

The University of Texas at Austin
Dept. of Electrical and Computer Engineering
Midterm #1

Date: October 12, 2007

Course: EE 345S Evans

Name: _____ ***Set*** _____ ***Solution*** _____
Last, First

- The exam is scheduled to last 50 minutes.
- Open books and open notes. You may refer to your homework assignments and the homework solution sets.
- Calculators are allowed.
- You may use any standalone computer system, i.e. one that is not connected to a network.
Please disable all wireless connections on your computer system.
- Please turn off all cell phones, pagers, and personal digital assistants (PDAs).
- All work should be performed on the quiz itself. If more space is needed, then use the backs of the pages.
- **Fully justify your answers.**

<i>Problem</i>	<i>Point Value</i>	<i>Your score</i>	<i>Topic</i>
1	25		Digital Filter Analysis
2	30		Upconversion
3	25		Digital Filter Design
4	20		Potpourri
<i>Total</i>	100		

Problem 1.1 Digital Filter Analysis. 25 points.

A causal discrete-time linear time-invariant filter with input $x[k]$ and output $y[k]$ is governed by the following difference equation

$$y[k] = a^2 y[k-2] + (1-a) x[k]$$

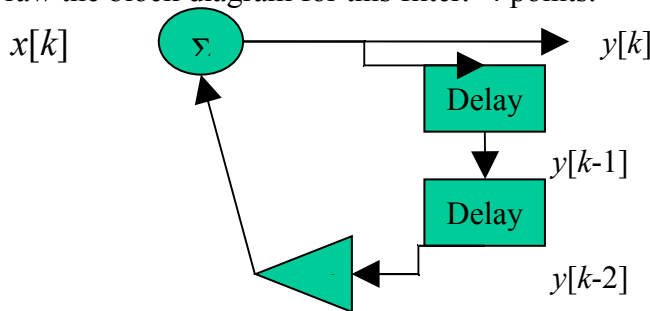
where a is a real-valued constant with $0 < a < 1$.

Note: The output is a combination of the current input and the output two samples ago.

- (a) Is this a finite impulse response filter or infinite impulse response filter? Why? 2 points.

The current output $y[k]$ depends on previous output $y[k-2]$. Hence, the filter is IIR.

- (b) Draw the block diagram for this filter. 4 points.



- (c) What are the initial conditions and what values should they be assigned? 4 points.

$y[0] = a^2 y[-2] + (1-a) x[0]$ Hence, the initial conditions are $y[-1]$ and $y[-2]$,
 $y[1] = a^2 y[-1] + (1-a) x[1]$ i.e., the initial values of the memory locations for
 $y[2] = a^2 y[0] + (1-a) x[2]$ $y[k-1]$ and $y[k-2]$. These initial conditions should
be set to zero for the filter to be linear & time-invariant

- (d) Find the equation for the transfer function in the z -domain including the region of convergence. 5 points.

Take the z -transform of both sides of the difference equation:

$$Y(z) = a^2 z^{-2} Y(z) + (1-a) X(z)$$

$$Y(z) - a^2 z^{-2} Y(z) = (1-a) X(z)$$

$$(1 - a^2 z^{-2}) Y(z) = (1-a) X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1-a}{1 - a^2 z^{-2}} = \frac{1-a}{(1 - az^{-1})(1 + az^{-1})} \quad \text{Hence, poles are located at } z = a \text{ and } z = -a.$$

Since the system is causal, the region of convergence is $|z| > a$.

- (e) Find the equation for the frequency response of the filter. 5 points.

System is stable because the two poles are located inside the unit circle since $0 < a < 1$.

Because the system is stable, we can convert the transfer function to a frequency response:

$$H_{freq}(\omega) = H(z) \Big|_{z=e^{j\omega}} = \frac{1-a}{1 - a^2 e^{-j2\omega}}$$

- (f) Is this filter lowpass, bandpass, bandstop, highpass, notch, or allpass? Why? What value of the parameter a would you use? 5 points.

Poles are at angles 0 rad/sample (low frequency) and π rad/sample (high frequency).

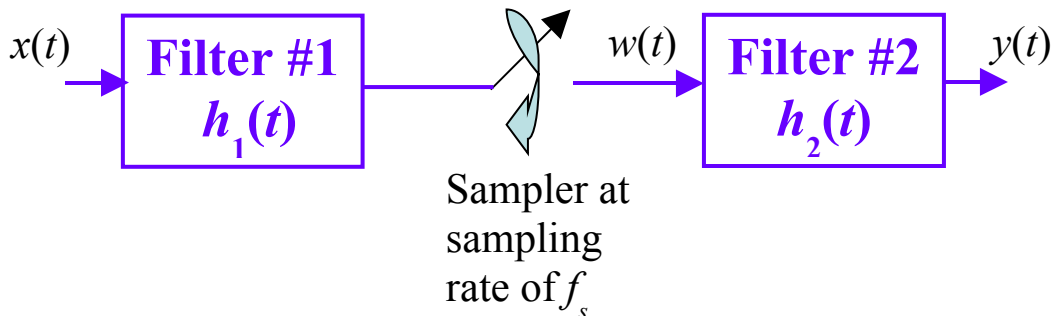
When $a \approx 0$, the filter is close to allpass.

When $a \approx 1$, the filter is bandstop. Poles close to the unit circle indicate the passband(s).

Problem 1.2 Upconversion. 30 points.

You're the owner of The Zone AM radio station (AM 1300 kHz), and you've just bought KLBJ (AM 590 kHz). As a temporary measure, you decide to broadcast the same content (speech/audio) over both stations.

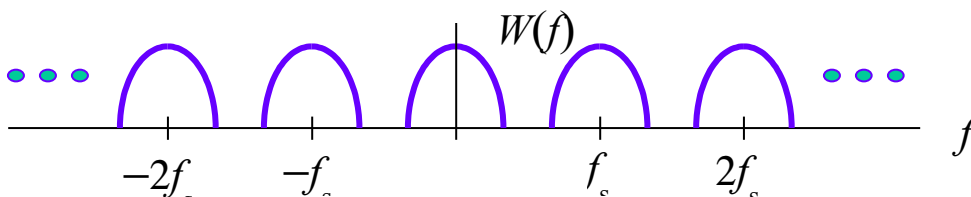
Carrier frequencies for AM radio stations are separated by 10 kHz. The speech/audio content is limited to a bandwidth of 5 kHz.



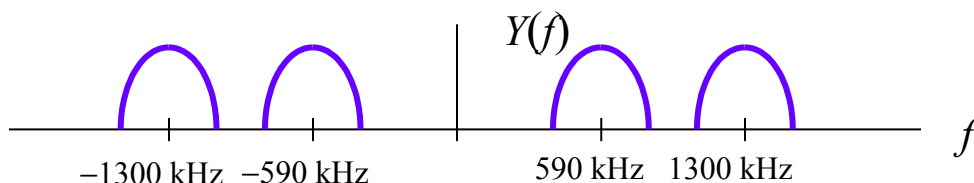
The output $y(t)$ should contain an AM radio signal at carrier frequency 590 kHz and an AM radio signal at carrier frequency 1300 kHz. The input $x(t) = 1 + k_a m(t)$, where $m(t)$ is the speech/audio signal to be broadcast. Since $m(t)$ could come from an audio CD, the bandwidth of $m(t)$ could be as high as 22 kHz. **Note: Filter #1 is anti-aliasing filter. Filter #2 is a two-passband bandpass filter.**

(a) Continuous-Time Analysis. 15 points.

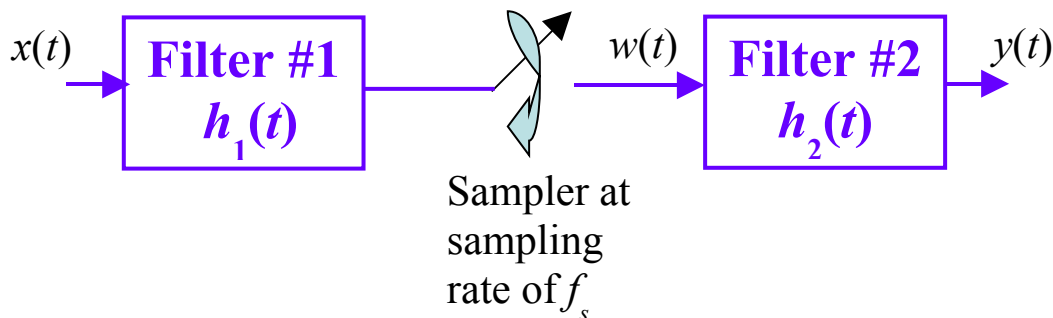
- 1) Specify a passband frequency, passband deviation, stopband frequency, and stopband attenuation for filter #1. **Speech/audio bandwidth for AM radio is limited to 5 kHz. Filter #1 enforces this requirement (see homework problems 2.3 and 3.3). Assuming the ideal passband response is 0 dB, $A_{\text{pass}} = -1$ dB and $A_{\text{stop}} = -90$ dB. The 90 dB of comes from the dynamic range of the audio CD. Also, $f_{\text{stop}} < 5$ kHz. We'll choose $f_{\text{pass}} = 4.3$ kHz and $f_{\text{stop}} = 4.8$ kHz. The transition region is roughly 10% of f_{pass} .**
- 2) Give the sampling rate f_s of the sampler. **We want to produce replicas of filter #1 output centered at 590 kHz and 1300 kHz. Also $f_s > 2 f_{\text{max}}$ and $f_{\text{max}} = 4.8$ kHz. So, $f_s = 10$ kHz.**
- 3) Draw the spectrum of $w(t)$. **Each lobe below is $2 f_{\text{max}}$ wide.**



- 4) Give the filter specifications to design filter #2. **Filter #2 passbands are 585.7–594.3 kHz and 1295.7–1304.3 kHz, and stopbands are 0–585 kHz, 595–1295 kHz, and greater than 1305 kHz. These bands have counterparts in negative frequencies.**
- 5) Draw the spectrum of $y(t)$. **Each lobe below is $2 f_{\text{max}}$ wide.**



The block diagram for the system is repeated here for convenience:



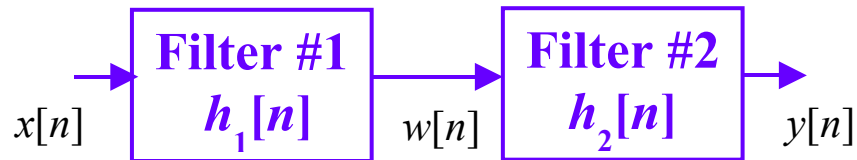
(b) Discrete-Time Implementation. 15 points.

- 1) Give a second sampling rate to convert the continuous-time system to a discrete-time system. **There are two conditions on the second sampling rate, as seen in homework problem 3.2. First, we'll need to pick a second sampling rate f_{s2} for $x(t)$, $w(t)$, and $y(t)$ that minimizes aliasing. The maximum frequencies of interest for $x(t)$ and $y(t)$ are 22 kHz and 1305 kHz, respectively. In theory, $w(t)$ is not bandlimited. Second, we'll need to pick the second sampling rate to be an integer multiple of f_s . In summary,**

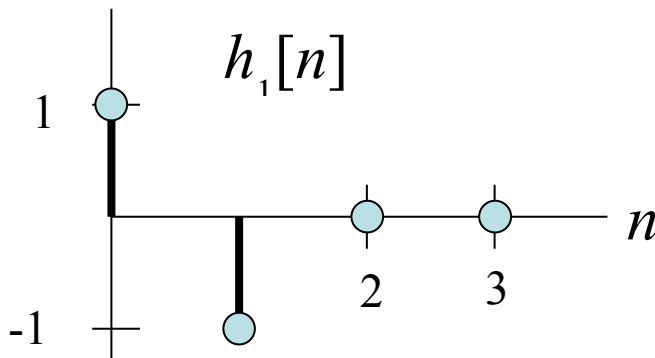
$$f_{s2} > 2 \text{ (1305 kHz)} \quad \text{and} \quad f_{s2} = k f_s, \text{ where } k \text{ is an integer}$$
- 2) Would you use a finite impulse response (FIR) or infinite impulse response (IIR) filter for filter #1? Why? **In audio, phase is important. AM radio stations generally broadcast single-channel audio. (AM stereo had gains in popularity in the 1990s, but has been in decline due to digital radio.) Assuming single-channel transmission, filter #1 should have linear phase. Hence, filter #1 should be FIR.**
- 3) What filter design method would you use to design filter #1? Why? **I would use the Parks-McClellan (Remez exchange) algorithm to design the shortest linear phase FIR filter.**
- 4) Would you use a finite impulse response (FIR) or infinite impulse response (IIR) filter for filter #2? Why? **As per part (2), filter #2 should have linear phase, and hence be FIR. Also, filter #2 is a multiband bandpass filter. It is not clear how to use classical IIR filter design methods to design such a filter.**
- 5) What filter design method would you use to design filter #2? Why? **Through homework assignments, we have designed multiband FIR filters using the Parks-McClellan (Remez exchange) algorithm. The Kaiser window method is for lowpass FIR filters. The FIR Least Squares method could be used. I would use the Parks-McClellan (Remez exchange) algorithm to design the shortest multiband linear phase FIR filter.**

Problem 1.3 Digital Filter Design. 25 points.

Consider the following cascade of two causal discrete-time linear time invariant (LTI) filters with impulse responses $h_1[n]$ and $h_2[n]$, respectively:



Filter #1 has the following impulse response:



The (group) delay through filter #1 is $\frac{1}{2}$ sample. **Note: Filter #1 is a first-order difference filter.**

Design filter #2 so that it satisfies all three of the following conditions:

- Cascade of filter #1 and filter #2 has a bandpass magnitude response,
- Cascade of filter #1 and filter #2 has (group) delay that is an integer number of samples, and
- Filter #2 has minimum computational complexity.

Group delay is defined as the negative of the derivative (with respect to frequency) of the phase response. As discussed in lecture, a linear phase FIR filter with N coefficients has a group delay of $(N-1)/2$ samples for all frequencies. So, a first-order difference filter has a delay of $\frac{1}{2}$ samples.

As we saw in the mandrill (baboon) image processing demonstration, a cascade of a highpass filter (first-order differencer) and a lowpass filter (averaging filter) has a bandpass response, provided that there is overlap in their passbands.

A two-tap averaging filter has a group delay of $\frac{1}{2}$ samples. A cascade of a first-order difference filter and a two-tap averaging filter would have a group delay of 1 sample.

A two-tap averaging filter with coefficients equal to one would only require 1 addition per output sample. No multiplications required. This is indeed low computational complexity.

$$h_2[n] = \delta[n] + \delta[n-1]$$

Problem 1.4. Potpourri. 20 points.

Please determine whether the following claims are true or false and support each answer with a brief justification. If you give a true or false answer without any justification, then you will be awarded zero points for that answer.

- (a) The automatic order estimator for the Parks-McClellan (a.k.a. Remez Exchange) algorithm always gives the shortest length FIR filters to meet a piecewise constant magnitude response specification. 4 points. **False, for two different reasons. First, the Parks-McClellan algorithm designs the shortest length linear phase FIR filters with floating-point coefficients to meet a piecewise constant magnitude response specification. Second, the automatic order estimator is an important but nonetheless empirical formula developed by Jim Kaiser. It can be off the mark by as much as 10%. Sometimes, the order returned by the automatic order estimator does not meet the filter specifications.**
- (b) All linear phase finite impulse response (FIR) filters have even symmetry in their coefficients. Assume that the FIR coefficients are real-valued. 4 points. **False. It is true that FIR filters that have even symmetry in their coefficients (about the mid-point) have linear phase. However, as mentioned in lecture, FIR filters with coefficients that have odd symmetry (about the mid-point) also have linear phase.**
- (c) If linear phase finite impulse response (FIR) floating-point filter coefficients were converted to signed 16-bit integers by multiplying by 32767 and rounding the results to the nearest integer, the resulting filter would still have linear phase. 4 points. **True. An FIR filter has linear phase if the coefficients have either odd symmetry or even symmetry about the mid-point. Multiplying the coefficients by a constant does not change the symmetry about the mid-point. In addition, $\text{round}(-x) = -\text{round}(x)$. Hence, rounding does not affect symmetry either.**
- (d) Floating-point programmable digital signal processors are only useful in prototyping systems to determine if a fixed-point version of the same system would be able to run in real time. 4 points. **False, due to the word “only”. It is true that floating-point programmable DSPs are useful in feasibility studies because committing the design time and resources to map a system into fixed-point arithmetic and data types. Beyond that, however, floating-point programmable DSPs are commonly used in low-volume products (e.g. sonar imaging systems and radar imaging systems) and in high-end audio products (e.g. pro-audio, car audio, and home entertainment systems).**
- (e) In the TMS320C6000 family of programmable digital signal processors, consider an equivalent fixed-point processor and floating-point processor, i.e. having the same clock speed, same on-chip memory sizes and types, etc. The fixed-point processor would have lower power consumption. 4 points. **This one could go either way. True. If the data types in a floating-point program were converted from 32-bit floats to 16-bit short integers, and the floating-point computations were converted to fixed-point computations, then the fixed-point processor would consume less power. The fixed-point processor would only need to load from on-chip memory half as often, and multiplication (addition) would take 2 cycles (1 cycle) instead of 4 cycles. Fixed-point multipliers and adders take far fewer gates than their floating-point counterparts, which saves on power consumption. False. If a floating-point program were run by emulating floating-point computations to the same level of precision on a fixed-point processor, then the fixed-point processor would actually consume more power.**