw_In_Situ(4),'%.1f') ' dB_A               '
       num2str(RMSD_A_weight(4),'%.1f') ' dB'],['ISO 3741       '
       num2str(Overall_A_weight_L_w_3741_nv,'%.1f') ' dB_A               '
       num2str(RMSD_A_weight(5),'%.1f') ' dB'],['ISO 3745       '
       num2str(Overall_A_weight_L_w_3745_Full,'%.1f') ' dB_A
       N/A'],'Location','SouthEast')
if Source == 1
    ylim([35 85])
elseif Source == 2
    ylim([30 90])
elseif Source == 3
    ylim([10 100])
end
grid

%Just PED and GED (C-Weighted)
figure(4)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(C_weight_L_w_In_Situ(1,:)),'g',freq_model,abs(C_weight_L_w_In_Situ(4,:)),
'k',f_3741,C_weight_L_w_3741_nv,'m',f_3745_Full(ind_l:ind_h),C_weight_L_w_3745_Full,'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB_C)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist.
            ' num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
            'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (On-Axis)(Probe Dist. '
            num2str(distances(r_ED_close),'%.2f') 'm, ' num2str(distances(r_ED_far),'%.2f')
            'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'])
```

```
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (45\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 3
    if Source == 1
        title(['DODEC in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Rev. Chamber - L_w Estimates From the 6-Mic Probe (90\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_C_weight_Lw_In_Situ(1),'%.1f') ' dB_C, RMSD = '
       num2str(RMSD_C_weight(1),'%.1f') ' dB)'],['GED              '
       num2str(Overall_C_weight_Lw_In_Situ(4),'%.1f') ' dB_C               '
       num2str(RMSD_C_weight(4),'%.1f') ' dB'],['ISO 3741      '
       num2str(Overall_C_weight_L_w_3741_nv,'%.1f') ' dB_C              '
       num2str(RMSD_C_weight(5),'%.1f') ' dB'],['ISO 3745      '
       num2str(Overall_C_weight_L_w_3745_Full,'%.1f') ' dB_C
       N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid

%% Plots - Compare the In Situ Directivity Estimimates Between the Pressure and GED Methods

figure(5)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,ones(1,length(freq_model)),'r:',freq_model,gamma_Official,'g',freq_model,gamm
a(1,:),'b',freq_model,gamma(4,:),'m')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
if Source == 1
    if Angle_num == 1
        title(['DODEC in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['DODEC in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (45\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 3
        title(['DODEC in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (90\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
```

```
    ylim([0 10])
elseif Source == 2
    if Angle_num == 1
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe
              (On-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ',
              ref_source(ref_num), ')'] )
    elseif Angle_num == 2
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe
              (45\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 3
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe
              (90\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
    ylim([0 20])
elseif Source == 3
    if Angle_num == 1
        title(['HORN in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['HORN in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (45\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 3
        title(['HORN in Rev. Chamber - In Situ Direct. Factor Est. from the 6-Mic Probe (90\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
    ylim([0 45])
end
legend('Omni-directional','Anechoic Chamber','In Situ (PED)','In Situ
       (GED)','Location','NorthWest')
grid

%% Plots - Compare the In Situ Room Constant Estimimates Between the Pressure and 4 ED Methods
(Calculated using the known reference source directivity)

figure(6)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,R_Oct3,'g',freq_model,R_in_situ(1,:),'b',freq_model,R_in_situ(4,:),'m')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
if Source == 1
    if Angle_num == 1
        title(['DODEC in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['DODEC in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (45\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 3
        title(['DODEC in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (90\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
elseif Source == 2
    if Angle_num == 1
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe
              (On-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
```

```
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    elseif Angle_num == 2
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe
            (45\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    elseif Angle_num == 3
        title(['SINGLE DRIVER in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe
            (90\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    end
elseif Source == 3
    if Angle_num == 1
        title(['HORN in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (On-
            Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    elseif Angle_num == 2
        title(['HORN in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (45\circ
            Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    elseif Angle_num == 3
        title(['HORN in Rev. Chamber - In Situ Room Const. Est. from the 6-Mic Probe (90\circ
            Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
            num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
            ')'] )
    end
end
legend('T60','In Situ (PED)','In Situ (GED)','Location','NorthWest')
ylim([0 75])
grid
```

Two_Point_In_Situ_Method_Microflown.m

```
%% 2-Point In Situ Approach with A Reference Directivity Source (USING GENERALIZED ENERGY
%  DENSITY)(11/27/13)
%% Only the Microflown Probe was used in these Experiments

clear all; close all; clc;
set(0,'DefaultAxesFontName','Times New Roman');
set(0,'DefaultAxesFontSize',18);
set(0,'DefaultAxesFontWeight','demi');
set(0,'DefaultAxesLineWidth',0.01);
set(0,'DefaultLineLineWidth',2.5);
scsz = get(0,'ScreenSize');
format shortG;

%% Define calculation parameters
Source = 1;          %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
Ref_Source = 2;      %1:Dodec; 2:Single Driver; 3:Horn-loaded compression driver
Angle_num = 2;       %1 or 2 only (measurement position number)(1: off-axis; 2: on-axis)
FullCorrections = 1; %0: no correction terms condidered in the 2-point in situ method; 1: all
corrections considered

%For All Angles Valid ED positions are: 1,3 or 6 only
r_ED_close = 1;      %closer ED probe position to the SOURCE UNDER TEST for the in situ method
                     %(ED)
r_ED_far = 6;        %further ED probe position from the SOURCE UNDER TEST for the in situ method
                     %(ED)
r_ED_close_ref = 1;  %closer ED probe position to the REFERENCE SOURCE for the in situ method
                     %(ED)
r_ED_far_ref = 6;    %further ED probe position from the REFERECE SOURCE for the in situ method
                     %(ED)
absorp_form = 2; %1:Sabine absorption formulation; 2:Eyring absorption formulation
air_absorp = 1;  %1:air absorption considered; 0:air absorption not considered (In calculating
                 %room absorption by T60 - NOT FOR IN SITU APPROACH)
```

```
%% Define constants
Pr = 101.325;       %reference ambient atmospheric pressure in kilopascals (1 atm)
T0 = 293.15;        %reference air temperature (K)
T01 = 273.16;       %triple-point isotherm temperature of water (K)
freq_model = [100 125 160 200 250 315 400 500 630 800 1000 1250 1600 2000 2500 3150 4000 5000
              6300 8000 10000];
p_ref = 20e-6;
BETA = 1/4;         %the Generalized Energy Density weighting factor (1/4 is associated with
                    %optimum spatial uniformity within the GED field)
pow_ref = 1e-12;    %reference sound power
ENBW = 1.5;         %equal noise bandwidth for the Hanning window function
theta0 = 314;
theta1 = 296;

%% Input the Measurement-Specific Data
hr = 34;        %relative humidity (%)
T_C = 22.6;     %ambient atmospheric temperature (C)
Pa_mb = 855;    %ambient atmospheric pressure (millibar)
c = 20.05*sqrt(273 + T_C);            %adiabatic speed of sound
Pa = Pa_mb*0.1;                       %ambient atmospheric pressure (kPa)
T = T_C + 273.15;                     %ambient atmospheric temperature (K)
[rho,rho_dry] = getRho(Pa_mb,T_C,hr); %density of air

%% Define the room-specific data
Lx = 9.249;                          %dimensions of the room (m)
Ly = 9.206;
Lz = 2.742;
S = 2*Lx*Ly + 2*Lx*Lz + 2*Ly*Lz;  %surface area of walls
V = Lx*Ly*Lz;                       %volume of the room
T_60 = [0.88 1.10 1.34 1.22 1.20 1.22 1.13 1.04 0.92 0.73 0.59 0.53 0.79 0.99 0.99 0.93 0.79 0.71
        0.64 0.49 0.39];
A = (55.26/c)*(V./T_60);          %equivalent absorption area of the room (m^2)

%% Load in the ED data

if Angle_num == 1
    distances = [0.826  1.63   2.565  3.593  4.47  5.373];
    alt = 10;      %altitude of angle (rounded to nearest 5 degrees)
    azimuth = 170; %azimuth angle (take 270 minus value in the array)(rounded to the nearest 5
                   %degrees)
    if Source == 1
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
              Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
              Directivity Factor Tests (7_2_13)\Dodec Data\';
        r_ED_probe_1 = distances(1);
        data_b_1 = 'Dodec MF Probe Pos 1 - 1 Background.mat';
        data_1 =   'Dodec MF Probe Pos 1 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'Dodec MF Probe Pos 1 - 3 Background.mat';
        data_3 =   'Dodec MF Probe Pos 1 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'Dodec MF Probe Pos 1 - 6 Background.mat';
        data_6 =   'Dodec MF Probe Pos 1 - 6.mat';
    end
    if Source == 2
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
              Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
              Directivity Factor Tests (7_2_13)\Single Driver Data\';
        r_ED_probe_1 = distances(1);
        data_b_1 = 'SD MF Probe Pos 1 - 1 Background.mat';
        data_1 =   'SD MF Probe Pos 1 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'SD MF Probe Pos 1 - 3 Background.mat';
        data_3 =   'SD MF Probe Pos 1 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'SD MF Probe Pos 1 - 6 Background.mat';
        data_6 =   'SD MF Probe Pos 1 - 6.mat';
    end
    if Source == 3
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
```

```
            Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
            Directivity Factor Tests (7_2_13)\Horn Data\';
        r_ED_probe_1 = distances(1);
        data_b_1 = 'Horn MF Probe Pos 1 - 1 Background.mat';
        data_1 =   'Horn MF Probe Pos 1 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'Horn MF Probe Pos 1 - 3 Background.mat';
        data_3 =   'Horn MF Probe Pos 1 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'Horn MF Probe Pos 1 - 6 Background.mat';
        data_6 =   'Horn MF Probe Pos 1 - 6.mat';
    end
    if Ref_Source == 1
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                   Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                   Directivity Factor Tests (7_2_13)\Dodec Data\';
        ref_data_b_1 = 'Dodec MF Probe Pos 1 - 1 Background.mat';
        ref_data_1 =   'Dodec MF Probe Pos 1 - 1.mat';
        ref_data_b_3 = 'Dodec MF Probe Pos 1 - 3 Background.mat';
        ref_data_3 =   'Dodec MF Probe Pos 1 - 3.mat';
        ref_data_b_6 = 'Dodec MF Probe Pos 1 - 6 Background.mat';
        ref_data_6 =   'Dodec MF Probe Pos 1 - 6.mat';
    end
    if Ref_Source == 2
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                  Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                  Directivity Factor Tests (7_2_13)\Single Driver Data\';
        ref_data_b_1 = 'SD MF Probe Pos 1 - 1 Background.mat';
        ref_data_1 =   'SD MF Probe Pos 1 - 1.mat';
        ref_r_ED_probe_3 = distances(3);
        ref_data_b_3 = 'SD MF Probe Pos 1 - 3 Background.mat';
        ref_data_3 =   'SD MF Probe Pos 1 - 3.mat';
        ref_r_ED_probe_6 = distances(6);
        ref_data_b_6 = 'SD MF Probe Pos 1 - 6 Background.mat';
        ref_data_6 =   'SD MF Probe Pos 1 - 6.mat';
    end
    if Ref_Source == 3
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                  Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                  Directivity Factor Tests (7_2_13)\Horn Data\';
        ref_data_b_1 = 'Horn MF Probe Pos 1 - 1 Background.mat';
        ref_data_1 =   'Horn MF Probe Pos 1 - 1.mat';
        ref_data_b_3 = 'Horn MF Probe Pos 1 - 3 Background.mat';
        ref_data_3 =   'Horn MF Probe Pos 1 - 3.mat';
        ref_data_b_6 = 'Horn MF Probe Pos 1 - 6 Background.mat';
        ref_data_6 =   'Horn MF Probe Pos 1 - 6.mat';
    end
end
if Angle_num == 2
    distances = [0.815  1.67   2.958  3.858  4.49  5.44];
    alt = 0;            %altitude of angle (rounded to nearest 5 degrees)
    azimuth = 270; %azimuth angle (take 270 minus value in the array)(rounded to the nearest 5
                   %degrees)
    if Source == 1
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
              Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
              Directivity Factor Tests (7_2_13)\Dodec Data\';
        r_ED_probe_1 = distances(1);
        data_b_1 = 'Dodec MF Probe Pos 2 - 1 Background.mat';
        data_1 =   'Dodec MF Probe Pos 2 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'Dodec MF Probe Pos 2 - 3 Background.mat';
        data_3 =   'Dodec MF Probe Pos 2 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'Dodec MF Probe Pos 2 - 6 Background.mat';
        data_6 =   'Dodec MF Probe Pos 2 - 6.mat';
    end
    if Source == 2
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
              Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
              Directivity Factor Tests (7_2_13)\Single Driver Data\';
```

```
        r_ED_probe_1 = distances(1);
        data_b_1 = 'SD MF Probe Pos 2 - 1 Background.mat';
        data_1 =   'SD MF Probe Pos 2 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'SD MF Probe Pos 2 - 3 Background.mat';
        data_3 =   'SD MF Probe Pos 2 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'SD MF Probe Pos 2 - 6 Background.mat';
        data_6 =   'SD MF Probe Pos 2 - 6.mat';
    end
    if Source == 3
        loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
              Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
              Directivity Factor Tests (7_2_13)\Horn Data\';
        r_ED_probe_1 = distances(1);
        data_b_1 = 'Horn MF Probe Pos 2 - 1 Background.mat';
        data_1 =   'Horn MF Probe Pos 2 - 1.mat';
        r_ED_probe_3 = distances(3);
        data_b_3 = 'Horn MF Probe Pos 2 - 3 Background.mat';
        data_3 =   'Horn MF Probe Pos 2 - 3.mat';
        r_ED_probe_6 = distances(6);
        data_b_6 = 'Horn MF Probe Pos 2 - 6 Background.mat';
        data_6 =   'Horn MF Probe Pos 2 - 6.mat';
    end
    if Ref_Source == 1
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                  Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                  Directivity Factor Tests (7_2_13)\Dodec Data\';
        ref_data_b_1 = 'Dodec MF Probe Pos 2 - 1 Background.mat';
        ref_data_1 =   'Dodec MF Probe Pos 2 - 1.mat';
        ref_data_b_3 = 'Dodec MF Probe Pos 2 - 3 Background.mat';
        ref_data_3 =   'Dodec MF Probe Pos 2 - 3.mat';
        ref_data_b_6 = 'Dodec MF Probe Pos 2 - 6 Background.mat';
        ref_data_6 =   'Dodec MF Probe Pos 2 - 6.mat';
    end
    if Ref_Source == 2
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                  Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                  Directivity Factor Tests (7_2_13)\Single Driver Data\';
        ref_data_b_1 = 'SD MF Probe Pos 2 - 1 Background.mat';
        ref_data_1 =   'SD MF Probe Pos 2 - 1.mat';
        ref_data_b_3 = 'SD MF Probe Pos 2 - 3 Background.mat';
        ref_data_3 =   'SD MF Probe Pos 2 - 3.mat';
        ref_data_b_6 = 'SD MF Probe Pos 2 - 6 Background.mat';
        ref_data_6 =   'SD MF Probe Pos 2 - 6.mat';
    end
    if Ref_Source == 3
        ref_loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                  Research\Official Thesis Research\OFFICIAL THESIS FILES\WSC 3252\In Situ
                  Directivity Factor Tests (7_2_13)\Horn Data\';
        ref_data_b_1 = 'Horn MF Probe Pos 2 - 1 Background.mat';
        ref_data_1 =   'Horn MF Probe Pos 2 - 1.mat';
        ref_data_b_3 = 'Horn MF Probe Pos 2 - 3 Background.mat';
        ref_data_3 =   'Horn MF Probe Pos 2 - 3.mat';
        ref_data_b_6 = 'Horn MF Probe Pos 2 - 6 Background.mat';
        ref_data_6 =   'Horn MF Probe Pos 2 - 6.mat';
    end
end

%% Calculate the 1/3 octave band frequency-dependent room constant, R (Based on T60 Data - Not In
%  Situ R)
%Generate the Total Absorption Coefficient (Eq 1-2.6.7 of Phscs 661 notes)
h = hr*(Pr/Pa)*10^(-6.8346*(T01/T)^(1.261)+4.6151); %the molar concentration of water vapor (%)
fro = (Pa/Pr)*(24+4.04e4*h*((0.02+h)/(0.391+h)));   %the relaxation frequency of oxygen
frN = (Pa/Pr)*(T0/T)^(1/2)*(9+280*h*exp(-4.170*((T0/T)^(1/3)-1))); %the relaxation frequency of
                                                                   %nitrogen
alpha_prop = freq_model.^2.*((1.84e-11*(Pr/Pa)*(T/T0)^(1/2)) + (T0/T)^(5/2).*(((0.01275*exp(-
2239.1/T))./(fro+(freq_model.^2/fro)))+((0.1068*exp(-3352.0/T))./(frN+(freq_model.^2/frN)))));
%the total absorption due to wave propagation in air
%Calculate the room constant, R
```

```
if absorp_form == 1                                     %Sabine absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = (0.161*V)./(S*T_60);               %the Sabine absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S); %the total absorption coefficient in the
room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant
    end
    if air_absorp == 1                                                  %includes air absorption
        alpha_wall = ((0.161*V)./(S*T_60)) - ((8*alpha_prop*V)/S); %the Sabine absorption
                                                                    %coefficient of the wall
                                                                    %(considering the effects of
                                                                    %air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);              %the total absorption in the
                                                                    %room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                  %the 1/3 octave band
                                                                    %frequency-dependent room
                                                                    %constant (including air
                                                                    %absorption)
    end
end
if absorp_form == 2                                     %Eyring absorption formulation
    if air_absorp == 0                                  %no air absorption
        alpha_wall = 1-exp(-(0.161*V)./(S*T_60));       %The Eyring absorption coefficient of
                                                        %the wall (neglecting air absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S); %the total absorption in the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';      %the 1/3 octave band frequency-dependent
                                                        %room constant (no air absorption)
    end
    if air_absorp == 1                                                    %includes air absorption
        alpha_wall = 1-exp((1/S)*(8*alpha_prop*V-((0.161*V)./T_60))); %the Eyring absorption
                                                                      %coefficient of the wall
                                                                      %(considering air
                                                                      %absorption)
        alpha_tot = alpha_wall + ((8*alpha_prop*V)/S);                %the total absorption in
                                                                      %the room
        R_Oct3 = ((alpha_tot*S)./(1-alpha_tot)).';                    %the 1/3 octave band
                                                                      %frequency-dependent room
                                                                      %constant (including air
                                                                      %absorption)
    end
end

%% Load in the data from the anechoic chamber and create 1/3 octave band directivity factors for
%  the Reference Source (Not In Situ Directivity Factors)
theta = (0:5:180)'*pi/180;     %polar angles of the measurement sphere
phi = (0:5:355)*pi/180;        %azimuth angles of the measurement sphere
%Load in the FRF data from the SOURCE UNDER TEST

if Source == 1    %Dodec
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Dodec Directivity\Data\';
    load([loc_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 2    %Single Driver
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Source == 3    %Horn
    loc_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                    Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                    Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Horn Directivity\Data\';
    load([loc_Dir_Fact 'Horn_TransferFunctionData.mat'])
```

```
    load([loc_Dir_Fact 'f.mat'])
    load([loc_Dir_Fact 'EASERA_freq_lim.mat'])
end

%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;              %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
                                              %octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
[~,summed_weighted_Oct3_H2_sq] = MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h
theta = 90 - alt;
phi = 270 - azimuth;

if phi < 0
    phi = 360 + phi;
end

theta_ind = (theta+5)/5;
phi_ind = (phi+5)/5;
gamma_Official = squeeze(Dir_Fact_Oct3(theta_ind,phi_ind,:)).';
%Load in the FRF data from the REFERECE SOURCE
theta = (0:5:180)'*pi/180;     %polar angles of the measurement sphere
phi = (0:5:355)*pi/180;        %azimuth angles of the measurement sphere

if Ref_Source == 1    %Dodec
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Dodec
                        Directivity\Data\';
    load([loc_ref_Dir_Fact 'Dodec_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Ref_Source == 2    %Single Driver
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\SD Directivity\Data\';
    load([loc_ref_Dir_Fact 'Single_Driver_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end
if Ref_Source == 3    %Horn
    loc_ref_Dir_Fact = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED
                        Research\Official Thesis Research\OFFICIAL THESIS FILES\Official Sound
                        Powers\ISO 3745 (Full Grid)\Official ISO 3745 Data\Horn
                        Directivity\Data\';
    load([loc_ref_Dir_Fact 'Horn_TransferFunctionData.mat'])
    load([loc_ref_Dir_Fact 'f.mat'])
    load([loc_ref_Dir_Fact 'EASERA_freq_lim.mat'])
end

%Generate the area weighting factors, S_m_n
r_sq_MAT = ones(size(FRF))*(70*0.0254)^2; %radius of mic array
```

```
N = length(phi);    %number of azimuth angle meausurements
Sp = zeros(size(FRF));
Sp([1 end],:,:) = 4*pi*r_sq_MAT([1 end],:,:)/N*sin(min(diff(theta))/4).^2;
Sp(2:end-1,:,:) = 2*r_sq_MAT(2:end-1,:,:).*min(diff(phi)).*...
    sin(repmat(theta(2:end-1),[1 length(phi) length(f)]))*...
    sin(min(diff(theta))/2);             %the area weighting factors
%Create 1/3 octave band squared magnitude FRF values
H_sq = abs(FRF).^2;              %the squared magnitude FRF values
[f_Oct3,H_Oct3_sq] = MyOct3Bands_MAT(f,H_sq); %the 1/3 octave band center frequencies; the 1/3
                                              %octave band squared magnitude FRF values
%Create 1/3 octave band values of the area-weighted squared magnitude FRF
%values summed over the surface of the measurement sphere
summed_weighted_H2_sq = sum(sum(Sp.*H_sq));
summed_weighted_H2_sq = repmat(summed_weighted_H2_sq,[length(theta) length(phi) 1]);
[~,summed_weighted_Oct3_H2_sq] = MyOct3Bands_MAT(f,summed_weighted_H2_sq);
%Calculate the 1/3 octave band directivity factor matrix
Dir_Fact_Oct3 = (4*pi*r_sq_MAT(1,1,1)*H_Oct3_sq)./summed_weighted_Oct3_H2_sq;
%Truncate the 1/3 octave band to the frequency range limited by the EASERA T60 values
ind_l = find(f_Oct3 == freq_model(1));
ind_h = find(f_Oct3 == freq_model(end));
Dir_Fact_Oct3 = Dir_Fact_Oct3(:,:,ind_l:ind_h);
clear  ind_l  ind_h
theta = 90 - alt;
phi = 270 - azimuth;

if phi < 0
    phi = 360 + phi;
end

theta_ind = (theta+5)/5;
phi_ind = (phi+5)/5;
gamma_Official_ref = squeeze(Dir_Fact_Oct3(theta_ind,phi_ind,:)).';

%% Calculate the Schroeder Frequency of the Room
f_Sch = round( sqrt( (c^3)/(4*log(10)) )*sqrt( mean(T_60)/V ) );

%% Calculate the Angle-Dependent Critical Distance (Distance where the direct and reverberant
%  field energy are equal)
r_c = sqrt((gamma_Official.*R_Oct3.')/(16*pi));
r_c_ref = sqrt((gamma_Official_ref.*R_Oct3.')/(16*pi));
r_c_Mat = [freq_model;
           r_c;
           r_c_ref]
r_Close = distances(r_ED_close);
r_Far = distances(r_ED_far);

%% CALUCULATE THE SOUND POWER USING THE IN SITU DIRECTIVITY FACTORS AND THE HOPKINS/STRYKER
%  EQUATION

%% Calculate the four acoustic energy density quantities (PED, KED, TED, and GED) in 1/3 octave
%  bands (from the Microflown Probe)
%Load in the Background Noise data (Narrowband Squared Values)
load([loc data_b_1])
eval('bg_sq_p_1 = MAT_003_Autopower_Pressure(:,2)'';');
eval('bg_sq_Ux_1 = MAT_004_Autopower_Ux(:,2)'';');
eval('bg_sq_Uy_1 = MAT_005_Autopower_Uy(:,2)'';');
eval('bg_sq_Uz_1 = MAT_006_Autopower_Uz(:,2)'';');
eval('f_MF = MAT_003_Autopower_Pressure(:,1)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([loc data_b_3])
eval('bg_sq_p_3 = MAT_003_Autopower_Pressure(:,2)'';');
eval('bg_sq_Ux_3 = MAT_004_Autopower_Ux(:,2)'';');
eval('bg_sq_Uy_3 = MAT_005_Autopower_Uy(:,2)'';');
eval('bg_sq_Uz_3 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([loc data_b_6])
eval('bg_sq_p_6 = MAT_003_Autopower_Pressure(:,2)'';');
eval('bg_sq_Ux_6 = MAT_004_Autopower_Ux(:,2)'';');
eval('bg_sq_Uy_6 = MAT_005_Autopower_Uy(:,2)'';');
```

```
eval('bg_sq_Uz_6 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_b_1])
eval('ref_bg_sq_p_1 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_bg_sq_Ux_1 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_bg_sq_Uy_1 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_bg_sq_Uz_1 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_b_3])
eval('ref_bg_sq_p_3 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_bg_sq_Ux_3 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_bg_sq_Uy_3 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_bg_sq_Uz_3 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_b_6])
eval('ref_bg_sq_p_6 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_bg_sq_Ux_6 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_bg_sq_Uy_6 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_bg_sq_Uz_6 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
%Load in the Uncorrected Measurement data (Narrowband Squared Values)
load([loc data_1])
eval('p_sq_uc_1 = MAT_003_Autopower_Pressure(:,2)'';');
eval('Ux_sq_uc_1 = MAT_004_Autopower_Ux(:,2)'';');
eval('Uy_sq_uc_1 = MAT_005_Autopower_Uy(:,2)'';');
eval('Uz_sq_uc_1 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([loc data_3])
eval('p_sq_uc_3 = MAT_003_Autopower_Pressure(:,2)'';');
eval('Ux_sq_uc_3 = MAT_004_Autopower_Ux(:,2)'';');
eval('Uy_sq_uc_3 = MAT_005_Autopower_Uy(:,2)'';');
eval('Uz_sq_uc_3 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([loc data_6])
eval('p_sq_uc_6 = MAT_003_Autopower_Pressure(:,2)'';');
eval('Ux_sq_uc_6 = MAT_004_Autopower_Ux(:,2)'';');
eval('Uy_sq_uc_6 = MAT_005_Autopower_Uy(:,2)'';');
eval('Uz_sq_uc_6 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_1])
eval('ref_p_sq_uc_1 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_Ux_sq_uc_1 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_Uy_sq_uc_1 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_Uz_sq_uc_1 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_3])
eval('ref_p_sq_uc_3 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_Ux_sq_uc_3 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_Uy_sq_uc_3 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_Uz_sq_uc_3 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz
load([ref_loc ref_data_6])
eval('ref_p_sq_uc_6 = MAT_003_Autopower_Pressure(:,2)'';');
eval('ref_Ux_sq_uc_6 = MAT_004_Autopower_Ux(:,2)'';');
eval('ref_Uy_sq_uc_6 = MAT_005_Autopower_Uy(:,2)'';');
eval('ref_Uz_sq_uc_6 = MAT_006_Autopower_Uz(:,2)'';');
clear  MAT_003_Autopower_Pressure  MAT_004_Autopower_Ux  MAT_005_Autopower_Uy
       MAT_006_Autopower_Uz

%% Correct for the Equal Noise Bandwidth
bg_sq_p_1 = bg_sq_p_1/ENBW;
bg_sq_Ux_1 = bg_sq_Ux_1/ENBW;
```

```
bg_sq_Uy_1 = bg_sq_Uy_1/ENBW;
bg_sq_Uz_1 = bg_sq_Uz_1/ENBW;
bg_MF_Data_Mat_1 = cat(1,bg_sq_p_1,bg_sq_Ux_1,bg_sq_Uy_1,bg_sq_Uz_1);
bg_sq_p_3 = bg_sq_p_3/ENBW;
bg_sq_Ux_3 = bg_sq_Ux_3/ENBW;
bg_sq_Uy_3 = bg_sq_Uy_3/ENBW;
bg_sq_Uz_3 = bg_sq_Uz_3/ENBW;
bg_MF_Data_Mat_3 = cat(1,bg_sq_p_3,bg_sq_Ux_3,bg_sq_Uy_3,bg_sq_Uz_3);
bg_sq_p_6 = bg_sq_p_6/ENBW;
bg_sq_Ux_6 = bg_sq_Ux_6/ENBW;
bg_sq_Uy_6 = bg_sq_Uy_6/ENBW;
bg_sq_Uz_6 = bg_sq_Uz_6/ENBW;
bg_MF_Data_Mat_6 = cat(1,bg_sq_p_6,bg_sq_Ux_6,bg_sq_Uy_6,bg_sq_Uz_6);
ref_bg_sq_p_1 = ref_bg_sq_p_1/ENBW;
ref_bg_sq_Ux_1 = ref_bg_sq_Ux_1/ENBW;
ref_bg_sq_Uy_1 = ref_bg_sq_Uy_1/ENBW;
ref_bg_sq_Uz_1 = ref_bg_sq_Uz_1/ENBW;
ref_bg_MF_Data_Mat_1 = cat(1,ref_bg_sq_p_1,ref_bg_sq_Ux_1,ref_bg_sq_Uy_1,ref_bg_sq_Uz_1);
ref_bg_sq_p_3 = ref_bg_sq_p_3/ENBW;
ref_bg_sq_Ux_3 = ref_bg_sq_Ux_3/ENBW;
ref_bg_sq_Uy_3 = ref_bg_sq_Uy_3/ENBW;
ref_bg_sq_Uz_3 = ref_bg_sq_Uz_3/ENBW;
ref_bg_MF_Data_Mat_3 = cat(1,ref_bg_sq_p_3,ref_bg_sq_Ux_3,ref_bg_sq_Uy_3,ref_bg_sq_Uz_3);
ref_bg_sq_p_6 = ref_bg_sq_p_6/ENBW;
ref_bg_sq_Ux_6 = ref_bg_sq_Ux_6/ENBW;
ref_bg_sq_Uy_6 = ref_bg_sq_Uy_6/ENBW;
ref_bg_sq_Uz_6 = ref_bg_sq_Uz_6/ENBW;
ref_bg_MF_Data_Mat_6 = cat(1,ref_bg_sq_p_6,ref_bg_sq_Ux_6,ref_bg_sq_Uy_6,ref_bg_sq_Uz_6);
p_sq_uc_1 = p_sq_uc_1 /ENBW;
Ux_sq_uc_1 = Ux_sq_uc_1/ENBW;
Uy_sq_uc_1 = Uy_sq_uc_1/ENBW;
Uz_sq_uc_1 = Uz_sq_uc_1/ENBW;
uc_MF_Data_Mat_1 = cat(1,p_sq_uc_1,Ux_sq_uc_1,Uy_sq_uc_1,Uz_sq_uc_1);
p_sq_uc_3 = p_sq_uc_3 /ENBW;
Ux_sq_uc_3 = Ux_sq_uc_3/ENBW;
Uy_sq_uc_3 = Uy_sq_uc_3/ENBW;
Uz_sq_uc_3 = Uz_sq_uc_3/ENBW;
uc_MF_Data_Mat_3 = cat(1,p_sq_uc_3,Ux_sq_uc_3,Uy_sq_uc_3,Uz_sq_uc_3);
p_sq_uc_6 = p_sq_uc_6 /ENBW;
Ux_sq_uc_6 = Ux_sq_uc_6/ENBW;
Uy_sq_uc_6 = Uy_sq_uc_6/ENBW;
Uz_sq_uc_6 = Uz_sq_uc_6/ENBW;
uc_MF_Data_Mat_6 = cat(1,p_sq_uc_6,Ux_sq_uc_6,Uy_sq_uc_6,Uz_sq_uc_6);
ref_p_sq_uc_1 = ref_p_sq_uc_1 /ENBW;
ref_Ux_sq_uc_1 = ref_Ux_sq_uc_1/ENBW;
ref_Uy_sq_uc_1 = ref_Uy_sq_uc_1/ENBW;
ref_Uz_sq_uc_1 = ref_Uz_sq_uc_1/ENBW;
ref_uc_MF_Data_Mat_1 = cat(1,ref_p_sq_uc_1,ref_Ux_sq_uc_1,ref_Uy_sq_uc_1,ref_Uz_sq_uc_1);
ref_p_sq_uc_3 = ref_p_sq_uc_3 /ENBW;
ref_Ux_sq_uc_3 = ref_Ux_sq_uc_3/ENBW;
ref_Uy_sq_uc_3 = ref_Uy_sq_uc_3/ENBW;
ref_Uz_sq_uc_3 = ref_Uz_sq_uc_3/ENBW;
ref_uc_MF_Data_Mat_3 = cat(1,ref_p_sq_uc_3,ref_Ux_sq_uc_3,ref_Uy_sq_uc_3,ref_Uz_sq_uc_3);
ref_p_sq_uc_6 = ref_p_sq_uc_6 /ENBW;
ref_Ux_sq_uc_6 = ref_Ux_sq_uc_6/ENBW;
ref_Uy_sq_uc_6 = ref_Uy_sq_uc_6/ENBW;
ref_Uz_sq_uc_6 = ref_Uz_sq_uc_6/ENBW;
ref_uc_MF_Data_Mat_6 = cat(1,ref_p_sq_uc_6,ref_Ux_sq_uc_6,ref_Uy_sq_uc_6,ref_Uz_sq_uc_6);
%Convert the squared voltages to squared pressure and particle velocity magnitude
[bg_MF_p_sq_1,bg_MF_U_mag_sq_1,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,bg_MF_Data_Mat_1);
[bg_MF_p_sq_3,bg_MF_U_mag_sq_3,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,bg_MF_Data_Mat_3);
[bg_MF_p_sq_6,bg_MF_U_mag_sq_6,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,bg_MF_Data_Mat_6);
[ref_bg_MF_p_sq_1,ref_bg_MF_U_mag_sq_1,~,~,~] =
                          get_MicroF_Vals_Uncorrected(f_MF,ref_bg_MF_Data_Mat_1);
[ref_bg_MF_p_sq_3,ref_bg_MF_U_mag_sq_3,~,~,~] =
                          get_MicroF_Vals_Uncorrected(f_MF,ref_bg_MF_Data_Mat_3);
[ref_bg_MF_p_sq_6,ref_bg_MF_U_mag_sq_6,~,~,~] =
                          get_MicroF_Vals_Uncorrected(f_MF,ref_bg_MF_Data_Mat_6);
[uc_MF_p_sq_1,uc_MF_U_mag_sq_1,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,uc_MF_Data_Mat_1);
[uc_MF_p_sq_3,uc_MF_U_mag_sq_3,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,uc_MF_Data_Mat_3);
[uc_MF_p_sq_6,uc_MF_U_mag_sq_6,~,~,~] = get_MicroF_Vals_Uncorrected(f_MF,uc_MF_Data_Mat_6);
```

```
[ref_uc_MF_p_sq_1,ref_uc_MF_U_mag_sq_1,~,~,~] =
                            get_MicroF_Vals_Uncorrected(f_MF,ref_uc_MF_Data_Mat_1);
[ref_uc_MF_p_sq_3,ref_uc_MF_U_mag_sq_3,~,~,~] =
                            get_MicroF_Vals_Uncorrected(f_MF,ref_uc_MF_Data_Mat_3);
[ref_uc_MF_p_sq_6,ref_uc_MF_U_mag_sq_6,~,~,~] =
                            get_MicroF_Vals_Uncorrected(f_MF,ref_uc_MF_Data_Mat_6);
%Create NOISE-CORRECTED data (Narrowband Squared Values)
MF_p_sq_1 = uc_MF_p_sq_1 - bg_MF_p_sq_1;
MF_p_sq_3 = uc_MF_p_sq_3 - bg_MF_p_sq_3;
MF_p_sq_6 = uc_MF_p_sq_6 - bg_MF_p_sq_6;
ref_MF_p_sq_1 = ref_uc_MF_p_sq_1 - ref_bg_MF_p_sq_1;
ref_MF_p_sq_3 = ref_uc_MF_p_sq_3 - ref_bg_MF_p_sq_3;
ref_MF_p_sq_6 = ref_uc_MF_p_sq_6 - ref_bg_MF_p_sq_6;
MF_U_mag_sq_1 = uc_MF_U_mag_sq_1 - bg_MF_U_mag_sq_1;
MF_U_mag_sq_3 = uc_MF_U_mag_sq_3 - bg_MF_U_mag_sq_3;
MF_U_mag_sq_6 = uc_MF_U_mag_sq_6 - bg_MF_U_mag_sq_6;
ref_MF_U_mag_sq_1 = ref_uc_MF_U_mag_sq_1 - ref_bg_MF_U_mag_sq_1;
ref_MF_U_mag_sq_3 = ref_uc_MF_U_mag_sq_3 - ref_bg_MF_U_mag_sq_3;
ref_MF_U_mag_sq_6 = ref_uc_MF_U_mag_sq_6 - ref_bg_MF_U_mag_sq_6;
%Convert the narrow band squared values to 1/3 octave band squared values
[f_MF_Oct3,MF_p_sq_Oct3_1] = MyOct3Bands(f_MF,MF_p_sq_1);
[~,MF_p_sq_Oct3_3] = MyOct3Bands(f_MF,MF_p_sq_3);
[~,MF_p_sq_Oct3_6] = MyOct3Bands(f_MF,MF_p_sq_6);
[~,ref_MF_p_sq_Oct3_1] = MyOct3Bands(f_MF,ref_MF_p_sq_1);
[~,ref_MF_p_sq_Oct3_3] = MyOct3Bands(f_MF,ref_MF_p_sq_3);
[~,ref_MF_p_sq_Oct3_6] = MyOct3Bands(f_MF,ref_MF_p_sq_6);
[~,MF_U_mag_sq_Oct3_1] = MyOct3Bands(f_MF,MF_U_mag_sq_1);
[~,MF_U_mag_sq_Oct3_3] = MyOct3Bands(f_MF,MF_U_mag_sq_3);
[~,MF_U_mag_sq_Oct3_6] = MyOct3Bands(f_MF,MF_U_mag_sq_6);
[~,ref_MF_U_mag_sq_Oct3_1] = MyOct3Bands(f_MF,ref_MF_U_mag_sq_1);
[~,ref_MF_U_mag_sq_Oct3_3] = MyOct3Bands(f_MF,ref_MF_U_mag_sq_3);
[~,ref_MF_U_mag_sq_Oct3_6] = MyOct3Bands(f_MF,ref_MF_U_mag_sq_6);
%Create a matrix of the PED, KED, TED, and GED vectors (1/3 octave bands)
PED_MF_1 = (MF_p_sq_Oct3_1 / (2*rho*c^2)).';
PED_MF_3 = (MF_p_sq_Oct3_3 / (2*rho*c^2)).';
PED_MF_6 = (MF_p_sq_Oct3_6 / (2*rho*c^2)).';
KED_MF_1 = ((rho/2) * MF_U_mag_sq_Oct3_1).';
KED_MF_3 = ((rho/2) * MF_U_mag_sq_Oct3_3).';
KED_MF_6 = ((rho/2) * MF_U_mag_sq_Oct3_6).';
TED_MF_1 = (PED_MF_1 + KED_MF_1);
TED_MF_3 = (PED_MF_3 + KED_MF_3);
TED_MF_6 = (PED_MF_6 + KED_MF_6);
GED_MF_1 = (BETA*(PED_MF_1) + (1-BETA)*KED_MF_1);
GED_MF_3 = (BETA*(PED_MF_3) + (1-BETA)*KED_MF_3);
GED_MF_6 = (BETA*(PED_MF_6) + (1-BETA)*KED_MF_6);
ED_MF_MAT_full_1 = cat(1,PED_MF_1,KED_MF_1,TED_MF_1,GED_MF_1);
ED_MF_MAT_full_3 = cat(1,PED_MF_3,KED_MF_3,TED_MF_3,GED_MF_3);
ED_MF_MAT_full_6 = cat(1,PED_MF_6,KED_MF_6,TED_MF_6,GED_MF_6);
ref_PED_MF_1 = (ref_MF_p_sq_Oct3_1 / (2*rho*c^2)).';
ref_PED_MF_3 = (ref_MF_p_sq_Oct3_3 / (2*rho*c^2)).';
ref_PED_MF_6 = (ref_MF_p_sq_Oct3_6 / (2*rho*c^2)).';
ref_KED_MF_1 = ((rho/2) * ref_MF_U_mag_sq_Oct3_1).';
ref_KED_MF_3 = ((rho/2) * ref_MF_U_mag_sq_Oct3_3).';
ref_KED_MF_6 = ((rho/2) * ref_MF_U_mag_sq_Oct3_6).';
ref_TED_MF_1 = (ref_PED_MF_1 + ref_KED_MF_1);
ref_TED_MF_3 = (ref_PED_MF_3 + ref_KED_MF_3);
ref_TED_MF_6 = (ref_PED_MF_6 + ref_KED_MF_6);
ref_GED_MF_1 = (BETA*(ref_PED_MF_1) + (1-BETA)*ref_KED_MF_1);
ref_GED_MF_3 = (BETA*(ref_PED_MF_3) + (1-BETA)*ref_KED_MF_3);
ref_GED_MF_6 = (BETA*(ref_PED_MF_6) + (1-BETA)*ref_KED_MF_6);
ref_ED_MF_MAT_full_1 = cat(1,ref_PED_MF_1,ref_KED_MF_1,ref_TED_MF_1,ref_GED_MF_1);
ref_ED_MF_MAT_full_3 = cat(1,ref_PED_MF_3,ref_KED_MF_3,ref_TED_MF_3,ref_GED_MF_3);
ref_ED_MF_MAT_full_6 = cat(1,ref_PED_MF_6,ref_KED_MF_6,ref_TED_MF_6,ref_GED_MF_6);
%Truncate the 1/3 octave band squared data to go from 100 Hz to 10 kHz
ind_l = findnearest(freq_model(1),f_MF_Oct3,0);
ind_h = findnearest(freq_model(end),f_MF_Oct3,0);
f_MF_Oct3 = f_MF_Oct3(ind_l:ind_h);
ED_MF_MAT_1 = ED_MF_MAT_full_1(:,ind_l:ind_h);
ED_MF_MAT_3 = ED_MF_MAT_full_3(:,ind_l:ind_h);
ED_MF_MAT_6 = ED_MF_MAT_full_6(:,ind_l:ind_h);
ref_ED_MF_MAT_1 = ref_ED_MF_MAT_full_1(:,ind_l:ind_h);
```

```
ref_ED_MF_MAT_3 = ref_ED_MF_MAT_full_3(:,ind_l:ind_h);
ref_ED_MF_MAT_6 = ref_ED_MF_MAT_full_6(:,ind_l:ind_h);

%% Calculate the Third-Octave Band IN SITU ROOM CONSTANT Using A Reference Source, Then The IN
%  SITU DIRECTIVITY FACTOR Of The source, and From it The Sound Power (Using All Four EDs)!!!
%Calculate the constant b for each ED probe position
b = distances(r_ED_far)/distances(r_ED_close);
b_ref = distances(r_ED_far_ref)/distances(r_ED_close_ref);
R_in_situ = zeros(4,length(freq_model));
gamma = zeros(4,length(freq_model));
pow_MF = zeros(4,length(freq_model));
L_w_MF_In_Situ = zeros(4,length(freq_model));
Overall_Lw_MF_In_Situ = zeros(4,1);
A_weight_L_w_MF_In_Situ = zeros(4,length(freq_model));
Overall_A_weight_Lw_MF_In_Situ = zeros(4,1);
C_weight_L_w_MF_In_Situ = zeros(4,length(freq_model));
Overall_C_weight_Lw_MF_In_Situ = zeros(4,1);

for ii = 1:4
    if FullCorrections == 0                %Does NOT Include Air Absorption
        %Calculate the Room Constant based on the Measurement Data from the Reference Source and
        %its Known Directivity Factor
        if r_ED_close_ref == 1
            if r_ED_far_ref == 3
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (ref_ED_MF_MAT_1(ii,:)./ref_ED_MF_MAT_3(ii,:))) ) ./ (
                    gamma_Official_ref.*((ref_ED_MF_MAT_1(ii,:)./(ref_ED_MF_MAT_3(ii,:)*b_ref^2))
                                - 1) );
            end
            if r_ED_far_ref == 6
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (ref_ED_MF_MAT_1(ii,:)./ref_ED_MF_MAT_6(ii,:))) ) ./ (
                    gamma_Official_ref.*((ref_ED_MF_MAT_1(ii,:)./(ref_ED_MF_MAT_6(ii,:)*b_ref^2))
                                - 1) );
            end
        end
        if r_ED_close_ref == 3
            if r_ED_far_ref == 6
                R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                                (ref_ED_MF_MAT_3(ii,:)./ref_ED_MF_MAT_6(ii,:))) ) ./ (
                    gamma_Official_ref.*((ref_ED_MF_MAT_3(ii,:)./(ref_ED_MF_MAT_6(ii,:)*b_ref^2))
                                - 1) );
            end
        end
        %Calculate the directivity factor of the source under test using in situ Room Constant
        %just calculated
        if r_ED_close == 1
            if r_ED_far == 3
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (ED_MF_MAT_1(ii,:)./ED_MF_MAT_3(ii,:))) ) ./ (
                              R_in_situ(ii,:).*((ED_MF_MAT_1(ii,:)./(ED_MF_MAT_3(ii,:)*b^2)) -
                              1) );
            end
            if r_ED_far == 6
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (ED_MF_MAT_1(ii,:)./ED_MF_MAT_6(ii,:))) ) ./ (
                          R_in_situ(ii,:).*((ED_MF_MAT_1(ii,:)./(ED_MF_MAT_6(ii,:)*b^2)) - 1) );
            end
        end
        if r_ED_close == 3
            if r_ED_far == 6
                gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                              (ED_MF_MAT_3(ii,:)./ED_MF_MAT_6(ii,:))) ) ./ (
                          R_in_situ(ii,:).*((ED_MF_MAT_3(ii,:)./(ED_MF_MAT_6(ii,:)*b^2)) - 1) );
            end
        end

    end
    if FullCorrections == 1                %Includes Air Absorption
        DELTA_ref = 10.^( (alpha_prop*distances(r_ED_close_ref).*(1.0053-
                    0.0012*alpha_prop*distances(r_ED_close_ref)).^1.6)/10 );
```

```
    DELTAb_ref = 10.^( (alpha_prop*b_ref*distances(r_ED_close_ref).*(1.0053-
                 0.0012*alpha_prop*b_ref*distances(r_ED_close_ref)).^1.6)/10 );
    DELTA = 10.^( (alpha_prop*distances(r_ED_close).*(1.0053-
            0.0012*alpha_prop*distances(r_ED_close)).^1.6)/10 );
    DELTAb = 10.^( (alpha_prop*b*distances(r_ED_close).*(1.0053-
             0.0012*alpha_prop*b*distances(r_ED_close)).^1.6)/10 );
    %Calculate the Room Constant based on the Measurement Data from the Reference Source and
    %its Known Directivity Factor
    if r_ED_close_ref == 1
        if r_ED_far_ref == 3
            R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                              (ref_ED_MF_MAT_1(ii,:)./ref_ED_MF_MAT_3(ii,:))) ) ./ (
  gamma_Official_ref.*(((ref_ED_MF_MAT_1(ii,:).*DELTAb_ref)./(ref_ED_MF_MAT_3(ii,:)*b_ref^2))
                              - DELTA_ref) );
        end
        if r_ED_far_ref == 6
            R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                              (ref_ED_MF_MAT_1(ii,:)./ref_ED_MF_MAT_6(ii,:))) ) ./ (
  gamma_Official_ref.*(((ref_ED_MF_MAT_1(ii,:).*DELTAb_ref)./(ref_ED_MF_MAT_6(ii,:)*b_ref^2))
                              - DELTA_ref) );
        end
    end
    if r_ED_close_ref == 3
        if r_ED_far_ref == 6
            R_in_situ(ii,:) = ( 16*pi*distances(r_ED_close_ref)^2*(1-
                              (ref_ED_MF_MAT_3(ii,:)./ref_ED_MF_MAT_6(ii,:))) ) ./ (
  gamma_Official_ref.*(((ref_ED_MF_MAT_3(ii,:).*DELTAb_ref)./(ref_ED_MF_MAT_6(ii,:)*b_ref^2))
                              - DELTA_ref) );
        end
    end
    %Calculate the directivity factor of the source under test using in situ Room Constant
    %just calculated
    if r_ED_close == 1
        if r_ED_far == 3
            gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                          (ED_MF_MAT_1(ii,:)./ED_MF_MAT_3(ii,:))) ) ./ (
                    R_in_situ(ii,:).*(((ED_MF_MAT_1(ii,:).*DELTAb)./(ED_MF_MAT_3(ii,:)*b^2))
                          - DELTA) );
        end
        if r_ED_far == 6
            gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                          (ED_MF_MAT_1(ii,:)./ED_MF_MAT_6(ii,:))) ) ./ (
                    R_in_situ(ii,:).*(((ED_MF_MAT_1(ii,:).*DELTAb)./(ED_MF_MAT_6(ii,:)*b^2))
                          - DELTA) );
        end
    end
    if r_ED_close == 3
        if r_ED_far == 6
            gamma(ii,:) = ( 16*pi*distances(r_ED_close)^2*(1-
                          (ED_MF_MAT_3(ii,:)./ED_MF_MAT_6(ii,:))) ) ./ (
                    R_in_situ(ii,:).*(((ED_MF_MAT_3(ii,:).*DELTAb)./(ED_MF_MAT_6(ii,:)*b^2))
                          - DELTA) );
        end
    end
end
% Calculate the 1/3 octave band frequency-dependent sound power output of the source (using
% all 4 ED)
if FullCorrections == 0         %Does NOT Include Air Absorption
    if r_ED_close == 1
        if r_ED_far == 3
            if ii == 3
                pow_MF_First = (ED_MF_MAT_1(ii,:)/2) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_3(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            else
                pow_MF_First = (ED_MF_MAT_1(ii,:)) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
```

```
                pow_MF_Second = (ED_MF_MAT_3(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            end
            pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
            pow_MF(ii,:) = mean(pow_MF_Mat,1);
        end
        if r_ED_far == 6
            if ii == 3
                pow_MF_First = (ED_MF_MAT_1(ii,:)/2) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_6(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            else
                pow_MF_First = (ED_MF_MAT_1(ii,:)) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_6(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            end
            pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
            pow_MF(ii,:) = mean(pow_MF_Mat,1);
        end
    end
    if r_ED_close == 3
        if r_ED_far == 6
            if ii == 3
                pow_MF_First = (ED_MF_MAT_3(ii,:)/2) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_6(ii,:)/2) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            else
                pow_MF_First = (ED_MF_MAT_3(ii,:)) ./ (
                               (gamma(ii,:)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_6(ii,:)) ./ (
                                (gamma(ii,:)/(4*pi*distances(r_ED_far)^2*c)) +
                                (4./(R_in_situ(ii,:)*c))  );
            end
            pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
            pow_MF(ii,:) = mean(pow_MF_Mat,1);
        end
    end
end
 if FullCorrections == 1          %Includes Air Absorption
     if r_ED_close == 1
        if r_ED_far == 3
            if ii == 3
                pow_MF_First =  (ED_MF_MAT_1(ii,:)/2) ./ (
                               ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_3(ii,:)/2) ./ (
                               ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
            else
                pow_MF_First =  (ED_MF_MAT_1(ii,:)) ./ (
                               ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
                pow_MF_Second = (ED_MF_MAT_3(ii,:)) ./ (  \
                               ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                               (4./(R_in_situ(ii,:)*c))  );
            end
            pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
            pow_MF(ii,:) = mean(pow_MF_Mat,1);
        end
        if r_ED_far == 6
```

```
                if ii == 3
                    pow_MF_First =  (ED_MF_MAT_1(ii,:)/2) ./ (
                                    ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                    pow_MF_Second = (ED_MF_MAT_6(ii,:)/2) ./ (
                                    ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_MF_First =  (ED_MF_MAT_1(ii,:)) ./ (  \
                                    ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                    pow_MF_Second = (ED_MF_MAT_6(ii,:)) ./ (
                                    ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                end
                pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
                pow_MF(ii,:) = mean(pow_MF_Mat,1);
            end
        end
        if r_ED_close == 3
            if r_ED_far == 6
                if ii == 3
                    pow_MF_First =  (ED_MF_MAT_3(ii,:)/2) ./ (
                                    ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                    pow_MF_Second = (ED_MF_MAT_6(ii,:)/2) ./ (
                                    ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                else
                    pow_MF_First =  (ED_MF_MAT_3(ii,:)) ./ (
                                    ((gamma(ii,:).*DELTA)/(4*pi*distances(r_ED_close)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                    pow_MF_Second = (ED_MF_MAT_6(ii,:)) ./ (
                                    ((gamma(ii,:).*DELTAb)/(4*pi*distances(r_ED_far)^2*c)) +
                                    (4./(R_in_situ(ii,:)*c))  );
                end
                pow_MF_Mat = cat(1,pow_MF_First,pow_MF_Second);
                pow_MF(ii,:) = mean(pow_MF_Mat,1);
            end
        end
    end
     if FullCorrections == 0    %Does NOT Include Reference Power Corrections or Acoustic
                                %Radiation Impedance Corrections
        % Convert Sound Power Estimates to Levels
        L_w_MF_In_Situ(ii,:) = abs(10*log10(pow_MF(ii,:)/pow_ref));
    end
    if FullCorrections == 1     %Includes Reference Power Corrections and Acoustic Radiation
                                %Impedance Corrections
        C1 = -10*log10(Pa/101.325) + 5*log10((273.15+T_C)/314);
        C2 = -10*log10(Pa/101.325) + 15*log10((273.15+T_C)/296);
        % Convert Sound Power Estimates to Levels
        L_w_MF_In_Situ(ii,:) = abs(10*log10(pow_MF(ii,:)/pow_ref)) + C1 + C2;
    end
    % Calculate the Overall Sound Power Level (1/3 Oct)
    Overall_Lw_MF_In_Situ(ii) = 10*log10(sum(10.^(abs(L_w_MF_In_Situ(ii,:))/10),2));

    %% Add Weightings
    %A-Weighting
    A_f = (12200^2*freq_model.^4)./(
(freq_model.^2+20.6^2).*sqrt((freq_model.^2+107.7^2).*(freq_model.^2+737.9^2)).*(freq_model.^2+12
          200^2) );
    A_weight = 2+20*log10(A_f);
    A_weight_L_w_MF_In_Situ(ii,:) = L_w_MF_In_Situ(ii,:) + A_weight;
    Overall_A_weight_Lw_MF_In_Situ(ii) =
                                    10*log10(sum(10.^(abs(A_weight_L_w_MF_In_Situ(ii,:))/10),2));
    %C-Weighting
    C_f = (12200^2*freq_model.^2)./( (freq_model.^2+20.6^2).*(freq_model.^2+12200^2) );
    C_weight = 0.06+20*log10(C_f);
    C_weight_L_w_MF_In_Situ(ii,:) = L_w_MF_In_Situ(ii,:) + C_weight;
    Overall_C_weight_Lw_MF_In_Situ(ii) =
                                    10*log10(sum(10.^(abs(C_weight_L_w_MF_In_Situ(ii,:))/10),2));
```

```
end

%% Load in the sound power estimates from the ISO 3741 method (for comparison)
loc = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
      Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3741\Offical ISO 3741
      Data\';

if Source == 1
    load([loc 'REPEAT Processed Dodec SP ISO 3741 Data']);
end
if Source == 2
    load([loc 'REPEAT Processed Single Driver SP ISO 3741 Data']);
end
if Source == 3
    load([loc 'REPEAT Processed Horn SP ISO 3741 Data']);
end

eval('L_w_3741_nv = L_w;');
eval('Overall_L_w_3741_nv = Overall_L_w;');
eval('f_3741 = f_Oct3;');
clear L_w   Overall_L_w   f_Oct3_cal
%Apply Weightings
%A-Weighting
A_weight_L_w_3741_nv = L_w_3741_nv + A_weight.';
Overall_A_weight_L_w_3741_nv = 10*log10(sum(10.^(abs(A_weight_L_w_3741_nv)/10),1));
%C-Weighting
C_weight_L_w_3741_nv = L_w_3741_nv + C_weight.';
Overall_C_weight_L_w_3741_nv = 10*log10(sum(10.^(abs(C_weight_L_w_3741_nv)/10),1));

%% Load in the sound power estimates from the ISO 3745 method - Full Grid (for comparison)
loc2 = 'G:\Dr Sommerfeldt\Danny Marquez\Graduate Research\Sound Power using GED Research\Official
       Thesis Research\OFFICIAL THESIS FILES\Official Sound Powers\ISO 3745 (Full Grid)\Official
       ISO 3745 Data\Processed SP\';

if Source == 1
    load([loc2 'Dodec SP ISO 3745 Full Data']);
end
if Source == 2
    load([loc2 'Single Driver SP ISO 3745 Full Data']);
end
if Source == 3
    load([loc2 'Horn SP ISO 3745 Full Data']);
end

eval('L_w_3745_Full = L_w;');
eval('Overall_L_w_3745_Full = Overall_L_w;');
eval('f_3745_Full = f_Oct3_cal;');
clear L_w   A_weight_L_w   Overall_L_w   A_weight_Overall_L_w   f_Oct3_cal
%Apply Weightings
%A-Weighting
A_weight_L_w_3745_Full = L_w_3745_Full(ind_l:ind_h) + A_weight.';
Overall_A_weight_L_w_3745_Full = 10*log10(sum(10.^(abs(A_weight_L_w_3745_Full)/10),1));
%C-Weighting
C_weight_L_w_3745_Full = L_w_3745_Full(ind_l:ind_h) + C_weight.';
Overall_C_weight_L_w_3745_Full = 10*log10(sum(10.^(abs(C_weight_L_w_3745_Full)/10),1));

%% Calculate Statistics (Root-Mean-Square Deviation)
RMSD = zeros(4,1);
RMSD_A_weight = zeros(4,1);
RMSD_C_weight = zeros(4,1);

for ii = 1:4
    RMSD(ii,:) = sqrt( mean((abs(L_w_MF_In_Situ(ii,:)) - L_w_3745_Full(8:28).').^2) );
    RMSD_A_weight(ii,:) = sqrt( mean((abs(A_weight_L_w_MF_In_Situ(ii,:)) -
A_weight_L_w_3745_Full.').^2) );
    RMSD_C_weight(ii,:) = sqrt( mean((abs(C_weight_L_w_MF_In_Situ(ii,:)) -
C_weight_L_w_3745_Full.').^2) );
end

RMSD(5,1) = sqrt( mean((L_w_3741_nv.' - L_w_3745_Full(8:28).').^2) );
RMSD_A_weight(5,1) = sqrt( mean((A_weight_L_w_3741_nv.' - A_weight_L_w_3745_Full.').^2) );
```

```
RMSD_C_weight(5,1) = sqrt( mean((C_weight_L_w_3741_nv.' - C_weight_L_w_3745_Full.').^2) );

%% Plot and Compare the Sound Power Estimates
ref_source = ['Dodec','Single Driver','Horn'];

if Ref_Source == 1
    ref_num = 1:5;
end
if Ref_Source == 2
    ref_num = 6:18;
end
if Ref_Source == 3
    ref_num = 19:22;
end

%All EDs
figure(1)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(L_w_MF_In_Situ(1,:)),'b',freq_model,abs(L_w_MF_In_Situ(2,:)),'g',freq_mod
el,abs(L_w_MF_In_Situ(3,:)),'r',freq_model,abs(L_w_MF_In_Situ(4,:)),'k',f_3741,L_w_3741_nv,'m',f_
3745_Full(8:28),L_w_3745_Full(8:28),'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (100\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_Lw_MF_In_Situ(1),'%.1f') ' dB, RMSD = '
       num2str(RMSD(1),'%.1f') ' dB)'],['KED              '
       num2str(Overall_Lw_MF_In_Situ(2),'%.1f') ' dB               ' num2str(RMSD(2),'%.1f') '
       dB'],['TED             ' num2str(Overall_Lw_MF_In_Situ(3),'%.1f') ' dB               '
       num2str(RMSD(3),'%.1f') ' dB'],['GED              '
       num2str(Overall_Lw_MF_In_Situ(4),'%.1f') ' dB               ' num2str(RMSD(4),'%.1f') '
       dB'],['ISO 3741       ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB               '
       num2str(RMSD(5),'%.1f') ' dB'],['ISO 3745       ' num2str(Overall_L_w_3745_Full,'%.1f') '
```

```
       dB                  N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid

%Just PED and GED
figure(2)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(L_w_MF_In_Situ(1,:)),'g',freq_model,abs(L_w_MF_In_Situ(4,:)),'k',f_3741,L
_w_3741_nv,'m',f_3745_Full(8:28),L_w_3745_Full(8:28),'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (100\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm,, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (On-
Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num), ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_Lw_MF_In_Situ(1),'%.1f') ' dB, RMSD = '
       num2str(RMSD(1),'%.1f') ' dB)'],['GED            '
       num2str(Overall_Lw_MF_In_Situ(4),'%.1f') ' dB               ' num2str(RMSD(4),'%.1f') '
       dB'],['ISO 3741       ' num2str(Overall_L_w_3741_nv,'%.1f') ' dB               '
       num2str(RMSD(5),'%.1f') ' dB'],['ISO 3745       ' num2str(Overall_L_w_3745_Full,'%.1f') '
       dB                  N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid
```

```
%Just PED and GED (A-Weighted)
figure(3)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(A_weight_L_w_MF_In_Situ(1,:)),'g',freq_model,abs(A_weight_L_w_MF_In_Situ(
4,:)),'k',f_3741,A_weight_L_w_3741_nv,'m',f_3745_Full(8:28),A_weight_L_w_3745_Full,'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB_A)')
if Angle_num == 1
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (100\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_A_weight_Lw_MF_In_Situ(1),'%.1f') ' dB_A, RMSD = '
       num2str(RMSD_A_weight(1),'%.1f') ' dB)'],['GED              '
       num2str(Overall_A_weight_Lw_MF_In_Situ(4),'%.1f') ' dB_A               '
       num2str(RMSD_A_weight(4),'%.1f') ' dB'],['ISO 3741       '
       num2str(Overall_A_weight_L_w_3741_nv,'%.1f') ' dB_A               '
       num2str(RMSD_A_weight(5),'%.1f') ' dB'],['ISO 3745       '
       num2str(Overall_A_weight_L_w_3745_Full,'%.1f') ' dB_A
       N/A'],'Location','SouthEast')
if Source == 1
    ylim([35 85])
elseif Source == 2
    ylim([30 90])
elseif Source == 3
    ylim([10 100])
end
grid

%Just PED and GED (C-Weighted)
figure(4)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,abs(C_weight_L_w_MF_In_Situ(1,:)),'g',freq_model,abs(C_weight_L_w_MF_In_Situ(
4,:)),'k',f_3741,C_weight_L_w_3741_nv,'m',f_3745_Full(8:28),C_weight_L_w_3745_Full,'c')
xlabel('Frequency (Hz)')
ylabel('L_w (dB_C)')
```

```
if Angle_num == 1
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (100\circ
              Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (100\circ Off-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
if Angle_num == 2
    if Source == 1
        title(['DODEC in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 2
        title(['SINGLE DRIVER in Medium Room - L_w Estimates From the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
    if Source == 3
        title(['HORN in Medium Room - L_w Estimates From the Microflown Probe (On-Axis)(Probe
              Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'])
    end
end
legend(['PED (OAL = ' num2str(Overall_C_weight_Lw_MF_In_Situ(1),'%.1f') ' dB_C, RMSD = '
       num2str(RMSD_C_weight(1),'%.1f') ' dB)'],['GED              '
       num2str(Overall_C_weight_Lw_MF_In_Situ(4),'%.1f') ' dB_C              '
       num2str(RMSD_C_weight(4),'%.1f') ' dB'],['ISO 3741       '
       num2str(Overall_C_weight_L_w_3741_nv,'%.1f') ' dB_C              '
       num2str(RMSD_C_weight(5),'%.1f') ' dB'],['ISO 3745       '
       num2str(Overall_C_weight_L_w_3745_Full,'%.1f') ' dB_C
       N/A'],'Location','SouthEast')
if Source == 1
    ylim([50 90])
elseif Source == 2
    ylim([45 90])
elseif Source == 3
    ylim([30 100])
end
grid

%% Plots - Compare the In Situ Directivity Estimimates Between the Pressure and GED Methods

figure(5)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,ones(1,length(freq_model)),'r:',freq_model,gamma_Official,'g',freq_model,gamm
a(1,:),'b',freq_model,gamma(4,:),'m')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
if Source == 1
    if Angle_num == 1
        title(['DODEC in Medium Room - In Situ Direct. Factor Est. from the Microflown Probe
              (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
```

```
    elseif Angle_num == 2
        title(['DODEC in Medium Room - In Situ Direct. Factor Est. from the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
    ylim([0 10])
elseif Source == 2
    if Angle_num == 1
        title(['SINGLE DRIVER in Medium Room - In Situ Direct. Factor Est. from the Microflown
              Probe (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['SINGLE DRIVER in Medium Room - In Situ Direct. Factor Est. from the Microflown
              Probe (On-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
    ylim([0 20])
elseif Source == 3
    if Angle_num == 1
        title(['HORN in Medium Room - In Situ Direct. Factor Est. from the Microflown Probe
              (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['HORN in Medium Room - In Situ Direct. Factor Est. from the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
    ylim([0 45])
end
legend('Omni-directional','Anechoic Chamber','In Situ (PED)','In Situ
      (GED)','Location','NorthWest')
grid

%% Plots - Compare the In Situ Room Constant Estimimates Between the Pressure and 4 ED Methods
(Calculated using the known reference source directivity)

figure(6)
set(gcf,'Position',[0 0 scsz(3) scsz(4)])
semilogx(freq_model,R_Oct3,'g',freq_model,R_in_situ(1,:),'b',freq_model,R_in_situ(4,:),'m')
xlabel('Frequency (Hz)')
ylabel('Magnitude')
if Source == 1
    if Angle_num == 1
        title(['DODEC in Medium Room - In Situ Room Const. Est. from the Microflown Probe
              (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['DODEC in Medium Room - In Situ Room Const. Est. from the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
elseif Source == 2
    if Angle_num == 1
        title(['SINGLE DRIVER in Medium Room - In Situ Room Const. Est. from the Microflown Probe
              (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['SINGLE DRIVER in Medium Room - In Situ Room Const. Est. from the Microflown Probe
              (On-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
elseif Source == 3
```

```
    if Angle_num == 1
        title(['HORN in Medium Room - In Situ Room Const. Est. from the Microflown Probe
              (100\circ Off-Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    elseif Angle_num == 2
        title(['HORN in Medium Room - In Situ Room Const. Est. from the Microflown Probe (On-
              Axis)(Probe Dist. ' num2str(distances(r_ED_close),'%.2f') 'm, '
              num2str(distances(r_ED_far),'%.2f') 'm)(Ref. Direct. Src. - ', ref_source(ref_num),
              ')'] )
    end
end
legend('T60','In Situ (PED)','In Situ (GED)','Location','NorthWest')
ylim([0 170])
grid
```

findnearest.m

```
function [r,c,V] = findnearest(srchvalue,srcharray,bias)
% Usage:
% Find the nearest numerical value in an array to a search value
% All occurances are returned as array subscripts
% Output:
% For 2D matrix subscripts (r,c) use:
%       [r,c] = findnearest(srchvalue,srcharray,gt_or_lt)
% To also output the found value (V) use:
%       [r,c,V] = findnearest(srchvalue,srcharray,gt_or_lt)
% For single subscript (i) use:
%         i   = findnearest(srchvalue,srcharray,gt_or_lt)
% Inputs:
%    srchvalue = a numerical search value
%    srcharray = the array to be searched
%    bias      = 0 (default) for no bias
%                -1 to bias the output to lower values
%                 1 to bias the search to higher values
%                (in the latter cases if no values are found
%                 an empty array is ouput)
% By Tom Benson (2002)
% University College London
% t.benson@ucl.ac.uk

if nargin<2
    error('Need two inputs: Search value and search array')
elseif nargin<3
    bias = 0;
end

% find the differences
srcharray = srcharray-srchvalue;

if bias == -1   % only choose values <= to the search value
    srcharray(srcharray>0) =inf;
elseif bias == 1  % only choose values >= to the search value
    srcharray(srcharray<0) =inf;
end

% give the correct output
if nargout==1 | nargout==0
    if all(isinf(srcharray(:)))
        r = [];
    else
        r = find(abs(srcharray)==min(abs(srcharray(:))));
    end
elseif nargout>1
    if all(isinf(srcharray(:)))
        r = [];c=[];
    else
        [r,c] = find(abs(srcharray)==min(abs(srcharray(:))));
    end
```

```
    if nargout==3
        V = srcharray(r,c)+srchvalue;
    end
end
```

get_MicroF_Vals_Uncorrected.m

```
%% Microflown Probe with Singal Conditioner set to UNCORRECTED MODE
%This function takes a matrix of Microflown data (in V^2) of size (4,length(f))
%(pressure,ux,uy,uz)' and applies the probe-specific calibration. The
%outputs are frequency-dependent squared pressure magnitude, squared
%particle velocity magnitude, and the three squared particle velocity components.

function [MicroF_p_sq,MicroF_U_mag_sq_uc,MicroF_Ux_sq_uc,MicroF_Uy_sq_uc,MicroF_Uz_sq_uc] =
          get_MicroF_Vals_Uncorrected(f,DataMat)

 %% Input the Pressure Microphone Sensitivity Equations
 %Input the Pressure Sensitivity at 1 kHz
 S_p_at_1khz = 51.8;   %mV/Pa
 %Input the Sensititvity cornerfrequencies
 f_c1p = 21;            %Hz
 f_c2p = 182;           %Hz
 f_c3p = 16838;         %Hz
 %Input the Phase cornerfrequencies
 c1p = 20;             %Hz
 c2p = 164;            %Hz
 c3p = 18288;          %Hz
 %Input the Frequency-Dependent Sensitivity Equation
 S_p = S_p_at_1khz * (sqrt(1+(f./f_c3p).^2)./(sqrt(1+(f_c1p./f).^2).*sqrt(1+(f_c2p./f).^2)));
%mV/Pa
 %Input the Frequency-Dependent Pressure Phase Response of the Probe
 theta_p = atand(c1p./f) + atand(c2p./f) + atand(f./c3p);   %deg

 %% Input the BLUE Particle Velocity Sensor Sensitivity Equations
 %(Blue = x Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_x = 24.284;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_x = 89;           %Hz
 f_c2u_x = 584;          %Hz
 f_c3u_x = 5110.44;      %Hz
 f_c4u_x = 196;          %Hz
 %Input the Phase cornerfrequencies
 c1u_x = 36;           %Hz
 c2u_x = 337;          %Hz
 c3u_x = 26278.11;     %Hz
 c4u_x = 325;          %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
 S_u_x_uc =  S_u_250Hz_x ./
(sqrt(1+(f_c1u_x./f).^2).*sqrt(1+(f/f_c2u_x).^2).*sqrt(1+(f/f_c3u_x).^2).*sqrt(1+(f_c4u_x./f).^2)
          );  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in Corrected Mode
 theta_u_x = atand(c1u_x./f) + atand(c4u_x./f);        %deg

 %% Input the RED Particle Velocity Sensor Sensitivity Equations
 %(Red = Y Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_y = 30.17;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_y = 142;        %Hz
 f_c2u_y = 692;        %Hz
 f_c3u_y = 4870;       %Hz
 f_c4u_y = 141;        %Hz
 %Input the Phase cornerfrequencies
 c1u_y = 44;           %Hz
 c2u_y = 362;          %Hz
 c3u_y = 20632;        %Hz
 c4u_y = 359;          %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
```

```
 S_u_y_uc =  S_u_250Hz_y ./
(sqrt(1+(f_c1u_y./f).^2).*sqrt(1+(f/f_c2u_y).^2).*sqrt(1+(f/f_c3u_y).^2).*sqrt(1+(f_c4u_y./f).^2)
);  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in
 %Corrected Mode
 theta_u_y = atand(c1u_y./f) + atand(c4u_y./f);        %deg

 %% Input the GREEN Particle Velocity Sensor Sensitivity Equations
 %(Green = Z Direction)
 %Input the Sensitivity in HIGH GAIN
 S_u_250Hz_z = 8.314;     %V/(m/s) (Microflown Cal)
 %Input the Sensititvity cornerfrequencies
 f_c1u_z = 66;         %Hz
 f_c2u_z = 571;        %Hz
 f_c3u_z = 7858;       %Hz
 f_c4u_z = 66;         %Hz
 %Input the Phase cornerfrequencies
 c1u_z = 21;           %Hz
 c2u_z = 555;          %Hz
 c3u_z = 15846;        %Hz
 c4u_z = 65;           %Hz
 %Input the Frequency-Dependent Sensitivity Equation in UNCORRECTED MODE
        S_u_z_uc =  S_u_250Hz_z ./
(sqrt(1+(f_c1u_z./f).^2).*sqrt(1+(f/f_c2u_z).^2).*sqrt(1+(f/f_c3u_z).^2).*sqrt(1+(f_c4u_z./f).^2)
);  %uncorrected mode
 %Input the Frequency-Dependent Velocity Phase Response Equation in
 %Corrected Mode
 theta_u_z = atand(c1u_z./f) + atand(c4u_z./f);        %deg

 %% Convert the voltages to pressure and particle velocity magnitudes
 MicroF_p_sq = ( (sqrt((DataMat(1,:)))*1000) ./ S_p ).^2;  %Pa^2
 MicroF_Ux_sq_uc = ( sqrt(DataMat(2,:)) ./ S_u_x_uc ).^2;   %(m/s)^2
 MicroF_Uy_sq_uc = ( sqrt(DataMat(3,:)) ./ S_u_y_uc ).^2;   %(m/s)^2
 MicroF_Uz_sq_uc = ( sqrt(DataMat(4,:)) ./ S_u_z_uc ).^2;   %(m/s)^2
 MicroF_U_mag_sq_uc = MicroF_Ux_sq_uc + MicroF_Uy_sq_uc + MicroF_Uz_sq_uc;  %(m/s)^2

end
```

getRho.m

```
%This function calculates the density of air based on pressure (in mB),
%temperature (degrees Celsius), and relative humidity (in percent). The two
%outputs are calculated for humid air and dry air. The humid air
%calculation should be more accurate.

function [rho_humid,rho_dry] = getRho(pressure,temp,humidity)
    R = 287.05;
    P = pressure*100;
    R_d = 287.05;
    T = temp + 273.15;
    p_sat = 6.1078*10^( (7.5*temp)/(temp + 237.3) );
    P_v = humidity*p_sat;
    rho_humid = ( P/(R_d*T) ) * ( 1-((0.378*P_v)/P) );
    rho_dry = (P/(R*T));
end
```

MyOct3Bands.m

```
%FOR ARRAYS
%This function takes narrow band squared data magnitudes (proportional to
%intensity) and converts them into third-octave (squared summed energy)
%bands. It also outputs a new frequency array that corresponds to the
%third-octave bands to which the narrow band data were assigned.

function [f_Oct3,data_Oct3] = MyOct3Bands(f_narrow,data_narrow_sq)
    oct_c = [                          0.8   1     1.25  1.6   2 ...
```

```
        2.5   3.15  4     5     6.3   8     10    12.5  16    20 ...
        25    31.5  40    50    63    80    100   125   160   200 ...
        250   315   400   500   630   800   1000  1250  1600  2000 ...
        2500  3150  4000  5000  6300  8000  10000 12500 16000 20000 ...
        25000 31500 40000 50000 63000 80000 100000];

    if f_narrow(1) < 0.8
         f_narrow(1) = 0.8;
    end

    BandNum = round(10*log10(f_narrow));
    bands = unique(BandNum);
    data_Oct3 = zeros(length(bands),1);

    for i = 1:length(bands)
        inds = BandNum == bands(i);
        data_Oct3(i) = sum(data_narrow_sq(inds));
    end

    f_Oct3 = oct_c(bands+2).';
end
```

MyOct3Bands_MAT.m

```
%FOR 3-D MATRICES of size  (:,:,freqs)
%This function takes narrow band squared data magnitudes (proportional to
%intensity) and converts them into third-octave (squared summed energy)
%bands. It also outputs a new frequency array that corresponds to the
%third-octave bands to which the narrow band data were assigned.

function [f_Oct3,data_Oct3] = MyOct3Bands_MAT(f_narrow,data_narrow_sq)
    oct_c = [                         0.8   1     1.25  1.6   2 ...
        2.5   3.15  4     5     6.3   8     10    12.5  16    20 ...
        25    31.5  40    50    63    80    100   125   160   200 ...
        250   315   400   500   630   800   1000  1250  1600  2000 ...
        2500  3150  4000  5000  6300  8000  10000 12500 16000 20000 ...
        25000 31500 40000 50000 63000 80000 100000];

    if f_narrow(1) < 0.8
         f_narrow(1) = 0.8;
    end

    BandNum = round(10*log10(f_narrow));
    bands = unique(BandNum);
    data_Oct3 = zeros(length(data_narrow_sq(:,1,1)),length(data_narrow_sq(1,:,1)),length(bands));

    for i = 1:length(bands)
        inds = BandNum == bands(i);
        data_Oct3(:,:,i) = sum(data_narrow_sq(:,:,inds),3);
    end

    f_Oct3 = oct_c(bands+2).';
end
```