EE102B Spring 2018-19 Pauly
Signal Processing and Linear Systems II

Homework 7

Due May 24

1. Determine the following z-transforms and regions of convergence using the transforms you know from the class notes, and the z-transform theorems.

   (a) $y[n] = 2\delta[n] + 3u[n-3]$

   (b) $y[n] = \left(\frac{1}{3}\right)^n u[n-2] + \left(\frac{9}{10}\right)^{(n-3)} u[n]$

   (c) $y[n] = \left(\frac{1}{2}\right)^n \cos\left(\frac{\pi n}{4}\right) u[n-1]$
   Hint You can delaying the cosine, but this is a little more difficult. Instead, consider the effect of deleting the first sample of the cosine.

   (d) $y[n] = u[n-2] * ((2/3)^n\, u[n])$

   (e) $y[n] = 2^n\, u[-n-2]$

2. Assume that the z-transform of $x[n]$ is
$$X(z) = \frac{1 + 2z^{-1}}{1 - \frac{1}{2}z^{-1}}.$$
   Use the z-transform theorems to find the z-transforms of the following signals

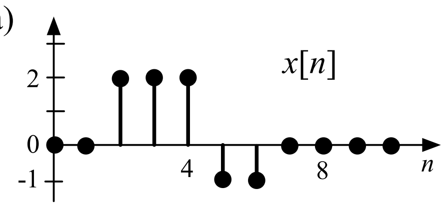
   (a) $y[n] = x[n] + x[n-3]$

   (b) $y[n] = (1+n)x[n]$

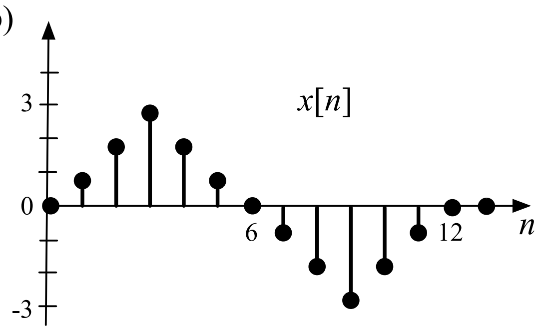
   (c) $y[n] = (\frac{1}{2})^n x[n]$

3. Use the first difference and running sum z-transform theorems to find the z transforms of these signals.

a)



b)



4. In lecture we stated the theorem for the z-transforms of time reversed signals.

   (a) Show that the z-transform of $y[n] = x[-n]$ is

   $$Y(z) = X(1/z).$$

   (b) If α is a pole of $X(z)$, show that $1/\alpha$ is a pole of $Y(z)$.

   (c) Plot α and $1/\alpha$, and the unit circle for $\alpha = 0.5e^{j\pi/4}$. Describe the relationship between the two with respect to the unit circle.

   (d) Assume $x[n]$ is a right-sided signal, and the unit circle is in the ROC for $X(z)$. Is the unit circle in the ROC for $Y(z)$?

Laboratory

This week we will be looking at plotting the poles and zeros of the transfer function of a discrete time system, and two different perspectives on plotting its frequency response.

Several of the matlab functions we will be developing in this lab duplicate the functionality of routines in the matlab signal processing toolbox. However, we do not assume that you have access to the signal processing toolbox. Also, it is often useful to write these routines yourself, in order to understand exactly what they do.

Task 1: Plotting the poles and zeros of a transfer function.

In order to plot the poles and zeros of a discrete-time transfer function, it should be written as a rational function in z. One example might be

$$H(z) = \frac{z^2 - z}{z^2 - 0.9z + 0.2}.$$

This function has two zeros and two poles. To find the zeros and poles we can use the matlab `roots()` command, after defining `b` polynomial coefficient vector for the numerator and `a` for the denominator

```
>> b = [1 -1 0];
>> a = [1 -0.9 0.2];
>> zs = roots(b)
zs = 0
     1
>> ps = roots(a)
ps = 0.4
     0.5
```

The transfer function may then be written as

$$H(z) = \frac{z(z-1)}{(z-0.4)(z-0.5)}$$

Often the transfer function will be written in terms of increasing powers of z^{-1}. The previous example would be

$$H(z) = \frac{1 - z^{-1}}{1 - 0.9z^{-1} + 0.2z^{-2}}$$

This transfer function appears to have one zero and two poles. However, if we multiply both the numerator and denominator by z^2 we get the previous result.

From the matlab perspective, in either case the numerator and denominator are represented by coefficient vectors. If we simply require that both be of the same length by padding with zero coefficients if necessary, both representations have the same coefficient vectors. For example, in this case we add one zero to the `b` vector

```
>> b = [1 -1 0];
>> a = [1 -0.9 0.2];
```

Check that these both give the same result if the transfer function is written in increasing powers of z^{-1}, or decreasing powers of z.

If we are starting with a transfer function in z^{-1}, by adding zero coefficients so that the numerator and denominator have the same order, we have effectively rewritten the transfer function in terms of z.

Use this to write a matlab m-file which plots the poles and zeros of a transfer function

$$H(z) = \frac{\sum_{i=0}^{l} b_i z^{-i}}{\sum_{i=0}^{k} a_k z^{-k}}$$

Assume that the b_i coefficients are stored in a matlab vector `b`, and the a_i coefficients are stored in a matlab vector `a`.

The m-file should be invoked with the command

```
>> pzplot(b,a);
```

This m-file should

- Append zeros to the either the numerator and denominator coefficient vector to make them the same length,
- Find the poles and zeros using matlabs `roots()` function

  ```
  >> ps = roots(a);
  >> zs = roots(b);
  ```

  and,
- Plot the result.

For the plotting routine use the following m-file, which is available on the eeclass web site

```
function draw_pz_plot(zs,ps)
%
%  function draw_ps_plot(zs,ps)
%
%  Draw a pole zero plot, where
%
%    zs -- vector of zeros
%    ps -- vector of poles
%

% convert zs and ps to row vectors

zs = zs(:).';
ps = ps(:).';

% find the scale factor

mx = max(abs([zs ps 1.0]));
mx = mx*1.125;

% plot axes
```

```
plot([0 0], [-mx mx],'-')
text(0.05*mx,mx,'Im');
hold on
plot([-mx mx], [0 0],'-');
text(mx,-0.05*mx,'Re');

% plot unit circle

f = [-0.5:0.01:0.5];
plot(cos(2*pi*f),sin(2*pi*f),':');

% plot poles and zeros

plot(real(zs),imag(zs),'o');
plot(real(ps),imag(ps),'x');

% make the plot square

axis(1.25*[-mx mx -mx mx]);
axis('square');
hold off
```

Test your routine for this $H(z)$

$$H(z) = \frac{2 - z^{-1}}{1 - \frac{\sqrt{2}}{2}z^{-1} + \frac{1}{2}z^{-2}},$$

and submit the plot. The result should look like Fig. 1.

Task 2: Evaluating the Frequency Response of a Transfer Function

The frequency response of a discrete time system is the transfer function evaluated along the unit circle,

$$H(e^{j\Omega}) = H(z)|_{z=e^{j\Omega}}$$

In this task we will take a transfer function and plot it's magnitude and phase as evaluated along the unit circle.

We will assume that the transfer function is provided as the coefficient vectors `b` and `a`, which are the coefficients of the numerator and denominator polynomials written in increasing powers of z^{-1}. To evaluate the frequency response, we first append zeros to to `a` or `b`, as above. These are then the same coefficient vectors as when the transfer function is written in decreasing powers of z. We then evaluate the transfer function using `polyval()`.

We want to evaluate the transfer function along the unit circle. We define a normalized frequency variable, and evaluate z along the unit circle

```
>> Omega = 2*pi*[0:255]/512;
>> z = exp(i*Omega);
```

The frequency response of the transfer function can then be evaluated as

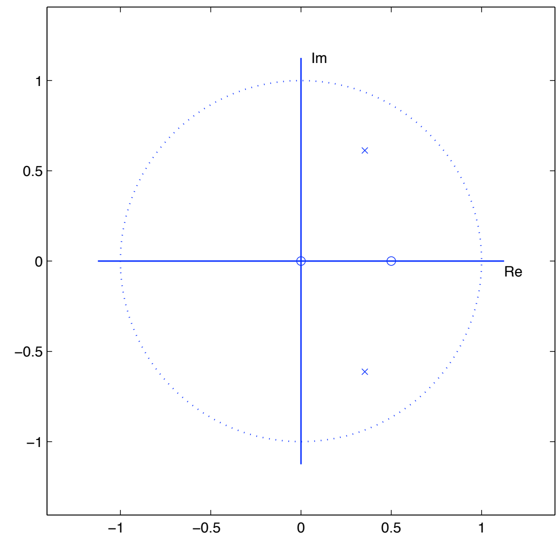


Figure 1: Pole/zero plot for the example in Task 1.

```
>> hf = gain* polyval(b,z)./polyval(a,z);
```

where gain is a constant multiplier for the transfer function, which is 1 here.

Write a matlab m-file that takes the coefficient vectors `b` and `a` corresponding to the transfer function being written in increasing powers of z^{-1}, and

- Appends zeros to make the orders of `a` and `b` to be the same,
- Calculates the frequency response using polyval,
- Plots the magnitude (`abs()`, not dB) and phase response in degrees using a 2x1 subplots, with the axes properly labeled.

The m-file should be invoked with the command

```
>> fr_plot(b,a, gain)
```

In this case `gain = 1`. Test your m-file on the transfer function from Task 1. The result should look like Fig. 2

Task3: Surface Plot of the Frequency Response

It can be difficult to visualize the frequency response of a transfer function given the locations of poles and zeros. One way to help with this visualization is to plot the magnitude of the transfer function over the entire unit disk (the interior of the unit circle). The poles of the transfer function appear as peaks, and the zeros as valleys.

We evaluate the transfer function over the entire complex plane in the same way as we evaluated it along the unit circle in the previous task, by using `polyval()`. However, instead of evaluating at a vector of points along the unit circle, we instead evaluate over a 2D matrix, which contains the values of z over the part of the complex plane we are interested in. If we want to be sure to include the unit disk, we can define the `z2` matrix to be

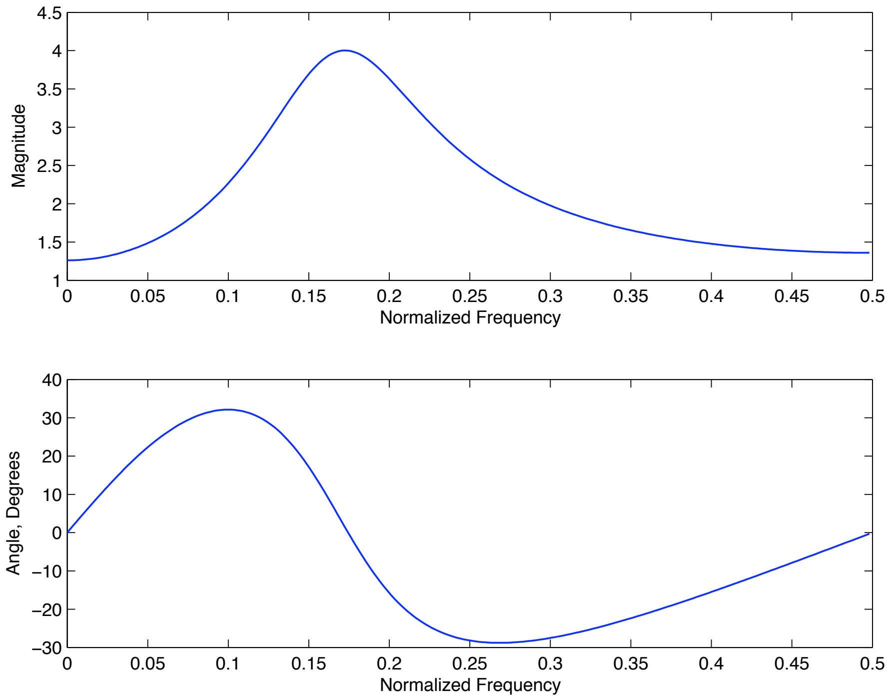


Figure 2: Frequency Response plot for the transfer function in Task 1. Here the frequency axis is $\Omega/2\pi$ so that the plot divisions work out evenly, but you can plot vs Ω if you prefer.

```
>> z1 =1.25 * [-32:31]/32;
>> onz = ones(1,64);
>> z2 = z1'*onz + i * onz'*z1;
```

The matrix `z2` has a real part that increases linearly along the columns of the matrix, and an imaginary part that increases linearly along the rows. The factor of 1.25 simply ensures that the unit disk is well within the matrix.

We can evaluate the transfer function as before,

```
>> hz = gain*polyval(b,z2)./polyval(a,z2);
```

where again we have assumed that `b` and `a` have been zero padded to be the same length. In order to clip this function to the unit disk, we need a matrix that is unity inside the unit circle, and 0 outside the unit circle. We can do this with

```
>> circ = abs(z2)< 1;
```

To display the surface, we use the surf command. We first compute the magnitude, and then limit this to a user specified range. Then we display the surface plot, clipped to the unit disk:

```
>> %
>> % compute magnitude
>> %
>> hzm = abs(hz);
```

```
>> %
>> % limit to  mx, which should be user specified
>> %
>> mx = 5;
>> hzm = min(hzm,mx);
>> %
>> % plot the surface
>> %
>> surf(z1,z1,hzm.*circ)
>> %
>> % set the view angle to something appropriate
>> %
>> axis ij
>> view(20, 20);
```

After the surface is plotted with `surf()`, the view angle is set with `view()` where the arguments are the azimuth and elevation angles in degrees. You can also set these interactively by clicking the rotate button on the figure, and dragging the mouse. Click the "rotate3D" button in the Figure window.

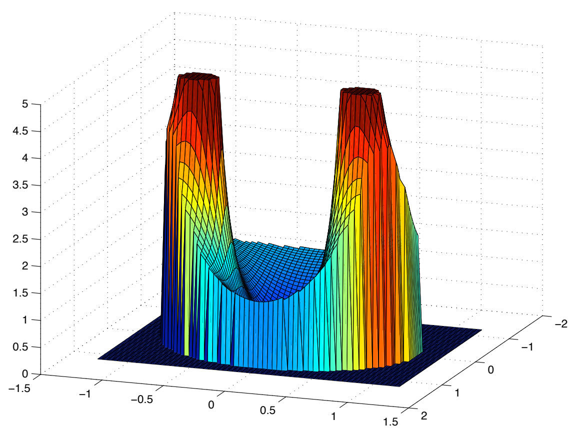


Figure 3: Surface plot for the transfer function in Task 1.

Write an m-file that performs these operations. The m-file should be invoked with the command

```
>> fr_surf(b, a, mx, gain)
```

Again, `gain = 1` here, but will be more useful for Task 4 below. Your m-file should assume that `b` and `a` are the coefficients of polynomials in increasing powers of z^{-1}, and may need to be zero padded.

Test your routine on the transfer function of Task 1. The result should look like Fig. 3. The shape of the function along the unit circle is clearly recognizable as that plotted in Task 2. Note that the frequency response is large near the poles, as expected.

Task 4: Example Filter Functions

Consider the two IIR discrete-time filters

$$H_1(z) = 0.1026\frac{1 + 2z^{-1} + z^{-2}}{1 - 0.9865z^{-1} + 0.4468z^{-2}}$$

and

$$H_2(z) = 0.0619\frac{1 + 2z^{-1} + z^{-2}}{1 - 1.2654z^{-1} + 0.6153z^{-2}}$$

Both are Chebychev lowpass filters with a normalized cutoff frequency of 0.125. The first has a passband ripple of 0.5 dB, and the second 3 dB. We would expect that the second filter would have sharper cutoff, and that this is due to the poles being closer to the unit circle.

For each of these filters, plot the poles and zeros, the frequency response, and the surface plot. Do the poles change as you would expect? Clip the maximum to about 2.5 for the surface plot. Make sure to specify the gain for `fr_plot()` and `fr_surf()`. The gain is the leading constant. Since these are lowpass filters, we expect the magnitude of the frequency response to be about 1, with some ripple. Make sure this is what you get.