2009-04-21

Development of an Adaptive Equalization Algorithm Using Acoustic Energy Density

Panu Tapani Puikkonen
Brigham Young University - Provo

DEVELOPMENT OF AN ADAPTIVE EQUALIZATION ALGORITHM USING ACOUSTIC ENERGY DENSITY

by

Panu Puikkonen

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Physics and Astronomy

Brigham Young University

August 2009

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Panu Puikkonen

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

Date	Timothy W. Leishman, Chair
Date	Scott D. Sommerfeldt
Date	David G. Long

BRIGHAM YOUNG UNIVERSITY

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

Date

Timothy W. Leishman
Chair, Graduate Committee

Accepted for the Department

J. Ward Moody, Graduate Coordinator
Department of Physics and Astronomy

Accepted for the College

Thomas W. Sederberg, Associate Dean
College of Physical and Mathematical Sciences

ABSTRACT

DEVELOPMENT OF AN ADAPTIVE EQUALIZATION ALGORITHM USING ACOUSTIC ENERGY DENSITY

Panu Puikkonen

Department of Physics and Astronomy

Master of Science

Sound pressure equalization of audio signals using digital signal processors has been a subject of ongoing study for many years. The traditional approach is to equalize sound at a point in a listening environment, but because of its specific dependence on the room frequency response between a source and receiver position, this equalization generally causes the spectral response to worsen significantly at other locations in the room.

This work presents both a time-invariant and a time-varying implementation of an adaptive acoustic energy density equalization filter for a one-dimensional sound field. Energy density equalization addresses the aforementioned challenge and others that relate to sound equalization. The theory and real-time implementation of time-invariant sound pressure and energy density equalizers designed using the least-squares method are presented, and their performances are compared. An implementation of a time-varying energy

density equalizer is also presented.

Time-invariant equalization results based on real-time measurements in a plane-wave tube are presented. A sound pressure equalizer results in a nearly flat spectral magnitude at the point of equalization. However, it causes the frequencies corresponding to spatial nulls at that point to be undesirably boosted elsewhere in the sound field, where those nulls do not exist at the same frequencies. An energy density equalization filter identifies and compensates for all resonances and other global spectral effects of the tube and loudspeaker. It does not attempt to equalize the spatially varying frequency nulls caused by local pressure nodes at the point of equalization.

An implementation of a time-varying energy density equalizer is also presented. This method uses the filtered-x filter update to adjust the filter coefficients in real-time to adapt to changes in the sound field. Convergence of the filter over time is demonstrated as the closed end of the tube is opened, then closed once again. Thus, the research results demonstrate that an acoustic energy density filter can be used to time-adaptively equalize global spectral anomalies of a loudspeaker and a one-dimensional sound field.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The research presented in this thesis demonstrates that one can adaptively equalize the total acoustic energy density at a point in a one-dimensional sound field to produce better global equalization than one can achieve using traditional sound pressure equalization techniques. However, before the impact of this result can be fully appreciated, the reader must first come to understand the purpose of sound equalization and past work that has been conducted in the field. This chapter accordingly explains the motivations for and fundamental characteristics of equalization through a descriptive and theoretical overview. It also presents needed historical background of previous equalization research. It concludes with a discussion of research objectives followed by an overview of the structure of the thesis.

1.1 Why Equalize?

High-fidelity sound playback attempts to deliver program material from a recorded medium to a listener in its pure form. Ideally, a listener hears a recording identically to how it was originally performed. This is not possible, however, as the signal is

colored during its travel. Linear coloration can be defined as either magnitude or phase distortion of the signal [1]. As a recording is played back through a sound system, the signal is distorted by the various audio elements that form the system. Playback devices, amplifiers, and loudspeakers have nonuniform frequency responses due to their imperfect components. This causes certain frequencies to be amplified and others to be suppressed in the signal path.

Before sound reaches a listener it also travels through an acoustic space. A one-dimensional sound field in an enclosure is discussed in this study; free-field equalization was recently studied by Chester [2]. The coloration caused by the acoustic environment is often much more significant than that of the audio system components. Such an environment can be modeled as a linear system whose behavior at a particular listening position is characterized by an impulse response [3]. The shape of this impulse response is characterized in part by how rapidly sound attenuates in the enclosure, which process is dependent on the surface materials, the geometry of the space, and the source and listener positions [4]. The Fourier transform of the impulse response describes its frequency and phase content. This is called the room frequency response (RFR).

The sound arriving at a listener position can be grouped into three major categories: the direct sound, the sound reflected once from the surfaces of the enclosure boundaries, and the sound waves that have experienced two or more reflections. The reflections in the last category are responsible for the reverberant sound field [5]. These arrive later in time and tend to be more faint. Reflections cause constructive or destructive interference of sound waves due to the superposition principle.

Sound at a specific location in a room can be amplified by a global room resonance or attenuated due to a spectral null for a given frequency. Global resonances are caused by the interaction of the sound source with its environment and a null is

caused by a standing wave excited in the enclosure. Resonances may result in drastic coloration of the perceived sound. The observability of such resonances depends on the source and listener positions. If an enclosure is excited by a sinusoid with a frequency corresponding to a room resonance, and neither the source nor the listener location is near a null of a standing wave excited by that frequency, the room resonance amplifies the tone. However, these resonances are not perceived everywhere in an enclosure. A room resonance for a given frequency is not always present in the measured RFR. This is due to a null at the listener (or source) position. The RFR varies for different unique positions of the source and/or receiver.

A sound field can be passively or actively controlled in an effort to minimize the effects of coloration. For frequencies above the Schroeder frequency, which marks when a sound field becomes statistically diffuse [6], passive methods of sound field control are well established and understood. Individual reflections can be diffused by the appropriate acoustical treatments, and the sound energy can be controlled by placing absorptive materials into the enclosure. At lower frequencies, controlling the sound field with passive methods becomes much more difficult because the wavelengths are generally comparable to the dimensions of the room. In this frequency range active control becomes more feasible because specific anomalies can be more readily located in the RFR [7].

Analog equalization techniques have been used for years to try to manipulate the frequency spectrum of program material [8]. They attempt to compensate for the coloration introduced by audio and acoustic systems so that a flat or tailored frequency spectrum is perceived by the listener. Such compensation can also be accomplished with digital filters with greater accuracy and efficiency, as explained in Sec. 1.4.1.

1.2 Fundamental Inverse Filtering Theory

Before exploring the history of sound equalization, it is necessary to establish basic concepts and vocabulary. The phrase"room equalization" is often used to describe a broad category of DSP techniques that attempt to undo the physical and perceptual artifacts generated by sound reproduction inside an enclosure [9]. These artifacts are due to the imperfect mechanical loudspeaker response and the acoustic room response.

The data presented in this thesis is digitally sampled, and in the discrete time domain takes the form $a(nT)$ where a is the signal, n is the sampled number, and T is the sampling period. The variable T is known to be present, but left out. As recorded sound is being reproduced the loudspeaker and the room that it passes through to the receiver act as a filter to color the sound. This process can be described mathematically as linear convolution in the discrete time domain:

$$y(n) = x(n) * h(n). \tag{1.1}$$

Here $y(n)$ is the received sound, $x(n)$ is the original electrical input signal, and $h(n)$ is the loudspeaker/room response which takes into consideration the coloration introduced by both. This system is shown in Fig. 1.1. The effect of an enclosure on sound transmission can also be described by a linear, time-invariant filter $H(k)$ in the discrete frequency domain. Upper case letters are used to indicate values in the discrete frequency domain, and the variable k denotes a discrete frequency bin in the frequency domain. The filter $H(k)$ represents the fast Fourier transform (FFT) of the digital discrete-time impulse response $h(n)$ such that

$$H(k) = \mathrm{FFT}[h(n)]. \tag{1.2}$$

Inverse filtering attempts to recover the original signal by convolving the colored signal with the inverse RFR of the sound transmission path.

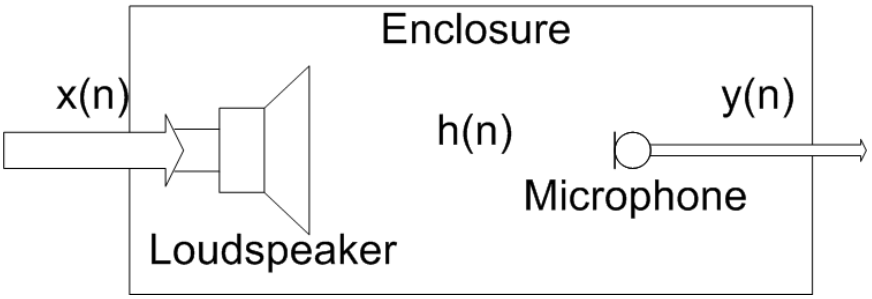


Figure 1.1 Signal path from a source to a receiver

Digital equalization commonly attempts to correct both the magnitude and phase response of the sound. While it can also attempt to compensate for nonlinear distortion, the focus here is on linear distortion [10]. For real-time applications, this process is accomplished by digitally prefiltering the audio signal with an inverse filter before it is transmitted into the enclosure. A recorded signal can also be deconvolved by its inverse filter in the post-processing stage [8, 11]. An ideal inverse filter, $H_i(z)$ where z represents the discrete z-domain, cancels the effects of the acoustic space perfectly, such that

$$h(n) * h_i(n) = \delta(n) \tag{1.3}$$

where $*$ denotes linear convolution, and

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}, \tag{1.4}$$

or

$$H(k)H_i(k) = 1, \tag{1.5}$$

where

$$H_i(k) = \frac{1}{H(k)}. \tag{1.6}$$

However, the goal of room equalization is not always to remove all reverberation from an audio signal. Doing so makes every room sound like an anechoic chamber.

Many listening areas benefit from some amount of reverberation, or ambiance, thus the preferred equalization target is to reduce reverberation in a room by shaping the transmitted sound with an appropriate inverse filter [12].

1.3 Minimum and Mixed Phase Explained

An RFR, which is a linear time-invariant system, can be expressed in the z-domain as

$$H(z) = \frac{\prod_{k=1}^{M}(1 - c_k z^{-1})}{\prod_{k=1}^{N}(1 - d_k z^{-1})}, \tag{1.7}$$

with zeros at $z = c_k$ and poles at $z = d_k$. If such a system is inverted, the poles become zeros and vice versa [13], and the inverse can be expressed as

$$H_i(z) = \frac{\prod_{k=1}^{N}(1 - d_k z^{-1})}{\prod_{k=1}^{M}(1 - c_k z^{-1})}. \tag{1.8}$$

In order for a system to be stable and causal, all poles must be inside the unit circle. When all of the poles and zeros are inside the unit circle (meaning that $|c_k|$ and $|d_k|$ are less than one) its inverse $H_i(f)$ is also stable and causal. Such a system is said to be minimum phase.

If a frequency response is not minimum phase, it can be mixed-phase or maximum-phase. A mixed-phase system exists when the criteria described above are not met and some zeros are outside the unit circle. As these zeros become poles they make the filter unstable. A simple example of such a system is a single zero filter with a coefficient of 1.5. As the system is inverted this zero becomes a pole located at $z = 1.5$ and a unit impulse causes the system to be unstable because of the feedback loop created by the pole. If all zeros are outside the unit circle the system is said to be maximum-phase. Such a system is rare and not practical to implement.

An RFR can also be expressed as

$$H(z) = |H(z)|\exp[i\phi(z)] \tag{1.9}$$

where $|H(z)|$ is its magnitude and $\phi(z)$ is its phase. Furthermore, the phase can be expanded to

$$\phi(z) = \phi_{min}(z) + \phi_{ap}(z) \tag{1.10}$$

where $\phi_{min}(z)$ describes the minimum-phase portion, and $\phi_{ap}(z)$ describes the all-pass or nonminimum-phase portion of the room response. With these definitions $H(z)$ can be expressed as

$$H(z) = H_{min}(z)H_{ap}(z) \tag{1.11}$$

where $H_{min}(z)$ is the minimum-phase component, and $H_{ap}(z)$ is the all-pass component:

$$H_{min}(z) = |H(z)|\exp[i\phi_{min}(z)] \tag{1.12}$$

and

$$H_{ap}(z) = \exp[i\phi_{ap}(z)]. \tag{1.13}$$

This can be accomplished using a mathematical method called homomorphic deconvolution [14,15]. The magnitude $|H_{ap}(z)|$ is always one. When $\phi_{ap}(z) = 0$, $H_{ap}(z) = 1$ and $H(z)$ is minimum phase [16].

If an RFR is minimum phase, an inverse filter theoretically exists and is guaranteed to be minimum phase and causal as well, such that

$$H_i(z) = H(z)^{-1}. \tag{1.14}$$

For such cases, a minimum-phase filter completely removes the room effect from a signal. However, in practice most RFRs are mixed-phase. If a mixed-phase room is equalized with a minimum-phase inverse filter the all-pass portion remains, and its

presence is often clearly audible [17–19]. This is due to the fact that the all-pass component, even though it has a flat frequency spectrum, carries most of the reverberant energy according to a listening test performed by Johansen and Rubak [1]. A minimum-phase filter has its energy concentrated toward the beginning of the time record, so enclosures with late reflections and long reverberation times cannot be completely equalized with such a method [20, 21]. The remaining room response is perceived as actual reverberation or smearing of sound [11, 22]. The research community has not yet concluded whether it is only possible to equalize the minimum-phase portion of a room response, or if the whole mixed-phase response can be equalized in practice.

1.4 Historical Background

1.4.1 In the Beginning: Analog Equalizers and DSP Technology

The use of analog filtering to improve the quality of sound reproduction has been practiced for over forty years. In the 1960s, Boner and Boner [23] used narrow band equalization to decrease the effects of resonances in rooms, and Conner [24] devised a method using 1/3 octave band measurements to determine the inverse response of a room. Their work gave a new direction for development in audio. In his paper, Conner remarks that over the preceding twenty five years the sound reinforcement equipment in use had not changed much. Over the next decade this stagnant trend began to change. 1/3 octave band equalization became common in a number of applications, and different equalization needs were recognized for sound reinforcement, monitoring, and hi-fi systems.

Equalization technology continued to improve into the 1970s, and inverse filtering a room response using DSP techniques was first seriously studied by Neely and Allen in 1979 [16]. Their study discussed the invertibility of minimum and mixed-phase RFRs. With the advent of DSP technology, equalization could then also be approached from a time-domain perspective, which was previously not feasible [5]. The coming of age of DSP technology made it possible for researchers to apply more complicated filtering schemes to attempt to overcome limitations of analog equalization, but it wasn't until the late 1980s that DSPs became powerful enough to be used in commercial products [25, 26]. Since then, digital equalization has continued to become more common in commercial technology.

1.4.2 Practical Problems

The most basic scenario for room equalization study involves one source, a loudspeaker, and one receiver, a microphone. This setup is shown in Fig. 1.1. An excitation signal is played from the source and it is received at the listening position. In practice, an inverse filter describes the sound field at one unique listening location in a room, or two according to the acoustic reciprocity principle [27]. More reciprocal pairs may exist in rooms with perfect symmetry. A frequency response for the connecting space can be computed by knowing the output and input signals [28–31]. This frequency response describes all the linear sound coloration that occurs between the source and receiver during playback, which includes both the acoustic effects and the audio system imperfections. In principle, it should be possible to compute a perfect inverse filter for a given impulse response [26].

In practice, simulations have provided near perfect results [22] and real-time implementations have resulted in significant improvement in both the time and frequency-domain responses at the equalization location [9, 32–35]. Single point equalization

begins to fail as the listener/microphone is moved away from the point of equalization. In fact, the sound quality away from the equalization point can become worse than if no equalization had been applied at all. A study highlighting this problem was recently published by Bharitkar, et. al [36].

Thus a change in the location of the source or the listener often not only eliminates the benefit of equalization, but can actually have a strong negative impact on sound quality. Changes as small as a few tenths of an acoustic wavelength can cause significant degradation in the sound field [37,38]. This can be due to the directivity of the loudspeaker system or the physical characteristics of the room [8]. Room modes cause the sound field in a room to be nonuniform; changes in the environment such as people moving, opening doors and windows, and rearranging furniture also modify the RFR [22, 39, 40]. Two popular methods used to quantify the successfulness of equalization are understanding the measure of distortion introduced as the receiver is moved and describing the size of the effective area of equalization [4, 37, 41, 42]. In 1985, Mourjopoulos proposed a method to analyze the error introduced during deconvolution using an inverse filter that created an inexact match [22]. However, even though the major problems associated with room equalization are clearly defined, no standardized method is currently established to assess the errors introduced to sound in an enclosure as a result of single point equalization.

1.4.3 Inverse Filter Design Considerations

Analog equalizers are usually graphic or parametric, shaping the spectrum using minimum-phase filters. Such filters cannot equalize a mixed-phase response, and their frequency resolution is limited in comparison to the much more complex RFR to be equalized. As a result, ideal room equalization cannot be achieved by analog methods [8]. In spite of these shortcomings, analog equalizers are used in a variety of venues

such as movie theaters, recording studios, concert halls, and music venues. Analog filters are usually made up of first and second order filters that try to match the inverse of a measured RFR steady state magnitude, and they face practical challenges such as the need to equalize over large listening areas and equalizing to nonflat frequency response curves [32].

Digital equalization has continued to be the subject of significant research in the fields of audio and acoustics since the late 1970s, and during that time many approaches have been taken to study the problem. In hindsight, the work by Neely and Allen in 1979 [16] established the nature of the equalization problem impressively well. They described how a room response can be broken down to its minimum-phase and all-pass parts. Since then the question of minimum-phase vs. mixed-phase equalization has been the subject of ongoing study. Another focus of interest for researchers is the quest for the optimal digital filter implementation. Several design methods have been presented over the years concerning filters designed either in the time or frequency domain. Each one of them tries to outperform previous methods and overcome some of the formidable challenges that have surfaced.

The most prevalent filter design technique in the time domain is called the least-squares (LS) method and the design of such filters is well documented [14, 28, 29, 31]. This method finds the best solution to the problem, in the least squares sense, and its goal is to design a finite impulse response (FIR) filter that deconvolves the room response in the time domain. Such an inverse filter is by definition causal and finite. The direct inverse of a mixed-phase room response is acausal (two-sided) and infinite in length [14]. An approximate FIR inverse filter can also be designed for such a system [14,21]; thus, a stable filter can be designed that attempts to equalize a mixed-phase RFR. The LS inverse filter design method has been used in this research, and the steps to design such a filter are detailed in Sec. 2.4.1.

Also, several frequency-domain designs exist for filters. The simplest involves inverting a complex RFR and computing the inverse FFT (IFFT) [16,43]. An auditory problem may result because a multiplication or division in the frequency domain is a circular operation in the time domain. Audible wrapping artifacts may be created as a result. Such wrap-around effects can be lessened by zero padding the signal, but not eliminated [8]. The theory of directly inverting an RFR to derive an inverse filter, including associated drawbacks, is addressed in greater detail in Sec. 2.4.2.

The types of inverse filters designed in the z-domain can be broken up into three categories: all-zero (numerator only), all-pole (denominator only), and the combination of the two, pole-zero. These filter types also have another electrical engineering nomenclature associated with them. All-pole infinite impulse response (IIR) filters are called autoregressive (AR), all-zero FIR filters are called moving average (MA), and IIR filters that are a combination of the two are called ARMA [44]. All of the aforementioned terms are used interchangeably in the literature to describe filter implementations. Many useful sources are available in the electrical engineering literature that explain FIR and IIR filter designs [28, 29, 31, 45].

Researchers have explored several different frequency-domain methods based on the principle of multi-resolution equalization, where higher frequencies are analyzed with decreasing frequency resolution to optimize the use of computing power. Some of the methods include Bark frequency scale representations [10, 46], warped-frequency FFT and Z-transform scales [10, 47, 48], the use of wavelets for time-dependent frequency warping [46, 49], nonuniform filter banks [50–53], and fractional octave transforms [54, 55].

Clearly, many different approaches have been taken to find the optimal equalization method. A summary of possible choices made when choosing a filter implementation is presented in Fig. 1.2 (see also [56]). Hatziantoniou and Mourjopoulos claimed

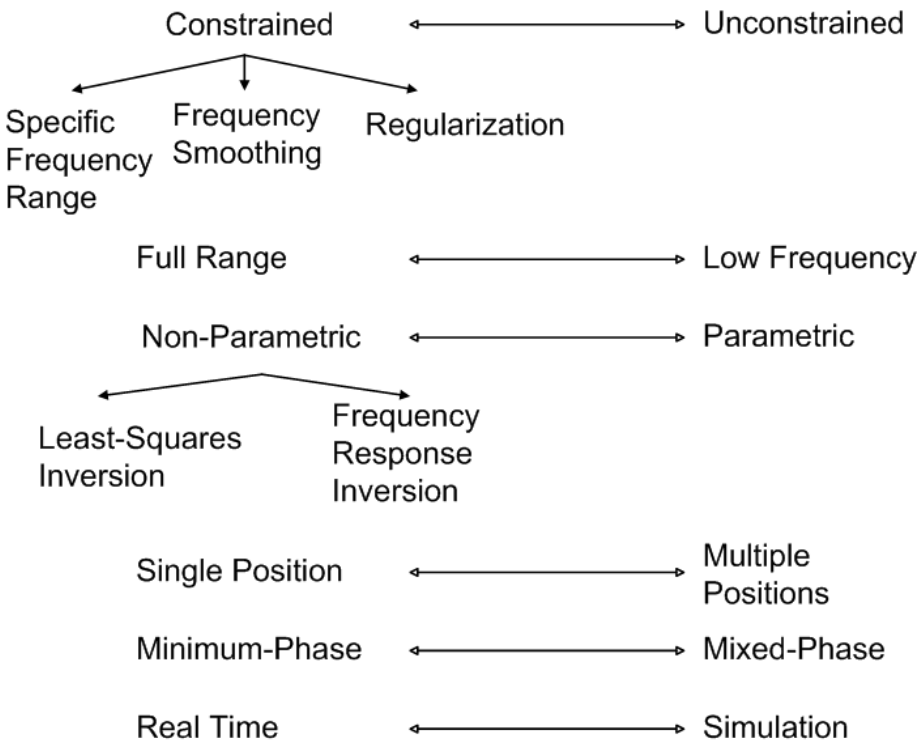


Figure 1.2 Digital filter design considerations

that none of the methods offered a generally effective solution to the equalization problem [57]. They have recently proposed a new method called complex smoothing to address issues discussed in Sec. 1.4.2. Fractional octave smoothing methods in the past have only accounted for the magnitude response with the loss of phase information. Their new method retains the phase information so that the proper impulse response can be recovered. The authors claim that the method is successful at achieving global equalization [9, 33, 34, 57, 58]. Recent studies using this same algorithm conclude that the algorithm shows promise for improved global equalization [59, 60].

1.4.4 Subjective Research

The subjective quality of equalization depends on the ability of the listener to detect nonideal characteristics in sound. The definition of ideal sound varies as people perceive and interpret sound differently. Thus the success of any particular method is not determined merely by the shape of computer generated graphs, but by whether the deconvolved signal actually sounds good to listeners. Such a result means that the space is adequately dereverberated. Some overarching themes exist that characterize deviations from the ideal sound, and their severity. The propagation of sound in rooms is strongly influenced by wave effects, such as room nulls and resonances, diffraction, and interference. The two latter effects may be even more prominent in small rooms [61]. Also, resonances are often audible after the primary sound has decayed because they are no longer masked by it.

Olive, et al. researched the detection and quantification of these resonances, building on similar work done earlier by other authors that involved studying the cases of continuous broad-band and transient sounds individually [62, 63]. The test results revealed that low Q resonances are more audible in continuous signals, and high Q resonances are more audible in transient signals. Furthermore, for low Q values, frequency notches are as audible as resonances. As the Q value is increased notches become less and less audible in broad band signals. Thus high Q notches in the inverse filter spectrum do not degrade the sound as much as high Q resonances when a listener moves around an equalized space. The duration of reverberation is an unreliable indicator for determining whether resonances are audible. Instead, their audibility is based more on the frequency-domain characteristics of the sound. The most significant result of the work by Toole and Olive is that in broad-band signals, resonances are more audible than notches. Thus equalization should focus more on the suppression of spectral peaks [64, 65].

A single reflection can also be audible and annoying. The audibility of such a reflection depends on its level, delay time, direction, and signal type. Echos are perceived for reflections wherein the delay time exceeds 30 ms [12]. An RFR describes the level and frequency of post-echos present in the perceived sound. An undesired time-domain effect that makes equalization difficult is generated by the actual inverse filter itself. As a noncausal inverse filter is created either in the time or frequency domain, pre-echoes are created due to inversion error which is a mismatch between the room response and the inverse filter. This type of echo is introduced when a modeling delay is added to a noncausal filter response, estimating a causal filter. This effect can also be present in minimum-phase filters [32, 34].

Several listening tests have been performed by different researchers, but no universal method has been established to quantify the audible effectiveness of equalization. Three such listening tests [8, 10, 58] are presented here to demonstrate what types of methods have been used to analyze the effectiveness of the physical implementation of inverse filters. Because the end goal of equalization research is to create a successful product, it is crucial that accurate listening tests are performed to evaluate whether a given method is truly effective.

Worley, et al.

The first study [58] compared a pure reference signal with a signal filtered with an anechoic loudspeaker and a complex smoothing inverse filter. The anechoic loudspeaker filter attempts to only equalize the loudspeaker response, thus leaving the room effects untouched while complex smoothing attempts to equalize the whole RFR. Both equalization methods were also evaluated for their effectiveness when listener positions were changed. Eight people participated in this listening test, and the tests were administered in different rooms where the subject would hear the sound

from a pair of loudspeakers. The subjects reported that in general they perceived a room to be smaller after the signal was filtered with complex smoothing, and that they did not notice a deterioration in sound as they moved off axis from the speaker to different locations in a room.

The conclusion that the listeners could move around the room without noticing a deterioration in sound quality is significant because no previous study has made such a bold claim. The anechoic loudspeaker was found to be inferior in all cases, and no attention was drawn to what the loudspeaker filter was able to do well. This test demonstrates that the room response cannot be ignored, and that neither filter works well at equalizing low frequencies.

Norcross, et al.

In the second case [8], two methods, time-domain LS [66] and frequency deconvolution [67], were compared for the application of loudspeaker response equalization. This process is identical to loudspeaker/room response equalization in its method. Measurements of a test signal (a maximum-length sequence) on-axis and 45 degrees off-axis were taken in an anechoic chamber with an omnidirectional microphone. The test audio sample was prefiltered off-line with the derived impulse responses from these measurements and then with the derived inverse filters. The chosen test method was to playback the audio samples via headphones. Before engaging in the actual experiment the subjects completed a training session where the procedure of recording their observations was explained to them.

The results of this study showed that the subjective performance of both filters was highly dependent on the impulse response being inverted and sometimes, especially with the frequency-domain method, the inverse filtered audio sounded worse. The inverse filters were not successful in improving sound quality off-axis in any of the

performed tests, and introducing a modeling delay to a minimum-phase LS inverse filter degraded the quality of the filter. A frequency spectrum shaping method called regularization [67] was also tested with the frequency-domain filter, and the amount of regularization needed for acceptable equalization level varied greatly for different impulse responses. Sometimes regularization also lowered the audio quality. Complex smoothing was also applied to both filter methods, as described by Mourjopoulos and Hatziantoniou [57]. This method improved performance in most cases, and never degraded it. Complex smoothing created much better results than regularization.

Karjalainen, et al.

The third study [10] also focused on the equalization of a loudspeaker response. Seven test subjects participated. The listener would sit in an anechoic chamber and listen to sound samples from a loudspeaker that consisted of the original signal prefiltered by one of the following filters: an FIR, a WFIR (warped), or a WIIR filter. Each filter was implemented with five different lengths of coefficients. As expected, the higher the filter order, the better the equalization. The results showed that the FIR equalizer is the most computationally efficient, WFIR is 2.5 to 3.5 times slower, and the WIIR is about 2 times slower. The subjective listening results varied greatly for each combination of filter design method and filter order. The authors concluded that a better testing procedure should be used, or that the listening tests should be more carefully conducted to produce accurate results. One possible explanation for the large variability in the results was an "inhomogeneous listening panel" with varying amounts of listener training. The authors also suggested that an important new direction would be to develop computational models that correspond to the human auditory system.

Summary

Clearly, an experiment comparing the effectiveness of all possible inverse filters will never take place due to the great number of possible implementations. Listening tests can be administered in many different ways, such as via loudspeakers in a room, in an anechoic chamber, or using headphones where the signal is prefiltered to simulate a specific room. The quality of the experimental results depends on the the participating listeners' experience; the more nonhomogeneous the group, the more general and descriptive the results. At times, drastic discrepancies arise between theoretical results and the actual effectiveness of a filter. Thus computer-simulated results do not guarantee that a given method successfully equalizes a sound field. Listening tests are another aspect of room equalization where the best method has not been discovered yet, and researchers continue to look for new and more accurate ways to evaluate filter performance.

1.4.5 Extended Applications

Equalization research has primarily focused on the simplified case of a single source and a single receiver. Such a setup is appealing because it is theoretically and practically the most straightforward to analyze, but most real world systems are more complex. In fact, little of the music listened to comes from a single source to a single listener. As stated earlier, it may be possible to equalize sound at a point in a room, but such a method is not effective in improving the listening experience in concert halls, living rooms, offices, cars, parks etc. Because of this discrepancy, many researchers have explored solutions to these more challenging real-world problems. Some of the more popular explorations include the equalization of sound from a stereo source [66,68–73], greater global control in rooms by using multiple measure-

ment microphones [3, 25, 36, 73, 74], sound in cars [35, 75], loudspeaker equalization, and modal equalization.

Stereo Sound

For years, stereo sound has been the most popular way to listen to music. This includes headphones, computer speakers, television, boom boxes etc., and comes with its own complex equalization challenges. The goal for optimal stereo listening is to reproduce two sound recordings from two separate locations identically at two points in the listening area, usually at human ears [72]. The most prevalent problem in stereo equalization is acoustic crosstalk. An ear does not only receive the desired signal from the appropriate source (i.e. the left ear hears the left loudspeaker), but also a contribution from the other loudspeaker. The sound diffracts around the head, and thus the crosstalk is frequency dependent; its negative effect diminishes with increased frequency [68]. Much of the work focusing on digital equalization of stereophonic sounds has been theoretical, with a focus on the mathematical derivation of the filter, and thus the psychoacoustic effects have not been considered to a great extent. However, Nelson et al. [72] did note that Schroeder successfully implemented analog acoustic crosstalk cancelation [76] in the 1970s, which serves as motivation for current work in the digital domain.

Cars

Cars are another specialized focus of sound equalization. As multiple loudspeaker sound systems are becoming more common in high-end cars, an interest in the acoustics of car interiors continues to increase. Cars suffer from acoustic interference and resonances in a similar way that rooms do. This creates a specific need for equalization of low frequency resonances for smooth sound [35].

Multiple Microphones

As room equalization research progresses, a natural transition is to attempt to equalize sound for more listeners and a larger listening area. Several impulse responses are measured at different locations, and the corresponding magnitude RFRs are averaged [3]. This results in a magnitude-only frequency spectrum. The minimum-phase response is then computed and inverted to create an inverse filter [74]. Bharitkar et al. performed a listening test simulation with such a system and they found that the size of the equalization region varied depending on the distance of the listener from the source and frequency; it grew greater with greater distance and with lower frequency.

In 1989, Elliott and Nelson presented an equalization method where the target was to minimize the sum of the mean-square errors between various measurement points in an enclosure [25]. They detected some improvement at each of the four locations used in the process, although differences persisted. A later study [73] showed some improvement in the RFR smoothing, but the multiple-microphone method is not able to equalize as successfully at any point as when minimizing the squared error at only one point. Both studies implemented a form of the filtered-x LMS algorithm [77], which has its primary application in active noise control (ANC).

Loudspeaker Equalization

Equalization techniques similar to those presented here have also been applied to loudspeaker response-only equalization in a free field environment where relevant concepts such as off-axis behavior of an equalized loudspeaker and phase equalization can be studied. An inverse filter derived for the on-axis response of a loudspeaker does not perform as well off-axis [10]. Not only does the room response change as the listener moves away from the point of equalization, but the speaker response changes

as well. This further complicates the problem.

Modal Equalization

A less common method known as modal decay equalization has also been researched as an alternative method to traditional room equalization techniques. Room equalization attempts to completely remove the room response, where modal decay equalization attempts to reduce reverberation time by controlling the decay lengths of specific low-frequency modes. By shortening the decay times of the modes with very long decay to the range of mid and high frequencies the modes become less audible [7].

These examples have been presented to demonstrate the variety of applications where digital sound equalization has been applied, and to illustrate that different equalization applications provide their own unique challenges. These challenges will not be overcome until the simple case of single source/single receiver equalization is better understood.

1.4.6 Current Events

The study of room equalization or dereverberation continues to attract attention in the fields of acoustics, audio, and engineering. A recent paper in 2004 by Hatziantoniou and Mourjopoulos reflects well the current state of this research [9]. The paper focuses on the inversion of a mixed-phase room response. Hatziantoniou states that the early studies introduced more than twenty years ago highlighted several important theoretical and practical problems. These include the aforementioned challenges of a room having a noncausal, mixed-phase RFR, large variations in the RFR between different source-listener locations in an acoustic space, the long filter lengths required to accurately model an RFR with an FIR filter, and the instability

of IIR filters. Their work presents a new approach to room equalization, which highlights the ongoing trend that even though many different methods have been studied in search of a solution to the room response deconvolution problem, no best method has been found.

1.5 Purpose of Current Research

1.5.1 Shortfalls of Past Work

Effective room equalization has many limitations and constraints set by principles of room acoustics, psychoacoustics, and signal processing [56]. According to Neely and Allen, a true causal inverse of an RFR does not exist in general because most RFRs are mixed-phase. From an engineering perspective this means that not all of the poles and zeros of the Laplace transform of the system lie in the left-hand plane, or in the case of the Z-transform, are not within the unit circle [16]. In other words, an exact inverse filter does not exist for many real-world systems. Thus the research has been split between the two choices, developing a mixed-phase, noncausal filter which can be unstable but attempts to equalize all of the room response, or a minimum-phase filter which is stable, but only attempts to equalize the frequency magnitude and leaves the all-pass portion of the frequency response unaccounted for [59].

It has been shown that the mixed-phase component of an RFR is audible and cannot be ignored if optimal equalization is desired [1]. Satisfactory equalization of an RFR can be achieved at a point in a room, but because of the dependence of the RFR on the source and receiver positions, this equalization generally causes the spectral response to worsen significantly at other locations in the room [3, 8, 32]. This places limitations on how large the listening area can be. A typical method to increase the size of a listening area is to use multiple-point equalization. This

method can be impractical to implement due to the hardware requirements, and the resulting equalization is more inaccurate at any specific listening location because of the spatial averaging. This method looks for an average inverse filter response for a room which may not be the direct inverse for any point in the room. The tradeoff is accuracy vs. size of the listening zone; as the optimal listening zone grows the quality of equalization decreases.

Yet another challenge arises in time-adaptive equalization. The sound field in the listening area can change due to variations in temperature gradients, the number of listeners, listener positions, and other factors over time. Time-adaptive equalization requires a sensor, usually a microphone, to be in the field being equalized. For equalization of high frequencies, sensors need to be very close to the ear because of the small zone of equalization. To meet this requirement, sensors would need to be mounted in locations close to the head of a listener, such as in the head rest of a couch or car seat, and this is often impractical [78]. A method that successfully improves sound quality over a large listening area would overcome this and many of the other challenges facing equalization today.

1.5.2 Needs of the Industry

Audio products that implement some form of room equalization technology have been available on the market since the early 1990s. One of the earliest adaptive implementations was a product from B&W loudspeakers developed by Craven and Gerzon [26]. Their product, and others available today, implement a variation of the methods previously described and suffer from similar shortcomings.

A demand for a better equalization method exists in the audio industry. Once technology surfaces that is able to overcome many or all of the aforementioned shortcomings it will immediately find application in such areas as home stereo listening,

multi-channel sound such as home theater systems, professional sound systems, cars, and many other applications where audio systems are used.

1.5.3 Research Objectives

The objective of this research is to develop a time-adaptive equalization algorithm using energy density that works in real-time in a one-dimensional environment. This consists of the following intermediary goals:

- Conduct an in-depth literature search of research done in the field in the past.
- Choose an equalizer design method and develop a proof of concept in MATLAB.
- Take an existing DSP RTOS system and adapt it to perform equalization.
- Assemble a hardware setup that interacts with the DSP and the physical system.
- Implement an adaptive equalization inverse filter on the DSP that equalizes acoustic energy density in a one-dimensional enclosure.
- Implement a time-adaptive equalization filter on the DSP that equalizes acoustic energy density in a one-dimensional enclosure.

1.5.4 Plan of Development

This chapter presents a theoretical overview of the equalization challenge and a historical overview of previous work done in the field of sound equalization. Chapter 2 discusses the theory behind inverse equalization in detail. First, the physical quantities of sound pressure, particle velocity, and energy density are presented, as well as their application to the sound equalization problem. The theory of convolution is presented and two filter design methods are discussed along with their pros and cons.

Much attention is devoted to the LS equalization filter design method as it has the desirable quality of generating a minimum-phase energy density inverse filter. The chapter concludes with a section that describes time-adaptive filters for both sound pressure and energy density equalization.

Chapter 3 describes the hardware setup used to conduct the experiments. This includes the DSP system and the audio rack unit used for input and output (IO) control. The functionality of the hardware and signal flow are explained. This is followed by a description of the time-adaptive equalization experiments.

Chapter 4 presents the measurement results. First, the ability of a pressure filter to equalize the sound field in a plane-wave tube is discussed, and graphs are used to illustrate that such a filter is able to only equalize sound pressure at a point in the field. Next, the results of energy density equalization in the same tube are presented. It is shown that an energy density inverse filter is able to detect and equalize all resonances in the sound field. This is possible because the total energy in a one-dimensional sound field is approximately uniform, thus the the energy density spectrum is nearly uniform throughout the sound field. Lastly, the ability of an energy density inverse filter to adapt to time-varying conditions via the filtered-x algorithm update scheme is demonstrated. Chapter 5 provides a summary of the research results and concluding remarks.

Chapter 2

Theory

This chapter explains the theory pertaining to the gathering and analysis of data presented in this research. First, the theory of sound pressure, particle velocity, and energy density is presented. Then linear and circular convolutions are explained with their application to filter design and an example that clarifies their fundamental differences. Next, a discussion and examples of the direct inverse and least squares inverse filter design methods are presented. The latter was used to obtain the time-invariant experimental results. This chapter concludes with a description of time-adaptive sound equalization algorithms.

2.1 Physical Quantities: Sound Pressure, Particle Velocity, and Energy Density

2.1.1 Sound Pressure

The theory relating to sound pressure, particle velocity, and energy density assumes that air behaves as a homogeneous, isotropic fluid wherein the speed of sound

c is constant. Pierce states that air in its ambient state within an enclosure can be characterized by two important properties: pressure and density (p_0 and ρ_0). An acoustic disturbance, a "small-amplitude perturbation to an ambient state," generates variations in the air density and pressure [79], such that the total pressure and density become

$$p_t(\vec{r},t) = p_0 + p(\vec{r},t) \tag{2.1}$$

and

$$\rho_t(\vec{r},t) = \rho_0 + \rho(\vec{r},t), \tag{2.2}$$

where $p(\vec{r},t)$ and $\rho(\vec{r},t)$ are the pressure and density variations introduced by the acoustic disturbance as functions of both position $\vec{r}$ and time t. Both the acoustic pressure and density satisfy the linear wave equation

$$\nabla^2 p(\vec{r},t) - \frac{1}{c^2}\frac{\partial^2 p(\vec{r},t)}{\partial t^2} = 0, \tag{2.3}$$

shown here with the dependent variable $p(\vec{r},t)$. Under time-harmonic conditions, such that

$$p(\vec{r},t) = Re\{\hat{p}(\vec{r},\omega)e^{j\omega t}\} = Re\{p_{pk}(\omega)e^{j\phi_p(\omega)}e^{j\omega t}\}, \tag{2.4}$$

where p_{pk} is the peak amplitude and ϕ_p is the phase of the pressure, the linear wave equation reduces to the Helmholtz equation:

$$\nabla^2 \hat{p}(\vec{r},\omega) + k^2 \hat{p}(\vec{r},\omega) = 0. \tag{2.5}$$

2.1.2 Particle Velocity

Sound pressure and particle velocity are related by the linearized Euler's equation [27, 80]

$$\rho_0 \frac{\partial \vec{u}(\vec{r},t)}{\partial t} = -\nabla p(\vec{r},t). \tag{2.6}$$

Under time-harmonic conditions,

$$\vec{u}(\vec{r},t) = Re\{\hat{\vec{u}}(\vec{r},\omega)e^{j\omega t}\} = Re\{\vec{u}_{pk}(\omega)e^{j\phi_u(\omega)}e^{j\omega t}\}. \tag{2.7}$$

In this case, Euler's equation becomes

$$j\omega\rho_0\hat{\vec{u}}(\vec{r},\omega) = -\nabla\hat{p}(\vec{r},\omega), \tag{2.8}$$

or

$$\hat{\vec{u}}(\vec{r},\omega) = \frac{-1}{j\omega\rho_o}\nabla\hat{p}(\vec{r},\omega). \tag{2.9}$$

Particle velocity at a position in an enclosure can be estimated from sound pressure measurements. In a one-dimensional sound field with the independent spatial variable x, the gradient becomes $\frac{d}{dx}$ that can be estimated with a linear approximation method. Two pressure measurements,

$$\hat{p}_1 = \hat{p}(x_1,\omega) \tag{2.10}$$

and

$$\hat{p}_2 = \hat{p}(x_2,\omega) \tag{2.11}$$

are required from adjacent microphones separated by a distance

$$\Delta x = x_2 - x_1. \tag{2.12}$$

This method could also be referred to simply as the point-slope method, as the gradient is found by a straightforward rise over run computation:

$$\frac{d\hat{p}(x,\omega)}{dx} \approx \frac{\hat{p}(x_2,\omega) - \hat{p}(x_1,\omega)}{x_2 - x_1} = \frac{\hat{p}_2 - \hat{p}_1}{\Delta x}. \tag{2.13}$$

Substituting Eq. (2.13) into Eq. (2.9) then yields

$$\hat{\vec{u}}(x,\omega) \approx \frac{j}{\omega\rho_o}\Big(\frac{\hat{p}_2 - \hat{p}_1}{\Delta x}\Big)\vec{e}_x. \tag{2.14}$$

This expression is an estimate of the complex particle velocity amplitude at a point

$$x = x_1 + \frac{\Delta x}{2}, \tag{2.15}$$

halfway between the two measurement positions. The method of deriving particle velocity presented here can also be used in a three-dimensional sound field using three orthogonally arranged pairs of microphones. Since in Cartesian coordinates,

$$\nabla \hat{p}(\vec{r}) = \frac{\partial \hat{p}(\vec{r},\omega)}{\partial x}\vec{e}_x + \frac{\partial \hat{p}(\vec{r},\omega)}{\partial y}\vec{e}_y + \frac{\partial \hat{p}(\vec{r},\omega)}{\partial z}\vec{e}_z, \tag{2.16}$$

Eq. (2.14) can be adapted to compute the three directional components of particle velocity, $\hat{u}_x, \hat{u}_y$, and $\hat{u}_z$, such that

$$\hat{\vec{u}}(\vec{r}) = \frac{j}{\omega\rho_o}\left\{\left[\frac{\hat{p}(x_2,\omega)-\hat{p}(x_1,\omega)}{x_2-x_1}\vec{e}_x\right] + \left[\frac{\hat{p}(y_2,\omega)-\hat{p}(y_1,\omega)}{y_2-y_1}\vec{e}_y\right] + \left[\frac{\hat{p}(z_2,\omega)-\hat{p}(z_1,\omega)}{z_2-z_1}\vec{e}_z\right]\right\}, \tag{2.17}$$

where

$$\vec{r} = (x_1 + \frac{\Delta x}{2})\vec{e}_x + (y_1 + \frac{\Delta y}{2})\vec{e}_y + (z_1 + \frac{\Delta z}{2})\vec{e}_z. \tag{2.18}$$

2.1.3 Energy Density

A sound source must deliver energy into the air in order to generate sound waves. Beranek defines the energy density in a gas as "the sound energy in a given infinitesimal part of the gas divided by the volume of that part of the gas," in joules per cubic meter [81].

As with mechanical energy, the total energy density consists of a sum of potential and kinetic energy densities. The kinetic energy density comes from the motion of air particles due to the wave, while the potential energy density is stored in air compression [82]. Pierce defines the instantaneous acoustic kinetic energy density as [83]

$$w_k = \frac{1}{2}\rho_0 u^2(\vec{r},t), \tag{2.19}$$

where $u^2(\vec{r},t)$ is the squared vector magnitude of particle velocity

$$u^2(\vec{r},t) = |\vec{u}(\vec{r},t)|^2. \tag{2.20}$$

He defines the instantaneous acoustic potential-energy density as

$$w_p(\vec{r},t) = \frac{1}{2}\frac{p^2(\vec{r},t)}{\rho_0 c^2}. \tag{2.21}$$

The total energy density is then the sum of the two components:

$$w_t(\vec{r},t) = w_p(\vec{r},t) + w_k(\vec{r},t). \tag{2.22}$$

All three quantities are real valued. As instantaneous particle velocity and sound pressure are functions of position and time, the instantaneous energy density cannot be assumed to be uniform throughout an enclosure. However, the time average of Eq. (2.22), computed over an integral number of half periods or an interminably long time period, gives the time-averaged energy density at any point in an enclosure as a time independent expression [84],

$$\langle w_t(\vec{r},t)\rangle_t = \langle w_p(\vec{r},t)\rangle_t + \langle w_k(\vec{r},t)\rangle_t \tag{2.23}$$

or

$$\langle w_t(\vec{r},t)\rangle_t = \frac{1}{4}\frac{|\hat{p}(\vec{r})|^2}{\rho_0 c^2} + \frac{\rho_0}{4}\hat{\vec{u}}(\vec{r})\cdot\hat{\vec{u}}^*(\vec{r}), \tag{2.24}$$

where * denotes a complex conjugate (see [85]). The last term in this expression forms the squared vector magnitude and complex modulus of the complex particle velocity amplitude.

2.2 Hilbert Transform

This section outlines the process of computing the minimum-phase portion of an impulse response or a mixed-phase frequency response. A relationship exists between

the real and imaginary parts of the FFT of a causal sequence, but the same cannot be said about the relationship between its magnitude and phase [86]. In fact, a given magnitude spectrum can be represented by a number of different phase spectra. A special case is the minimum-phase frequency spectrum. In this case, a unique Hilbert transform relationship exists between the magnitude and phase of the spectrum. If either the magnitude or phase is known the other can be approximated [87]. The minimum-phase representation of any complex frequency spectrum can be determined by forcing its complex cepstrum to be causal such that $x_c[k] = 0$ for $n < 0$. The cepstrum is indicated by the subscript c [88].

The process begins by computing the FFT of the impulse response:

$$H(k) = \mathrm{FFT}[h(n)]. \tag{2.25}$$

The natural log of the frequency response magnitude is then computed as

$$A(k) = \ln |H(k)|. \tag{2.26}$$

This result is then used to compute the minimum-phase response of the frequency spectrum $H(k)$. The cepstrum of the signal is found by computing the IFFT of $A(k)$:

$$A_c(k) = \mathrm{IFFT}[A(k)]. \tag{2.27}$$

The minimum-phase frequency response by definition has no negative frequency information in its cepstrum [89]. The frequency response $H(k)$ can be forced to be minimum-phase by zeroing the negative frequencies of $A_c(k)$, a periodic function, as follows [90]:

$$M_c(k) = \begin{cases} A_c(k), & k = 0, N/2 \\ 2A_c(k), & 1 \leq k < N/2 \\ 0, & N/2 < k \leq N-1. \end{cases} . \tag{2.28}$$

Here $M_c(k)$ represents the modified cepstrum. This result is then transformed back into the frequency domain by computing the reverse operations of equations 2.25 and 2.26 as follows,

$$M(k) = e^{\mathrm{FFT}[M_c(k)]}. \tag{2.29}$$

The frequency response $M(k)$ is the minimum-phase equivalent of $|H(k)|$. A minimum-phase impulse response can then be generated by computing the IFFT of $M(k)$. This same method can also be applied to a digital filter, either in the time or frequency domain, to compute its minimum-phase response. A minimum-phase impulse response has the shortest possible time response for a given frequency spectrum; it thus has minimum energy delay [91]. Minimum energy delay is a desirable quality for real-time filtering because a filter is able to respond to variations in input more quickly as the length of the effective response decreases. This quality also minimizes the rise and fall times of the filter performance.

2.2.1 Equalizing Energy Density

A drawback of computing the minimum-phase of a frequency response as discussed in Sec. 1.3 is that the nonminimum-phase portion of the frequency response is ignored. This is not a problem when designing an energy density inverse filter because energy density is derived as a squared magnitude quantity and the resulting frequency response contains no phase information. An energy density spectrum with a bandwidth of 2 kHz, computed from experimental measurements in a plane-wave tube (discussed in Sec. 3.2), is shown in Fig. 2.1 (c). The IFFT of this zero-phase frequency response is shown in Fig. 2.1 (a). This is a mixed-phase impulse response, and is nonideal for linear filtering because the nonzero values before the peak (direct sound) may cause undesirable pre-ringing [32]. The minimum-phase impulse response computed using the Hilbert transform introduced in Sec. 2.2 is shown in Fig. 2.1 (b).

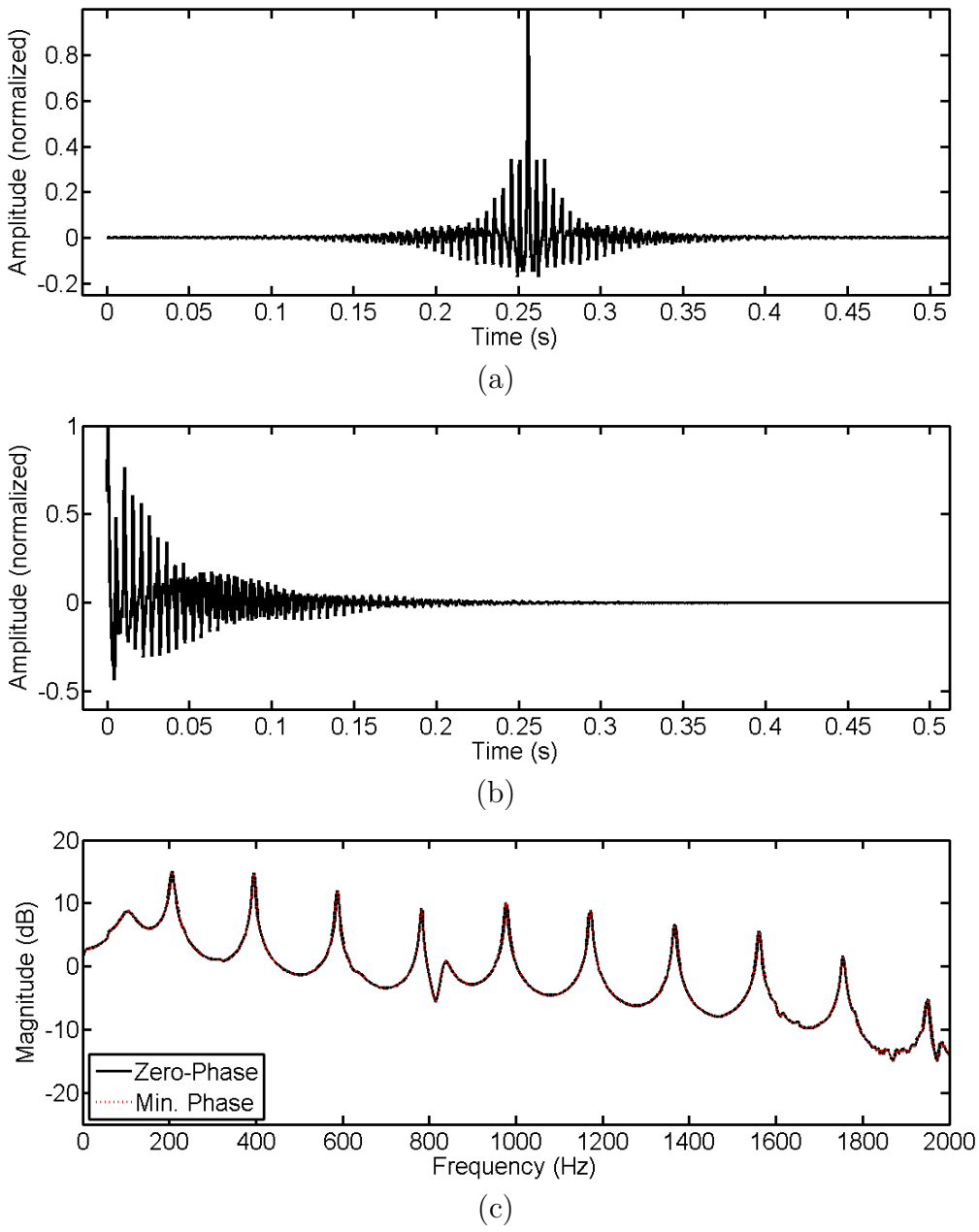


Figure 2.1 An impulse response before and after the Hilbert transform is used to find its minimum-phase response. (a) Zero-phase. (b) Minimum-phase. (c) Their frequency magnitudes.

The time signal in Fig. 2.1 (b), or the minimum-phase frequency response shown in 2.1 (c) can be used to design an inverse filter with any of the conventional design methods using an impulse or frequency response, as discussed in Sec. 1.4.3. Thus, once the energy density frequency response has been computed, an energy density inverse filter can be designed as easily as a sound pressure inverse filter.

2.3 Convolution

The discrete convolution sum has many applications in engineering and physics. It is defined as

$$y(n) = \sum_{l=-\infty}^{\infty} h(l)x(n-l), \tag{2.30}$$

where $h(n)$ and $x(n)$ are the two signals being convolved, $y(n)$ is the output, and l is a secondary discrete-time variable or position indicator. These limits of the summation can be modified because both the input and output of the filters used in this research are causal. By definition, the response cannot begin before the system receives an input. Thus, the convolution sum can be expressed as

$$y(n) = \sum_{l=0}^{N-1} h(l)x(n-l), \tag{2.31}$$

where N is the length of $h(n)$, the convolution filter, in samples.

Linear Convolution

For linear convolution the sequences $h(n)$ and $x(n)$ can be of different lengths, N_h and N_x, respectively, but they must be sampled at the same sampling rate. The convolution may then be written as

$$y(n) = x(n) * h(n) = \sum_{l=0}^{N-1} h(l)x(n-l), \tag{2.32}$$

where the length of $y(n)$ is $N_h + N_x - 1$ and $*$ denotes the linear convolution process. This can be computed in real-time, one sample at a time.

Circular (Periodic) Convolution

Circular convolution makes the assumption that the two signals being convolved are periodic. It is performed by block processing in the frequency domain. When a full buffer of data is available, the circular convolution is computed by multiplying the FFTs of the two signals together and computing the IFFT,

$$x(n) \otimes h(n) = \text{IFFT}\big\{\text{FFT}[x(n)]\text{FFT}[h(n)]\big\}, \tag{2.33}$$

where $\otimes$ denotes circular convolution and the two signals $h(n)$ and $x(n)$ are assumed to have the same length N.

2.3.1 When Circular Convolution Equals Linear Convolution

For very long sequences it is more computationally efficient to perform the convolution in the frequency domain than in the time domain. For two blocks of data of length N, the number of multiplications to perform circular convolution in the frequency domain is $2\big[2N\log_2 N\big] + N$, while for time-domain linear convolution the number is $2N^2 - N$ [92]. Under special circumstances, the circular convolution can have a result equal to that of a linear convolution. The two major differences between the two convolutions are (1) that circular convolution assumes periodicity of the two signals, and (2) that the length of the result of a circular convolution is $N = N_h = N_x$, whereas that of the linear convolution is $N_h + N_x - 1$, where N_h and N_x can be different.

To address these two issues, the signals $h(n)$ and $x(n)$ are zero padded for circular convolution to be the length of the desired linear convolution result, $N_h + N_x - 1$. This

is accomplished by adding $N_x - 1$ zeros to $h(n)$, and $N_h - 1$ zeros to $x(n)$. Then the FFTs of these two sequences are computed and multiplied together, and the resulting IFFT gives the convolution result, which is also $N_h + N_x - 1$ samples long. This can be expressed as

$$h_{N_h+N_x-1}(n) \otimes x_{N_h+N_x-1}(n) = \text{IFFT}\big[\text{FFT}[h(n)]\text{FFT}[x(n)]\big] = y_{N_h+N_x-1}(n), \quad (2.34)$$

which yields the same result as the linear convolution

$$h_{N_h}(n) * x_{N_x}(n) = y_{N_h+N_x-1}(n). \quad (2.35)$$

Figure 2.2 shows the sequential steps involved in the linear convolution of two arbitrary sequences: $h = [2 \quad 4]$ and $x = [3 \quad 2 \quad 1]$. According to Eq. (2.32) the sequence $x(n)$ is flipped across the point $n = 0$, and is then shifted to the right one sample at a time. The left column of the figure shows the shifting of x by one sample per plot, and the right column shows the corresponding output. The sum of the product of the two overlapping sequences is computed each iteration; it is the result for that iteration of the convolution. This shifting continues to produce a result every iteration as long as the two sequences overlap, and the length of the convolution result is $N_h + N_x - 1 = 4$. This is shown in Fig. 2.2 (d). The same result can be generated by applying Eq. (2.34) to the zero-padded sequences $h = [2 \quad 4 \quad 0 \quad 0]$ and $x = [3 \quad 2 \quad 1 \quad 0]$.

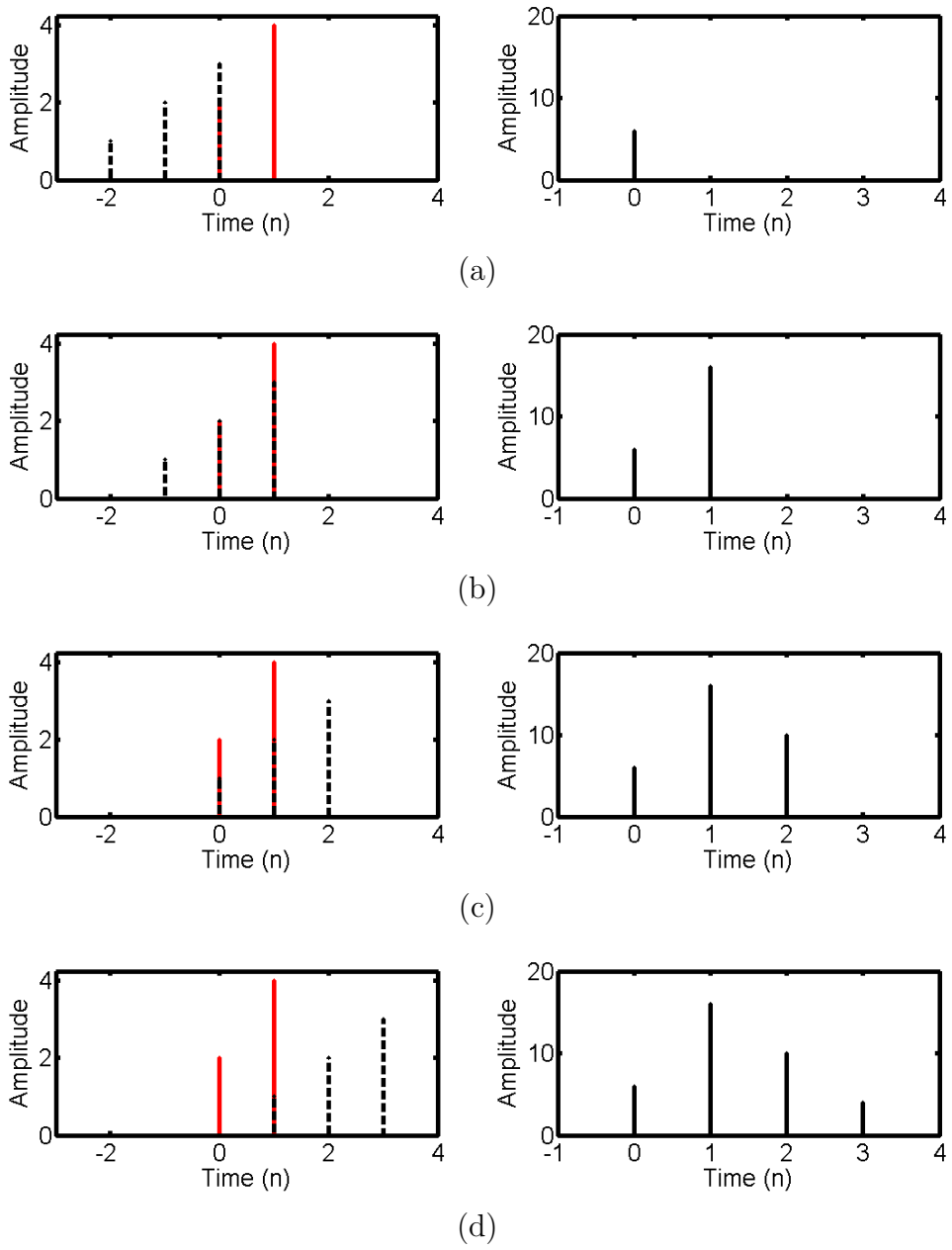


Figure 2.2 Linear convolution: the left column shows the shift of the black sequence, $x(n)$, over the red sequence, $h(n)$, one sample at a time. The right column shows the cumulative convolution output. The steps are denoted by plots (a) through (d).

2.4 Methods of Inverse Filter Design

2.4.1 FIR Least-Squares Inverse Filter Design

Theory

Linear convolution is the preferred choice for performing real-time time-domain filtering of a signal. The least squares criterion can be used to optimize a set of FIR filter coefficients to minimize the error between the desired output $d(n)$ and the actual output $y(n)$ of a system. Because an FIR filter is used, the total error is estimated and the accuracy of the estimate is dependent on the length of the filter N:

$$e(n) = d(n) - \sum_{l=0}^{N-1} h(l)x(n-l), \tag{2.36}$$

where $x(n)$ represents the input sequence, and $h(n)$ represents the filter coefficients. The sum of the squares of the error is expressed as

$$\epsilon = \sum_{n=0}^{\infty} \left[d(n) - \sum_{l=0}^{N-1} h(l)x(n-l) \right]^2 \tag{2.37}$$

for a discrete-time signal $x(n)$ of infinite length. When the input and the desired output signals are known a solution can be found for $h(n)$ that minimizes the squared error by solving the following system of linear equations:

$$\sum_{n=0}^{N-1} h(n)r_{xx}(n-l) = r_{dx}(l); \qquad l = 0, 1..., N-1, \tag{2.38}$$

where $r_{xx}(l)$ is the autocorrelation function of the input sequence, which is defined as

$$r_{xx}(l) = \sum_{n=0}^{\infty} x(n)x(n-l), \tag{2.39}$$

and $r_{dx}(l)$ is the cross correlation between $d(n)$ and $x(n)$ which is defined as

$$r_{dx}(l) = \sum_{n=0}^{\infty} d(n)x(n-l). \tag{2.40}$$

The set of optimum inverse filter coefficients $h(n)$ is an unknown, and when solved for presents the least-squares solution to the problem. A filter derived from Eq. (2.38) is known as the Wiener Filter, named after Norbert Wiener who introduced the subject of optimum least-squares filtering in the 1940s [93, 94]. Next, two ways to solve for these filter coefficients are presented.

When using this method to design a least-squares optimum inverse filter the desired output response is a delta function,

$$d(n) = \delta(n). \tag{2.41}$$

This simplifies the system of linear equations as the cross correlation between the input and the output is now also a delta function, such that

$$r_{dx}(l) = \begin{cases} x(0); & l = 0 \\ 0; & otherwise \end{cases}. \tag{2.42}$$

Equation (2.38) can be expressed in matrix form as

$$\mathbf{R}_{xx}\mathbf{h}_N = \mathbf{u}_N \tag{2.43}$$

where bold lower case letters represent vectors and bold upper case letters represent matrices, or

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \dots & r_{xx}(N-1) \\ r_{xx}(1) & \ddots & & r_{xx}(N-2) \\ \vdots & & \ddots & \vdots \\ r_{xx}(N-1) & r_{xx}(N-2) & \dots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} = \begin{bmatrix} x(0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.44}$$

It has the notable properties that all values along each diagonal are equal, and it is symmetric. A matrix of this form is called a Toeplitz matrix. Equation (2.43) can be solved for $\mathbf{h}_N$ in the following manner:

$$(\mathbf{R}_{xx}^H \mathbf{R}_{xx})\mathbf{h}_N = \mathbf{R}_{xx}^H \mathbf{u}_N. \tag{2.45}$$

This requires computing the Hermitian transpose of the autocorrelation matrix. Doing so becomes computationally intensive as the size of $\mathbf{R}_{xx}$ becomes large. This system of equations can be efficiently solved via the Levinson-Durbin recursion algorithm [95, 96], which solves the system for $\mathbf{h}_N$ as a close approximation to the actual solution. The filter designed using this method is defined to be causal and minimum-phase [14, 29, 31].

Practical Issues

Having the delta function as the desired output may not always provide the best results. This method has been used in an attempt to equalize a mixed-phase room response by substituting a delayed delta function $\delta(n - n_0)$ in the place of $\delta(n)$. This can better approximate the acausal behavior of a mixed-phase system having non zero values before the filter peak in time [14, 95]. In addition, the Simpson sidewards recursion algorithm iteratively computes the error energy for each delay value, and this energy quantity can be used to determine the optimal delay [14, 97]. A down side to this filter design method, according to Kim et al., is that equalization performance can deteriorate significantly with small variations in measurement points [75].

Example

A system is excited by white noise that is band limited between 0 Hz and 2 kHz and sampled at 4 kHz. Its computed impulse response is shown in Fig. 2.3 (a). The above LS inverse filter design method was applied to compute its inverse impulse response shown in 2.3 (b). Figure 2.3 (c) shows the result of the linear convolution of the two impulse responses, which approximates a delta function. This is a desirable result for deconvolution; the result shows that the FIR filter designed using this LS method can be used to deconvolve a reverberant signal in the time domain, thus

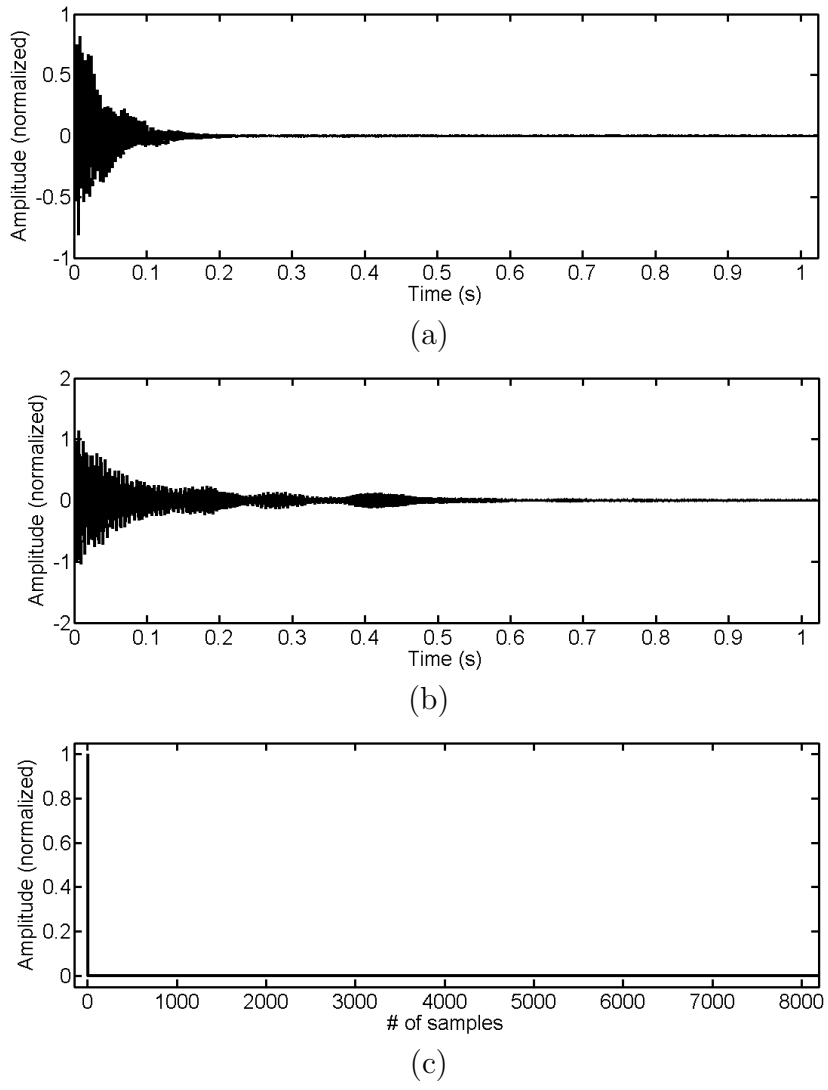


Figure 2.3 LS inverse filtering in the time domain. (a) Impulse response. (b) LS Inverse impulse response. (c) Linear convolution of the two.

removing the effects of the enclosure on the original sound. Figure 2.4 (a) shows the magnitude spectra of the tube frequency response (TFR) and its inverse filter: the room response is shown in black and the inverse filter response in red. The FFT of the time-domain linear convolution of the two [Fig. 2.3 (c)] is shown in Fig. 2.4 (b). The inverse filter does not perfectly deconvolve the signal; there are nonuniformities at 825 Hz, 1155 Hz, 1485 Hz, and 1815 Hz in the filtered spectrum. These frequencies correspond to the frequency nulls in the TFR. As evident in Fig. 2.4 (c), the coherence between the input and output signals is poor at these frequencies.

2.4.2 Direct Inversion Method

The direct inversion filter design method is a straightforward process. The inverse of a frequency response is computed as

$$H_i(k) = \frac{1}{H(k)}. \tag{2.46}$$

However, as demonstated by the following example, this method does not perform as well as the LS method. Figure 2.5 (a) shows a normalized minimum-phase impulse response. An inverse filter is first computed for this impulse response using the LS method described in the previous section, and then using the direct-inverse method. Figure 2.5 (b) shows both sets of inverse filter coefficients: those computed using the direct-inverse method in red, and those computed using the LS method in black. The two responses appear similar. A major perceivable difference is that the LS filter coefficients more closely approach zero at the end of the filter than the direct inverse coefficients. Figure 2.5 (c) shows the frequency response magnitudes of these two filters, which also appear to be very similar.

A direct inverse impulse response perfectly deconvolves a periodic signal [45], but not a nonperiodic signal. This drawback is illustrated in Fig. 2.6. The graphs show

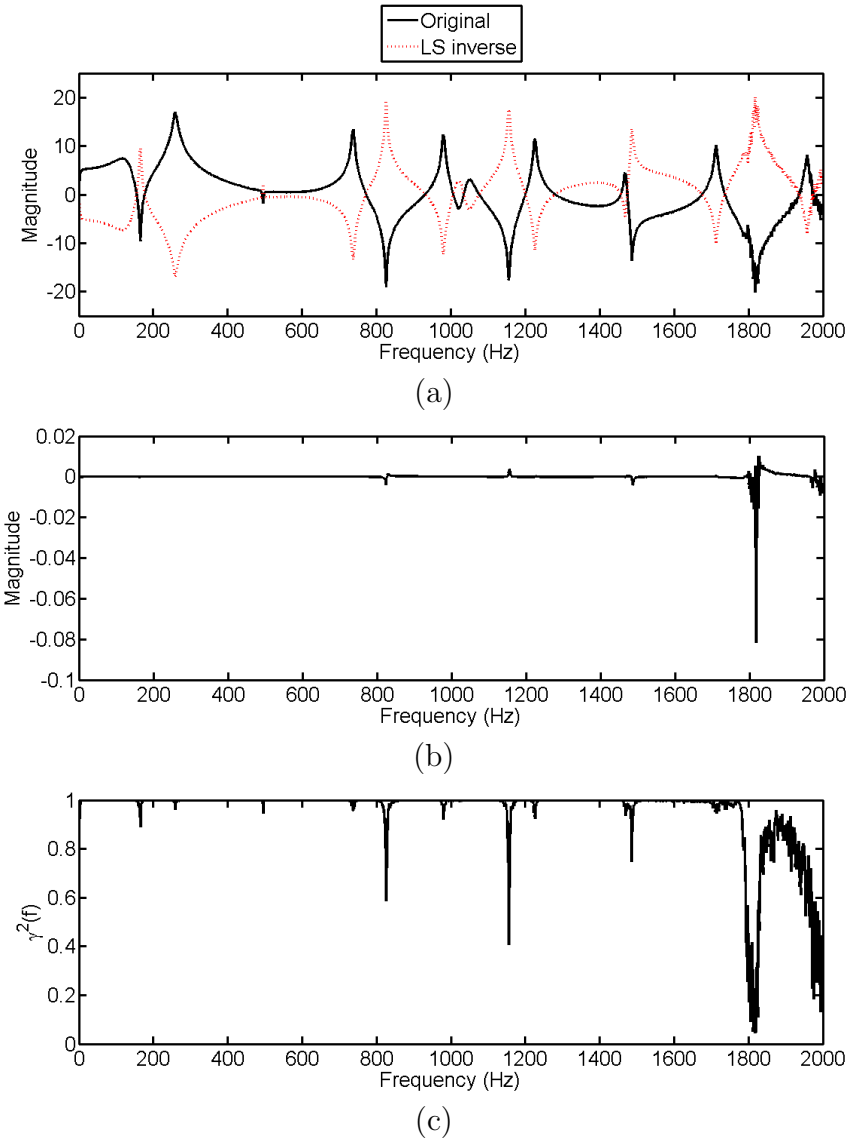


Figure 2.4 LS inverse filtering in the frequency domain. (a) The frequency response magnitude (in black) and the inverse filter magnitude (in red). (b) The magnitude of the linear convolution of the two. (c) Correlation between the input and output signals.

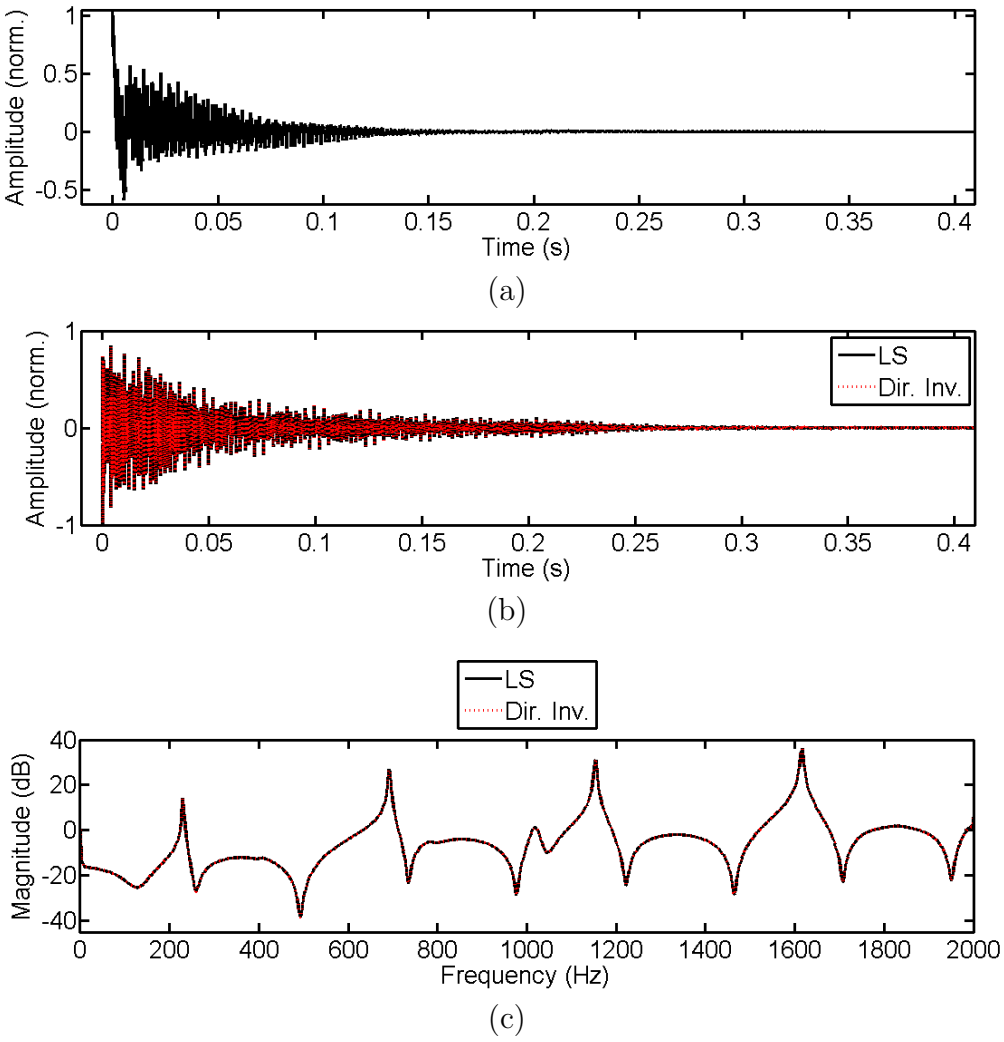


Figure 2.5 The direct inversion method. (a) Original minimum-phase impulse response. (b) The direct inverse and LS inverse filters in the time domain. (c) Magnitude spectra for the two filters.

the time and frequency-domain results of the impulse response linearly convolved with the direct-inverse and LS inverse filters. Figure 2.6 (a) shows the time-domain results of the convolutions. In the close up view of Fig. 2.6 (b), ripples are visible following the initial impulse and again at halfway through the convolution.

Since in linear time-domain convolution a dot product of the two overlapping sequences is performed at every iteration of the computation, if the two signals do not completely overlap, zeros fill in the gaps as illustrated in the simple convolution example in Sec. 2.3.1. This is not the case when the signal is periodic; the values being shifted repeat themselves. The LS method compensates for this by forcing the filter coefficients to approximately zero toward the end of the computed response. This decreases the mismatch created by linear convolution, so the method provides better results. As seen in Fig. 2.6 (b), the LS convolution result following the initial impulse is nearly perfect, and the ripples present after the halfway point of the computation result are much smaller than for the direct-inverse method.

Figure 2.6 (c) shows the frequency spectrum of the convolution results. The LS result has been offset by 3 dB for comparison. Clearly, the LS inverse filter reduces the undesired reverberation more than the direct inverse filter. The LS result has a dynamic range of approximately 1 dB, where the direct-inverse result has a dynamic range of approximately 5 dB. In fact, the LS method equalizes the impulse response almost perfectly. The ideal result is an impulse in the time domain and a flat spectrum in the frequency domain.

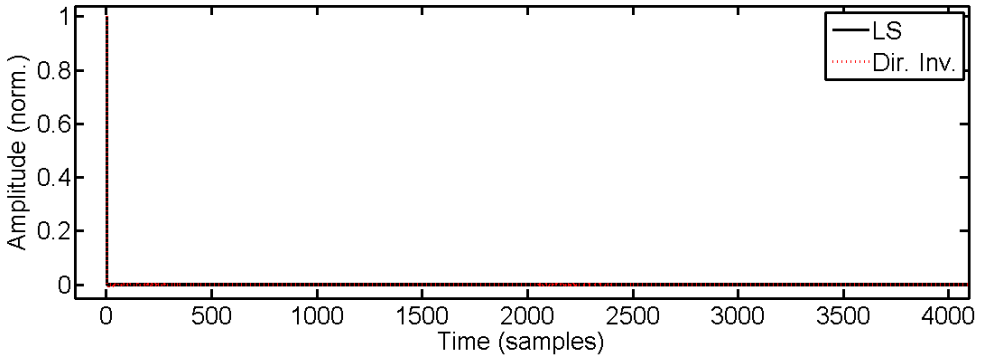
Other tests were performed for cases in which the impulse response was truncated or zero padded. When the impulse response was truncated, the direct-inverse filter performance deteriorated much more significantly than that of the LS filter. When the impulse response was zero padded, the performance of the direct inverse method improved and approached that of the LS method. Similar results were observed for

mixed-phase impulse responses. In conclusion, for the same filter lengths, the LS inverse filter performs better at equalization than a filter designed using the direct-inverse method for time-domain FIR filtering.

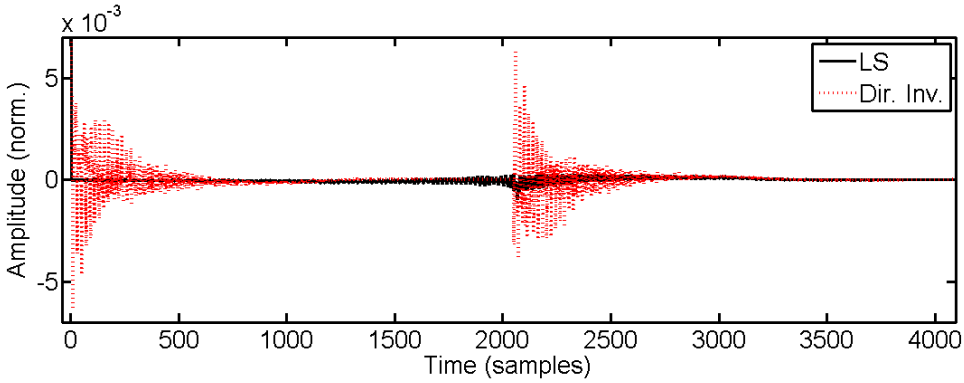
2.5 Time-Adaptive Equalization

Several time-adaptive techniques are considered for this research. One approach is to adapt the LS solution presented in the previous section continuously in real-time on a DSP. With this method the TFR is computed periodically, and if the new computed TFR is significantly different than the current one, a new set of minimum-phase inverse filter coefficients is computed in the background while the current FIR inverse filtering continues. When a new set of coefficients is finally computed, the current inverse filter coefficients are slowly transformed (morphed) into the new set to prevent an audible transition. Such a method is computationally intensive, and not practical to implement on the available hardware.

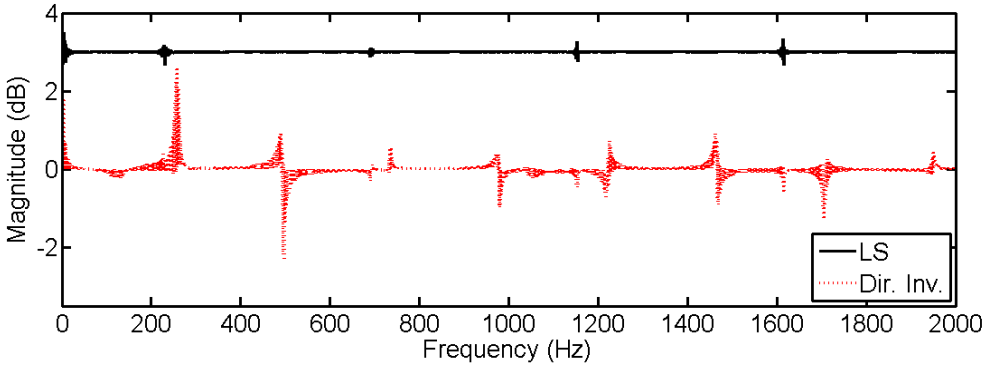
Existing morphing techniques popular with parametric equalization filters are also considered. Much of the published research on such techniques has been applied to short filters no greater than fourth order [98–104]. The same principles cannot be applied directly to the room equalization problem where the required filter lengths may be on the order of hundreds or thousands, and an inverse filter is not parametric. However, the research done on morphing filters does apply to the equalization problem because much of it focuses on the perception of audible artifacts created by changing the gain or the frequency position of a filter; both occur when an equalization filter adapts to changes in the TFR.



(a)



(b)



(b)

Figure 2.6 The direct inversion method. (a) The time-domain convolution results of the impulse response convolved with the direct inverse and LS inverse filters. (b) A close up of the time-domain convolution results. (c) The frequency spectra of the time-domain convolution results.

2.5.1 Introduction to the Filtered-x Algorithm

The LMS adaptive filter is a well-known time-adaptive filter design. It is inadequate for acoustical applications because the transfer function of the adaptive filter $W(z)$ is followed by the combined transfer function $H(z)$ consisting of electronic equipment, electroacoustic equipment, and an acoustic path. Because of the additional filtering of the reference signal by $H(z)$, the cross correlation estimate used to update the filter coefficients becomes invalid [31, 105, 106].

The filtered-x algorithm was developed as a modification to the LMS algorithm to ensure convergence. It has been used extensively in the field of active noise control (ANC). Its theory and functionality are well established, and discussed in many references, such as Refs. [105–109]. The time-domain implementation of the algorithm is typically used, but the frequency-domain implementation has also been a subject of investigation [106, 110–116]. Due to its popularity, the filtered-x algorithm and its many variations have been the subjects of many studies in published research. See for example Refs. [117–120].

The algorithm has been used for audio equalization in the time domain by Nelson, Elliott, and others [25, 35, 121, 122], often with a focus on stereo and multi-channel equalization [73, 123–125]. A theoretical overview of the algorithm follows. For a detailed derivation, see Refs. [105, 106]. A version of the filtered-x algorithm that equalizes energy density in a plane-wave tube is subsequently presented. The equalization algorithms presented here are modified versions of designs used for ANC applications in a plane-wave tube [126].

2.5.2 Filtered-x Theory Explained

The filtered-x algorithm is a feedforward system implemented with a series of FIR filters. A block diagram representing the algorithm is shown in Fig 2.7. This system performs sound pressure equalization. The reference signal is represented by $x(n)$. The desired signal $d(n)$ is a time-delayed version of $x(n)$, where $D(z)$ represents the time delay. The signal $u(n)$ is the output of the adaptive inverse filter $W(z)$ and $r(n)$ is the output of $\tilde{H}(z)$, the estimate of the system response. The latter can be derived either off-line or on-line; various methods for both cases are discussed in Ref. [105]. The signal $y(n)$ is detected in the field at the point of equalization as the filtered output of the system response $H(z)$.

The signal $x(n)$ is thus filtered by $W(z)$ before it passes into $H(z)$. As suggested earlier, the latter includes the response of the audio system, the acoustic effects of the listening environment, and the response of the error sensor. It also incorporates a time delay between the adaptive filter and the error sensor. A modeling delay $D(z)$ is used to estimate this system delay and to produce the desired signal $d(n)$. In order for the error to be computed correctly, the modeling delay must be at least the length of the time delay in $H(z)$. If $d(n)$ and $y(n)$ are synchronized in time, $W(z)$ converges to a filter with minimum-phase coefficients. If the modeling delay is larger than the system delay, the adaptive filter adjusts to the time delay difference by converging to a nonminimum-phase filter response.

The error signal $e(n)$ is computed as follows:

$$e(n) = d(n) - \sum_{l=0}^{N-1} h(l)u(n-l). \tag{2.47}$$

Here N is the length of $W(z)$ and $r(n)$ is called the filtered-x signal. It is computed as follows:

$$r(n) = \sum_{l=0}^{N-1} \tilde{h}(l)x(n-l). \tag{2.48}$$

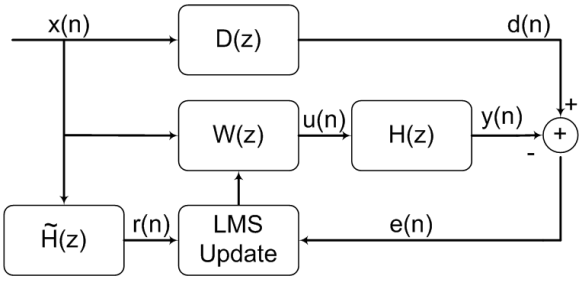


Figure 2.7 Block diagram of the filtered-x LMS algorithm.

Filtering $x(n)$ by $\tilde{H}(z)$ aligns $r(n)$ with $e(n)$ in time to yield a valid cross correlation estimate. This process gives the filtered-x algorithm its name. The LMS algorithm uses the method of steepest descent as an iterative approach to finding the minimum of a quadratic function. The algorithm adapts the $w(n)$ coefficients in the direction of the negative gradient of the mean-square error. This is represented mathematically as

$$\frac{\partial e^2(n)}{\partial w(n)} = 2e(n)\frac{\partial e(n)}{\partial w(n)} = -2e(n)r(n). \tag{2.49}$$

Thus, the filter coefficient update algorithm can be expressed as

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{r}(n). \tag{2.50}$$

Again, vector quantities are printed as bold letters and $\mathbf{r}(n)$ is defined as

$$\mathbf{r}(n) = [r(n) \quad r(n-1) \quad ... \quad r(n-N+1)]^T, \tag{2.51}$$

and μ is the convergence parameter that controls how quickly the filter coefficients converge to their optimum values and $e(n)$ goes to zero. If μ is too large, the system becomes unstable [105,106]. In the ideal case, where $d(n)$ equals $y(n)$ and $e(n)$ is zero, the signal is perfectly equalized. Because the gradient term has a negative sign, the negative gradient has a positive sign. Thus, a positive sign is used for the filter update in Eq. (2.50), as opposed to the negative sign typically seen in ANC applications.

2.5.3 The Filtered-x Algorithm with On-line System Identification

If the system response $H(z)$ is constant over time, the filtered-x algorithm converges to optimal $w(n)$ coefficients. However, $H(z)$ is often time-varying. The room response can vary greatly depending on where and when the audio is being reproduced. For example, a large hall may have a significantly different acoustical response when it is full and when it is empty.

It is possible to use an LMS technique to update $\tilde{H}(z)$ in real time to keep up with system changes. For a full derivation of this method see Ref. [127]. The block diagram in Fig. 2.8 shows how the approach is applied to audio equalization. A second LMS filter tracks potential changes in the system response $H(z)$. The program material $x(n)$ is sent through the delay $D(z)$ as discussed earlier. The signal $u(n)$ is filtered through $\tilde{H}(z)$ and the algorithm seeks to adapt the filter coefficients of $\tilde{H}(z)$ in such a manner that the output of $\tilde{H}(z)$ is equal to the output of $D(z)$. The error signal is computed as follows:

$$e_1(n) = a(n) - \sum_{l=0}^{N-1} \tilde{h}(l)u(n-l), \tag{2.52}$$

where $a(n)$ represents the output of $D(z)$ and $\tilde{h}(n)$ represents the filter coefficients of $\tilde{H}(z)$. The error $e_1(n)$ is the difference between $a(n)$ and $b(n)$, where $b(n)$ is the output of $\hat{H}(z)$. A new error signal $e_2(n)$ is defined as the difference between $e(n)$ and $e_1(n)$:

$$e_2(n) = e_1(n) - e(n), \tag{2.53}$$

and is used to update $\tilde{H}(z)$:

$$\tilde{\mathbf{h}}(n+1) = \tilde{\mathbf{h}}(n) + \mu_H e_2(n)\mathbf{u}(n). \tag{2.54}$$

The value μ_H is independent of μ, and must be separately optimized. As $e(n)$ and

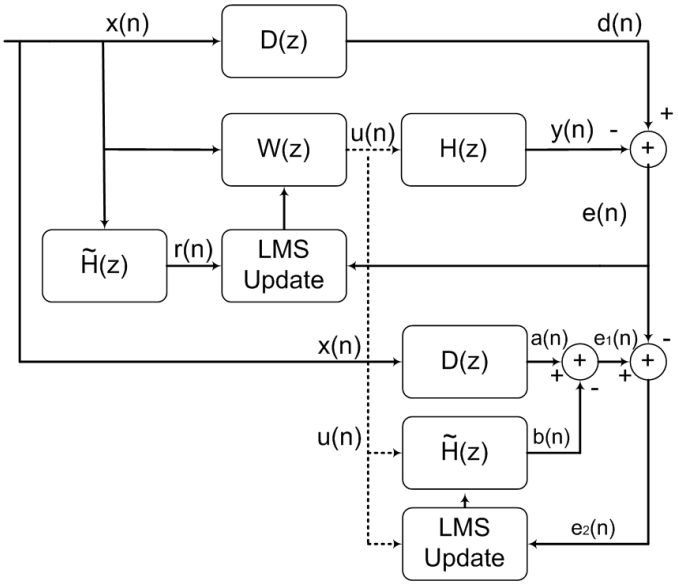


Figure 2.8 Block diagram of the filtered-x LMS algorithm with on-line system identification.

$e_1(n)$ approach zero $\tilde{H}(z)$ converges to its optimal values making it possible for $W(z)$ to accurately approximate the inverse system response.

2.5.4 Adaptation for Energy Density Equalization

The filtered-x algorithm can be modified to equalize energy density. It has been shown that instantaneous energy density, expressed as

$$w_t(\vec{r},t) = \frac{1}{2\rho_0 c^2} p^2(\vec{r},t) + \frac{1}{2}\rho_0 u^2(\vec{r},t), \tag{2.55}$$

is a positive-definite quadratic function, such that a global minimum exists that minimizes the function [128]. This means that energy density is a useful quantity for LMS applications. A method to control the energy density for ANC in a one-dimensional sound field is developed in [126]. The key modifications to adapt a pressure-based filtered-x algorithm to an energy density based algorithm is presented here. A detailed derivation can be found in Refs. [126, 128].

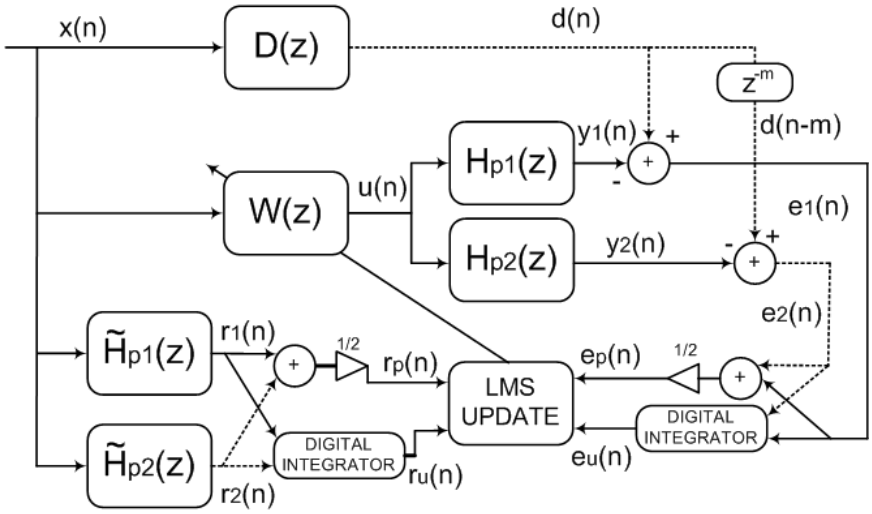


Figure 2.9 Block diagram of the filtered-x LMS algorithm adapted for energy density equalization.

A block diagram of the energy density filtered-x algorithm for equalization in a one-dimensional sound field is presented in Fig. 2.9. In this diagram $H_{p1}(z)$ and $H_{p2}(z)$ represent the electroacoustic system transfer functions associated with the paths to the two pressure sensors for energy density estimation. The filters $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$ are used to estimate these transfer functions. Two acoustic pressure measurements are necessary to derive energy density at a point in a one-dimensional sound field, as described in Sec. 2.1.3. A delay $D(z)$ is used to estimate the time delay of $H_{p1}(z)$ in Sec. 2.5.2. In addition, the digital delay z^{-m} compensates for the physical time delay between the two transfer functions $H_{p1}(z)$ and $H_{p2}(z)$ and it is equal to the separation of the incident pressure peaks in $h_{p1}(n)$ and $h_{p2}(n)$, given in samples (m). For this method to produce an accurate estimate of particle velocity the delay m must be greater than one sample, which requires the time for sound to propagate from one microphone to the next to be greater than one sampling period, $\frac{1}{f_s}$. This digital delay also assumes that the plane-wave tube is a duct wherein sound propagates only in one direction, when in fact the duct is a closed system wherein standing waves form from

sound waves propagating in both directions. The transfer function $W(z)$ represents the adaptive energy density inverse filter.

Two pressure errors, $e_1(n)$ and $e_2(n)$, are calculated by

$$e_1(n) = d(n) - y_1(n) \tag{2.56}$$

and

$$e_2(n) = d(n-m) - y_2(n), \tag{2.57}$$

where $d(n)$ is a delayed value of $x(n)$ and $d(n-m)$ is a delayed value of $d(n)$. The input signals to the DSP are represented by the electric signals $y_1(n)$ and $y_2(n)$. Since the goal is to estimate the energy density halfway between the microphones used for the pressure measurements, these two errors are used to compute an energy density error for the LMS update equation. The general energy density update equation is derived in Ref. [128]:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\left(\frac{1}{\rho_0 c^2} p(n)\mathbf{r}_{\mathbf{p}}(n) + \sum_{l=1}^{D} \rho_0 u_l(n)\mathbf{r}_{\mathbf{u_l}}(n)\right), \tag{2.58}$$

where $p(n)$ is the average measured pressure, $u_l(n)$ is a directional component of particle velocity, l is a position indicator, D is the number of dimensions for the measurement, and the summation computes the average particle velocity from the D directional components.

In a one-dimensional sound field this update expression can be reduced [128] to

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[e_p(n) + \rho_0 c e_u(n)]\mathbf{r}(n), \tag{2.59}$$

where $e_p(n)$ is the pressure error, $e_u(n)$ is the particle velocity error, and $\mathbf{r}(n)$ represents the vector of filtered-x pressure values. The reason for this is explained in Sec. 2.5.2. The pressure error is computed by

$$e_p(n) = \frac{e_1(n) + e_2(n)}{2} \tag{2.60}$$

and the particle velocity error is estimated by a discrete time-domain integral [126]

$$e_u(n) = e_u(n-1) - \frac{1}{\rho_0 \Delta x f_s}\big[e_2(n) - e_1(n)\big], \tag{2.61}$$

where the subscript u represents particle velocity, $e_u(n-1)$ represents the previous particle velocity approximation, Δx is the microphone separation, and f_s is the sample rate. Equation 2.61 can be used to compute particle velocity from two pressure measurements, or a particle velocity error from two pressure errors. A discrete frequency-domain version of this computation for estimating particle velocity is given in Eq. (2.14). The current value of $r_p(n)$ is computed by

$$r_p(n) = \frac{r_1(n) + r_2(n)}{2}. \tag{2.62}$$

The values $r_1(n)$ and $r_2(n)$ are the outputs of the filters $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$. Using this approximation the quantity $r_u(n)$ is not necessary [128]. This removes an extra filter from the update function and reduces the computational complexity of the algorithm.

On-line system identification is necessary for both the $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$ filters. This is performed with the basic LMS update as follows:

$$\tilde{\mathbf{h}}_{p1}(n+1) = \tilde{\mathbf{h}}_{p1}(n) + \mu_H e_{h1}(n)\mathbf{u}(n) \tag{2.63}$$

and

$$\tilde{\mathbf{h}}_{p2}(n+1) = \tilde{\mathbf{h}}_{p2}(n) + \mu_H e_{h2}(n)\mathbf{u}(n), \tag{2.64}$$

as shown in Fig. 2.10. The variable μ_H is the convergence parameter. The errors $e_{h1}(n)$ and $e_{h1}(n)$ are computed by

$$e_{h1}(n) = y_1(n) - a(n), \tag{2.65}$$

where $a(n)$ is the output of $u(n)$ filtered by $\tilde{H}_{p1}(z)$, and

$$e_{h2}(n) = y_2(n) - b(n), \tag{2.66}$$

where $b(n)$ is the output of $u(n)$ filtered by $\tilde{H}_{p2}(z)$.

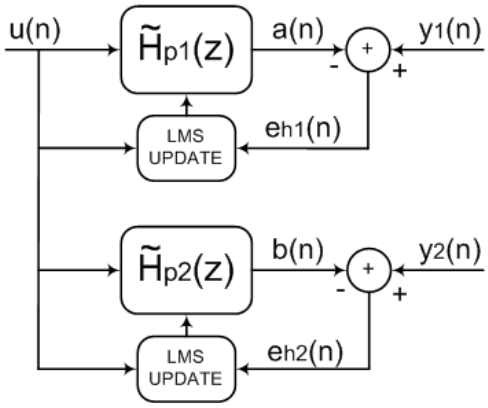


Figure 2.10 On-line system identification used with the energy density filtered-x algorithm.

2.5.5 Other Methods of Equalization Using the Filtered-x Algorithm

The method presented in the previous section is a direct application of the ANC system used by Sommerfeldt et. al for ANC [128]. Other ways to equalize energy density in the time domain were also considered and two of them are presented here. The first represents a system where only one desired signal $d(n)$ is available or preferred. The second presents a way to lower the time for the filtered-x algorithm to converge to its optimal inverse filter coefficients.

A Modification to the Filtered-x Algorithm for a Single Desired Signal

A major difference between one-dimensional energy density ANC and equalization is the lack of a second plant in the latter. This makes the energy density equalization algorithm feasible for an implementation wherein one desired signal $d(n)$ and two observed signals $y_1(n)$ and $y_2(n)$ are used to compute the error for the LMS update. A block diagram for this implementation is shown in Fig. 2.11. The on-line system identification method presented in the previous section also works for this implementation.

In this method pressure and particle velocity errors are directly computed, whereas in the the previous method two pressure errors are initially computed. The sound pressure error signal is computed as follows:

$$e_p(n) = d(n) - y_p(n), \tag{2.67}$$

where $y_p(n)$ is the average of the two microphone inputs samples:

$$y_p(n) = \frac{y_1(n) + y_2(n)}{2}. \tag{2.68}$$

A method for estimating particle velocity from two pressure measurements has been presented in Sec. 2.1.2, but in this case only one sample is used for the desired signal $d(n)$. Thus, another estimate for particle velocity must be used. In a one-dimensional sound field, the specific acoustic impedance Z_s can be expressed as

$$Z_s(x, \omega) = \frac{\hat{p}(x, \omega)}{\hat{u}(x, \omega)} \tag{2.69}$$

so the particle velocity may be expressed as

$$\hat{u}(x, \omega) = \frac{\hat{p}(x, \omega)}{Z_s(x, \omega)}. \tag{2.70}$$

The specific acoustic impedance for a single plane wave in a free field is equal to $\rho_0 c$. A set of minimum-phase filter coefficients can be computed from the specific acoustic impedance magnitude in Eq. (2.69) using the method described in Sec. 2.2. This filter can be used to compute a more accurate estimate for the desired particle velocity signal $d_u(n)$ than $\rho_0 c$, which is a constant.

Having an estimate for Z_s, the particle velocity error can be computed as follows if Z_s is a constant:

$$e_u(n) = \frac{d(n)}{Z_s} - y_u(n). \tag{2.71}$$

A specific acoustic impedance filter $Z_s^{-1}(z)$, where

$$Z_s^{-1}(z) = \frac{1}{Z_s(z)}, \tag{2.72}$$

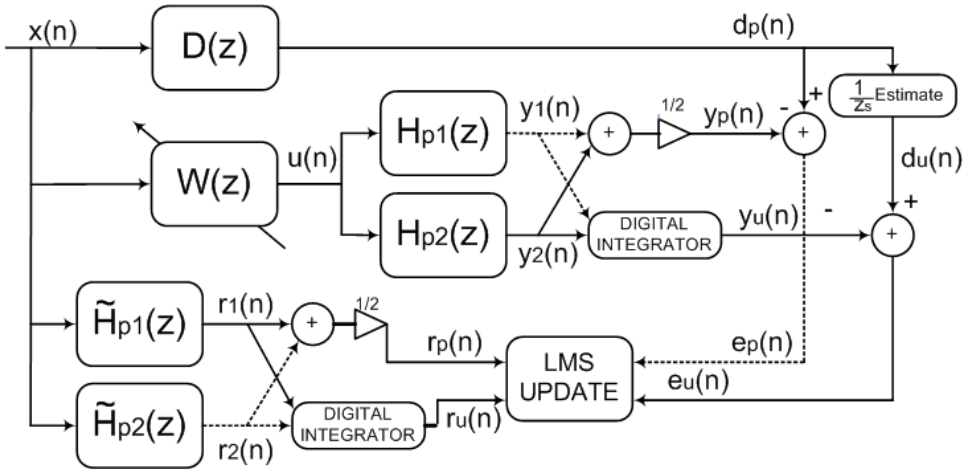


Figure 2.11 Block diagram of the modified energy density filtered-x LMS algorithm for when a single desired signal d(n) is used.

can also be used to compute the desired particle velocity estimate $d_u(n)$:

$$d_u(n) = \sum_{l=0}^{N-1} z_s^{-1}(l) d(n-l), \tag{2.73}$$

in which case Eq. (2.71) becomes

$$e_u(n) = d_u(n) - y_u(n). \tag{2.74}$$

The sound field particle velocity $y_u(n)$ is

$$y_u(n) = y_u(n-1) - \frac{1}{\rho \Delta x f_s}\big[y_2(n) - y_1(n)\big], \tag{2.75}$$

as described in the previous section. The same update equation is used as in the case with two desired signals:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[e_p(n) + \rho c e_u(n)]\mathbf{r}(n). \tag{2.76}$$

Using a $\frac{1}{Z_s}$ estimate to compute $d_u(n)$ assumes that sound pressure and particle velocity at the point of equalization are constant over time. For this filtered-x algorithm to perform accurate time-adaptive equalization the estimate must also be made time adaptive.

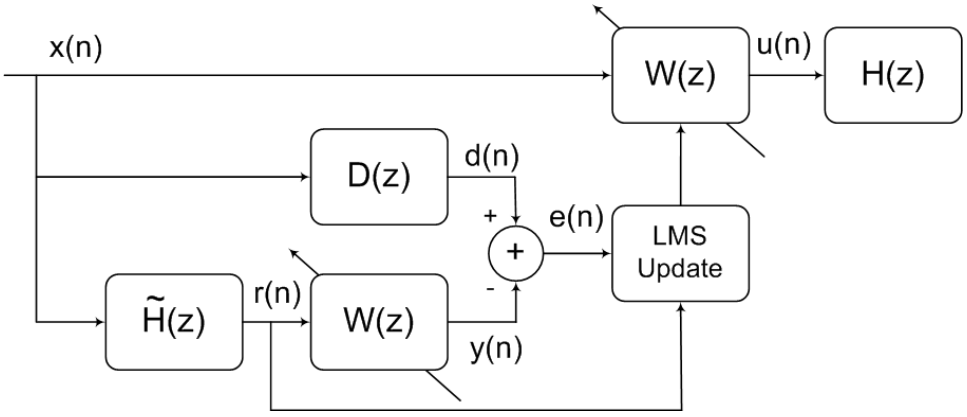


Figure 2.12 Block diagram of the modified filtered-x LMS algorithm.

Modified Filtered-x Algorithm

A method to perform sound equalization using the filtered-x algorithm without a microphone in the sound field has been studied by Goetze et. al [121, 129]. A block diagram of the algorithm is shown in Fig. 2.12. In this diagram $d(n)$ is the desired electric signal and $y(n)$ is the output of the filtered-x signal filtered by $W(z)$. The latter is an estimate of the ideal equalized sound sample perceived at the listening position. The computed error $e(n)$ is used in a standard sound pressure update equation to update both $W(z)$ filters.

One of the drawbacks of the filtered-x algorithm in comparison to the LMS algorithm is increased convergence time, which is a result of the time delay introduced to the system by the path $H(z)$, thus necessitating a lower μ for convergence. The modified filtered-x algorithm compensates for this by using $\tilde{H}(z)$ as the convergence target as opposed to $H(z)$. A sound field sensor is still needed for off-line identification to determine $\tilde{H}(z)$. Instead of computing the error between a desired electrical signal and a sample recorded from the sound field, the difference between the desired signal and the output of a second $W(z)$ filter is computed. This error signal is used to update both $W(z)$ filters. As $W(z)$ converges, it becomes the optimal inverse filter

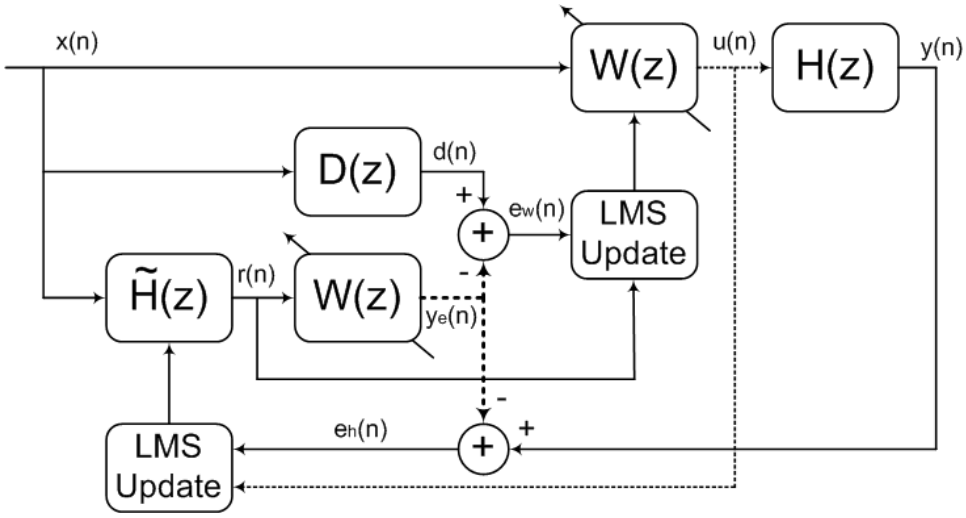


Figure 2.13 Block diagram of the modified filtered-x LMS algorithm with on-line system identification.

for $\tilde{H}(z)$. This is equivalent to a standard adaptive LMS filter, and has the added benefit that the time delay of $\tilde{H}(z)$ can be removed which guarantees that $W(z)$ converges to a minimum-phase inverse filter. The downside of this approach is that $\tilde{H}(z)$ cannot be updated in real time.

The ability of this algorithm to equalize sound pressure was tested and it performs well under steady-state conditions. It can be made time-adaptive by adding the on-line system identification function discussed in Sec. 2.5.3 to the algorithm, as shown in Fig. 2.13. In this application a microphone is located in the sound field that provides the input signal $y(n)$. This sample and $y(n)$ are used to compute an error signal $e_h(n)$ used in the on-line system identification:

$$e_h(n) = y_h(n) - y(n). \tag{2.77}$$

This error and $u(n)$ are used to update $\tilde{H}(z)$:

$$\tilde{\mathbf{h}}(n+1) = \tilde{\mathbf{h}}(n) + \mu e_h(n)\mathbf{u}(n). \tag{2.78}$$

This eliminates the constraint that $W(z)$ converges to a minimum-phase filter.

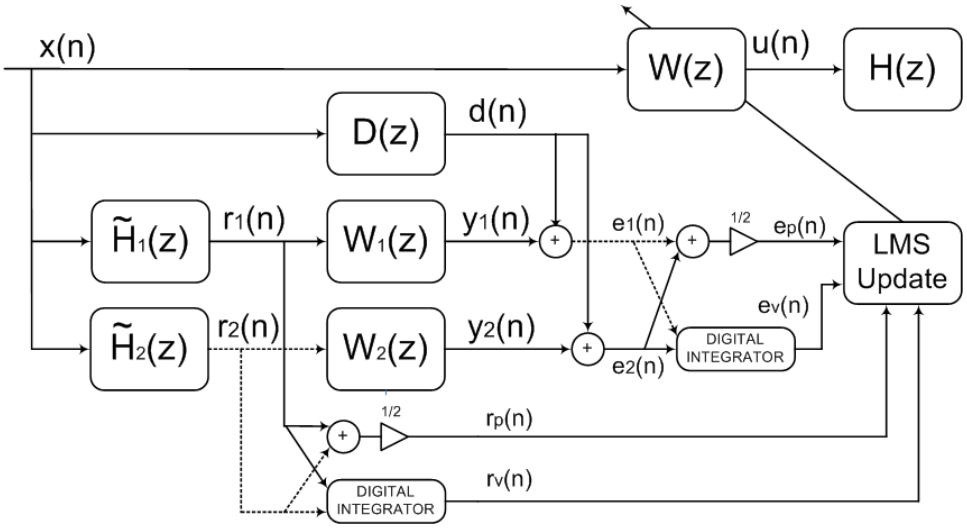


Figure 2.14 Block diagram of the modified filtered-x LMS algorithm for energy density equalization.

The application of this modified filtered-x algorithm to energy density is also considered, and a possible block diagram is shown in Fig. 2.14. Here the errors $e_p(n)$ and $e_u(n)$ are used together with $r_p(n)$ and $r_u(n)$ to update the energy density inverse filter $W(z)$ as described in Sec. 2.5.4. The filters $W_1(z)$ and $W_2(z)$ are pressure inverse filters. This implementation faces the same challenge as the method presented in Sec. 2.5.4; the energy density in the sound field must be approximated. With the addition of another secondary path to the algorithm, the number of required additional filters increases by two: $\tilde{H}_2(z)$ and $W_2(z)$. These two filters significantly increase the number of computations per cycle and for this reason the algorithm was not studied further. The algorithm is included here to present another approach to time-domain equalization, which may find applications in the future.

Chapter 3

Experimental Setup

The primary goal of this research is to develop a system to adaptively equalize a sound field in real-time using energy density. A two step process is used. First, different filter design methods are tested in MATLAB, and the best method is chosen. Then several C++ algorithms are developed and integrated into an existing RTOS running on a Texas Instruments TMS320C6713 DSP processor. A hardware system is designed and assembled from various components that has the appropriate amount of inputs and outputs, and all necessary analog filtering, gain, and voltage regulation. Much effort went into the adjustment of the hardware and software to produce reliable results. This chapter first discusses the methods that are used to develop a proof of concept and generate simulation results. It then provides a detailed explanation of the experimental setup, including explanations of the hardware and signal flow. The same experimental setup is used for time-adaptive equalization, and the the signal flow of the time-adaptive filtered-x algorithm is explained.

3.1 Development of the Equalization Method via Off-Line Processing Techniques

All functions and algorithms implemented on the DSP are first developed and tested on a computer using MATLAB. The following is a description of the process used to test the effectiveness of the designed inverse filters. First, an impulse response is acquired at a point in an enclosure. Its one-sided autocorrelation function serves as the input to the Levinson-Durbin recursion that computes the inverse filter coefficients.

The inverse filter coefficients are convolved with the impulse response to see how well the room effect is neutralized. Ideally, the result is an impulse response in the time domain and a flat magnitude response in the frequency domain. However, it is not possible to obtain a flat magnitude response with the LS method as explained in Sec. 2.4.1.

To hear how well an inverse filter deconvolves a room response, a reference sound clip is first convolved with the impulse response, then with the computed inverse filter response. In each case, the sample rate of the clip matches that of the impulse response and inverse filter. The result is saved into a WAV file and the effectiveness of the deconvolution is analyzed by listening to the result from a loudspeaker or through headphones. Any perceived undesirable artifacts are also visible in the graphical result of the impulse response deconvolution test described earlier. Results of such tests for pressure and ED-based equalizations are presented in Sec. 4.1.

3.2 Experimental Setup

This section provides details of the hardware used in the 1-D experimental setup. A block diagram of the hardware system is shown in Fig. 3.1. The plane-wave tube used is 70 cm long and has a radius of 10 cm, shown in Fig. 3.2. The signal flow of the system is also described in detail.

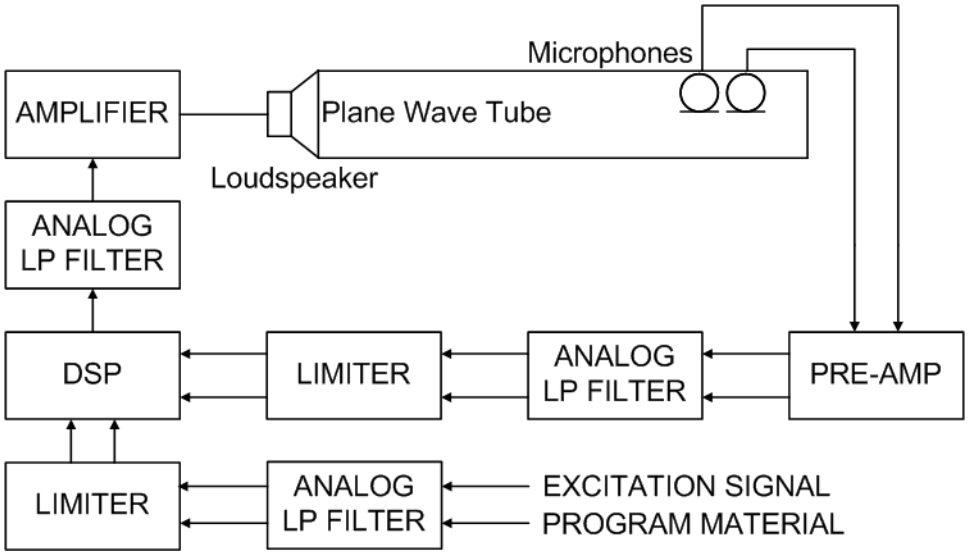


Figure 3.1 Experimental Setup Block Diagram

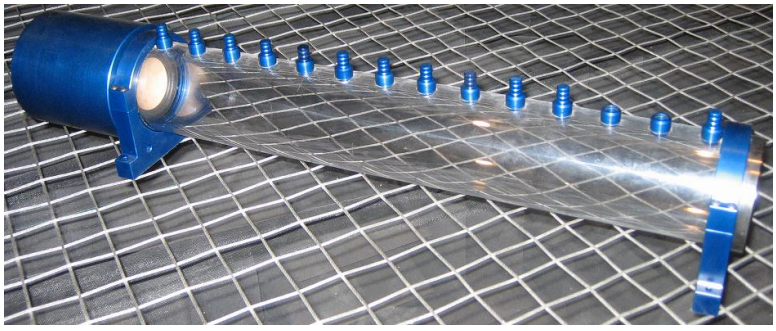


Figure 3.2 Plane-Wave Tube

Figure 3.3 The Rack Setup

3.2.1 Hardware

The hardware used for this research is assembled in a portable equipment rack for easy transportation. It consists of two four channel dBX 1046 compressor/limiters, a PCB 487B07 twelve channel ICP power supply, two four channel Krohn-Hite 3384 tunable analog filters, and a two channel Crown D-75 power amplifier, as seen in Fig. 3.3. The amplifier is used to power the loudspeaker in the plane-wave tube and the ICP power supply serves as a pre-amp for the system microphones. The Krohn-Hite filters provide reconstructive low-pass filtering of the output signal and anti-alias filtering of the microphone signals. The dBX units are used as voltage limiters to protect the DSP inputs which have an operating range of +/- 2.5 volts,

and also for fine tuning the system gain at different stages. Inputs and outputs for each device are accessible through the BNC connectors on the front patch panel of the rack unit. For sound pressure equalization, one microphone is used to probe the sound field in the tube. Two microphones are used to derive the energy density spectrum.

The Texas Instruments TMS320C6713 DSP chip is the heart of this experimental setup. It is capable of performing up to 1350 million floating point operations per second (MFLOPS) at 225 MHz. The chip is interfaced to a board manufactured by Orsys (pictured in Fig. 3.4). It is equipped with 256 Kb of on-chip memory, 32 Mb of off-chip RAM, and 8 Mb of nonvolatile off-chip memory. The DSP also interacts with a separate board having 8 input and 4 output channels with 16 bit data at up to 250 MHz.

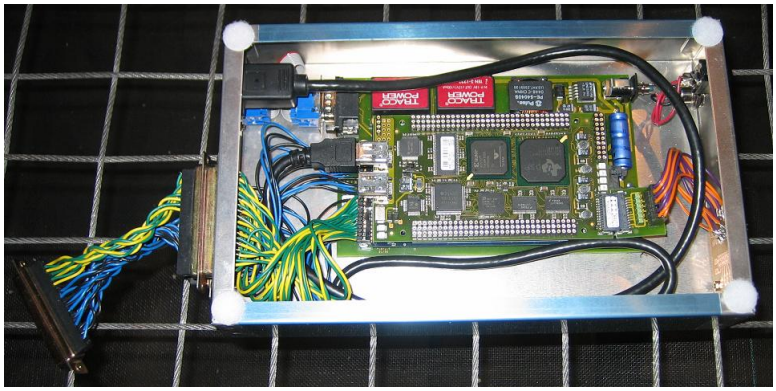


Figure 3.4 DSP Board

3.3 Steady State LS Equalization Signal Flow

This section describes how the algorithms described in the previous chapter are implemented in software. First a detailed description of how to compute a TFR is presented, followed by an explanation of how this response is used to compute an

inverse filter. A flow diagram of these processes is shown in Fig. 3.5. A description of how these inverse filter coefficients are used to filter a signal in real-time is also given.

System Identification

In order to design an inverse filter for a receiving point in an enclosure, it is first necessary to identify the frequency response between the source and the receiver. This section describes how to do this with two signals: an electrical reference signal that is also output into the plane-wave tube via a loudspeaker, and an input signal from a microphone. Figure 3.6 shows a close up of the measurement microphones in the plane-wave tube. To derive a pressure frequency response only one microphone is used, but to derive an ED frequency response both microphones are used.

A white noise signal is used to excite the plane-wave tube. An analog reconstruction filter is used at the output and an anti-aliasing filter is used at the input of the DSP. The DSP records a certain number of samples from the tube using a microphone. The length of this recording, in samples, is equal to the desired FFT length. The length of the recording, in seconds, is dependent on the DSP output sample rate; for example, with a sample rate of 4 KHz, and with 2048 samples of data, each recording is 512 ms long. At the end of the recording the DSP computes the FFTs of the input and reference signals. These results are cumulatively summed into buffers and scaled by the number of averaged recordings. For example, if the DSP averages 30 recordings the scaling factor is $\frac{1}{30}$. The DSP can average any number of recordings. After the final recording is finished, the frequency response at the measurement location is computed in the following manner:

$$H(k) = \frac{S_{xy}(k)}{S_{xx}(k)}. \tag{3.1}$$

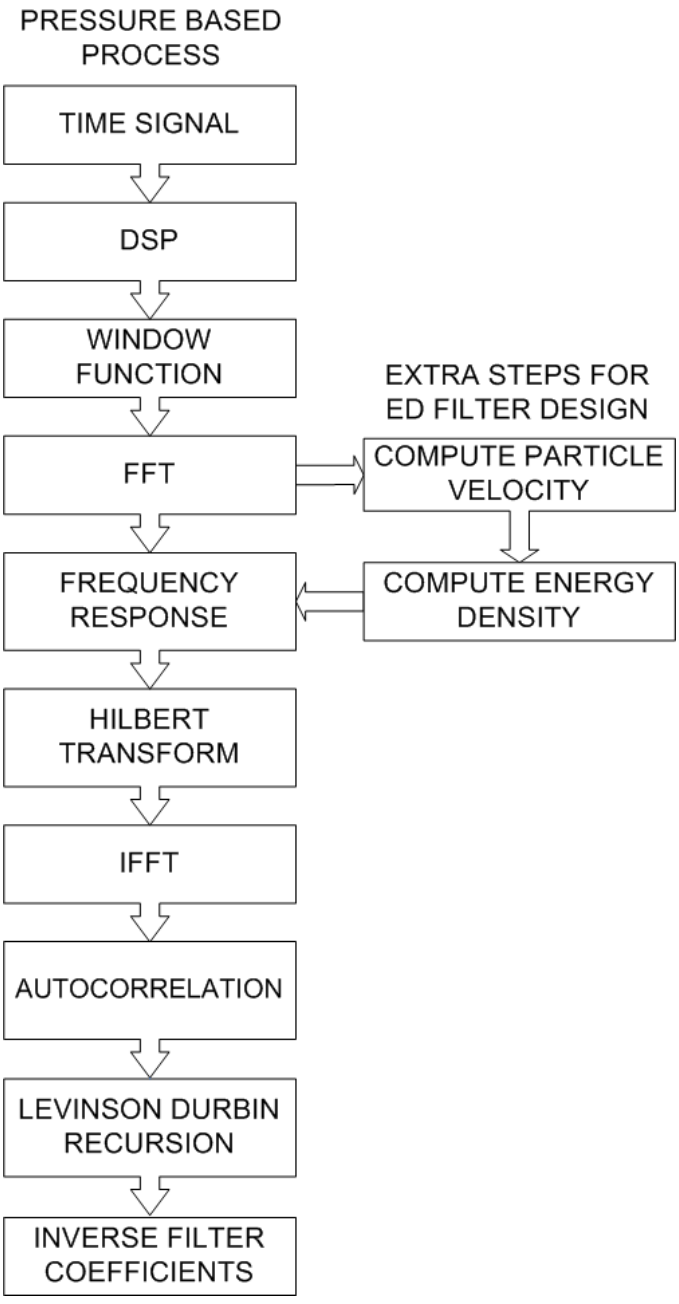


Figure 3.5 System Identification Signal Flow

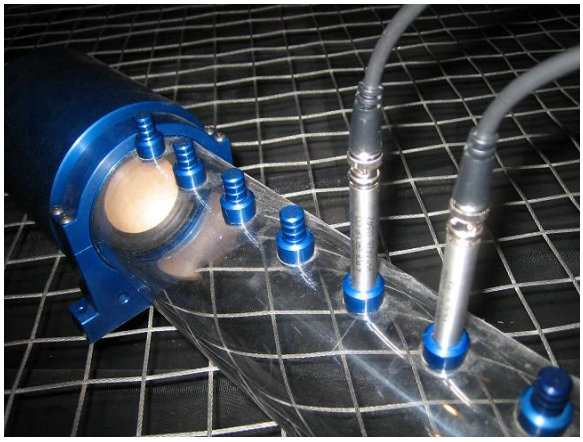


Figure 3.6 Measurement Microphones

The system input and output are represented by x and y, and the term $S_{xx}(k)$ represents the one-sided autospectral density function,

$$S_{xx}(k) = \frac{1}{n_d} \sum_{i=1}^{n_d} X_i^*(k) X_i(k), \tag{3.2}$$

where n_d is the number of ensemble averages. $S_{xy}(k)$ represents

$$S_{xy}(k) = \frac{1}{n_d} \sum_{i=1}^{n_d} X_i^*(k) Y_i(k). \tag{3.3}$$

Many ways exist to estimate the frequency response of a system. Equation 3.1 is optimal when a system does not have input noise, but uncorrelated output noise is present [130]. The minimum-phase frequency response is found by computing the Hilbert transform of the measurement frequency response. The IFFT of this minimum-phase response yields a minimum-phase impulse response. Its one-sided autocorrelation function is the input to the Levinson-Durbin recursion algorithm producing the FIR inverse filter coefficients, which are then stored on the DSP. The number of taps places constraints on how long of an inverse filter can be implemented for a given sample rate in real-time. The filtering operation must take less time than is available between outputs.

This process changes slightly when computing an ED based inverse filter. The

ED frequency response is derived using the method explained in Sec. 2.1.3, then its minimum-phase ED frequency response is computed using the Hilbert transform, and the subsequent steps are the same as above.

Inverse Filtering

Program material is passed into the DSP where it is filtered by the inverse filter coefficients in real-time, one sample per cycle, using a time-domain FIR filter. The length of the cycle is determined by the sample rate of the DSP. The signal is then passed to the loudspeaker. The sample rate of the program material should be at least equal to the DSP sample rate. If it is higher, the DSP down-samples the signal to match the sample rate of the DSP outputs.

3.4 Time-Adaptive Filtered-x Equalization Setup and Signal Flow

A filtered-x algorithm is used to adapt the energy density inverse filter coefficients to match changing conditions in the physical system. The derivation of this method is included in Sec. 2.5.4, and the block diagram is shown in Fig. 2.9. The inverse filter $W(z)$ is initialized to the normalized values of the $W(z)$ values computed during the system identification process described in the previous section. The two filters estimating the physical paths between the loudspeaker and the measurement microphones, $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$, are initialized using an off-line system identification process that utilizes a basic LMS filter update (shown in Fig. 3.7):

$$\tilde{\mathbf{h}}_{p1}(n+1) = \tilde{\mathbf{h}}_{p1}(n) + \mu_H e_{h1}(n)\mathbf{x}(n) \tag{3.4}$$

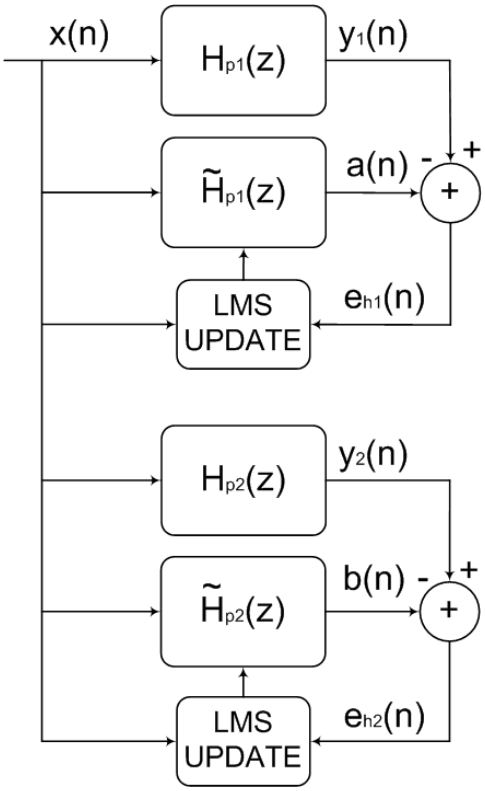


Figure 3.7 Block diagram of the off-line system identification algorithm.

for $\tilde{H}_{p1}(z)$ and

$$\tilde{\mathbf{h}}_{p2}(n+1) = \tilde{\mathbf{h}}_{p2}(n) + \mu_H e_{h2}(n)\mathbf{x}(n) \tag{3.5}$$

for $\tilde{H}_{p2}(z)$, where μ_H is the convergence parameter for the adaptive $\tilde{H}(z)$ filters, and $x(n)$ is the current sample of the excitation signal. The two errors are computed as follows:

$$e_{h1}(n) = y_1(n) - a(n), \tag{3.6}$$

where $a(n)$ is the output of $\tilde{H}_{p1}(z)$ and

$$e_{h2}(n) = y_2(n) - b(n), \tag{3.7}$$

where $b(n)$ is the output of $\tilde{H}_{p2}(z)$.

Every sampling period, the algorithm computes the output of $W(z)$, $u(n)$, which is sent to the loudspeaker, and the values of the two inputs $y_1(n)$ and $y_2(n)$ sampled

by the two measurement microphones. The $W(z)$, $\tilde{H}_{p1}(z)$, and $\tilde{H}_{p2}(z)$ coefficients are updated. This process is explained in detail in Sec. 2.5.4. As the physical system changes, these three filters slowly adapt to match the new conditions. The main parameters that impact the performance of the algorithm are the sample rate, length of $W(z)$, length of $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$, μ_H, and μ. These variables are all correlated in function; changing one impacts how well the others perform. Filter lengths of 400 samples are used. When making modifications to the algorithm one might make changes to the parameters in the order presented here until desirable results are achieved.

For this research, the experimental setup for time adaptive-equalization is the same as for steady state equalization with different system settings. The system is again excited with white noise and a filter cutoff frequency of 1100 Hz is used, along with a sample rate of 6000 Hz. The filtered-x algorithm has performance limitations; in order for the system to remain stable for greater bandwidths (such as 1500 and 2000 Hz) μ must be such a low value that it prevents the algorithm from converging. Otherwise after a period of time the system becomes unstable and the filter coefficients diverge.

Chapter 4

Results

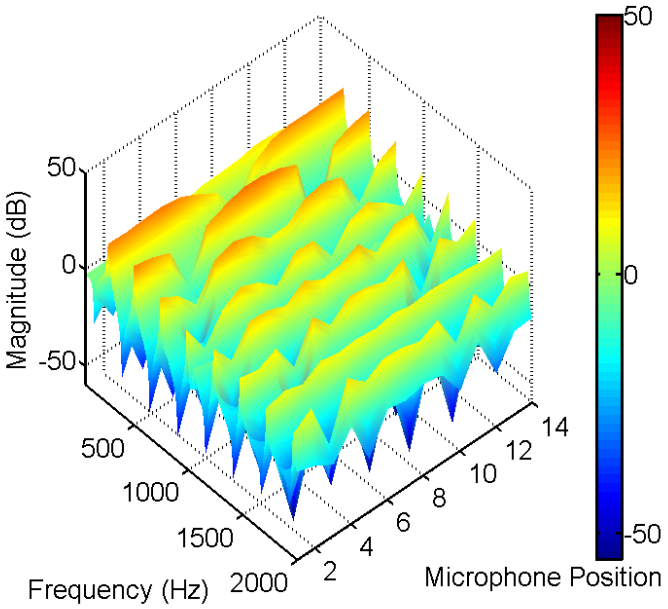
This chapter presents the results of the research. The first section presents results of time-invariant equalization in the plane-wave tube. The equalization filters are computed using the LS method, which gives the optimal inverse filter coefficients for a given tube frequency response (TFR). Results of sound pressure equalization at two nonsymmetric locations in the plane-wave tube are presented. These results are compared to the performance of two energy density inverse filters derived at approximately the same locations. The second section discusses time-variant energy density equalization where the plane-wave tube is modified to force a change in the sound field. The filter coefficients derived using the LS method are used as initial values for the time-adaptive filter. The adaptive filter updates are performed by applying the filtered-x algorithm. All surface and contour plots presented in this chapter consist of computed frequency response magnitude spectra.

4.1 Time-Invariant Equalization Results

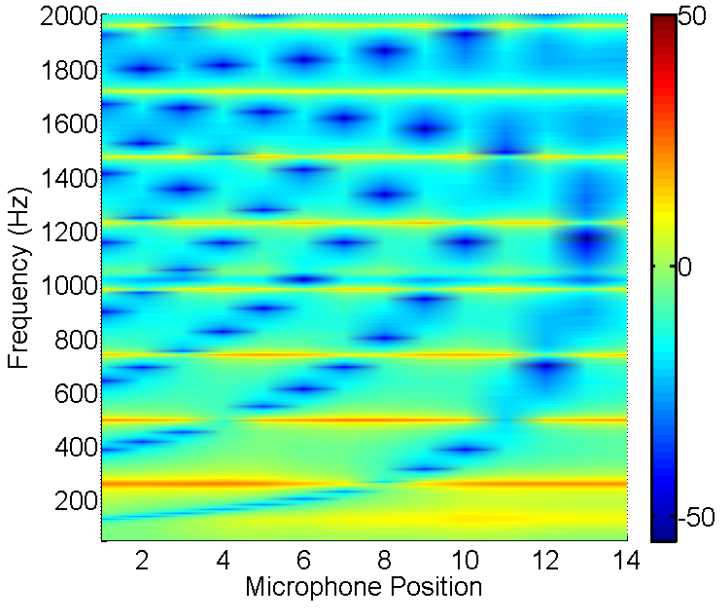
This section presents experimental results based on measurements taken in the plane-wave tube at a sample rate of 5000 Hz. The LS method is capable of nearly flat sound pressure equalization at a point, but the same filter applied to another point in the tube does not improve the response due to the mismatch in TFRs. In comparison, an energy density filter derived at any point in a one-dimensional field performs equally well in compensating for the system resonances, but such a filter does not address the nulls in the system. This is illustrated by comparing impulse responses and inverse filters computed by the DSP, and their corresponding spectra, at two locations in the plane-wave tube. (The impulse responses and inverse filters are 2048 samples long.)

The impulse responses are convolved with filter responses via time-domain post-processing for comparison and discussion. All conclusions are verified by a comparison to waterfall plots of measurements taken during real-time DSP filtering of white noise in the tube with each inverse filter. Figure 4.1 shows two dimensional spatial-spectral plots of the pressure field in the tube excited with white noise; Fig. 4.1 (a) shows a surface plot of the field and Fig. 4.1 (b) presents a top-down contour plot or color map. The global resonances and spatial nulls are clearly visible in these plots, as the spectra vary for each microphone position. These nulls are typically formed when a microphone is placed where a node exists for that particular frequency. Recordings of a band-limited ($f_c = 2000Hz$) sound clip made at microphone positions 7 and 10 are included in Appendix A, as well as the original dry sound clip.

As shown in Fig. 3.2, the plane-wave tube has 14 microphone ports on the top that either hold a half-inch microphone or a plug of the same size. The ports have 5 cm center-to-center spacing, with the first being 2.3 cm from the end of the tube and 3.2



(a)



(b)

Figure 4.1 Spatial-spectral plots of the pressure field in the plane-wave tube excited with white noise. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

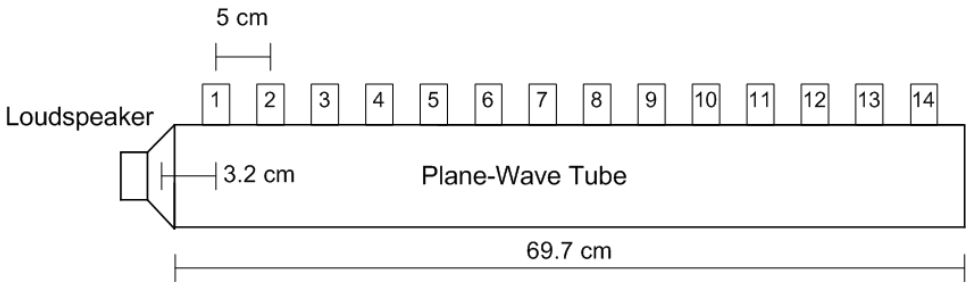


Figure 4.2 Plane-wave tube with a loudspeaker and 14 microphone ports.

cm from the geometric center of the loudspeaker. These are referred to as microphone positions $1 - 14$, with the first position closest to the source (shown in Fig. 4.2) and the last 2.3 cm from the end of the tube. The total length of the tube is 69.7 cm. Pressure equalization performance is compared between microphone positions 7 (33.2 cm from the source) and 10 (48.2 cm from the source).

Energy density filters are computed using microphone pairs at positions 7 and 8 (38.2 cm from the source), then positions 10 and 11 (53.2 cm from the source). The energy density is effectively computed at positions 7.5 (35.7 cm from the source) and 10.5 (50.7 cm from the source), since the average pressure and particle velocity are evaluated halfway between the two microphones in the pair. Results for pressure-based equalization are presented first, followed by results for ED equalization. The conclusions are verified by comparing the impulse response deconvolutions presented to the actual filtered results in the tube.

4.1.1 Pressure Equalization

For the pressure-based equalization experiments, the TFR was computed at positions 7 and 10 to derive the minimum-phase impulse response and the inverse filter coefficients using the LS method. The two normalized impulse responses and their frequency responses, along with their normalized inverse filters and their frequency

responses, are shown in Fig. 4.3 for position 7 and in Fig. 4.4 for position 10. All time-domain expressions have been normalized such that their maximum magnitudes are 1. All subsequent impulse responses presented in this section are also normalized. The frequency responses are generated by computing the FFTs of the impulse responses. The sound pressure inverse filters are as long as the time-domain window length and converge to approximately zero by the end of the window.

The effects of global tube resonances and local nodes caused by standing waves are clearly apparent in Fig. 4.3 (b) and Fig. 4.4 (b). The eight global resonances formed in the tube are clearly visible in Fig. 4.3 (b), as is the loudspeaker resonance at approximately 130 Hz and the loudspeaker cavity resonance at approximately 1030 Hz. A gradual downward slope can be observed in this frequency magnitude plot, as in all the other spectra measured in the tube. This slope is caused by the nonideal anechoic response of the loudspeaker. Because the two measurements were taken at different locations, the nulls in the responses occur at different frequencies, as they result from different standing waves.

The spectra in Figs. 4.3 and 4.4 have a dynamic range of nearly 80 dB. Prefiltering a signal using a filter with such a wide dynamic range before playback presents challenges. If the input signal to the DSP has spectral peaks that leave less than 40 dB of headroom on the DSP the signal will clip. To avoid this, the program material must be delivered at a level low enough to not produce clipping at the digital-to-analog converter (DAC). This general problem is present in all equalization systems that use a prefiltering method to varying degrees, depending on the dynamic range of the given inverse filter frequency response. These boosts to the system caused by the inverse filter response that compensate for local nodes can cause physical damage to the system.

Figure 4.5 (a) shows the TFR magnitude at position 7 overlaid with its inverse

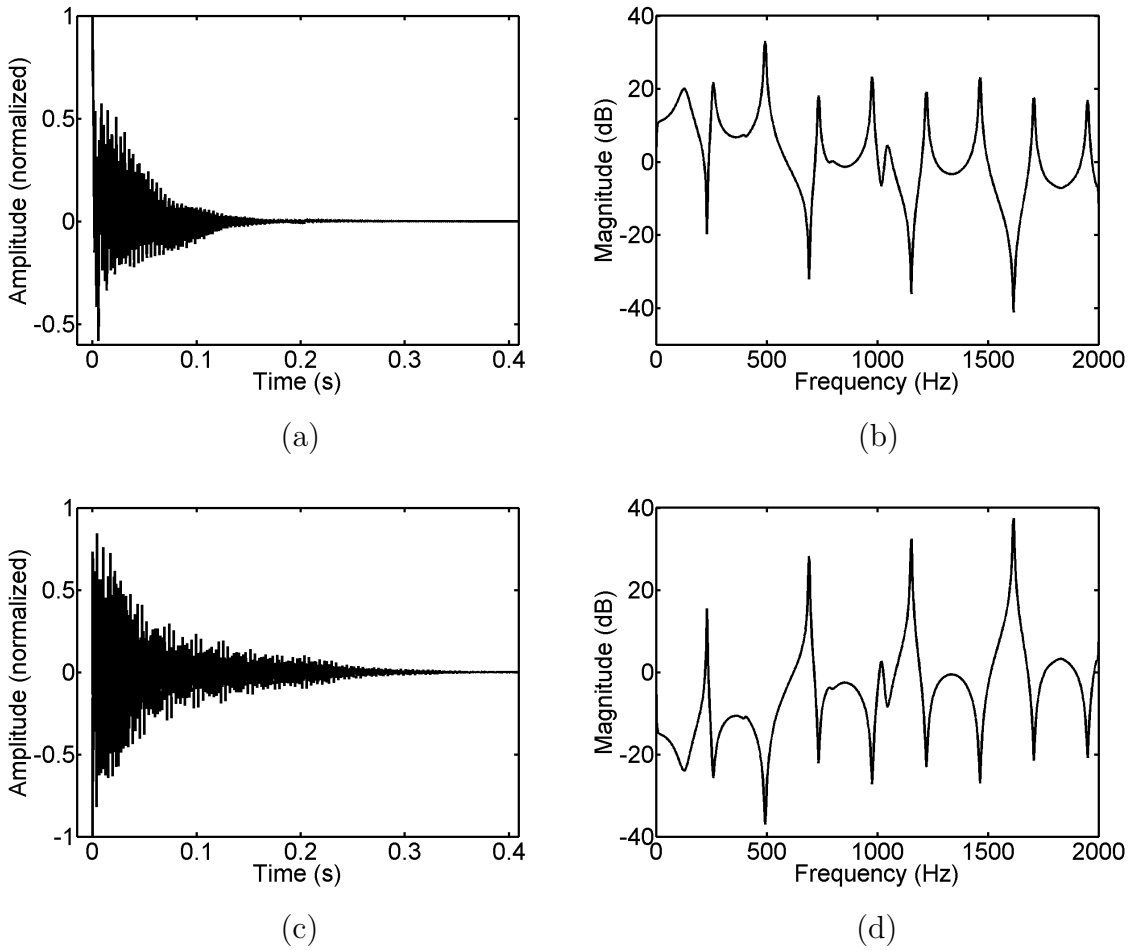


Figure 4.3 Sound pressure responses at microphone position 7. (a) The impulse response. (b) The frequency response magnitude. (c) The inverse filter coefficients. (d) The inverse filter magnitude.

filter spectrum. The LS algorithm provides a nearly perfect inverse of the system response at that position. There is a drop in measured coherence at all frequencies corresponding to resonances and nulls in the TFR, as is evident in Fig. 4.5 (b). The LS algorithm is not able to perfectly account for the nulls located at 231 Hz, 691 Hz, 1154 Hz, and 1615 Hz in the TFR, but it is able to equalize all resonances, including the loudspeaker and the cavity effects. It also compensates for the slope of the anechoic loudspeaker response. This leads to an imperfect equalized spectrum as seen in Fig. 4.5 (c). This magnitude spectrum presents the frequency-domain convolution

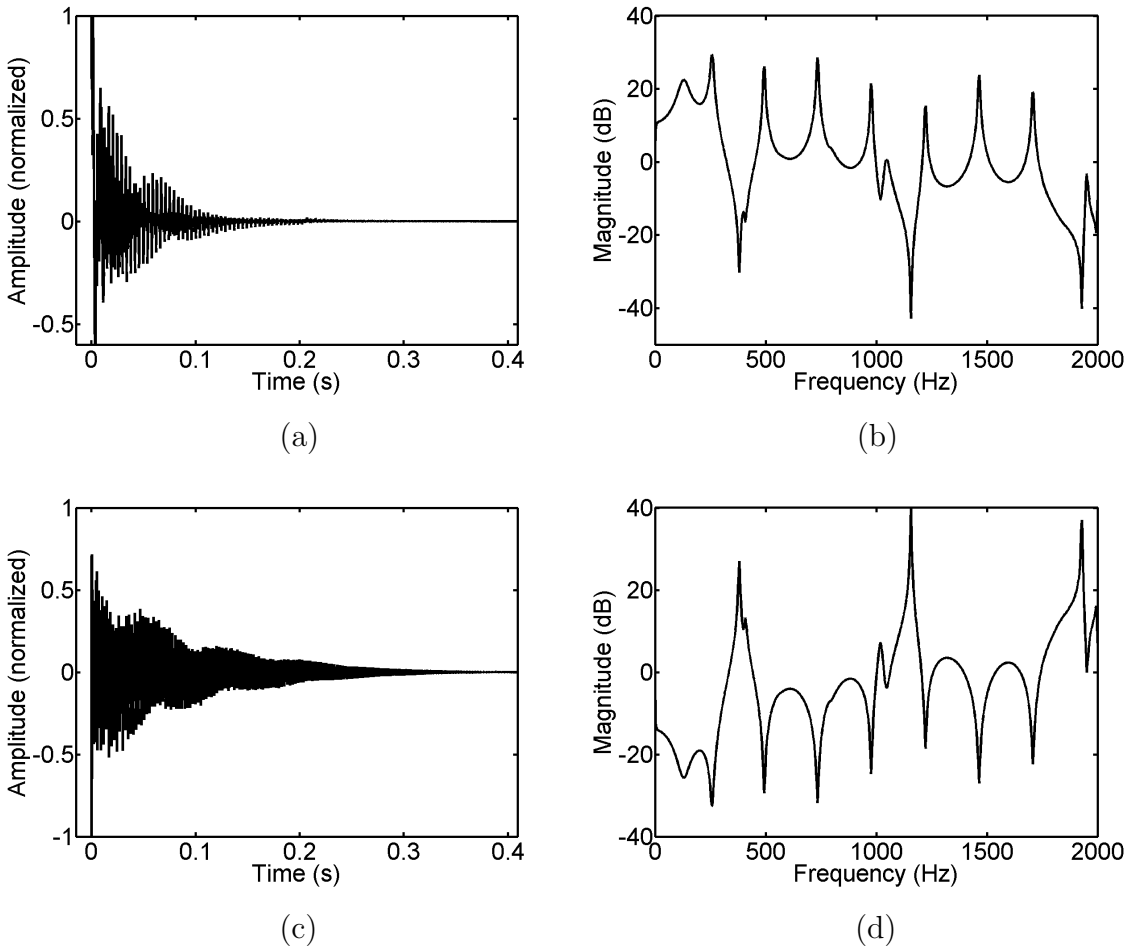


Figure 4.4 Sound pressure response at microphone position 10. (a) The impulse response. (b) The frequency response magnitude. (c) The inverse filter coefficients. (d) The inverse filter magnitude.

result of the impulse response deconvolved by the inverse filter coefficients. The four major deviations from the equalized response in Fig. 4.5 (c) correspond to the four null frequencies in the TFR shown in Fig. 4.5 (a). The time-domain convolution result shown in Fig. 4.5 (d) closely resembles a time-domain impulse. The TFR is equalized almost perfectly, and an audio signal equalized at this position with this filter sounds like the original recording.

Figure 4.6 shows a spectral-spatial surface plot of the plane-wave tube equalized in real-time with the inverse filter derived at position 7. This figure consists of transfer

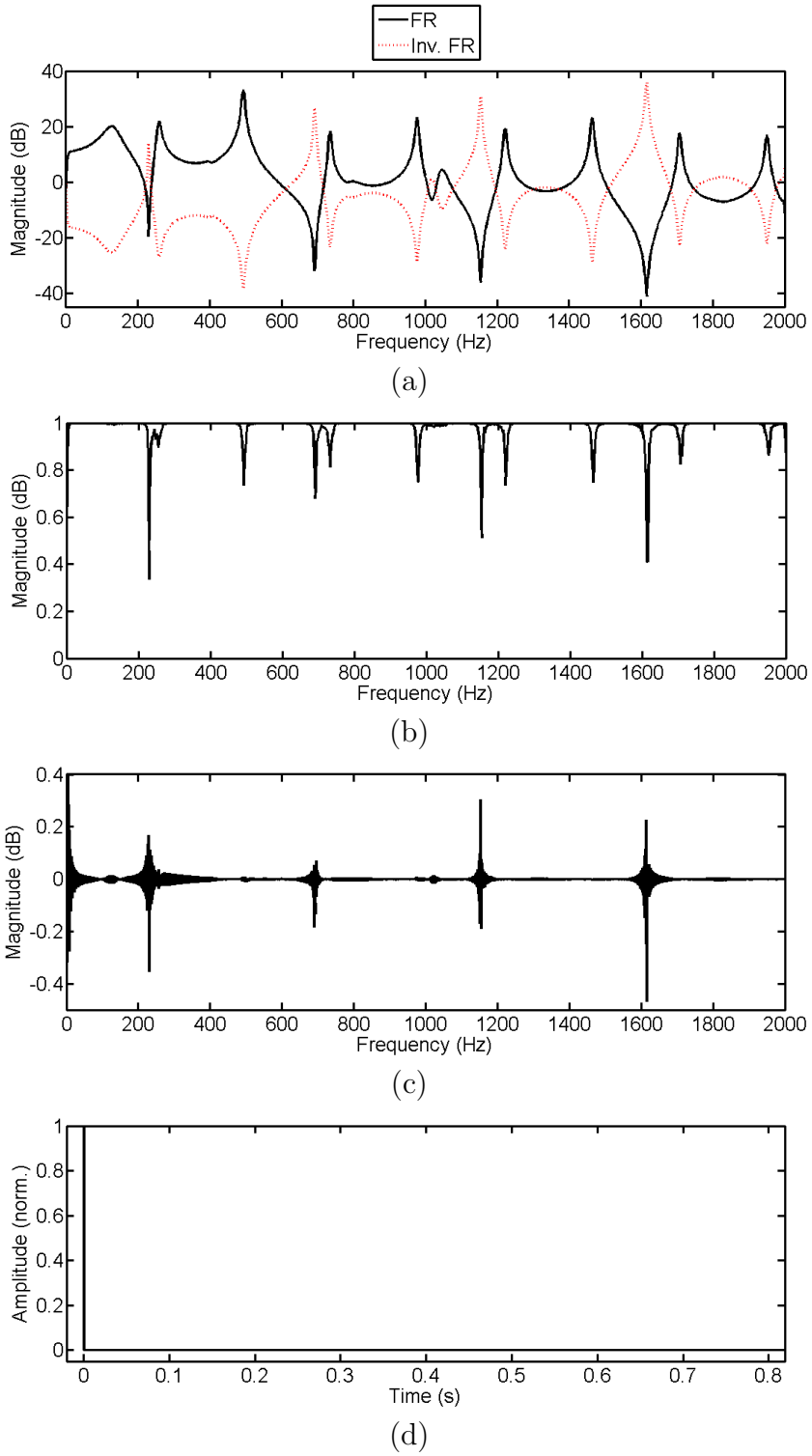


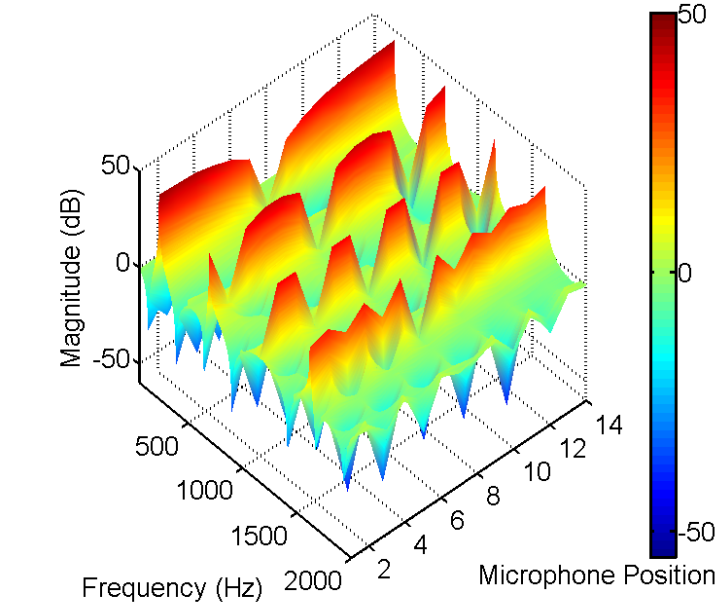
Figure 4.5 Sound pressure equalization at microphone position 7. (a) The tube frequency response and inverse filter frequency response magnitude. (b) Coherence of frequency response magnitude. (c) The convolution result in the frequency domain. (d) The convolution result in the time domain.

functions computed at each of the microphone positions and it shows the flattened frequency response at microphone position 7. This inverse filter equalizes the resonances and nulls detected at the microphone, but introduces new resonances as it compensates for the nulls at position 7 elsewhere in the tube. In fact, as can be seen in both Figs. 4.6 (a) and (b), the frequencies where the inverse filter compensates for the nulls at position 7 are greatly amplified at the other locations in the tube where these same pressure nulls do not exist. These new resonances present in the equalized sound field occur at 231 Hz, 691 Hz, 1154 Hz, and 1615 Hz, as indicated earlier.

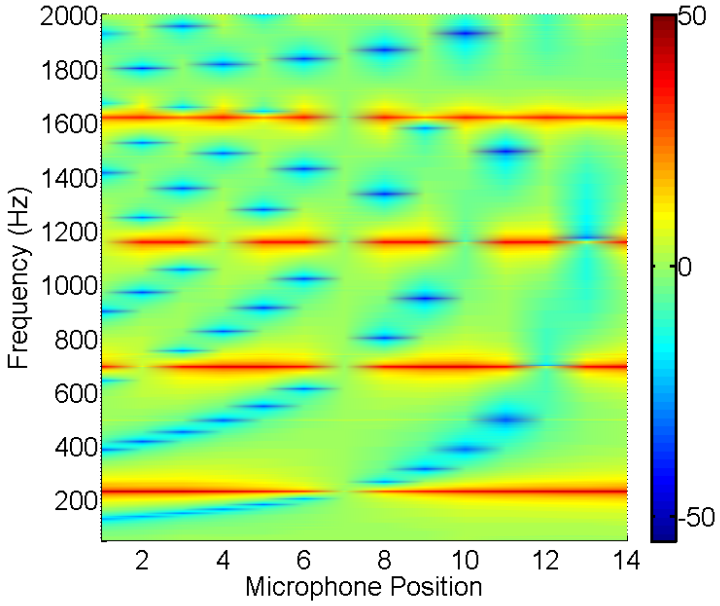
The sound pressure inverse filter derived at position 10 deconvolves the impulse response at the same position very well as shown in Fig. 4.7. Similar observations as for Fig. 4.5 can be made; the coherence function suffers at every frequency that corresponds to a null or a resonance in the TFR [Fig. 4.7 (b)], and the deviations in the magnitude spectrum of the convolution result occur at 387 Hz, 1156 Hz, and 1928 Hz, which correspond to the nulls in the TFR [Figs. 4.7 (a) and (c)]. The loudspeaker effects are also equalized. The time-domain convolution result is nearly an ideal impulse so that the reverberation at that location is almost completely removed [Fig. 4.7 (d)]. Again, an audio signal equalized at this position with the inverse filter sounds like the original recording.

Figure 4.8 shows the spectral-spatial plots of the plane-wave tube equalized in real time with the inverse filter derived at position 10. The frequency response at microphone position 10 is flattened by the inverse filter. While the spectrum is flat at that position the inverse filter attempts to compensate for the nulls present at position 10, such that the null frequencies are greatly amplified at all other locations in the tube where the nulls do not exist. These new resonances present in the equalized sound field correspond to the following frequencies: 387 Hz, 1156 Hz, and 1928 Hz.

Figure 4.9 shows the equalization result of a sound pressure filter derived at posi-



(a)



(b)

Figure 4.6 Spatial-spectral plots of the pressure field in the plane-wave tube after being filtered in real time with an inverse sound pressure filter derived at position 7. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

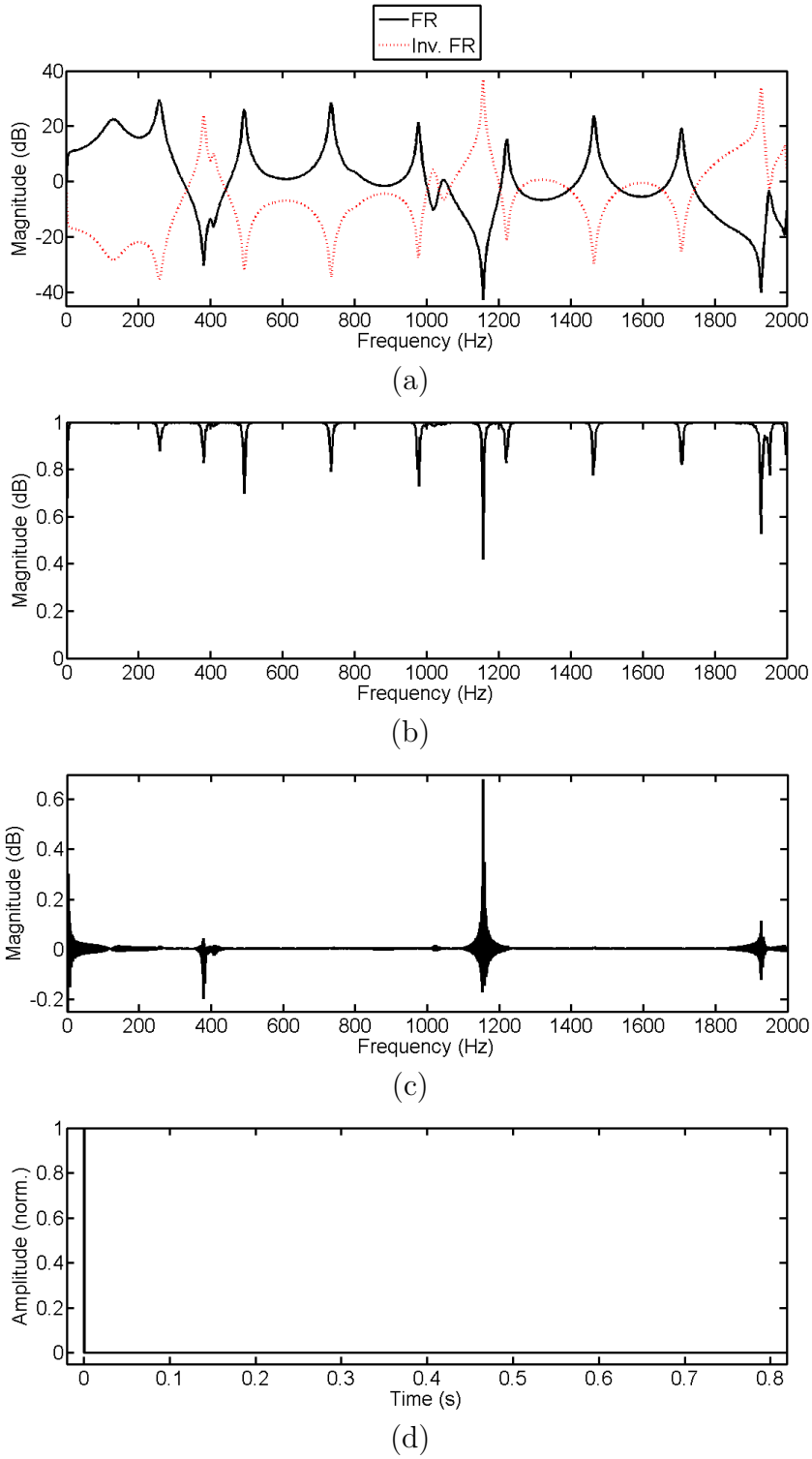
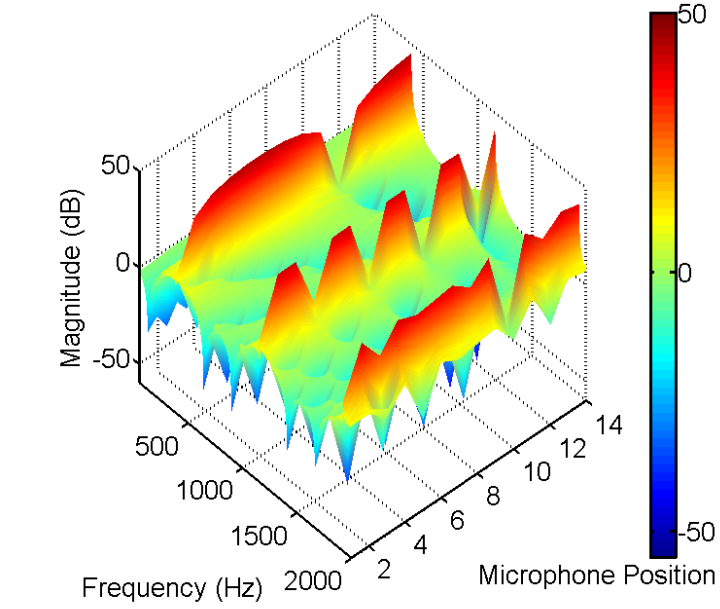
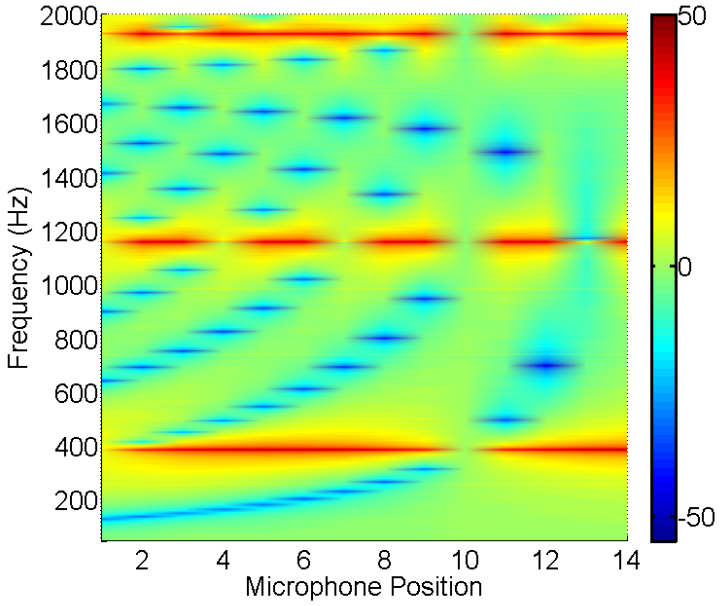


Figure 4.7 Sound pressure equalization at microphone position 10. (a) The tube frequency response and inverse filter frequency response magnitude. (b) Coherence of frequency response magnitude. (c) The convolution result in the frequency domain. (d) The convolution result in the time domain.



(a)



(b)

Figure 4.8 Spatial-spectral plots of the pressure field in the plane-wave tube after being filtered in real time with an inverse sound pressure filter derived at position 10. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

tion 10 used to attempt to deconvolve the tube response at position 7. The mismatch of the two spectral magnitudes is evident in Fig. 4.9 (a). The magnitude spectrum of the time-domain convolution result in Fig. 4.9 (b) shows that the mismatch results in poor equalization. The pressure inverse filter at position 10 compensates for spectral nulls that do not exist at position 7, thus great boosts result at 386 Hz and 1929 Hz. The nulls at 231 Hz, 691 Hz, and 1615 Hz are not equalized because the same nulls are not present at position 10. The two positions have a null near 1154 Hz so the mismatched pressure inverse filter compensates for that, but not perfectly as can be observed in Fig. 4.9 (b). This is due to the difference of 2 Hz between the two nulls.

Interestingly, the time-domain convolution result shows a larger decay than the unequalized impulse response shown in Fig. 4.3 (a). The original impulse response decayed in approximately 250 milliseconds, whereas the convolution result decays in approximately 400 milliseconds. Clearly, this filter does not successfully deconvolve the TFR at position 7. An audio signal equalized at this location with the mismatched filter sounds significantly worse than one equalized with the correct filter. Furthermore, it also does not improve the clarity, and other subjective impressions of the unequalized sound in the tube.

In a similar manner, Fig. 4.10 shows the equalization result of a sound pressure filter derived at position 7 used to deconvolve the tube response at position 10. The inverse filter magnitude is mismatched with the TFR. The global resonances are uniform across the sound field, but they cannot be observed with a microphone at all locations due to sound pressure nodes for a given frequency. Furthermore, the combination of nulls in the TFR is unique for each nonsymmetric TFR. For example, at position 7, spectral nulls exist at 231 Hz, 691 Hz, 1154 Hz, and 1615 Hz. Here the boost in the inverse filter magnitude at 1154 Hz compensates for the null at 1156 Hz, but not perfectly as seen in Fig. 4.10 (b). The other three nulls do not

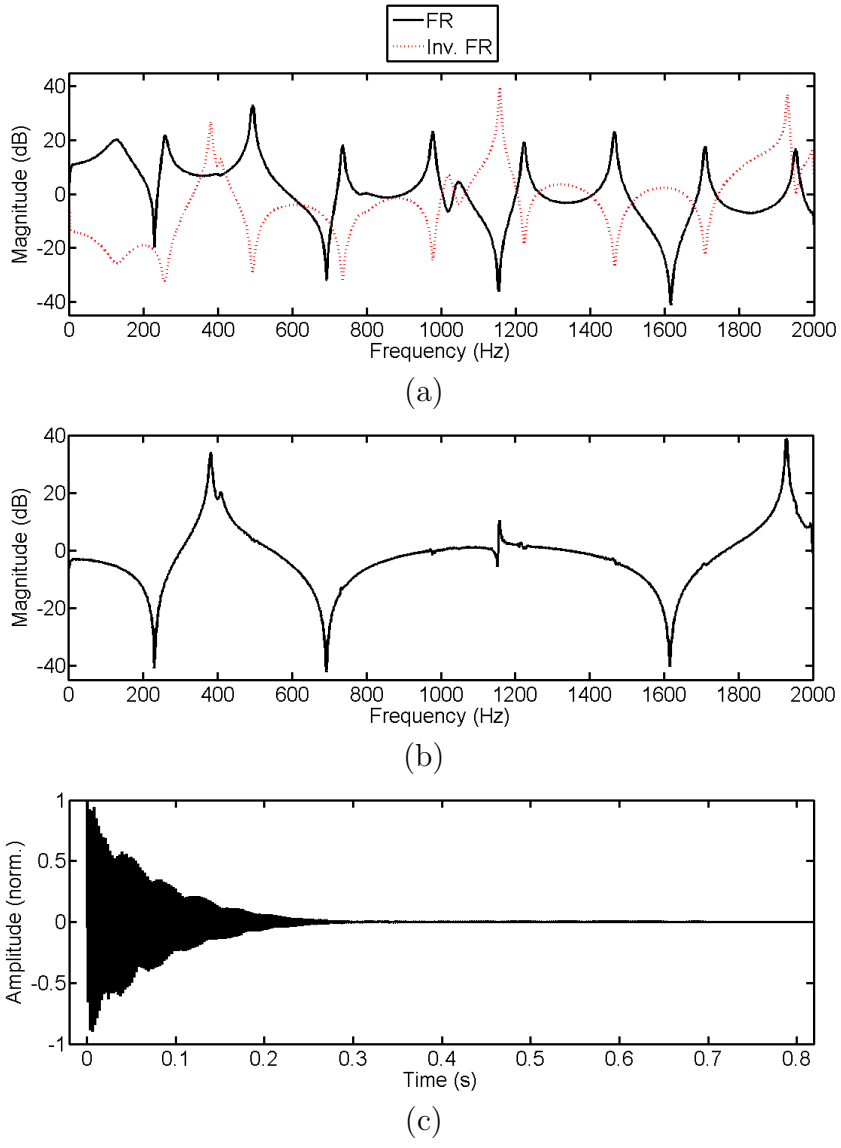


Figure 4.9 The impulse response at position 7 filtered by the sound pressure inverse filter computed at position 10. (a) The tube frequency response and the inverse filter frequency response magnitude. (b) The convolution result in the frequency domain. (c) The convolution result in the time domain.

exist at position 10 so there is an undesired boost at those frequencies caused by the mismatched inverse filter. The inverse filter spectrum does not have peaks that correspond to the nulls at position 10 so they are present in the equalized spectrum as new resonances shown in Fig. 4.10 (b).

Similar subjective observations are made regarding the audio quality of filtered program material as was made for the filter from position 10 used to deconvolve the TFR at position 7. Again, the equalized audio sounds significantly worse than the dry signal, and there is no improvement in clarity to the unequalized sound.

A sound-pressure-based equalizer cannot remove all resonances in an enclosure; the inverted TFR has nulls that do not exist at other locations so these frequencies are undesirably boosted at such locations. However, compensating for a resonance does not create a similar problem. If an inverse filter that compensates for a specific resonance is applied at a location where the resonance is not present due to a spatial node the filter only deepens the frequency null, which does not have a significant undesired effect on the perceived sound, as discussed in Sec. 1.4.4. This leads to a discussion about energy density equalization in the tube, wherein only spectral resonances are equalized.

4.1.2 Energy Density Equalization

Energy density TFRs for positions 7.5 and 10.5 are shown in Fig. 4.11 for position 7.5 and Fig. 4.12 for position 10.5 along with their impulse responses, inverse filter spectra, and the inverse filter time-domain coefficients. The frequency responses were generated by computing the FFTs of the impulse responses. The energy density impulse responses do not hold practical meaning for this research, but are shown here for completeness. The energy density spectra shown in Fig. 4.11 (b) and Fig. 4.12 (b) are nearly identical. They each show the various spectral resonances that are

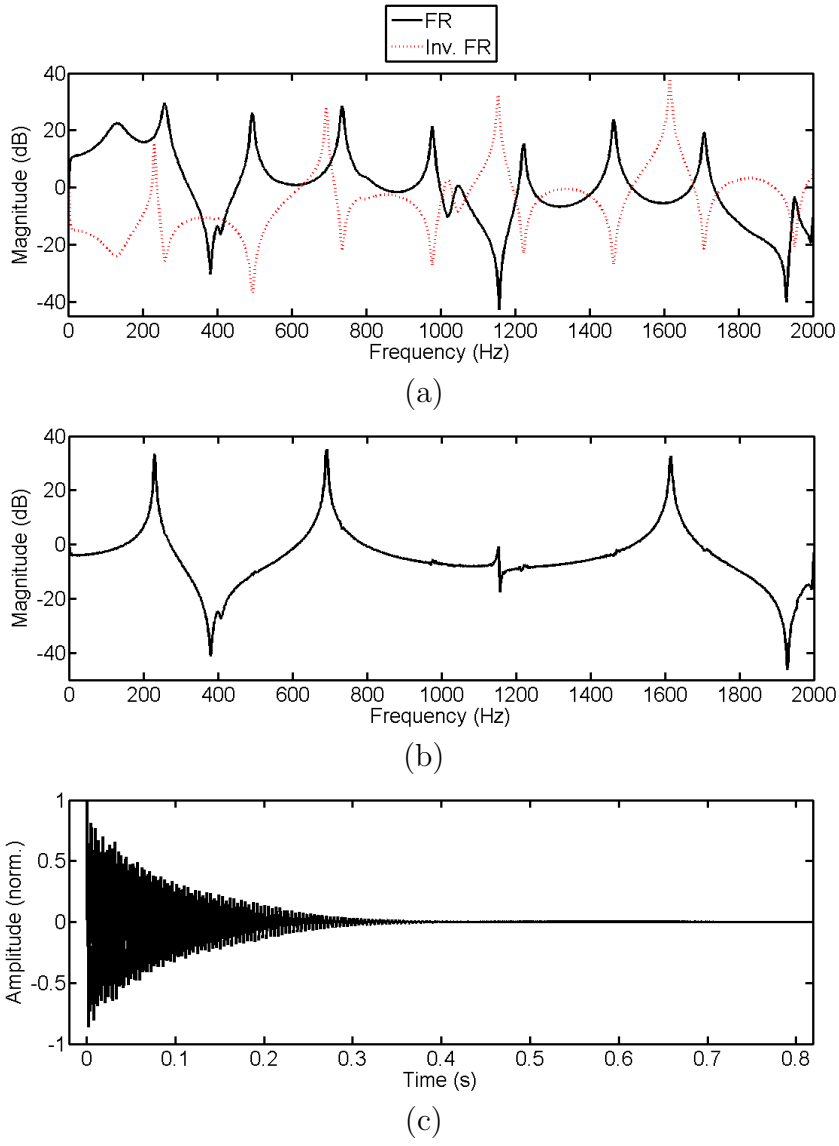


Figure 4.10 The impulse response at position 10 filtered by the sound pressure inverse filter computed at position 7. (a) The tube frequency response and the inverse filter frequency response magnitude. (b) The convolution result in the frequency domain. (c) The convolution result in the time domain.

not all observable in the pressure TFRs shown in Fig. 4.3 (b) and Fig. 4.4 (b). All resonances are identified because when energy density is measured at a given position, both potential and kinetic energies are detected at that location.

The total energy density in a one-dimensional sound field is nearly uniform, so at a point where a pressure minimum exists for a given frequency the particle velocity amplitude will be great and that frequency can be correctly detected with a particle velocity sensor. As a result, an energy density inverse filter does not attempt to compensate for sound pressure nulls that exist at the equalization position. As shown in the last section, such compensation leads to problems away from the equalization position. However, the inverse filters do have an upward slope to compensate for the gradual downward trend in the anechoic frequency response of the loudspeaker used in the experiment.

Figure 4.13 shows the time-domain convolution results of the TFR at position 7 deconvolved by the energy density inverse filter derived at position 7.5. The magnitude spectra of these two responses are shown in Fig. 4.13 (a). The energy density inverse filter spectrum has no frequency peaks, only nulls, save for one exception. The inverse filter response includes a peak that compensates for the loudspeaker resonance. These nulls correspond to each of the resonances in the ED spectrum at position 7. Fig. 4.13 (b) shows the magnitude TFR spectrum of the convolution result. All the resonances are equalized, but the inverse filter does not attempt to equalize any of the nulls. Thus, the nulls are still present at 231 Hz, 691 Hz, 1154 Hz, and 1615 Hz. However, an energy density inverse filter does provide the broad frequency boost necessary to equalize the downward sloping anechoic response of the loudspeaker, as well as the irregularity observed at the location of the speaker cavity resonance around 1030 Hz. The time-domain result is not an ideal delta function as seen in Fig. 4.13 (c). However, the response is significantly dereverberated. The dere-

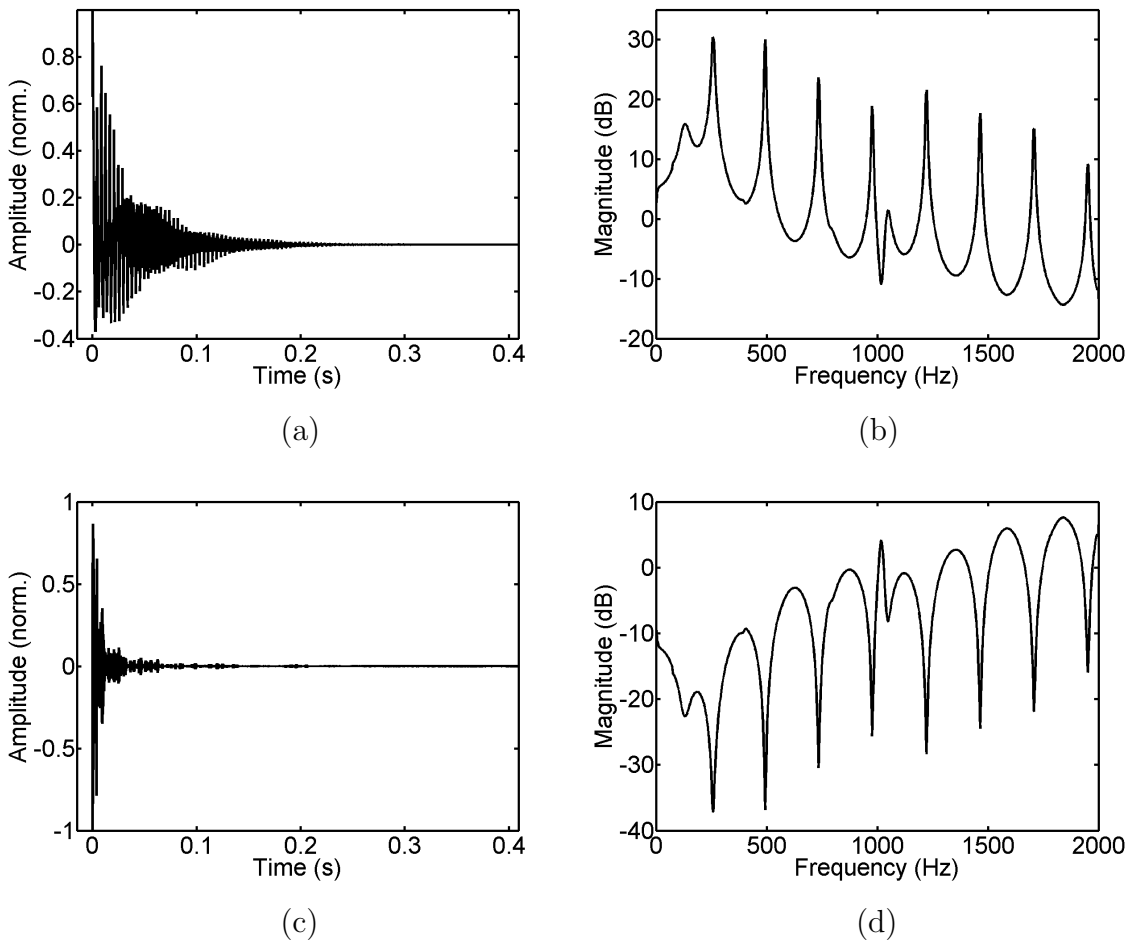


Figure 4.11 Energy density at microphone position 7.5. (a) The energy density impulse response. (b) The energy density frequency response magnitude. (c) The energy density inverse filter coefficients. (d) The energy density inverse filter magnitude.

verberated audio does not sound like the original recording because of the presence of spatial nulls.

Figure 4.14 shows similar results for the TFR at position 10 that is deconvolved by the energy density inverse filter derived at position 10.5. All resonance peaks are again accounted for, but the original frequency nulls caused by the sound pressure field at that location are still present at 387 Hz, 1156 Hz, and 1928 Hz. The results in Fig. 4.14 (b) and Fig. 4.14 (c) again show that the TFR is not perfectly equalized by

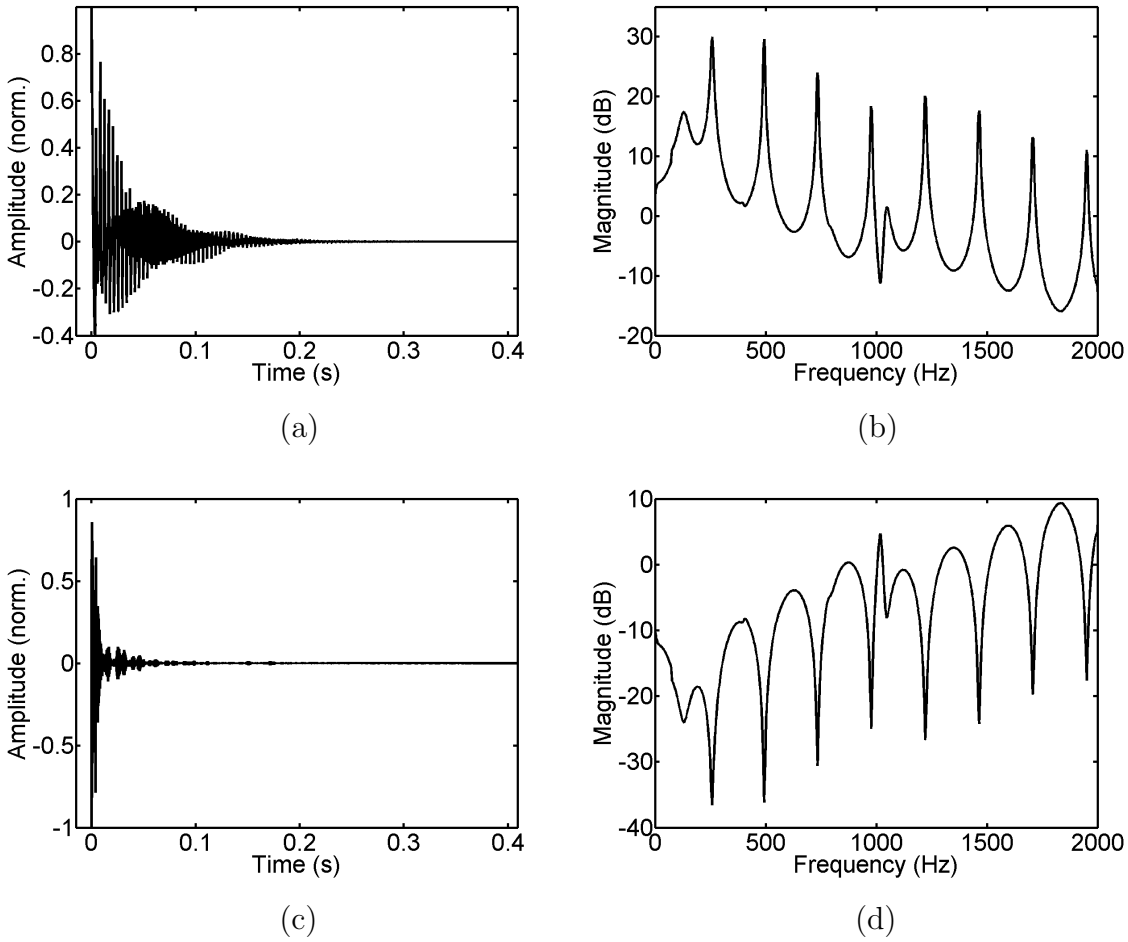


Figure 4.12 Energy density at microphone position 10.5. (a) The energy density impulse response. (b) The energy density frequency response magnitude. (c) The energy density inverse filter coefficients. (d) The energy density inverse filter magnitude.

the energy density inverse filter. The equalized audio sounds similar to the equalized audio at position 7, but there are perceivable differences between the two microphone positions because the spectral nulls occur at different frequencies.

Figure 4.15 shows the convolution result at position 7 using the energy density inverse filter response derived at position 10.5. The equalization result is nearly the same as if the energy density filter derived at position 7.5 was used. Also if the energy density inverse filter derived at position 7.5 was used to deconvolve the TFR

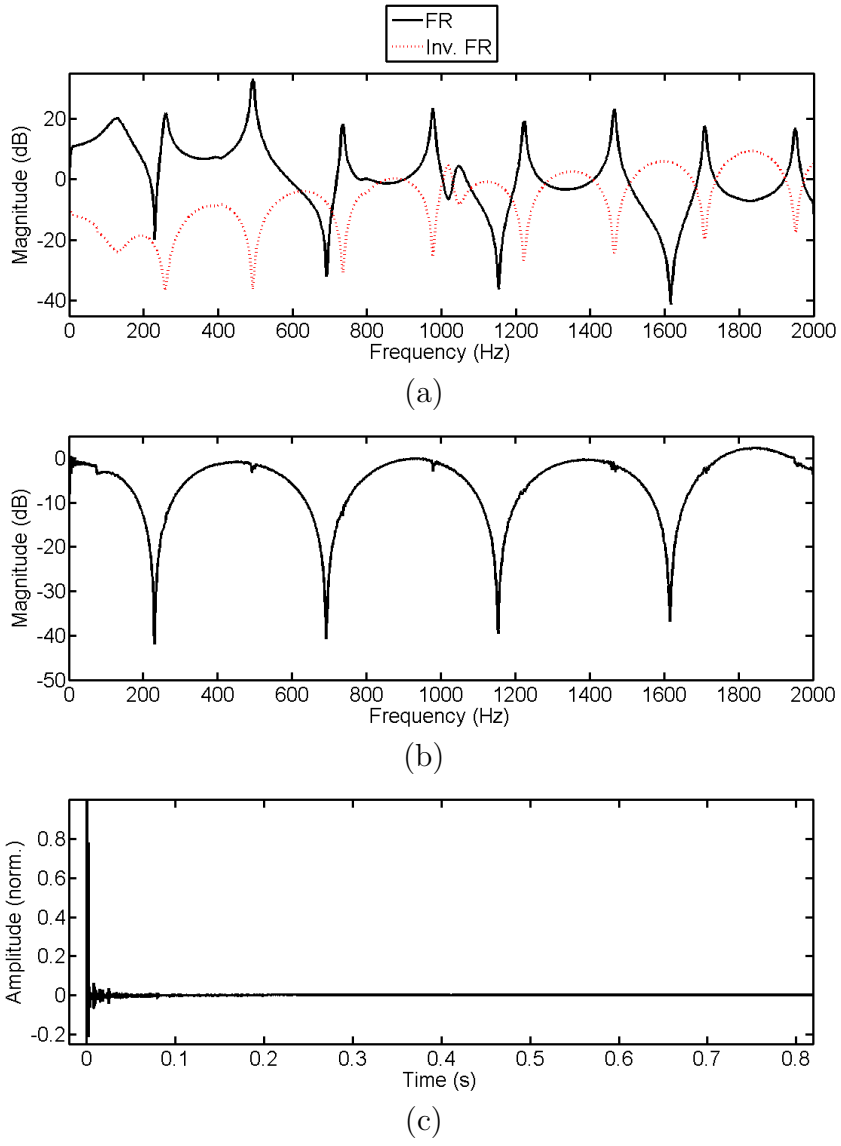


Figure 4.13 Energy density equalization at microphone position 7. (a) The tube frequency response magnitude at position 7 and the inverse filter frequency response magnitude at position 7.5. (b) The convolution result in the frequency domain. (c) The time-domain convolution result.

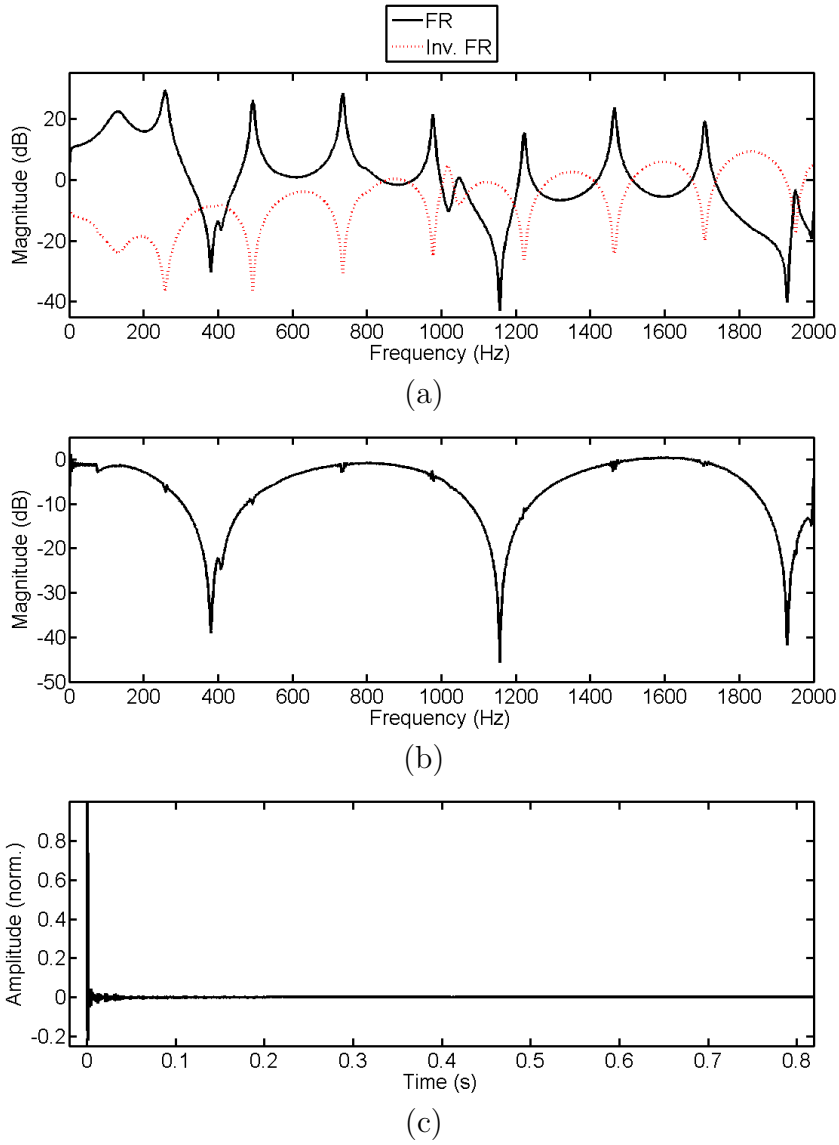


Figure 4.14 Energy density equalization at microphone position 10. (a) The tube frequency response magnitude at position 10 and the inverse filter frequency response magnitude at position 10.5. (b) The convolution result in the frequency domain. (c) The time-domain convolution result.

at position 10 the result is the same as if the filter derived at position 10.5 is used as shown in Fig. 4.16. This is where the advantage of energy density equalization is apparent. Even though the equalization result is not perfectly flat at any location in tube, undesirable resonances are also not excited anywhere in the tube. The global resonances are accurately detected by the energy density spectrum and equalized at all positions in a one-dimensional tube, regardless of where the energy density inverse filter is derived. However, the equalized response will still be spectrally different at every location in the sound field due to the presence of unequalized pressure minima at varying frequencies.

Figure 4.17 shows the sound pressure field equalized in real-time with an energy density inverse filter derived at position 7.5. These plots consist of transfer function computations at each of the microphone positions. Figure 4.17 (a) shows a surface plot of the measured sound field and Fig. 4.17 shows a top-down contour plot view of the field. All resonances are equalized in the enclosure, but the spatial nulls are still present.

4.1.3 Listening Test

A listening test was administered to fourteen volunteers. The test consisted of two experiments, each with the same two questions. In the first experiment, the volunteers heard two sound recordings from the tube reproduced over a single loudspeaker in a room. The excitation signal was a low-pass filtered ($f_c = 2000$ Hz) speech recording prefiltered by a sound pressure inverse filter derived at microphone position 7. The recordings were from positions 7 and 10. The volunteers received no explanation regarding the origin of the sound samples; they were merely asked to comment on their differences. Sample A, the recording at the point of equalization (position 7) was first played. Sample B, the recording at position 10, was subsequently played.

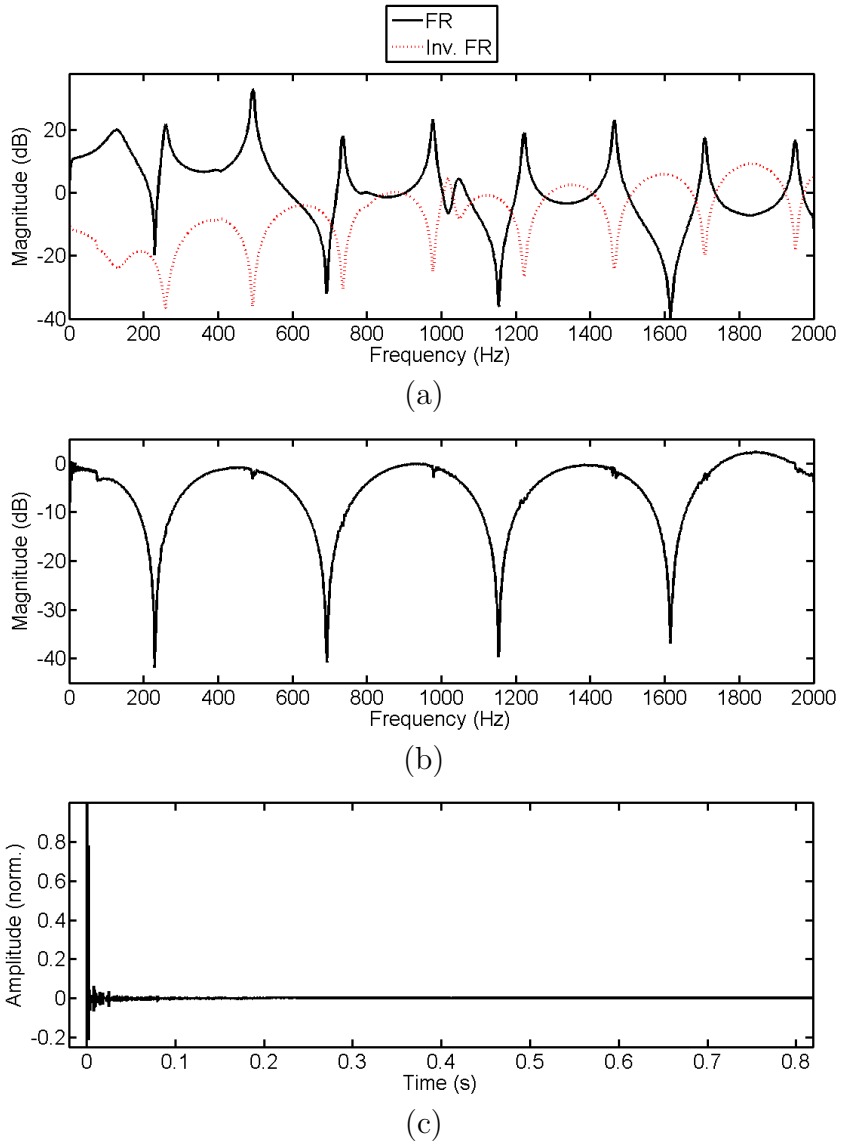


Figure 4.15 The impulse response at position 7 deconvolved by the energy density inverse filter computed at position 10.5. (a) The tube frequency response and the inverse filter frequency response magnitude. (b) The convolution result in the frequency domain. (c) The convolution result in the time domain.

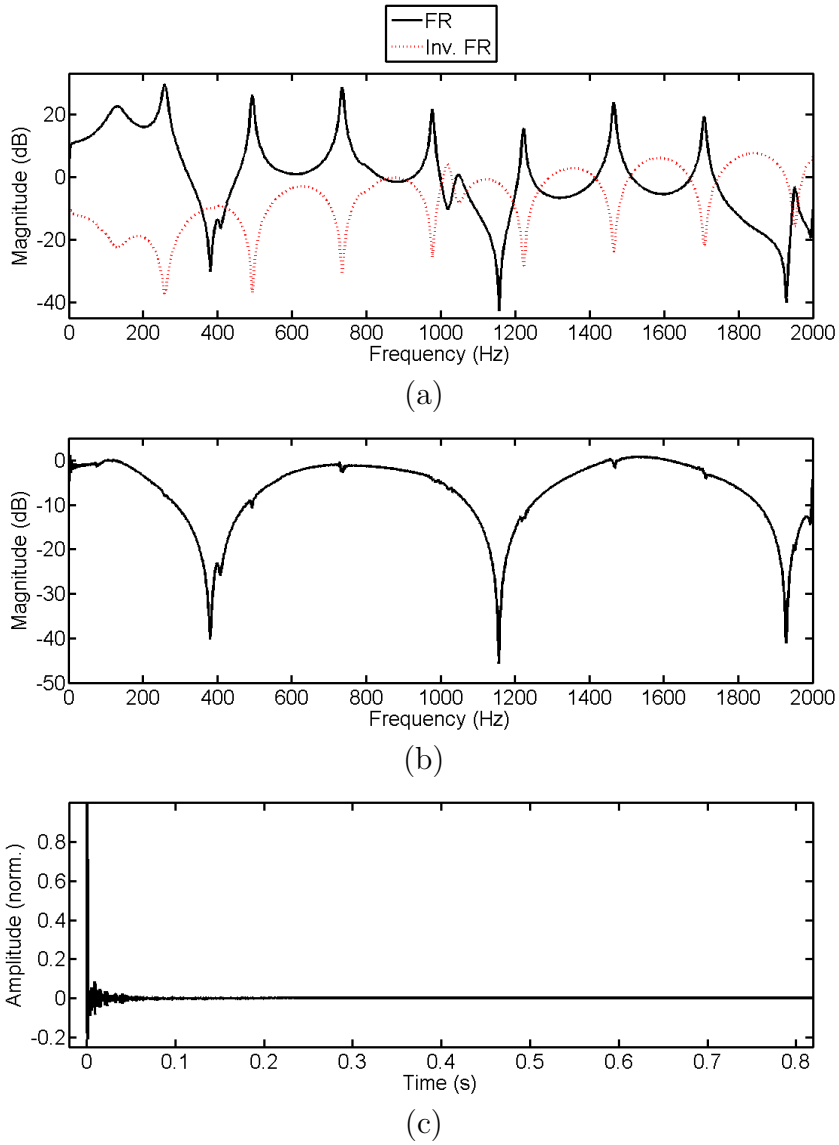
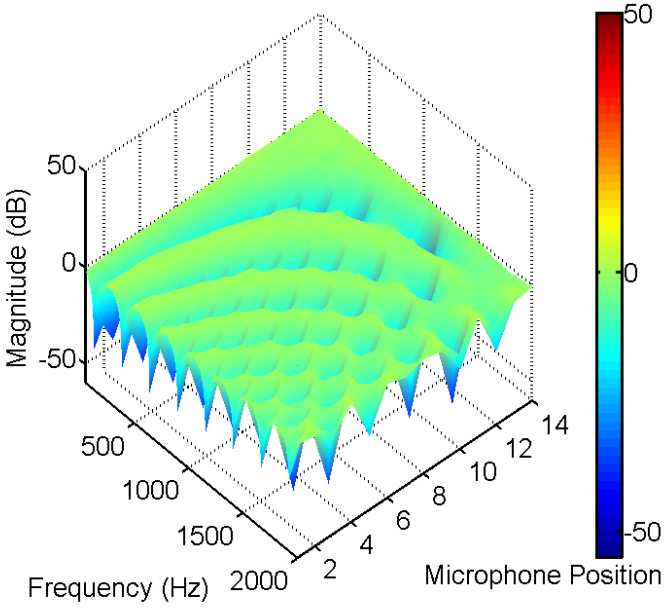
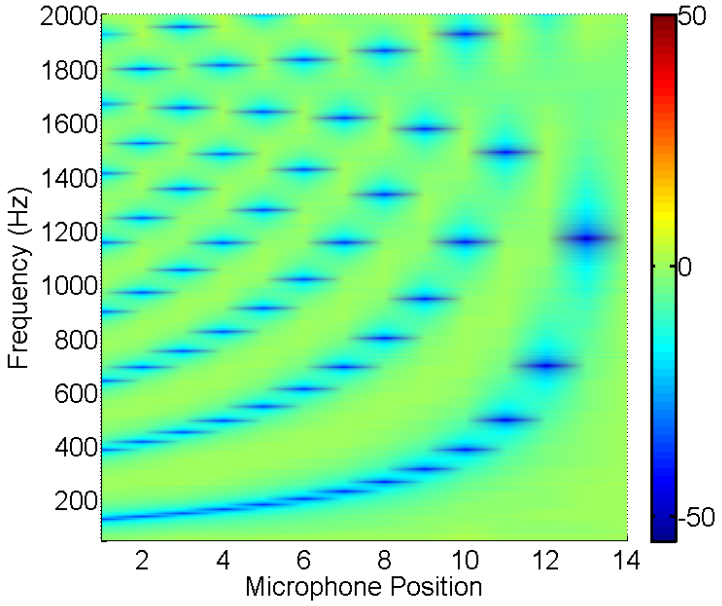


Figure 4.16 The impulse response at position 10 deconvolved by the sound pressure inverse filter computed at position 7.5. (a) The tube frequency response and the inverse filter frequency response magnitude. (b) The convolution result in the frequency domain. (c) The convolution result in the time domain.



(a)



(b)

Figure 4.17 Spatial-spectral plots of the pressure field in the plane-wave tube after being filtered in real time with an energy density inverse filter derived at position 7.5. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

Question 1 was whether sample B was more, less, or about as intelligible as sample A. Question 2 was whether sample B was more, less, or about as irritating as sample A. Each volunteer reported that sample B was less intelligible and more irritating.

In the second experiment, the volunteers again heard two recordings and were asked similar questions. This time sample A was recorded at position 7 in the sound field excited by the same bandlimited speech recording, but prefiltered with an energy density inverse filter derived at position 7.5. Sample B was recorded at microphone position 10. For the first question, the participants responded with the following percentages:

- 21% indicated that sample B was less intelligible than sample A
- 64% indicated that sample B was about as intelligible as sample A
- 15% indicated that sample B was more intelligible than sample A.

For the second question,

- 29% indicated that sample B was less irritating than sample A
- 57% indicated that sample B was about as irritating as sample A
- 14% indicated that sample B was more irritating than sample A.

The perceived differences between the two microphone positions for energy density equalization were not as polarized. The results indicate that 79% of participants perceived the sound farther away from the point of equalization about the same as or more intelligible than near the point of equalization. For irritability, 86% of the participants determined that the recording farther away from the point of equalization was about the same as or less irritating than near the point of equalization. These listening test results, although generated by a small focus group, demonstrate that

energy density equalization enhances the system response not only near the point of equalization, but also away from it. This is due to the equalization of global spectral effects in the one-dimensional sound field.

4.1.4 Summary

A pressure-based inverse filter is able to equalize sound at a point in a one-dimensional enclosure, but the same filter causes undesirable resonances to be excited at other locations. This happens because the filter attempts to compensate for pressure nulls at the point of equalization corresponding to spatial nodes. A sound pressure inverse filter is not able to equalize the TFR at any other location in the tube due to the mismatch of spectral nulls.

An energy density inverse filter attempts to equalize the total sound energy in an enclosure, and its spectrum accounts for all resonances and other global spectral effects of the tube and loudspeaker. At positions where pressure is at a minimum, the particle velocity amplitude is great and the total energy is still accurately detected at that position. Because energy density is uniform in a one-dimensional field, an energy density filter derived at any point in a one-dimensional field is nearly equal to one derived at any other point. It equalizes all global resonances but does not attempt to equalize the frequency nulls caused by pressure nodes. As a result, the TFR is not equalized to a perfectly flat response at any location, but the overall reverberation is significantly reduced due to the elimination of the undesirable long ringing resonances.

While it is true that the frequency spectrum in the tube after equalization by an energy density inverse filter is not completely flat, it is much more uniform over frequency away from the point of equalization. Energy density equalization is independent of position for a one-dimensional field. It equalizes not only the impact of

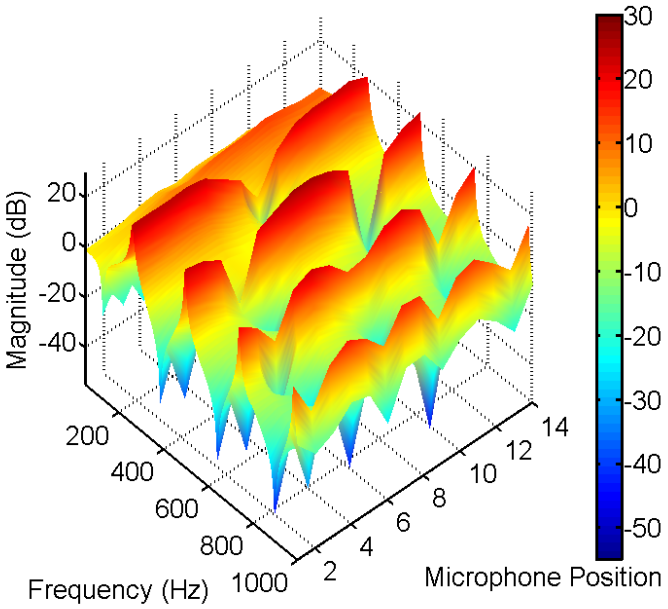
the acoustical enclosure, but also the electrical system. The remaining nulls are much less perceptible than resonances, as discussed in Sec. 1.4.4. Sound clips of the sound pressure and energy density equalization experiments discussed here are included in Appendix A.

4.2 Time-Varying Equalization Results

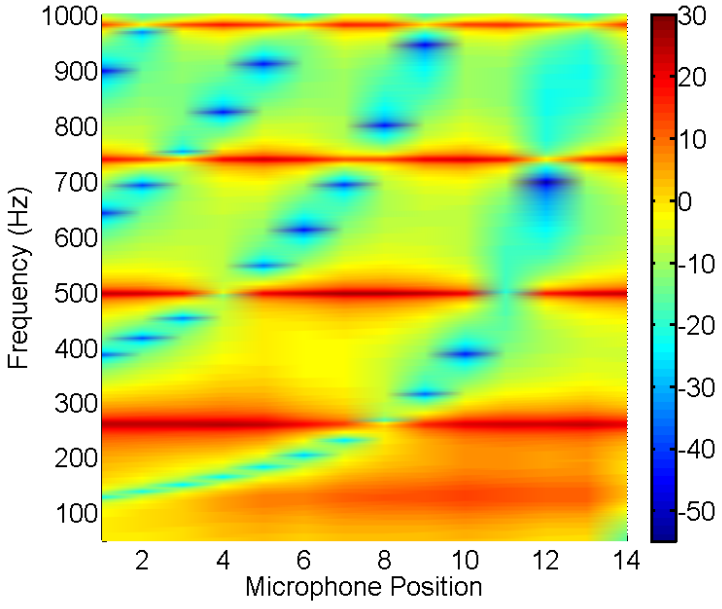
This section presents the result of three experiments using the time-adaptive energy density equalization filter presented in Sec. 2.5.4. The ability of the converged algorithm to equalize the sound field in the plane-wave tube is first evaluated. The rigid termination at the end of the tube is then removed and the system is allowed to converge for several minutes. After the algorithm has again converged, its ability to equalize the sound field in this altered system, a "closed-open" tube, is evaluated. Lastly, waterfall plots are presented that show the convergence of the filter over time as the closed end of the tube is opened, then closed once again. All the figures show the frequency response from 0 to 1000 Hz, with the two microphones used to update the filter located at positions 5 (23.2 cm from the source) and 6 (28.2 cm from the source).

4.2.1 Time-Adaptive Equalization in a Closed Tube

The sound pressure field in the plane-wave tube is shown in Fig. 4.18, with four notable tube resonances: 255, 495, 735, and 975 Hz. The frequency responses of the two transfer function estimates $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$ as well as that of $W(z)$ are shown in Fig. 4.19 (a). This figure shows that the time-adaptive filter correctly identifies and compensates for the tube resonances while not attempting to compensate for the spatial nulls present at either of the two microphone positions. The broad spectral



(a)



(b)

Figure 4.18 Spatial-spectral plots of the pressure field in the plane-wave tube excited with white noise. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

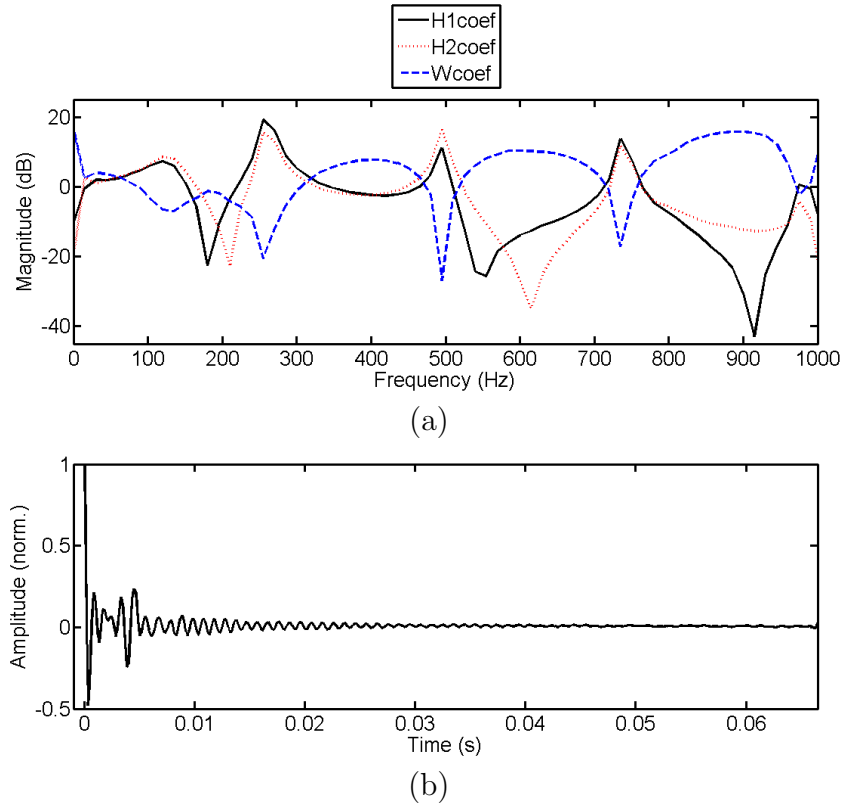
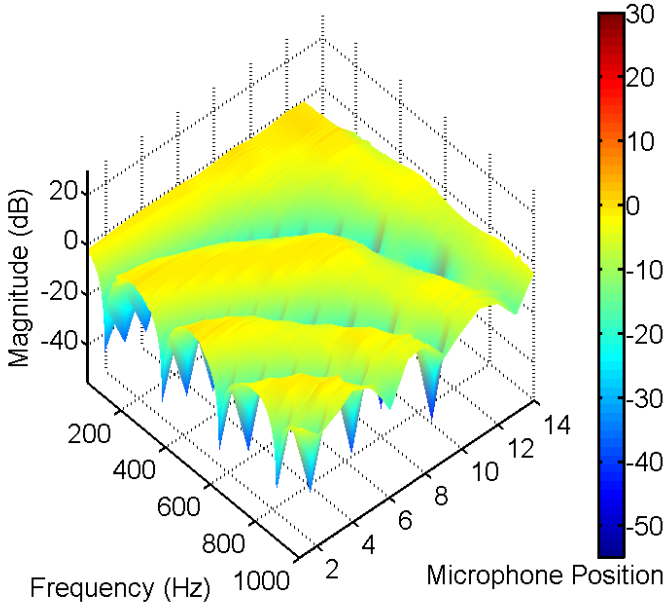


Figure 4.19 The converged time-adaptive energy density equalization filter for the closed plane-wave tube. (a) The W(z) frequency response compared to those of $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$. (b) The W(z) coefficients in the time domain.

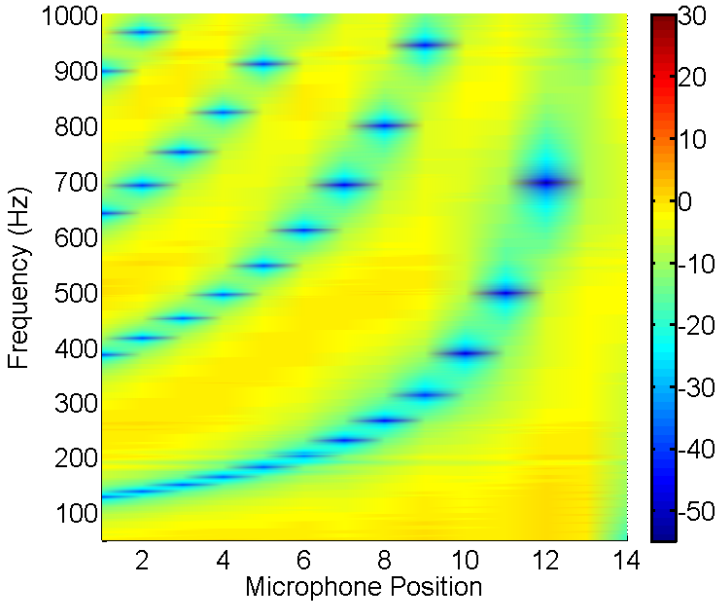
boost provided by the inverse filter to compensate for the downward slope of the anechoic loudspeaker response is also clearly visible. The equalized sound field in the tube is shown in Fig. 4.20. The time-adaptive energy density equalization filter compensates for all global tube resonances and other global spectral effects as explained in Sec. 4.1.2.

4.2.2 Time-Adaptive Equalization in an Open Tube

After the time-adaptive filter converged to the conditions in the tube, the rigid end cap was removed, so as to drastically change the sound pressure field, as shown in Fig. 4.21. In the new system the resonances are shifted to 90, 150, 360, 585, and

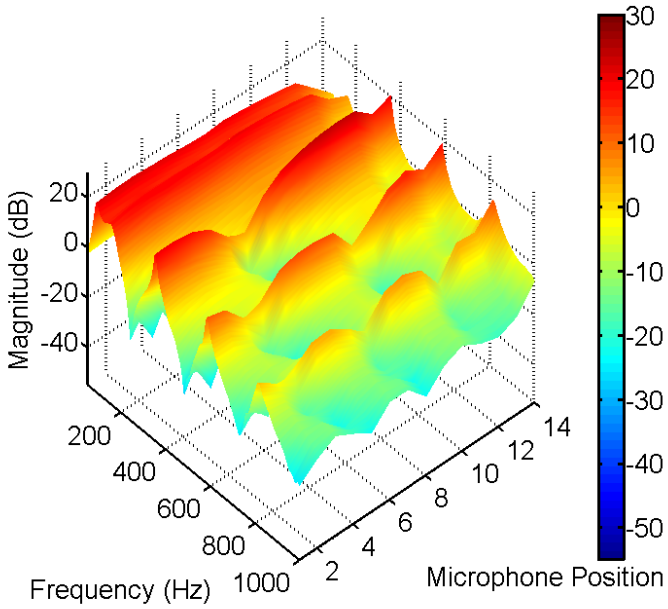


(a)

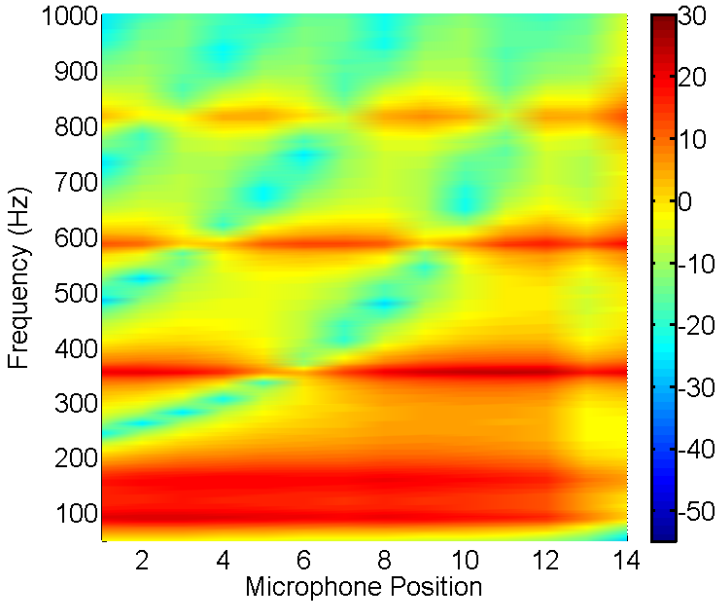


(b)

Figure 4.20 The pressure field in the plane-wave tube equalized with a time-adaptive energy density inverse filter derived at position 5.5. (a) Three dimensional view of the equalized sound field. (b) Microphone position vs. frequency.



(a)



(b)

Figure 4.21 Spatial-spectral plots of the pressure field in the plane-wave tube with the end removed and excited with white noise. (a) Surface plot of the sound field. (b) Contour plot of the sound field.

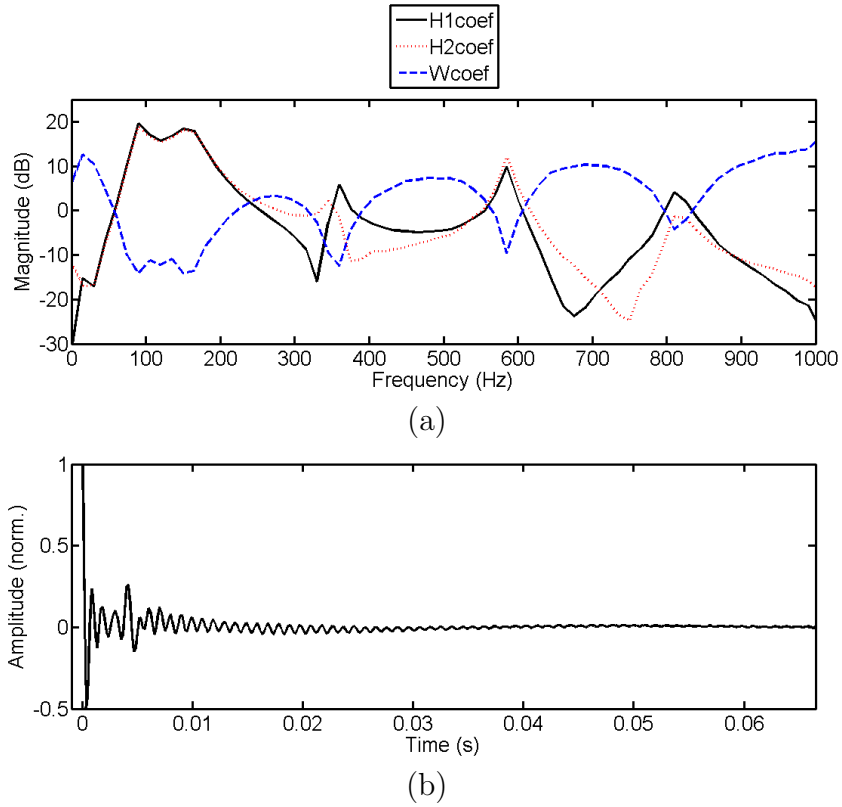
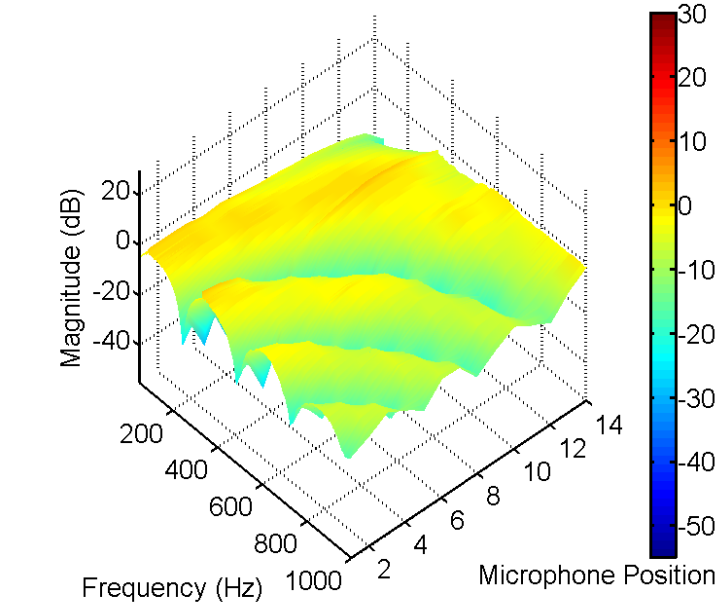
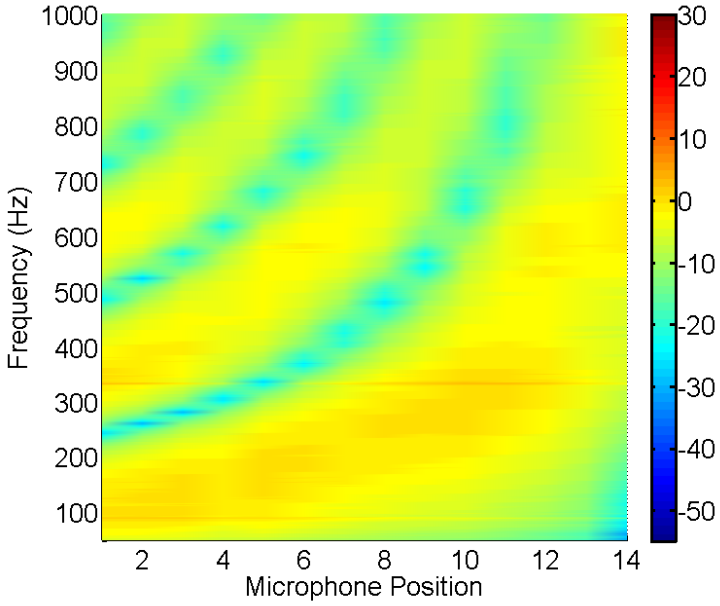


Figure 4.22 The converged time-adaptive energy density equalization filter for the open-ended plane-wave tube. (a) The W(z) frequency response compared to those of $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$. (b) The W(z) coefficients in the time domain.

810 Hz. The converged frequency responses of the two transfer function estimates $\tilde{H}_{p1}(z)$ and $\tilde{H}_{p2}(z)$ as well as that of $W(z)$ are shown in Fig. 4.22. These frequency responses are clearly different than those in Fig. 4.19; they have adapted to accurately describe the modified system. Again, the nulls in the $W(z)$ response compensate for all resonances and the gradual boost in magnitude compensates for the loudspeaker response. The equalized sound field is shown in Fig. 4.23. All resonances and other global spatial effects are equalized, but the filter does not attempt to compensate for any of the spatial nulls.



(a)



(b)

Figure 4.23 The pressure field in the plane-wave tube with the end removed equalized with a time-adaptive energy density inverse filter derived at position 5.5. (a) Three dimensional view of the equalized sound field. (b) Microphone position vs. frequency.

4.2.3 The Performance of the Filter Over Time

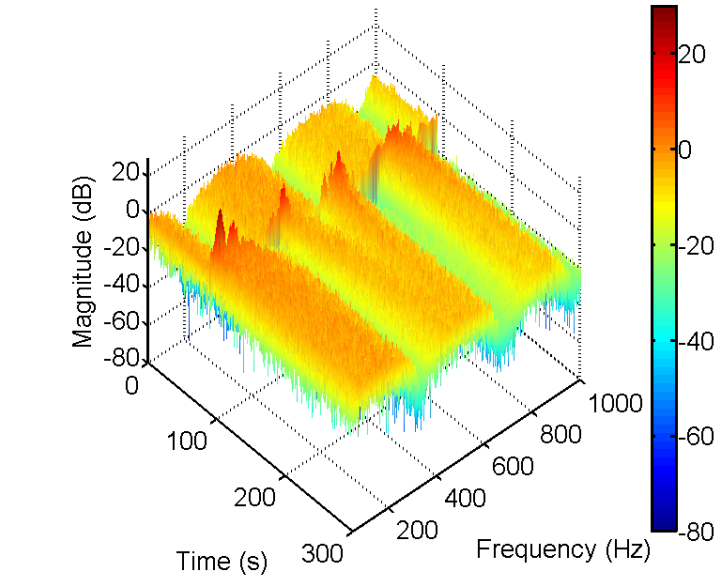
Tests were performed to evaluate the performance of the time-adaptive energy density equalization filter over time. This section presents results from two experiments: (1) how the system changes when the end of the tube is removed and (2) how the system changes when the rigid end is replaced.

The Tube Changed From a Closed-Closed to a Closed-Open System

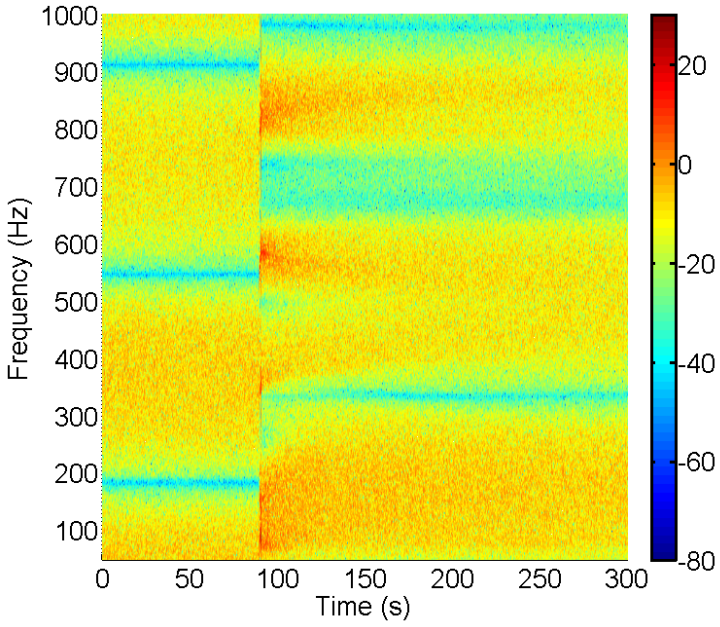
In the first experiment the microphone at position 5 records sound in the tube for five minutes. A series of frequency responses is computed from the sound pressure measurement to create a series of frequency responses over time. Figure 4.24 shows these frequency responses over time from 0 to 300 s. Over the time period 0 to 90 s, the system is in a steady state; all resonances are equalized and nulls exist at 180, 555, and 915 Hz. At 90 s, the rigid end of the tube is removed and the resonances of the new system at 90, 150, 360, 585, and 810 Hz form. The spatial nulls also shift to 330, 675, and 975 Hz. The time-adaptive filter adapts to the new system and compensates for the new resonances in approximately 175 seconds. It also provides a desirable boost at approximately 725 Hz as the inverse filter coefficients adjust to their optimal values. This can be seen in Fig. 4.24 (b).

The Tube Changed From a Closed-Open to a Closed-Closed System

In the case of the system being changed from an open tube back to a closed tube, the steady-state inverse filter initially equalizes all resonances as mentioned, with nulls at 330, 675, and 975 Hz. Figure 4.25 shows this changing system over a period of 300 seconds. As the end of the tube is placed back on at 90 seconds the tube resonances at 255, 495, 735, and 975 Hz temporarily appear and the nulls shift to 180, 555, and 915 Hz. Over time the time-adaptive filter coefficients adjust to



(a)



(b)

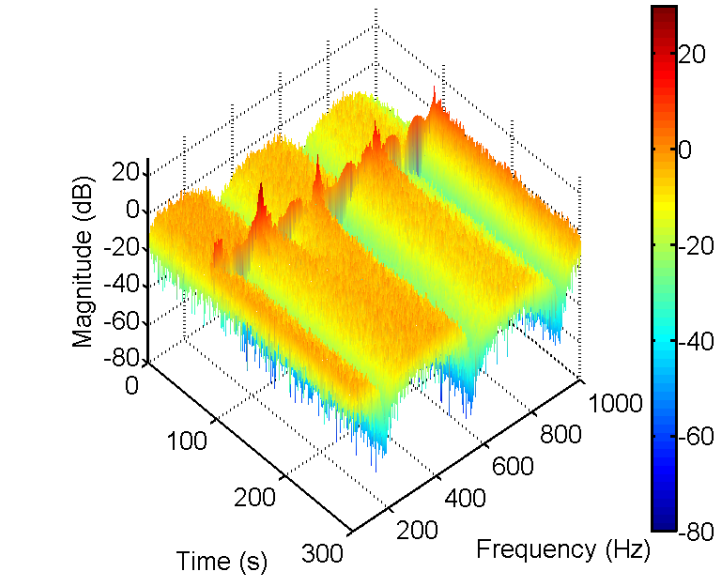
Figure 4.24 The time-varying pressure spectrum at position 5 in the plane-wave tube equalized with a time-adaptive energy density inverse filter derived at position 5.5 in real-time. The rigid cap of the tube is removed at 90 seconds. (a) Surface plot of the frequency response at position 5 over time. (b) contour plot of the frequency response at position 5 over time.

compensate for the new system resonances, and after approximately 200 seconds the resonances are removed from the system. The resonance at 975 is close to the cutoff frequency of the excitation signal and the filtered-x algorithm compensates for this resonance more slowly.

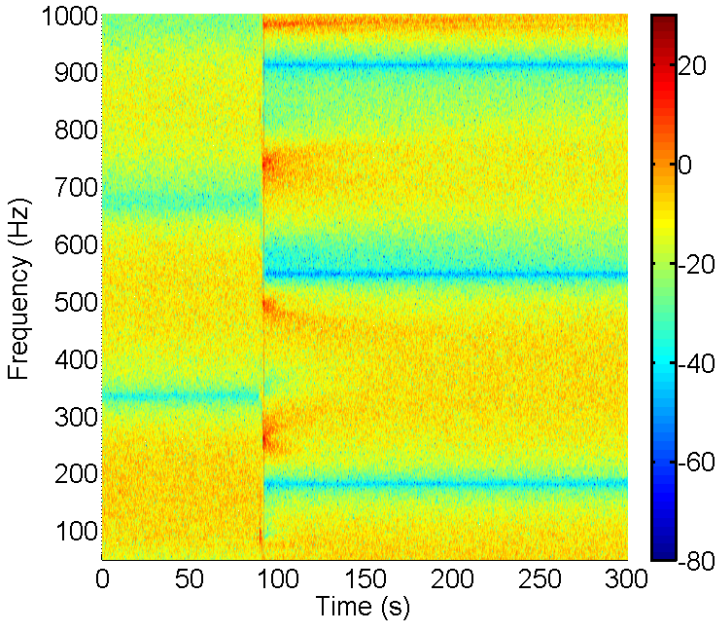
4.2.4 Time-Adaptive Equalization Summary

The filtered-x filter coefficient update scheme is used to implement a time-adaptive energy density equalization filter. This algorithm updates the $W(z)$ coefficients based on a current estimate of the energy density halfway between the sensor locations. Sound clips of the recordings used to generate Fig. 4.24 and Fig. 4.25 are included in Appendix A. The fig:ta-waterfallOpenToClosed algorithm also adapts to drastic changes in the system. This filter compensates for resonances and other global spatial effects in the system without attempting to equalize the spatially varying nulls. This method is subject to the limitations of the FIR LMS update: it models dips in the system more quickly than boosts, and the convergence parameter μ must be small enough for a specific configuration to ensure stability of the algorithm.

This time-adaptive equalizer was also used to prefilter program material such as pop music and speech. It was found that a lower value for μ is necessary for the algorithm to remain stable. This is especially important for the equalization of speech where many pauses are present while the algorithm attempts to make adjustments based on sound pressure measurements. Elliott and Nelson attribute this behavior to the degree of correlation between the filtered-x samples, which is not the case when a uncorrelated white noise excitation signal is used [25].



(a)



(b)

Figure 4.25 The time-varying pressure spectrum at position 5 in the plane-wave tube equalized with a time-adaptive energy density inverse filter derived at position 5.5 in real-time. The rigid cap of the tube is replaced at 90 seconds. (a) Surface plot of the frequency response at position 5 over time. (b) contour plot of the frequency response at position 5 over time.

Chapter 5

Conclusions

Control of acoustic energy density has been used for years in active noise control applications, and this research shows that the acoustic energy density field in an enclosure can also be successfully equalized for sound reproduction. The presented research results show that equalizing the acoustic energy density in a one-dimensional sound field compensates for all resonances and other global spectral effects of the tube and loudspeaker, whereas pressure-based equalization is typically effective only at the point of equalization. This chapter summarizes the significant research results and presents recommendations for future work.

5.1 Significant Research Results

A practical pressure-based inverse filter approximates the ideal inverse frequency response at a specific location in a sound field, compensating for the resonances and nulls detected at that position. The equalization result at that point is nearly flat in the frequency domain. However, a sound pressure inverse filter causes the frequencies corresponding to spatial nulls at the point of equalization to be undesirably boosted

elsewhere in the sound field, where those nulls do not exist at the same frequencies. These extreme boosts may also result in a reduction of the dynamic range of the audio system. A sound pressure inverse filter is thus only able to produce effective equalization at the point where it is derived, often making the response elsewhere in the sound field less acceptable. Generally, equalizing the sound pressure at a point in a one-dimensional sound field cannot equalize the field globally.

An energy density inverse filter offers the following benefits for a one-dimensional sound field:

- Because acoustic energy density is uniform in the field, an energy density equalization filter designed at one position is nearly the same as for any other position.
- An energy density equalization filter identifies and compensates for all resonances and other global spectral effects of the tube and the loudspeaker and does not attempt to equalize the spatially varying spectral nulls caused by local pressure nodes at the point of equalization.
- Because it primarily compensates for resonances, an energy density equalization filter has a significantly shorter time response length than a sound pressure equalization filter, which attempts to equalize both nulls and resonances.
- It also prevents reduction of dynamic range in an audio system that could result from large boosts in the inverse filter.

Because an energy density equalization filter does not attempt to undo any of the spatially dependent acoustic effects in a one-dimensional field, the equalized response does not result in a flat magnitude response at any position. However, the filter equalizes the global effects and the overall decay length in the enclosure is reduced through the elimination of undesirable resonances.

A method for time-adaptive energy density equalization demonstrated that a filter coefficient update method can accurately detect and compensate for changes in the resonances and other global spectral effects in an enclosure without attempting to compensate for the nulls at the point of equalization. Waterfall plots clearly demonstrate that the method adapts to drastic changes in the system over time and returns again to a steady-state equalization. This method is subject to all limitations associated with LMS algorithms.

The search for the best method of sound equalization has been pursued for several decades using both analog and digital filters. Many products have entered the market, which have implemented a type of sound pressure equalization with their own benefits and drawbacks. With the significant benefits that it has to offer, acoustic energy density equalization shows promise to overcome many of the major problems of pressure-based inverse filter design.

5.2 Recommendations for Future Work

Many opportunities exist to pursue this research further. As mentioned in Sec. 1.4.1, sound equalization using DSPs has been a subject of ongoing study for many years. There are many possibilities to choose from when designing an inverse filter (see Fig. 1.2). This research has focused on two of the possible options: the LS inverse filter and the time-adaptive filtered-x algorithm. Because acoustic energy density can be estimated in the time or frequency domain as discussed in Sec. 2.1.3 and Sec. 2.5.4, it would be a natural choice to study the performance of well-known frequency domain inverse filtering techniques such as regularization and complex smoothing applied to energy density equalization. Other time-adaptive methods could also be applied to equalize energy density, such as the filtered-x frequency-domain update scheme, or

the method proposed in Sec. 2.5. Due to the requirement of computing FFTs to perform frequency-domain computations in real time, these methods would be best implemented with a DSP system that uses block processing.

Energy density equalization can also be expanded to three-dimensional enclosures. However, this will provide its own new and unique challenges. The frequency response at a point in a room is more complex than a frequency response in a one-dimensional sound field. As a result, a finer frequency resolution may be necessary for the filter. Three-dimensional room equalization also places greater demands on hardware; two microphones are required to derive an estimate of particle velocity in a plane-wave tube, while at least four are required for a three-dimensional sound field, as shown in Eq. (2.17). The author recommends that further work be undertaken to explore these and other possibilities of energy density sound field equalization for audio applications.

References

[1] L. G. Johansen and R. Rubak, "The Excess Phase in Loudspeaker/Room Transfer Functions: Can it be Ignored in Equalization Tasks?," presented at the 100th convention of the Audio Engineering Society (1996), preprint 4181.

[2] R. Chester, *Error Sensor Strategies for Active Noise Control and Active Acoustic Equalization in a Free Field*, *MS Thesis* (Brigham Young University, Provo, UT, 2008).

[3] S. Bharitkar, P. Hilmes, and C. Kyriakakis, "Sensitivity of Multichannel Room Equalization to Listener Position," In *International Conference on Multimedia and Expo*, **1,** I–721–724 (2003).

[4] R. M. Howe and M. O. J. Hawksford, "Methods of Local Room Equalization and Their Effect Over the Listening Area," presented at the 91st convention of the Audio Engineering Society (1991), paper 3138.

[5] R. Genereux, "Adaptive Filters for Loudspeakers in Rooms," presented at the 93rd convention of the Audio Engineering Society (1992), paper 3375.

[6] M. R. Schroeder, "Schroeder Frequency Revisited," Journal of the Acoustical Society of America **99,** 3240 (1996).

[7] A. Makivirta, P. Antsalo, M. Karjalainen, and V. Valimaki, "Modal Equalization of Loudspeaker-Room Responses at Low Frequencies," Journal of the Audio Engineering Society **51,** 324–343 (2003).

[8] S. G. Norcross, G. A. Soulodre, and M. C. Lavoie, "Subjective Investigations of Inverse Filtering," Journal of the Audio Engineering Society **52,** 1003–1028 (2004).

[9] P. D. Hatziantoniou and J. N. Mourjopoulos, "Errors in Real-Time Room Acoustics Dereverberation," Journal of the Audio Engineering Society **52,** 883–899 (2004).

[10] M. Karjalainen, E. Piirila, A. Jarvinen, and J. Huopaniemi, "Comparison of Loudspeaker Equalization Methods Based on DSP Techniques," Journal of the Audio Engineering Society **47,** 14–31 (1999).

[11] J. N. Mourjopoulos, "Digital Equalization of Room Acoustics," Journal of the Audio Engineering Society **42,** 884–900 (1994).

[12] P. Rubak and L. G. Johansen, "Design and Evaluation of Digital Filters Applied to Loudspeaker/Room Equalization," presented at the 108th convention of the Audio Engineering Society (2000), paper 5172.

[13] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 209.

[14] J. Mourjopoulos, P. Clarkson, and J. Hammond, "A Comparative Study of Least-Squares and Homomorphic Techniques for the Inversion of Mixed Phase Signals," In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, **7,** 1858–1861 (1982).

[15] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 768.

[16] S. T. Neely and J. B. Allen, "Invertibility of a Room Inverse Response," Journal of the Acoustical Society of America **66,** 165–169 (1979).

[17] J. Kates, "Digital Analysis of Loudspeaker Performance," In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, **2,** 377–380 (1977).

[18] D. Preis, "Phase Distortion and Phase Equalization in Audio Signal Processing - A Tutorial Review," Journal of the Audio Engineering Society **30,** 774–794 (1982).

[19] P. D. Bauman, S. P. Lipshitz, and J. Vanderkooy, "Cepstral Techniques for Transducer Measurement: Part II," presented at the 76th convention of the Audio Engineering Society (1985), preprint 2172.

[20] J. K. Hammond and J. Mourjopoulos, "Cepstral Methods Applied to the Analysis of Room Impulse Response," presented at the Institute of of Acoustics Autumn Conference pp. 51–54 (1980).

[21] O. Kirkeby, F. Orduna, P. A. Nelson, and H. Hameda, "Inverse Filtering in Sound Reproduction," Measurement and Control **29,** 261–266 (1993).

[22] J. Mourjopoulos, "On the Variation and Invertibility of Room Impulse Response Functions," Journal of Sound and Vibration **102,** 217–228 (1985).

[23] C. P. Boner and C. R. Boner, "A Procedure for Controlling Room-Ring Modes and Feedback Modes in Sounds Systems with Narrow-Band Filters," Journal of the Audio Engineering Society **13,** 297–299 (1965).

[24] W. K. Conner, "Theoretical and Practical Considerations in the Equalization of Sounds Systems," Presented at the 17th annual meeting of the Audio Engineering Society (1965), preprint 400.

[25] S. J. Elliott and P. A. Nelson, "Multiple-Point Equalization in a Room Using Adaptive Digital Filters," Journal of the Audio Engineering Society **37,** 899–907 (1989).

[26] P. G. Craven and M. A. Gerzon, "Practical Adaptive Room and Loudspeaker Equalizer for Hi-Fi Use," presented at the 7th Audio Engineering Society UK DSP Conference (1992), paper DSP-12.

[27] L. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders, *Fundamentals Of Acoustics* (John Wiley and Sons, New York, NY, 1999).

[28] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999).

[29] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing : Principles, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, NJ, 1996).

[30] J. S. Bendat and J. S. Piersol, *Random Data Analysis and Measurement Procedures* (John Wiley and Sons, New York, NY, 2000).

[31] M. H. Hayes, *Statistical Digital Signal Processing and Modeling* (John Wiley and Sons, Hoboken, NJ, 1996).

[32] L. D. Fielder, "Analysis of Traditional and Reverberation-Reducing Methods of Room Equalization," Journal of the Audio Engineering Society **51,** 3–26 (2003).

[33] P. D. Hatziantoniou and J. N. Mourjopoulos, "Results for Room Acoustics Equalisation Based on Smoothed Responses," presented at the 114th convention of the Audio Engineering Society (2003), paper 5779.

[34] P. D. Hatziantoniou and J. N. Mourjopoulos, "Real-Time Room Equalization based on Complex Smoothing: Robustness Results," presented at the 116th convention of the Audio Engineering Society (2004), paper 6070.

[35] S. J. Elliott, L. P. Bhatia, F. S. Deghan, A. H. Fu, M. S. Stewart, and D. W. Wilson, "Practical Implementation of Low-Frequency Equalization Using Adaptive Digital Filters," Journal of the Audio Engineering Society **42,** 988–998 (1994).

[36] S. Bharitkar, P. Hilmes, and C. Kyriakakis, "Robustness of Spatial Average Equalizations: A Statistical Reverberation Model Approach," Journal of the Acoustical Society of America **116,** 3491–3497 (2004).

[37] B. D. Radlovic, R. C. Williamson, and R. A. Kennedy, "Equalization in an Acoustic Reverberant Environment: Robustness Results," IEEE Transactions on Speech and Audio Processing **8,** 311–319 (2000).

[38] A. H. Benade, "From Instrument to Ear In a Room: Direct or via Recording," Journal of the Audio Engineering Society **33,** 218–233 (1985).

[39] H. Kuttruff, *Room Acoustics*, 4th ed. (Taylor and Francis, New York, NY, 2000).

[40] A. O. Santillan, C. S. Pedersen, and M. Lydolf, "Experimental Implementation of a Low-Frequency Global Sound Equalization Method Based on Free Field Propagation," Applied Acoustics **68,** 1063–1085 (2007).

[41] L.-H. Kim, J.-S. Lim, and K.-M. Sung, "A New Robust Acoustic Crosstalk Cancellation Method With Sum and Difference Filter for 3D Audio System," IEICE

Transactions on Fundamentals of Electronics, Communications and Computer Sciences **E85-A,** 2159–2163 (2002).

[42] D. B. Ward and G. W. Elko, "New Robust System for 3D Audio Using Loudspeakers," In *IEEE International Conference on Acoustics, Speech and Signal Processing*, **2,** 781–784 (2000).

[43] J. Mourjopoulos, "Digital Equalization Methods for Audio Systems," presented at the 84th convention of the Audio Engineering Society (1988), preprint 2598.

[44] M. Karjalainen, P. A. A. Esquef, P. Antsalo, A. Makivirta, and V. Valimaki, "Frequency-Zooming ARMA Modeling of Resonant and Reverberant Systems," Journal of the Audio Engineering Society **50,** 1012–1029 (2002).

[45] B. P. Lathi, *Signal Processing and Linear Systems* (Oxford University Press, New York, NY, 1998).

[46] F. T. Agerkvist, "Time-Frequency Auditory Model Using Wavelet Packets," Journal of the Audio Engineering Society **44,** 37–50 (1996).

[47] A. Oppenheim, D. Johnson, and K. Steiglitz, "Computation of Spectra With Unequal Resolution Using The Fast Fourier Transform," **59,** 299–301 (1971).

[48] H. W. Strube, "Linear Prediction on a Warped Frequency Scale," Journal of the Acoustical Society of America **68,** 1071–1076 (1980).

[49] G. Evangelista and S. Cavaliere, "Discrete-Time Frequency Warped Wavelet Transforms," In *IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, **3,** 2105–2108 (1997).

[50] S. Malvar, *Signal Processing with Lapped Transforms* (Artech House, NorWood, MA, 1992).

[51] P. P. Vadyanathan, *Multirate Systems and Filter Banks* (Prentice Hall, Englewood Cliffs, NJ, 1993).

[52] M. Bobrek and D. B. Koch, "Music Signal Segmentation Using Tree-Structured Filter Banks," Journal of the Audio Engineering Society **46,** 412–427 (1998).

[53] S. K. Mitra, "Structural Subband Decomposition: A New Concept in Digital Signal Processing," In *IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, **1,** 31–34 (1997).

[54] N. Martin, J. Mars, J. Martin, and C. Chorier, "A Capon's Time-Octave Representation Application in Room Acoustics," IEEE Transactions on Signal Processing **43,** 1842–1854 (1995).

[55] M. D. Ortigueira, "Fractional Discrete-Time Linear Systems," In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, **3,** 2241–2244 (1997).

[56] M. Karjalainen, T. Paatero, J. N. Mourjopoulos, and P. D. Hatziantoniou, "About Room Response Equalization and Dereverberation," In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 183–186 (New Paltz, NY, United States, 2005).

[57] P. D. Hatziantoniou and J. N. Mourjopoulos, "Generalized Fractional-Octave Smoothing of Audio and Acoustic Responses," Journal of the Audio Engineering Society **48,** 259–280 (2000).

[58] J. W. Worley, P. D. Hatziantoniou, and J. N. Mourjopoulos, "Subjective Assessments of Real-Time Room Dereverberation and Loudspeaker Equalization," presented at the 118th convention of the Audio Engineering Society (2005), paper 6461.

[59] A. Maamar, I. Kale, and B. Daoud, "Room equalization based on iterative simple complex smoothing of acoustic impulse responses," In *IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, **5,** 109–112 (2006).

[60] S. Bharitkar, C. Kyriakakis, and T. Holman, "Variable-Octave Complex Smoothing for Loudspeaker-Room Response Equalization," International Conference on Consumer Electronics 2008, Digest of Technical Papers (2008).

[61] H. Kuttruff, "Sound Fields in Small Rooms," presented at the 15th international conference of the Audio Engineering Society (1998), paper 15-002.

[62] R. Buecklein, "Audibility of Frequency Response Irregularities," Journal of the Audio Engineering Society **29,** 126–131 (1981).

[63] P. A. Fryer, "Intermodulation Distortion Listening Tests," presented at the 50th convention of the Audio Engineering Society (1975), paper L-10.

[64] F. E. Toole and S. E. Olive, "Modification of Timbre by Resonances: Perception and Measurement," Journal of the Audio Engineering Society **36,** 122–142 (1988).

[65] S. E. Olive, P. L. Schuck, J. G. Ryan, S. L. Sally, and M. E. Bonneville, "Detection Thresholds of Resonances at Low Frequencies," Journal of the Audio Engineering Society **45,** 116–128 (1997).

[66] O. Kirkeby and P. A. Nelson, "Digital Filter Design for Inversion Problems in Sound Reproduction," Journal of the Audio Engineering Society **47,** 583–595 (1999).

[67] O. Kirkeby, P. A. Nelson, H. Hamada, and F. Orduna-Bustamante, "Fast Deconvolution of Multichannel Systems Using Regularization," IEEE Transactions on Speech and Audio Processing **6,** 189–194 (1998).

[68] P. Damaske, "Head-Related Two-Channel Stereophony With Loudspeaker Reproduction," **50,** 1109–15 (1971).

[69] D. Griesinger, "Equalization and Spatial Equalization of Dummy-Head Recordings for Loudspeaker Reproduction," Journal of the Audio Engineering Society **37,** 20–29 (1989).

[70] H. Moller, "Reproduction of Artificial-Head Recordings Through Loudspeakers," Journal of the Audio Engineering Society **37,** 30–33 (1989).

[71] D. H. Cooper and J. L. Bauck, "Prospects for Transaural Recording," Journal of the Audio Engineering Society **37,** 3–19 (1989).

[72] P. A. Nelson, H. Hamada, and S. J. Elliott, "Adaptive Inverse Filters for Stereophonic Sound Reproduction," IEEE Transactions on Signal Processing **40,** 1621–1632 (1992).

[73] P. A. Nelson, F. Orduna-Bustamante, and H. Hamada, "Multichannel Signal Processing Techniques in the Reproduction of Sound," Journal of the Audio Engineering Society **44,** 973–989 (1996).

[74] S. Bharitkar and C. Kyriakakis, "A Classification Scheme for Acoustical Room Responses," In *Sixth International Symposium on Signal Processing and its Applications*, **2,** 671–674 vol.2 (2001).

[75] L.-H. Kim, J.-S. Lim, C. Choi, and K.-M. Sung, "Equalization of Low Frequency Response in Automobile," IEEE Transactions on Consumer Electronics **49,** 243–252 (2003).

[76] M. R. Schroeder, "Models of Hearing," In *Proceedings of the IEEE*, **63,** 1332–1350 (1975).

[77] B. Widrow and S. D. Stearns, *Adaptive Signal Processing* (Prentice Hall, Englewood cliffs, NJ, 1985).

[78] R. Walker, "Equalisation of Room Acoustics and Adaptive Systems in the Equalisation of Small Room Acoustics," presented at the 15th international conference of the Audio Engineering Society (1998), paper 15-005.

[79] A. D. Pierce, *Acoustics: An Introduction to Its Physical Principles and Applications*, 4th ed. (Acoustical Society of America, Melville, NY, 1994), p. 14.

[80] A. D. Pierce, *Acoustics: An Introduction to Its Physical Principles and Applications*, 4th ed. (Acoustical Society of America, Melville, NY, 1994), p. 15.

[81] L. L. Beranek, *Acoustics*, 2nd ed. (Acoustical Society of America, Melville, NY, 1993), p. 12.

[82] L. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders, *Fundamentals Of Acoustics* (John Wiley and Sons, New York, NY, 1999), p. 124.

[83] A. D. Pierce, *Acoustics: An Introduction to Its Physical Principles and Applications*, 4th ed. (Acoustical Society of America, Melville, NY, 1994), p. 39.

[84] L. Kinsler, A. R. Frey, A. B. Coppens, and J. V. Sanders, *Fundamentals Of Acoustics* (John Wiley and Sons, New York, NY, 1999), p. 125.

[85] A. D. Pierce, *Acoustics: An Introduction to Its Physical Principles and Applications*, 4th ed. (Acoustical Society of America, Melville, NY, 1994), p. 40.

[86] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 674.

[87] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 675.

[88] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 781.

[89] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing* (Prentice Hall, Upper Saddle River, NJ, 1975), p. 511.

[90] S. L. J. Marple, "Computing the discrete-time 'analytic' signal via FFT," Conference Record of the Asilomar Conference on Signals, Systems & Computers **2,** 1322 – 1325 (1997).

[91] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 2nd ed. (Prentice Hall, Upper Saddle River, NJ, 1999), p. 674.

[92] T. Bose, *Digital Signal And Image Processing* (John Wiley and Sons, New York, NY, 2004).

[93] M. H. Hayes, *Statistical Digital Signal Processing and Modeling* (John Wiley and Sons, Hoboken, NJ, 1996), pp. 169–174.

[94] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing : Principles, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, NJ, 1996), pp. 711–719.

[95] M. H. Hayes, *Statistical Digital Signal Processing and Modeling* (John Wiley and Sons, Hoboken, NJ, 1996), p. 172.

[96] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing : Principles, Algorithms, and Applications* (Prentice Hall, Englewood Cliffs, NJ, 1996), p. 716.

[97] P. Clarkson, J. Mourjopoulos, and J. Hammond, "spectral, Phase, and Transient Equalization for Audio Systems," Journal of the Audio Engineering Society **33,** 127–132 (1985).

[98] J. N. Mourjopoulos, E. D. Kyriakis-Bitzaros, and C. E. Goutis, "Theory and Real-Time Implementation of Time-Varying Digital Audio Filters," Journal of the Audio Engineering Society **38,** 523–536 (1990).

[99] U. Zoelzer, B. Redmer, and J. Bucholtz, "Strategies for Switching Digital Audio Filters," presented at the 95th convention of the Audio Engineering Society (1993), preprint 3714.

[100] C. Hanna, "real-Time Control of DSP Parametric Equalizers," presented at the 13th international conference of the Audio Engineering Society (1994), paper 13-041.

[101] Y. Ding and D. Rossum, "Filter Morphing of Parametric Equalizers and Shelving Filters for Audio Signal Processing," Journal of the Audio Engineering Society **43,** 821–826 (1995).

[102] V. Valimaki and T. Laakso, "Suppression of transients in variable recursive digital filters with a novel and efficient cancellation method," IEEE Transactions on Signal Processing **46,** 3408–3414 (1998).

[103] R. Clark, E. Ifeachor, and G. Rogers, "Filter Morphing - Topologies, Signals and Sampling Rates," presented at the 113th convention of the Audio Engineering Society (2002), paper 5661.

[104] J. Laroche, "On the Stability of Time-Varying Recursive Filters," Journal of the Audio Engineering Society **55,** 460–471 (2007).

[105] S. M. Kuo and D. R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (John Wiley and Sons, New York, NY, 1996).

[106] S. Elliott, *Signal Processing for Active Control* (Academic Press, San Diego, CA, 2001).

[107] K. Gee, *Multi-channel active control of axial cooling fan noise*, *MS Thesis* (Brigham Young University, Provo, UT, 2002).

[108] B. Faber, *Active minimization of acoustic energy density in a mock tractor cab*, *MS Thesis* (Brigham Young University, Provo, UT, 2004).

[109] B. Monson, *Optimization of active noise control for small axial cooling fans*, *MS Thesis* (Brigham Young University, Provo, UT, 2006).

[110] K. M. Reichard and D. C. Swanson, "Frequency-Domain Implementation of the Filtered-x Algorithm with On-Line System Identification," Second Conference on Recent Advances in Active Control of Sound and Vibration pp. 562–573 (1993).

[111] J. J. Shynk, "Frequency-domain and multirate adaptive filtering," IEEE Signal Processing Magazine **9,** 14 – 37 (1992).

[112] R. M. Aarts, A. W. M. Mathijssen, P. C. W. Sommen, and J. Garas, "Efficient Block Frequency Domain Filtered-X Applied to Phantom Sound Source Gen-

eration," presented at the 104th convention of the Audio Engineering Society (1998), paper 4650.

[113] B. Rafaely and S. Elliot, "A computationally efficient frequency-domain LMS algorithm with constraints on the adaptive filter," IEEE Transactions on Signal Processing **48,** 1649–1655 (2000).

[114] S. G. Norcross, M. Bouchard, and G. A. Soulodre, "Adaptive Strategies for Inverse Filtering," presented at the 119th convention of the Audio Engineering Society (2005), paper 6563.

[115] A. Leite and J. S. Ferreira, "An Improved Adaptive Room Equalization in the Frequency Domain," presented at the 118th convention of the Audio Engineering Society (2005), paper 6501.

[116] L. Rugini and G. Leus, "Basis Expansion Adaptive Filters for Time-Varying System Identification," 2nd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing pp. 153–156 (2007).

[117] S. Douglas, "An efficient implementation of the modified filtered-X LMS algorithm," Signal Processing Letters, IEEE **4,** 286–288 (1997).

[118] S. Miyagi and H. Sakai, "Performance comparison between the filtered-error LMS and the filtered-X LMS algorithms [ANC]," The 2001 IEEE International Symposium on Circuits and Systems **2,** 661–664 (2001).

[119] S. Miyagi and H. Sakai, "Mean-square performance of the filtered-reference/ filtered-error LMS algorithm," IEEE Transactions on Circuits and Systems I: Regular Papers **52,** 2454–2463 (2005).

[120] Y. Hinamoto and H. Sakai, "Analysis of the filtered-X LMS algorithm and a related new algorithm for active control of multitonal noise," IEEE Transactions on Audio, Speech, and Language Processing **14,** 123–130 (2006).

[121] S. Goetze, M. Kallinger, A. Mertins, and K. Kammeyer, "A Decoupled Filtered-x LMS Algorithm for Listening-Room Compensation," International Workshop on Acoustic Echo and Noise Control pp. 562–573 (September 2008).

[122] J. Kuriyama and Y. Furukawa, "Adaptive Loudspeaker System," Journal of the Audio Engineering Society **37,** 919–926 (1989).

[123] P. Nelson, F. Orduna-Bustamante, and H. Hamada, "Inverse filter design and equalization zones in multichannel sound reproduction," IEEE Transactions on Speech and Audio Processing **3,** 185–192 (1995).

[124] A. Santillan, "Spatially extended sound equalization in rectangular rooms," The Journal of the Acoustical Society of America **110,** 1989–1997 (2001).

[125] J. C. Sarris, F. Jacobsen, and G. E. Cambourakis, "Sound equalization in a large region of a rectangular enclosure," Journal of the Acoustical Society of America **116,** 3271 – 3274 (2004).

[126] Y. Park and S. D. Sommerfeldt, "Global attenuation of broadband noise fields using energy density control," Journal of the Acoustical Society of America **101,** 350–359 (1997).

[127] S. M. Kuo and D. R. Morgan, *Active Noise Control Systems: Algorithms and DSP Implementations* (John Wiley and Sons, New York, NY, 1996), p. 225.

[128] S. D. Sommerfeldt and P. J. Nashif, "An adaptive filtered-x algorithm for energy-based active control," Journal of the Acoustical Society of America **96,** 300–306 (1994).

[129] S. Goetze, M. Kallinger, A. Mertins, and K.-D. Kammeyer, "Multi-Channel Listening-Room Compensation using a Decoupled Filtered-X LMS Algorithm," Asilomar Conference on Signals, Systems, and Computers (October 2008).

[130] J. S. Bendat and J. S. Piersol, *Random Data Analysis and Measurement Procedures* (John Wiley and Sons, New York, NY, 2000), p. 201.

Appendix A

Sound Recordings

The following recordings were made in the plane-wave tube to demonstrate the real-time performance of the equalization filters presented in Ch. 4. First, recordings of the time-invariant pressure and energy density equalization are presented. These are followed by recordings of the time-varying energy density equalization. The steady-state background noise exciting the tube resonances in the equalized clips is caused by noise in the system. Additional gain was required following the DSP to compensate for a required reduction in its input signal level that prevented clipping at the output. This is particularly noticeable with pressure equalization that introduced large boosts corresponding to the nulls at the point of equalization.

Table A.1 The original band-limited ($f_c = 2000$ Hz) dry sound clip used for time-invariant equalization testing and recordings from microphone positions 7 and 10 while the sound clip was played in the tube.

Dry sound.

Sound sample recorded at microphone position 7.

Sound sample recorded at microphone position 10.

Table A.2 Equalization results using a sound pressure equalization filter derived at microphone position 7.

Sound sample recorded at position 7 with filter derived at position 7.

Sound sample recorded at position 10 with filter derived at position 7.

Table A.3 Equalization results using a sound pressure equalization filter derived at microphone position 10.

Sound sample recorded at position 10 with filter derived at position 10.

Sound sample recorded at position 7 with filter derived at position 10.

Table A.4 Equalization results using an energy density equalization filter derived at microphone position 7.5.

Sound sample recorded at position 7 with filter derived at position 7.5.

Sound sample recorded at position 10 with filter derived at position 7.5.

Table A.5 Equalization results using an energy density equalization filter derived at microphone position 10.5.

Sound sample recorded at position 7 with filter derived at position 10.5.

Sound sample recorded at position 10 with filter derived at position 10.5.

Table A.6 Time-varying equalization results using an energy density equalization filter derived at microphone position 5.5. Band-limited ($f_c = 1100$ Hz) white noised was used as the excitation signal. In the first recording, the cap of the tube is removed, and in the second it is replaced.

Sound sample recorded at position 5 in the plane-wave tube that is equalized with a time-adaptive energy density inverse filter derived at position 5.5 in real time. The rigid cap of the tube is removed at time $t = 10$ seconds.

Sound sample recorded at position 5 in the plane-wave tube that is equalized with a time-adaptive energy density inverse filter derived at position 5.5 in real time. The rigid cap of the tube is replaced at time $t = 10$ seconds.

Appendix B

C++ Code

The source code used to obtain the real-time experimental results while running filtering on a DSP is included here. Also function stubs are included at the end for the time-varying energy density equalization algorithms that were presented, but not used in this research.

```
/*****************************************************
*
*    file     : LOOP.c
*    project  : acoustic energy density equalization
*    author   : panu puikkonen
*
*****************************************************

// extra includes for testing
#include <stdlib.h>
#include <stdio.h>

#include "Analog_IO.h"
#include "ProcTimer.h" /* for measuring computation time */
#include "6x11dsk.h"

#include "EDconstants.h"
#include "inverseFilter.h"
```

```
// ~~~~~~~~~~ Timing Variables
float elapsedMicroseconds;
float elapsedMicroseconds1;
unsigned int startTime, endTime, timerOverhead;

// for the FFT function
#include <math.h>
#include "twiddletable.h"
#include "DSPF_sp_icfftr2_dif.h"
#include "DSPF_sp_cfftr2_dit.h"
#include "DSPF_sp_dotprod.h"
#include "DSPF_sp_maxval.h"
#include "DSPF_sp_fir_gen.h"

near float InSig [kMAXnumAnalogIN],
  OutSig[kMAXnumAnalogOUT];

// ~~~~~~~~~~ Code Execution Switches
int filterOn     = 0,
filteredXon  = 0,
filteredXedOn= 0,
passThrough  = 0,
runSystemID  = 0,
computeDelay = 0;

/* panu's box */
// I/O controls for channels
int filterIn = 0;
int refIn    = 1;
int mic1in  = 2;
int mic2in  = 3;

int filterOut = 0;

// Text output controls, initialized to one when reached in code
int firstPassThrough = 0;
int firstFilter      = 0;
int firstFilteredX   = 0;
int firstFilteredXed = 0;
int firstComputeDelay= 0;

// ~~~~~~~~~~ filtered-x algorithm variables
// Fc = 1100 Hz
```

```
float leak = 0.99999995; // leaky LMS algorithm
float mu   = 4e-12;      // for Wcoef
float muH  = 1e-11;      // for H1coef and H2coef
// a multiplier for the initialization of Wcoef after off-line sysID
float WcoefScale = 1;

float muE = 0;
float muEed = 0;
float muHe1 = 0;
float muHe2 = 0;

// filtered-x variables
// place holders for the various indeces used
// to keep track of C++ circular buffers
int start = 5;
int zLoc = 5;
int hLoc = 5;
int wLoc = 5;

// These three values are used with the time-domain integration
// of U (derived in Dr. Sommerfeldt's papers cited in Panu's thesis)
float lambda = 0.999666;
float alpha  = 0;
float weight = 0.01;

float temp  = 0;
float input = 0;
float d     = 0;
float e     = 0;
float y     = 0;
float u    = 0;
float r1   = 0;


float rVelocity = 0; // buffers for the LMS update
float rvMean    = 0;
float velocity  = 0;
float velMean   = 0;

float r2 = 0;
float rp = 0;
float rv = 0;
float yp = 0;
```

```
float yv = 0;
float ep = 0;
float ev = 0;

float uBuffer[kHcoef];
float H1coef[kHcoef];
float H2coef[kHcoef];
float hBuffer[kHcoef];

float r2Buffer[kWcoef];
float rvBuffer[kWcoef];
float rpBuffer[kWcoef];
float wBuffer[kWcoef];
float Wcoef[kWcoef];
float xBuffer[kWcoef];
float r1Buffer[kWcoef];

float zBuffer[kDelay];

float error[keBufferLength]; // these are to output error to a buffer
int eCount = 0;

float Pmag[kIDlength];

// ~~~~~~~~~~ updating energy density directly -
// Dr. Sommerfeldt's alternative method
float muED = 1e-30; // this for ED adaptive filtering
float yED = 0;
float WcoefPrev[kIDlength];

// ~~~~~~~~~~ online system identification variables
float eH1   = 0;
float eH2   = 0;
float muEh1 = 0;
float muEh2 = 0;

// ~~~~~~~~~~ Delay estimation variables for delay()
float e1 = 0;
float e2 = 0;

float d1 = 0;
float d2 = 0;
float y1 = 0;
```

```
float y2 = 0;

float dBuffer[kHcoef];

// ~~~~~~~~~~ FFT variables
int N = kIDlength;
float binWidth = 0;

float w[kIDlength];              // twiddle factors for fft

// ~~~~~~~~~~ sysID variables
int sample    = 0;
int bufferLoc = 0;
int highPass  = 75;              // high pass filter zeroes bins of U

int avgCount  = 1;               // current iteration of averages
float numAvgs = 20;              // number of averages

float coherence[kIDlength];      // coherence
float ED[kIDlength];             // ED magnitude
float FIRcoef[kIDlength];
float hMult[kIDlength];          // for the Hilbert transform
float impRes[kIDlength];         // impulse response coefficients
float mic1[kIDlength];           // records mic signal
float mic2[kIDlength];           // records mic signal
float mic1FFTmag[kIDlength];     // magnitude spectrum
float mic2FFTmag[kIDlength];     // magnitude spectrum
float omega[kIDlength];          // for particle velocity derivation
float refSig[kIDlength];         // filtered reference signal
double autocor[kIDlength];       // temp buffer
float Sxx[kIDlength];            // input  autospectrum
float Syy[kIDlength];            // output autospectrum
float FR[kIDlength];             // transfer function
float window[kIDlength];         // time domain window

float avgPress[kIDlengthX2];     // holds (P2+P1)/2
float hComplex[kIDlengthX2];     // complex transfer function
float FRminPhase[kIDlengthX2];   // minimum phase transfer function
float mic1FFT[kIDlengthX2];      // complex fft
float mic2FFT[kIDlengthX2];      // complex fft
float refSigFFT[kIDlengthX2];    // complex fft
float Sxy[kIDlengthX2];          // input-output cross spectrum
float U[kIDlengthX2];            // complex particle velocity
```

```
float hilbertScale = 0;          // used for the hilbert transform
float scale = 0;                 // scales FFTs
float maxVal = 0;

/*************************************************************
loopInit()
This function runs once when the program is first started,
and before the system enters its normal sampling mode. All
necessary variables are initialized in this function.
*************************************************************

void loopInit(void)
{
int i;
float index = 0;

// Initialize the proc timer
proc_timer_reset();
proc_timer_start();
startTime = proc_timer_read();
endTime = proc_timer_read();
timerOverhead = endTime-startTime;
// TIMER_COUNTS_PER_MICROSECOND found in ProcTimer.h

// for systemID
binWidth = (float)kDefaultSampleRate/kIDlength;

// for all FFTs of length N
gen_w_r2(w, N);    // generates twiddle factors for use with the FFT

    // The twiddle factors need to be bit reversed before FFT
    bit_rev(w, N>>1);

for (i=0; i<N; i++)  // initializes all buffers
{
autocor[i]    = 0;
coherence[i]  = 0;
ED[i]         = 0;
FIRcoef[i]    = 0;
FR[i]         = 0;
hMult[i]      = 0;
impRes[i]     = 0;
```

```
mic1[i]       = 0;
mic1FFTmag[i] = 0;
mic2[i]       = 0;
mic2FFTmag[i] = 0;
omega[i]      = 0;
refSig[i]     = 0;
Sxx[i]        = 0;
Syy[i]        = 0;

WcoefPrev[i] = 0; // this is for edUpdate()

window[i] = 0.5*(1-cos(2*PI*(index/N))); // hanning window
index++;
}

for (i=0; i<N*2; i++)// buffers for complex values
{
avgPress[i]    = 0;
hComplex[i]    = 0;
FRminPhase[i]  = 0;
mic1FFT[i]     = 0;
mic2FFT[i]     = 0;
refSigFFT[i]   = 0;
Sxy[i]         = 0;
U[i]           = 0;
}

for (i=0; i<kWcoef; i++) // These buffers are filtered by Wcoef
{
wBuffer[i]  = 0;
xBuffer[i]  = 0;
r1Buffer[i] = 0;
r2Buffer[i] = 0;
rpBuffer[i] = 0;
rvBuffer[i] = 0;

Wcoef[i]  = 0;
H1coef[i] = 0;
H2coef[i] = 0;
}

    // These buffers are filtered by H1coef and H2coef
for (i=0; i<kHcoef; i++)
```

```
{
dBuffer[i] = 0;
uBuffer[i] = 0;
hBuffer[i] = 0;
H1coef[i]  = 0;
H2coef[i]  = 0;
}

    // This buffer holds previous values of x(n)
for (i=0; i<kDelay; i++)
{
zBuffer[i] = 0;
}

    // This can be used to record the error
for (i=0; i<keBufferLength; i++)
{
error[i] = 0;
}

// used to compute particle velocity
omega[0] = 1;

for (i=1; i<=N/2; i++)
{
omega[i] = 2*PI*binWidth*i;
}

for (i=1; i<N/2; i++)
{
omega[kIDlength-i] = omega[i];
}

// multiplier buffer for the Hilbert transform
for (i=0; i<N/2; i++)
{
if (i==0)
hMult[i] = 1;
else
hMult[i] = 2;
}

    // Necessary scaling factor for the Hilbert transform
```

```
hilbertScale = 1/(float) N;

    // number of averages computed in the off-line sysID
    scale = 1/numAvgs;

    // used in time-domain integration of U
alpha = -1/(Rho*X*kDefaultSampleRate);
}


/*******************************************************
loop()
This function is called once at the beginning of a
sampling period and determines what computations are
performed during that time interval, 1/Fs. A simple
binary system is used to turn functions on and off and
 a text prompt is displayed the first time a function
 is called. The proc_timer computes how long it takes
 to run all the code that iteration and the result is
 displayed in the variable elapsedMicroseconds. This
 function can be used anywhere in the code to benchmark
 the performance of a block of code.
*******************************************************

near void loop(void)
{
startTime = proc_timer_read();

    if(passThrough)
    {
     dataPassThrough();

if (firstPassThrough==0)
{
printf("Data passing through!\n");
firstPassThrough = 1;
}
    }
else if(runSystemID)
{
systemID();
}
else if(filterOn)
```

```
{
FIR();

if (firstFilter==0)
{
printf("Filter Time!\n");
firstFilter = 1;
}
}

else if (filteredXon)
{
filteredX();

if (firstFilteredX==0)
{
printf("Filtered-x EQ!\n");
firstFilteredX = 1;
}
}

else if (filteredXedOn)
{
filteredXed();

if (firstFilteredXed==0)
{
printf("Filtered-x ED EQ!\n");
firstFilteredXed = 1;
}
}

else if (computeDelay)
{
delay();

if (firstComputeDelay==0)
{
printf("Computing delay!\n");
firstComputeDelay = 1;
}
}
```

```
endTime = proc_timer_read();
elapsedMicroseconds =
    (endTime - startTime)/TIMER_COUNTS_PER_MICROSECOND;
}


void systemID()
{
systemIDinput();
}


/*******************************************************
systemIDinput()
The excitation signal is played separately into the tube,
and samples are recorded. After kIDlength samples have
been recorded computations() is called. This repeats
numAvgs times, and then the minimum phase inverse filter is
computed, and the filter coefficients are stored in Wcoef.
*******************************************************/

void systemIDinput()
{
int i,j,k;
k = 0;

ADC_read(InSig);

if(sample < kIDlength)
{
        // the input samples are windowed
mic1[sample]   = window[sample]*InSig[mic1in];
mic2[sample]   = window[sample]*InSig[mic2in];
refSig[sample] = window[sample]*InSig[refIn];

sample++;
}

else if(sample >= kIDlength && avgCount < numAvgs)
    {
        sample = 0; // restarts the count for the next block

        avgCount++;
```

```
        computations();
    }

    else
    {
    avgCount    = 1;
    sample      = 0;
        bufferLoc   = 0;
    runSystemID = 0;

    printf("Recording ended!\n");

     computations();
     transferFunction();
     hilbertTransform();

// scales impRes
        // finds maxVal of the computed minimum-phase impulse response
maxVal = DSPF_sp_maxval(impRes, N);

for (i=0; i<N; i++)
{
impRes[i] = impRes[i]/maxVal;      // normalizes impRes
}

for (i=0; i<N; i++)
{
for (j=k; j<N; j++)
{
                // computes the one sided autocorrelation function
autocor[i] += (impRes[j]*impRes[j-k]);
}
k++;
}

        // computes the minimum-phase inverse filter coefficients
levinsonDurbin(autocor, FIRcoef);

// scales Wcoef
maxVal = DSPF_sp_maxval(FIRcoef, N);  // finds the maxVal of FIRcoef

for (i=0; i<N; i++)
{
```

```
FIRcoef[i] = FIRcoef[i]/maxVal;   // normalizes FIRcoef
}

for (i=0; i<kWcoef; i++)
{
            // assigns values to the time-adaptive coefficients
Wcoef[i] = FIRcoef[i]*WcoefScale;
}

     printf("Inverse filter computed!\n");
    }
}


/*************************************************************
computations()
Copies input buffers into arrays twice the length to prepare
for the complex FFT computation.  Computes FFT of the two
microphone recordings and proceeds to call the appropriate
functions to derive the time averaged energy density from
these two recordings.
*************************************************************

void computations() // called in systemIDinput()
{
int i;

for (i=0; i<N; i++)  // test "input signal"
{
mic1FFT[2*i+1] = 0;
mic2FFT[2*i+1] = 0;

refSigFFT[2*i+1] = 0;
}

for (i=0; i<N; i++)  // test "input signal"
{
mic1FFT[2*i] = mic1[i];
mic2FFT[2*i] = mic2[i];

refSigFFT[2*i] = refSig[i];
}
```

```
// FFTs
DSPF_sp_cfftr2_dit(mic1FFT, w, N);
bit_rev(mic1FFT, N);
DSPF_sp_cfftr2_dit(mic2FFT, w, N);
bit_rev(mic2FFT, N);

DSPF_sp_cfftr2_dit(refSigFFT, w, N);
bit_rev(refSigFFT, N);

frequencyAverages();
particleVelocity();
energyDensity();

/*
//This part computes the IFFTs if ever necessary
bit_rev(mic1FFT, N);
DSPF_sp_icfftr2_dif(mic1FFT, w, N);
bit_rev(mic2FFT, N);
DSPF_sp_icfftr2_dif(mic2FFT, w, N);

// We are now in the time domain
divide(mic1FFT, N);
divide(mic2FFT, N);
*/
}

/*********************************************************
frequencyAverages()
Averages magnitude spectrums for mic1 and mic2, and refSig.
Computes the average pressure between the two microphone
locations. Computes the auto spectrum of the input, and
the cross spectrum between the input and the averaged
pressure. All buffers are averaged.
*********************************************************

void frequencyAverages()  // called in computations()
{
int i;

// Visual check of the frequency content
for (i=0; i<N; i++)
{
```

```
avgPress[2*i]   = 0.5 * (mic1FFT[2*i] + mic2FFT[2*i]);
avgPress[2*i+1] = 0.5 * (mic1FFT[2*i+1]+mic2FFT[2*i+1]);

//magnitude = sqrt( A*conj(A))
mic1FFTmag[i] += scale*(mic1FFT[2*i]*mic1FFT[2*i]+
                        mic1FFT[2*i+1]*mic1FFT[2*i+1]);
mic2FFTmag[i] += scale*(mic2FFT[2*i]*mic2FFT[2*i]+
                        mic2FFT[2*i+1]*mic2FFT[2*i+1]);

Sxx[i] += scale*(refSigFFT[2*i]*refSigFFT[2*i] +
                 refSigFFT[2*i+1]*refSigFFT[2*i+1]);
Syy[i] += scale*(mic1FFT[2*i]*mic1FFT[2*i]     +
                 mic1FFT[2*i+1]*mic1FFT[2*i+1]);

Sxy[2*i]   += scale*(mic1FFT[2*i]*refSigFFT[2*i]   +
                     mic1FFT[2*i+1]*refSigFFT[2*i+1]);
Sxy[2*i+1] += scale*(mic1FFT[2*i]*refSigFFT[2*i+1] -
                     mic1FFT[2*i+1]*refSigFFT[2*i]);
}
}


/*****************************************************
particleVelocity()
computes particle velocity, U, according to the equation
U = (P2-P1)/(Rho*omega*X) explained in Panu's thesis
*****************************************************

void particleVelocity() // called in computations()
{
float denominator = 0;
int i;

for (i=0; i<N; i++)
{
denominator = 1/(Rho*omega[i]*X);

U[2*i]   = (mic1FFT[2*i+1]-mic2FFT[2*i+1])*denominator;
U[2*i+1] = (mic2FFT[2*i]-mic1FFT[2*i])*denominator;
}

// Zeros the first highPass/binWidth values on both ends,
    // otherwise the values would be unnaturally large
```

```
U[0] = 0;
U[1] = 0;

    // compensates for the large error near 0 Hz
for (i=1; i<highPass/binWidth; i++)
{
U[2*i]   = 0;
U[2*i+1] = 0;
U[kIDlengthX2+1-2*i] = 0;
U[kIDlengthX2  -2*i] = 0;
}
}


/************************************************************
energyDensity()
Determines average pressure (P1+P2)/2, and derives energy
density using the equation
ED = 1/4*rho*abs(U)^2 + 1/4*abs(P)^2/(rho*C^2)) explained in
Panu's thesis.
************************************************************

void energyDensity() // called in computations()
{
int i;

float velocity = 0;
float pressure = 0;

// derive ED
for (i=0; i<N; i++)
{
velocity = U[2*i]*U[2*i] + U[2*i+1]*U[2*i+1];
pressure = avgPress[2*i]*avgPress[2*i] +
                  avgPress[2*i+1]*avgPress[2*i+1];
//ED[i] += scale * (0.25/(Rho*C*C)*pressure +
                  0.25*Rho*velocity);
ED[i]   += scale * (pressure + Rho*Rho*C*C*velocity);
}

printf("Computations done!\n");
}
```

```
/************************************************************
transferFunction()
Here a pressure or an energy density transfer function can be
 computed depending on which line of code you run. The ED TF
 is computed as |ED|/|Sxx|
and the pressure TF is computed as explained in Chapter 3 of
Panu's thesis. The coherence function is also computed here
between the output and the input to one of the microphones.
************************************************************

void transferFunction()  // called in systemIDinput()
{
int i;

// computes the tranfer function and coherence
for (i=0; i<N; i++)
{
// computes transfer function

FR[i] = sqrt(ED[i]/Sxx[i]);
/*
hComplex[2*i]   = Sxy[2*i]  /Sxx[i];
hComplex[2*i+1] = Sxy[2*i+1]/Sxx[i];

FR[i] = sqrt(hComplex[2*i]*hComplex[2*i] +
                hComplex[2*i+1]*hComplex[2*i+1]);
//*/
coherence[i] = (Sxy[2*i]*Sxy[2*i]+Sxy[2*i+1]*Sxy[2*i+1])/
                       (Sxx[i]*Syy[i]);

}
}


/************************************************************
hilbertTransform()
This is a line-by-line application of code developed in matlab.
matlab code to implement:
1. x = ifft(ed_log);
2. x_times_h = x.*h;
3. x_fft = fft(x_times_h);
```

```
4. min_phase = exp(x_fft);
5. impulse_min_phase = real(ifft(min_phase));
*************************************************************

void hilbertTransform() // called in systemIDinput()
{
int i = 0;
float loc1 = 0;
float loc2 = 0;

// complex buffer for the transform
for (i=0; i<N; i++)
{
FRminPhase[2*i] = logf(FR[i]);
FRminPhase[2*i+1] = 0;
}

// 1.
bit_rev(FRminPhase, N);
DSPF_sp_icfftr2_dif(FRminPhase, w, N);

// 2.
for (i=0; i<N; i++)
{
FRminPhase[2*i]=hilbertScale*FRminPhase[2*i]*hMult[i];
FRminPhase[2*i+1]=hilbertScale*FRminPhase[2*i+1]*hMult[i];
}

// 3.
DSPF_sp_cfftr2_dit(FRminPhase, w, N);
bit_rev(FRminPhase, N);

// 4.
for (i=0; i<N; i++)
{
loc1 = FRminPhase[2*i];
loc2 = FRminPhase[2*i+1];
FRminPhase[2*i]   = expf(loc1)*cos(loc2);
FRminPhase[2*i+1] = expf(loc1)*sin(loc2);
}

// 5.
bit_rev(FRminPhase, N);
```

```
DSPF_sp_icfftr2_dif(FRminPhase, w, N);

for (i=0; i<N; i++)
{
impRes[i] = FRminPhase[2*i]/N;
}

}


/*********************************************************
delay()
estimates the delay of H(z). A block diagram for this
method is given in Chapter 3 of Panu's thesis.
*********************************************************

//delay function

void delay()
{
// available variables are e,y,d,u,r

int i;
y1 = 0;
y2 = 0;

ADC_read(InSig);
input = InSig[filterIn];
d1    = InSig[mic1in];
d2    = InSig[mic2in];

hLoc--;
    if (hLoc < 0)
     hLoc = kHcoef-1;

    dBuffer[hLoc] = input;

////////////////////////////////////////////////

    for (i=0; i<kHcoef; i++)
{

    y1 += H1coef[i] * dBuffer[hLoc];
```

```
        y2 += H2coef[i] * dBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

////////////////////////////////////////////////
// the LMS update

// at mic1 and mic2
e1 = d1 - y1;
e2 = d2 - y2;

muHe1 = muH*e1;
muHe2 = muH*e2;

    for (i=0; i<kHcoef; i++)
{
        H1coef[i] += muHe1 * dBuffer[hLoc];
        H2coef[i] += muHe2 * dBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

////////////////////////////////////////////////
// prepare data for output

for(i=0; i<kMAXnumAnalogOUT; i++)
    {
      OutSig[i] = 0;
    }

    // outputs the current inverse filtered sample
OutSig[filterOut] = input;

DAC_write(OutSig);
}


/*****************************************************
```

```
filteredX()
This method is explained in detail in Chapter 2 of Panu's
thesis. The index buffers are decremented every iteration,
and the newest sample is always at the location of the
index, and the oldest is at index-1. Checks are included to
ensure that wrapping around the ends of the buffer work.
*******************************************************

// THIS IS FILTERED-X FOR PRESSURE
void filteredX()
{
int i;
u = 0;
r1 = 0;

ADC_read(InSig);
input = InSig[filterIn];
y = InSig[mic1in];

////////////////////////////////////////////////
// updates current locations for the circular
// buffer shared by x, z, and w. They're all the
// same length.

    wLoc--; // buffer for W
    if (wLoc < 0)
     wLoc = kWcoef-1;

    wBuffer[wLoc] = input;

    hLoc--; // buffer for H1coef and H2coef
    if (hLoc < 0)
     hLoc = kHcoef-1;

    hBuffer[hLoc] = input;


    zLoc--; // digital delay for Z
    if (zLoc < 0)
     zLoc = kDelay-1;

    zBuffer[zLoc] = input;
```

```
if (zLoc > 0) // d(n)
     d = zBuffer[zLoc-1];
else
d = zBuffer[kDelay-1];

////////////////////////////////////////////////
// determine current u(n), r(n), and d(n)

// u(n) = w(n)^T x(n)
    for (i=0; i<kWcoef; i++)
{
        u += Wcoef[i] * wBuffer[wLoc];

        wLoc++;
        if(wLoc == kWcoef) // C++ circular buffer
            wLoc = 0;
}

for (i=0; i<kHcoef; i++)
{
        r1 += H1coef[i] * hBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef) // C++ circular buffer
            hLoc = 0;
}

r1Buffer[wLoc] = r1;

uBuffer[hLoc] = u;

// THIS IS FILTERED-X FOR PRESSURE
////////////////////////////////////////////////
// update W

    e = d-y;

muE = mu*e;

    for (i=0; i<kWcoef; i++)
{
        Wcoef[i] = Wcoef[i] + muE * r1Buffer[wLoc];
```

```
        wLoc++;
        if(wLoc == kWcoef)
            wLoc = 0;
}

////////////////////////////////////////////
// update H

y = 0;

for (i=0; i<kHcoef; i++)
{
        y += H1coef[i] * uBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

//eH1 = d - y;
eH1 = (d - y) - e;

muEh1 = muH*eH1;

    for (i=0; i<kHcoef; i++)
{
        H1coef[i] = H1coef[i] + muEh1 * uBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

////////////////////////////////////////////
// prepare data for output
for(i=0; i<kMAXnumAnalogOUT; i++)
    {
      OutSig[i] = 0;
    }
    // outputs the current inverse filtered sample
OutSig[filterOut] = u;
```

```
DAC_write(OutSig);
}
// THIS IS FILTERED-X FOR PRESSURE


/*********************************************************
filteredXed()
A block diagram and a detailed explanation of this methods
implementing two d(n)s is given in Chapter 2 of Panu's thesis.
*********************************************************

// THIS IS FILTERED-X FOR ENERGY DENSITY
near void filteredXed()
{
int i;
u = 0;
r1 = 0;
r2 = 0;

ADC_read(InSig);
input = InSig[filterIn];
y1 = InSig[mic1in];
y2 = InSig[mic2in];

    wLoc--; // buffer for W
    if (wLoc < 0)
     wLoc = kWcoef-1;

    wBuffer[wLoc] = input;

    hLoc--; // buffer for H1coef and H2coef
    if (hLoc < 0)
     hLoc = kHcoef-1;

    hBuffer[hLoc] = input;

    zLoc--; // digital delay for Z
    if (zLoc < 0)
     zLoc = kDelay-1;

    zBuffer[zLoc] = input;

    // if the delay between the two samples is more than
```

```
    // one sample this indexing system needs to adjusted

if (zLoc > 0) // d(n)
     d1 = zBuffer[zLoc-1];
else
d1 = zBuffer[kDelay-1];

if (zLoc > 1) // d(n)
     d = zBuffer[zLoc-2];
    else if (zLoc == 1)
     d = zBuffer[kDelay-1];
else
d = zBuffer[kDelay-2];

////////////////////////////////////////////////
// determine current u(n), r(n), and d(n)

// u(n) = w(n)^T x(n)
    for (i=0; i<kWcoef; i++)
{
        u += Wcoef[i] * wBuffer[wLoc];

        wLoc++;
        if(wLoc == kWcoef)
            wLoc = 0;
}

    for (i=0; i<kHcoef; i++)
{
        r1 += H1coef[i] * hBuffer[hLoc];
        r2 += H2coef[i] * hBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

r1Buffer[wLoc] = r1;
r2Buffer[wLoc] = r2;

uBuffer[hLoc] = u;

Wupdate();
```

```
HhatUpdate();
//edUpdate();

////////////////////////////////////////////////
// prepare data for output
for(i=0; i<kMAXnumAnalogOUT; i++)
    {
      OutSig[i] = 0;
    }

    // outputs the current inverse filtered sample
OutSig[filterOut] = u;

DAC_write(OutSig);
}
// THIS IS FILTERED-X WITH AN ONLINE SYSTEM UPDATE


/*********************************************************
Wupdate()
Updates the filter coefficients for the filtered-x ED
algorithm; the theory for this update method is given in
Chapter 2 of Panu's thesis.
*********************************************************

void Wupdate()
{

int i = 0;

e1 = d  - y1;
e2 = d1 - y2;

velocity = lambda*velocity + alpha*(e2-e1); // u
velMean  = weight*velocity + (1-weight)*velMean;
ev       = velocity - velMean;

// pressure and particle velocity error signals
ep = (e1+e2)/2;  // dp-yp

// filtered-x pressure and particle velocity

rp = (r1+r2)/2; // pressure
```

```
rVelocity = lambda*rVelocity + alpha*(r2-r1); // u
rvMean    = weight*rVelocity + (1-weight)*rvMean;
rv        = rVelocity - rvMean;

rvBuffer[wLoc] = rv;
rpBuffer[wLoc] = rp;

e = ep + Rho*C*ev;
muEed = mu * e;

    for (i=0; i<kWcoef; i++)
{
        Wcoef[i] = leak*Wcoef[i] + muEed * rpBuffer[wLoc];

        wLoc++;
        if(wLoc == kWcoef)
            wLoc = 0;
}

// error for output
if (eCount < keBufferLength)
{
error[eCount] = e;
eCount++;
}
/*
// tests HhatUpdate()
e = d-y1;

muE = mu*e;

    for (i=0; i<N; i++)
{
        Wcoef[i] += muE * r1Buffer[start];

        start++;
        if(start == N)
            start = 0;
}
*/
}
```

```
/*****************************************************
HhatUpdate()
Online system identification to adaptively estimate H1
and H2, the transfer functions to the two field
microphones. The theory for this function is
explained in Chapter 2 of Panu's thesis.
*****************************************************

void HhatUpdate()
{
int i = 0;

d1 = y1;
d2 = y2;

y1 = 0;
y2 = 0;

for (i=0; i<kHcoef; i++)
{
        y1 += H1coef[i] * uBuffer[hLoc];
        y2 += H2coef[i] * uBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
}

eH1 = d1-y1 ;//- e;
eH2 = d2-y2 ;//- e;

muEh1 = muH*eH1;
muEh2 = muH*eH2;

    for (i=0; i<kHcoef; i++)
{
        H1coef[i] += muEh1 * uBuffer[hLoc];
        H2coef[i] += muEh2 * uBuffer[hLoc];

        hLoc++;
        if(hLoc == kHcoef)
            hLoc = 0;
```

```
}
}


/*************************************************
edUpdate()
Online system identification to adaptively estimate
H1 and H2, the transfer functions to the two field
microphones.
*************************************************
/*
void edUpdate()
{
int   i = 0;

// pressure and particle velocity from microphones
velocity = lambda*velocity + alpha*(y2-y1); // u
velMean  = weight*velocity + (1-weight)*velMean;
yv       = velocity - velMean;

yp = (y1+y2)/2;

yED = (1/(2*Rho*C*C)*yp*yp + Rho/2*yv*yv); // y^2(t)

e = d*d - yED;  // d^2(t) - y^2(t)
ep = e;         // ep is the previous e^2(t)

    for (i=0; i<N; i++)
{   temp = (e - ep)/(Wcoef[i] - WcoefPrev[i]);
        Wcoef[i] = leak*Wcoef[i] + temp;
        // current becomes previous for next iteration
WcoefPrev[i] = Wcoef[i];
}
}
*/
/*******************************************************
FIR()
reads in a sample and filters it through a Wcoef length
FIR filter. If you want to perform non-time adaptive
filtering this is the function to call, and the
coefficients that are the output from levinsonDurbin()
are used.
*******************************************************
```

```
void FIR()
{
int i;
double y = 0;

ADC_read(InSig);

// FIR filter
start--;
    if (start < 0)
     start = N-1;

    xBuffer[start] = InSig[filterIn];

// This implements a circular buffer
    for (i=0; i<N; i++)
{
        y += FIRcoef[i] * xBuffer[start];

        start++;
        if(start > N-1)
            start = 0;
}


// prepare data for output
for(i=0; i<kMAXnumAnalogOUT; i++)
    {
      OutSig[i] = 0;
    }


OutSig[filterOut] = y; // ED

DAC_write(OutSig);
}


/*********************************************
dataPassThrough()
reads in twelve values and outputs them.
*********************************************
```

```
void dataPassThrough(void)
{
int i;

ADC_read(InSig);

for(i=0; i<12; i++)
OutSig[i] = InSig[i];

for(i=0; i<kMAXnumAnalogOUT; i++)
    {
     OutSig[i] = 0;
    }

OutSig[0] = InSig[0];


DAC_write(OutSig);
}



//~~~~~~~~~~~~~~~~~~~~~~
// Alternative Functions
//~~~~~~~~~~~~~~~~~~~~~~

////////////////////////////////////////////
// The 1/(rhoc) approximation method
////////////////////////////////////////////

/*****************************************
Wupdate()
Replace the other Wupdate() with this function. Also
in filteredXed(), replace the existing d(n) section
with the following that only uses a single d(n)

if (zLoc > 0) // d(n)
     d = zBuffer[zLoc-1];
else
d = zBuffer[kDelay-1];

Using this method a 1/(rhoc) approximation is made
```

```
for particle velocity U.

Updates the filter coefficients for the filtered-x
ED algorithm according to the theory explained in
Chapter 2 of Panu's thesis under the "acoustic
impedance filter" method using the 1/(rhoc) plane
wave in a free field approximation.
*****************************************
/*
void Wupdate()
{

int i = 0;

// pressure and particle velocity from microphones
velocity = lambda*velocity + alpha*(y2-y1); // u
velMean  = weight*velocity + (1-weight)*velMean;
yv       = velocity - velMean;

// pressure and particle velocity error signals
ep = d - (y1+y2)/2;  // dp-yp
ev = d/(Rho*C) - yv; // dv-yv

// filtered-x pressure and particle velocity

rp = (r1+r2)/2; // pressure

rVelocity = lambda*rVelocity + alpha*(r2-r1); // u
rvMean    = weight*rVelocity + (1-weight)*rvMean;
rv        = rVelocity - rvMean;

rvBuffer[wLoc] = rv;
rpBuffer[wLoc] = rp;

e = ep + Rho*C*ev;
muEed = mu * e;

    for (i=0; i<kWcoef; i++)
{
        Wcoef[i] = leak*Wcoef[i] + muEed * rpBuffer[wLoc];

        wLoc++;
        if(wLoc == kWcoef)
```

```
            wLoc = 0;
}

// error for output
if (eCount < keBufferLength)
{
error[eCount] = e;
eCount++;
}

}
*/

////////////////////////////////////////////
// The acoustic impedance filter, Zs, method
////////////////////////////////////////////

/*******************************************************
acousticImpedanceFilter()
Call this function at the end of systemIDinput().
This filter estimates the specific acoustic impedance at
the sensor location - used to derive d_v(n) from d_p(n)
*******************************************************
/*
void ZsFilter()
{
int i = 0;
float loc1 = 0;
float loc2 = 0;

for (i=0; i<N; i++)
{
FR[i] = sqrt(Umag[i]/Pmag[i]);

if (FR[i] == 0)
FR[i] = 1e-10;
}

// complex buffer for the transform
for (i=0; i<N; i++)
{
FRminPhase[2*i] = logf(FR[i]);
FRminPhase[2*i+1] = 0;
```

```
}

// 1.
bit_rev(FRminPhase, N);
DSPF_sp_icfftr2_dif(FRminPhase, w, N);

// 2.
for (i=0; i<N; i++)
{
FRminPhase[2*i]   = hilbertScale*FRminPhase[2*i] *
                            hMult[i];
FRminPhase[2*i+1] = hilbertScale*FRminPhase[2*i+1]*
                            hMult[i];
}

// 3.
DSPF_sp_cfftr2_dit(FRminPhase, w, N);
bit_rev(FRminPhase, N);

// 4.
for (i=0; i<N; i++)
{
loc1 = FRminPhase[2*i];
loc2 = FRminPhase[2*i+1];
FRminPhase[2*i]   = expf(loc1)*cos(loc2);
FRminPhase[2*i+1] = expf(loc1)*sin(loc2);
}

// 5.
bit_rev(FRminPhase, N);
DSPF_sp_icfftr2_dif(FRminPhase, w, N);

for (i=0; i<N; i++)
{
Zcoef[i] = FRminPhase[2*i]/N;
}
}*/


/************************************************
Wupdate()
Replace the other Wupdate() with this function. Also
in filteredXed(), replace the existing d(n) section
```

```
with the following that only uses a single d(n)

if (zLoc > 0) // d(n)
     d = zBuffer[zLoc-1];
else
d = zBuffer[kDelay-1];

Using this method particle velocity U is approximated
by a Zs filter.

Updates the filter coefficients for the filtered-x ED
algorithm according to the theory explained in Chapter 2
of Panu's thesis under the "acoustic impedance filter"
method.
*****************************************************
/*
void Wupdate()
{

int i = 0;
zsOut = 0;

start--;
    if (start < 0)
     start = N-1;

zsBuffer[start] = d;

for (i=0; i<N; i++)
{
        zsOut += Zcoef[i] * zsBuffer[start];

        start++;
        if(start == N)
            start = 0;
}

ev = zsOut - yv;
ep = d - (y1+y2)/2;  // dp-yp

//////////////////trial for Zs////////////

// filtered-x pressure and particle velocity
```

```
rp = (r1+r2)/2; // pressure

rVelocity = lambda*rVelocity + alpha*(r2-r1); // u
rvMean    = weight*rVelocity + (1-weight)*rvMean;
rv        = rVelocity - rvMean;

rvBuffer[wLoc] = rv;
rpBuffer[wLoc] = rp;

e = ep + Rho*C*ev;
muEed = mu * e;

    for (i=0; i<kWcoef; i++)
{
        Wcoef[i] = leak*Wcoef[i] + muEed * rpBuffer[wLoc];

        wLoc++;
        if(wLoc == kWcoef)
            wLoc = 0;
}

// error for output
if (eCount < keBufferLength)
{
error[eCount] = e;
eCount++;
}
}
*/
```

Appendix C

MATLAB Code

This file includes some of the short, handy bits of code for different functions I've needed to implement in MATLAB during my research. Hopefully you will find something useful here. The code is divided into cells with each new topic beginning with %%.

```
%% white noise

% Creates a white noise signal

fs = 44100; % Hz
t = 900; % sec.

noise = randn(1,t*fs);
noise = noise ./(max(abs(noise)))*0.95;
% noise = filter(lowPass,1,noise);

% subplot(2,1,1)
% plot(noise)
% subplot(2,1,2)
% plot(20*log10(abs(fft(noise))));

wavwrite(noise,fs,'noise44100Hz')
```

```
%% low pass FIR filter

[audio, Fs] = wavread('noise-3dB@44kHz');

audio1 = filter(lowPass,1,audio);

wavwrite(audio1,Fs,'noise-3dBLP1900Hz@44kHz')

% plots
N = 4096;
Fs = 44100;
freqScale = 0:Fs/N:Fs-(Fs/N);
xLim = Fs/2; % Hz

figure
plot(freqScale,20*log10(abs(fft(audio,N))),'k');
hold on;
plot(freqScale,20*log10(abs(fft(audio1,N))));
xlim([0 Fs/2])


% surface plot
% This code is a stub; it will not run. For my applications
% F corresponds to my frequency axis, X corresponds to tube
% positions, and A holds the values to fill the plot with

[F,X] = ndgrid(f,x);
A = [impRes10 impRes10ED impRes7 impRes7ED];
surf(X,F,A)


%% looks at how signal level changes after filtering
% with the non-normalized levinson filter the output is
% magnified about five fold

load FIRcoef.dat;
load FIRcoefED.dat;

FIRcoef = FIRcoefED;

fs = 4000; % Hz
t = 5; % sec.
```

```
noise = randn(1,t*fs);
noiseNorm = noise./(max(abs(noise)));

noiseFilt = filter(FIRcoef,1,noiseNorm)*.15; % .15 pressure, .2 ED

% average peak value
avgBefore = sum(abs(noiseNorm))/length(noiseNorm)
avgAfter  = sum(abs(noiseFilt))/length(noiseNorm)
ratio = avgAfter/avgBefore

% plots
N = length(noiseFilt);
Fs = 4000;
freqScale = 0:Fs/N:Fs-(Fs/N);
xLim = Fs/2; % Hz

tLength = N/Fs;
increment = tLength/N;
t = 0:increment:tLength-increment;
tLim = max(t);

figure
subplot(2,1,1)
plot(t,noiseFilt,':')
hold on;
plot(t,noiseNorm,'k')

subplot(2,1,2)
plot(freqScale,20*log10(abs(fft(noiseNorm))),'k');
hold on;
plot(freqScale,20*log10(abs(fft(noiseFilt))));
title('after')
legend('before','after',4)
xlim([0 fs/2])

%% Hilbert transform

hzLog = log(abs(hz));
x = ifft(hzLog)';

% creates the h vector
N = length(hz);
h = zeros(N,1);
```

```
h(1) = 1;
h(2:round(N/2)) = 2;
h(round(N/2+1)) = 1;

hzMinPhase = exp(fft(x.*h));    % frequency domain
impMP = real(ifft(hzMinPhase)); % time domain


%% reading and writing a wave file

[audio, Fs] = wavread('file');
wavwrite(z,Fs,'file')


%% computing particle velocity from two pressure measurements

load mic1FFT.dat;
load mic2FFT.dat;
load omega.dat;

rho = 1.21;
X = 0.05;

for i = 1:length(mic1FFT)/2
    p1(i) = mic1FFT(2*i-1)+mic1FFT(2*1);
    p2(i) = mic2FFT(2*i-1)+mic2FFT(2*1);
end

U = j*(p2-p1)./(omega*rho*X)';

% plots
N = length(p1);
Fs = 4000;
freqScale = 0:Fs/N:Fs-(Fs/N);

figure
subplot(2,2,1)
plot(freqScale,20*log10(abs(j*(p2-p1))),'k');
xlim([0 Fs/2])

subplot(2,2,2)
plot(freqScale,abs(j*(p2-p1)),'k');
xlim([0 Fs/2])
```

```
subplot(2,2,3)
plot(freqScale,omega*rho*X)
xlim([0 Fs/2])

subplot(2,2,4)
plot(freqScale,20*log10(abs(j*(p2-p1)./(omega*rho*X)')),'k')
hold on;
plot(freqScale,20*log10(abs(U)),':')
xlim([0 Fs/2])


%% look at the plot of particle velocity, U

close all; clear all;

load U.dat

N = length(U)/2;
Fs = 8000;

freqScale = 0:Fs/N:Fs-(Fs/N);
xLim = Fs/2;

u = zeros(length(U)/2,1);

for k = 1:N
    u(k) = U(2*k-1) + j*U(2*k);
end

% plots
figure
plot(freqScale,20*log10(abs(u)),'k');
title('particle velocity');
xlim([0 xLim])


%% FIRcoef.h output

% Creates an output file of filter coefficients that
% can be loaded on to the DSP

% function Coutput(FIRcoef,name)
```

```
fid=fopen(name,'w');

fprintf(fid,'float FIRcoef[kFiltLength]={');
fprintf(fid,'%f,\n',FIRcoef(1:length(FIRcoef)-1));
fprintf(fid,'%f};\n',FIRcoef(end));

fclose(fid);

% to call this function:
% coutput(FIRcoef,'FIRcoef.h')


%% compares two signals in time and frequency domain

load FIRcoef.dat
load impRes.dat

%%%%%%%
% plots

N = length(impRes);
Fs = 8000;
binWidth = Fs/N;
freqScale = 0:Fs/N:Fs-(Fs/N);
xLim = Fs/2;

figure
plot(freqScale,20*log10(abs(fft(impRes))), 'r')
hold on;
plot(freqScale,20*log10(abs(fft(impResED))), 'k')
title('a*(f)b(f)/a*(f)a(f)'); xlim([0 xLim])
legend('pressure FR','ED FR',3)

figure
subplot(1,2,1)
plot(impRes)
subplot(1,2,2)
plot(impResED)

%% complex math
% Complex math computations when real and  imaginary
% terms are stored in separate bins.
```

```
clc; close all;

a = 4;
b = -0.37;
c = -14;
d = 0.001;

A = 4-j*0.37;
B = -14 +j*0.001;

% complex multiplication
res1 = a*c-b*d + j*(a*d+b*c)
res2 = A*B

% complex division
res3 = (a*c+b*d)/(c^2+d^2) + j*(b*c-a*d)/(c^2+d^2)
res4 = A/B

% 1 divided by a complex number
res5 = a/(a^2+b^2) - j*b/(a^2+b^2)
res6 = 1/A

% complex exponential, e^(a+jb)
a = 3.4445223;
b = -1.5555432;
complex = a + b*j;
num1 = exp(complex)
num2 = exp(a)*cos(b) + j*exp(a)*sin(b)


%% autocorrelation

clc; close all;

x = [0 1 2 3 4 5 4 3 2 1];
autocor = zeros(1,length(x));

l = 1;

% first method
for m = 1:length(x)
    for n = l:length(x)
```

```
        autocor(m) = autocor(m) + x(n)*x(n-l+1);
    end
    l = l+1;
end

% matlab convolution
matCor = conv(x,x);

% plots
stem(autocor,'.')
hold on;
stem(matCor(length(autocor)+1:length(matCor)),'r:.')


%% simple circular buffer

clc; close all;

i = 1:32;
array = i.*ones(1, 32);

start = 17;
filtC = 0.5 * ones(1,32);

for j = 1:32
    startCount(j) = array(start);
    temp(j) = array(start)*filtC(j);

    start = start + 1;
    if(start > 32)
        start = 1;
    end
end

start


%% calculates twiddle factors on the DSP

clc; close all;

n = 128;
w = zeros(1,n);
```

```
delta = 2 * pi / n;

for i=1:1:n/2
    w(2*i-1) = cos((i-1)*delta);
    w(2*i) = sin((i-1)*delta);
end

w(1:n)'


%% calculates omega on the DSP

% The way the FFT works in CCS is that the right
% half has two less values than the left, and
% [512] in C is the midpoint

x=0:7;
fs = 10;
binWidth = fs/length(x);
omega = zeros(1,length(x));
for k=0:length(x)/2
    omega(k+1) = 2*pi*binWidth*k;
end

for k = 1:length(x)/2
    omega(length(omega)+1-k) = 2*pi*binWidth*k;
end


%% Convolution
% I have found through my own tests that the results of
% my linear convolution function match the results of
% the matlab conv( ) function, which runs much faster,
% thus I have chosen to do my testing using the matlab
% function knowing the process is equivalent to the DSP
% convolution

load impRes7.dat;
load FIRcoef7.dat;
load FIRcoef7ED.dat;

impRes = impRes7;
Wcoef  = FIRcoef7;
```

```
EDcoef = FIRcoef7ED;

[audio, fs1] = wavread('5kHzSound-0dB@.wav');

% deconvolution, tests impRes and FIRcoef together
z = conv(audio,impRes);
y = conv(z,FIRcoef);

wavwrite(z./(1.05*max(z)),Fs,'pressure@7')
wavwrite(y./(1.05*max(y)),Fs,'pressure@7filter@7')

% energy density

z = conv(audio,impRes);
wavwrite(z./(1.05*max(z)),'ED@7filter@7')


%% Sound pressure filtered-x algorithm with online systemID

% setup parameters
H        = impResDiffPos';
Hhat     = impRes';
idealFIR = FIRcoefDiffPos';

% W    = ones(1,numCoefs)*0.001; %white noise
% W(length(W)/2) = 0.5;
% W = FIRcoefPlugsRem';
W = FIRcoef'*0.5;

t = 600;     % run time in seconds
fs = 3000;   % Hz
stop = t*fs;

% random noise
x = randn(1,t*fs);
x = x./max(abs(x));

% filter lengths
numCoefs = length(impRes);
% mu
% mu  = 0.0007; % for FIRcoefPlugsRem
% mu1 = 0.002;  % for FIRcoefPlugsRem
mu  = 0.0005; % for FIRcoefDiffPos
```

```
mu1 = 0.005;  % for FIRcoefDiffPos

% delay
delay = 150;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% initialize filters

Z    = zeros(1,delay);
Z(delay) = 1;

rBuffer = zeros(1,numCoefs);
uBuffer = zeros(1,numCoefs);
xBuffer = zeros(1,numCoefs);
zBuffer = zeros(1,delay);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the filtered-X algorithm

e = zeros(1,stop);

% algorithm
for n = 1:stop
    % first stage

    % previous inputs to W and Hhat
    xBuffer = [x(n) xBuffer(1:numCoefs-1)];
    % delay buffer for d(n)
%   zBuffer = [x(n) zBuffer(1:delay-1)];
%   d       = sum(Z.*zBuffer);

    % output of W
    u       = sum(W.*xBuffer);
    % previous inputs to H
    uBuffer = [u uBuffer(1:numCoefs-1)];
    % system output
    dHat    = sum(H.*uBuffer);

    %  W update
    d = x(n);

    e(n)   = d - dHat;
    eMu = e(n)*mu; % computes new error signal
```

```
    % output of Hhat
    r       = sum(Hhat.*xBuffer);
    % stored for error update
    rBuffer = [r rBuffer(1:numCoefs-1)];

    % LMS update; updates W coefficients
    W = W + eMu.*rBuffer;

    % H update
    HhatOut = sum(Hhat.*uBuffer);

    e2    = x(n)-HhatOut;
    e3    = e2-e(n);
    e3mu1  = e3*mu1; % computes new error signal

    % LMS update; updates H coefficients
    Hhat = Hhat + e3mu1.*uBuffer;
end


%% energy density filtered-x algorithm with online systemID

load impRes7.dat; % Fs = 3000 Hz
load impRes8.dat;
load FIRcoef7ED.dat;

load H1coefLP600Fs2000.dat;
load H2coefLP600Fs2000.dat;

% setup parameters
impRes1 = H1coefLP600Fs2000./max(abs(H1coefLP600Fs2000));
impRes2 = H2coefLP600Fs2000./max(abs(H2coefLP600Fs2000));

% impRes1 = filter(lowPass,1,impRes1);
% impRes2 = filter(lowPass,1,impRes2);

idealFIR = FIRcoef7ED; % compare converged coefs to this

W = zeros(1,length(idealFIR)); %initialize W
% W(1) = 0.2;
% W = FIRcoef7ED'*0.5;
```

```
mu = 1e-5; % mu 2e-4 works with unscaled impRes
const = 0.01; % 0.03 convergence constant for v(n) averaging
% 0.1 too big,
delay = 12; % holds x(n)

rho = 1.21;
c = 343;
delx = 0.05;
t = 300;     % run time in seconds
fs = 2000;   % Hz
stop = t*fs;

% random noise
x = randn(1,t*fs);
x = x./max(abs(x));
% x = filter(highPass,1,x);

% initialize filters

numCoefs = length(impRes1); % filter lengths
H1 = zeros(1,numCoefs);
H2 = zeros(1,numCoefs);
H1hat = zeros(1,numCoefs);
H2hat = zeros(1,numCoefs);

H1 = impRes1';
H2 = impRes2';
H1hat = impRes1';
H2hat = impRes2';

rpBuffer = zeros(1,numCoefs);
rvBuffer = zeros(1,numCoefs);
uBuffer = zeros(1,numCoefs);
xBuffer = zeros(1,numCoefs);

Z = zeros(1,delay); % holds x(n)
Z(delay) = 1;
zBuffer = zeros(1,delay);

alpha = -1/(rho*delx*fs); %*exp(-1/fs);
lambda = 0.999666;

yp = 0;
```

```
yv = 0;

ep = 0;
ev = 0;

v  = 0;
rv = 0;

rvCurr = 0;
vMean  = 0;
rvMean = 0;

e = zeros(1,stop);

% the filtered-X algorithm

% algorithm
for n = 1:stop

    % FIR filtering
    % previous inputs to W and Hhat
    xBuffer = [x(n) xBuffer(1:numCoefs-1)];
    % delay buffer for d(n)
    zBuffer = [x(n) zBuffer(1:delay-1)];

    d = sum(Z.*zBuffer);
    % output of W
    u = sum(W.*xBuffer);
    % previous inputs to H
    uBuffer = [u uBuffer(1:numCoefs-1)];

    y1 = sum(H1.*uBuffer); % system output
    y2 = sum(H2.*uBuffer);

    % measured pressure and particle velocity
    yp = (y1+y2)/2; % pressure

    % particle velocity error
    v = lambda*v + alpha*(y2-y1);

    vMean = const*v + (1-const)*vMean;
    yv = v-vMean;
```

```
    % compute errors
    ep = d - yp;
    ev = d/(rho*c) - yv;

    % filtered-x

    H1hatout = sum(H1hat.*xBuffer);
    H2hatout = sum(H2hat.*xBuffer);

    rp = (H1hatout+H2hatout)/2; % pressure estimate

    % estimated particle velocity
    rvCurr = lambda*rvCurr + alpha*(H2hatout-H1hatout);

    rvMean = const*rvCurr + (1-const)*rvMean;
    rv = rvCurr - rvMean; % particle velocity estimate

    % stored for error update
    rpBuffer = [rp rpBuffer(1:numCoefs-1)];
    rvBuffer = [rv rvBuffer(1:numCoefs-1)];

    W = W + mu*(ep + rho*c*ev)*rpBuffer; % LMS update
    e(n) = ep + rho*c*ev;
end


%% FIR filter expanded

load impRes.dat;
load FIRcoef.dat;

[audio, Fs] = wavread('sound8kHz');

x = audio;
impRes = impRes;

% linear convolution with the system
y = 0;

M = length(impRes);
xBuffer = zeros(size(impRes));
start = round(M/2);
```

```
% FIR filter w/ circular buffer
for n = 1:length(x)
    start = start - 1;

    if(start < 1)
        start = M;
    end

    temp = 0;
    xBuffer(start) = x(n);

    for j = 1:M
        temp = temp + impRes(j) * xBuffer(start);

        start = start + 1;
        if(start > M)
            start = 1;
        end
    end

    z(n) = temp;
end


%% Creating a waterfall plot from a time recording
% (frequency vs. time)

clear all; close all; clc;

fid=fopen('ID004_004.bin','r');
rec=fread(fid,inf,'single')';

fid=fopen('ID004_007.bin','r');
ref=fread(fid,inf,'single')';

fs = 6000;
N = 2048;
xLim = 1000; % Hz

tScale = 0:1/fs*N:length(rec)/fs - (1/fs*N);
freqScale = 0:fs/N:fs-(fs/N);

p = round(length(rec)/N); % # of possible averages
```

```
recFFT = zeros(p,N); % FFT
refFFT = zeros(1,N); % FFT

%hanning window on blocks
n = 0:N-1;
hann = 0.5*(1-cos(2*pi*(n./N)));
recTW = 0; % applies time window
refTW = 0;

% averaged reference magnitude
for i=1:p
    refTW = ref((1+(i-1)*N):(N*i)).*hann;
    refFFT = refFFT + 1/p*abs(fft(refTW));
end

% averaged pressure magnitude divided by reference
for i=1:p
    recTW = rec((1+(i-1)*N):(N*i)).*hann;
    recFFT(i,:) = 20*log10(abs(fft(recTW) ./ refFFT)) + 38;
end


%*********************************************************
% plots

% 3D surface plot

start = 35;   % 4096
stop  = 685;

start = 18;   % 2048
stop  = 343;

f = freqScale(start:stop);
x = tScale;
[F,X] = ndgrid(f,x);

A = recFFT(1,start:stop)';
for i=2:p
    A = [A recFFT(i,start:stop)'];
end
```

```
figure
surf(X,F,A)
shading interp
xlabel('Time (s)'); ylabel('Frequency (Hz)');
zlabel('Magnitude (dB)')
view([150 60])
axis square
colorbar
set(gcf,'Position', [157 37 790 686])

figure
surf(X,F,A)
shading interp
xlabel('Time (s)'); ylabel('Frequency (Hz)');
zlabel('Magnitude (dB)')
view([0 90])
colorbar
set(gcf,'Position', [157 37 790 686])


%% Creating 3D spectral plots from several
% simultaneous recordings in a sound field

% Parameters to be changed:

clear all; close all; clc;

testname = 'ID';
ID = 017;
fs = 5000;
CH = 0:15;
numChan = length(CH);
N = 4096;

for n=1:numChan
    if ID<10
        IDstring='00';
    elseif ID<100
        IDstring='0';
    else
        IDstring='';
    end
    if CH(n)<10
```

```
        CHstring='00';
    elseif CH<100
        CHstring='0';
    else
        CHstring='';
    end

    filename=[testname,IDstring,int2str(ID),...
    '_',CHstring,int2str(CH(n)),'.bin'];
    fid=fopen(filename,'r');
    x=fread(fid,inf,'single');
    Data(n,:)=x;
    fclose(fid);
end

%%%%%%%%%%%%%%%%%%%%%
%Loading data complete
%%%%%%%%%%%%%%%%%%%%%

%part a
p = length(Data(1,:))/N; % # of possible averages

FFTavg   = zeros(numChan,N); % FFT
FFTpress = zeros(numChan,N); % FFT
FR       = zeros(numChan,N); % FR

%hanning window on blocks
n = 0:N-1;
hann = 0.5*(1-cos(2*pi*(n./N)));
timeWindowed = 0;

% first half
for i=1:16
    temp = Data(i,:);

    for j=1:p
        timeWindowed = (temp((1+(j-1)*N):(N*j))).*hann;
        FFTavg(i,:) = FFTavg(i,:) + 1/p*abs(fft(timeWindowed));
    end
end

%*******************************************************
% plots
```

```
freqScale = 0:fs/N:fs-(fs/N);
xLim = 2000; % Hz
magScale = 1500;

% % pressure
% for i=1:numChan-2
%     figure
%     FFTpress(i,:) = 20*log10(FFTavg(i,:).*0.7);
%     plot(freqScale,FFTpress(i,:))
%     xlim([0 2000])
% end

% transfer function
for i=1:7
%     figure
    FR(i,:) = 20*log10(FFTavg(i,:)./(FFTavg(15,:)).*magScale);
%     plot(freqScale,FR(i,:))
%     xlim([0 xLim])
end

for i=8:14
%     figure
    FR(i,:) = 20*log10(FFTavg(i,:)./(FFTavg(16,:)).*magScale);
%     plot(freqScale,FR(i,:))
%     xlim([0 xLim])
end

% 3D surface plot

start = 42;  % N = 4096
stop  = 1639;

f = freqScale(start:stop);
x = 1:numChan-2;
[F,X] = ndgrid(f,x);

A = FR(1,start:stop)';
for i=2:numChan-2
    A = [A FR(i,start:stop)'];
end

figure
```

```
surf(X,F,A)
shading interp
xlabel('Microphone Position'); ylabel('Frequency (Hz)');
zlabel('Magnitude (dB)')
caxis([-55 50])
axis([1 14 50 2000 -50 45])
view([150 60])
axis square
colorbar
set(gcf,'Position', [157 37 790 686])

figure
surf(X,F,A)
shading interp
xlabel('Microphone Position'); ylabel('Frequency (Hz)');
zlabel('Magnitude (dB)')
caxis([-55 50])
axis([1 14 50 2000 -100 50])
view([0 90])
colorbar
set(gcf,'Position', [157 37 790 686])
```