EE 341
Lab 4: Digital Filtering and Music Equalizer

Note: *Each group provides only one report for all group members. All MATLAB files (.m files) must be uploaded.*

Part I: Digital Filtering

In this section, we will consider different types of digital filters and look at their characterization in different domains. This will give you some insight into how digital filters are implemented and designed and into the properties of different digital filter design algorithms. Most parts of this lab are given from Mari Ostendorf class website for EE 341, Spring 2014.

1. **Filter Implementation**

Digital filters are usually implemented using a difference equation

$$y[n]=(1/a_0)[b_0x[n] + b_1x[n-1] + \ldots + b_Mx[n-M] - a_1y[n-1] \ldots - a_Ny[n-N]]$$

assuming $a_0=1$ and a causal filter. In MATLAB, this is implemented with the filter command:

$$y=filter(b,a,x)$$

where vector $a=[\ a_0 \ldots a_N],\ b=[\ b_0 \ldots b_M],$ and x and y are finite-length input and output sequences. You can also implement a filter using the MATLAB convolution function (conv) given an impulse response. In order to understand the implementation differences, we will experiment with a moving average filter. Get a text file containing Microsoft stock price over 4 years from the class web site. Load the data using load command, and plot it. Note that the quick changes correspond to the high frequency component. People tend to invest based on the long-term trend, which we can find using a moving average filter.

Consider the impulse response for a 30-day moving average filter:

$h[n]= 1/30\ (\ u[n] - u[n-30]\ )$

which corresponds to

$y[n]=1/30(\ x[n] + x[n-1] + \ldots x[n-29]\ )$

Implement this filter in MATLAB using both the conv and filter functions. Then apply this filter with stock data and plot in red color with the original graph. Does the filtered result look smoother? Plot the DTFT magnitude of $h[n]$ to verify that this system corresponds to a low pass filter.

WRITTEN REPORT: Turn in your plot of the original and filtered stock data, and the plot of the system frequency response. Be sure to label the axes of the plots. Explain the differences in the results of the two implementations. How would you adjust the implementations so that they gave identical results?

2. **Filter Characterization**

For the next part of this assignment you will use frevalz01.m (on the class web page) to look at the behavior of system functions in the *z*-plane. (The *z*-plane is the DT version of the Laplace s-plane for CT systems.) This time the system functions you will examine will be various digital filters that you use MATLAB to design. MATLAB has numerous built-in functions for generating discrete time filters. In this problem we are going to use two to look at how FIR vs. IIR systems behave in the *z*-plane. This will give you some insight into how digital filters are designed and into the properties of different digital filter design algorithms. You will not be expected to understand the details of how these algorithms work; you only need to evaluate their behavior.
Both the filter design functions in the problems below ask you to specify cut-offs W in terms of a normalized frequency $0 < W < 1$, where $W = 1$ corresponds to half the sampling frequency. (Recall that normalized frequency is periodic with period 1, and [0,0.5] in normalized frequency corresponds to $[0, \pi]$ in radians for the DTFT.) So, a cut-off frequency of $0<W<1$ corresponds to $0 < \omega < \pi$ in radians for the DTFT, and $0<f<0.5$ in normalized frequency as plotted by frevalz01. Thus, a cut-off frequency of $W = 0.5$ corresponds to 2 $\omega_c = \pi$ / in radians and f_c=0.25 in normalized frequency on the frevalz01 frequency response.

a) **FIR (Finite Impulse Response) Digital Filters**
Use the MATLAB function fir1 to create a low pass FIR filter of order 10 with cutoff frequency of $.3\pi$. Use frevalz01 to study the system. Save your filter in a vector since you'll use it again in part 2c.

b) **IIR (Infinite Impulse Response) Digital Filters**
Use the MATLAB function butter to create a low pass Butterworth filter with cut-off frequency $.3\pi$ and filter order of 10. Save your filter in another vector since you'll use it again in part 2c.

c) **Filter Implementation**
Use the MATLAB function filter to implement the two low pass filters you produced in 2a and 2b. Apply the filters to:

- the stock market data;
- a pulse of length twenty: $x[n] = u[n] - u[n-20]$ where $x[n]$ is of total length 60 (so append 40 zeros on the end); and
- the *music* data on the class web site.

WRITTEN REPORT: Turn in your frevalz01 plots of the filter responses and time-domain plots of the filtered stock market and pulse signals comparing the different filters. Comment on the differences in the frequency response of the two filters (magnitude and phase) and how this impacts the outputs. In commenting on frequency responses, consider how close a filter matches an ideal low pass filter.

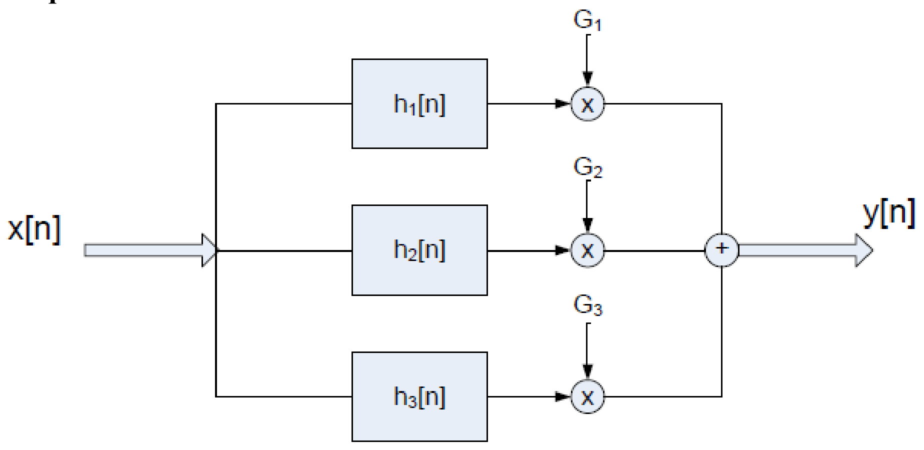
Part II: Music Equalizer

In this section, you will design a music equalizer using components designed by former EE grad student (and 341 TA) Somsak Sukittanon. The idea of an equalizer is to break the sound file into multiple frequency bands by filtering, weight, and reconstruct the signal by summing. Audio mixing boards do this, though

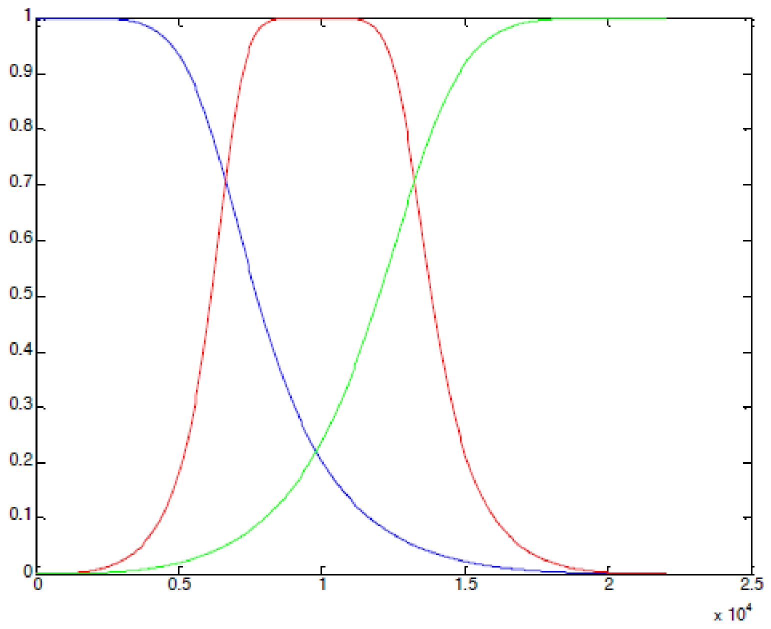
有什么意义 为什么这么做?

with many more than the three channels we have here. The term “equalizer” comes from the fact that people often want to boost the gain of low energy component sounds or reduce the gain of loud components.

Simple 3-band equalizer



For the case where we have 3 filters (3 difference equations), there is one LPF, one BPF, and one HPF. All of them cover the full frequency range, as shown below. If there are more channels, the filters become narrower and you need additional BPFs.



In this lab, you will build a 3-band equalizer by writing a function that takes as input gain terms (G1, G2, G3 and later G1-G5), applies the filters, multiplies the filter outputs by the gain terms, and sums the results. Since the way our ears hear changes in amplitude is more like a logarithmic scale than a linear scale, it is useful to refer to gain in terms of Decibels (dB). Overall dB gain for input and output sequences a filter is commonly defined as:

$$10\log_{10}\frac{\sum_n |y^2[n]|}{\sum_n |x^2[n]|} = 10\log_{10}|h^2[n]|$$

A gain of ½ corresponds to -6 dB in magnitude, a gain of 2 is an increase of 6dB in magnitude, and a gain of 1 is a change of 0 dB in magnitude. Normally, an equalizer can drop or boost the gain in dB from -20 to 20 dB in any one band. *What range of values does this require for the gain terms?*

3-band equalizer

1. Implement a 3-band equalizer using the filters specified below. *Identify the set of coefficients that corresponds to each different filter.*

Filter coefficients						**LP, HP, or BP?**
B1 =	0.0495	0.1486	0.1486	0.0495		
A1 =	1.0000	-1.1619	0.6959	-0.1378		
B2 =	0.1311	0	-0.2622	0	0.1311	
A2 =	1.0000	-0.4824	0.8101	-0.2269	0.2722	
B3 =	0.0985	-0.2956	0.2956	-0.0985		
A3 =	1.0000	0.5772	0.4218	0.0563		

2. Download the music.wav file from the class web site. The music file is standard CD quality, sampled at 44.1 KHz with a 16-bit word size for each channel. Test your equalizer by comparing the output when G1=G2=G3=1 to the original. The sound quality should still be good.

3. Consider different set of gains (for example G1=G2=0 and G3=1 or G1=1 and G2=G3=0) and discuss about the filtered sound in the different cases.