Inverse Filter*

CJ Steuernagel

Abstract

This module describes how we implemented the inverse filter for the laser microphone

We observed that the system did not transmit sound information perfectly, and transmitted speech signals suffered some distortion. This distortion happens for two reasons: (1) the physical properties of the glass cause it to respond differently to different frequencies, and (2) low-frequency vibrations caused by air-conditioning systems and other building vibrations are constantly present in the window. We attempted to compensate for this observed distortion by building an inverse filter. We accomplished this in three steps:

1 Step 1: Measure the Frequency Response

In order to accurately model the system, we needed to measure its frequency response. We blasted a 30-second sound clip of pure white noise at the window and recorded the signal measured by the detection unit. Since we knew the input of the system (the white noise) had a completely flat spectrum, the output's spectrum should represent the frequency response. To compute the spectrum of the output (the recorded signal), we windowed portions of the signal using a Hamming window, computed the FFT's of each windowed portion, and then averaged the FFT's. This average FFT represents the frequency response of our system.

*Version 1.1: Dec 19, 2007 5:19 pm -0600
†http://creativecommons.org/licenses/by/2.0/

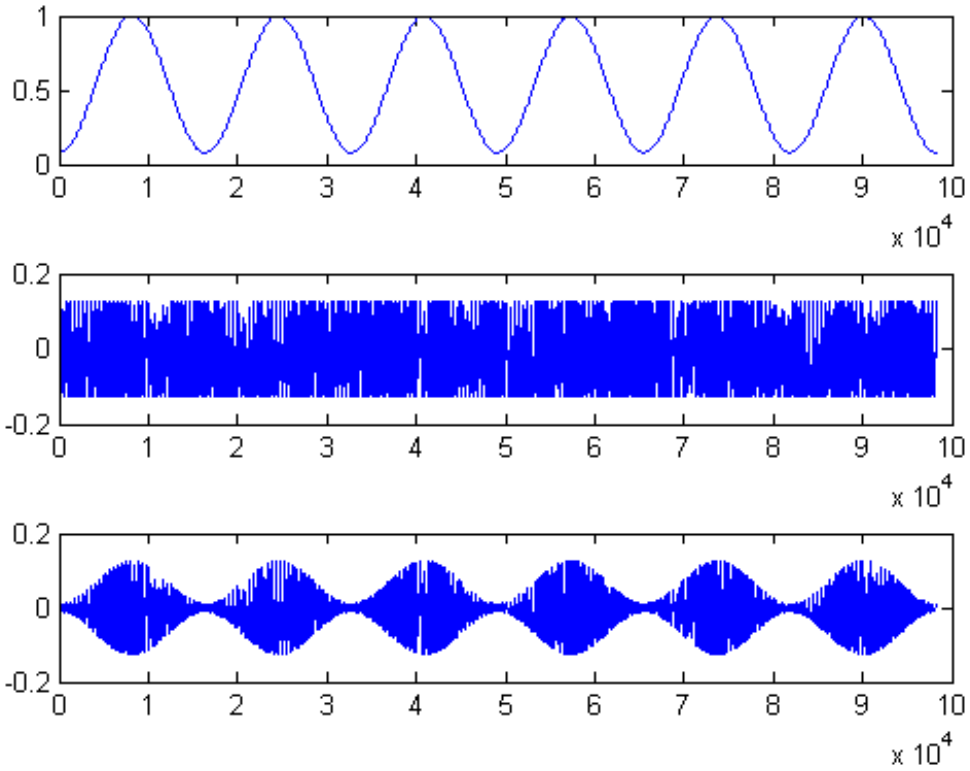
1
0.5
0
0 1 2 3 4 5 6 7 8 9 10
x 10^4
0.2
0
-0.2
0 1 2 3 4 5 6 7 8 9 10
x 10^4
0.2
0
-0.2
0 1 2 3 4 5 6 7 8 9 10
x 10^4

Figure 1

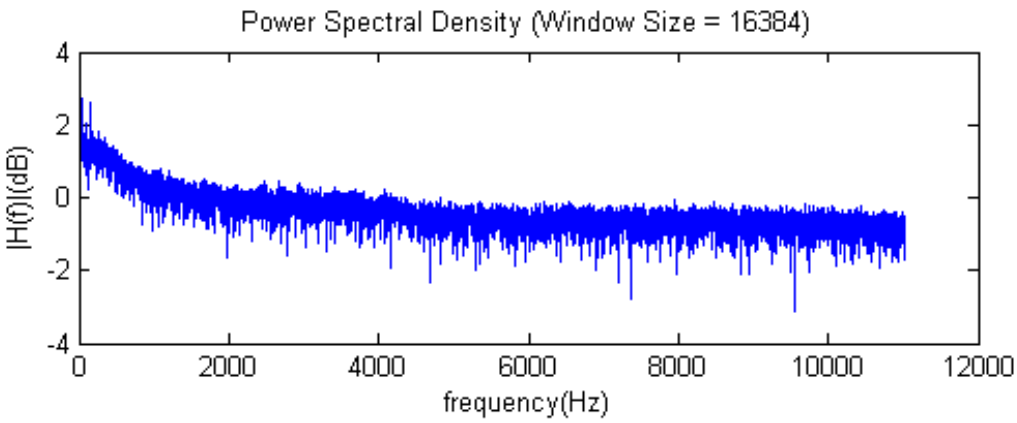

Figure 2

The plot shows some strong low-frequency vibrations in the window. We attributed these to the air-conditioning unit in the building and to other random vibrations in the environment. We also noticed that the window responded better to low frequencies than to high frequencies. This could be a result of the physical properties of the glass as well as the physical dimensions of the window.

2 Step 2: Model the System

Once we had a good idea of the system's frequency response, we attempted to model the system using a linear prediction filter. We used a linear prediction filter because it made the inverse filter simple to implement, and it guaranteed that the inverse filter would be inherently stable and have a linear phase response. A linear prediction filter estimates its next output by the current input and a linear combination of n previous outputs:

$$y[n] = g \cdot x[n] + \sum_{k=1}^{N} a_k \cdot y[n-1]$$

Figure 3

The first step to building this filter is to compute the autocorrelation coefficients of the recorded signal. The autocorrelation coefficients are a measure of the correlation between samples of the signal. Since the filter must accurately estimate the output based on previous outputs, it must preserve the correlation between samples. One autocorrelation coefficient r[i] can be expressed as:

$$r_i = \sum_{n=0}^{N-i+1} s[n]s[n+i]$$

Figure 4

The next step to building the filter was to compute the filter coefficients. We used a recursive algorithm called Burden's Algorithm to do this. We set the first coefficient a[0] = 1 and then compute the other coefficients recursively:

$$K_i = -\frac{r_i + a_1 \cdot r_{i-1} + \cdots + a_{i-1} \cdot r_i}{E_{i-1}}$$

$$E_i = (1 - K_i)^2 \cdot E_{i-1}$$

$$a_i = K_i$$

Figure 5

We could perform this recursion as many times as we needed to compute the desired amount of coefficients. We wrote a MATLAB program to perform the algorithm N times on the windowed signal to generate N coefficients. We used these coefficients in the feedback branches of the filter. We found that we could accurately model the system using a linear prediction filter with 50 coefficients. The frequency response of this filter has a similar shape to the measured frequency response of the system:

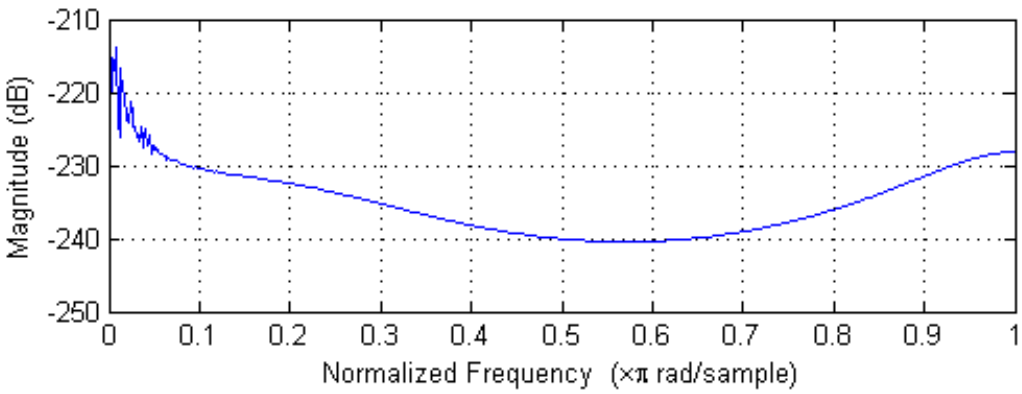

Figure 6

Step 3: Build the Inverse Filter

3 Step 3: Build the Inverse Filter

The linear prediction filter is simple to invert. Since it uses only the previous outputs to generate the next output, it is an all-pole filter with only feedback branches. To build the inverse filter, we used all the feedback coefficients that we generated using Burden's Algorithm as the feed-forward coefficients of the inverse filter. The frequency response of the inverse filter looks like:

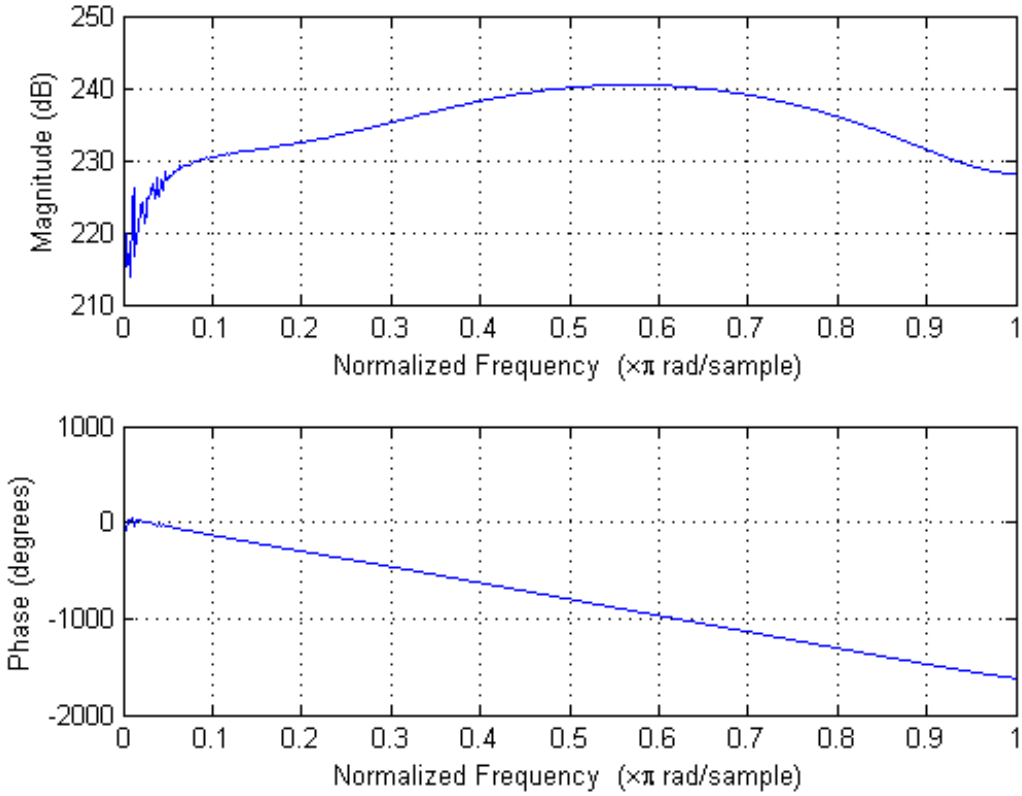

Figure 7

We observed that the inverse filter accurately inverted the response of the system. It successfully attenuated the low-frequency window vibrations, and it amplified the higher frequencies that the system attenuated.