Semester Project

Building an Equalizer

EE 262 Lab Section A2
Department of Electrical Engineering
University of Missouri-Rolla

EE262 A2
Eddie Brown
Randy Dimmett
Almir Mutapcic
December 07, 1998

Abstract:

Equalization is a signal processing technique that cancels out distortions caused by noise sources and imperfections in mediums. This report describes the procedure and results of simulating an equalizer using MATLAB. The equalization process has many real-world applications such as removing distortion of audio signals due to poor acoustics in a room. The methods for determining frequency response of a sound distorting room are also discussed.

Table of Contents:

Introduction:

High-fidelity audio systems are present in nearly every household around the world. Today great sounding music is produced by a multitude of electronic gadgets such as digital CD players, minidisc players, DVD systems, and even computers implementing new technologies such as mpeg-layer3 and realaudio encoding. Most people are satisfied with the quality of produced music; however, many do not know that this quality could be improved. Equalization, a method that cancels out sound distortion due to imperfections in the speaker or due to poor acoustic features of a room, can greatly improve output of audio systems.

The concept of equalization is not limited to sound and acoustic applications. It has been successfully implemented in applications such as removing distortion in a telephone line, radio frequency link, and digital communication channels.

This report describes the design process for an equalizer utilizing MATLAB. The methods for determining the frequency response of an arbitrary room are discussed in Background and Theory.

Background and Theory:

Ideally, a microphone in a recording studio will produce a voltage that is proportional to the air pressure in the room. The signal in the room is captured and recorded on a compact disk. The CD may be read by a CD player that exactly reproduces this voltage from the disk, and passes it to an amplifier. The amplifier multiplies this signal by a constant, and passes it to a speaker. The speaker produces an air-pressure change that is proportional to the voltage. The pressure wave travels, undistorted, through the air, to reach the listener's ear.

In the real world, the recording studio, CD player and amplifier are all close to ideal. However, the speaker and room usually introduce significant distortion. An equalizer must be built to undo the distortion of the speaker and the room by adjusting the CD player's output before it is sent through the room. It is assumed the speaker and room look like a linear, time-invariant filter with another unique frequency response. Since the equalizer is in series with the room, the overall frequency response is the product of the two individual frequency responses. If the equalizer's frequency response is equal to the inverse of the room's, then the equalizer will exactly remove the effects of the room (see figure 1).

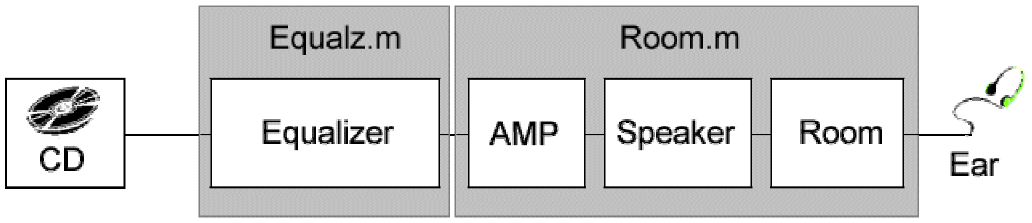


Figure 1. Equalizer diagram

To design an equalizer, the frequency response of the room must be measured. The most commonly used method for frequency response measurement is to place a microphone at the location of listener's ear. Then, the frequency response of the room can be obtained by comparing the voltage coming out of the CD player and the voltage recorded by the microphone, using the following relationship:

H_{room} = (frequency response) = (frequency response of Vout) / (frequency response of Vin)

If an impulse signal is used as input for testing purposes, the frequency response of the voltage input is one.

Therefore, if an impulse is used as the input, H_{room} = (frequency response of output)

MATLAB's function `fft()` can be used to determine the frequency response of a signal, so that in MATLAB, H_{room} = `fft(y)`, where *y* is the output of the room (what the ear hears).

The equalizer filter then should be defined as the inverse of the room's response so the room's response is cancelled and the final output signal is the same as the CD output *(these relationships work only in frequency domain)*:

$$\text{Ear\_In} = (\text{CD\_Out}) \mathbf{*} (H_{equal}) \mathbf{*} (H_{room})$$

If H_{equal} (the response of the equalizer filter) is defined to be the inverse of H_{room} (the response of the room) the resulting output (Ear_In) is very similar to the original CD source.

$$\text{Ear\_In} = (\text{CD\_out}) * (1/H_{room}) * (H_{room}) = \text{CD\_out}$$

The above method works well because the impulse signal has the optimal frequency response for equalizer application (it's uniform for all frequencies). However, the impulse is an ideal theoretical signal that is very hard to produce in practice. Therefore, different test signals such as chirp and step could be used instead of the impulse signal, even though they give only an approximate frequency response of the room.

Procedure, Results, and Analysis:

Implementing the equalizer in MATLAB

Equalz.m, a MATLAB M-file, was created to read in a CD output waveform from a file *cdout.wav*. The equalizer filter was constructed and applied to this input signal creating a pre-distorted version of CD output that was saved in the file *ampin.wav*. This pre-distorted signal was passed through the room creating an output signal (saved to *earin.wav*), that was compared to the original CD ouput.

Determining the frequency response of sample rooms

A MATLAB filter function was given to describe the behavior of a room. To determine the frequency response of the room, an impulse signal was sent through the filter model (transfer function) of the room. The resulting output was the impulse response of the room. The frequency response was obtained by taking the fast Fourier transform (fft) of the impulse response.

Designing the equalizing filter and realizing design tradeoffs

The equalizer's frequency response was then obtained by taking the inverse of the room's frequency response. To prevent infinite spikes in the equalizer response, a small value (in the range of about 20-30 dB) was added to the room's frequency response to get rid of any possible zeros. This step is necessary for some filters, but adding to the frequency response results in unwanted noise and inaccuracies in the final result. This really wasn't a problem, however, since the value added was so small.

Measuring equalizer quality

To measure the quality of the equalizer, the final signal’s spectrogram was compared to the spectrogram of the CD output. The two signals were also sounded and observed.

Types of filters tested

Room0.m filter:

> The first filter testing was a simple filter with the following transfer function:
> H_room = $3s^3+5s^2+3s+3$
>
> The equalizer worked very well for this simple filter.

Room1.m filter:

> The second filter tested was a Yulewalk filter. The results were good for this filter as the CD output and speaker output waveforms matched together.

Room2.m filter:

> The third filter tested was a Chebychev filter. The equalizer did *not* work well with this filter until 20-30 dB was added to the Chebychev response to remove the zeroes in frequency response. By adding the uniform noise, the equalizer's poles were moved inside the stability region of the z-plane unit circle.

See appendix II for various plots from each test of these filters.

<u>Notes about `wavread()` and `wavwrite()` functions</u>

The MATLAB functions `wavread()` and `wavwrite()` had to be used carefully or else various problems would arise. `Wavwrite()` writes signals with maximum amplitudes of ±1 volts. Therefore, before writing any signal to a *wav* file, it had to be normalized or else any amplitude above ±1 volts would be clipped.

However, because of normalizing, another problem occurred. The original CD output volume was not preserved. Hence the volume level of the final output signal was not the same as the input volume, and in some cases, had to be greatly adjusted, which was implemented with *volume.m.*

Conclusion:

This project involved constructing an equalizer that reversed the distorting effects caused by a given room.

This procedure was accomplished by first measuring the frequency response of the room. Then, the equalizing filter was designed by inverting the room's frequency response utilizing MATLAB. Certain amount of uniform noise was added to the room's frequency response in order to prevent division by zero when calculating the equalizer's response. However, this design tradeoff does not greatly effect the quality of the produced time-domain signal.

The quality of the signals was verified by visual comparisons of the frequency response and spectrograms, as well as by comparing audio outputs.

The performance of the equalizer was tested in opposition of three different filters. The obtained results were independent of the used filter type, verifying that the equalizer design should work for any reasonable situation.

Appendix I – MATLAB code

Script: Equalz.m

```
% EQUALZ.M - Equalizer script (EE 262 semester project)
%
%   Eddie Brown
%   Randy Dimmett
%   Almir Mutapcic
%
%   Linear Systems II - EE 262
%   December 07, 1998
%
%   This script designs an equalizer for given room specs.
%   The sound file is processed through the equalizer filter
%   to counter and correct distortions produced in the sound
%   wave due to the room's filtering effects

% prepare workspace
close all; clear all;
format compact; clc;
disp(' Semester Project - Building an Equalizer');
disp('=---------------------------------------=');
disp(' ');

% input room filter filename
roomfile=input('Enter room file: ','s');
[cdout, fs]=wavread('CDOUT.WAV');

% impulse signal
t=[1:length(cdout)]/fs;
x=t*0;x(1)=1;
x=x/max(abs(x));
wavwrite(x,fs,16,'AMPIN.WAV');

run(roomfile)
roomf=abs(fft(y));

% Defining and adjusting frequency scale
N=length(roomf);
freq=[-N/2:(N/2)-1];
freq=freq*fs/N;

% defining equalizer
equalf=1./(roomf+0.003);

% passing cdout through equalizer filter
cdoutf=fft(cdout);
ampinf=cdoutf.*equalf;

ampin=real(ifft(ampinf));
ampin=ampin-mean(ampin);
ampin=ampin./max(abs(ampin));
wavwrite(ampin,fs,16,'AMPIN.WAV');

% presenting results (plots and sounds)
pick=0;
while(pick < 7)
   pick=menu('Pick a Plot',...
      'Frequency Response of Room',...
      'Frequency Response of Equalizer',...
      'DB Plot of Room and Equalizer',...
      'Spectagram Plot of CD ouput',...
      'Spectagram Plot of Equal output',...
      'Process the room fiter',...
      'Quit');
   switch pick
   case 1
      figure
      plot(freq, fftshift(abs(roomf)));
      title('Frequency response of the room')
      ylabel('Magnitude'); xlabel('Frequency (Hz)');
      zoom on; grid on;
   case 2
      figure
      plot(freq, fftshift(abs(equalf)));
      title('Frequency response of the equalizer')
      ylabel('Magnitude'); xlabel('Frequency (Hz)');
      zoom on; grid on;
   case 3
      figure
      plot(freq, fftshift(20*log10(abs(equalf))),'r--'); hold on;
      plot(freq, fftshift(20*log10(abs(roomf)))); hold off;
      title('dB plot of the room and the equalizer')
      ylabel('Magnitude (dB)'); xlabel('Frequency (Hz)');
      legend('Equalizer','Room');
      zoom on; grid on;
   case 4
      figure; specgram(cdout)
      title('Spectrogram of cdout signal')
   case 5
      figure; specgram(ampin)
      title('Spectrogram of ampin signal')
   case 6
      run(roomfile)
      spick=0;
      while(spick < 4)
      spick=menu('Room filter controls',...
      'Volume optimization',...
      'Sound CD output',...
      'Sound optimized speaker output (earin)',...
      'Back to the main menu');
       switch spick
        case 1
           volume;
           disp(' '); disp('Optimized volume is '), V
        case 2
          sound(cdout)
        case 3
          volume;
          sound(V*y)
        end
      end
   end
end
```

Script: Volume.m

```
% VOLUME.M - Volume adjustment for equalizer script
%
%   Eddie Brown
%   Randy Dimmett
%   Almir Mutapcic
%
%   Linear Systems II - EE 262
%   December 07, 1998
%
%   This script computes V, the optimal volume adjustment
%   for speaker output signal produced by script EQUALZ.
%   Usage: sound(V*y) should be equal to sound(cdout)

ywindow=y(ceil(.1*length(y):length(y)));
V=abs(max(cdout))/abs(max(ywindow));
```

Test room scripts (room0.m, room1.m, and room2.m)

Script: room0.m

```
[x,fs]=wavread('AMPIN.WAV');
b=[3 5 3 3];
a=[1];

y=filter(b,a,x);
wavwrite(y,fs,16,'EARIN.WAV');
```

Script: room1.m

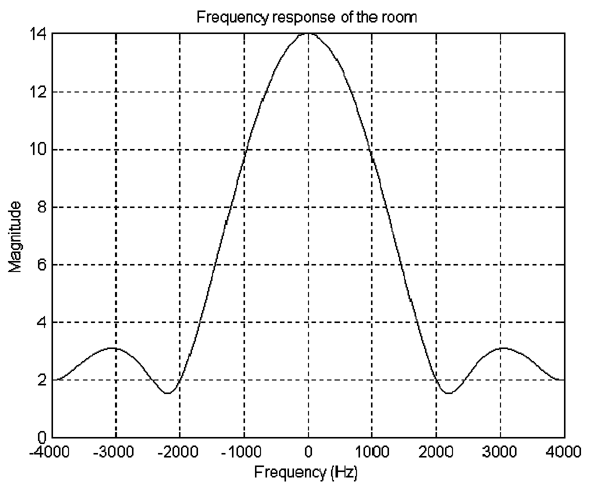
```
[x,fs]=wavread('AMPIN.WAV');
[b,a]=yulewalk(5,[0 1000/(fs/2) 3000/(fs/2) 1],[1 1 .1 .1]);

y=filter(b,a,x);
wavwrite(y,fs,16,'EARIN.WAV');
```

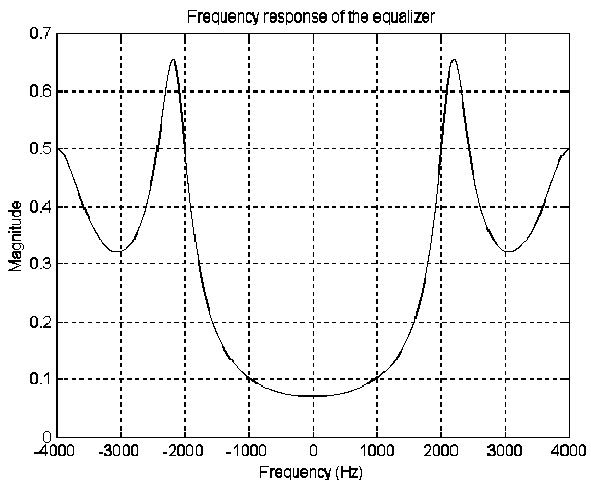
Script: room2.m

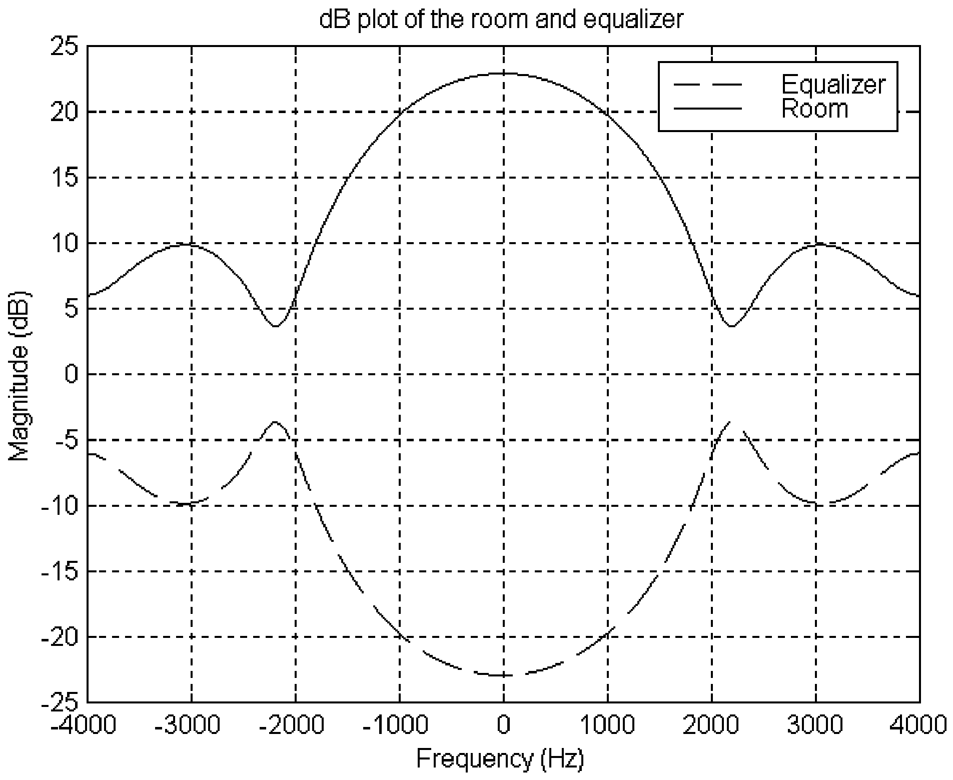
```
[x,fs]=wavread('AMPIN.WAV');
[b,a]=cheby2(7,50,.3);

y=filter(b,a,x);
wavwrite(y,fs,16,'EARIN.WAV')
```

Appendix II – Plots and Graphs

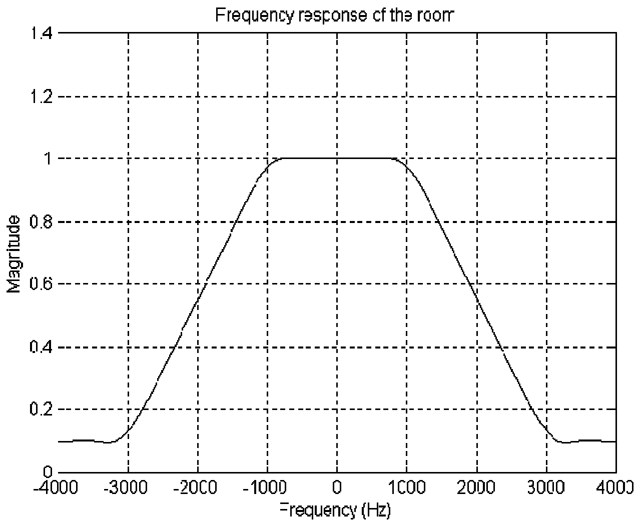
$H_{room} = 3s^3+5s^2+3s+3$

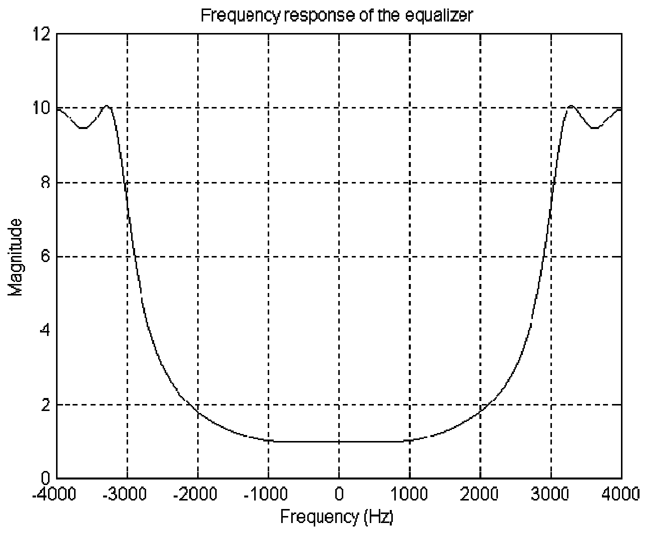


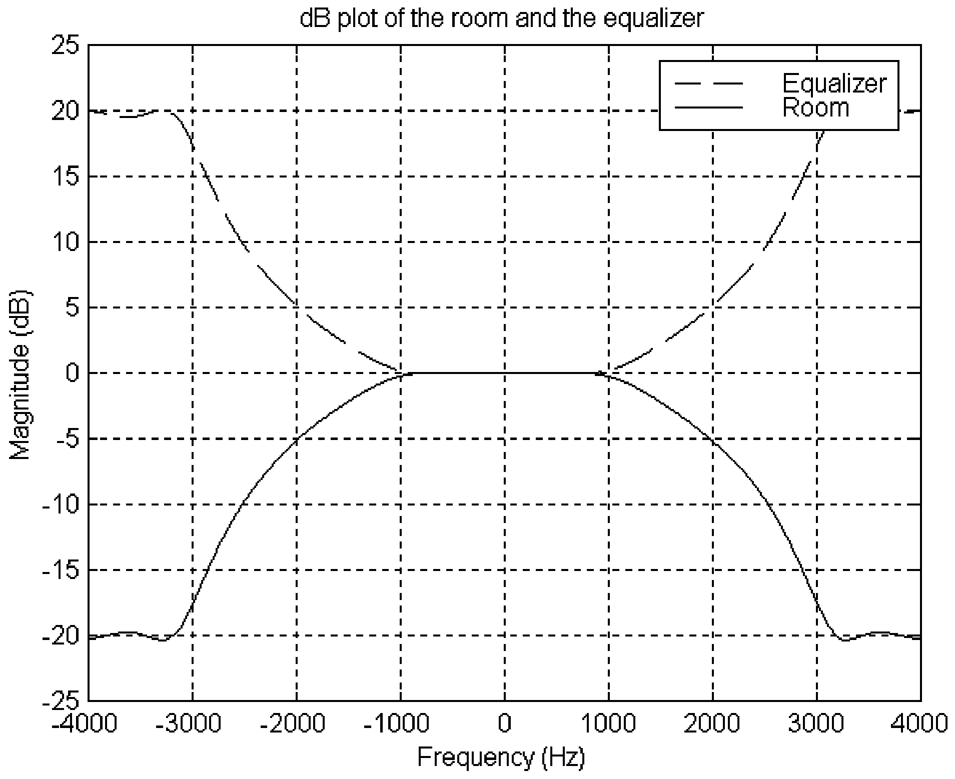




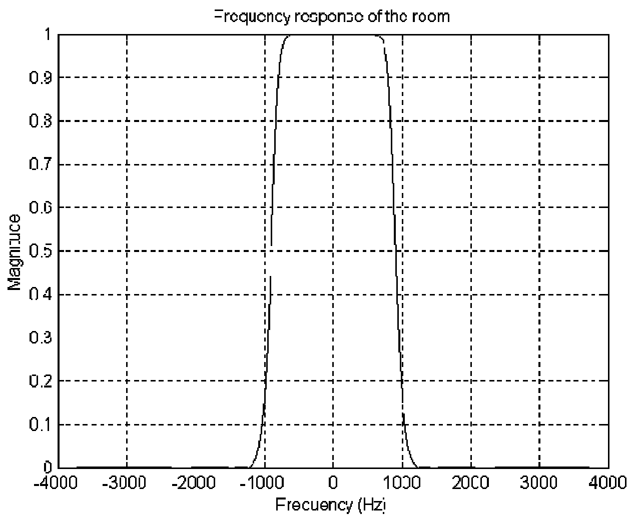
H_{room} = yulewalk(5,[0 1000/(fs/2) 3000/(fs/2) 1],[1 1 .1 .1]);

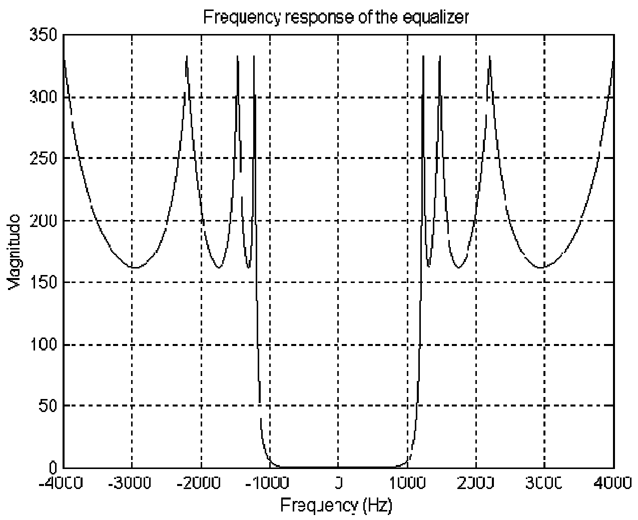


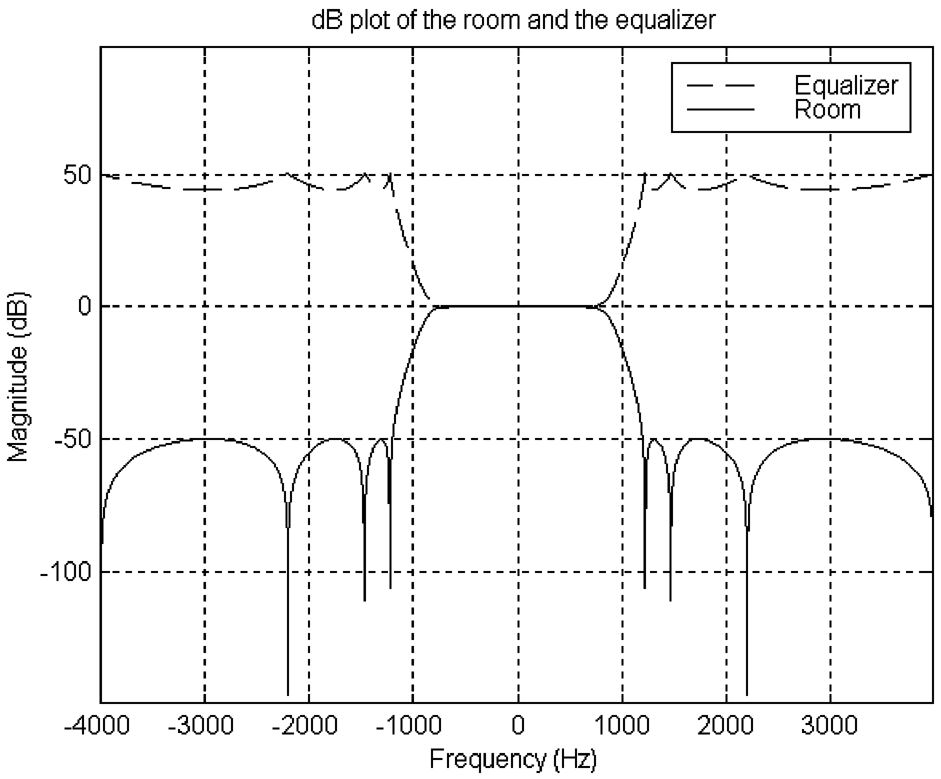




H_{room} = cheby2(7,50,.3)







π