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

Date: March 9, 2018　　　　　　　　　　Course: EE 445S 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(s).***
- Please turn off all cell phones.
- No headphones allowed.
- 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.** If you decide to quote text from a source, please give the quote, page number and source citation**.**

Problem	*Point Value*	*Your score*	*Topic*
1	28		Filter Analysis
2	24		Mixers
3	24		Filter Design
4	24		Potpourri
Total	100		

Lectures 3 5 & 6

HW 0.4 1.1 2.1 3.1 & 3.3 | Lab #3

Problem 1.1 *Filter Analysis*. 28 points.

Consider the following causal linear time-invariant (LTI) discrete-time filter with input $x[n]$ and output $y[n]$ described by

$$y[n] = a_1\, y[n-1] + x[n] + b_1\, x[n-1] + b_2\, x[n-2]$$

for $n \geq 0$, where coefficients a_1, b_1 and b_2 are real-valued constants.

(a) What are the initial conditions and their values? Why? *3 points*.

Let $n = 0$: $y[0] = a_1\, y[-1] + x[0] + b_1\, x[-1] + b_2\, x[-2]$.

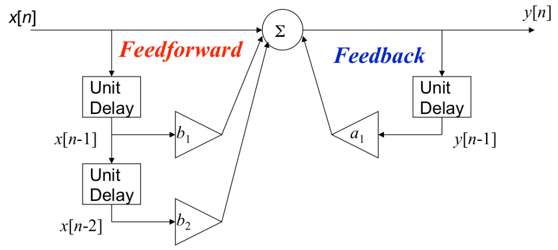
Let $n = 1$: $y[1] = a_1\, y[0] + x[1] + b_1\, x[0] + b_2\, x[-1]$.

From the above, the initial conditions are $y[-1]$, $x[-1]$ and $x[-2]$.

The initial conditions have to be zero to guarantee linearity and time-invariant properties.

(b) Draw the block diagram of the filter relating input $x[n]$ and output $y[n]$. *6 points*.

Any of the three IIR direct forms could be used here, e.g. from Lecture Slide 6-9



(c) Derive a formula for the transfer function in the z-domain. *4 points*.

Take the z-transform of both sides of the difference equation using the knowledge that all of the initial conditions are zero (Lecture Slide 6-8):

$$Y(z) = a_1 z^{-1}\, Y(z) + X(z) + b_1 z^{-1}\, X(z) + b_2 z^{-2}\, X(z)$$

$$(1 - a_1 z^{-1})\, Y(z) = (1 + b_1 z^{-1} + b_2 z^{-2})\, X(z)$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1}} \text{ for } |z| > |a_1|$$

(d) Give the conditions on a_1, b_1 and b_2 for the filter to be bounded-input bounded-output (BIBO) stable. *4 points*.

Pole must be inside the unit circle (Lecture Slides 6-12 & 6-13): $|\, a_1 \,| < 1$

A zero only plays a role in BIBO stability if it cancels a pole that leads to BIBO instability. If either zero equals a_1, the filter is in theory BIBO stable for any value of a_1 although an implementation of the original difference equation may not be BIBO stable.

(e) Give a formula for the discrete-time frequency response of the filter. *4 points*.

When the LTI system is BIBO stable according to part (d),

$$H_{freq}(\omega) = H(z)]_{z=e^{j\omega}} = \frac{1 + b_1 e^{-j\omega} + b_2 e^{-2j\omega}}{1 - a_1 e^{-j\omega}}$$

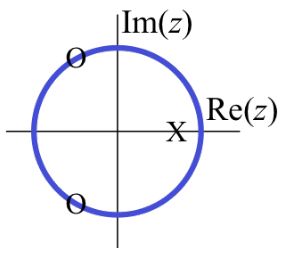
(f) Give numeric values for coefficients a_1, b_1 and b_2 to design a lowpass filter that also eliminates frequencies at $\omega = 2\pi/3$ and $\omega = -2\pi/3$ in the stopband. Draw the pole-zero diagram. *7 points*

Zeros are at $z_0 = e^{j2\pi/3}$ and $z_1 = e^{-j2\pi/3}$

$$(1 - z_0 z^{-1})(1 - z_1 z^{-1}) = 1 - (z_0 + z_1) z^{-1} + z_0 z_1 z^{-2}$$

So, $b_1 = -(z_0 + z_1) = -2\cos\left(\frac{2\pi}{3}\right) = 1$ **and** $b_2 = z_0 z_1 = 1$

Passband is centered at $\omega = 0$. Pole is at $a_1 = 0.9\, e^{j0} = 0.9$

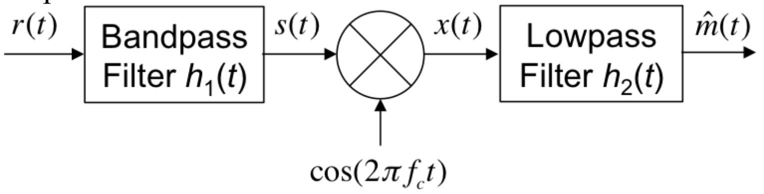


Trivial pole at $z = 0$ not shown

Lectures 1 4 5 & 6

HW 0.2 0.3 1.3 2.2 2.3 3.1 3.2 & 3.3 | Labs #2 and #3

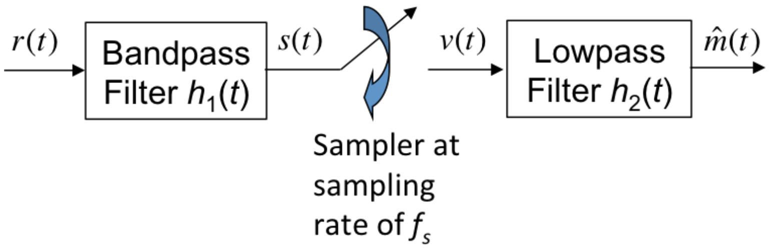
Problem 1.2 *Mixers.* 24 points.
Mixing provides an efficient implementation in analog continuous-time circuits for sinusoidal amplitude **demodulation** of the form



This is a sequel to Problem 1.2 on Fall 2017 Midterm #1.

We had discussed connections between downsampling and downconversion in class on Feb. 23rd when going over solutions for homework #2 and I had followed the discussion with an announcement on Canvas on Feb. 25th (see page 5 below).

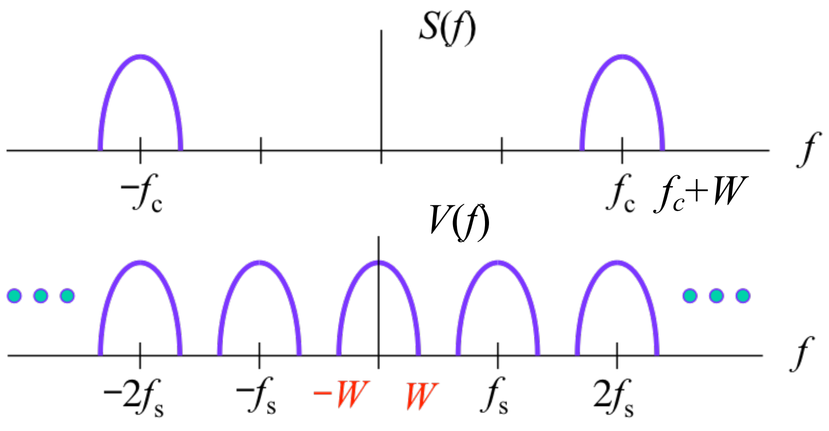
Here, $r(t)$ is the received bandpass signal of the form $r(t) = m(t)\cos(2\pi f_c t)$ where
$m(t)$ is the baseband message signal with bandwidth W, and
f_c is the carrier frequency such that $f_c > W$



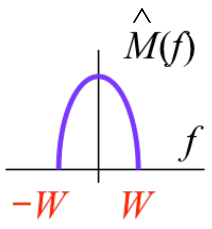
(a) Give the passband and stopband frequencies for the bandpass filter. *3 points.*
The bandpass filter passes the transmission band frequencies $[f_c - W, f_c + W]$ and attenuates out-of-band frequencies as much as possible to improve the signal-to-noise ratio. Passband frequencies are $f_c - W$ and $f_c + W$. Stopband frequencies are $f_c - 1.2\ W$ and $f_c + 1.2\ W$ to allow each transition bandwidth to be 10% of the passband bandwidth.

(b) Give the passband and stopband frequencies for the lowpass filter. *3 points*
$f_{pass} = W$ and $f_{stop} = 1.1\ W$ to allow transition bandwidth be 10% of the passband bandwidth.

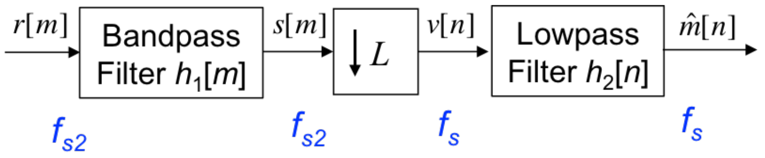
(c) Draw the spectrums for $s(t)$, $v(t)$ and $\hat{m}(t)$. You do not need to draw the spectrum for $r(t)$. *9 points.*
$s(t)$ is a bandpass signal. Sampling replicates $S(f)$ at offsets of integer multiples of sampling rate f_s. Each band in $V(f)$ and $S(f)$ is $2W$ wide. $\hat{m}(t)$ is the estimated baseband signal.



For bandpass sampling, $f_c = k\, f_s$ where k is an integer and $f_s > 2\ W$.



(d) In order to simulate the mixer in discrete-time, e.g. in MATLAB, we use discrete-time filters for the lowpass and highpass filters and replace the sampling block with a downsampling block.



i. Give the constraints on the sampling rate to convert the mixer to discrete time. *6 points.*

We have two sampling rates at play. Sampling rate f_s is used for the sampler in the continuous-time circuit for downconversion using mixing. Sampling rate f_{s2} is used to convert the overall system to discrete time.

(1) Nyquist-Shannon Sampling Theorem gives $f_{s2} > 2\,(f_c + W)$ and

(2) The downsampler by L will convert the sampling rate for the input signal of f_{s2} to a lower sampling rate on the output of f_s where $L = f_{s2} / f_s$. Hence, $f_{s2} = L\, f_s$.

Sampling rate f_{s2} is used for signals $r(t)$ and $s(t)$ and the bandpass filter. Sampling rate f_s is used for signals $v(t)$ and $\hat{m}(t)$ and the lowpass filter.

ii. Determine the downsampling factor. *3 points.* $L = f_{s2} / f_s$

```
% MATLAB code for midterm #1
% problem 1.2(d) for Spring 2018
%
% Programmer: Prof. Brian L. Evans
% The University of Texas at Austin
% Date: March 18, 2018
%
% Sinusoidal amplitude modulation
% r(t) = m(t) cos(2 pi fc t)
%                                ^
% r(t)     s(t)         v(t)     m(t)
% ---> BPF ---> Sampler ---> LPF --->
% fs2      fs2             fs       fs

% System Parameters
%   Continuous Time
W  =   300;    % Message bandwidth Hz
fs =  1000;    % Sampling rate in Hz
k  =    15;    % kth replica at fc
fc =  k*fs;    % Carrier frequency Hz
%   Discrete Time
L =    4*k;    % Downsampling factor
fs2 = L*fs;    % Sampling rate #2
%   fs2 > 2 (fc + W) and fs2 = L fs
%   With W < fc, fs2 = L fc, L >= 4

% Simulation Parameters
%   Bandpass signal amplitudes have
%   near-zero mean, and downsampling
%   by keeping every Lth sample gives
%   values close to zero.  Floating
%   point calculations lose accuracy
%   when values are below eps, where
%   eps is the largest positive value
%   for which 1.0 + eps = 1.0. In my
%   MATLAB, eps is 2.2204 x 10^(-16).
SimGain = 3.69*10^11;

% Time ref for m(t), r(t) & s(t)
Tmax = 10; Ts2 = 1/fs2;
t2 = Ts2 : Ts2 : Tmax;
N2 = length(t2);  n2 = 1 : N2;
```

```
% Baseband message w/ frequencies 0 to W
fpass = W;  npass = -100*L : 100*L;
hprot = sinc(2*(fpass/fs2)*npass);
hprot = hprot / sum(abs(hprot).^2);
noise = randn(1, N2);
m = filter(hprot, 1, noise);
% r[n2] = m[n2] cos(wc n2)
wc = 2*pi*fc/fs2;
r = m .* cos(wc*n2);

% DOWNCONVERSION BY DOWNSAMPLING
% Design bandpass filter with
% passband from fc - W to fc + W
% 1. Design lowpass prototype
% 2. Modulate by cos(2 pi fc t)
fpass = W;  npass = -10*L : 10*L;
hprot = sinc(2*(fpass/fs2)*npass);
wc = 2*pi*fc/fs2;
hbpf = hprot .* cos(wc*npass);
hbpf = hbpf / sum(abs(hbpf).^2);

% Design lowpass filter with passband
% frequency using truncated sinc pulse
fpass = W;  npass = -100 : 100;
hlpf = sinc(2*(fpass/fs)*npass);
hlpf = hlpf / sum(abs(hlpf).^2);

% Signal processing steps
r = SimGain * r;
s = filter(hbpf, 1, r);     % Bandpass
v = s(1:L:end);             % Downsample
mhat = filter(hlpf, 1, v);  % Lowpass

% Compare message signals
plotspec(m, Ts2);
title('Original message signal');
Ts = 1/fs; figure;
plotspec(mhat, Ts);
title('Estimated message signal');
PowerInOrgMessage = sum(abs(m.^2))
PowerInEstMessage = sum(abs(mhat.^2))
SimGain
```

The comments in the MATLAB code for the `Simulation Parameters` indicate that a very wide range of amplitude values is needed for downconversion using downsampling, e.g. 11 orders of magnitude in the above simulation. This is reflected in the `SimGain` parameter. The reason is that the downsampling operation samples many amplitude values that are very close to zero. This also means that the approach is very sensitive to additive noise and interference that are in the transmission band.

Beginning of the class Canvas announcement sent on Feb. 25, 2018:

On Friday in lecture, a student asked about the connection between downconversion and downsampling and the connection between upconversion and upsampling. Downsampling can be used as a method for downconversion, and upsampling can be used as a method for upconversion. Downsampling and upsampling change the sampling rate through discrete-time methods, and were introduced on homework #2.

On Friday, I worked out how to perform upconversion and downconversion in continuous time using sampling. The sinusodial [sinusoidal] amplitude modulation approach to upconversion involves a message signal being input into a lowpass filter followed by multiplication by cos(2 pi fc t) followed by bandpass filtering to produce a bandpass transmission. In the sampling approach, the multiplication by cos(2 pi fc t) is replaced by sampling at fs where fc = m fs where m is a positive integer and fs > 2 B where B is the baseband bandwidth. The sampling approach to upconversion uses aliasing to its advantage.

Similarly, the sinusodial [sinusoidal] amplitude demodulation approach for downconversion a receives a bandpass signal, then applies bandpass filtering, multiplication by cos(2 pi fc t) and lowpass filtering to give an estimate of the message signal. In the sampling approach, multiplication by cos(2 pi fc t) is replaced by sampling at fs where fc = m fs where m is a positive integer and fs > 2 B where B is the baseband bandwidth. The sampling approach to downconversion, which is also known as bandpass sampling, uses aliasing to its advantage.

Lecture slides 4-12 and 4-13 also describe the continuous-time analysis. Problem 1.4 on Midterm #1 in Fall 2013 and Problem 1.4 on Midterm #1 in Fall 2012 describe upconversion using sampling, and Problem 1.4(d) on Midterm #1 in Fall 2013 describes downconversion through sampling.

For a discrete-time approach, we'll use fs2 as the overall sampling rate. In this case, we will be oversampling by choosing fs2 > 2 fmax where fmax = fc + B where B is the bandwidth in Hz of the baseband message signal in continuous time.

For the upsampling approach to upconversion, the discrete-time baseband message signal would be upsampled and then bandpass filtered in discrete time to keep the replica that is centered at wc = 2 pi fc / fs2. One possible upsampling factor is L = fs2 / fc where L is an integer. Please see problem 1.2(d) on the Spring 2017 Midterm #1 Exam.

For the downsampling approach to downconversion, the received discrete-time bandpass signal centered at discrete-time frequency wc = 2 pi fc / fs2 would be downsampled and then lowpass filtered in discrete time to extract the discrete-time baseband signal. One possible downsampling factor is M = fs2 / fc where M is an integer. Please see Problem 1.3 on Midterm #1 in Spring 2012.

End of the class Canvas announcement sent on Feb. 25, 2018.

Please note that the solution for Problem 1.3(b) on Spring 2012 Midterm is related to Problem 1.2(d) above with the following specific settings:

$B = 2\,W$
$f_{s2} = M\,f_c$

Due to bandpass sampling in Problem 1.2(d) above, $f_c = k\,f_s$ where k is an integer, so

$f_{s2} = M\,f_c = M\,(k\,f_s) = (M\,k)\,f_s$
$L = M\,k$

Lectures 1 4 5 & 6	
HW 2.1 2.2 2.3 3.1 3.2 & 3.3	Lab #3

Problem 1.3 *Filter Design.* 24 points.

Every time that a particular tone at continuous-time frequency f_0 in Hz is detected, a particular audio effects system plays the tone at frequency f_0 and a tone at frequency $3\,f_0$.

Assume that 20 Hz $< f_0 <$ 5000 Hz and that the sampling rate is $f_s > 6\,f_0$.

The audio effects system will be running continuously. When frequency f_0 is not present, the audio effects system could generate very low volume sounds.

(a) Design a **second-order discrete-time** linear time-invariant (LTI) infinite impulse response (IIR) filter to detect frequency f_0 by giving formulas for the locations of the two poles and two zeros of the filter. Normalize the gain at continuous-time frequency f_0 to be 1. *9 points.*

Bandpass filter centered at $\omega_0 = 2\pi \frac{f_0}{f_s}$ where $\omega_0 \in (-\frac{\pi}{3}, \frac{\pi}{3})$ since $f_s > 6f_0$.

Poles are located at $p_0 = 0.9e^{j\omega_0}$ and $p_1 = 0.9e^{-j\omega_0}$ to ensure BIBO stability.

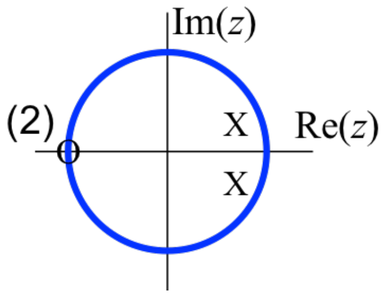
Place both zeros at z = -1 as in part (b) when the poles are close to z = 1 because $\omega_0 \approx 0$, or place the zeros at z = 1 and z = -1 for a more selective bandpass filter, or use the zeros for the bandpass resonator at z = 0 and $z = \cos(\omega_0)$ from homework 2.1(d).

The transfer function for a second-order IIR filter (biquad) from Lecture Slide 6-6 is

$$H(z) = C\frac{(1 - z_0 z^{-1})(1 - z_1 z^{-1})}{(1 - p_0 z^{-1})(1 - p_1 z^{-1})} \text{ for } |z| > 0.9$$

To normalize the gain at frequency ω_0, we substitute $z = e^{j\omega_0}$ into $H(z)$ and solve for C.

(b) Draw the pole-zero diagram for the poles and zeros given in part (a). *6 points.*



```
f0 = 440; fs = 8*f0;
z0 =  -1; z1 = -1;   % Try z0 = 1
numer = [1 -(z0+z1) z0*z1];
r = 0.90; w0 = 2*pi*f0/fs;
p0 = r*exp(j*w0); p1 = r*exp(-j*w0);
denom = [1 -(p0+p1) p0*p1];
%%% Normalize response at f0 to 1
z = exp(j*w0); zv = [1 z^(-1) z^(-2)];
C = (denom * zv') / (numer * zv');
figure; zplane(C*numer, denom);
figure; freqz(C*numer, denom);
```

(c) When the discrete-time IIR filter outputs a tone at continuous-time frequency f_0, what additional signal processing step(s) would you apply to the filter output to generate tones at continuous-time frequencies f_0 and $3\,f_0$? *9 points.* **Let the IIR filter output be $y[n]$ and $y[n] \approx \cos(\omega_0\, n)$.**

***Solution #1*: Modulate $y[n]$ by $\cos(2\omega_0 n)$ to produce discrete-time frequencies ω_0 and $3\omega_0$.**

***Solution #2*: Input $y[n]$ into a cubing block to give $y^3[n]$. That is, $y^3[n] = y[n]\,y[n]\,y[n]$. In the frequency domain, $Y(e^{j\omega})$ is a pair of Dirac deltas located at $-\omega_0$ and ω_0 with area π. $\frac{1}{2\pi}Y(e^{j\omega}) * Y(e^{j\omega})$ gives Dirac deltas at $-2\omega_0$, 0, and $2\omega_0$. $\frac{1}{2\pi}(\frac{1}{2\pi}Y(e^{j\omega}) * Y(e^{j\omega})) * Y(e^{j\omega})$ gives Dirac deltas at $-3\omega_0$, $-\omega_0$, ω_0, and $3\omega_0$. Or, $\cos^3(\omega_0 n) = \frac{3}{4}\cos(\omega_0 n) + \frac{1}{4}\cos(3\omega_0 n)$.**

***Solution #3*: Sinusoidal demodulation by $\cos(\omega_0\, n)$ and sinusoidal modulation by $\cos(3\,\omega_0\, n)$.**

	Lectures 3 & 6
HW 0.4 1.1 2.1 2.3 & 3.3	Labs #2 & #3

Problem 1.4. *Potpourri.* 24 points.

(a) Consider a discrete-time infinite impulse response (IIR) filter that is causal, linear time-invariant (LTI), and bounded-input bounded-output (BIBO) stable and that is defined in terms of its poles, zeros and gain. When implementing the filter in 32-bit IEEE floating-point arithmetic and data:

i. Describe how an implementation could cause the filter to become BIBO unstable. *6 points*.

Assume that the poles, zeros and gain are represented in 32-bit IEEE floating-point, which uses 24 bits of mantissa + sign and 8 bits for the exponent.

When expanding the factored form of the transfer function in the *z*-domain to an unfactored form using 32-bit IEEE floating-point arithmetic, one would compute

$$\boldsymbol{H(z) = C\frac{(1-z_0z^{-1})(1-z_1z^{-1})}{(1-p_0z^{-1})(1-p_1z^{-1})} = C\frac{1-(z_0+z_1)z^{-1}+z_0z_1z^{-2}}{1-(p_0+p_1)z^{-1}+p_0p_1z^{-2}}}$$

The feedback coefficient $-(p_0+p_1)$ would lose one bit of accuracy in the worst case due to a carry bit, and the feedback coefficient p_0p_1 would lose 24 bits of accuracy in the mantissa in the worst case. Using cascade of biquads helps reduce loss of accuracy.

Converting an *M*th-order IIR filter into a single section would cause a loss of 24(*M*-1) bits of accuracy in the worst case for the feedback coefficient in front of the z^{-M} term.

Due to the loss of accuracy in the feedback coefficients, factoring the denominator to find the new location of poles may reveal that some of the poles have moved onto or outside the unit circle. Please see Lecture Slides 6-20 and 6-23.

ii. Describe how an implementation could cause the loss of LTI properties. *6 points.*

By setting one or more of the initial conditions to a value other than zero.

iii. Give an example of a particular causal, LTI, BIBO stable discrete-time IIR filter for which its causal, LTI and BIBO stable properties are preserved when the filter is implemented in 32-bit IEEE floating-point arithmetic and data. *6 points.*

Let's consider a first-order, causal, LTI, BIBO stable, discrete-time IIR filter: $y[n] = 0.5\,y[n\text{-}1] + 0.5\,x[n]$. For LTI, y[-1] = 0.0. If $x[0] = 2.0$ and $x[n] = 1.0$ for $n > 1$, then $y[n] = 1.0$ for $n \geq 0$.

```
N = 1000;
x = ones(1, N);
x(1) = 2;
a = [1 -0.5];
b = [0.5];
y = filter(b, a, x);
sum(abs(y - 1.0))^2
```

(b) For a finite impulse response (FIR) filter with *N* coefficients, what is the increase in the number of multiplication-addition operations if the input signal, FIR coefficients and output signal were complex-valued instead of real-valued? *6 points.*

1 complex multiplication $(a + jb)\,(c + jd) = (ac - bd) + j\,(bc + ad)$ would take 4 real-valued multiplications and 2 real-valued additions.

1 complex addition $(a + jb) + (c + jd) = (a + c) + j\,(b + d)$ would take 2 real-valued additions.

1 complex multiplication-addition would take 4 real-valued multiplication-additions.

An FIR filter with *N* coefficients would take *N* multiplication-additions to compute one output value.

A complex-valued FIR filter would take 4 times as many real-valued multiplication-additions to compute one output value.