ABSTRACT

GARNER III, GLENWOOD. Nonlinear Acoustic Characterization of Targets. (Under the direction of Professor Hamid Krim).

Recent techniques in nonlinear vibro-acoustics have demonstrated improved sensing capabilities for landmine detection. These methods however, place the transmit and/or receive devices extremely close to a potentially dangerous target. This paper discusses a novel approach where ultrasonic parametric arrays are used to achieve excitation at standoff ranges in air. When two frequencies, f_1 and f_2 are directed to excite a target, the nonlinear response consists of sum and difference frequencies.

The difference frequency may be carefully swept to produce an acoustic signature of the target, reflecting its size and density information. As part of this research, a more accurate third order nonlinear ultrasonic propagation model is developed to analyze signal strength and frequency at the target. Due to the inefficient mixing of the ultrasonic tones, reflected signals have very small amplitude. This thesis develops high-resolution spectral analysis techniques (e.g. multiple signal classification (MUSIC) algorithm) to extract particularly weak signals (in low signal to noise ratio scenario) and thus substantially improve target characteristics estimation performance and provides a viable and practical approach to perform acoustic imaging. Experimental results demonstrate for the first time, a capacity to remotely classify a hollow target from a solid one, with resonance patterns predicting the approximate size of the target.

Report Documentation Page					*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE **2008**		2. REPORT TYPE			3. DATES COVERED **00-00-2008 to 00-00-2008**
4. TITLE AND SUBTITLE **Nonlinear Acoustic Characterization of Targets**					5a. CONTRACT NUMBER
					5b. GRANT NUMBER
					5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)					5d. PROJECT NUMBER
					5e. TASK NUMBER
					5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **North Carolina State University,Department of Electrical and Computer Engineering,Raleigh,NC,27695**					8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)					10. SPONSOR/MONITOR'S ACRONYM(S)
					11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT **see report**					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**107**	

Nonlinear Acoustic Characterization of Targets

by
Glenwood Garner III

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Electrical Engineering

Raleigh, North Carolina

2008

Approved By:

Dr. Wesley E. Snyder

Dr. Michael B. Steer

Dr. Hamid Krim
Chair of Advisory Committee

DEDICATION

To my parents: Glen and Vicky Garner.

Who have taught me more than any person, book, or university ever can.

And to my sister: Leigh Garner.

Who has been the best role model any little brother can have.

BIOGRAPHY

Glen Garner completed his undergraduate studies at NC State University in December of 2005. After a brief employment at Caterpillar, he returned to NC State in the Fall of 2006 to pursue a Master's Degree. He is a member of the National Society of Collegiate Scholars, Tau Beta Pi, IEEE, the Graduate Student Association, and Eta Kappa Nu. Glen is very active in extracurricular activities, competing in the IEEE Hardware Design contest, presiding over Eta Kappa Nu, and working as an Engineering Entrepreneur Mentor. Since the Spring of 2007, he has been working under a joint research effort between Dr. Hamid Krim and Dr. Michael B. Steer. His research interests include standoff acoustic analysis, nonlinear acoustic propagation modeling, and RF-acoustic applications. Glen will continue his graduate studies under the direction of Dr. Steer following the completion of his Master's Degree.

ACKNOWLEDGMENTS

The research for this thesis would not be possible without the support of Dr. Hamid Krim, Dr. Michael B. Steer, and Dr. Kevin Gard. This material is based upon work funded by the U. S. Army Research Laboratory and the U. S. Army Research Office under grant number W911NF-07-1-0004.

In addition, I would like to thank Tara Britt for recognizing the potential I didn't know I had and teaching me that you can always accomplish more.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	**Unit**	**Description**
α	$\mathrm{Np} \cdot \mathrm{m}^{-1}$	Attenuation coefficient
$\alpha_{classical}$	$\mathrm{Np} \cdot \mathrm{m}^{-1}$	Classical attenuation coefficient
α_{th}	$\mathrm{Np} \cdot \mathrm{m}^{-1}$	Thermal attenuation coefficient
α_{tv}	$\mathrm{Np} \cdot \mathrm{m}^{-1}$	Thermoviscous attenuation coefficient
α_v	$\mathrm{Np} \cdot \mathrm{m}^{-1}$	Viscous attenuation coefficient
β	unit less	Second order nonlinear coefficient
B_e	Hz	Frequency resolution
c	$\mathrm{m} \cdot \mathrm{s}^{-1}$	Sound speed
c_0	$\mathrm{m} \cdot \mathrm{s}^{-1}$	Equilibrium sound speed
C_p	$\mathrm{J} \cdot \mathrm{mol}^{-1} \cdot \mathrm{K}^{-1}$	Specific heat at constant pressure
ϕ	rad	Phase
f	Hz	Frequency
f_r	Hz	Resonant frequency
F_s	Hz	Sample rate
γ	unit less	Ratio of specific heats
η	unit less	Third order nonlinear coefficient
I_{REF}	$\mathrm{W} \cdot \mathrm{m}^{-2}$	Reference intensity
J_n	unit less	n^{th} order Bessel function of first kind
k	$\mathrm{rad} \cdot \mathrm{m}^{-1}$	Wave number
κ	$\mathrm{W}(\mathrm{m} \cdot \mathrm{K})^{-1}$	Heat conduction coefficient
λ	m	Wavelength
l	m	Length
L_r	dB	Return loss
L_t	dB	Transmission loss
μ	$\mathrm{Pa} \cdot \mathrm{s}$	Shear viscosity coefficient
μ_B	$\mathrm{Pa} \cdot \mathrm{s}$	Bulk viscosity coefficient
M	unit less	Acoustic Mach number
n	mol	Molecular amount
p	Pa	Excess pressure
p_0	Pa	Equilibrium pressure
P	Pa	Total pressure
P_0	Pa	Excitation pressure
Pr	unit less	Prandtl number
P_{REF}	Pa	Reference pressure
P_{RMS}	Pa	Root mean square pressure
ρ	$\mathrm{kg} \cdot \mathrm{m}^{-3}$	Density
ρ_0	$\mathrm{kg} \cdot \mathrm{m}^{-3}$	Equilibrium density

Symbol	**Unit**	**Description**
$\delta\rho$	$\mathrm{kg \cdot m^{-3}}$	Excess density
R	$\mathrm{J \cdot K^{-1} \cdot mol^{-1}}$	Ideal gas constant
R_p	unit less	Pressure reflection coefficient
S	$\mathrm{m^2}$	Surface area
t	s	Time
T	K	Absolute temperature
T_e	s	Sampling interval
T_p	unit less	Pressure transmission coefficient
u	$\mathrm{m \cdot s^{-1}}$	Fluid particle velocity
u_0	$\mathrm{m \cdot s^{-1}}$	Fluid particle excitation velocity
V	$\mathrm{m^3}$	Volume
$\widetilde{V}$	$\mathrm{Pa \cdot s}$	Viscosity number
υ	$\mathrm{m^2 \cdot s^{-1}}$	Kinematic viscosity coefficient
ω	$\mathrm{rad \cdot s^{-1}}$	Radian frequency
x	m	Euclidean coordinate distance
x_D	m	Discontinuity distance
Z	$\mathrm{Pa \cdot s \cdot m^{-1}}$	Acoustic impedance

Chapter 1

Introduction

1.1 Motivation

The use of sound, especially at ultrasonic frequencies, has been used extensively for the purposes of biological imaging and non-destructive testing. These methods are limited in their imaging abilities for two key reasons: in non-destructive testing, the transducer must be coupled to the test material by a liquid medium for impedance matching so as to transmit as much energy as possible into the test object. In addition to this limitation, ultrasound is only able to measure range by calculating the time delay of a reflected signal. Such techniques rely on orthogonal reflections from acoustically reflective surfaces. Furthermore, there is inherent noise in linear acoustic systems because the reflected, measured acoustic energy is often masked by the transmitted energy, which is always at the same frequency.

Recently, techniques that use two frequencies, and the nonlinear effect in air or other media have shown promise in gathering even more information about our environment. This information may include not only range and shape data, but also resonance and density measurements. This can lead to better examination techniques for buried landmines and archeological artifacts, which form the focus of this thesis. Furthermore, research in this field can improve techniques for air coupled ultrasonic inspection where a large impedance mismatch limits the ability to transmit high-energy signals from one media to another[5].

Current technologies that are used for the detection of buried objects rely on the transmission and measurement of directed energy. These methods include ground penetrating radar, infrared, neutron activation analysis, and acoustics[6]. Most of these technologies, however, are limited in their ability to differentiate targets from other debris in the soil.

Furthermore, they must heavily rely on assumptions about the target or surrounding media such as density, compliance, and moisture. Donskoy[7] proposed a nonlinear acoustic technique for landmine detection in which two tones are used to insonify the soil. Here insonify is used to describe the process of exposing the target to acoustic or sonar energy[8]. The key assumption to this approach is that the top plate of the landmine must have a stiffness less than or equal to that of the surrounding soil. Because of the compliant nature of the landmine top plate, a nonlinear interaction is created between it and the soil above. It is this nonlinear interaction, shown in Figure 1.1, which creates a difference frequency, indicating the presence of a buried compliant object or landmine. This technique is superior to other

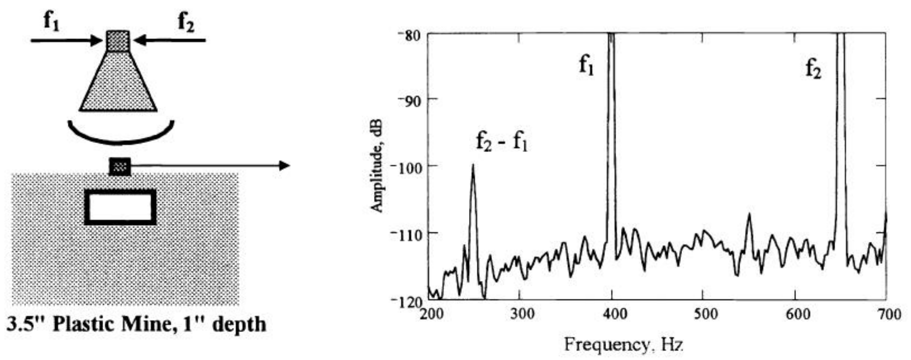


Figure 1.1: Acoustic spectrum of 3.5” plastic landmine. Excitation signals are visible at f_1 = 400 Hz and f_2 = 650 Hz. The nonlinearity at $f_2 - f_1$ = 250 Hz indicates the presence of a landmine[1].

current forms of acoustic detection because the excitation signals are not needed in the detection stage. In linear detection, information about the target is contained in the reflection of the excitation signal. This reflected energy is usually very weak and easily masked by the incident energy as well as reflections from many non-target objects such as the surface of the soil, rocks, inhomogeneities, and other battlefield debris. Because the reflected signal is at a different frequency than the excitation signals, nonlinear vibro-acoustic techniques are much more discriminating than other techniques. Although Donskoy and Korman[2] have demonstrated the viability of this technique, both rely on geophones and loudspeakers that must be placed dangerously close to the target. Figure 1.2 shows Korman’s test setup where the loudspeakers and geophones must be placed directly above the landmine.

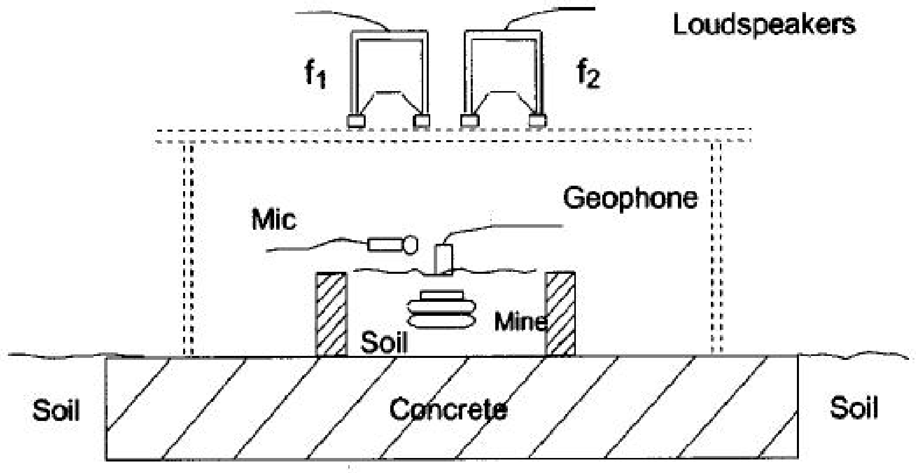


Figure 1.2: Test setup for Korman's 2002 nonlinear two-tone acoustic test for VS 2.2 landmine. The mine is buried at a depth of 3.6 cm and located 7.9 cm above the concrete base. The loudspeakers, microphone, and geophone must be placed directly above the mine[2].

1.2 Contribution

This thesis proposes the use of highly directional ultrasonic parametric arrays to provide nonlinear excitation of targets at standoff ranges. Using the concept of "scattering of sound by sound", we can use two pure tones, f_1 and f_2, to generate a difference frequency in air. This concept was presented by Yoneyama[4] and is discussed in greater detail in Section 1.3.4. This difference frequency may be swept to provide a frequency response or acoustic signature of a target. The acoustic signature indicates at which frequencies a target absorbs and reflects energy and possibly the existence of resonant behaviors. While detecting buried landmines is still beyond the scope of this research, discriminating between solid and hollow targets, and observing resonant behaviors can still provide useful information. An example application for determining resonant behaviors is in archeological digging. If the resonant length of a partially buried fossil is known, time can be saved in the process of unearthing the object. The ability to discriminate between hollow and solid targets can also provide a useful tool in the drug war. A car door, which is supposed to be hollow, may have a different acoustic response if it is packed with drugs. This thesis describes the techniques used to examine several different targets and the results that validate our approach.

The remainder of this chapter serves as a review of acoustic-driven techniques.

Chapter 2 presents a signal processing block diagram and discusses the techniques used to analyze the data. It also includes an air propagation model proposed to improve acoustic imaging techniques in the future. Chapter 3 describes the experimental setup and discusses the necessity to build an anechoic chamber in support of this research. Finally, Chapter 4 reveals the results of this research, and concludes with what future work may include in this field.

1.3 Review of Acoustic Principles

Before delving into nonlinear acoustic detection, a review of the fundamentals of acoustics is necessary. The remainder of this chapter discusses fundamental wave propagation, basic acoustic interactions, and what assumptions are made for the work presented in this thesis. This will provide the reader with appropriate background knowledge to understand the physical phenomena that take place.

1.3.1 Ideal Linear Wave Equation

The ideal linear wave equation, which is presented here, is derived from first principles by Blackstock[3]. In this derivation, the concept of a fluid particle is introduced in order to neglect the random path and velocities of individual molecules. A fluid particle's size can be defined as the smallest volume spanned by a length of the same order magnitude as the average mean free path of a molecule. For air, this length l is approximately 0.1 mm and the number of molecules contained in l^3 is about 25,000. This volume, shown in Figure 1.3 is used to derive the equation of continuity. In Figure 1.3, ρ defines the density of the fluid, u defines the fluid velocity, and S defines the surface area of the flow. The net change in the mass of this fixed volume can be written mathematically as

$$\frac{\partial}{\partial t}(S\rho\Delta x) = \rho u S|_x - \rho u S|_{x+\Delta x}. \tag{1.1}$$

Rearranging terms and taking the limit as $\Delta x \to 0$ yields the equation of continuity,

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0. \tag{1.2}$$

The next physical property to model is the conservation of momentum. Figure 1.4 models the momentum flow and pressure forces acting on the fluid particle. The momentum

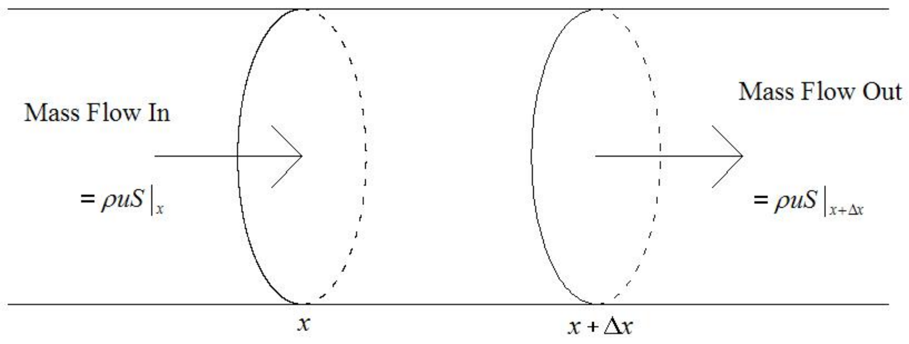


Figure 1.3: Mass flow through the surface of a cylindrical fluid particle[3].

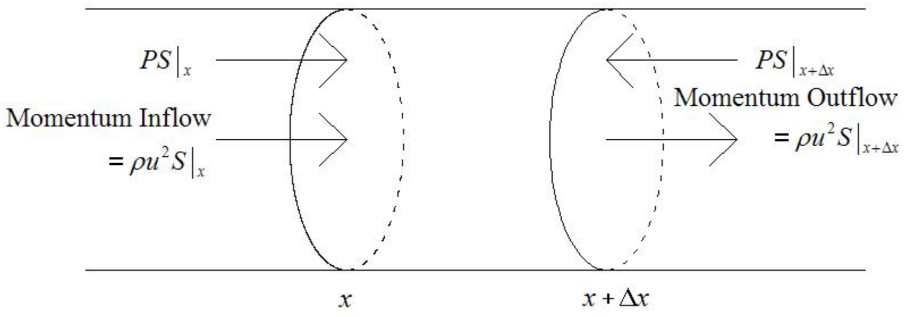


Figure 1.4: Momentum inflow and outflow and forces acting on cylindrical fluid particle[3].

inflow and outflow as well as the pressure acting on the fluid particle can be modeled using

$$\frac{\partial}{\partial t}(\rho u S \Delta x) = \rho u^2 S|_x - \rho u^2 S|_{x+\Delta x} + PS|_x - PS|_{x+\Delta x}, \tag{1.3}$$

where P is the pressure acting on the surface of the fluid particle. Equation (1.3) can be simplified by dividing through by $S\Delta x$ and applying Equation (1.2). Taking the limit as $\Delta x \to 0$ yields the conservation of momentum equation,

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x}\right) + \frac{\partial P}{\partial x} = 0. \tag{1.4}$$

Finally, the ideal linear wave equation is based upon a thermodynamic equation of state such as the ideal gas law, $PV = nRT$, where V is a gas volume, n is the number of moles, R is the ideal gas constant, and T the absolute temperature. For gases, the most frequently used isentropic equation of state is the adiabatic gas law,

$$\left(\frac{P}{p_0}\right) = \left(\frac{\rho}{\rho_0}\right)^{\gamma} \tag{1.5}$$

where γ is the ratio of specific heats and p_0 and ρ_0 are the static values of P and ρ respectively. A more useful equation of state valid for both liquids and gases is a Taylor series expansion of the general isentropic equation of state about the condensation $(\rho - \rho_0)/\rho_0$ so that

$$P = p_0 + A\left(\frac{\rho - \rho_0}{\rho_0}\right) + \frac{B}{2!}\left(\frac{\rho - \rho_0}{\rho_0}\right)^2 + \frac{C}{3!}\left(\frac{\rho - \rho_0}{\rho_0}\right)^3 + \cdots . \tag{1.6}$$

The coefficients A, B, C... are determined from experimental observations or calculated using nonlinear approximations. Equation (1.6) will be later used to provide a third-order nonlinear wave equation approximation. Introducing the variable for sound speed

$$c^2 = \frac{\partial P}{\partial \rho}, \tag{1.7}$$

and rearranging terms, Equation (1.6) becomes

$$p = c_0^2 \delta\rho \left[1 + \frac{B}{2!A}\frac{\delta\rho}{\rho_0} + \frac{C}{3!A}\left(\frac{\delta\rho}{\rho_0}\right)^2 + \cdots\right]. \tag{1.8}$$

In Equation (1.8), p represents the excess pressure defined by $p \equiv P - p_0$ and $\delta\rho$ is the excess density defined by $\delta\rho \equiv \rho - \rho_0$. Equations (1.2), (1.4), and (1.8) may be made linear by using a small signal approximation. This approximation is valid for even the loudest sounds that are experienced on a day-to-day basis. The small signal approximation is based on the assumption that excitation pressures, which affect density and particle velocity, are very small in comparison to steady-state values. This can be seen by the fact that 134 dB sound pressure level (SPL), which is equivalent to an excitation pressure of 100 Pa, is miniscule in comparison to 101.325 kPa ambient air pressure. Using the assumption that $|p| \ll \rho_0 c_0^2$, we can linearize Equations (1.2), (1.4), and (1.8) to yield

$$\frac{\partial \delta\rho}{\partial t} + \rho\frac{\partial u}{\partial x} = 0, \tag{1.9}$$

$$\rho\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = 0, \tag{1.10}$$

$$p = c_0^2 \delta\rho. \tag{1.11}$$

Combining Equations (1.9–1.11) produces a single second order differential equation of pressure with respect to distance and time,

$$c_0^2\frac{\partial^2 p}{\partial x^2} - \frac{\partial^2 p}{\partial t^2} = 0. \tag{1.12}$$

Equation (1.12) is the linear, non-dissipative, acoustic, planar wave equation and has a real solution of the form

$$p(x,t) = P_0\cos(\omega t - kx), \tag{1.13}$$

where P_0 is the excitation amplitude, ω is the frequency in radians, and $k = \omega/c$ is the wave number. This solution to the wave equation, as well as those presented in Sections 1.3.2 and 1.3.3 are for one-dimensional plane wave propagation, which assumes that wave fronts are uniform in the y-z plane and located in the acoustic far field defined by [9]

$$x \gg \frac{d^2}{\lambda}, \tag{1.14}$$

where d is the diameter of an acoustic source and $\lambda = c/f$ is the wavelength of the acoustic signal.

1.3.2 Lossy Linear Wave Equation

Dissipation of an acoustic signal is primarily due to thermoviscous losses associated with the media in which it travels. Dissipation is often ignored for audible frequencies in air because the acoustic attenuation coefficient, α, is proportional to the square of the frequency. However, as frequency increases to ultrasonic levels, the dissipation reaches levels high enough to limit propagation to only a few meters. In order to develop a wave equation to model this effect, we must consider the equations of continuity, momentum, and state [3]. Viscous effects do not alter the continuity equation, and are ignored in the linearized equation of state because they appear as higher order terms. Only the equation of momentum is changed in this derivation, adding an additional term to the right-hand side yielding[3]

$$\rho_0 \frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = \mu \tilde{V} \frac{\partial^2 u}{\partial x^2}, \tag{1.15}$$

where μ is the shear viscosity coefficient and $\tilde{V} = 4/3 + \mu_B/\mu$. Equations (1.9), (1.11), and (1.15) are now combined to yield the linear lossy wave equation of pressure with respect to distance and time,

$$\frac{\upsilon \tilde{V}}{c_0^2} \frac{\partial^2 p}{\partial x^2} \frac{\partial p}{\partial t} + \frac{\partial^2 p}{\partial x^2} - \frac{1}{c_0^2} \frac{\partial^2 p}{\partial t^2} = 0, \tag{1.16}$$

where $\upsilon = \mu/\rho_0$ is the kinematic viscosity coefficient. Equation (1.16) has a real solution of the form

$$p(x,t) = P_0 e^{-\alpha x} \cos(\omega t - kx), \tag{1.17}$$

where α is the attenuation coefficient in Nepers per meter. The attenuation coefficient, α, can be calculated for both viscous and thermal dissipation. Typically, attenuation coefficients are computed for viscous effects and altered heuristically to account for thermal effects. Both viscous and thermal attenuation are represented here using

$$\alpha_v = \frac{\tilde{V}\upsilon\omega^2}{2c_0^3} \tag{1.18}$$

$$\alpha_{th} = \frac{(\gamma-1)\kappa\omega^2}{2\rho_0 c_0^3 C_p}, \tag{1.19}$$

where κ is the heat conduction coefficient and C_p is the specific heat at constant pressure. These two absorption coefficients can be summed to produce the total absorption coefficient

$$\alpha_{tv} = \frac{\omega^2\upsilon}{2c_0^3}\left[\left(\frac{4}{3}+\frac{\mu_B}{\mu}\right)+\frac{\gamma-1}{Pr}\right], \tag{1.20}$$

where Pr is the Prandtl number and $\mu_B/\mu = 0.61$. Equation (1.20) is very close to the classical definition of acoustic attenuation given by[10][11]

$$\alpha_{classical} = \frac{\omega^2\upsilon}{2c_0^3}\left(\frac{4}{3}+\frac{\gamma-1}{Pr}\right). \tag{1.21}$$

For air at 20°C, $Pr = 0.711$, $\gamma = 1.402$, $\upsilon = 15.11\times10^{-6}\ m^2/s$, and $c = 343$ m/s, Equation (1.20) yields a thermoviscous absorption coefficient of $1.84\times10^{-11}\ f^2$, where f is in hertz.

1.3.3 Lossy Nonlinear Wave Equation

When amplitude and frequency become very large, small signal models tend to break down and lose accuracy due to the nonlinearity of air. In this case, a nonlinear propagation model must be derived to account for the formation of higher order harmonics. The nonlinear derivation here will focus on the state equation presented in Equation (1.6), while keeping terms of order greater than one. Referring back to the original state equation given in Equation (1.5), and taking the derivative indicated in Equation (1.7), an expression for sound speed is given by

$$c^2 = \frac{\gamma P}{\rho} = \gamma RT. \tag{1.22}$$

It is important to observe that P, ρ, and T in Equation (1.22) are total, non-static, values, which now make sound speed non-constant. Combining Equations (1.5) and (1.22), we can express P and ρ in terms of c as

$$P = \rho_0\left(\frac{c}{c_0}\right)^{\frac{2\gamma}{(\gamma-1)}}. \tag{1.23}$$

Using Equation (1.23), we can eliminate P and ρ from the continuity and momentum equations and rewrite them as

$$\frac{Dc}{Dt} + \frac{\gamma - 1}{2} c \nabla \cdot \mathbf{u} = 0 \tag{1.24}$$

$$\frac{D\mathbf{u}}{Dt} + \frac{2}{\gamma - 1} c \nabla \cdot c = 0, \tag{1.25}$$

respectively. For plane waves, Equations (1.24) and (1.25) reduce to

$$\frac{\partial c}{\partial t} + u \frac{\partial c}{\partial x} + \frac{\gamma - 1}{2} c \frac{\partial u}{\partial x} = 0 \tag{1.26}$$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{2}{\gamma - 1} c \frac{\partial c}{\partial x} = 0. \tag{1.27}$$

Equations (1.26) and (1.27) represent a second order system that can be reduced by only taking into consideration the forward traveling wave component represented by

$$\frac{\partial u}{\partial t} + (c_0 + \beta u) \frac{\partial u}{\partial x} = 0, \tag{1.28}$$

with nonlinear parameter $\beta = (\gamma + 1)/2$. After expanding u as a Taylor series and performing extensive algebraic manipulations, a second order solution to Equation (1.28) of the pressure p with respect to x and t can be written as

$$p(x,t) = P_0 \left(\sin(\omega t - kx) + \frac{1}{2} \beta M k x \sin 2(\omega t - kx) \right), \tag{1.29}$$

where $M = u_0/c_0$. It is clear from Equation (1.29) that the second harmonic amplitude is directly proportional to frequency, amplitude, and propagation distance and can be ignored if all three are small enough. If the Mach number is expressed with respect to pressure, $M = P_0/(Z_0 c_0)$, the second order nonlinear equation may be written as

$$p(x,t) = P_0 \sin(\omega t - kx) + \frac{P_0^2 \beta \omega x}{2 Z_0 c_0^2} \sin 2(\omega t - kx), \tag{1.30}$$

where Z_0 is the characteristic impedance of air. Since the second harmonic is dependent on the square of the amplitude, large excitation pressures are the predominant cause of harmonic distortion. This is evident at a rock concert where music sounds distorted due to large propagation distances and extreme amplification. Combining Equation (1.30) with the results obtained in Section 1.3.2 results in the lossy nonlinear second order wave equation

$$p(x,t) = P_0 e^{-\alpha_1 x} \sin(\omega t - kx) + \frac{P_0^2 \beta \omega x}{2 Z_0 c_0^2} e^{-\alpha_2 x} \sin 2(\omega t - kx). \tag{1.31}$$

Higher order solutions to the nonlinear wave equation exist, but very little is know about higher order nonlinear parameters for air. If air is treated as an ideal gas, the second order nonlinear parameter is $\beta = 1.2$. In Chapter 2, a third order nonlinear air model is derived using a perturbation method.

1.3.4 Modulation of Sound by Sound

This section discusses the physical mechanisms involved in using ultrasonic parametric arrays for standoff analysis of targets. In 1982, Yoneyama[4] discussed the nonlinear interaction of ultrasound with air as the "scattering of sound by sound". This work is primarily based on the nonlinear wave equation derived by Westervelt[12] to account for deficiencies of classical models at high frequencies. Westervelt's answer to the high frequency problem is the addition of sum and difference frequencies, which accounts for the nonlinear interaction between air molecules and is expressed as

$$\begin{aligned} \nabla^2 p_s - \frac{1}{c_0^2}\frac{\partial^2 p_s}{\partial t^2} &= -\rho_0 \frac{\partial q}{\partial t}, \\ q &= \frac{\beta}{\rho_0^2 c_0^4}\frac{\partial}{\partial t} p_1^2. \end{aligned} \tag{1.32}$$

In Equation (1.32), p_s is the secondary wave pressure, p_1 is the primary wave pressure, and β is the second order nonlinear parameter. The nonlinear interaction of air molecules is due to the difference in the forces in the compression and tension phases of acoustic propagation. At any point along its propagation axis, a longitudinal acoustic wave produces areas of high and low pressure. In the high-pressure (compression) phase of propagation, two neighboring air molecules move together with minimal compression. In the low-pressure (tension) phase of propagation, the retreating air molecule has a lesser pull on its neighbor due to the weak attraction between them. There is slight delay before the molecule being pulled on can catch up, producing a "slapping" or cavitation effect. This produces a rectification at higher frequencies just as a diode does in an electrical circuit.

This natural rectification allows the demodulation of an AM waveform. In the simplest interactions, two pure tones f_1 and f_2 mix to produce second order inter-modulation products of $f_1 \pm f_2$. Yoneyama elaborated on this simple phenomenon to demodulate an AM waveform with secondary pressure given by

$$p_s(x,t) = p_1 \left[1 + mg\left(t - \frac{x}{c_0}\right)\right] e^{-\alpha x} \sin\omega\left(t - \frac{x}{c_0}\right), \tag{1.33}$$

where m is the modulation index and g represents an arbitrary base band signal. As in Equation (1.32), p_s is the secondary pressure, which in this application seems to emanate from thin air. Furthermore, because the modulating waveform g is carried by a high frequency signal with wavelength much smaller than the size of the transducer, these signals are highly directional as shown in Figure 1.5. The only significant drawback of using the

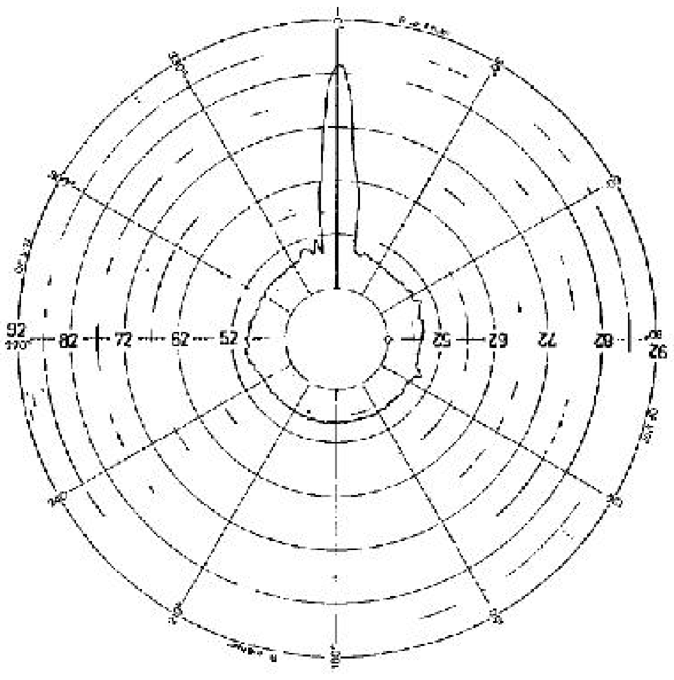


Figure 1.5: Directivity of secondary wave at 10.0 kHz, for a point at 4 m, $m = 0.5$, and input voltage of 10 V [4].

nonlinear effect of air to demodulate ultrasonic signals is the poor efficiency of the demodulation process. Another way of expressing Equation (1.33) is

$$p_s(t) = \frac{\beta P_1^2 A}{16\pi\rho_0 c_0^4 x\alpha} \frac{\partial^2}{\partial t^2} E^2 \left(t - \frac{x}{c_0} \right). \tag{1.34}$$

Here A is the cross sectional area of the transducer and $E(t)$ is the modulation envelope. Inspecting Equation (1.34), it can be seen that the secondary pressure is proportional to the second derivative of the square of the modulation envelope. The second derivative term produces a slope in the frequency domain of 12 dB per octave, and the square term adds significant distortion in double-sideband AM modulation. As an example, using Equation (1.34) with $\beta = 1.2$, $A = 0.2\ m^2$, $\alpha = 0.7$, $c_0 = 348\ m/s$, and $\rho_0 = 1.18\ kg/m^3$, a 130 dB carrier modulated with a 1 kHz signal produces about 66 dB of audible sound at 1 m. With the ability to only place sound intensities in the 60 dB range on the surface of the target, it is apparent why high resolution spectral techniques, like those discussed in Chapter 2, are necessary for this type of detection.

1.3.5 Decibel Scale

The time is taken here to discuss logarithmic scales as they are used in the field of acoustics. This is a necessary evolution, as it helps to understand the relevant intensities that are invoked in a comparison to Ohm's law as it relates to electricity. Table 1.1 relates the basic measurable acoustic parameters to their equivalent electrical properties.

Table 1.1: Conversion between acoustic and electric variables

Acoustic Variable	**Electric Variable**
Pressure (P) Pascals	Voltage (V) Volts
Particle Velocity (u) m/s	Current (I) Amps
Acoustic Impedance (Z) $Pa \cdot s/m^3$	Resistance (R) Ohms
Sound Intensity (I) W/m^2	Power (P) Watts

This is helpful in understanding the derivation of sound pressure level (SPL) as a means of representing acoustic power. Using the relationship between all four acoustics parameters in Table 1,

$$Z = \frac{P}{u} = \frac{I}{u^2} = \frac{P^2}{I}, \tag{1.35}$$

it is easy to see that sound intensity is the same as electric power. Just as logarithms are used to express electric power with respect to a reference power, they are used to express sound intensity with respect to a reference intensity of 20 μPa. This is considered to be the lowest intensity perceivable by the human ear at 2 kHz. Representing sound intensity in logarithmic scale can be accomplished by either of the following equations,

$$SPL_{dB} = 10\log_{10}\left(\frac{I}{I_{ref}}\right) = 20\log_{10}\left(\frac{P_{RMS}}{P_{ref}}\right). \tag{1.36}$$

This representation is useful in measuring the wide range of amplitudes that are typical in everyday environments. Table 1.2 presents several examples of sound pressure level with respect to excitation amplitudes from various sources.

Table 1.2: Examples of sound pressure and corresponding sound pressure level (SPL)

Source	**Sound Pressure**	**Sound Pressure Level**
1 atmosphere	101325 Pa	191.085 dB
Thermoacoustic device	12000 Pa	176 dB
Jet engine at 30 m	630 Pa	150 dB
Threshold of pain	100 Pa	130 dB
Jack hammer at 1 m	2 Pa	100 dB
Hearing damage	6×10^{-1} Pa	85 dB
Passenger car at 10 m	$2 \times 10^{-2} - 2 \times 10^{-1}$ Pa	60-80 dB
Normal talking at 1 m	$2 \times 10^{-3} - 2 \times 10^{-2}$ Pa	40-60 dB
Very calm room	$2 \times 10^{-4} - 6 \times 10^{-4}$ Pa	20-30 dB
Auditory threshold at 2 kHz	20×10^{-6} Pa	0 dB

1.3.6 Acoustic Absorption and Reflection

Because this thesis addresses the interaction of ultrasonic acoustics on solid targets, a discussion of the interaction of sound at a boundary is included. Just as in transmission line theory, when a voltage traveling on a line with characteristic impedance Z_0 encounters a circuit with a different characteristic impedance Z_l, some energy is reflected while the load absorbs the rest. Ultrasonic techniques for non-destructive testing were first employed by Sokoloff[13] in 1929, and have become commonplace today with the growth of microelectronics and high sensitivity transducers. However, due to the large impedance mismatch between air and solids, most ultrasonic inspection techniques require the test item to be immersed in water or use a gel coupling medium[5]. Neither of these methods are suitable for standoff detection and until recently, technology did not exist to efficiently transmit ultrasound for non-contact measurements[14]. With the recent advent of piezo-electric ceramic transducers incorporating a polymer matching layer, parametric arrays can be constructed to provide highly directional, high amplitude ultrasound.

In this derivation, only plane waves of normal incidence will be considered to avoid rigorous trigonometric equations. The incident, reflected, and transmitted values for pressure, intensity, and particle velocity will be defined by subscripts i, r, and t respectively and can been seen in Figure 1.6. Just as in electromagnetic theory, the characteristic impedance describes the conductive properties of a medium and is defined for air by

$$Z_0 = \rho_0 c_0. \tag{1.37}$$

Conservation of energy requires the use of boundary conditions at $x = 0$ to define the

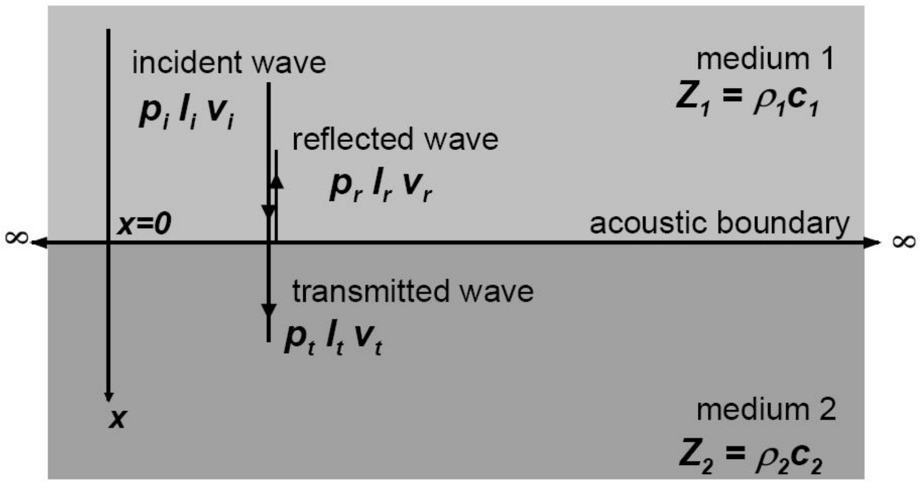


Figure 1.6: Transmission and reflection of an acoustic wave at normal incidence at a boundary of two materials with different impedance [5].

forward and backward traveling wave with pressure p_r, particle velocity u_r, and intensity I_r defined by

$$\begin{aligned} u_i + u_r &= u_t|_{x=0}, \\ p_i + p_r &= p_t|_{x=0}, \\ I_i - I_r &= I_t|_{x=0}. \end{aligned} \tag{1.38}$$

Solving for the incident, reflected, and transmitted pressures with respect to density, sound speed, and particle velocity yields

$$\begin{aligned} p_i &= \rho_1 c_1 u_i|_{x=0}, \\ p_r &= \rho_1 c_1 u_r|_{x=0}, \\ p_t &= \rho_2 c_2 u_t|_{x=0}. \end{aligned} \tag{1.39}$$

By combining Equations (1.39) and (1.37), Equation (1.38) becomes

$$\begin{aligned} u_i + u_r &= u_t|_{x=0}, \\ u_i Z_1 - u_r Z_1 &= u_t Z_2|_{x=0} \end{aligned} \tag{1.40}$$

and

$$\begin{aligned} p_i + p_r &= p_t|_{x=0}, \\ \frac{p_i}{Z_1} - \frac{p_r}{Z_1} &= \frac{p_t}{Z_2}|_{x=0}. \end{aligned} \tag{1.41}$$

If we define the reflection coefficient as the ratio of the reflected wave to the incident wave, Equations (1.38), (1.40), and (1.41) can be combined to get

$$R_p = \frac{p_r}{p_i} = \frac{Z_2 - Z_1}{Z_2 + Z_1}. \tag{1.42}$$

The transmission coefficient is defined in a similar fashion and is computed using

$$T_p = \frac{p_t}{p_i} = \frac{2Z_1}{Z_2 + Z_1}. \tag{1.43}$$

Table 1.3 lists the acoustic impedance of several common materials and the reflection and transmission coefficient for each one with respect to air. The values listed help further clarify why a high-resolution spectral technique is needed to analyze resonant responses in this standoff nonlinear acoustic detection technique.

Table 1.3: Reflection and transmission coefficients for some common materials with respect to air, $Z_{air} = 413.5$ MRayl.

Material	**Acoustic Impedance (Rayl)**	**Reflection Coefficient**	**Transmission Coefficient**
Air	413.5	0	1
Water	1.48×10^6	0.99944	0.00055
Wood (pine)	1.57×10^6	0.99947	0.00052
Fiberglass	2.86×10^6	0.99971	0.00028
Concrete	8.00×10^6	0.99989	0.00010
Glass	12.5×10^6	0.99993	0.00006
Sand	19.7×10^6	0.99995	0.00004
Steel	46.0×10^6	0.99998	0.00001

Chapter 2

Signal Processing

2.1 Introduction

This chapter focuses on the development of signal models and signal processing algorithms to overcome the low signal to noise ratio (SNR) discussed in Chapter 1. Due to the high loss of the nonlinear demodulation of ultrasonic signals and the poor acoustic coupling to the target, reflected signals are very close to the noise floor. For the anechoic chamber used in these experiments, the noise floor associated with acoustic noise plus measurement noise is approximately 15 dB. However, the noise floor and SNR are closely related to the sampling frequency and sample size. If measurements are made using a sampling rate of 500 kHz and taking 500,000 samples, a noise floor of 6-8 dB is achievable, but highly impractical due to computation time.

What is required is a high-resolution technique that is capable of analyzing relatively small data samples. The goal is to sweep the difference frequency in sufficiently small increments to observe narrowband phenomenon in the frequency response. Unfortunately, the frequency resolution is inversely proportional to the SNR due to the large size of the data set. For example, if we want a frequency sweep from 0-10 kHz in 100 Hz increments, and have a sample rate of 200 kHz with $N = 20,000$ samples, our data set has 2 million measurements. It is easy to see how quickly data sets can grow if more frequency resolution, bandwidth, or SNR is needed.

2.2 Third Order Nonlinear Air Model

Before signal processing can be performed on any data set, a comprehensive data model must be established to define the signal to be analyzed. For example, the data model using Fourier analysis defines the data to be periodic in time. Data models also help predict how a signal varies to changes in the medium which it propagates in. An example directly related to this research would be the change in sound speed with respect to temperature and pressure, which is the cause of its nonlinear behavior. Scientists have used nonlinear analysis to characterize microwave devices, biological tissues, organic liquids, and especially metal components [15, 16, 17, 18]. Nonlinear characterization has been especially useful in characterizing small bubbles and defects within living tissue and metal castings. The underlying theory points to the fact that defects and air pockets significantly impact second order nonlinear harmonic generation. Thus a measurement of the nonlinearity of a material compared to what it is known to be can detect defects that are not visible using other methods. Chen[16] reports that higher harmonics ($n \geq 3$) possess superior characteristics for improved clarity and sharper contrast in biological imaging.

Third order nonlinear models have been developed for biological tissues, metals, and microwave devices for both their academic understanding and prospective applications [16, 19]. Nonlinear behavior of acoustics in air, especially at ultrasonic frequencies, have been of interest to scientists for quite some time. However, due to the lack of advanced measurement equipment such as piezoelectric microphones and transducers, nonlinear ultrasonic acoustics has mostly remained a theoretical interest. With the advent of matching layers to transmit/capture a significant amount of energy from a transducer, the field of ultrasonic acoustics has experienced a resurgence of activity. Both the fields of biology and microwave technology understand the value of accurate behavioral models, which should carry over to the field of acoustics. Furthermore, the nonlinear model presented in this section goes beyond the treatment of air as an ideal gas, and the limitations imposed by other third order models.

2.2.1 Bessel Approximation

A classical treatment of nonlinear acoustics is the Bessel-Fubini solution, which allows one to trace the growth of higher order harmonics. The modeling of acoustic waves may be done with the parameters of pressure (p), particle velocity (u), and displacement

(ξ). In this treatment, we will consider the parameter of particle velocity, and represent the ideal linear wave equation, Equation (1.13) with respect to particle velocity as[20]

$$u(x,t) = u_0 \cos(\omega t - kx). \tag{2.1}$$

If we define sound speed with respect to particle velocity as[5]

$$c = c_0 + \frac{\gamma + 1}{2} u, \tag{2.2}$$

and consider the second order parameter of nonlinearity β, we can describe the distortion of a sinusoidal waveform mathematically as

$$u(x,t) = u_0 \sin\left(\omega\left(t - \frac{x}{c_0 + \beta u}\right)\right) \qquad 0 \leq x \leq x_D. \tag{2.3}$$

In Equation (2.3), x_D represents the discontinuity distance, or the distance at which the waveform changes from a sinusoidal nature to an almost sawtooth shape. This phenomenon is caused by the difference in wave propagation speed between the crests and troughs. The crests represent high pressure (high density) areas while the troughs represent low pressure (low density) areas. Since sound travels faster in higher density media, the crest of the wave tries to overtake the trough, forming a sawtooth shape as can be seen in Figure 2.1. The distance at which this occurs is based on the frequency and amplitude of the excitation waveform and is represented as

$$x_D = \frac{1}{k\beta M}, \tag{2.4}$$

where M is the acoustic Mach number $M = u_0/c_0$. The frequency content of Equation (2.3) can better be seen by representing it as a Fourier series:

$$u(x,t) = u_0 \sum_{n=1}^{\infty} B_n \sin(n(\omega t - kx)), \qquad x < x_D \tag{2.5}$$

$$B_n = \frac{1}{\pi} \int_0^{2\pi} \sin(\omega t - kx + \beta) \sin(n(\omega t - kx)) d(\omega t - kx). \tag{2.6}$$

The integration of Equation (2.6) yields

$$B_n = 2\frac{x_D}{nx} J_n\left(\frac{nx}{x_D}\right), \tag{2.7}$$

for n^{th} order Bessel functions, J_n, of the first kind. Putting Equation (2.7) into Equation (2.5), one obtains a final solution with respect to pressure of[5, 20]

$$p(x,t) = 2P_0 \sum_{n=1}^{\infty} \frac{J_n\left(\frac{nx}{x_D}\right)}{\left(\frac{nx}{x_D}\right)} \sin(n(\omega t - kx)), \qquad x < x_D. \tag{2.8}$$

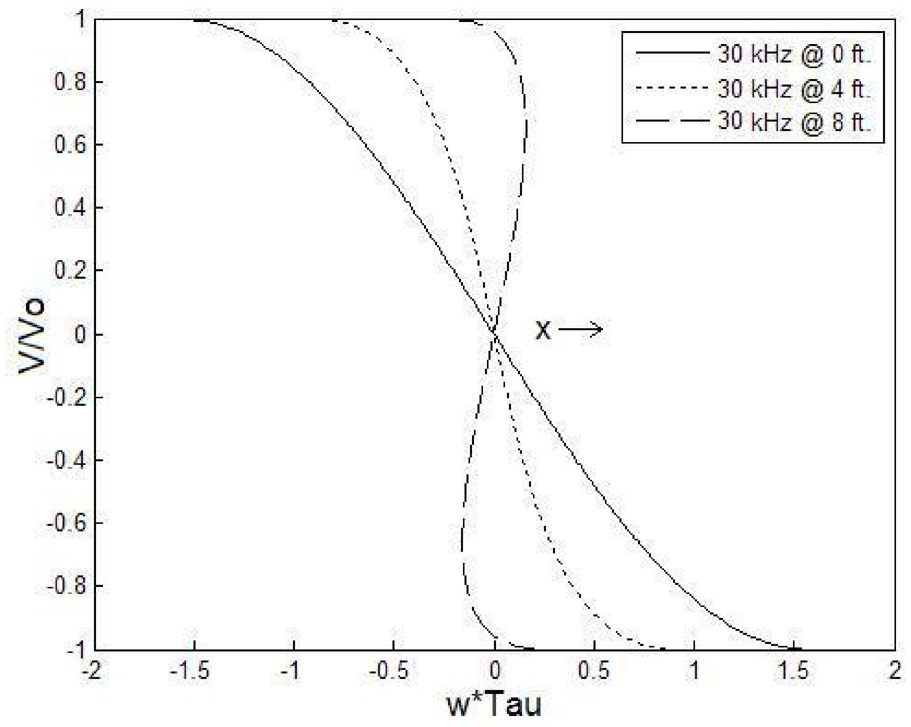


Figure 2.1: Image of a 30 kHz, 117 Pa sine wave being distorted by nonlinear effects of the air. Solid line represent propagation distance $x = 0$, dotted line $x = 4$ m, and dashed line $x = 8$ m is the theoretical waveform shape beyond discontinuity distance $x = x_D$.

This equation describes the spectral content of the higher order harmonics as their amplitudes grow at the expense of the fundamental frequency. Notice however that this model only works for distances less than the discontinuity distance, x_D. This severely limits the usefulness of this approximation for high-frequency, high-amplitude signals since the discontinuity distance may be very small.

2.2.2 Derivation of Nonlinear Parameters

One solution to providing a more accurate model at further ranges than the Bessel-Fubini equation is called the perturbation method. This approach begins with the nonlinear equation of state repeated here for convenience,[3]

$$P = p_0 + A\left(\frac{\rho-\rho_0}{\rho_0}\right) + \frac{B}{2!}\left(\frac{\rho-\rho_0}{\rho_0}\right)^2 + \frac{C}{3!}\left(\frac{\rho-\rho_0}{\rho_0}\right)^3 + \cdots . \tag{2.9}$$

In Equation (2.9), B/A and C/A are the ratio of the quadratic to linear terms and cubic to linear terms of the Taylor series expansion. Their definitions can be represented by partial derivatives of pressure with respect to density, evaluated at equilibrium conditions and constant entropy, which are denoted by subscripts "0" and "s"[16]. The first, second, and third order terms of Equation (2.9) can be solved for using

$$A = \rho_0 \left(\frac{\partial p}{\partial \rho}\right)_{s,0} \tag{2.10}$$

$$B = \rho_0^2 \left(\frac{\partial^2 p}{\partial \rho^2}\right)_{s,0} \tag{2.11}$$

$$C = \rho_0^3 \left(\frac{\partial^3 p}{\partial \rho^3}\right)_{s,0}. \tag{2.12}$$

Noticing that $(\partial p/\partial \rho) = c_0^2$ in Equation (2.10), the equation for A can be simplified. Taking the ratio of B/A and C/A further reduces the complexity to only second order derivatives of sound speed with respect to pressure given by[21]

$$A = \rho_0 c_0^2 \tag{2.13}$$

$$\frac{B}{A} = 2\rho_0 c_0 \left(\frac{\partial c}{\partial \rho}\right)_{s,0} \tag{2.14}$$

$$\frac{C}{A} = \frac{3}{2}\left(\frac{B}{A}\right)^2 + 2\rho^2 c^3 \left(\frac{\partial^2 c}{\partial p^2}\right)_s. \tag{2.15}$$

The problem with evaluating Equations (2.14) and (2.15) is finding a nonlinear equation for sound speed in air. Typically, air is taken as an ideal gas, and sound speed is linearized to $\sqrt{\gamma RT/M}$. This problem may be overcome by rearranging Equation (1.23), which is derived from the adiabatic state equation and definition of sound speed, to be

$$c = c_0 \left(\frac{P}{p_0}\right)^{\frac{\gamma-1}{2\gamma}}. \tag{2.16}$$

This definition for sound speed is infinitely differentiable and easily provides a solution to Equations (2.14) and (2.15). The first and second derivatives of Equation (2.16) are given by

$$\frac{\partial c}{\partial p} = \frac{\left(\frac{P}{p_0}\right)^{\frac{\gamma-1}{2\gamma}} (\gamma-1)c_0}{2\gamma P} \tag{2.17}$$

$$\frac{\partial^2 c}{\partial p^2} = \frac{\left(\frac{P}{p_0}\right)^{\frac{\gamma-1}{2\gamma}} (\gamma-1)^2 c_0}{4\gamma^2 P^2} - \frac{\left(\frac{P}{p_0}\right)^{\frac{\gamma-1}{2\gamma}} (\gamma-1)c_0}{2\gamma P^2}. \tag{2.18}$$

These derivatives produce solutions to Equation (2.9) of $B/A = .3998$ and $C/A = -.2398$, which are very close to the values for an ideal gas of $B/A = \gamma - 1$ and $C/A = \gamma - 2$, where $\gamma = 1.4$ for air. It is important to notice from Equations (2.17) and (2.18) that the nonlinear parameters $\beta = 1 + \frac{1}{2}\frac{B}{A}$ and $\eta = 1 + \frac{1}{2}\frac{C}{A}$ are proportional to the excitation pressure, which is in keeping with the results of Chen and Keller.

2.2.3 Perturbation Analysis

Using the results for the nonlinear parameters β and η obtained from Section 2.2.2, we can apply a perturbation solution to solve for the amplitudes of higher order harmonics. Following the derivation given by Chen, we can model a finite amplitude plane wave in a lossless medium using[16]

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0 \\ &\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \frac{\partial p}{\partial x} = 0 \\ &p = p(\rho, s). \end{aligned} \tag{2.19}$$

It is possible to find a solution to Equation (2.19) by expressing ρ, p, and v as a series,

$$\begin{aligned} \rho &= \sum_{n=0}^{\infty} \rho^{(n)} \\ u &= \sum_{n=0}^{\infty} u^{(n)} \\ p &= \sum_{n=0}^{\infty} p^{(n)}, \end{aligned} \tag{2.20}$$

and substituting these back into Equation (2.19). We can now independently express the amplitudes of the second and third harmonics as second order partial differential equations of pressure with respect to distance and time, and given by

$$\frac{\partial^2 p^{(2)}}{\partial x^2} - \frac{1}{c_0^2}\frac{\partial^2 p^{(2)}}{\partial t^2} = -\frac{\partial}{\partial x}\left(\rho_0^{-1}c_0^2(\beta-1)\frac{\partial(\rho^{(1)})^2}{\partial x} + \rho^{(1)}\frac{\partial u^{(1)}}{\partial t} + \rho_0 u^{(1)}\frac{\partial u^{(1)}}{\partial x}\right)$$
$$+\rho_0^{-1}c_0^2(\beta-1)\frac{\partial^2(\rho^{(1)})^2}{\partial x^2} + \frac{\partial^2 \rho^{(1)}u^{(1)}}{\partial x\partial t} - \rho_0^{-1}c_0^2(\rho-1)\frac{\partial^2(\rho^{(1)})^2}{\partial t^2}, \tag{2.21}$$

$$\frac{\partial^2 p^{(3)}}{\partial x^2} - \frac{1}{c_0^2}\frac{\partial^2 p^{(3)}}{\partial t^2} = -2\rho_0^{-1}(\beta-1)\frac{\partial^2}{\partial t^2}(\rho^{(1)}\rho^{(2)}) - \rho_0^{-2}(\eta-1)\frac{\partial^2}{\partial t^2}(\rho^{(1)})^3$$
$$-\frac{\partial}{\partial x}\left(\rho^{(1)}\frac{\partial u^{(2)}}{\partial t} + \rho^{(2)}\frac{\partial u^{(1)}}{\partial t} + \rho_0 u^{(1)}\frac{\partial u^{(2)}}{\partial x} + \rho^{(1)}u^{(1)}\frac{\partial u^{(1)}}{\partial x} + \rho_0 u^{(2)}\frac{\partial u^{(1)}}{\partial x}\right)$$
$$+\frac{\partial^2}{\partial t\partial x}(\rho^{(1)}u^{(2)} + \rho^{(2)}u^{(1)}). \tag{2.22}$$

In Equations (2.21) and (2.22), the superscript in parenthesis refers to the order of the harmonic and is not an exponent. Assuming the wave at $x = 0$ is sinusoidal, Equations (2.21) and (2.22) can be solved with respect to the pressures $p^{(2)}$ and $p^{(3)}$ respectively. Summarizing the total solution, we can represent the third order nonlinear propagation model as

$$p^{(1)}(x,t) = P_0 e^{-\alpha_1 x}\sin(\omega t - kx) \tag{2.23}$$
$$p^{(2)}(x,t) = \frac{\beta\omega P_0^2 x}{2\rho_0 c_0^3}e^{-\alpha_2 x}\sin(2(\omega t - kx)) \tag{2.24}$$
$$p^{(3)}(x,t) = GfP_0^3(Ff\beta^2x^2 + 2\eta x)e^{-\alpha_3 x}\sin(3(\omega t - kx)) \tag{2.25}$$
$$G = \frac{\pi}{2\rho_0^2 c_0^5} \tag{2.26}$$
$$F = \frac{6\pi}{c_0}, \tag{2.27}$$

where the $e^{-\alpha x}$ term has been added to each harmonic to account for dissipative losses in a medium. The individual harmonics can then be summed to produce the final solution of

$$p(x,t) = \sum_{n=1}^{3} p(x,t)^{(n)}. \tag{2.28}$$

Figure 2.2 shows how the waveform changes as it propagates in the positive x direction. Notice how the approximations all look similar until the discontinuity distance x_D, where the Bessel-Fubini solution breaks down. At this point, the perturbation and ideal gas solution become more accurate as the amplitude of the Bessel-Fubini approximation falls too low. These plots look time reversed from those in Figure 2.1 because they are plotted with respect to time and not distance, thus flipping them left to right. Figure 2.3 shows a more detailed view of how the amplitudes of the first, second, and third harmonics of the perturbation, Bessel-Fubini, and ideal gas approximations grow and decay with distance. Finally, Figure 2.4 shows how the total acoustic power decreases with distance. These plots can be reproduced and altered by using the “Air_Model_GUI.m” program given in Appendix C.

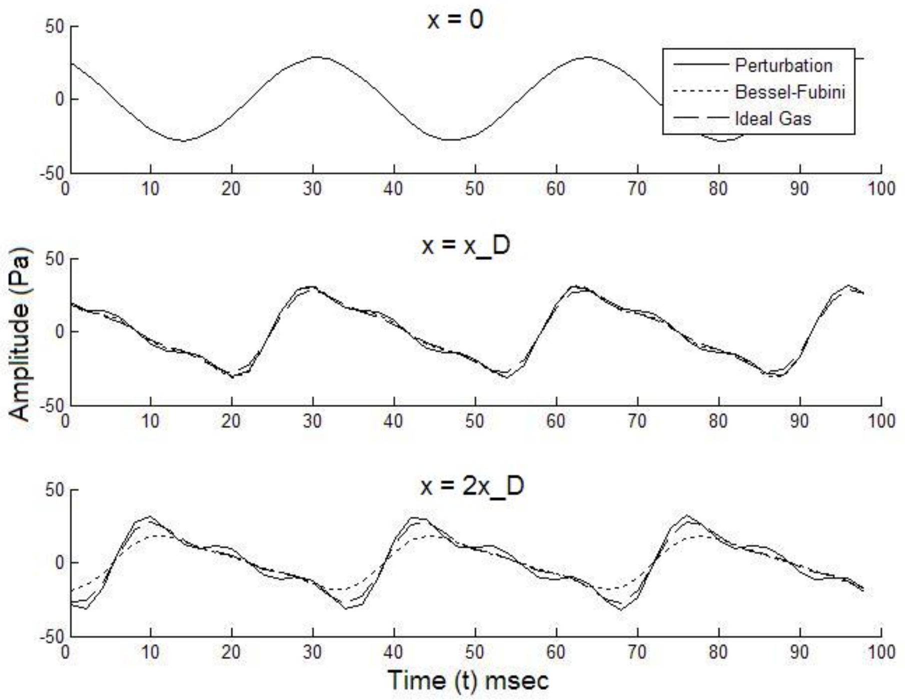


Figure 2.2: 30 kHz sine wave at 120 dB SPL amplitude plotted at $x = 0$, $x = x_D$, and $x = 2x_D$. Notice how the Bessel-Fubini approximation falls away beyond $x = x_D$, which is where this model becomes invalid.

2.2.4 Fourier Transform Analysis

The Fourier Transform was initially used in our spectral analysis, but was quickly dismissed due to the necessity of long data sets to achieve sufficient SNRs. However, the use of the Discrete Fourier Transform (DFT) was an integral part of our processing algorithm, in its role of implementing a prewhitening filter. On this account, we provide here a brief discussion on its application.

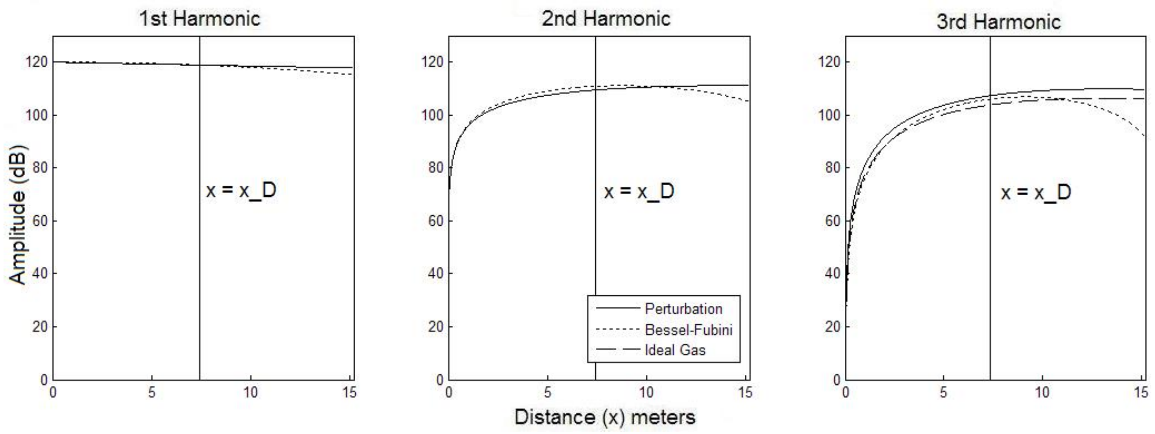


Figure 2.3: Harmonic amplitudes of 30 kHz sine wave at 120 dB SPL amplitude. Notice how all three amplitudes are reasonably close for $x < x_D = 7.6$ m, which is where the Bessel-Fubini approximation becomes invalid and the perturbation approximation becomes more accurate.

Time domain signals of incident and reflected pressure waveforms are fundamental to a pursued detection procedure. While the incident waveform does not contain any data about the composition of the target, it does provide a reference energy level as a means of comparing the amount of acoustic energy absorbed by solid and hollow targets of the same shape. The incident waveform also confirms that the parametric array is correctly pointed at the target, ensuring that it is receiving maximum power. In order to maintain the best spectral resolution from a DFT, we must preserve the time-bandwidth product[22]

$$T_e B_e = 1. \tag{2.29}$$

In Equation (2.29),

$$B_e = \frac{1}{T_e} = \frac{1}{NT} = \frac{F_s}{N}, \tag{2.30}$$

where T_e is the observation interval, B_e is the frequency resolution, N is the number of samples, and F_s the sampling rate. By capturing 50,000 samples at a sample rate of 50 kHz, we are able to achieve a frequency resolution of 1 Hz, which is the maximum achievable by the signal generators being used. The calculated frequency resolution is then used in generating the frequency "bins" in the single sided DFT using[22]

$$X[k] = \frac{2}{N} \sum_{n=0}^{N-1} x[n] e^{-j2\pi k \frac{n}{N}} \tag{2.31}$$

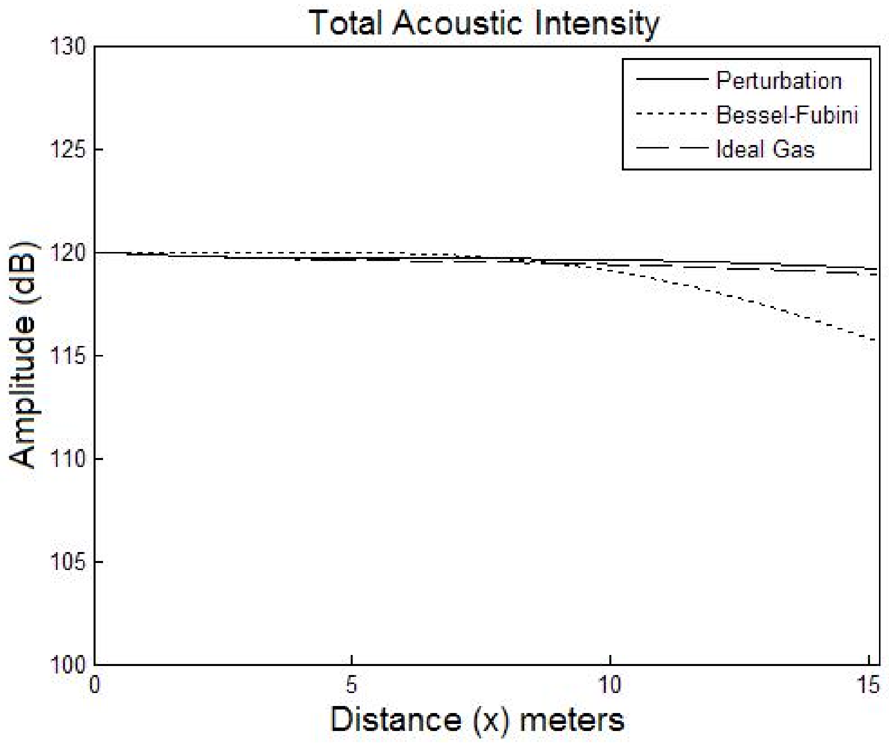


Figure 2.4: Total acoustic intensity of 30 kHz sine wave at 120 dB SPL amplitude plotted with respect to distance. Notice how all propagation models start at 120 dB at $x = 0$, and how the Bessel-Fubini model becomes more inaccurate beyond $x = x_D$.

and mapped to the frequency domain using the substitution

$$X[f] = X[k]|_{k=\frac{fN}{F_s}} \tag{2.32}$$

Equation (2.31) is implemented as a MATLAB function call "general_ft.m" and a "mex_fft.m" executable to improve run time. The code for these functions is available in Appendix A. Also necessary in implementing the prewhitening filter, is an Inverse Discrete Fourier Transform (IDFT) to reconstruct the filtered time domain signal. This IDFT takes the real and imaginary Fourier coefficients and produces a time domain representation while preserving the phase information. If the phase information is neglected in the reconstruction process, an accurate time domain signal is still produced with respect to magnitude. This provides an ability to build "perfect" filters in the frequency domain and transform them to the time domain for convolution with an input signal. Reconstruction of a time domain signal from the single sided DFT is computed using

$$x[n] = \sum_{k=0}^{N-1} X[k] e^{j2\pi k \frac{n}{N}}, \tag{2.33}$$

where $X[k]$ is a complex Fourier coefficient. Since the DFT and IDFT are implemented in C, all values must be "real". In order to reconstruct the original using only real values, Equation (2.33) can be rewritten as

$$x[n] = \sum_{k=0}^{N-1} Re(X[k]) \cos(2\pi k \frac{n}{N}) - Im(X[k]) \sin(2\pi k \frac{n}{N}). \tag{2.34}$$

Equation (2.34) is implemented as a MATLAB function call "general_ift.m" and a "mex_ifft.m" executable as well. Code for these functions is available in Appendix B. Figure 2.1 (A) shows an input signal with frequency domain representation (B), perfect reconstruction (C), and no-phase reconstruction (D).

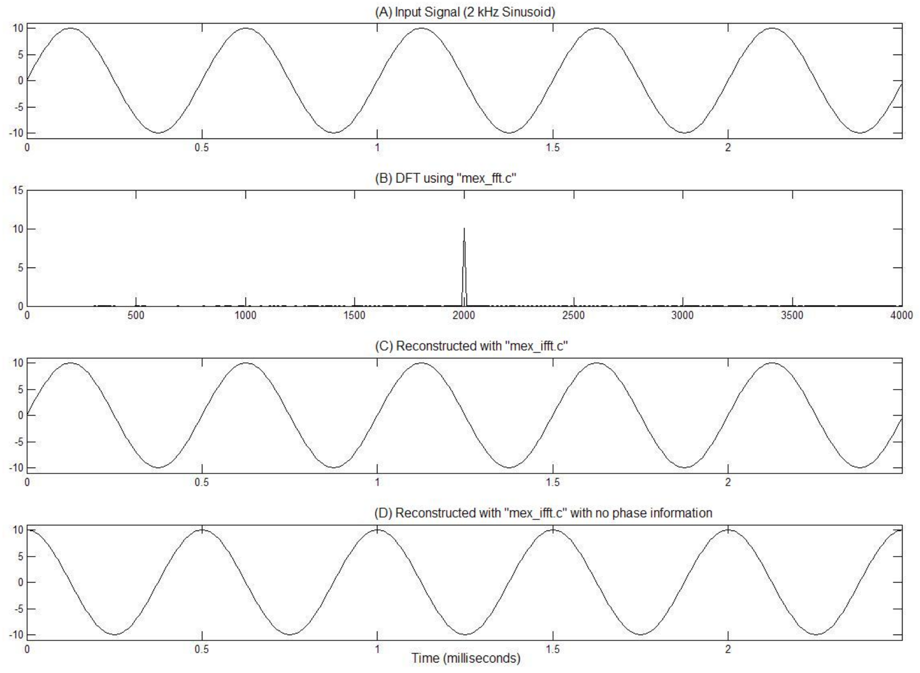


Figure 2.5: Demonstration of "mex_fft.c" and "mex_ifft.c" with 2 kHz input signal (A), frequency domain representation (B), perfect reconstruction (C), and no-phase reconstruction (D).

The Multiple Signal Classification (MUSIC) algorithm was chosen due to its superior performance in comparison to the DFT. However, the MUSIC algorithm makes two assumptions about the input data. First, it assumes that the data is zero mean. Second, it assumes that the additive noise is white. This initially presented a problem, as the measurement equipment that is used generates $1/f$ "pink" noise, otherwise known as flicker noise when referring to electronic devices[23]. Figure 2.2 depicts the signal processing block diagram that was developed to condition the data prior to implementing the MUSIC algorithm. The first operation that is performed on the data set is to average it so that

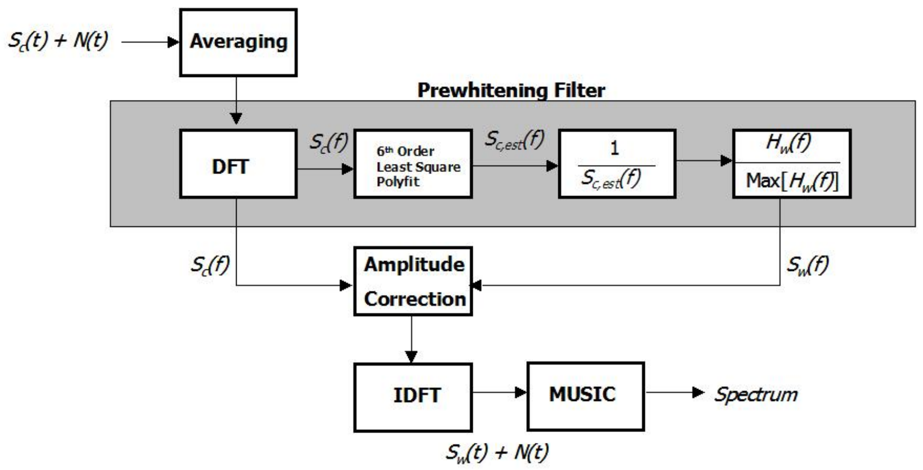


Figure 2.6: Signal processing block diagram

its mean is zero. Since our signals are theoretically purely sinusoidal, this step is largely precautionary. However, if there is a DC component added by the measurement equipment, this step will eliminate it. Averaging is simply and mathematically described by

$$x[n]_{avg} = x[n] - E\{x[n]\}, \tag{2.35}$$

where E represents the expected value of a one dimensional vector (time series). The input signal applied to the prewhitening filter, thus will have white noise.

2.2.5 Prewhitening Filter

As previously stated, the MUSIC algorithm assumes that the time domain input signal is zero mean and contains only white noise. The purpose of the prewhitening filter is to remove the $1/f$ noise generated by the measurement equipment. To this end, we must first obtain an estimate of the noise and then use that estimate to generate an appropriate filter. Since this detection algorithm sweeps a single tone across a bandwidth of interest, it is assumed that the prewhitening filter is only acting on one sinusoidal frequency embedded in pink noise at any given time. It has been shown though, that this filter is capable of whitening any noise spectrum containing narrowband signals. Proceeding with a DFT of the zero mean input as described by Equation (2.31), yields an output spectrum whose exponential decreasing shape is of interest. Recall that we also have to preserve the single sinusoid spectrum (spike) as best we can. To this end, a sixth order least-square polynomial approximation is adopted as an estimator and has a solution of the form[24]

$$N_{est}(f) = a_0 + a_1 f + \ldots + a_k f^k, \tag{2.36}$$

where k is the approximation order, and used in the following derivations. The residual of Equation (2.36) is given by

$$R^2 = \sum_{i=1}^{N} [N_{(i)c} - (a_0 + a_1 f_i + \ldots + a_k f_i^k)]^2, \tag{2.37}$$

where $N_{(i)c}$ is the i^{th} DFT noise coefficient. To minimize the residual, its partial derivatives with respect to each of the approximation coefficients are obtained and set to zero. This yields a Vandermonde matrix relating these coefficients to the actual noise values. These partial derivatives are expressed as

$$\begin{aligned}
\frac{\partial(R^2)}{\partial a_0} &= -2\sum_{i=1}^{N}[N_{(i)c} - (a_0 + a_1 f_i + \ldots + a_k f_i^k)] = 0 \\
\frac{\partial(R^2)}{\partial a_1} &= -2\sum_{i=1}^{N}[N_{(i)c} - (a_0 + a_1 f_i + \ldots + a_k f_i^k)]f = 0 \\
&\vdots \\
\frac{\partial(R^2)}{\partial a_k} &= -2\sum_{i=1}^{N}[N_{(i)c} - (a_0 + a_1 f_i + \ldots + a_k f_i^k)]f^k = 0.
\end{aligned} \tag{2.38}$$

taking the derivatives in Equation (2.38) and distributing the summations leads to

$$\begin{aligned}
&a_0 n + a_1 \sum_{i=1}^{N} f_i + \ldots + a_k \sum_{i=1}^{N} f_i^k = \sum_{i=1}^{N} N_{(i)c} \\
&a_0 \sum_{i=1}^{N} f_i + a_1 \sum_{i=1}^{N} f_i^2 + \ldots + a_k \sum_{i=1}^{N} f_i^{k+1} = \sum_{i=1}^{N} f_i N_{(i)c} \\
&a_0 \sum_{i=1}^{N} f_i^k + a_1 \sum_{i=1}^{N} f_i^{k+1} + \ldots + a_k \sum_{i=1}^{N} f_i^{2k} = \sum_{i=1}^{N} f_i^k N_{(i)c}.
\end{aligned} \tag{2.39}$$

Equation (2.39) can be written in matrix form as

$$\begin{bmatrix} n & \sum_{i=1}^{N} f_i & \cdots & \sum_{i=1}^{N} f_i^k \\ \sum_{i=1}^{N} f_i & \sum_{i=1}^{N} f_i^2 & \cdots & \sum_{i=1}^{N} f_i^{k+1} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} f_i^k & \sum_{i=1}^{N} f_i^{k+1} & \cdots & \sum_{i=1}^{N} f_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} N_{(c)i} \\ \sum_{i=1}^{N} N_{(c)i} f_i \\ \vdots \\ \sum_{i=1}^{N} N_{(c)i} f_i^k \end{bmatrix} \tag{2.40}$$

Simplifying the notation, we can obtain the matrix for a least squares fit by writing

$$\begin{bmatrix} 1 & f_1 & \cdots & f_1^k \\ 1 & f_2 & \cdots & f_2^k \\ \vdots & \vdots & \ddots & \vdots \\ 1 & f_n & \cdots & f_n^k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} N_{(c)1} \\ N_{(c)2} \\ \vdots \\ N_{(c)n} \end{bmatrix}. \tag{2.41}$$

Using matrix notation to represent Equation (2.41), the equation for a polynomial fit is given by

$$\mathbf{y} = X\mathbf{a}. \tag{2.42}$$

The solution results by premultiplying both sides by the transpose of X

$$X^T\mathbf{y} = X^T X\mathbf{a}, \tag{2.43}$$

to yield $\mathbf{a}$, as

$$\mathbf{a} = (X^T X)^{-1} X^T\mathbf{y}. \tag{2.44}$$

A sixth order least squares polynomial approximation was heuristically chosen to provide a close fit to the noise curve while minimizing the order to not follow the sinusoidal spike that is being swept from 0-10 kHz. Once an approximation of the colored noise is achieved, the reciprocal is taken and normalized to have maximum gain of one. This is mathematically expressed as

$$\begin{aligned} \hat{H}_w(f) &= \frac{H_w(f)}{Max\{H_w(f)\}} \\ H_w(f) &= N_{est}(f)^{-1}, \end{aligned} \tag{2.45}$$

where $\hat{H}_w(f)$ is the normalized fitler. Finally, the filter is applied to the input signal producing a sinusoid in white noise of slightly decreased amplitude as can be seen in Figure 2.6.

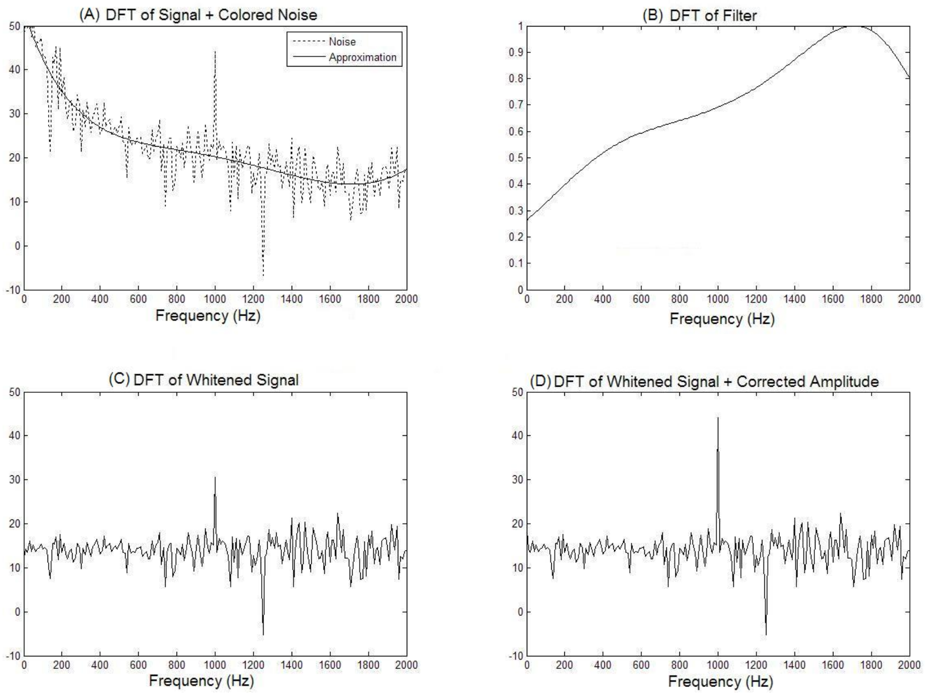


Figure 2.7: A 2 kHz sinusoid in colored pink noise (A), prewhitening filter (B), whitened sinusoid (C), and whitened sinusoid with corrected amplitude (D).

2.2.6 Amplitude Correction

As can be seen in Figure 2.6, the prewhitening filter attenuates the amplitude of the signal being filtered. For pink noise, the algorithm generates a high-pass prewhitening filter that has very gentle slope in what amounts to the transition band. If the signal of interest falls in this area, its amplitude will be greatly decreased and will not be accurately reflected in the output of the MUSIC analysis. To compensate for this, a correction coefficient is introduced to rescale the amplitude of the sinusoid being observed.

Figure 2.5 depicts how the correction coefficient is generated. The magnitude of the sinusoid in the original DFT is divided by the magnitude of the sinusoid in the whitened DFT. This ratio is then multiplied by the magnitude of the sinusoid in the whitened spectrum to return it to the original value. If the sinusoid being investigated is not greater than the noise floor, the prewhitening assumes its magnitude is that of the noise floor, and still produces a sinusoid in white noise of that amplitude and at that frequency.

2.3 Multiple Signal Classification (MUSIC) Algorithm

The MUSIC algorithm was initially developed by Schmidt[25], Bienvenu, and Kopp[26] to detect emitter location by direction of arrival (DOA) in sensor arrays. Schmidt proved that the MUSIC algorithm was asymptotically unbiased in estimating 1) the number of incident wavefronts; 2) DOA; 3) strength and cross correlation of incident waveforms; 4) noise/interference ratio. As a bonus, the MUSIC algorithm is also useful as a frequency estimator on time series data[25]. Furthermore, it has the ability to provide an unbiased estimate of the null-spectrum of uncorrelated data irregardless of the SNR[27], thus making it very tolerant to noisy applications.

The question might arise as to why the MUSIC algorithm is being chosen over other spectral methods such as the DFT, maximum likelihood estimate (MLE), or minimum mean square error (MMSE). The DFT is the most well known for its ease of use, but also for its shortcomings. In this application, it is the necessity to capture enormous data sets to minimize spectral leakage and to lower the noise floor that disqualifies the DFT. Figure (2.7) depicts just how tolerant the MUSIC algorithm is to noise as it is capable of estimating the frequency when the signal seem to be totally corrupted. MMSE has been discussed

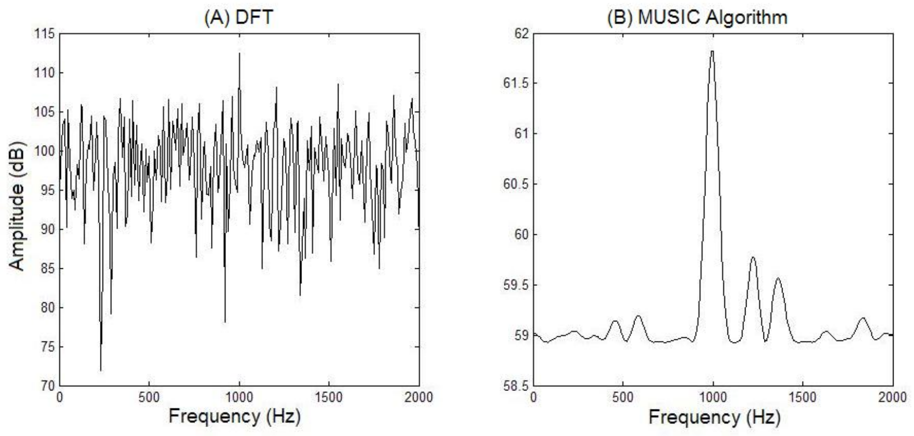


Figure 2.8: Frequency spectrum of 1 kHz signal in noise computed with Discrete Fourier Transform (A) and MUSIC algorithm (B).

in the literature for frequency estimation, but closed form solutions are difficult and lead to computationally complex algorithms[28]. As a result, we opt for the MUSIC algorithm, which was stated by Schmidt[25] to asymptotically approach the Cramer-Rao accuracy bound (CRB) and proven by Stoica[29] to outperform MLE.

To understand why the MUSIC algorithm was chosen, it is helpful to begin with the data model[29]

$$\begin{aligned}\mathbf{y}(\mathbf{t}) &= \mathbf{A}(\theta)\mathbf{x}(\mathbf{t}) + \mathbf{e}(\mathbf{t}) \\ \mathbf{A}(\theta) &= [\mathbf{a}(\omega_1), \mathbf{a}(\omega_2), \ldots, \mathbf{a}(\omega_n)]. \end{aligned} \tag{2.46}$$

In Equation (2.46), $\mathbf{y}(\mathbf{t}) \in \mathbf{C}^{m\times 1}$ is the noisy data vector, $\mathbf{x}(\mathbf{t}) \in \mathbf{C}^{n\times 1}$ is the vector of signal amplitudes, $\mathbf{e}(\mathbf{t}) \in \mathbf{C}^{m\times 1}$ is the additive noise, and $\mathbf{A}(\theta) \in \mathbf{C}^{m\times n}$ is a Vandermonde matrix of signal frequencies where $\theta_{\mathbf{n}} = [\omega_n^0 \ldots \omega_n^{m-1}]^T$. In order to use the model presented in Equation (2.46), some basic assumptions must be made. Table (2.1) lists the assumptions that are used for both MUSIC and MLE algorithms. In order to estimate the frequencies

Table 2.1: Assumptions needed for MUSIC and MLE frequency estimator algorithms.

	Assumption	**MUSIC**	**MLE**
1	$m > n$, and the vectors $a(\omega)$ corresponding to $(n+1)$ different values of ω are linearly independent	Yes	Yes
2	$E\{e(t)\} = 0$, $E\{e(t)e^*(t)\} = \sigma I$, and $E\{e(t)e^T(t)\} = 0$	Yes	Yes
3	The matrix $P = E\{x(t)x^*(t)\}$ is nonsingular (positive definite), and $N > m$	Yes	No
4	$E\{e(t)e^*(s)\} = E\{e(t)e^T(s)\} = 0$ for $t \neq s$, and $e(t)$ is Gaussian distributed.	No	Yes

of N complex sine waves from single experiment data, we can use the model

$$y_k = \sum_{p=1}^{N} \gamma_p e^{j\omega_p k} + e_k \qquad k = 1, \ldots, n. \tag{2.47}$$

Equation (2.47) may be expressed as the data model of the MUSIC algorithm by using the following notation, and letting m be some integer greater than n such that

$$\begin{aligned}\mathbf{y}(\mathbf{t}) &= [y_t \ldots y_m]^T \\ \mathbf{a}(\omega_\mathbf{t}) &= [1 e^{j\omega_t} \ldots e^{j(m-1)\omega_t}]^T \\ \mathbf{x}(\mathbf{t}) &= [\gamma_1 e^{j\omega_1 t} \ldots \gamma_N e^{j\omega_N t}]^T \\ \mathbf{e}(\mathbf{t}) &= [e_t \ldots e_m]^T \qquad t = 1, \ldots, n. \end{aligned} \tag{2.48}$$

In the single experiment case, $\mathbf{e}(\mathbf{t})$ and $\mathbf{e}(\mathbf{s})$ are correlated for $t \neq s$, contradicting assumption 4 and making the MLE algorithm not applicable. Assuming that MLE could be used to estimate the frequencies, Stoica shows that MUSIC still reaches the CRB faster than MLE for data sets of length n, with m sensors. Looking at the ratios of MUSIC and MLE to the CRB,

$$var_{ML}(\hat{\omega})/var_{CR}(\hat{\omega}) = 1 + \frac{1}{m * SNR} \tag{2.49}$$

$$var_{MU}(\hat{\omega})/var_{CR}(\hat{\omega}) = 1 + \frac{(A^*A)^{-1}}{SNR}, \tag{2.50}$$

we see that each is dependent on the signal to noise ratio and m. For MLE to reach the CRB, both m and n must approach infinity, which is clearly unattainable. However, for MUSIC to reach the CRB, m must only be greater than n for relatively large n. It is known that MUSIC is inefficient for correlated signals, but in this case of single observations, this drawback does not present a problem.

Therefore, we are left with the MUSIC algorithm, where assumptions 1,2, and 3 are satisfied by the model above as long as $2N < n + 1$. This final constraint simply states that the number of complex sinusoids one is trying to analyse can not exceed the number of measurement samples, which is easily achieved.

Finally, we proceed with analyzing the eigen-decomposition of the data covariance matrix, which is the essence of the MUSIC algorithm. Assuming that conditions 1,2, and 3 are met, the covariance matrix of observation vector $\mathbf{y}(\mathbf{t})$ is given by

$$\mathbf{R} = E\{\mathbf{y}(\mathbf{t})\mathbf{y}^*(\mathbf{t})\} = \mathbf{A}(\theta)\mathbf{P}\mathbf{A}^*(\theta) + \sigma^2\mathbf{I}, \tag{2.51}$$

where $\mathbf{P} = E\{\mathbf{x}(\mathbf{t})\mathbf{x}^*(\mathbf{t})\}$ is the autocorrelation of the measured data, $\mathbf{I}$ is an $[m \times m]$ identity matrix, and σ^2 is the noise variance. Taking the eigen decomposition of $\mathbf{R}$, we obtain m eigenvalues, D, and an $[m \times m]$ matrix, V, of eigenvectors which span the column-space of A. Taking the eigenvectors corresponding to the first n largest eigenvalues, we can define a matrix, $\mathbf{E_S}$, that spans the signal subspace[30]. The remaining $N = m - n$ eigenvectors span the noise subspace and are defined by the $[m \times N]$ matrix $\mathbf{E_N}$[25]. Due to the orthogonality of the frequency vectors $\mathbf{a}(\omega)$ in $\mathbf{A}(\theta)$ and the noise subspace, we can compute the spectrum as the inverse of the inner product given by

$$\mathbf{J}(\omega) = \frac{\mathbf{1}}{\mathbf{a}^*(\omega)\mathbf{E_N}\mathbf{E_N^*}\mathbf{a}(\omega)} \tag{2.52}$$

2.4 Summary

We now dispose of the necessary tools to analyze the time domain data captured in our experimental problem. We began with a data model that employs a perturbation analysis to model a lossy, nonlinear plane wave propagating in the positive x direction. This model has been shown to outperform the classical Bessel-Fubini approximation for $n \geq 3$ order and for distances beyond $x = x_D$. This model has been shown to work in simulation, but anechoic chamber data should be analyzed to confirm these solutions. However, the ERL currently lacks equipment at this time capable of generating high amplitude ultrasound at the correct frequency to generate this phenomenon. Second, we must preprocess the data to remove $1/f$ "pink" noise introduced by the measurement equipment, and average the data so that it has zero mean. Finally, it has been shown that the MUSIC algorithm is optimal in the sense that it efficiently approaches the CRB and is the only spectral analysis tool that can effectively process this low SNR data.

Chapter 3

Experimental Setup

3.1 Anechoic Chamber

It is interesting to note, that using the aforementioned nonlinear detection technique requires measurements, which in turn require that a suitable environment be constructed. Due to the inefficient demodulation from two ultrasonic frequencies to one baseband tone, and the fact that this difference frequency lies right in the middle of the noisy sonic spectrum, the Electronics Research Laboratory (ERL) opted to construct an anechoic chamber. Given our rather wider interest in acoustic, electromagnetic, and acoustically modulated electromagnetic phenomenon, building a dual purpose chamber was concluded as the best option. The purpose of this chamber is to 1) attenuate acoustic reflections to better emulate free space; 2) attenuate electromagnetic reflections to better emulate free space; 3) attenuate both acoustic and electromagnetic transmissions for safety reasons. This final objective is necessary due to the high input power levels necessary to generate measurable nonlinear behavior. For example, it takes 120 dB sound pressure level (SPL) of ultrasonic energy to produce 66 dB of audible sound at 1 m. Typically, acoustic energy at this amplitude causes instant permanent hearing damage. While most scientists believe that high amplitude ultrasound is harmless to humans, there is little research confirming its effects in confined spaces or with directed sources. Furthermore, acoustic transmission attenuation is a byproduct of the anechoic chamber's RF shielding, which is necessary due to the well known dangers of high energy microwaves.

Since ERL is interested in measuring life-sized targets, the anechoic chamber was constructed as large as practically possible. Overall external chamber dimensions are 96 inches in width, 72 inches in height, and 144 inches in length. Due to the thickness of absorbent materials used in construction, the internal usable dimensions are 76 inches in width, 52 inches in height, and 120 inches in length. Chamber construction began by building a raised floor to provide wire runs to various test equipment located within the chamber. The floor measures 109 inches in width by 156 inches in length and rests on top of 9”×4” posts. Starting from the lower most layer, the floor is comprised of one layer 3/4” plywood, one layer cement board, one layer copper mesh, and two layers of 6.0 mm thick Acoustiblok. The walls and ceiling of the anechoic chamber are all constructed using the same process. Support for the walls is by way of an extruded aluminum space frame manufactured by 80/20 Corporation. The outermost layer of each wall and ceiling panel is comprised of copper mesh, manufactured by TWP Inc.[31], forming a Faraday cage around the entire chamber. An exploded view of the anechoic chamber can be seen in Figure 3.1.

Forming the foundation of each wall and ceiling panel is a layer of 3.0 mm thick Acoustiblok. This high-density rubberized material provides almost 2/3 of the through-wall attenuation above 1.0 kHz[32]. The Acoustiblok sandwiches the copper mesh against the supporting frame and is held in place with 1.5 in. nylon bolts spaced 4 in. apart. At every seam in the Acoustiblok, acoustical sealant and tape is used to further improve soundproofing. Attached to the inside surface of the Acoustiblok are 2 ft. by 4 ft. panels of QuietBoard glued to Melamine foam, both manufactured by American Micro Industries. Each panel is attached by six nylon bolts, and forms the surface to which the RF absorbing foam tiles are glued. Acoustiblok sealant is again used at all panel joints to ensure soundproofing. The innermost layer consists of RF absorbing geometric tiles. Two types of this tile, each measuring 2 ft. square, were used. The pyramidal style tile was used in the most sensitive areas, such as along the back wall where the most intense energy will accumulate. Eggshell tiles were used to fill in the remaining spaces. All of the tiles were arranged in a manner that reduced the possibility of generating standing waves under continuous signal generation.

Acoustic performance of the ERL anechoic chamber is based on transmission and reflection loss given by[33]

$$L_t = 20\log_{10}\left(\frac{P_t}{P_i}\right) \tag{3.1}$$

$$L_r = 20\log_{10}\left(\frac{P_r}{P_i}\right), \tag{3.2}$$

where P_t, P_r, and P_i represents the acoustic pressure of the transmitted, reflected, and

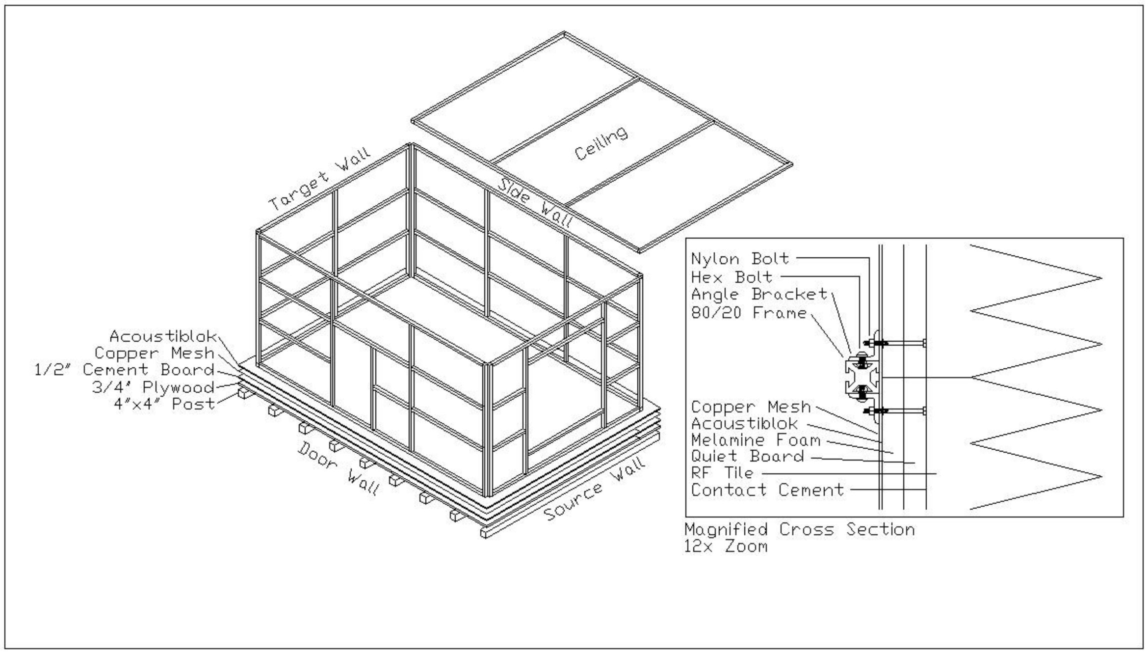


Figure 3.1: Exploded view of chamber frame and floor. Magnified cross section shows wall construction detail. The Acoustiblok and copper mesh are held in place with 0.25 x 1.5 in. nylon flat head bolts and nuts attaching these layers to each angle bracket. The Melamine/Quiet Board panels are held in place with 0.25 x 4 in. nylon carriage bolts. Finally, RF tiles are glued to the Quiet Board using contact cement.

incident waves respectively. These parameters are equivalent to those obtained from a S-parameter matrix of a two port device. For these equations, a factor of 20 is used since pressure is the electrical equivalent of voltage, and power is voltage squared. Figure 3.2 depicts the acoustic performance of the ERL anechoic chamber. RF characterization was also performed, but is not included in this document since it does not pertain to acoustics.

3.2 Test Setup

Two excitation signals, of frequency f_1 and f_2 are produced using two Marconi 2024 signal generators, and are amplified/transmitted using Audio Spotlight parametric arrays from Holosonic as depicted in Figure 3.3. For these experiments, the reflected wave microphone and parametric arrays are located approximately 2.0 meters from the target.

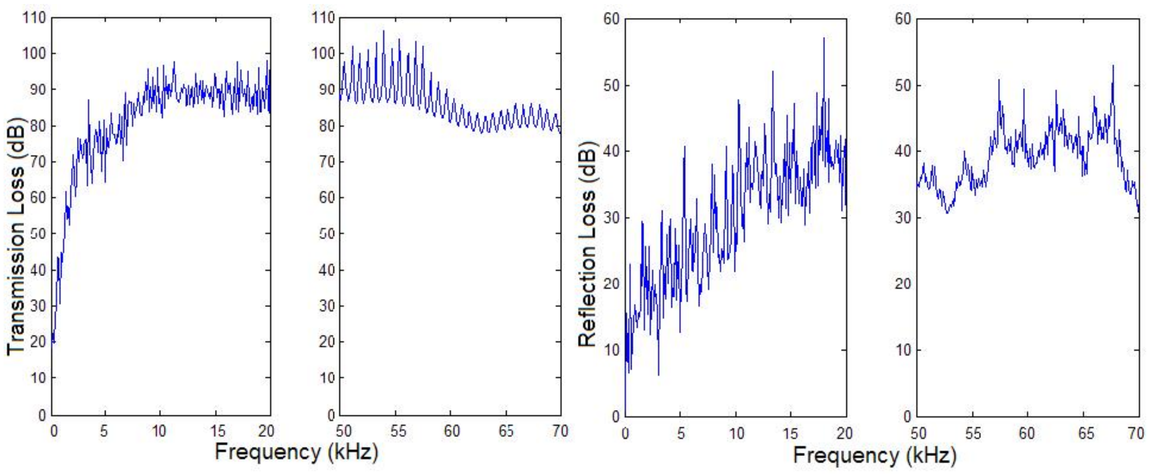


Figure 3.2: Acoustic transmission and reflection loss of back wall of anechoic chamber. Above 50 kHz where higher power signals can be generated, the periodicity of the attenuation indicates that a resonance is being generated.

The stationary tone f_1 is kept at 55 kHz while f_2 is swept from 55-65 kHz in 100 Hz increments.

Measurements of the incident and reflected sound pressure amplitudes are recorded using two 377B01 condenser microphones from Piezotronics, and digitized by a PXI-5922 High Speed Digitizer. One microphone is used to record the incident wave, while the other measures the reflected wave. A sample rate of 50 kHz is used and 50,000 time samples are captured for each frequency increment. A summary of the test equipment specifications and part numbers may be found in Appendix H.

Targets chosen for this experiment include glass, metal, wood, and fiberglass containers of both regular and irregular shapes as shown in Figure 3.4. Targets of regular shape were perfect 10-inch cubes. Both the glass and wooden cubes have sidewall thickness of approximately 1/4-inch, while the metal cube is constructed of 20-gauge steel. An irregular shaped target, emulating a piece of stone allowed us to test a more realistic target. It is constructed of molded fiberglass measuring 1/16-inch thick, with the solid variant filled with concrete in lieu of sand stone at 2.3 g/cm^3.

To single out the one phenomenon we are trying to capture, solid and hollow versions of the same target were placed in identical positions on the test stand. Usually, a hollow target can be filled with some material (sand in these experiments) without moving

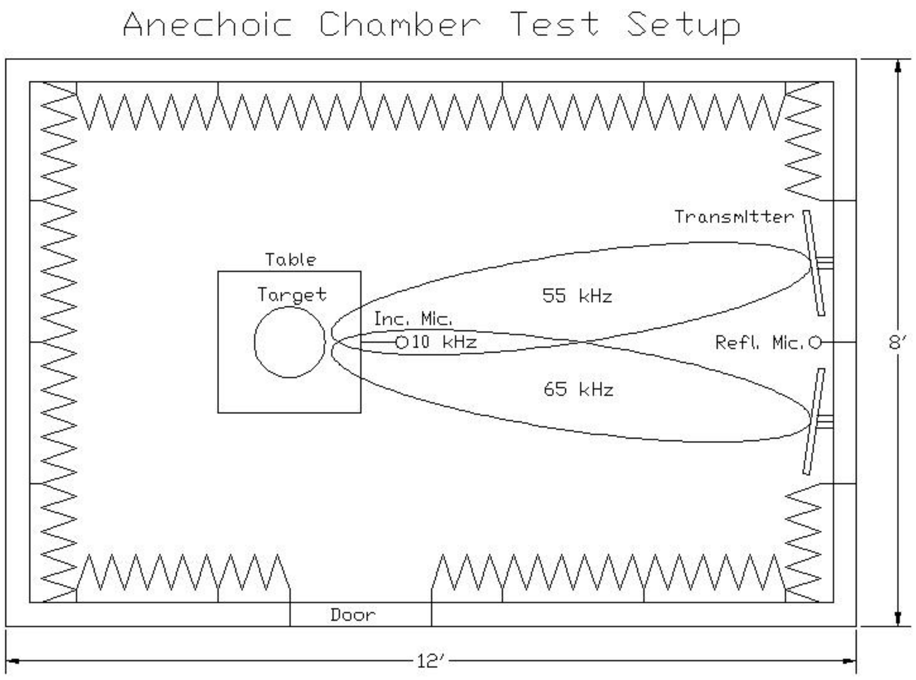


Figure 3.3: Top view of anechoic chamber showing two Audio Spotlight transmitters and two condenser microphones. The incident microphone is only used to monitor incident sound pressure levels and is not necessary for detection.

it on the test stand. This ensures that it is the target's composition that produces a given acoustic signature, and not a slight change in its alignment.

At the surface of the test object, the two tones mix to produce inter-modulation products. The transmitted signals are shown in Figure 3.5 and may be modeled using[34]

$$X^1(t) = |X_1|\cos(\omega_1 t + \phi_1) + |X_2|\cos(\omega_2 t + \phi_2), \tag{3.3}$$

to yield second order inter-modulation products of

$$\begin{aligned} X^2(t) = \left(\frac{1}{2}\right)^2 & [X_1^2 ej2w_1t + 2X_1X_1^* \\ & +2X_1X_2e^{j(w_1+w_2)t} + 2X_1X_2^*e^{j(w_1-w_2)t} \\ & +(X_1^*)^2e^{-j2w_1t} + 2X_1^*X_2e^{j(w_2-w_1)t} \\ & +2X_1^*X_2^*e^{-j(w_1+w_2)t} + X_2^2e^{j2w_2t} \\ & +2X_2X_2^* + (X_2^*)^2e^{-j2w_2t}], \end{aligned} \tag{3.4}$$

where X_n is the amplitude of the original cosine at f_1 and f_2, producing new frequencies

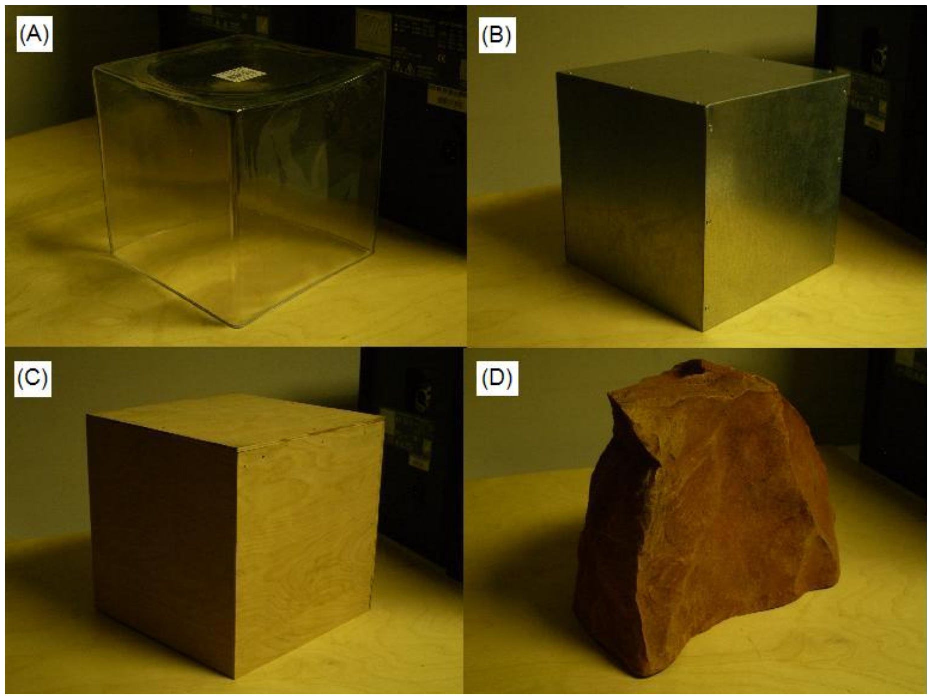


Figure 3.4: Targets used include 10-inch cubes of glass (A), metal (B), wood (C), and simulated sandstone (D).

at $f_2 - f_1$ and $f_2 + f_1$ as well as third order inter-modulation tones and higher frequency tones which are not of interest. The base band tone at frequency $f_2 - f_1$ is swept from 0.1-10 kHz to provide a frequency response or acoustic signature of the target. This difference frequency may also couple into the target object where it can potentially excite a resonance.

3.3 Measurements and Signal Processing

Upon appending the time domain measurements at each frequency increment to the same LabView data file, we proceed with the signal processing techniques discussed in Chapter 2 to analyze the data. Given the large data sets, and the fact that at this stage of research, we are monitoring four channels of data, a Matlab graphical user interface (GUI) environment was developed to aid in the signal processing. This program allows the user to easily monitor all four signals, and to change the parameters used to implement the prewhitening filter and MUSIC algorithm. It also provides real time graphical algorithm

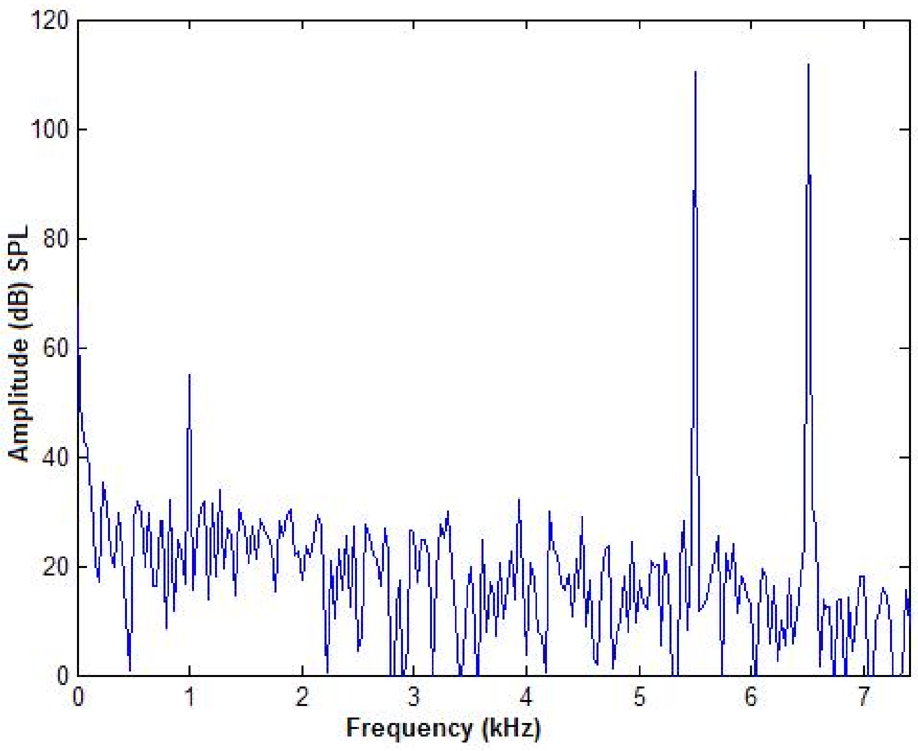
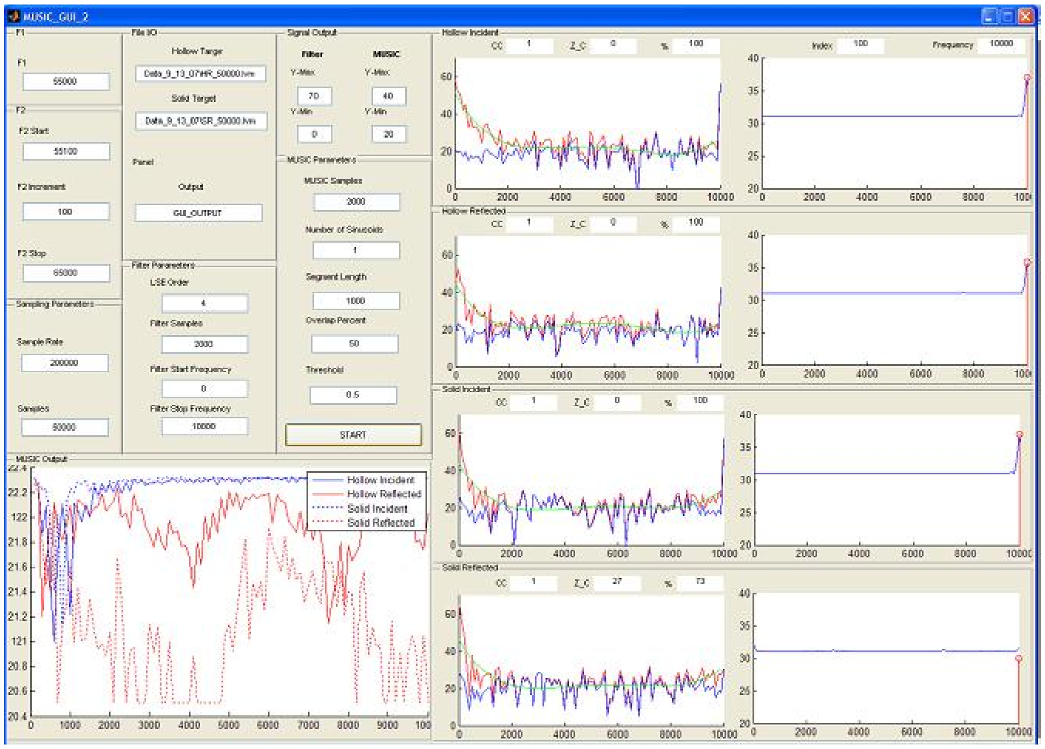


Figure 3.5: Spectrum showing excitation signals $f_1 = 55$ kHz, $f_2 = 65$ kHz, and difference frequency $f_2 - f_1 = 10$ kHz generated in air.

feedback so the user may evaluate the noise model and terminate the program as needed, e.g., a parameter needs to be changed. Figure 3.6 shows a screenshot of the "Music_Gui.m" program and Tables 3.1 and 3.2 define the inputs and outputs respectively. The code used to implement "Music_Gui.m" is available in Appendices D-G.

Upon entering all of the correct parameters and input files, a user may begin to analyze the data by simply initiating the "start" button. The four clusters of two graphs show the noise model used to implement the prewhitening filter and the MUSIC coefficients, $\mathbf{J}(\omega)$, for hollow-incident, hollow-reflected, solid-incident, and solid-reflected signals respectively. The prewhitening filter graphs show the noisy data in red, the noise approximation in green, and the filtered data in blue. The MUSIC coefficients graph shows the total MUSIC pseudospectrum, $\mathbf{J}(\omega)$, in blue and the coefficients, $\widehat{J}(\omega_n)$, chosen by the algorithm in red. Ideally, the coefficients shown as a red stem plot should match the peak in the MUSIC pseudospectrum. However, if the algorithm can not find a peak in the pseudospectrum at the frequency being examined, the coefficient is replaced by the noise


MUSIC_GUI_2

F1
F1: 55000

F2
F2 Start: 55100
F2 Increment: 100
F2 Stop: 65000

Sampling Parameters
Sample Rate: 200000
Samples: 50000

File I/O
Hollow Targer: Data_9_13_07HR_50000.lvm
Solid Target: Data_9_13_07SR_50000.lvm
Panel
Output: GUI_OUTPUT

Filter Parameters
LSE Order: 4
Filter Samples: 2000
Filter Start Frequency: 0
Filter Stop Frequency: 10000

Signal Output
Filter Y-Max: 70
Filter Y-Min: 0
MUSIC Y-Max: 40
MUSIC Y-Min: 20

MUSIC Parameters
MUSIC Samples: 2000
Number of Sinusoids: 1
Segment Length: 1000
Overlap Percent: 50
Threshold: 0.5

Hollow Incident: CC 1, Z_C 0, % 100, Index 100, Frequency 10000
Hollow Reflected: CC 1, Z_C 0, % 100
Solid Incident: CC 1, Z_C 0, % 100
Solid Reflected: CC 1, Z_C 27, % 73

Figure 3.6: "Music_GUI.m" signal processing application.

floor of the pseudospectrum, which is approximately 30. When this occurs, the zeroed coefficients counter is incremented to also update the percentage of valid coefficients. This may be seen in the last MUSIC graph of Figure 3.6, where the MUSIC coefficient does not match the pseudospectrum. Finally, the individual $\widehat{J}(\omega_n)$ are then combined to generate a real time frequency response plot, $\mathbf{S}(\mathbf{f})$, located below the input fields.

The program "MUSIC_GUI.m" described above produces an acoustic signature of both the hollow and solid variants of the same target. By inspection, it may be seen whether these targets respond differently if they are hollow or solid, thus showing that a detection algorithm could be implemented. While the scope of this thesis is not to derive a detection algorithm, the "MUSIC_GUI.m" program does produce an output file so that further signal processing may be pursued. This may include plotting the data using different scales, or taking the integral of the acoustic signature, $\mathbf{S}(\mathbf{f})$, as shown in Chapter 4, to determine the average value, which may be a good indicator of the targets composition.

Table 3.1: MUSIC GUI input parameter definitions.

Parameter	**Definition**
F1	Stationary frequency f_1.
F2 Start	f_2 frequency sweep start.
F2 Increment	f_2 frequency sweep increment.
F2 Stop	f_2 frequency sweep stop.
Sample Rate	Sample rate used by LabView Virtual Instrument (VI).
Samples	Samples captured per frequency increment by LabView VI, used to parse input data file.
Hollow Target	Filename of hollow target data file (must specify path name).
Solid Target	Filename of solid target data file (must specify path name).
Output	Output file name (saved to same location as MUSIC_GUI.m).
LSE Order	Least square estimate order for noise model.
Filter Samples	The number of samples used by DFT to approximate noise (must be less than Samples).
Filter Start Frequency	Start frequency of prewhitening filter (should be zero for pink noise).
Filter Stop Frequency	Stop frequency of prewhitening filter.
Y-Max, Y-Min	Plot parameters for viewing prewhitening filter and MUSIC output (X-Min, X-Max defined by range of frequency sweep.
Number of Sinusoids	The number of sinusoids the MUSIC algorithm is looking for.
Segment Length	The length of partitioned blocks.
Overlap Percent	The percentage of overlap between consecutive blocks.
Threshold	Specifies the cutoff for the signal and noise subspace separation.

Table 3.2: MUSIC GUI output parameter definitions.

Parameter	**Definition**
Index	The current loop index.
Frequency	The current frequency increment.
CC	The current correction coefficient.
Z_C	The total number of zeroed MUSIC coefficients.
%	The percentage of valid MUSIC coefficients.

Chapter 4

Results and Conclusions

4.1 Results

Three out of the five targets tested in the course of this research effort show a significant difference between hollow and solid variants. These targets include a metal cube, glass cube, and simulated rock. The plastic container shown in Figure 4.5 shows some difference in solid versus hollow variants, while the wood cube shown in Figure 4.3 shows practically no difference. To verify the performance of the MUSIC algorithm, a constant amplitude frequency sweep is applied. The expected flat spectrum, shown in Figure 4.1, validates the algorithm's performance. Most of these acoustic signatures were generated with only 2,000 MUSIC samples, as opposed to the 50,000 samples that were captured and used to generate similar results using only the DFT. This shows that MUSIC analysis can produce equivalent acoustic spectra using significantly less data and in far less time. Table 4.1 summarizes these results and provides a metric for the difference between hollow and solid variants. This metric is given as a percentage and is computed by

$$difference = \frac{|S_{hr} - S_{sr}|}{S_i - S_{cutoff}} \times 100\%, \tag{4.1}$$

where S_{cutoff} is the noise floor of the MUSIC pseudospectrum. In Equation 4.1, S_{hr}, S_{sr}, and S_i are the frequency response average values of the hollow reflected, solid reflected, and incident frequency responses respectively. The average value of each response is computed by

$$S_{avg} = \frac{1}{f_{stop} - f_{start}} \int_{f_{start}}^{f_{stop}} S(f)df, \tag{4.2}$$

where $S(f)$ is the frequency response of the target.

Using the above metric, the metal cube displays the greatest difference between solid and hollow variants, as may be seen in Figure 4.2. Figure 4.2 shows that the empty metal cube reflects less acoustic energy. This is in contrast to what appears in the responses of the glass cube, simulated rock, and plastic containers, whose empty responses show that more acoustic energy is reflected. Material properties that can affect the amount of reflected acoustic energy include elasticity, acoustic impedance, and homogeneity of the target. The difference between a metal cube and other targets is most likely based on their structural make. The glass cube, simulated rock, and plastic container are all molded from one continuous, homogenous material. This means that acoustic energy incident on these hollow targets gets trapped in the shell and can not pass into the hollow interior due to the large impedance mismatch between solids and air. On solid targets of this type, the mechanical interaction between the shell and interior medium allows acoustic energy to propagate into the interior where it is dissipated. The wood and metal cubes are constructed from individual pieces held together with mechanical fasteners. This may prevent trapped absorbed acoustic energy in the shell of a hollow target from traveling as easily around the perimeter. Furthermore, the metal cube has significantly greater acoustic impedance as compared to the other targets while still being compliant. This is in contrast to the glass cube and simulated rock which have low impedance and high rigidity. The material properties are also summarized in Table 4.1.

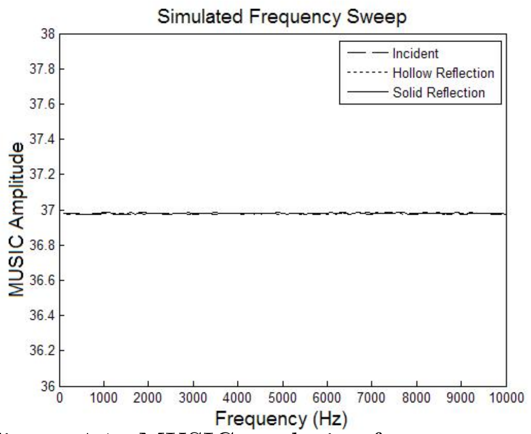


Figure 4.1: MUSIC analysis of constant amplitude frequency sweep to test algorithm.

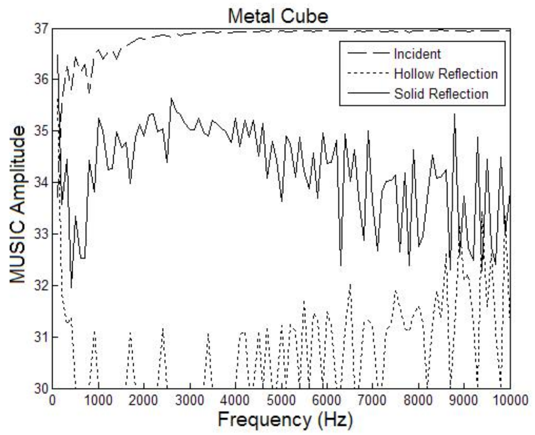


Figure 4.2: Metal cube shows the greatest difference in hollow and solid variants.

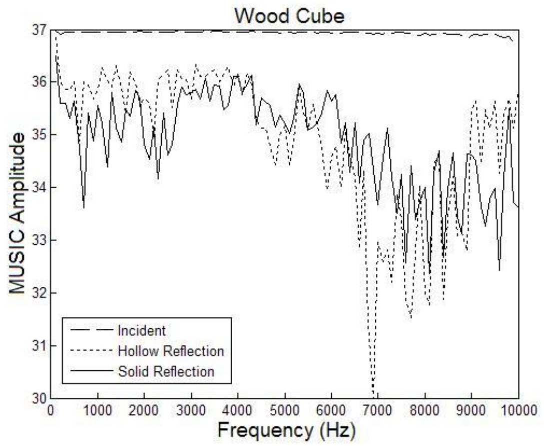


Figure 4.3: Wood cube shows no difference in hollow and solid variants.

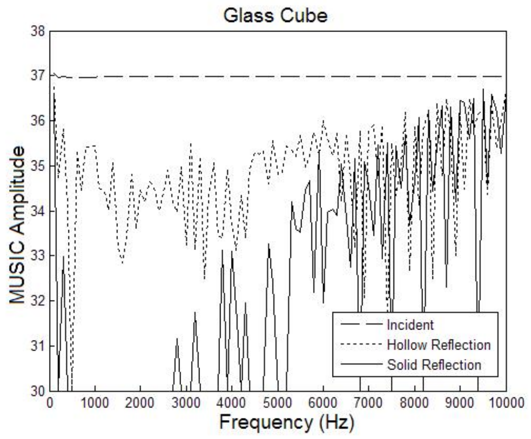


Figure 4.4: Glass cube shows difference in solid and hollow variants, but required more samples to do so.

One important observation to make about Figure 4.6 is the existence of a resonant behavior in a hollow target. These are evident in the equally spaced dips in the reflected sound energy. In Figure 4.6, these dips fall at intervals of 3.5 kHz. Knowing the speed of sound in the fiberglass shell to be 2,650 m/s, one can calculate the resonant length to be

$$L = \frac{v_s}{2f_r}. \tag{4.3}$$

Here v_s is the sound speed in the material, and f_r is the resonant frequency. For the hollow rock, Equation (4.3) yields a resonant length of 0.37 m, which is very close to the actual rock size of 0.40 m. If such a resonant response can be found in all targets, it will be possible to estimate not only the composition of a target, but also its size.

4.2 Future Work

While the results for this research are thus far promising, there is still much work to be done in the way of improved sensing, increased SNR, and expanded functionality. First of all, the distinction should be made between mechanical and acoustic resonance. Mechanical resonance is the movement of the target's surface due to some applied harmonic force and is given by Equation (4.3). Mechanical resonance is difficult to measure with microphones due to the poor coupling between solids and air, and could be more easily measured with equipment such as a Laser Doppler Vibrometer (LDV). Acoustic resonance

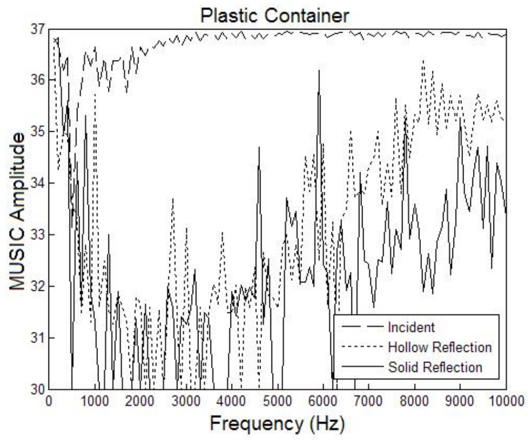


Figure 4.5: Plastic container shows some difference in hollow and solid variants.

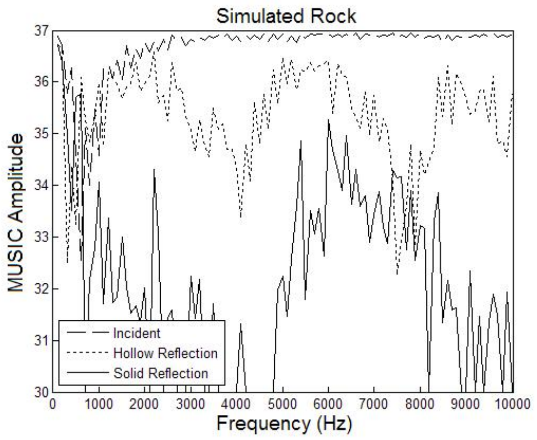


Figure 4.6: Simulated rock shows significant difference in hollow and solid variants and possible resonant response.

is the movement of air molecules within a target, which if modeled as a Helmholtz resonator has a resonant frequency given by[35]

$$f_r = \frac{c}{2\pi}\sqrt{\frac{S}{Vl}}. \tag{4.4}$$

In Equation (4.4), c, S, l, and V are the sound velocity, cross sectional area of the neck, length of the neck, and volume of the cavity, respectively. Since the resonant frequency is inversely proportional to the volume, acoustic resonance can give more precise indication of the amount of substance in a target in addition to it being hollow or solid.

Another improvement that can be made is to increase the transmit power of the ultrasonic sources. The amplifiers and transducers from Holosonic used in this research have a limit of 13.6 Vpp input. This limit is primarily due to the large operating bandwidth of 10 kHz. If smaller bandwidth transducers are used, such as those from Airmar, input signals as high as 1000 Vpp can be used. These transducers have a much narrower bandwidth of approximately four percent of the center frequency, and must be operated at low duty cycles. Furthermore, the use of pulsed waveforms such as radar can increase the relative instantaneous power applied to the target. Linear frequency modulated (LFM) chirps having sufficiently wide bandwidth in the frequency domain, simulate a delta function in the time domain providing a more realistic impulse response of a target.

In addition, using an LFM chirp as in acoustic radar can provide range data as well as target composition. Using the well known method of stretch processing (pulse

Table 4.1: Summary of results for hollow targets (HT) and solid targets (ST).

Target	**MUSIC Samples**	**Elasticity (GPa)**	**Acoustic Impedance (MRayl)**	**Avg. HT Amp.**	**Avg. ST Amp.**	**% Diff.**
Simulation	2000	N/A	N/A	36.61	36.61	N/A
Metal Cube	2000	190-210	47	30.46	33.89	53.37
Simulated Rock	2000	45	2.85	34.98	31.86	49.68
Glass Cube	9000	65-90	12.5	34.40	32.02	35.87
Plastic Container	2000	0.8	2.33	32.78	31.90	13.87
Wood Cube	9000	8-13	1.5-3.0	34.53	34.54	0.24

compression), a high resolution range spectrum can be produced by comparing the reflected LFM chirp to the transmitted one. In stretch processing, the transmitted signal is modeled as[36]

$$s_t(t) = \cos\left(2\pi\left(f_0 t + \frac{\mu}{2}t^2\right)\right), \tag{4.5}$$

where f_0 is the LFM start frequency, and $\mu = B/\tau^{'}$ is the LFM bandwidth divided by the pulse duration. If we introduce a time delay $\Delta\tau = 2R/c$ and attenuation coefficient a for a radar cross section (RCS), antenna gain, and range attenuation, the received signal is given by

$$s_r(t) = a\cos\left[2\pi\left(f_0(t-\Delta\tau) + \frac{\mu}{2}(t-\Delta\tau)^2\right)\right]. \tag{4.6}$$

Mixing of the transmitted and received waveforms produces a signal with an instantaneous frequency that is dependent on the range(s) of the target. Assuming a peak in the frequency response at f_1, the range of the target is computed by

$$R_1 = \frac{f_1 c\tau^{'}}{2B}. \tag{4.7}$$

Using the specifications of the LabView equipment already available in the lab, a simulation of a stretch processor implemented in Matlab shows an achievable range resolution of 5 cm as shown in Figure 4.7.

In summary, the use of chirped signals can increase the effective power incident on the target, while providing range data. With the addition of a LDV, mechanical resonance can be more easily measured, leaving the standoff microphones to focus on measuring acoustic resonance, which provides greater volume resolution.

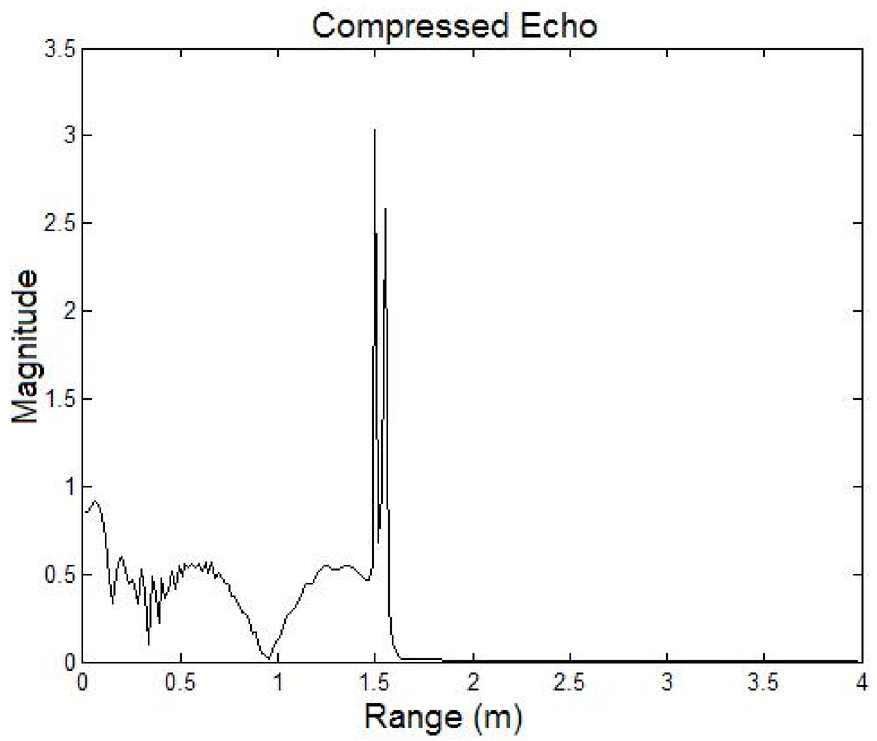


Figure 4.7: DFT of pulsed compressed received signal showing resolvable targets at 1.5 and 1.55 m.

4.3 Conclusion

A novel approach using nonlinear ultrasonic parametric arrays for standoff analysis of targets has been presented. It has been shown that this approach is capable of distinguishing between solid and hollow variants of the same target, with the potential to gain even more information about the target with minimal additional complexity. Furthermore, the third order nonlinear air model presented in Chapter 2, performs better than conventional models in predicting third order harmonic growth in simulation. This model will help to better predict nonlinear ultrasonic propagation so that estimates of incident energy on targets at standoff ranges will be more accurate. The combination of this work, with that of finite element models investigated by Vetreno, may allow faster, higher dimensional models to be generated[37].

Furthermore, using the MUSIC algorithm for spectral estimation, provides acoustic signatures using only 2,000 time samples as opposed to 50,000 for Fourier analysis. This results in a significant decrease in computation time while being more tolerant to low SNR.

As a final contribution, the “Air_Model_GUI.m” and “MUSIC_GUI.m” applications have been included to aid in future development and provide easier algorithm implementation.

Bibliography

[1] D. Donskoy. Detection and discrimination of nonmetallic land mines. In *SPIE Conference on Detection and Remediation Technologies for Mines and Minelike Targets IV*, pages 239–246, Orlando, FL, 1999.

[2] M. S. Korman and J. M. Sabatier. Nonlinear acoustic techniques for landmine detection. *Acoustical Society of America Journal*, 116:3354–3369, December 2004.

[3] David T. Blackstock. *Fundamentals of Physical Acoustics*. John Wiley and Sons, Inc., 2000.

[4] M. Yoneyama, J. I. Fujimoto, Y. Kawamo, and S. Sasabe. The audio spotlight: An application of nonlinear interaction of sound waves to a new type of loudspeaker design. *Acoustical Society of America Journal*, 73:1532–1536, May 1983.

[5] Rainer Stoessel. *Air-Coupled Ultrasound Inspection as a New Non-Destructive Testing Tool for Quality Assurance*. PhD thesis, University of Stuttgart, 2004.

[6] B. Bros and C. Bruschini. Sensor technologies for detection of antipersonnel mines. A survey of current research and system developments. In *Proceedings of the International Symposium on Measurement and Control in Robotics*, pages 211–217, 1996.

[7] D. Donskoy, A. Ekimov, N. Sedunov, and M. Tsionskiy. Nonlinear seismo-acoustic land mine detection and discrimination. *Acoustical Society of America Journal*, 111:2705–2714, June 2002.

[8] John P. Fish. *Sound Underwater Images: A Guide to the Generation and Interpretation of Side Scan Sonar*. Lower Cape Publishing Company, 1990.

[9] Michael Moser. *Engineering Acoustics, an introduction to noise control.* Springer-Verlag, Heidelberg Berlin, 2004.

[10] M. Greenspan. Rotational relaxation in nitrogen, oxygen, and air. *Acoustical Society of America Journal*, 31:155–160, 1959.

[11] A. D. Pierce. *Acoustics: An Introduction to Its Physical Principles and Applications.* McGraw-Hill, New York, 1981.

[12] P. J. Westervelt. Parametric Acoustic Array. *Acoustical Society of America Journal*, 35(4):535–537, April 1963.

[13] S. Sokoloff. Zur Frage der Fortpflanzung ultraakustischer Schwingungen in verschiedenen Koerpern. *Elektr. Nachr.-Technik*, 6:454–461, 1929.

[14] H. Dabirikhah and C. W. Turner. Novel airborne ultrasound transducer. *Acoustical Imaging*, 21:183–190, 1995.

[15] Aaron L. Walker. Behavioral modeling and characterization of nonlinear operation in RF and microwave systems. PhD dissertation, North Carolina State University, 2005.

[16] Xiao chen Xu et al. Theoretical calculation and experimental study on the third-order nonlinearity paramter C/A for organic liquids and biological fluids. *Acoustical Society of America Journal*, 113(3):1743–1748, 2003.

[17] Mingxi Deng. Characterization of surface properties of a solid plate using nonlinear lamb wave approach. *Ultrasonics*, 44:1157–1162, 2006.

[18] H. I. Ringermacher and R. S. Williams. Nonlinear ultrasonic characterization of oxygen impurities in titanium II. Technical report, Defense Technical Information Center OAI-PMH Repository [http://stinet.dtic.mil/oai/oai] (United States), 2002.

[19] Joseph B. Keller and Martin H. Millman. Perturbation theory of nonlinear electromagnetic wave propagation. *Phys. Rev.*, 181(5):1730–1747, May 1969.

[20] O. V. Rudenko and S. I. Soluyan. *Theoretical Foundations of Nonlinear Acoustics.* Consultants Bureau, New York, 1977.

[21] A. B. Coppens et. al. Parameter of nonlinearity in fluids II. *Acoustical Society of America Journal*, 38:797–804, 1965.

[22] S. L. J. Marple. *Digital spectral analysis with applications.* Englewood Cliffs, NJ, Prentice-Hall, Inc., 1987, 512 p., 1987.

[23] P. Dutta and P. M. Horn. Low-frequency fluctuations in solids: 1/f noise. *Review of Modern Physics*, 53(3):497–516, Jul 1981.

[24] John H. Mathews and Kurtis D. Fink. *Numerical Methods Using MATLAB.* Simon & Schuster, 1998.

[25] R. O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34:276–280, March 1986.

[26] G. Bienvenu and L. Kopp. Adaptivity to background noise spatial coherence for high resolution passive methods. *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '80.*, 5:307–310, Apr 1980.

[27] M. Kaveh and A. Barabell. The statistical performance of the MUSIC and the minimum-norm algorithms in resolving plane waves in noise. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(2):331–341, 1986.

[28] D.E. Johnston and P.M. Djuric. Bayesian detection and MMSE frequency estimation of sinusoidal signals via adaptive importance sampling. *International Symposium on Circuits and Systems, ISCAS*, 2:417–420, 1994.

[29] P. Stoica and Nehorai Arye. Music, maximum likelihood, and cramer-rao bound. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(5):720–741, 1989.

[30] F.-L. Luo and Y.-D. Li. Real-time neural computation of the noise subspace for the MUSIC algorithm. *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 1:485–488, 1993.

[31] R. W. Evans. Design guidelines for shielding effectiveness, current carrying capability, and the enhancement of conductivity of composite materials. Technical Report 4784, National Aeronautics and Space Administration, 1997.

[32] Acoustiblok Inc. Product description: Acoustical data, February 2008. http://www.acoustiblok.com/products.html.

[33] Glenwood Garner et. al. Acoustic-RF anechoic chamber design and evaluation. In *IEEE Radio and Wireless Symposium*, Orlando, FL, 2008.

[34] M. B. Steer and J. F. Sevic. Nonlinear RF and microwave circuit analysis, in CAD, simulation, and modeling. In Mike Golio, editor, *The RF and Microwave Handbook*. CRC Press, Inc., Boca Raton, FL, USA, 2001.

[35] A. Miklós, P. Hess, and Z. Bozóki. Application of acoustic resonators in photoacoustic trace gas analysis and metrology. *Review of Scientific Instruments*, 72:1937–1955, April 2001.

[36] Bassem R. Mahafza. *Radar Systems Analysis and Design Using MATLAB.* CRC Press, Inc., Boca Raton, FL, USA, 2000.

[37] JoAnna R. Vetreno. Analytic models for acoustic wave propagation in air. Master's thesis, North Carolina State University, 2007.

Appendix

Appendix A. Discrete Fourier Transform Matlab Code

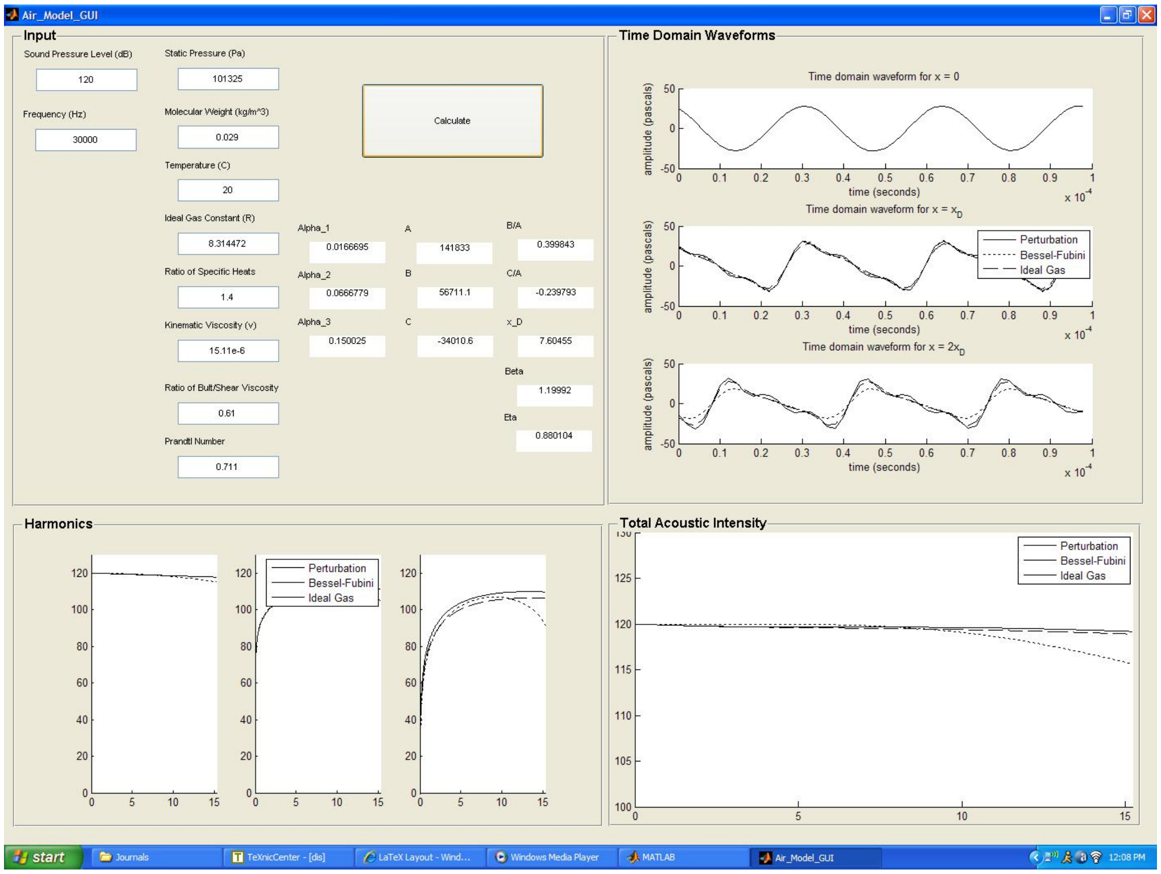
```
// function [Xk_mag, frequencies, k, Xk_real, Xk_imag] = general_ft(data,
//          delta_t, start_freq, end_freq)
//
// % Discrete Fourier Transform
// %
// %  usage:    general_ft(S_t, delta_t, start_freq, end_freq)
// %
// %  inputs:   S_t    =  time domain data
// %            delta_t     =  time between data samples
// %            start_freq  =  start of Fourier spectrum
// %            end_freq    =  end of Fourier spectrum
// %
// %  outputs:  Xk_mag = magnitude spectrum
// %            frequencies = frequency vector
// %            k = k vector (0 1 2 ... (end_freq-start_freq)/df)
// %            Xk_real = real magnitude spectrum
// %            Xk_imag = imaginary magnitude spectrum
//
// %length of input time domain data signal
// N = length(data);
// %frequency resolution from time/bandwidth product
// df = 1/(N*delta_t);
// %total time duration of input signal
// total_time = N*delta_t;
//
// for i=1:(((end_freq-start_freq)/df)+1)
//     %generation of output frequency vector
//     frequencies(i) = start_freq+((i-1)*df);
//     %generation of k vector
//     k(i) = frequencies(i) * total_time;
//     Xk_real(i) = 0;
//     Xk_imag(i) = 0;
//     for j=1:N
//         Xk_real(i) = (Xk_real(i)+(data(j)*cos(-2
//              *3.14159265358979323846*(j-1)*k(i)/N)));
//         Xk_imag(i) = (Xk_imag(i)+(data(j)*sin(-2
//              *3.14159265358979323846*(j-1)*k(i)/N)));
//     end
//     %real amplitude correction
//     Xk_real(i) = 2*Xk_real(i)/N;
//     %imaginary amplitude correction
//     Xk_imag(i) = 2*Xk_imag(i)/N;
//     %magnitude spectrum
//     Xk_mag(i) = sqrt((Xk_real(i)^2)+(Xk_imag(i)^2));
//     %conversion to dB scale
//     Xk_dB(i) = 20*log10((Xk_mag(i)*(sqrt(2)/2))/0.00002);
// end
//%uncomment to output amplitude in dB scale.
// %Xk_mag = Xk_dB;

#include <math.h>
#include <stdio.h>
#include "mex.h"
#include "matrix.h"
#include <stdlib.h>

void mexFunction(int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[]){

 double *data;
 double *d_t;
 double *f_st;
 double *f_sp;
 double *real;
 double *imag;
 double *mag;
 double *freq;
 double *k;

 int i,j;
 int N;
 double total_time;
 double samples;
 double d_f;
 double Pi = 3.14159265358979323846;

 data = mxGetPr(prhs[0]);
 d_t = mxGetPr(prhs[1]);
 f_st = mxGetPr(prhs[2]);
 f_sp = mxGetPr(prhs[3]);

 N = mxGetN(prhs[0]);
 total_time = N*(*d_t);

 if (*f_sp>*f_st){
     d_f = 1/total_time;
```

```
        samples = ((*f_sp-(*f_st))/d_f)+1;
    }
    else{
        d_f = 0;
        samples = 1;
    }

    plhs[0] = mxCreateDoubleMatrix(1,samples,mxREAL);
    plhs[1] = mxCreateDoubleMatrix(1,samples,mxREAL);
    plhs[2] = mxCreateDoubleMatrix(1,samples,mxREAL);
    plhs[3] = mxCreateDoubleMatrix(1,samples,mxREAL);
    plhs[4] = mxCreateDoubleMatrix(1,samples,mxREAL);

    mag = mxGetPr(plhs[0]);
    freq = mxGetPr(plhs[1]);
    k = mxGetPr(plhs[2]);
    real = mxGetPr(plhs[3]);
    imag = mxGetPr(plhs[4]);

    for(i=0; i<samples; i++){
        freq[i] = (*f_st)+(i*d_f);
        k[i] = freq[i]*total_time;
        real[i] = 0;
        imag[i] = 0;
        for(j=0; j<N; j++){
            real[i] = (real[i]+(data[j]*cos(-2*Pi*j*k[i]/N)));
            imag[i] = (imag[i]+(data[j]*sin(-2*Pi*j*k[i]/N)));
        }
        real[i] = 2*real[i]/N;
        imag[i] = 2*imag[i]/N;
        mag[i] = sqrt(pow(real[i],2)+pow(imag[i],2));
        //%Uncomment to output magnitude in dB scale
        //mag[i] = 20*log10((mag[i]*(sqrt(2)/2))/0.00002);
    }
}
```

Appendix B. Inverse Discrete Fourier Transform Matlab Code

```
// function [S_rec] = general_ift(Xk_real,Xk_imag,k,N)
//
// % Inverse Discrete Fourier Transform
// %
// %  usage:    general_ift(Xk_real, Xk_imag, k, N)
// %
// %  inputs:   Xk_real = real fourier coefficients
// %            Xk_imag = imaginary fourier coefficients
// %            k = k vector (0 1 2 ... (end_freq-start_freq)/df)
// %            N = number of output time samples
// %
// %  outputs:  S_rec = reconstructed signal of length N
//
// for n=1:N
//     S_rec(n) = 0;
//     for k_ind = 1:length(k)
//         S_rec(n) = S_rec(n)+((Xk_real(k_ind)*cos(2*pi*k(k_ind)*
//              (n-1)/N))-(Xk_imag(k_ind)*sin(2*pi*k(k_ind)*(n-1)/N)));
//     end
// end

#include <math.h>
#include <stdio.h>
#include "mex.h"
#include "matrix.h"
#include <stdlib.h>

void mexFunction(int nlhs, mxArray *plhs[],
    int nrhs, const mxArray *prhs[]){

 double *Xk_real;
 double *Xk_imag;
 double *k;
 double *N;
 double *Srec;

 int i,j;
 int samps;
 double Pi = 3.14159265358979323846;

 Xk_real = mxGetPr(prhs[0]);
 Xk_imag = mxGetPr(prhs[1]);
 k = mxGetPr(prhs[2]);
 N = mxGetPr(prhs[3]);

 samps = mxGetN(prhs[0]);
 plhs[0] = mxCreateDoubleMatrix(1,*N,mxREAL);
 Srec = mxGetPr(plhs[0]);

 for(i=0; i<*N; i++){
    Srec[i] = 0;
    for(j=0; j<samps; j++){
        Srec[i] = Srec[i]+((Xk_real[j]*cos(2*Pi*k[j]*i/(*N)))
            -(Xk_imag[j]*sin(2*Pi*k[j]*i/(*N))));
    }
 }
}
```

Appendix C. Air Model GUI Application Matlab Code


Air_Model_GUI

Input

Sound Pressure Level (dB): 120

Frequency (Hz): 30000

Static Pressure (Pa): 101325

Molecular Weight (kg/m^3): 0.029

Temperature (C): 20

Ideal Gas Constant (R): 8.314472

Ratio of Specific Heats: 1.4

Kinematic Viscosity (v): 15.11e-6

Ratio of Bult/Shear Viscosity: 0.61

Prandtl Number: 0.711

Alpha_1: 0.0166695

Alpha_2: 0.0666779

Alpha_3: 0.150025

A: 141833

B: 56711.1

C: -34010.6

B/A: 0.399843

C/A: -0.239793

x_D: 7.60455

Beta: 1.19992

Eta: 0.880104

```
function varargout = Air_Model_GUI(varargin)
% AIR_MODEL_GUI M-file for Air_Model_GUI.fig
%       AIR_MODEL_GUI, by itself, creates a new AIR_MODEL_GUI or raises
%       the existing singleton*.
%
%       H = AIR_MODEL_GUI returns the handle to a new AIR_MODEL_GUI or
%       the handle to the existing singleton*.
%
%       AIR_MODEL_GUI('CALLBACK',hObject,eventData,handles,...) calls the
%       local function named CALLBACK in AIR_MODEL_GUI.M with the given
%       input arguments.
%
%       AIR_MODEL_GUI('Property','Value',...) creates a new AIR_MODEL_GUI
%       or raises the existing singleton*.  Starting from the left,
%       property value pairs are applied to the GUI before
%       Air_Model_GUI_OpeningFunction gets called.  An unrecognized
%       property name or invalid value makes property application stop.
%       All inputs are passed to Air_Model_GUI_OpeningFcn via varargin.
```

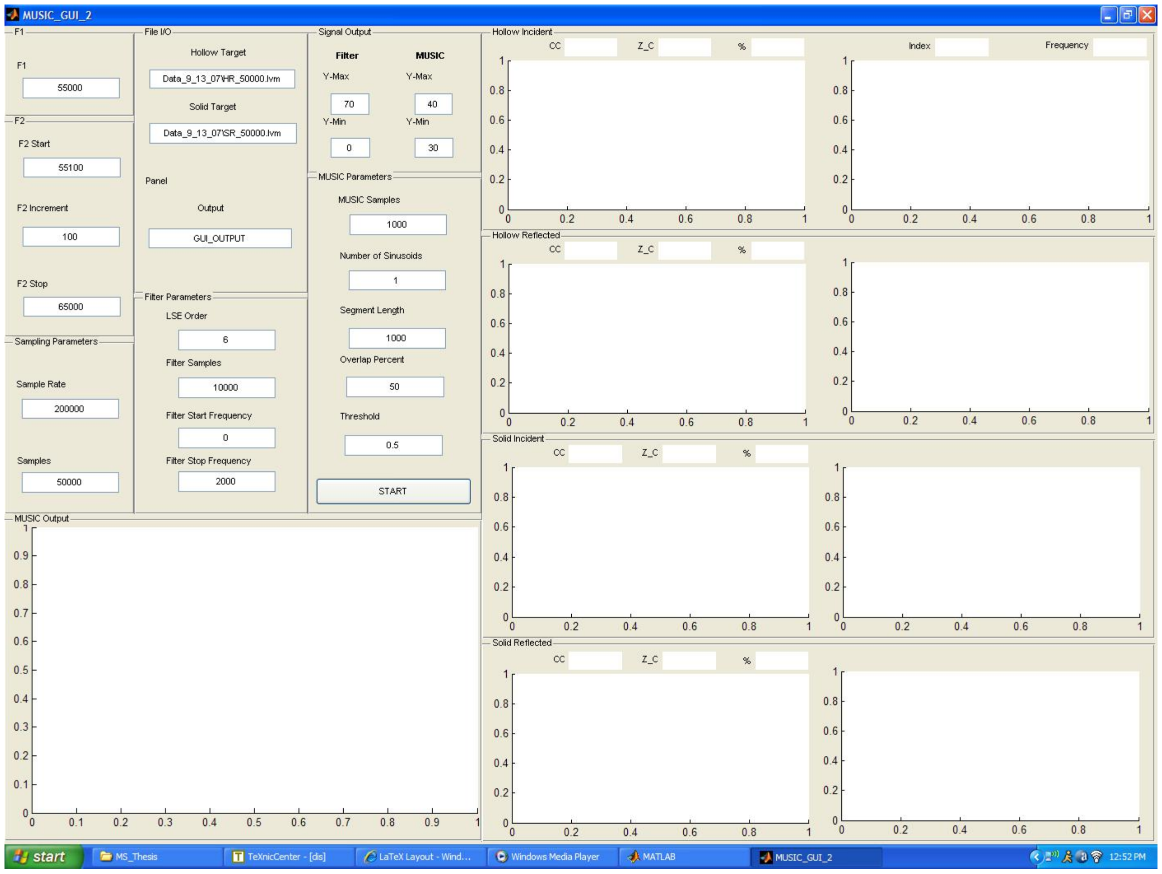
```
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
%      one instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Air_Model_GUI

% Last Modified by GUIDE v2.5 25-Feb-2008 14:54:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Air_Model_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @Air_Model_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Air_Model_GUI is made visible.
function Air_Model_GUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Air_Model_GUI (see VARARGIN)

% Choose default command line output for Air_Model_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

```
% UIWAIT makes Air_Model_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Air_Model_GUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

%Create SPL Input
function edit1_Callback(hObject, eventdata, handles)
    handles.SPL = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit1_CreateFcn(hObject, eventdata, handles)
    handles.SPL = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create frequency input
function edit2_Callback(hObject, eventdata, handles)
    handles.f = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit2_CreateFcn(hObject, eventdata, handles)
    handles.f = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Gamma input
function edit3_Callback(hObject, eventdata, handles)
    handles.Y = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit3_CreateFcn(hObject, eventdata, handles)
    handles.Y = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create ideal gas constant input
function edit4_Callback(hObject, eventdata, handles)
    handles.R = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit4_CreateFcn(hObject, eventdata, handles)
    handles.R = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create temperature input
function edit5_Callback(hObject, eventdata, handles)
    handles.T = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit5_CreateFcn(hObject, eventdata, handles)
    handles.T = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create molecular weight input
function edit6_Callback(hObject, eventdata, handles)
    handles.M = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit6_CreateFcn(hObject, eventdata, handles)
    handles.M = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create ambient pressure input
function edit7_Callback(hObject, eventdata, handles)
    handles.Po = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit7_CreateFcn(hObject, eventdata, handles)
    handles.Po = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create kinematic viscosity input
function edit8_Callback(hObject, eventdata, handles)
    handles.v = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit8_CreateFcn(hObject, eventdata, handles)
    handles.v = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create bulk viscosity ratio input
function edit9_Callback(hObject, eventdata, handles)
    handles.u_B = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit9_CreateFcn(hObject, eventdata, handles)
    handles.u_B = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Prandtl number input
function edit10_Callback(hObject, eventdata, handles)
    handles.Pr = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit10_CreateFcn(hObject, eventdata, handles)
    handles.Pr = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%This code runs when button is pushed
function pushbutton1_Callback(hObject, eventdata, handles)

    clc;

    SPL = handles.SPL;                  %Input sound pressure level
    f = handles.f;                      %frequency
```

```
Po = handles.Po;                    %ambient pressure
M = handles.M;                      %molecular mass of air
tempC = handles.T;                  %temp celsius
R = handles.R;                      %ideal gas constant
Y = handles.Y;                      %ratio of specific heat for air
v = handles.v;                      %kinematic viscosity coefficient
u_B = handles.u_B;                  %bulk viscosity coefficient
Pr = handles.Pr;                    %Prandtl number

Rs = R/M;                           %specific gas constant
c = 331 + 0.6*tempC;                %speed of sound
tempK = 273.15 + tempC;             %temp kelvin
rho_o = Po/(Rs*tempK);              %ambient density
Zo = c*rho_o;                       %characteristic impedence
w = 2*pi*f;                         %radial frequency
P = sqrt(2)*((10^(SPL/20))*20e-6);  %Conversion to pressure amplitude
U = P/(rho_o*c);                    %Conversion to particle veloctiy
Mu = U/c;                           %acoustic mach number
k = w/c;                            %wave number
V = (4/3)+u_B;                      %viscosity number

A = rho_o*(c^2);
%1st derivative of sound speed with respect to pressure
dcdp = ((((P+Po)/Po)^((Y-1)/(2*Y)))*(Y-1)*c)/(2*Y*(P+Po));
%2nd derivative of sound speed with respect to pressure
d2cdp2 = (((((P+Po)/Po)^((Y-1)/(2*Y)))*((Y-1)^2)*c)/(4*(Y^2)
    *((P+Po)^2)))-(((((P+Po)/Po)^((Y-1)/(2*Y)))
    *(Y-1)*c)/(2*Y*((P+Po)^2)));
BoA = 2*rho_o*c*dcdp;
CoA = ((3/2)*(BoA^2))+(2*(rho_o^2)*(c^3)*d2cdp2);
B = BoA*A;
C = CoA*A;

Bair = 1+(0.5*(BoA));              %2nd order nonlinear coefficient
Nair = 1+(0.5*(CoA));              %3rd order nonlinear coefficient
Gair = pi/(2*(rho_o^2)*(c^5));
Fair = 6*pi/c;

x_D = 1/(k*Bair*Mu);               %discontinuity distance

%1st order attenuation coefficient
alpha_1 = (((1*w)^2)*v*(V+((Y-1)/Pr)))/(2*(c^3));
%2nd order attenuation coefficient
alpha_2 = (((2*w)^2)*v*(V+((Y-1)/Pr)))/(2*(c^3));
```

```
%3rd order attenuation coefficient
alpha_3 = (((3*w)^2)*v*(V+((Y-1)/Pr)))/(2*(c^3));

Fs = 500000;                        %sample rate
time_samps = 50;                    %number of time samples
dt = 1/Fs;                          %sample period
t = 0:dt:(time_samps-1)*dt;         %time vector
x = x_D/1000:x_D/1000:2*x_D;        %1 dimension propagation distance

for i=1:length(x)
    %perturbation equations for 1st, 2nd, and 3rd harmonic
    P1(i,:) = exp(-alpha_1*x(i))*P*sin(w*t-k*x(i));
    P2(i,:) = exp(-alpha_2*x(i))*((Bair*w*(P^2)
        *x(i))/(2*rho_o*(c^3)))*sin(2*(w*t-k*x(i)));
    P3(i,:) = exp(-alpha_3*x(i))*(Gair*f*(P^3))
        *((Fair*f*(Bair^2)*(x(i)^2))+(2*Nair*x(i)))
        *sin(3*(w*t-k*x(i)));

    %bessel equations for 1st, 2nd, and 3rd harmonic
    B1(i,:) = 2*P*(besselj(1,x(i)/x_D)/(x(i)/x_D))
        *sin(1*(w*t-k*x(i)));
    B2(i,:) = 2*P*(besselj(2,2*x(i)/x_D)/(2*x(i)/x_D))
        *sin(2*(w*t-k*x(i)));
    B3(i,:) = 2*P*(besselj(3,3*x(i)/x_D)/(3*x(i)/x_D))
        *sin(3*(w*t-k*x(i)));

    %ideal gas treatment for 1st, 2nd, and 3rd harmonic
    S1(i,:) = exp(-alpha_1*x(i))*P*sin(w*t-k*x(i));
    S2(i,:) = exp(-alpha_2*x(i))*(P*0.5*Mu*Bair*k*x(i))
        .*sin(2*(w*t-k*x(i)));
    S3(i,:) = exp(-alpha_3*x(i))*(P*(Mu^2)*((Bair*k*x(i)).^2))
        .*(((sin(w*t-k*x(i))).*(cos(w*t-k*x(i)).^2))
        -(0.5*sin(w*t-k*x(i)).^3));
end

%add all harmonic sinusoids together
P_tot = P1+P2+P3;
B_tot = B1+B2+B3;
S_tot = S1+S2+S3;

set(handles.text11,'String',alpha_1);
set(handles.text13,'String',alpha_2);
set(handles.text15,'String',alpha_3);
set(handles.text17,'String',A);
set(handles.text19,'String',B);
set(handles.text21,'String',C);
set(handles.text29,'String',BoA);
set(handles.text23,'String',CoA);
set(handles.text25,'String',x_D);
set(handles.text31,'String',Bair);
set(handles.text33,'String',Nair);

axes(handles.axes1)
guidata(hObject,handles);
    subplot(3,1,1)
    hold on
    plot(t,P_tot(1,:),'k')
    plot(t,B_tot(1,:),'k:')
    plot(t,S_tot(1,:),'k--')
    hold off
    title('Time domain waveform for x = 0');
    xlabel('time (seconds)');
    ylabel('amplitude (pascals)');

    subplot(3,1,2)
    hold on
    plot(t,P_tot(1001,:),'k')
    plot(t,B_tot(1001,:),'k:')
    plot(t,S_tot(1001,:),'k--')
    hold off
    legend('Perturbation','Bessel-Fubini','Ideal Gas')
    title('Time domain waveform for x = x_D');
    xlabel('time (seconds)');
    ylabel('amplitude (pascals)');

    subplot(3,1,3)
    hold on
    plot(t,P_tot(2000,:),'k')
    plot(t,B_tot(2000,:),'k:')
    plot(t,S_tot(2000,:),'k--')
    hold off
    title('Time domain waveform for x = 2x_D');
    xlabel('time (seconds)');
    ylabel('amplitude (pascals)');
guidata(hObject,handles);

for i=1:length(x)
    %Perturbation amplitudes
    P1(i,:) = P*exp(-alpha_1*x(i));
    P2(i,:) = (((Bair*w*(P^2)*x(i))/(2*rho_o*(c^3))))
        *exp(-alpha_2*x(i));
    P3(i,:) = ((Gair*f*(P^3))*((Fair*f*(Bair^2)*(x(i)^2))
        +(2*Nair*x(i))))*exp(-alpha_3*x(i));

    %Bessel amplitudes
    B1(i,:) = 2*P*(besselj(1,x(i)/x_D)/(x(i)/x_D));
    B2(i,:) = 2*P*(besselj(2,2*x(i)/x_D)/(2*x(i)/x_D));
    B3(i,:) = 2*P*(besselj(3,3*x(i)/x_D)/(3*x(i)/x_D));

    %Ideal gas amplitudes
    S1(i,:) = P*exp(-alpha_1*x(i));
    S2(i,:) = ((P*0.5*Mu*Bair*k*x(i)))*exp(-alpha_2*x(i));
    S3(i,:) = (0.5*(P*(Mu^2)*((Bair*k*x(i)).^2)))*exp(-alpha_3*x(i));
end

P1_dB = 20*log10((P1/sqrt(2))/20e-6);
P2_dB = 20*log10((P2/sqrt(2))/20e-6);
P3_dB = 20*log10((P3/sqrt(2))/20e-6);

B1_dB = 20*log10((B1/sqrt(2))/20e-6);
B2_dB = 20*log10((B2/sqrt(2))/20e-6);
B3_dB = 20*log10((B3/sqrt(2))/20e-6);

S1_dB = 20*log10((S1/sqrt(2))/20e-6);
S2_dB = 20*log10((S2/sqrt(2))/20e-6);
S3_dB = 20*log10((S3/sqrt(2))/20e-6);

axes(handles.axes3)
guidata(hObject, handles)
    subplot(1,3,1)
    hold on
    plot(x,P1_dB,'k');
    plot(x,B1_dB,'k:');
    plot(x,S1_dB,'k--');
    hold off
    axis([0 2*x_D 0 130])

    subplot(1,3,2)
    hold on
    plot(x,P2_dB,'k');
    plot(x,B2_dB,'k:');
    plot(x,S2_dB,'k--');
    hold off
    axis([0 2*x_D 0 130])
    legend('Perturbation','Bessel-Fubini','Ideal Gas')

    subplot(1,3,3)
    hold on
    plot(x,P3_dB,'k');
    plot(x,B3_dB,'k:');
    plot(x,S3_dB,'k--');
    hold off
    axis([0 2*x_D 0 130])
guidata(hObject, handles)

%sum the RMS pressure of each harmonic together
P_tot = (P1/sqrt(2)).^2+(P2/sqrt(2)).^2+(P3/sqrt(2)).^2;
B_tot = (B1/sqrt(2)).^2+(B2/sqrt(2)).^2+(B3/sqrt(2)).^2;
S_tot = (S1/sqrt(2)).^2+(S2/sqrt(2)).^2+(S3/sqrt(2)).^2;
%compute total pressure in dB scale
P_tot_dB = 10*log10(P_tot/(20e-6)^2);
B_tot_dB = 10*log10(B_tot/(20e-6)^2);
S_tot_dB = 10*log10(S_tot/(20e-6)^2);

axes(handles.axes4)
guidata(hObject, handles)
    hold on
    plot(x,P_tot_dB,'k')
    plot(x,B_tot_dB,'k:')
    plot(x,S_tot_dB,'k--')
    hold off
    legend('Perturbation','Bessel-Fubini','Ideal Gas')
    axis([0 2*x_D 100 130])
guidata(hObject, handles)
```

Appendix D. MUSIC GUI Application Matlab Code



F1
F1: 55000

F2
F2 Start: 55100
F2 Increment: 100
F2 Stop: 65000

Sampling Parameters
Sample Rate: 200000
Samples: 50000

File I/O
Hollow Target: Data_9_13_07\HR_50000.lvm
Solid Target: Data_9_13_07\SR_50000.lvm

Panel
Output: GUI_OUTPUT

Filter Parameters
LSE Order: 6
Filter Samples: 10000
Filter Start Frequency: 0
Filter Stop Frequency: 2000

Signal Output
Filter Y-Max: 70
Filter Y-Min: 0
MUSIC Y-Max: 40
MUSIC Y-Min: 30

MUSIC Parameters
MUSIC Samples: 1000
Number of Sinusoids: 1
Segment Length: 1000
Overlap Percent: 50
Threshold: 0.5

```
function varargout = MUSIC_GUI_2(varargin)
% MUSIC_GUI_2 M-file for MUSIC_GUI_2.fig
%       MUSIC_GUI_2, by itself, creates a new MUSIC_GUI_2 or raises the
%       existing singleton*.
%
%       H = MUSIC_GUI_2 returns the handle to a new MUSIC_GUI_2 or the
%        handle to the existing singleton*.
%
%       MUSIC_GUI_2('CALLBACK',hObject,eventData,handles,...) calls the
%       local function named CALLBACK in MUSIC_GUI_2.M with the given
%       input arguments. MUSIC_GUI_2('Property','Value',...) creates a new
%       MUSIC_GUI_2 or raises the existing singleton*.  Starting from the
%       left, property value pairs are applied to the GUI before
%       MUSIC_GUI_2_OpeningFunction gets called.  An unrecognized property
%       name or invalid value makes property application stop.  All inputs
%       are passed to MUSIC_GUI_2_OpeningFcn via varargin. *See GUI
%       Options on GUIDE's Tools menu.  Choose "GUI allows only one
%       instance to run (singleton)".
```

```
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MUSIC_GUI_2

% Last Modified by GUIDE v2.5 26-Feb-2008 10:42:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MUSIC_GUI_2_OpeningFcn, ...
                   'gui_OutputFcn',  @MUSIC_GUI_2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before MUSIC_GUI_2 is made visible.
function MUSIC_GUI_2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MUSIC_GUI_2 (see VARARGIN)

% Choose default command line output for MUSIC_GUI_2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MUSIC_GUI_2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
```

```
function varargout = MUSIC_GUI_2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
%Create frequency f1 input
function edit1_Callback(hObject, eventdata, handles)
    handles.f1 = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit1_CreateFcn(hObject, eventdata, handles)
        handles.f1 = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create frequency start input
function edit2_Callback(hObject, eventdata, handles)
    handles.freq_sweep_st = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit2_CreateFcn(hObject, eventdata, handles)
        handles.freq_sweep_st = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create frequency increment input
function edit3_Callback(hObject, eventdata, handles)
    handles.delta_f = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit3_CreateFcn(hObject, eventdata, handles)
        handles.delta_f = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create frequency sweep stop input
function edit4_Callback(hObject, eventdata, handles)
    handles.freq_sweep_sp = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit4_CreateFcn(hObject, eventdata, handles)
        handles.freq_sweep_sp = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create sampling frequency input
function edit5_Callback(hObject, eventdata, handles)
    handles.Fs = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit5_CreateFcn(hObject, eventdata, handles)
        handles.Fs = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create time samples input
function edit6_Callback(hObject, eventdata, handles)
    handles.time_samps = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit6_CreateFcn(hObject, eventdata, handles)
        handles.time_samps = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create input file name input
function edit7_Callback(hObject, eventdata, handles)
    handles.Hollow_Target_File = get(hObject,'String');
    guidata(hObject,handles);
    function edit7_CreateFcn(hObject, eventdata, handles)
        handles.Hollow_Target_File = get(hObject,'String');
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create input file name input
function edit8_Callback(hObject, eventdata, handles)
    handles.Solid_Target_File = get(hObject,'String');
    guidata(hObject,handles);
    function edit8_CreateFcn(hObject, eventdata, handles)
        handles.Solid_Target_File = get(hObject,'String');
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create output file name input
function edit9_Callback(hObject, eventdata, handles)
    handles.Output_File = get(hObject,'String');
    guidata(hObject,handles);
    function edit9_CreateFcn(hObject, eventdata, handles)
        handles.Output_File = get(hObject,'String');
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Fitlter LSE order input
function edit18_Callback(hObject, eventdata, handles)
    handles.LSE_Order = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit18_CreateFcn(hObject, eventdata, handles)
    handles.LSE_Order = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Filter time samples input
function edit10_Callback(hObject, eventdata, handles)
    handles.Filter_Samples = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit10_CreateFcn(hObject, eventdata, handles)
        handles.Filter_Samples = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Filter start frequency input
function edit11_Callback(hObject, eventdata, handles)
    handles.Filter_Start_Freq = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit11_CreateFcn(hObject, eventdata, handles)
        handles.Filter_Start_Freq = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Filter stop frequency input
function edit12_Callback(hObject, eventdata, handles)
    handles.Filter_Stop_Freq = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit12_CreateFcn(hObject, eventdata, handles)
        handles.Filter_Stop_Freq = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Filter plot maximum y-axis input
function edit20_Callback(hObject, eventdata, handles)
    handles.Filter_Max = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit20_CreateFcn(hObject, eventdata, handles)
    handles.Filter_Max = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create Filter plot minimum y-axis input
function edit21_Callback(hObject, eventdata, handles)
    handles.Filter_Min = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit21_CreateFcn(hObject, eventdata, handles)
    handles.Filter_Min = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC plot maximum y-axis input
function edit22_Callback(hObject, eventdata, handles)
    handles.Music_Max = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit22_CreateFcn(hObject, eventdata, handles)
    handles.Music_Max = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC plot minimum y-axis input
function edit23_Callback(hObject, eventdata, handles)
    handles.Music_Min = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit23_CreateFcn(hObject, eventdata, handles)
    handles.Music_Min = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC time samples input
function edit24_Callback(hObject, eventdata, handles)
    handles.music_samps = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit24_CreateFcn(hObject, eventdata, handles)
    handles.music_samps = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC number of sinusoids to detect input
function edit14_Callback(hObject, eventdata, handles)
    handles.num_of_sins = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit14_CreateFcn(hObject, eventdata, handles)
        handles.num_of_sins = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC segment length input
function edit15_Callback(hObject, eventdata, handles)
    handles.segment_length = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit15_CreateFcn(hObject, eventdata, handles)
        handles.segment_length = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC overlap percent input
function edit16_Callback(hObject, eventdata, handles)
    handles.overlap_percent = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit16_CreateFcn(hObject, eventdata, handles)
        handles.overlap_percent = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Create MUSIC threshold input
function edit17_Callback(hObject, eventdata, handles)
    handles.threshold = str2double(get(hObject,'String'));
    guidata(hObject,handles);
    function edit17_CreateFcn(hObject, eventdata, handles)
        handles.threshold = str2double(get(hObject,'String'));
        guidata(hObject,handles);
    if ispc && isequal(get(hObject,'BackgroundColor'),
        get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
%Excecute on button press
function pushbutton1_Callback(hObject, eventdata, handles)
    clc
    guidata(hObject,handles);

    try
        %read inputs from GUI
        freq_sweep_st = handles.freq_sweep_st;
        delta_f = handles.delta_f;
        freq_sweep_sp = handles.freq_sweep_sp;

        f2 = freq_sweep_st:delta_f:freq_sweep_sp;
        f1 = linspace(handles.f1,handles.f1,length(f2));
        d_f = f2-f1;
```

```
Fs = handles.Fs;
time_samps = handles.time_samps;
dt = 1/Fs;

Order = handles.LSE_Order;
filt_samps = handles.Filter_Samples;
filt_st = handles.Filter_Start_Freq;
filt_sp = handles.Filter_Stop_Freq;

music_samps = handles.music_samps;
NS = handles.num_of_sins;
SL = handles.segment_length;
OLP = handles.overlap_percent;
Thresh = handles.threshold;

try
    %load data
    Hollow_Data = load(char(handles.Hollow_Target_File));
    Solid_Data = load(char(handles.Solid_Target_File));
    %parse data
    try
        for ind=1:length(d_f)
            l_ind = ((ind-1)*time_samps)+1;
            u_ind = ind*time_samps;
            Hollow_inc(ind,:) = Hollow_Data(l_ind:u_ind,2);
            Hollow_ref(ind,:) = Hollow_Data(l_ind:u_ind,3);
            Solid_inc(ind,:) = Solid_Data(l_ind:u_ind,2);
            Solid_ref(ind,:) = Solid_Data(l_ind:u_ind,3);
        end
            %initialize output values
            Z_HI = 0;
            Z_HR = 0;
            Z_SI = 0;
            Z_SR = 0;

            M_freq_HI = zeros(1,length(d_f));
            M_freq_HR = zeros(1,length(d_f));
            M_freq_SI = zeros(1,length(d_f));
            M_freq_SR = zeros(1,length(d_f));

            M_amp_HI = zeros(1,length(d_f));
            M_amp_HR = zeros(1,length(d_f));
            M_amp_SI = zeros(1,length(d_f));
```

```
M_amp_SR = zeros(1,length(d_f));

M_amp_HI_dB = zeros(1,length(d_f));
M_amp_HR_dB = zeros(1,length(d_f));
M_amp_SI_dB = zeros(1,length(d_f));
M_amp_SR_dB = zeros(1,length(d_f));

for ind=1:length(d_f)
    FOI = ind*delta_f;  %Frequency of interest
    set(handles.text44,'String',ind);
    set(handles.text43,'String',FOI);
        %read block of noisy data
        S_n_t = Hollow_inc(ind,1:filt_samps);
        fft_st = 0;
        fft_sp = d_f(length(d_f));
        %Call the prewhitening filter
        [S_w_t, S_w_f, S_n_f, freq, apx_dB,
            filt_freq, CC] = f_white_GUI
            (S_n_t(1:filt_samps),dt,fft_st,fft_sp,
            filt_st,filt_sp,ind,delta_f, Order);
        %plot noisy data, approximation, and whitening
        %data
        axes(handles.axes3)
        cla(handles.axes3,'reset');
        guidata(hObject,handles);
            hold on
            plot(freq,S_n_f,'r');
            plot(freq,S_w_f,'b');
            plot(filt_freq, apx_dB, 'g');
            axis([fft_st,fft_sp,handles.Filter_Min,
                handles.Filter_Max]);
            hold off
        set(handles.text20,'String',CC);
        guidata(hObject,handles);
        %analyze data with MUSIC algorithm
        d = (floor((filt_samps-music_samps)/2))+1;
        [M_amp_HI(ind), M_freq_HI(ind), Spectrum,
            Frequencies, Z] = my_music_GUI
            (S_w_t(d:(d-1)+music_samps),Fs,FOI,
            delta_f, NS, SL, OLP, Thresh);
        %keep track of zeroed coefficients
        %and plot data
        Z_HI = Z_HI + Z;
        P_HI = 100*(1-(Z_HI/length(d_f)));
```

```
axes(handles.axes4)
cla(handles.axes4,'reset');
guidata(hObject,handles);
    hold on
    plot(Frequencies,Spectrum,'b');
    stem(M_freq_HI(ind),M_amp_HI(ind),'r');
    hold off
    axis([0 10000 handles.Music_Min
        handles.Music_Max]);
    set(handles.text19,'String',Z_HI);
    set(handles.text18,'String',P_HI);
guidata(hObject,handles);

S_n_t = Hollow_ref(ind,1:filt_samps);
fft_st = 0;
fft_sp = d_f(length(d_f));

[S_w_t, S_w_f, S_n_f, freq, apx_dB, filt_freq,
    CC] = f_white_GUI(S_n_t(1:filt_samps),dt,
    fft_st,fft_sp,filt_st,filt_sp,ind,delta_f,
    Order);

axes(handles.axes5)
cla(handles.axes5,'reset');
guidata(hObject,handles);
    hold on
    plot(freq,S_n_f,'r');
    plot(freq,S_w_f,'b');
    plot(filt_freq, apx_dB, 'g');
    axis([fft_st,fft_sp,handles.Filter_Min,
        handles.Filter_Max]);
    hold off
set(handles.text27,'String',CC);
guidata(hObject,handles);

d = (floor((filt_samps-music_samps)/2))+1;
[M_amp_HR(ind), M_freq_HR(ind), Spectrum,
    Frequencies, Z] = my_music_GUI
    (S_w_t(d:(d-1)+music_samps),Fs,FOI,
    delta_f, NS, SL, OLP, Thresh);

Z_HR = Z_HR + Z;
P_HR = 100*(1-(Z_HR/length(d_f)));
axes(handles.axes6)
```

```
cla(handles.axes6,'reset');
guidata(hObject,handles);
    hold on
    plot(Frequencies,Spectrum,'b');
    stem(M_freq_HR(ind),M_amp_HR(ind),'r');
    hold off
    axis([0 10000 handles.Music_Min
        handles.Music_Max]);
    set(handles.text26,'String',Z_HR);
    set(handles.text25,'String',P_HR);
guidata(hObject,handles);

S_n_t = Solid_inc(ind,1:filt_samps);
fft_st = 0;
fft_sp = d_f(length(d_f));

[S_w_t, S_w_f, S_n_f, freq, apx_dB, filt_freq,
    CC] = f_white_GUI(S_n_t(1:filt_samps),dt,
    fft_st,fft_sp,filt_st,filt_sp,ind,
    delta_f, Order);

axes(handles.axes7)
cla(handles.axes7,'reset');
guidata(hObject,handles);
    hold on
    plot(freq,S_n_f,'r');
    plot(freq,S_w_f,'b');
    plot(filt_freq, apx_dB, 'g');
    axis([fft_st,fft_sp,handles.Filter_Min,
        handles.Filter_Max]);
    hold off
set(handles.text33,'String',CC);
guidata(hObject,handles);

d = (floor((filt_samps-music_samps)/2))+1;
[M_amp_SI(ind), M_freq_SI(ind), Spectrum,
    Frequencies, Z] = my_music_GUI
    (S_w_t(d:(d-1)+music_samps),Fs,FOI,
    delta_f, NS, SL, OLP, Thresh);

Z_SI = Z_SI + Z;
P_SI = 100*(1-(Z_SI/length(d_f)));
axes(handles.axes8)
cla(handles.axes8,'reset');
```

```
guidata(hObject,handles);
    hold on
    plot(Frequencies,Spectrum,'b');
    stem(M_freq_SI(ind),M_amp_SI(ind),'r');
    hold off
    axis([0 10000 handles.Music_Min
        handles.Music_Max]);
    set(handles.text32,'String',Z_SI);
    set(handles.text31,'String',P_SI);
guidata(hObject,handles);

S_n_t = Solid_ref(ind,1:filt_samps);
fft_st = 0;
fft_sp = d_f(length(d_f));

[S_w_t, S_w_f, S_n_f, freq, apx_dB, filt_freq,
    CC] = f_white_GUI(S_n_t(1:filt_samps),dt,
    fft_st,fft_sp,filt_st,filt_sp,ind,
    delta_f, Order);

axes(handles.axes9)
cla(handles.axes9,'reset');
guidata(hObject,handles);
    hold on
    plot(freq,S_n_f,'r');
    plot(freq,S_w_f,'b');
    plot(filt_freq, apx_dB, 'g');
    axis([fft_st,fft_sp,handles.Filter_Min,
        handles.Filter_Max]);
    hold off
set(handles.text39,'String',CC);
guidata(hObject,handles);

d = (floor((filt_samps-music_samps)/2))+1;
[M_amp_SR(ind), M_freq_SR(ind), Spectrum,
    Frequencies, Z] = my_music_GUI
    (S_w_t(d:(d-1)+music_samps),Fs,FOI,
    delta_f, NS, SL, OLP, Thresh);

Z_SR = Z_SR + Z;
P_SR = 100*(1-(Z_SR/length(d_f)));
axes(handles.axes10)
cla(handles.axes10,'reset');
guidata(hObject,handles);
```

```
                    hold on
                    plot(Frequencies,Spectrum,'b');
                    stem(M_freq_SR(ind),M_amp_SR(ind),'r');
                    hold off
                    axis([0 10000 handles.Music_Min
                        handles.Music_Max]);
                    set(handles.text38,'String',Z_SR);
                    set(handles.text37,'String',P_SR);
                guidata(hObject,handles);
                %Convert MUSIC coefficients to dB scale
                M_amp_HI_dB(ind) = 20*log10((M_amp_HI(ind)/
                    sqrt(2))/20e-6);
                M_amp_HR_dB(ind) = 20*log10((M_amp_HR(ind)/
                    sqrt(2))/20e-6);
                M_amp_SI_dB(ind) = 20*log10((M_amp_SI(ind)/
                    sqrt(2))/20e-6);
                M_amp_SR_dB(ind) = 20*log10((M_amp_SR(ind)/
                    sqrt(2))/20e-6);
                %plot in real time
                axes(handles.axes11)
                cla(handles.axes11,'reset');
                guidata(hObject,handles);
                    hold on
                    plot(M_freq_HI(1:ind),M_amp_HI_dB(1:ind),
                        'b');
                    plot(M_freq_HR(1:ind),M_amp_HR_dB(1:ind),
                        'r');
                    plot(M_freq_SI(1:ind),M_amp_SI_dB(1:ind),
                        'b:');
                    plot(M_freq_SR(1:ind),M_amp_SR_dB(1:ind),
                        'r:');
                    hold off
                    legend('Hollow Incident',
                        'Hollow Reflected','Solid Incident',
                        'Solid Reflected');
                guidata(hObject,handles);
        end
        %save output data
        Output = [M_freq_HI' M_amp_HI' M_freq_HR' M_amp_HR'
            M_freq_SI' M_amp_SI' M_freq_SR' M_amp_SR'];
        File_IO = handles.Output_File;
        save File_IO Output;
%throw error if data is not parsed correctly
catch
```

```
            disp('Error parsing data - Please check frequency
                increments/time samples');
            lasterror
        end
    %throw error if filenames are misspelled
    catch
        disp('Error loading file(s) - Please check spelling/syntax');
        lasterror
    end
%throw error if non-numeric data is entered
catch
    disp('Error reading data - Please check frequency
        increments/time samples');
    lasterror
end
```

Appendix E. Prewhitening Filter Matlab Code

```
function [S_w_C,amp_filt_dB_C,amp_dB,freq,apx_dB,filt_freq,CC]
= f_white_GUI(S_n_t,dt,fft_st,fft_sp,filt_st,filt_sp,ind,delta_f, order)

% Prewhitening Filter: whitens narrow band signals in 1/f noise
%
%  usage:
%  f_white_GUI(S_n_t,dt,fft_st,fft_sp,filt_st,filt_sp,ind,delta_f,order)
%
%  inputs:   S_n_t = noisy time domain data
%            dt = delta t
%            fft_st = DFT start frequency
%            fft_sp = DFT end frequency
%            filt_st = filter start frequency
%            filt_sp = filter stop frequency
%            ind = calling loop index
%            delta_f = frequency step
%            order = LSE approximation order
%
%  outputs:  S_w_C = whitened time domain data
%            amp_filt_dB_C = whitened and amplitude corrected spectrum
%            amp_dB = noisy unfiltered spectrum
%            freq = frequency vector
%            apx_dB = noise approximation in dB
%            filt_freq = filter frequency vector
%            CC = correction coefficient

%Take the DFT of the input signal
[amp,freq,k,re,im] = mex_fft(S_n_t,dt,fft_st,fft_sp);
%Determine the frequency resolution
f_res = 1/(dt*length(S_n_t));
%Compute the phase component
phi = atan(im/re);
%Use dB scale
amp_dB = 20*log10((amp*(sqrt(2)/2))/20e-6);

warning ('off');
%Compute the upper and lower filter frequencies
l_ind = (floor(filt_st/f_res))+1;
u_ind = ceil(filt_sp/f_res);
filt_freq = freq(l_ind:u_ind);
%Least square polyfit of noise over filter frequencies
P = polyfit(filt_freq,amp_dB(l_ind:u_ind),order);
warning ('on');
```

```
%Approximation of noise in dB and its minimum
apx_dB = polyval(P,filt_freq);
apx_min = min(apx_dB);

%Some code to deal with noise floor being negative dB
if apx_min<0
    apx_dB = -apx_dB;
    if min(apx_dB)<0
        apx_dB = (apx_dB-(min(apx_dB)))+1;
    end
    %The filter is essentially the inverse of the approximation
    filt_dB = 1./apx_dB;
    %And needs to be normalized to one
    filt_dB = filt_dB*(1/max(filt_dB));
    amp_filt_dB = 1*amp_dB;
    amp_filt_dB(1:length(filt_dB)) = amp_dB(1:length(filt_dB)).*filt_dB;
else
    filt_dB = 1./apx_dB;
    filt_dB = filt_dB*(1/max(filt_dB));
    amp_filt_dB = 1*amp_dB;
    amp_filt_dB(1:length(filt_dB)) = amp_dB(1:length(filt_dB)).*filt_dB;
end
%Determine what the correction coefficient is and what index
%it applies to
if (ind*delta_f)>=filt_sp
    CC_ind = (((ind*delta_f)-filt_st)/f_res)+1;
    CC = 1;
else
    CC_ind = (((ind*delta_f)-filt_st)/f_res)+1;
    CC = 1/filt_dB(CC_ind);
end
freq(CC_ind);
CC;
%Rescale the filtered amplitude to Pascals
amp_filt = ((10.^(amp_filt_dB/20))*20e-6)/(sqrt(2)/2);
%Find the real and imaginary parts
re_filt = amp_filt*cos(phi);
im_filt = amp_filt*sin(phi);
%Use inverse DFT to return to time domain
S_w = mex_ifft(re_filt,im_filt,k,length(S_n_t));

%Use the correction coefficient
amp_filt_dB_C = rescale(amp_filt_dB, CC, CC_ind);
```

```
%Rescale the filtered amplitude to Pascals
amp_filt_C = ((10.^(amp_filt_dB_C/20))*20e-6)/(sqrt(2)/2);
%Find the real and imaginary parts
re_filt_C = amp_filt_C*cos(phi);
im_filt_C = amp_filt_C*sin(phi);
%Use inverse DFT to return to time domain
S_w_C = mex_ifft(re_filt_C,im_filt_C,k,length(S_n_t));
```

Appendix F. MUSIC Algorithm Matlab Code

```
function [M_amp, M_freq, M_spectrum, M_frequencies, Z]
    = my_music_GUI(S_w_t, Fs, freq, delta_f, NS, SL, OLP, THRESH)

% Multiple Signal Classification
%
%  usage:    my_music_GUI(S_w_t, Fs, freq, delta_f, NS, SL, OLP THRESH)
%
%  inputs:   S_w_t = whitened time domain data
%            Fs = sampling frequency
%            freq = the current frequency being analyzed
%            delta_f = the frequency step in the sweep
%            NS = number of sinusoids to estimate
%            SL = segment length used to partition data
%            OLP = overlap percent in segmentation
%            THRESH = the threshold between noise and signal subspace
%
%  outputs:  M_amp = the amplitude of the MUSIC spectrum at the
%                    frequency being analyzed
%            M_freq = the frequency estimate of the MUSIC spectrum at
%                     the frequency being analyzed
%            M_spectrum = the entire MUSIC spectrum amplitudes
%            M_frequencies = the entire MUSIC spectrum frequencies
%            Z = flag thrown if frequency estimate doesn't match

%input parameters
WINNAME = 'Rectangular';
WINPARAM = '';
FFTLENGTH = 'NextPow2';
INPUTTYPE = 'Vector';
%define the pseudospectrum estimator
Hs = spectrum.music(NS,SL,OLP,WINNAME,THRESH,FFTLENGTH,INPUTTYPE);

%use Hs to calculate the pseudospectrum of S_w_t
HPS = pseudospectrum(Hs,S_w_t,'Fs',Fs,'SpectrumRange','half');
%Pull out the amplitude of HPS
M_spectrum = HPS.data;
%Convert to dB
M_spectrum = 20*log10((M_spectrum*sqrt(2)/2)/20e-6);
%Pull out the frequencies of HPS
M_frequencies = HPS.freq;

%Find the maximum of the spectrum
[M_amp, ind] = max(M_spectrum);
```

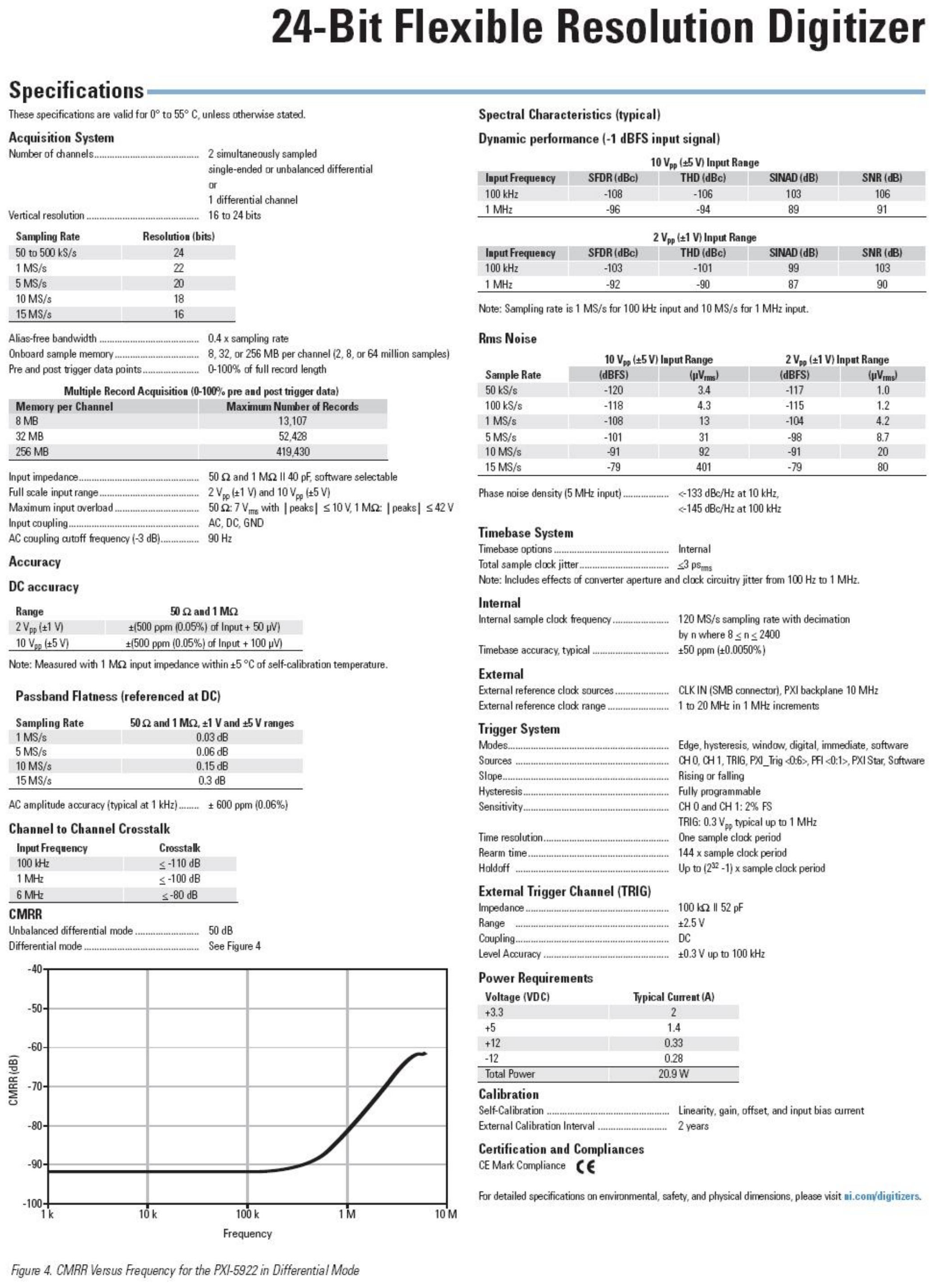
```
M_freq = M_frequencies(ind);
%Define upper and lower frequency thresholds
thresh_l = freq - (delta_f/2);
thresh_u = freq + (delta_f/2);

%See if MUSIC frequency estimate is within threshold
if (thresh_l<M_freq && M_freq<thresh_u)
    M_freq = 1*M_freq;
    M_amp = 1*M_amp;
    Z = 0;
%if it's not, throw the flag.
else
    M_freq = freq;
    %M_amp gets set to MUSIC noise floor (30)
    M_amp = 30;
    Z = 1;
end
```

Appendix G. Amplitude Correction Matlab Code

```
function [amp_C] = rescale(amp, CC, CC_ind)

% Amplitude Correction: rescales a single sinusoid in white noise to
% original amplitude
%
%  usage: rescale(amp, CC, CC_ind)
%
%  inputs:   amp = a row vector containing the DFT coefficients
%            CC = the computed correction coefficient from
%                 'f_white_GUI.m'
%            CC_ind = the index of the vector 'amp' that needs rescaling
%
%  outputs:  amp_C = the rescaled vector of DFT coefficients

amp_C = amp;

amp_C(CC_ind) = CC*amp_C(CC_ind);
```

Appendix H. Equipment Specifications

National Instruments PXI-5922 High Speed Digitizer



24-Bit Flexible Resolution Digitizer

Specifications

These specifications are valid for 0° to 55° C, unless otherwise stated.

Acquisition System

Number of channels	2 simultaneously sampled single-ended or unbalanced differential or 1 differential channel
Vertical resolution	16 to 24 bits

Sampling Rate	Resolution (bits)
50 to 500 kS/s	24
1 MS/s	22
5 MS/s	20
10 MS/s	18
15 MS/s	16

Alias-free bandwidth	0.4 x sampling rate
Onboard sample memory	8, 32, or 256 MB per channel (2, 8, or 64 million samples)
Pre and post trigger data points	0-100% of full record length

Multiple Record Acquisition (0-100% pre and post trigger data)

Memory per Channel	Maximum Number of Records
8 MB	13,107
32 MB	52,428
256 MB	419,430

Input impedance	50 Ω and 1 MΩ II 40 pF, software selectable
Full scale input range	2 V_{pp} (±1 V) and 10 V_{pp} (±5 V)
Maximum input overload	50 Ω: 7 V_{rms} with |peaks| ≤ 10 V, 1 MΩ: |peaks| ≤ 42 V
Input coupling	AC, DC, GND
AC coupling cutoff frequency (-3 dB)	90 Hz

Accuracy

DC accuracy

Range	50 Ω and 1 MΩ
2 V_{pp} (±1 V)	±(500 ppm (0.05%) of Input + 50 μV)
10 V_{pp} (±5 V)	±(500 ppm (0.05%) of Input + 100 μV)

Note: Measured with 1 MΩ input impedance within ±5 °C of self-calibration temperature.

Passband Flatness (referenced at DC)

Sampling Rate	50 Ω and 1 MΩ, ±1 V and ±5 V ranges
1 MS/s	0.03 dB
5 MS/s	0.06 dB
10 MS/s	0.15 dB
15 MS/s	0.3 dB

AC amplitude accuracy (typical at 1 kHz)	± 600 ppm (0.06%)

Channel to Channel Crosstalk

Input Frequency	Crosstalk
100 kHz	≤ -110 dB
1 MHz	≤ -100 dB
6 MHz	≤ -80 dB

CMRR

Unbalanced differential mode	50 dB
Differential mode	See Figure 4

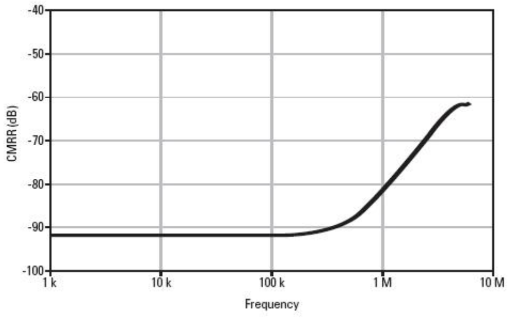


Figure 4. CMRR Versus Frequency for the PXI-5922 in Differential Mode

Spectral Characteristics (typical)

Dynamic performance (-1 dBFS input signal)

10 V_{pp} (±5 V) Input Range

Input Frequency	SFDR (dBc)	THD (dBc)	SINAD (dB)	SNR (dB)
100 kHz	-108	-106	103	106
1 MHz	-96	-94	89	91

2 V_{pp} (±1 V) Input Range

Input Frequency	SFDR (dBc)	THD (dBc)	SINAD (dB)	SNR (dB)
100 kHz	-103	-101	99	103
1 MHz	-92	-90	87	90

Note: Sampling rate is 1 MS/s for 100 kHz input and 10 MS/s for 1 MHz input.

Rms Noise

	10 V_{pp} (±5 V) Input Range		2 V_{pp} (±1 V) Input Range	
Sample Rate	(dBFS)	(μV_{rms})	(dBFS)	(μV_{rms})
50 kS/s	-120	3.4	-117	1.0
100 kS/s	-118	4.3	-115	1.2
1 MS/s	-108	13	-104	4.2
5 MS/s	-101	31	-98	8.7
10 MS/s	-91	92	-91	20
15 MS/s	-79	401	-79	80

Phase noise density (5 MHz input)	<-133 dBc/Hz at 10 kHz, <-145 dBc/Hz at 100 kHz

Timebase System

Timebase options	Internal
Total sample clock jitter	≤3 ps_{rms}

Note: Includes effects of converter aperture and clock circuitry jitter from 100 Hz to 1 MHz.

Internal

Internal sample clock frequency	120 MS/s sampling rate with decimation by n where $8 \leq n \leq 2400$
Timebase accuracy, typical	±50 ppm (±0.0050%)

External

External reference clock sources	CLK IN (SMB connector), PXI backplane 10 MHz
External reference clock range	1 to 20 MHz in 1 MHz increments

Trigger System

Modes	Edge, hysteresis, window, digital, immediate, software
Sources	CH 0, CH 1, TRIG, PXI_Trig <0:6>, PFI <0:1>, PXI Star, Software
Slope	Rising or falling
Hysteresis	Fully programmable
Sensitivity	CH 0 and CH 1: 2% FS TRIG: 0.3 V_{pp} typical up to 1 MHz
Time resolution	One sample clock period
Rearm time	144 x sample clock period
Holdoff	Up to (2^{32} -1) x sample clock period

External Trigger Channel (TRIG)

Impedance	100 kΩ II 52 pF
Range	±2.5 V
Coupling	DC
Level Accuracy	±0.3 V up to 100 kHz

Power Requirements

Voltage (VDC)	Typical Current (A)
+3.3	2
+5	1.4
+12	0.33
-12	0.28
Total Power	20.9 W

Calibration

Self-Calibration	Linearity, gain, offset, and input bias current
External Calibration Interval	2 years

Certification and Compliances

CE Mark Compliance CE

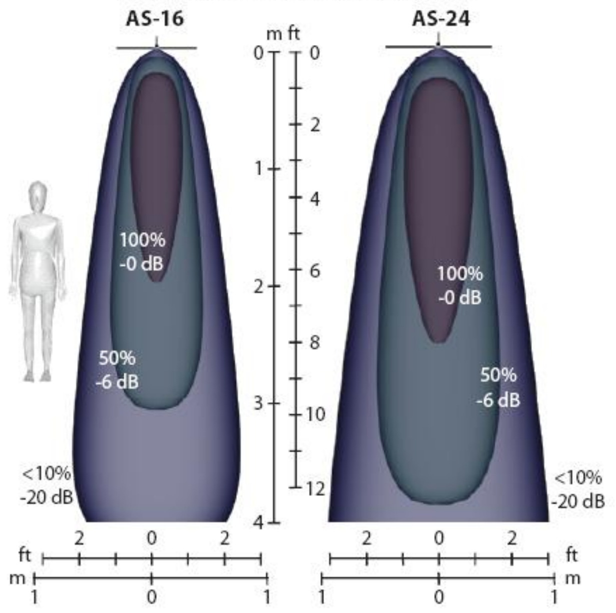
For detailed specifications on environmental, safety, and physical dimensions, please visit **ni.com/digitizers**.

Holosonic Audio Spotlight

audiospotlight®

Technical Specifications

Sound Field Distribution



Sound field distribution is shown with equal-loudness contours for a standard 1 kHz tone. The center area is loudest at 100% amplitude, while the sound level just outside the illustrated beam area is less than 10%.

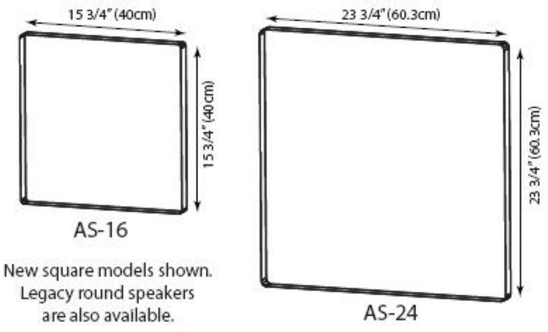
Audio Spotlight systems are much less sensitive to listener distance than traditional loudspeakers, but maximum performance is attained at roughly 1-2m (3-6 ft) from the listener.

Typical levels are 80 dB SPL at 1kHz for AS-16, and 85 dB SPL for AS-24 models. The larger AS-24 can output about twice the power and has twice low-frequency range of the AS-16.

Amplifier Specifications

Input:	RCA line-level audio
Power draw:	65W max (AS-24)
	50W max (AS-16)
Output:	BNC coax cable
	(25′ / 7m included)
Controls:	Volume, tone, on/off
Voltage:	100-240V 50/60Hz
Dimensions:	6″w x 7″d x 1.6″h
	(15cm x 18cm x 4cm)

Speaker Dimensions (thickness ~0.5″ / 1cm)

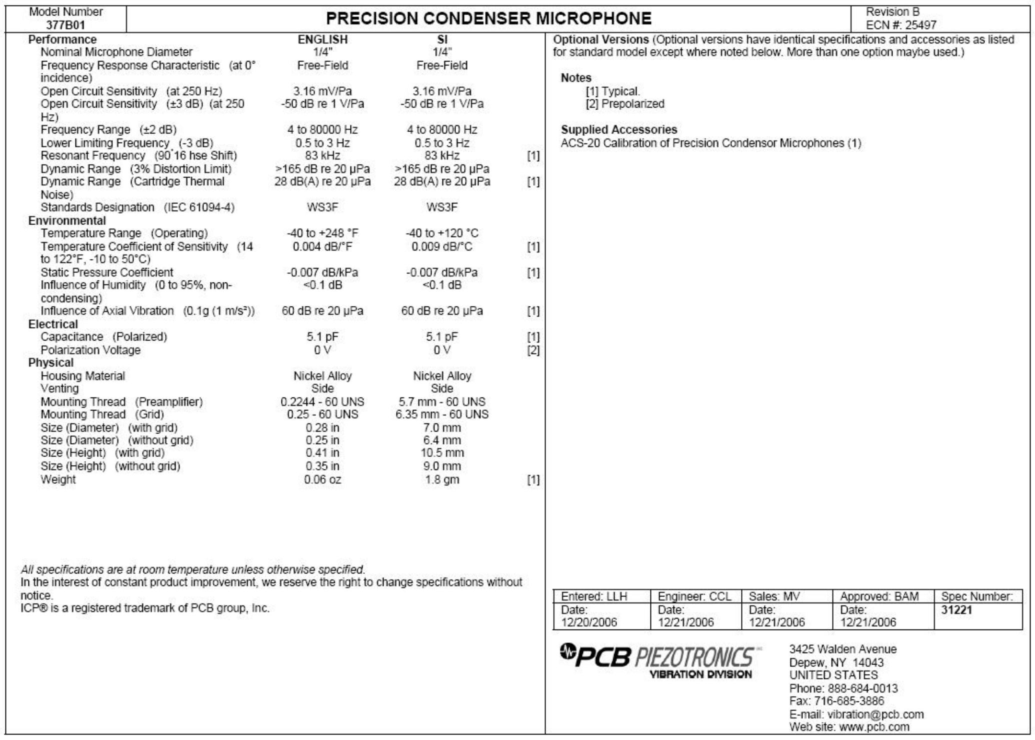


New square models shown. Legacy round speakers are also available.

Holosonic Research Labs, Inc.
www.holosonics.com
617-923-4000 info@holosonics.com

Add sound... and preserve the quiet.™

Piezotronic 377B01 Condenser Microphone



Model Number 377B01	PRECISION CONDENSER MICROPHONE			Revision B ECN #: 25497
Performance	**ENGLISH**	**SI**		
Nominal Microphone Diameter	1/4"	1/4"		
Frequency Response Characteristic (at 0° incidence)	Free-Field	Free-Field		
Open Circuit Sensitivity (at 250 Hz)	3.16 mV/Pa	3.16 mV/Pa		
Open Circuit Sensitivity (±3 dB) (at 250 Hz)	-50 dB re 1 V/Pa	-50 dB re 1 V/Pa		
Frequency Range (±2 dB)	4 to 80000 Hz	4 to 80000 Hz		
Lower Limiting Frequency (-3 dB)	0.5 to 3 Hz	0.5 to 3 Hz		
Resonant Frequency (90 16 hse Shift)	83 kHz	83 kHz	[1]	
Dynamic Range (3% Distortion Limit)	>165 dB re 20 µPa	>165 dB re 20 µPa		
Dynamic Range (Cartridge Thermal Noise)	28 dB(A) re 20 µPa	28 dB(A) re 20 µPa	[1]	
Standards Designation (IEC 61094-4)	WS3F	WS3F		
Environmental				
Temperature Range (Operating)	-40 to +248 °F	-40 to +120 °C		
Temperature Coefficient of Sensitivity (14 to 122°F, -10 to 50°C)	0.004 dB/°F	0.009 dB/°C	[1]	
Static Pressure Coefficient	-0.007 dB/kPa	-0.007 dB/kPa	[1]	
Influence of Humidity (0 to 95%, non-condensing)	<0.1 dB	<0.1 dB		
Influence of Axial Vibration (0.1g (1 m/s²))	60 dB re 20 µPa	60 dB re 20 µPa	[1]	
Electrical				
Capacitance (Polarized)	5.1 pF	5.1 pF	[1]	
Polarization Voltage	0 V	0 V	[2]	
Physical				
Housing Material	Nickel Alloy	Nickel Alloy		
Venting	Side	Side		
Mounting Thread (Preamplifier)	0.2244 - 60 UNS	5.7 mm - 60 UNS		
Mounting Thread (Grid)	0.25 - 60 UNS	6.35 mm - 60 UNS		
Size (Diameter) (with grid)	0.28 in	7.0 mm		
Size (Diameter) (without grid)	0.25 in	6.4 mm		
Size (Height) (with grid)	0.41 in	10.5 mm		
Size (Height) (without grid)	0.35 in	9.0 mm		
Weight	0.06 oz	1.8 gm	[1]	

Optional Versions (Optional versions have identical specifications and accessories as listed for standard model except where noted below. More than one option maybe used.)

Notes
[1] Typical.
[2] Prepolarized

Supplied Accessories
ACS-20 Calibration of Precision Condensor Microphones (1)

All specifications are at room temperature unless otherwise specified.
In the interest of constant product improvement, we reserve the right to change specifications without notice.
ICP® is a registered trademark of PCB group, Inc.

Entered: LLH	Engineer: CCL	Sales: MV	Approved: BAM	Spec Number:
Date: 12/20/2006	Date: 12/21/2006	Date: 12/21/2006	Date: 12/21/2006	**31221**

PCB PIEZOTRONICS
VIBRATION DIVISION

3425 Walden Avenue
Depew, NY 14043
UNITED STATES
Phone: 888-684-0013
Fax: 716-685-3886
E-mail: vibration@pcb.com
Web site: www.pcb.com