ENEE 425
Spring 2010 Assigned Homework
Oppenheim and Shafer (3rd ed.)
Instructor: S.A. Tretter

Note: The dates shown are when the problems were assigned. Homework will be discussed in class at the beginning of the next class after they were assigned unless stated differently.

Note: The numbers in parentheses are for the 2nd edition.

1. January 28: 4.2, 4.3, 4.4 (4.2, 4.3, 4.4)
2. February 2: Special Problem set on Hilbert Transforms
   (a) Hint: $\cos\omega_0 t \overset{\mathcal{F}}{\longleftrightarrow} \pi\delta(\omega-\omega_0)+\pi\delta(\omega+\omega_0)$
   $\sin\omega_0 t \overset{\mathcal{F}}{\longleftrightarrow} -j\pi\delta(\omega-\omega_0)+j\pi\delta(\omega+\omega_0)$
      i. Find the Hilbert transforms of $x(t)=\cos\omega_0 t$ and $y(t)=\sin\omega_0 t$
      ii. Find the analytic signals $x_+(t)$ and $y_+(t)$
      iii. Find $X_+(\omega)$ and $Y_+(\omega)$

   (b) Let $x(t)$ have the rectangular shaped low-pass Fourier transform
   $$X(\omega)=\begin{cases}1 & \text{for } |\omega|<W\\ 0 & \text{elsewhere}\end{cases}$$
      i. Find $x(t)$ and $\hat{x}(t)$
      ii. Find $x_+(t)$ and $X_+(\omega)$
3. February 18: 3.1 (a), (b); 3.3 (a), (b) (Same in 2nd ed.)
4. February 23: 3.5, 3.6 (a), (c), (d), (e). Just find inverse z-transforms using whatever method you prefer. Ignore the question about the existence of a Fourier transform. (Same in 2nd ed.)
5. February 25: 3.38, 3.57, 3.41 (3.33, 3.54, 3.37) (Do them in this order.)
6. March 2: Special Problem Set on Difference Equations

   Find the solutions to the following difference equations using one-sided z-transforms with the one-sided delay theorem.

   (a) $y(n)+3y(n-1)=x(n)$
   where $x(n)=2^{-n}u(n)$ and $y(-1)=1$

   (b) $y(n)-0.5y(n-1)=x(n)-0.5x(n-1)$
   where $x(n)=u(n)$ and $y(-1)=0$

   (c) $y(n)-0.5y(n-1)=x(n)-0.5x(n-1)$
   where $x(n)=u(n)$ and $y(-1)=1$

7. March 4: 3.23, 3.24 (3.21, 3.22)

8. March 9: Special Problem Set on Frequency Responses

   First make pole/zero plots for the following transfer functions. Then sketch their amplitude and phase responses using the geometrical approach based on vectors drawn from the poles and zeros to the unit circle. Finally, compute and plot the exact amplitude and phase responses using your favorite programming language or tool (mathcad, MATLAB, C, FORTRAN, etc.) The MATLAB functions freqz( ) and freqzplot( ) in the signal processing package are convenient for computing and plotting the theoretical amplitude and phase responses.

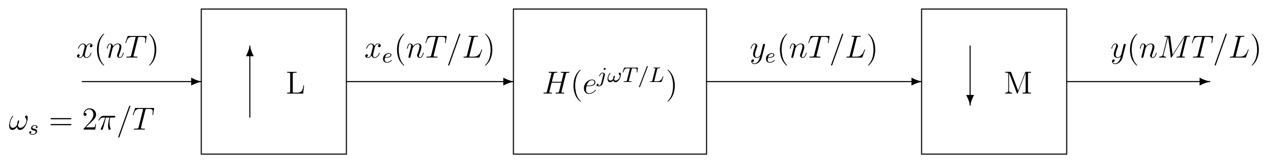
   (a) $\dfrac{1}{1+z^{-1}}$

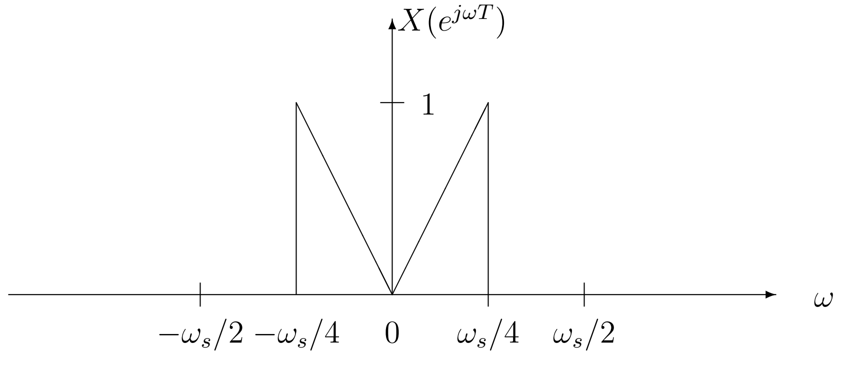
   (b) $\dfrac{z^{-2}}{(1-0.8z^{-1})(1-0.9z^{-1})}$

   (c) $\dfrac{(1+z^{-1})^2}{1+0.16z^{-2}}$

9. March 9 (Due April 6): Start Project 1. See page 4 of this document.

10. March 23: 4.32 (In our notation)
    Consider the discrete-time system shown in the figure below where





(i) L and M are positive integers.

(ii) $x_e(nT/L) = \begin{cases} x(nT/L) & n = kL,\ k \text{ is any integer} \\ 0 & \text{otherwise.} \end{cases}$

(iii) $y(nMT/L) = y_e(nMT/L)$

(iv) $H(e^{j\omega T/L}) = \begin{cases} M & |\omega| \leq \frac{L\omega_s}{8} \\ 0 & \frac{L\omega_s}{8} < |\omega| \leq \frac{L\omega_s}{2} \end{cases}$

(a) Assume $L = 2$ and $M = 4$, and that $X(e^{j\omega T})$, the DTFT of $x(nT)$, is real and is as shown in the figure above. Make an appropriately labeled sketch of $X_e(e^{j\omega T/L})$, $Y_e(e^{j\omega T/L})$, and $Y(e^{j\omega MT/L})$, the DTFTs of $x_e(nT/L)$, $y_e(nT/L)$, and $y(nMT/L)$, respectively. Be sure to clearly label salient amplitudes and frequencies.

(b) Now assume $L = 2$ and $M = 8$. Determine $y(nMT/L)$ in this case.
Hint: See which diagrams in your answer to part (a) change.

11. April 6: Do the digital filter design problems 2, 3, 4, and 5 on page 5 of this document.

12. April 8: You can begin working on the IIR portion (item 8) of Project 2 on page 7 of this document. This project must be turned in to receive a grade for the course. The due date will be set after we discuss FIR filter design.

13. April 13: You should aim to finish the IIR portion (item 8) of Project 2 this week.

14. April 15: Finish the IIR portion of Project 2 and begin the FIR portions.

15. April 20: Project 2 is due Thursday, April 29
April 20: Quantization and finite-word-length Problems 1, 2, 3, and 4 on page 9 of this document.

16. April 27: Quantization Problem 5

17. May 6: 8.5, 8.12, 8.14

Project 1. Problem on Filter Implementation and Amplitude Response Measurement

(This project must be turned in or else you will get a grade of "I" for this course.)

A 4th order IIR filter has a transfer function of the form $H(z) = KH_1(z)H_2(z)$ where $K = 0.0659434348$ and

$$H_1(z) = \frac{1 + 1.732154z^{-1} + z^{-2}}{1 - 0.620227z^{-1} + 0.195243z^{-2}}$$

$$H_2(z) = \frac{1 + 0.883717z^{-1} + z^{-2}}{1 - 0.444412z^{-1} + 0.678665z^{-2}}$$

Let the sampling rate be $f_s = 8000$ Hz.

1. Plot the theoretical amplitude response of the filter. You can use the MATLAB functions freqz( ) and freqzplot( ) in the signal processing package to compute and plot the response. The vertical axis should be the amplitude response in dB and the horizontal axis should be frequency in Hertz varying from 0 to $f_s/2 = 4000$ Hz.

2. Suppose the input $x_c[n] = \cos \omega nTu[n]$ has the output $y_c[n]$ and the input $x_s[n] = \sin \omega nTu[n]$ has the output $y_s[n]$. Then the input $x[n] = x_c[n] + jx_s[n] = e^{j\omega nT}u[n]$ has the output $y[n] = y_c[n] + jy_s[n]$. The amplitude of this output is $a[n] = (y_c^2[n] + y_s^2[n])^{1/2}$. After some time the filter output will reach a steady state and $a[n]$ will become a constant equal to the amplitude response on a linear scale since the magnitude of $x[n]$ is 1 for positive n. The amplitude response in dB is $\alpha = 20 \log_{10} a[n]$.

   (a) Write a MATLAB program to implement the filter $H(z)$ as a cascade of the scale factor K followed by $H_1(z)$ and then $H_2(z)$. Implement $H_1(z)$ and $H_2(z)$ as a second-order Type 1 direct form sections. Do not use MATLAB's built in filter implementation functions but program your own sums of products. Pretend you are writing this program to run in real-time in a DSP, perhaps, in your cell phone. For each input sample, you should run both filter sections and generate a single output sample. That is, operate the filter on a sample-by-sample basis. DO NOT generate an entire array of input samples, run the entire array through $H_1(z)$ to generate an intermediate output array, and then run the entire intermediate array through $H_2(z)$. Clearly, in your cell phone you cannot store all the samples of your input speech and wait until they have all been stored before doing the filtering.

   (b) Experimentally measure the amplitude response in dB of your filter by applying $x_c[n]$ and $x_s[n]$ for a set of frequencies spaced close enough together to make a smooth plot. The vertical axis should be amplitude response in dB and the horizontal axis frequency in Hertz over the range 0 to 4000 Hz. Start the filter state variables at zero for each test input and wait until n is large enough so the output has reached steady state. Then use the method above to compute the output amplitude at each frequency. Compare your experimental results with the theoretical ones.

Digital Filter Design Problems

1. A simple digital bandpass filter with zeros at $\omega = 0$ and $\omega_s/2$ and a peak near $\omega_s/4$ is required. The amplitude response must meet the specifications $|G^*(\omega_s/4)| = 1$ and $|G^*(\omega_s/8)| \leq 1/100$. Find a transfer function with a second degree numerator and denominator that meets these requirements. Draw the block diagram for a Type 1 Direct Form realization and write the difference equations required to implement it.

2. A digital filter approximation to a differentiator is desired. An ideal differentiator has the transfer function $G(s) = s$. An approach to designing the filter is shown in Figure 1. Let the guard filter be
$$A(s) = \frac{(1 - z^{-1})^2}{Ts^2}$$
where, as usual, $z = e^{sT}$. The filter $A(s)$ is a linear point connector with a one-sample delay to make it realizable. The desired response $G(s)$ is cascaded with the guard filter.

   (a) Find the transfer function of the digital filter approximation by computing $H(z) = Y(z)/X(z)$.

   (b) Write a difference equation for realizing $H(z)$.

   (c) Find the amplitude and phase responses of $H(z)$ and compare them with those of the ideal differentiator.

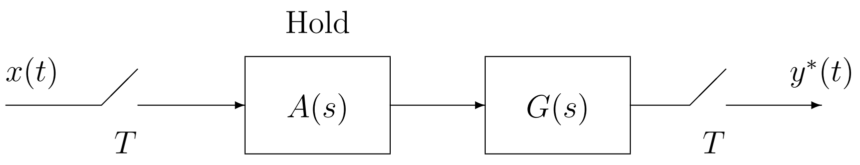


Figure 1: Using a Hold as a Guard Filter

3. Repeat the previous problem if the desired transfer function is $G(s) = 1/s$ which is an ideal integrator. Show that the resulting difference equation is equivalent to the classical trapezoidal integration rule.

4. (a) Find the transfer function of a lowpass digital filter with a 3 dB cutoff frequency of $\omega_s/6$ using the bilinear transformation and a third-order Butterworth analog prototype filter.

   (b) Sketch the amplitude response. Show the values at $\omega = 0$, $\omega_s/6$, and $\omega_s/2$.

   (c) Show how to realize it as a cascade of a 1st and 2nd order section and give the difference equations required for implementing it.

5. (a) Using a fourth-order Butterworth filter as the analog prototype filter and the bilinear transformation, design a digital lowpass filter with a 3 dB cutoff frequency of $\omega_s/6$.

(b) Sketch the amplitude response. Show the values at $\omega = 0$, $\omega_s/6$, and $\omega_s/2$.

(c) Show how to realize the digital filter as the cascade of two second-order Type 1 direct form sections and give the difference equations required to implement it.

Project 2

ENEE 425 Digital Filter Design and Implementation Project

- Get the data file `NoisySignal.wav` from http://www.ece.umd.edu/~tretter under the heading: **ENEE425 Digital Signal Processing**. This file contains samples of a desired lowpass band limited signal with a cutoff frequency of 1000 Hz plus unwanted noise consisting of a sum of sinusoids with frequencies above 1000 Hz.
- If you want to use UNIX and the SUN work stations, get the files `remez.bin`, `window.bin`, `iir.bin`.
- If you want to use PC's get `remez87.exe`, `window.exe`, and `iir.exe`.
- MATLAB also has FIR and IIR filter design functions. You can use them if you like.
- Instructions for using `remez87.exe`, `window.exe`, and `iir.exe` are on the class web site under the heading "Notes on Using the Digital Filter Design Programs."

1. Design a 101 tap lowpass FIR filter with a cutoff frequency of 1000 Hz using the program `window.exe` and a Hamming window to extract the desired signal. The attenuation above 1500 Hz should be at least 55 dB. Plot the amplitude response of your filter in dB. You can use the equivalent MATLAB filter design program if you wish.
2. Use the MATLAB function [x,fs,nbits] = wavread('NoisySignal.wav') to load the noisy signal file into the MATLAB array x. The variable fs is the sampling frequency for the wav file and nbits is the number of bits per sample. Use the MATLAB command h = load('your impulse response filename') to load your filter coefficients into the array h.
3. Look at the spectrum of the noisy signal by using the MATLAB command plot(20*log10(abs(fft(x)))). Include this plot in your report.
4. Pass the signal in the array x through your filter using time-domain discrete-time convolution. Let the filter output be the array y. MATLAB has a built-in convolution function called conv(h,x) but do not use it. Write out a sequence of instructions involving "for" loops to actually perform the convolution sum. In other words, make your own convolution function using the elementary operations of the sum of products of filter coefficients times data samples and "for" loops. This will give you a much greater understanding of the theory and practice than just using a pre-existing function. You can use the MATLAB convolution function to check your implementation.
5. Plot the spectrum of your filtered signal. Include it in your report.
6. Listen to the output of your filter by using the MATLAB command wavplay(y). You should hear a song connected to the University of Maryland. What is it?

7. Repeat the exercise using remez87.exe to design your FIR filter. The passband should extend from 0 to 1000 Hz. The stopband should extend from 1500 Hz to half the sampling rate.

8. Repeat the exercise again using an elliptic IIR lowpass filter. Implement your filter as a cascade of second-order sections. Pretend you are implementing the filter to run in real-time on a DSP. That is, run the cascade on a sample-by-sample basis. Use your "engineering judgement" to select the filter order and parameters.

9. This task is not required. If you would like to experiment some more, get the file RandNoisySignal.wav. It contains the same ideal music signal which is corrupted by additive highpass noise. The noise was generated by using the MATLAB function, rand(552719,1), to generate a vector of random numbers uniformly distributed between 0 and 1. This is called white noise. The noise was then highpass filtered with a 101 tap FIR filter generated by window.exe with a Kaiser window and a cutoff frequency of 1100 Hz. Use MATLAB and the fft() to plot the spectrum of the noisy signal. Use a lowpass filter to suppress the noise and plot the spectrum of the filtered signal. Listen to the filtered signal using wavplay().

Problems on Quantization and Finite Word-Length Arithmetic

1. Find the decimal equivalents of the following normalized two's complement numbers: (a) 01111 (b) 0011 (c) 1111 (d) 1011

2. Find the normalized two's complement representations for the following decimal numbers: (a) 3/4 (b) $-3/4$ (c) 7/8 (d) $-7/8$

3. A random variable X is rounded to quantization step size accuracy q. Suppose X is uniformly distributed over the interval $[-3q, 3q]$.

   (a) Find and sketch the probability density function (pdf) for the quantization error.

   (b) Find the signal to quantization noise power ratio in dB.

4. The input to an analog to digital converter is equally likely to fall anywhere between 10 and -10 volts. Find the number of converter bits required to achieve an 80 dB signal to quantization noise ratio.

5. A digital filter has the transfer function

$$G(z) = \frac{1}{(1 - 0.5z^{-1})(1 - 0.9z^{-1})} = \frac{1}{1 - 1.4z^{-1} + 0.45z^{-2}}$$

   The filter is implemented by the difference equation

$$y[n] = x[n] + 1.4y[n-1] - 0.45y[n-2]$$

   Let the input be $x[n] = 0$ for all time. The value of $y[n]$ is computed using full accuracy arithmetic for the products and sum on the right-hand side of the difference equation and the resulting sum of products is rounded to the nearest integer and stored to be used in the next iteration. That is, the past values of $y[\cdot]$ used on the right-hand side are the values that have been rounded to the nearest integer and stored.

   (a) If the initial conditions at time $n = 0$ are $y[-1] = 0$ and $y[-2] = 1$, calculate the future outputs until a steady-state pattern emerges.

   (b) Repeat (a) if $y[-1] = 0$ and $y[-2] = 10$.

   (c) Repeat (a) if $y[-1] = 10$ and $y[-2] = 10$.

   (d) Is there a problem with this filter? What would be the theoretical steady-state filter output for $x[n] = 0$? This problem illustrates a nonlinear effect called the dead band.