5

z Transform

5 5-1

5.1 Purpose 5-2

5.2 Background 5-2

5.3 Preparation 5-2

5.3.1 ZPLOT: The relation between *H(z)* and the pole-zero plot 5-2

5.3.2 FPLOT: The relation between *H(z)* and the frequency response, *H(w)* 5-4

5.4 Assignment 5-6

5.1 Purpose

The purpose of this lab is to help you understand the relation between the *z*-transform $H(z)$, the pole-zero plot, and the frequency response $H(\omega)$.

5.2 Background

To prepare for this exercise, review Chapters 4 and 5.

5.3 Preparation

We will develop two Matlab functions:

- **ZPLOT**: a function to plot the pole-zero plot, given $H(z)$
- **FPLOT**: a function to plot the frequency transform (magnitude and phase) of $H(\omega)$, given $H(z)$;

5.3.1 ZPLOT: The relation between *H(z)* and the pole-zero plot

We will assume that $H(z)$ has been given in the form of the ratio of two polynomials in z^{-1},

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{N-1} b_k z^{-k}}{\sum_{k=0}^{N-1} a_k z^{-k}} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{N-1} z^{-(N-1)}}{a_0 + a_1 z^{-1} + \cdots + a_{M-1} z^{-(M-1)}}.$$

You will determine the poles and zeros from the vectors, a, and b. Your first task is to write a Matlab function, `zplot`, that displays the pole-zero plot given vectors a and b:

```
function zplot(b, a)
% ZPLOT Plot a zero-pole plot.
%                                  -1                 -nb
%              B(z)   b(1) + b(2)z + .... + b(nb+1)z
%      H(z) = ---- = ---------------------------------
%                                  -1                 -na
%              A(z)   a(1) + a(2)z + .... + a(na+1)z
%
%    zplot(b, a) plots the zeros and poles which determined by vectors b
and a
%    The plot includes the unit circle and axes for reference, plotted in
black.
%   Zeros are represented with a blue 'o', each pole with a red 'x'.
```

This function is similar to the Matlab function `zplane`, except that our function is in powers of z^{-1} instead of *z*. Play with `zplane` to see how this works. So for example, if

$$H(z) = \frac{z^2 - z}{z^2 - z + 0.5} = \frac{1 - z^{-1}}{1 - z^{-1} + 0.5 z^{-2}}$$

Then we would call `zplot([1 -1], [1 -1 0.5])` to create the following pole-zero plot:

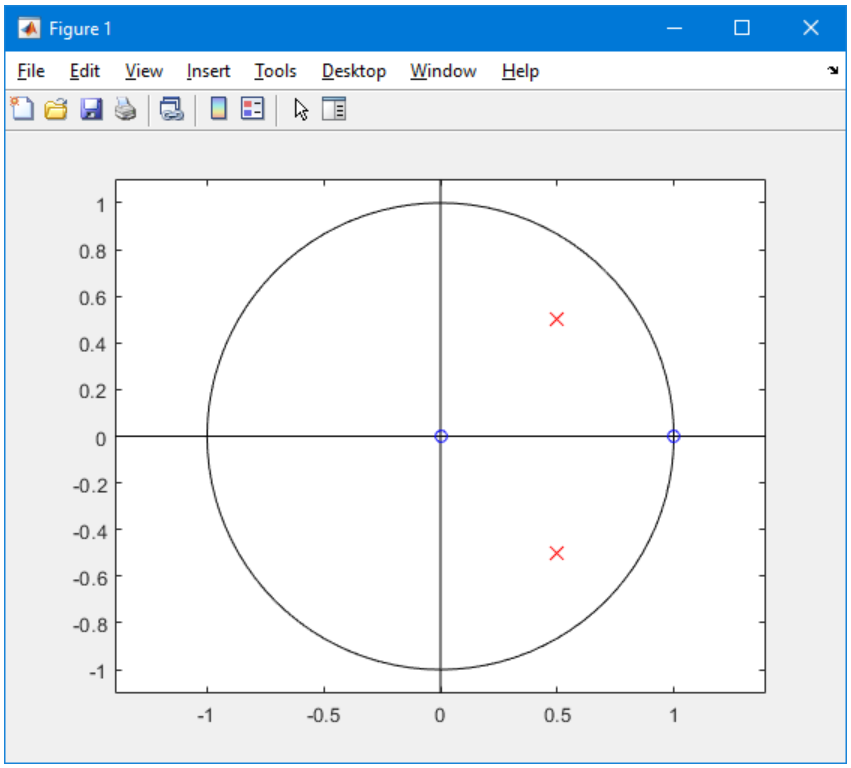

Figure 1

Note a few things about this plot:

- The image has an equal axis ratio (the same size ticks on the x-axis as on the y-axis). Explore Matlab's `axis` command to find out how to do this.
- The plot always shows at least the entire unit circle. To create the unit circle, I used the `rectangle` function (yes, that's right!)
- The size of the 'x' and 'o' can be changed using the 'MarkerSize' and 'LineWidth' attributes of the plot command. To read more about this, and thus to understand the plot command, type `doc plot`.
- I've chosen the range (i.e. maximum and minimum values of the x- and y-axes) so that there is at least 10% extra space on the left and right, top and bottom of the plot compared to the furthest-out pole or zero. Then I used the Matlab `axis equal` to set the equal axis ratio. Then I used `xlim` and `ylim` to discover what Matlab set as the limits of the x- and y-axes and finally, I redrew the axes so they would just fit into the plot.

In your function, you do **not** have to worry about the following:

- representing multiple poles and zeros on the plot. Just plot a single pole or zero at the correct position.
- pole-zero cancellations.
- non-causal systems

5.3.2 FPLOT: The relation between *H(z)* and the frequency response, *H(w)*

The Fourier transform is just the *z*-transform evaluated on the contour $z = e^{j\omega}$. We'll create a function called `fplot`, which is our own version of Matlab's `freqz` function:

```
function fplot(b, a)
% FPLOT Z-transform frequency response.
%    FPLOT(B,A,N) is the 256-point complex frequency response
%    of the filter B/A:
%                                  -1                -nb
%         jw  B(z)   b(1) + b(2)z + .... + b(nb+1)z
%      H(e) = ---- = ---------------------------------
%                                  -1                -na
%             A(z)   a(1) + a(2)z + .... + a(na+1)z
%
%    Numerator and denominator coefficients are given in vectors B and A.
The
%    frequency response is evaluated at 256 points equally spaced around
the
%    upper half of the unit circle (i.e. -pi<w<=pi)
%    If A isn't specified, it defaults to [1].
```

There's a bit of thinking to do here.

We will use Matlab's `fft` command to compute the frequency response (the DTFT) for us. To understand what `fft` does, realize that the `fft` is just computational way to compute the DTFT at a bunch of frequencies equally spaced between 0 and 2π. For example, consider a simple sequence of three impulses:

$$h[n] = \delta[n] + \delta[n-1] + \delta[n-2]$$

The *z*-transform is,

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

and so

$$H(\omega) = H(z)\big|_{z=e^{j\omega}} = 1 + e^{-j\omega} + e^{-j2\omega}$$

To compute this directly in Matlab, we make a vector of `h[n]`, then use Matlab's `fft` command to compute the Fourier transform at 16 points uniformly spaced between $\omega = 0$ and $\omega = 2\pi - 2\pi/16 = 15\pi/16$: $H(2\pi k/16)$, $\omega = 0 \le k < 16$.

This is how we do it:

```
h = [1 1 1];
H = fft(h, 16);
plot(abs(H));
```

- The frequency resolution is pretty crummy in this simple plot, only 16 points. This means we are basically sampling $H(\omega)$ for $\omega = 2\pi k/16, 0 \le k < 16$. Explore what happens if we increase the resolution of the plot by changing the 16 to 512, which is what we will want for `fplot`.
- In this simple version, we're only plotting the magnitude using Matlab's `abs` command.
- We're also ignoring the poles at $z = 0$. How do we include them?
- Here, we are plotting from $H(0)$ almost to $H(2\pi)$. Remember that point #1 of the array, `H`, corresponds to $H(0)$ and point #16 corresponds to $H(2\pi \cdot 15/16)$. Point #17 (if it existed) would correspond to $H(2\pi)$. In the function we need to write, `fplot`, we only want to plot from 0 to π, *inclusive*. How many points is that?
- Because of the way that the `fft` works, it is always most efficient (though not necessary in this case) to make the number of points a factor of 2, that why we will use 512 points in `fplot`.
- How do we plot $H(\omega)$ against ω (as a fraction of π) instead of against point number?
- How do we plot the phase?
- If $H(z)$ comprises both a numerator and a denominator term. How do we plot $|H(\omega)|$ and $\measuredangle H(\omega)$?

Now consider a transform

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

In this case,

$$H(\omega) = \frac{1}{1 + e^{-j\omega} + e^{-j2\omega}}$$

So,

```
H = 1 ./ fft([1 1 1], 16);
```

Finally, what would happen if we had

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots}.$$

Then,

```
H = fft(b, 16) ./ fft(a, 16);
```

Once you've completed your `fplot` function, here's what the frequency response of the previous example should look like:

```
fplot([1 -1], [0 -1 .5]);
```

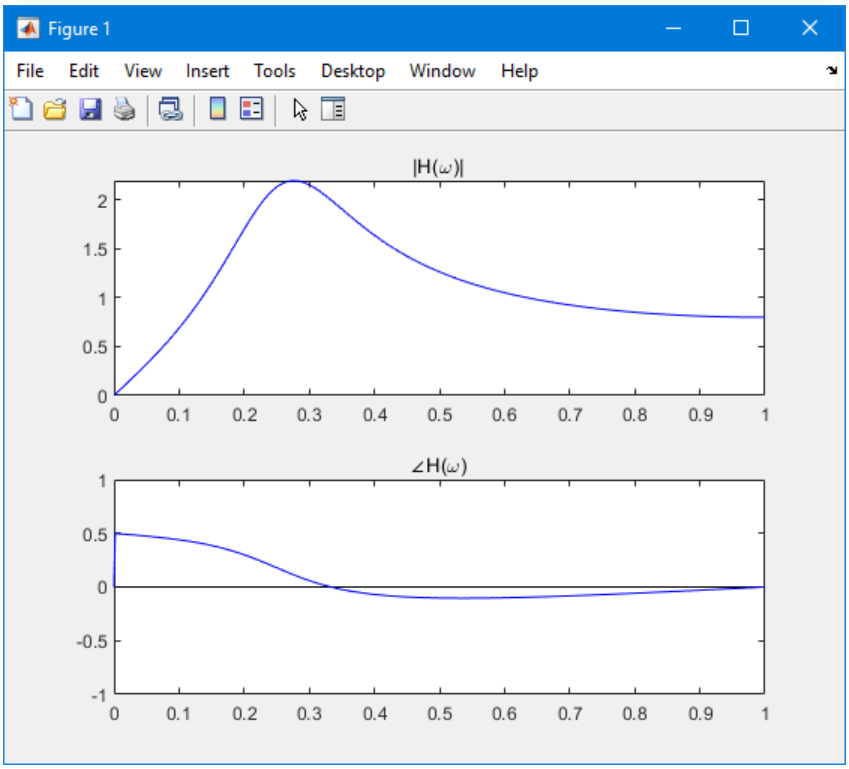

Figure 2

This plot displays points corresponding to frequencies, 0 to π *inclusive*. Note that the magnitude of this plot peaks at about $\omega = \pm\pi/4$, which is where the poles are closest to the unit circle. The magnitude is also minimum at $\omega = 0$, which is exactly what we expect (why?).

Some further technical points:

- Matlab is sometimes not very smart. If you try `fplot(1, 1)` it will give you a warning because it is trying to plot a constant (plus or minus a small amount). To fix this, it's smart to always set the lower y-limit of the magnitude plot to 0. Also, we'll set the y-limits of the phase plot to $[-1\ 1]$ as a fraction of π.

5.4 Assignment

Your assignment is to implement the three functions, `zplot` and `fplot`.

- Your functions should produce plots as close as possible to those shown in Figure 4.
- Your functions should *not* produce any text output nor should they call Matlab's `clf` command.
- Only your `fplot` function may call Matlab's subplot function, and it must *only* use `subplot(2, 1, 1)` and `subplot(2, 1, 2)`.
- Please download the following two functions:

- pzfplot.m is a function that I have written that will call your functions `zplot` and `fplot` to place the frequency response and plot-zero plot on a single figure. Here is the header for pzfplot.

```
function pzfplot(b, a)
% PZFPLOT Plot Frequency response and pole-zero plot of
      filter H(z)
```

  Just put `pzfplot` in the same directory as your functions. Figure 4 shows what `pzfplot([1 -1], [0 -1 .5])` should look like.

- Lab5_2024.m is (most) of your assignment. You will publish lab5_2024 to a pdf file as in previous labs.

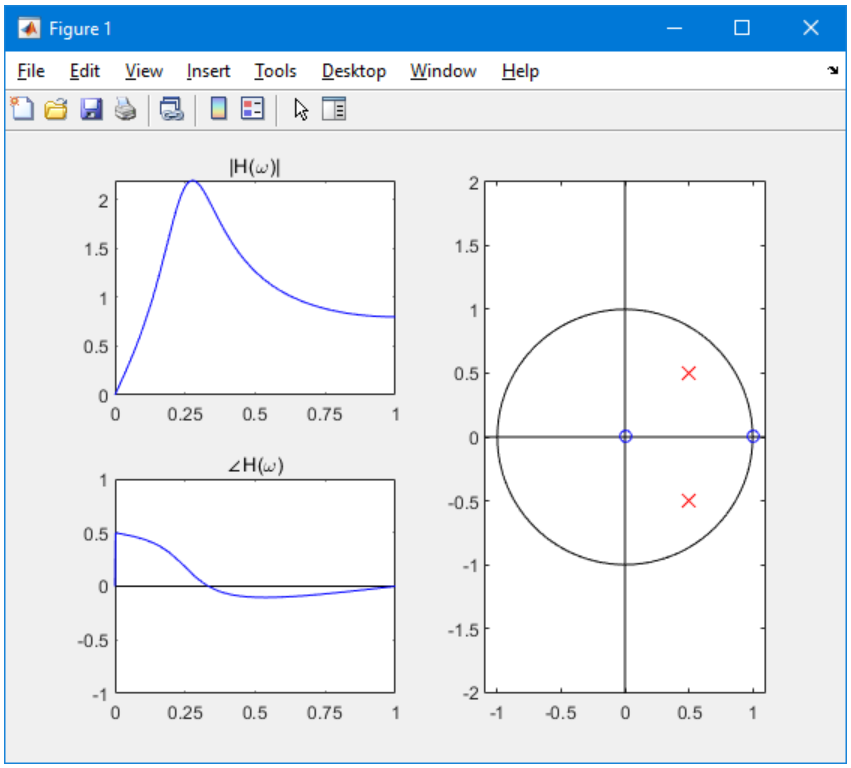

Figure 4

Look is the output of `pzfplot` for at a couple of interesting functions that you should be able to match with your code:

- **Rectangular window.** For example, when $h[n]$ is a rectangular window centered at $n = 0$, you'll get this:

```
pzfplot(ones(1, 11), 1);
```

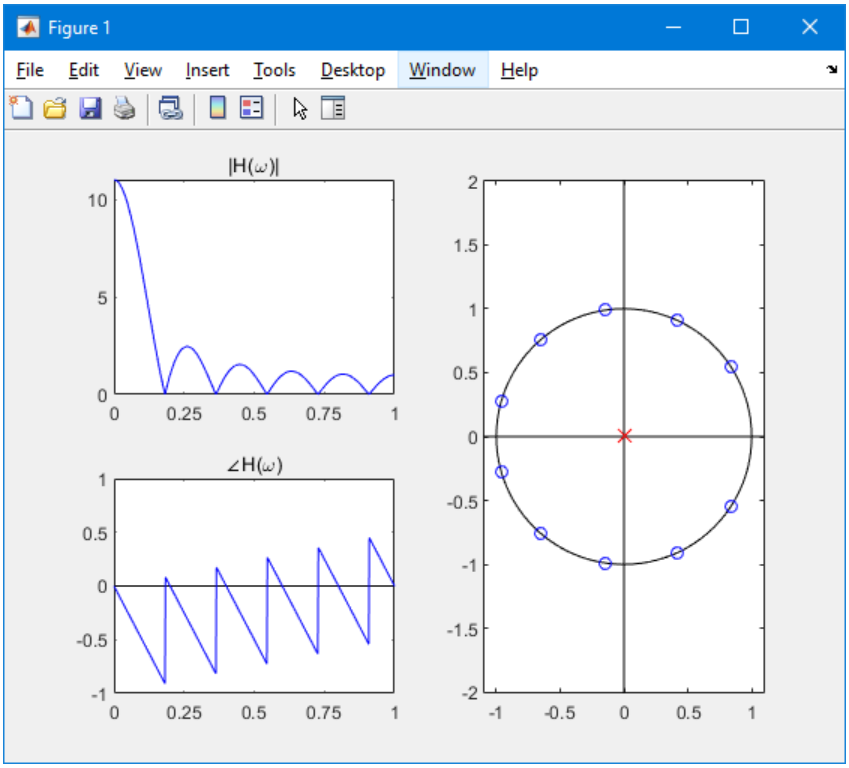

Figure 5

This is a FIR filter (of course). $|H(\omega)|$ is a sinc function, and $\measuredangle H(\omega)$ is linear. Why is this? The zeros are equally spaced on the unit circle. We have predicted both the fact that they are equally spaced and the fact that they are on the unit circle by knowing that $|H(\omega)|$ goes to zero at multiples of $2\pi/11$.

- **Kaiser window**. When $h[n]$ is a (causal) Kaiser window (we will design this filter in FIR filter lab), we get:

```
pzfplot(kaiser(11, 5), 1);
```

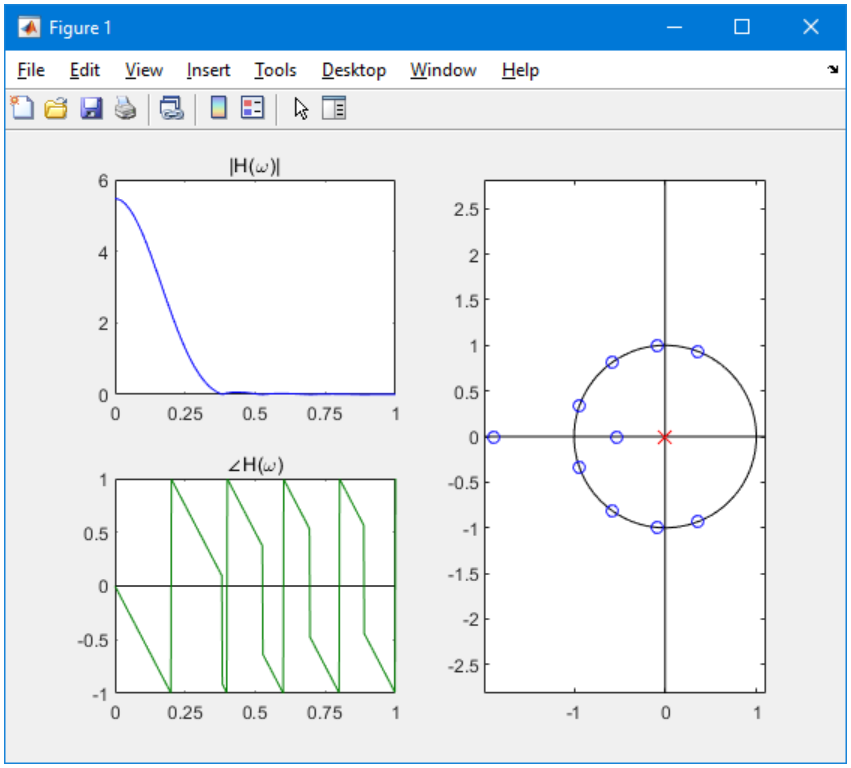

Figure 6

This is another FIR filter, in this case causal. Note that the impulse response is symmetrical, that is, real and even. This means that $H(\omega)$ will be real and even, which means that the phase will be exactly linear. The Kaiser window in this example is the same length as the rectangular window in the previous example, and both windows yield lowpass filters, but you can see a bunch of differences. Particularly note that, while the "main lobe" of $|H(\omega)|$ for the rectangular window is smaller than that of $|H(\omega)|$ for the Kaiser window (the main lobe of the rectangular window only extends to $\omega = 2\pi/11$), the other lobes -- the "side lobes" -- of $|H(\omega)|$ for the rectangular window's response are much higher in magnitude than the side lobes of $|H(\omega)|$ for the Kaiser window. This is the main reason we prefer the Kaiser window as a filter.