EE 341
Lab 5: Digital Filtering

In this lab, we will consider different types of digital filters and look at their characterization in time and frequency. This will give you some insight into how digital filters are implemented and designed and into the properties of different digital filter design algorithms.

1. Filter Implementation and Characterization

Digital filters are usually implemented using a difference equation

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

assuming 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

y=conv(x,h,'same')

given an impulse response, which you can find by using the filter command with a unit impulse function, e.g.

```
impulse = [1 zeros(1,49)];
h=filter(b,a,impulse);
```

will give you the impulse response up to length 50. The *'same'* flag will give you an output that is the same length as x, which will be comparable to what you get with the filter command. (Without that flag you will get the result with length corresponding to the sum of the lengths of x and h, which pads x with zeroes at the end.)

Both *filter* and *conv* assume zero values for the x signal before its start time.

Like continuous LTI systems, a digital filter can be characterized in the time domain and the frequency domain. The MATLAB function

freqz(b,a)

will give you log magnitude and unwrapped phase plots for the filter frequency response over positive normalized frequency [0,0.5]. (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.) Look at the *freqz* documentation to find out how you can get values to plot over radians or Hz.

In order to understand the behavior of different filters in different domains, we will experiment with a moving average filter and a first difference filter. Download the 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 30-day moving average filter:

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

Implement this filter in MATLAB using both the conv and filter functions to confirm that you get the same answer when applied on the stock data. Note that the big jump at the onset of the filtered signal is associated with zero-padding – start your plot at sample $n=30$ to avoid this artifact. Plot the original and filtered signal together, with the filtered signal plotted in red. Does the filtered result look smoother?

A simple filter for pulling out the high frequency trends is the first difference:

$y[n]=x[n]-x[n-1]$

Find the a and b coefficients for this filter, use them to filter the stock signal, and plot the result.

Plot the frequency response for both filters over the same time range and verify that the moving average system is a low-pass filter and the first difference system is a high-pass filter. Note the difference in scale of the two different filtered time signals.

IN-LAB CHECK-OFF: Show your TA your plot of the original and filtered stock data, the plot of the first-difference filtered signal, and the plot of the log magnitude frequency responses for each of the two systems. Be sure to label the axes of the plots.

2. Filter 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 frequency domain. 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 $0 < W < 1$, where $W=1$ corresponds to half the sampling frequency. So, a cut-off frequency of $0 < W < 1$ corresponds to $0 < \varpi < \pi$ in radians for the DTFT, and $0<f<0.5$ in normalized frequency. Thus, a cut-off frequency of $W=0.5$ corresponds to $\varpi_c = \pi/2$ in radians and $f_c=0.25$ in normalized.

i. **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 0.3π.

ii. **IIR (Infinite Impulse Response) Digital Filters**
Use the MATLAB function `butter` to create a low pass Butterworth filter of order 10 with cut-off frequency 0.3π.

Plot and compare the impulse and frequency responses for the two alternatives. Comment on the differences in the written report.

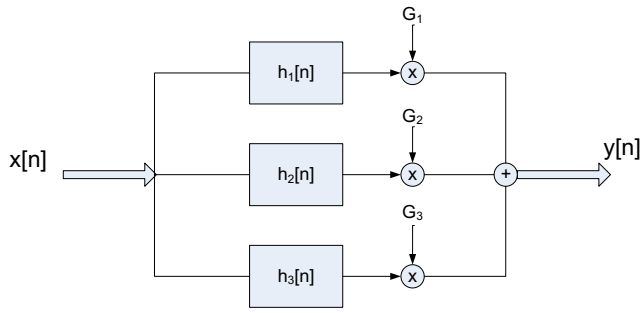
Use the MATLAB function `filter` to implement the two low pass filters you produced in 2a and 2b. Apply the filters to:

- the stock market data; and
- the music data on the class web site.

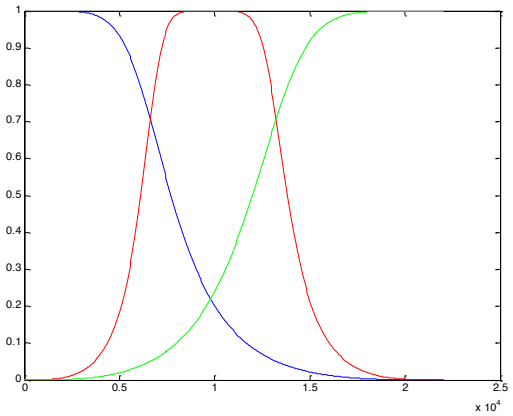
IN-LAB CHECK-OFF: Play the filtered music signals for your TA. How do the filtered signals sound compared to the original? Can you hear a difference between the results of the two different filters?

3. Music Equalizer

In this part, you will design a music equalizer, which breaks the sound file into multiple frequency bands by filtering, weighting, and summing to reconstruct the signal, as shown in the figure below for a simple 3-band equalizer. Audio mixing boards do this, though 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.



For the case where we have 3 filters (3 difference equations), there is one LPF, one BPF, and one HPF. Together, they cover the full frequency range, as shown below for a signal with a 44k sampling frequency. If there are more channels, the filters become narrower and you need additional BPFs.



In this lab, you will build a 4-band equalizer:

i. Design four 4th-order filters (an LPF, an HPF and two BPFs) with cut-offs at π/5, 2π/5, and 2π/3. (The high frequency bands are bigger since our perception of frequency is not on a linear scale.)

ii. Write a function that takes as input gain terms (G1, G2, G3, G4), applies the filters, multiplies the filter outputs by the gain terms, and sums the results.

Since the way our ears hear a change 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:

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

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?*

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=G4=1 to the original. The sound quality should still be good.

Using either the music.wav file or a 10-20s snippet of any music of your choice, create a modified version that boosts the frequencies in the vocal range (4-8kHz) by an audible amount. You may achieve this by boosting some bands and attenuating others.

IN-LAB CHECK-OFF: Play the original and modified sound for your TA. Let your TA try running your code with a different set of gains of his or her choice with the music.wav file. Which frequency band was boosted most?

Summary of Assignments and Deadlines

- In-lab group demonstrations (due in lab during the ninth week of class)
    - *Part 1:* Show your TA your plot of the original and filtered stock data, the plot of the first-difference filtered signal, and the plot of the log magnitude frequency responses for each of the two systems.
    - *Part 2:* Play the filtered music signals for your TA. How do the filtered signals sound compared to the original? Can you hear a difference between the results of the two different filters?
    - *Part 3:* Demonstrate your music equalizer.
- Written report
    - *Part 2:* Turn in your plots of the filter responses and time-domain plots of the filtered stock market signal 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 3:* Describe the filters you implemented for the music equalizer (give filter coefficients and include frequency response plots) and the mapping of human specified gains to the gain factor in the implementation. Describe the gains used to create your file with the boosted vocals.
- Files to be turned in via Canvas (due the day before your lab in the last week of class)
    - Written report
    - Electronic versions of the original and modified music from part 3
    - A zip file with all MATLAB code that you used in this lab