Digital Signal Processing (18-491/18-691)
Spring Semester, 2024

Problem Set 4

Issued: 2/15/24

Due: 2/22/24 at midnight via Gradescope, although submissions will be accepted until 2/24/24 without penalty. (The deadline is extended slightly because this assignment is a little longer than usual.)

Important note: Quiz 1 will be in class on Wednesday, February 28, during regular class hours. It will be based on material through this problem set. There will be a review session before the exam, most likely to be scheduled during the Monday evening preceding the exam or the Sunday afternoon or evening before then. Final time and place TBA. Most of the time will be spent answering questions and working problems .... come with questions or the review will be short and boring! The exam will cover through this problem set, or in other words the material we have discussed in OSYP Chapters 1-5, along with associated notes that have been passed out in class.

We list below the ground rules governing the quiz:

- The exam is open book with unlimited access to your class notes or closed-book with one sheet of notes and provided tables (depending on a vote of the class on Monday). You will not be allowed to search the internet for help in answering the questions. (The internet is unlikely to be helpful on this exam.)
- Similarly, you will not be allowed to use MATLAB to obtain your solutions, and it will not be helpful in any case. You may use the calculator on your telephone if you wish. Please keep your phones on airplane mode. Texting and phone calls during the exam are strictly prohibited.

Reading: We began our discussions of the past week or so with the computation of inverse z-transforms using the partial fraction method and linear constant-coefficient difference equations, and the development of the magnitude and phase of the DTFT from the locations of poles and zeros in the z-plane. We continued our discussion of the impact of pole/zero locations on frequency response, focusing on allpass systems, minimum and maximum-phase systems, zero-phase systems, and linear-phase systems. This material is discussed in OSYP Secs. 3.3-3.5 and 5.5-5.7, as well as the lecture notes on frequency response from pole/zero locations. We then went on to discuss continuous–time sampling and decimation and interpolation (aka downsampling and upsampling), following material in OSYP Secs. 4.0 through 4.6. We warmly recommend the discussion of sampling, decimation, and interpolation (Chapter 1) in the notes from 18-792 ADSP, which presents the same material in a form that more closely follows the sampling lecture of the past week.

During the coming week we will begin a discussion of the spectral representation of discrete-time periodic and finite-duration signals, leading to the discrete Fourier transform (DFT) and the fast Fourier transform (FFT).

Note: We will not be asking you to solve difference equations analytically or perform inverse z-transforms using contour integration in the complex z plane. We provide notes on these techniques for your reference only.

Problem 4.1: Listed below are three z-transforms:

1. $$X_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.5z^{-1}+0.125z^{-2}} \text{ for } |z| > \frac{\sqrt{2}}{4}$$

2. $$X_3(z) = \frac{1}{\left(1+\frac{1}{3}z^{-1}\right)^2} \text{ for } |z| > \frac{1}{3}$$

   (You can check your answer using the transform pair in OSYP Table 3.1, but to obtain full credit you must derive your answer using partial fractions, showing all work.)

For each of the z-transforms above, determine the inverse z-transform using partial fraction expansion, as well as the first three nonzero terms of the inverse z-transform using the "long-division" method. Convince yourself that the two methods produce equivalent answers.

(You can check your answer using the transform pair in OSYP Table 3.1, but to obtain full credit you must derive your answer using partial fractions, showing all work.)

Problem 4.2:

In Problem 1.4c and again in Problems 2.2 and 3.3 (!) we considered the convolution of the following two functions:

$$x[n] = (1/3)^{n-1}u[n-1] \text{ with } h[n] = (1/3)^{-n+1}u[-n+1]$$

As you know from the homework solutions, the result of this convolution is

$$y[n] = \begin{cases} \left(\frac{1}{8}\right)\left(\frac{1}{3}\right)^{-n}, & < 2 \\ \left(\frac{81}{8}\right)\left(\frac{1}{3}\right)^{n}, & n \geq 2 \end{cases}$$

(a) Write the z-transforms $X(z)$ and $H(z)$ for the time functions $x[n]$ and $h[n]$ along with the corresponding regions of convergence. (You already did this in Problem 3.3.)

(b) Is the system specified by $h[n]$ causal and stable? Relate your answer to the nature of the ROC.

(c) Obtain $y[n]$, the output of the system (*i.e.* the result of the convolution) using z-transform techniques with partial fraction expansion to compute the inverse z-transform. Compare your result to the answer you obtained for Problem 1.4(c), which you obtained via direct convolution. Which approach is easier?

Problem 4.3: Figure 4.3-1 below depicts the pole-zero plots for five different discrete-time LSI systems. While the axes are not labelled, the magnitudes of the pole and zero locations are all $1/4$, $1/2$, 0.9, 1, 1.1, and 2, while the phase angles of the pole and zero locations are 0, π, and $\pm\pi/4$.

(a) Figures 4.3-2 and 4.3-3 on the subsequent pages depict four possible transfer-function magnitudes (labelled A through D) and eight possible transfer-function phases (labeled E through L) that correspond to the LSI systems. The horizontal axis in all plots represents discrete-time frequency, plotted linearly. The vertical axis of the magnitude plots is also plotted linearly (i.e. not in decibels) with the minimum value equal to zero. The vertical axis of the phase plots is linear as well, running from $-\pi$ to π radians.

For each of the five pole-zero plots labelled System 1 through System 5, identify the corresponding magnitude and phase responses, again without using MATLAB. Please provide ample justification for your choices, as you will not receive any credit for this question without explaining your reasoning. **Note:** The magnitude plots can be multiplied by an arbitrary constant, which is why the vertical axes are not scaled.

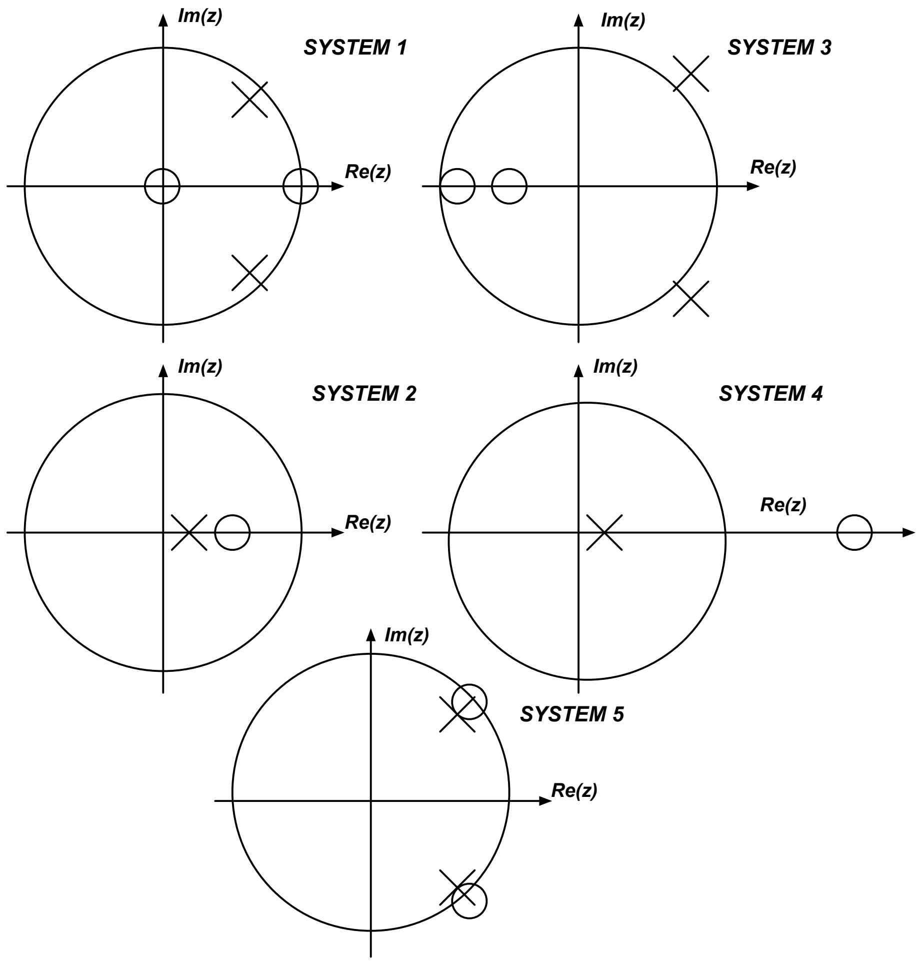


Figure 4.3-1. Pole-zero diagrams of LSI systems.

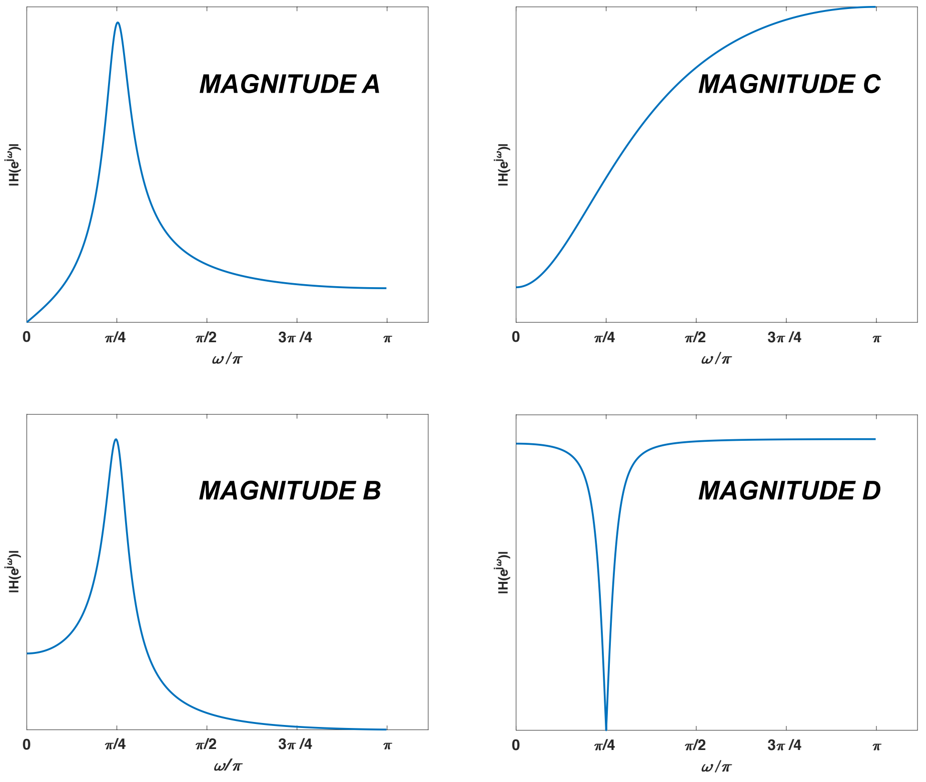


Figure 4.3-2. Potential magnitudes of frequency response of LSI systems.

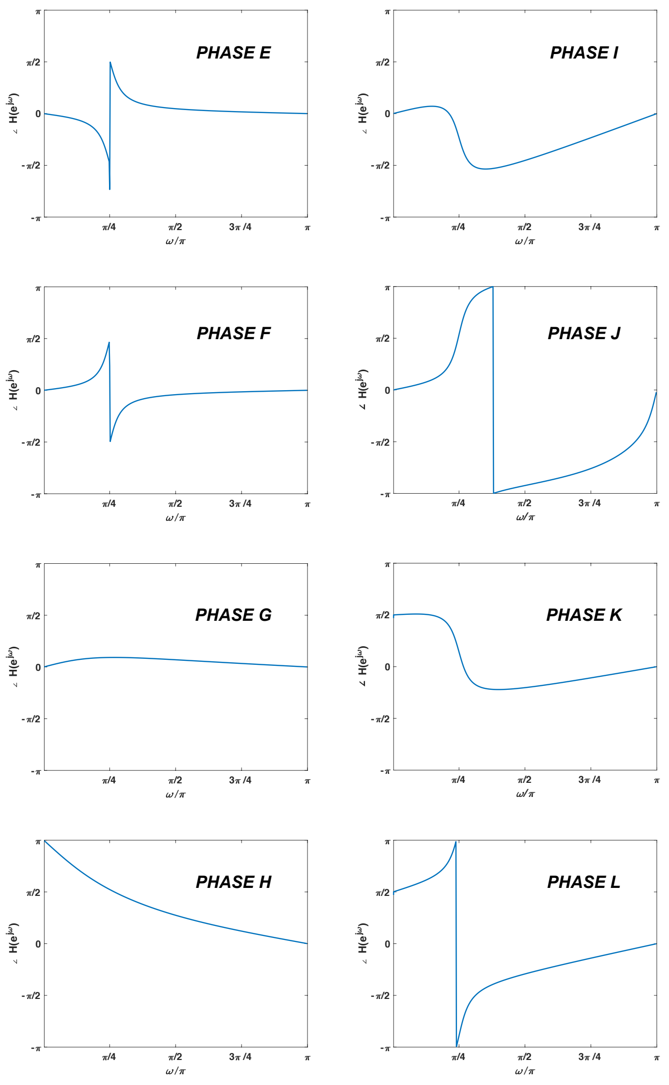


Figure 4.3-3. Potential phases of frequency response of LSI systems.

(b) Now consider the pole-zero patterns in Figure 4.3-1 once again.

1. For each of the five pole-zero patterns, sketch the pole-zero patterns of the corresponding **inverse system**. In each case, indicate whether the inverse system can be causal and stable. You must explain your answer to receive full credit.

2. For each of the five pole-zero patterns, sketch the pole-zero pattern of a second system that if cascaded with the original system would produce a composite system that is **allpass**. Indicate if the required second system does not exist, explaining your reasoning.

3. For each of the five pole-zero patterns, sketch the pole-zero pattern of a second system that if cascaded with the original system would produce a composite system that is **minimum**

phase, without changing the magnitude of the original frequency response. Indicate if the required second system does not exist, explaining your reasoning.

4. For each of the five pole-zero patterns, sketch the pole-zero pattern of a second system that if cascaded with the original system would produce a composite system that is **linear phase**, without changing the magnitude of the original frequency response. Indicate if the required second system does not exist, explaining your reasoning.

Problem 4.4:

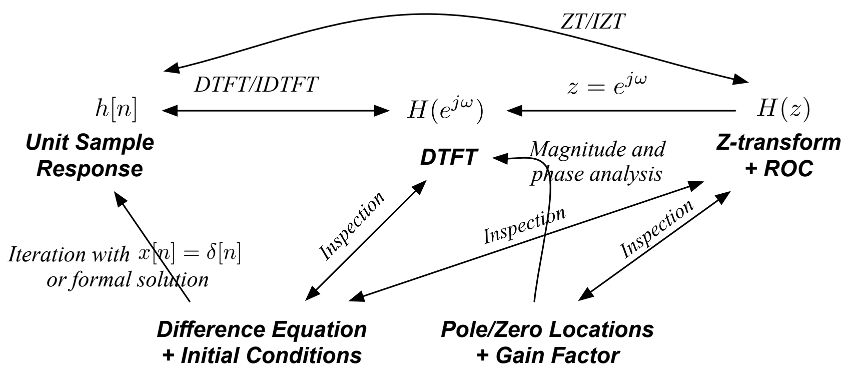

ZT/IZT
DTFT/IDTFT
$z = e^{j\omega}$
$h[n]$
$H(e^{j\omega})$
$H(z)$
Unit Sample Response
DTFT
Magnitude and phase analysis
Z-transform + ROC
Inspection
Inspection
Inspection
Iteration with $x[n] = \delta[n]$ or formal solution
Difference Equation + Initial Conditions
Pole/Zero Locations + Gain Factor

Figure 4.4-1. Summary of relationships between discrete-time representations.

Figure 4.4-1 above above summarizes the five complementary representations that we have learned to characterize discrete-time signals and systems. As you know, the unit sample response is used to characterize time-domain qualities, the DTFT is the basic frequency representation, the z-transform provides greater insight into stability, causality, and an easier approach to inverse transforms, the difference equations are used to actually implement digital filters, and the pole/zero locations provide valuable design insight.

This problem is a drill that addresses these relationships between the various representations. Feel free to use MATLAB to facilitate your solutions to the problem (but be sure to tell us how you use it), unless stated otherwise.

We know the following facts about a stable LSI system under consideration:

- The system has three poles located at $z = (-1 \pm j)/2$ and $z = \frac{1}{2}$

- The system has one zero located at $z = 1$

- $$H\left(e^{j\omega}\right)\Big|_{\omega=\pi} = 1$$

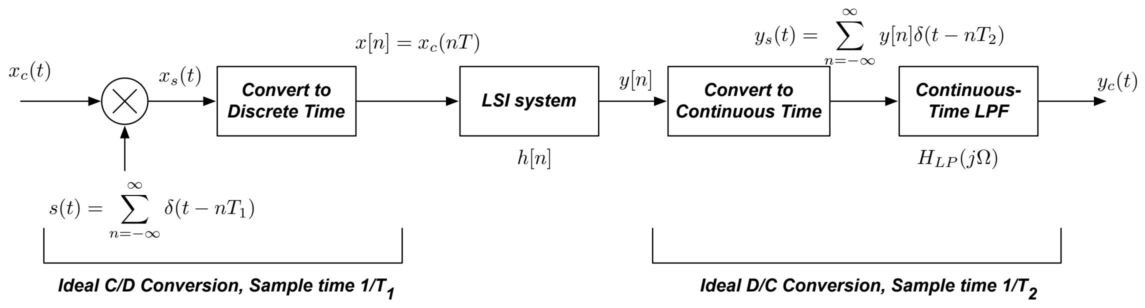
(a) Write the z-transform $H(z)$ that represents the transfer function of the system including all multiplicative constants.

(b) Write the difference equation characterizing the system, including the initial conditions that would be required if an input were first applied to the system at $n = 0$.

(c) Determine $h[n]$, the unit sample response of the system. (Again, feel free to use MATLAB liberally here.)

(d) Is the system causal? Is it linear phase? Why or why not?

Problem 4.5:



$x_c(t)$
$x_s(t)$
$x[n] = x_c(nT)$
Convert to Discrete Time
LSI system
$h[n]$
$y[n]$
$y_s(t) = \sum_{n=-\infty}^{\infty} y[n]\delta(t - nT_2)$
Convert to Continuous Time
Continuous-Time LPF
$H_{LP}(j\Omega)$
$y_c(t)$
$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_1)$
Ideal C/D Conversion, Sample time $1/T_1$
Ideal D/C Conversion, Sample time $1/T_2$

Figure 4.5-1. System for processing continuous-time signals in discrete time.

Figure 4.5-1 above is a block diagram of a generic system that converts a continuous-time signal $x_c(t)$ xinto its discrete-time representation $x[n]$ by uniform sampling, passes $h[n]$ through a discrete-time filter with sample response $h[n]$ and reconstructs the continuous-time signal $y_c(t)$ by lowpass filtering. Note that the sample times T_1 and T_2 associated with the sampling and reconstruction could be different.

Figure 4.5-2 on the next page summarizes the major steps of the sampling process. Note that the functions in the time domain are arbitrary, and that the triangular frequency representation is also arbitrary and clearly is not the shape of the actual CTFTs/DTFTs depicted. (Triangles are just easy to draw.) Also, the sampling period in the figure above is indicated by the generic T rather than T_1. Finally, note that the function $x_s(t)$ is a continuous-time function that consists of a sequence of delta functions, while the function $x[n]$ is a discrete-time sequence with the amplitudes of the discrete-time samples of $x[n]$ equal to the areas of the impulses in $x_c(t)$, and both of these are equal to the original sampled values of $x_c(t)$.

As we discussed in class, the sampling process is summarized by the equations:

$$\begin{aligned}
x_s(t) &= x_c(t)s(t) \\
x[n] &= x_c(nT_1) \\
x_s(t) &= \sum_{n=-\infty}^{\infty} x_c(nT_1)\delta(t - nT_1) \\
x[n] &= x_c(nT_1) = \sum_{l=-\infty}^{\infty} x_c[lT_1]\delta(n - l)
\end{aligned}$$

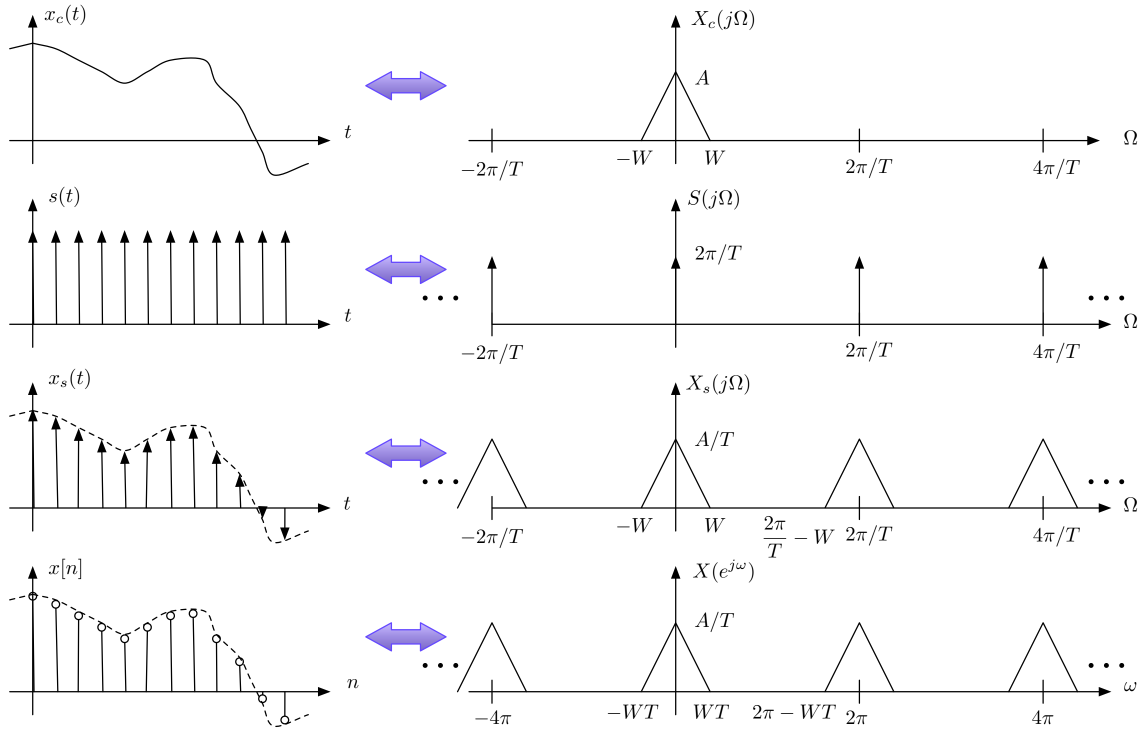


Figure 4.5-2. Time-domain and frequency-domain representations of stages of the sampling process.

and in the frequency domain,

$$
\begin{aligned}
S(j\Omega) &= \sum_{k=-\infty}^{\infty} \frac{2\pi}{T_1}\delta\left(\Omega - \frac{k2\pi}{T_1}\right) \\
X_s(j\Omega) &= \frac{1}{T_1}\sum_{k=-\infty}^{\infty} X\left(\left(\Omega - \frac{k2\pi}{T_1}\right)\right) \\
X(e^{j\omega}) &= \frac{1}{T_1}\sum_{k=-\infty}^{\infty} X\left(\left(\frac{\omega}{T_1} - \frac{k2\pi}{T_1}\right)\right)
\end{aligned}
$$

As you know, the maximum input frequency W must be less than half the sampling rate, π/T_1 to avoid aliasing distortion.

The reconstruction of the continuous-time output begins with the discrete-time filter output converted into a continuous-time train of delta functions $y_s(t)$ separated by T_2 seconds, and with areas equal to the amplitudes of the samples of $y[n]$.

The filter $H_{LP}(j\Omega)$ is typically an ideal lowpass filter with frequency response

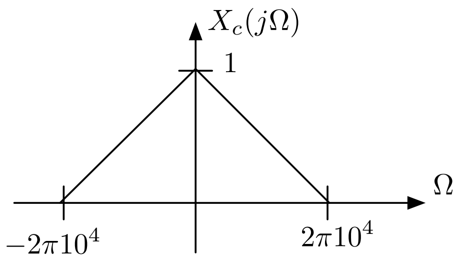
$$H(j\Omega) = \begin{cases} T_2, & |\Omega| < \frac{\pi}{T_2} \\ 0, & \text{otherwise} \end{cases}$$

Finally, assume that the discrete-time LSI system depicted above is also an ideal lowpass filter with frequency response

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| < 0.8\pi \\ 0, & \text{otherwise} \end{cases}$$

with $H(e^{j\omega})$ periodic with period 2π.

In working this problem, assume that the input signal $x_c(t)$ is bandlimited to 10 kHz, with the spectrum depicted below:



(a) Sketch and dimension in the frequency domain the functions $X_s(j\Omega)$, $X(e^{j\omega})$, $Y(e^{j\omega})$, and $Y_c(j\Omega)$ for each of the following values of T_1 and T_2:

1. $T_1 = T_2 = \dfrac{1}{2.5(10^4)}$

2. $T_1 = T_2 = \dfrac{1}{1.5(10^4)}$

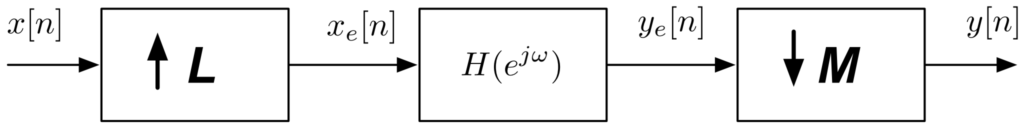
3. $T_1 = \dfrac{1}{2.5(10^4)}$ and $T_2 = \dfrac{1}{1.5(10^4)}$

(b) Now replace the input by the periodic time function $x_c(t) = \cos(2\pi 10000t)$ (*i.e.* a cosine with frequency 10 kHz). Assume for this part that the discrete-time filter is allpass with $H(e^{j\omega}) = 1$ for all ω.

Repeat part (a), subparts 1 and 2 (only) but with this cosine function as input. Also write out the time functions $y[n]$ and $y_c(t)$ that you obtain in each case.

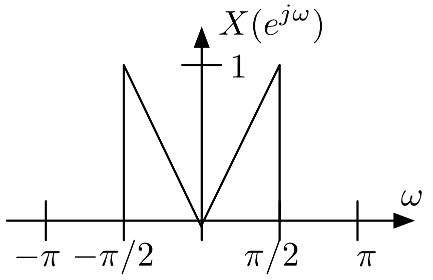
Note: It is important to remember that the frequency axis is scaled as signals are converted from $x_s(t)$ to $x[n]$ and back from $y[n]$ to $y_s(t)$. This in effect causes the delta functions to expand or contract in frequency, which affects their areas as discussed in Problem 4.5 (b) and (c). You need to be mindful of this in order to obtain the correct answers for this problem.

Problem 4.6:



Consider the multi-rate discrete-time system shown in the figure above. We know that:

- L and M are positive integers
- $x_e[n] = x[n/L]$ for $n = rL$ and zero otherwise
- $y[n] = y_e[nM]$
- $H(e^{j\omega}) = \begin{cases} L, & |\omega| \le \pi/4 \\ 0, & \pi/4 < |\omega| \le \pi \end{cases}$



(a) Assume that $L = 2$ and $M = 4$ and that $X(e^{j\omega})$, the DTFT of $x[n]$, is real and is as shown in the figure above. Sketch and dimension the functions $X_e(e^{j\omega})$, $Y_e(e^{j\omega})$, and $Y(e^{j\omega})$, the DTFTs of $x_e[n]$, $y_e[n]$, and $y[n]$, respectively. Be sure to label clearly all important magnitudes and frequencies.

(b) Now repeat part (a) except for the input $x[n] = \cos(0.2\pi n)$. Be mindful of the fact that expanding or contracting the horizontal axis of a function in time or frequency with continuous (Dirac) delta functions will cause the areas of those delta functions to increase or decrease in area correspondingly. (See the class notes on delta functions for more info on this.)

Matlab Problems

In working the MATLAB problems, turn in your results, a copy of the MATLAB code you developed to work the problem, as well as any additional comments you'd like to add to the MATLAB Gradescope submissions.

Reading: The MATLAB functions `roots`, `poly`, and `residuez` are especially useful in dealing with some of the issues addressed in this weeks problem set. Look over the `help` files associated with these functions. Look over the help files for more information on the other MATLAB commands cited in the problems below, as well.

For Python users: The recommended general local Python configuration is

```
- Python Version: v3.10.x
- Libraries & Recommended Versions
   - SciPy: 1.10.0
   - NumPy: 1.24.1
   - Matplotlib: 3.6.3
```

Some general recommendations for Python users include:

- Store signals as `numpy` arrays
- Read the online documentations of the functions we use to understand the differences between the MATLAB and Python implementations (although they are mostly like-for-like, nuanced differences may exist for a few of these functions)
- You would be able to find Python equivalents for most MATLAB functions by searching for `Python equivalent for <MATLAB_function_name>` online. `Stack overflow` often helps as well.

Problem C4.1: In `main_4_1.m`, Use the MATLAB function `residuez` to verify the partial fraction expansions you obtained in Problem 4.1.

For Python users: Suggested Python equivalent of `residuez`: `scipy.signal.residuez`

Problem C4.2:

In `main_4_2.m` use the MATLAB function `freqz` to verify the magnitudes and phases of the DTFTs that you obtained in Problem 4.3.

For Python users: Suggested Python equivalent of `freqz`: `scipy.signal.freqz`

Problem C4.3: In this problem you will design a simple lowpass filter from scratch based on pole and zero locations. While your filter will need to meet a set of typical specifications, do not expect it to be the best possible filter that could be designed. We will discuss filter design procedures extensively later in the course.

The specifications of your lowpass filter will be rather casual as these things go. You must design the filter so that its transfer function satisfies the following constraints:

1. $-10 \text{ dB} < \left|H(e^{j\omega})\right| < 10 \text{ dB for } |\omega| < 0.3\pi$

2. $\left|H(e^{j\omega})\right| < -50 \text{ dB for } 0.55\pi < |\omega| \leq \pi$

To constrain the design a bit, you are required to realize your filter with only four complex poles and four complex zeros. Keep in mind that the poles and zeros must occur in complex conjugate pairs for the unit sample response to remain real.

In `main_4_3.m`, do the following:

(a) Select a set of four pole and four zero locations by trial and error that enable you to satisfy the specifications above. This is probably most easily accomplished by guessing the pole and zero locations based on your knowledge and intuition about their impact on the frequency response, converting the pole and zero locations into the numerator and denominator polynomials of the transfer function using `poly`, and then using `freqz` to verify your selection. Please use the following MATLAB statements to generate normalized plots that have the same format (which facilitates grading):

```
[h,w] = freqz(b,a);
plot(w/pi,20*log10(sqrt(10)*abs(h)/max(abs(h))));
axis([0 1 -80 10])
```

where `b` and `a` are the numerator and denominator polynomials of the transfer function, respectively. When you are satisfied with your pole and zero locations, print the frequency response curve and turn it in with your written solutions. Be sure to indicate your choice of pole and zero locations.

(b) Write the transfer function of your filter as a z-transform. You may limit the coefficients to three significant figures.

(c) Obtain a closed-form analytical expression for the impulse response of your filter using any method with the aid of MATLAB. (The command `residuez` will be very useful.) Write the equation that specifies the unit sample response, limiting your coefficients to three significant figures.

(d) Obtain the linear constant-coefficient difference equation that characterizes your filter. Write this equation out, again using three significant figures for the coefficients.

(e) Read in the `PSO_B1short.wav` musical file that you used in Problem Set 2.[1] As in Problem C2.2, create a monophonic version of the audio file by averaging the left and right channel inputs. Using MATLAB, calculate the output of the filter when the input is the monophonic version of the audio. Use the the difference equation you developed in part (d) to obtain the output. Listen to the input and output using `soundsc` to hear the effects of the lowpass filtering.

(f) Now repeat part (e) but use the the MATLAB command `filter` to obtain the audio output. (See the `help` file for information on how to implement this.) Compare the resulting audio with your result from part (e).

Hints for Python users: Suggested Python equivalents of MATLAB functions:

[1]The music is the opening bars from a performance of Bach's first Brandenburg concerto that I was part of many years ago.

```
poly -> numpy.poly
freqz -> scipy.signal.freqz
residuez -> scipy.signal.residuez
filter -> scipy.signal.lfilter
audiowrite(output.wav,y/max(abs(y)),fs) -> scipy.io.wavfile.write('output.wav',x/numpy.abs(x
```

Code to generate normalized plots:

```
import numpy as np
from matplotlib import pyplot as plt
w,h = scipy.signal.freqz(b,a);
plt.xlim(0, 1)
plt.ylim(-80 10)
plt.plot(w/np.pi,20*np.log10(np.sqrt(10)*np.abs(h)/np.abs(h).max()))
plt.show()
```

What to turn in for your MATLAB Gradescope submission: Submit your program code for all three MATLAB questions, the frequency response curve, and the audio that is output from the filter, converting it to a `.wav` file using the command

```
audiowrite('output.wav',y/max(abs(y)),fs);
```

Include with your written solutions a printed program listing generated using the MATLAB `publish` command, a printed frequency response curve, plus your answers to the questions in parts (a) through (d).

Combine all these files, along with the (trivial) code from Questions C4.1 and C4.2 into a single .zip file archive. Upload the .zip archive to the Gradescope assignment "Problem Set 4 MATLAB."