

San Francisco State University

Engineering 451

Laboratory #8 - Z transform

Purpose

The purpose of this lab is to help you understand the relations between the z-transform, $H(z)$, the pole-zero plot, the frequency response, $H(\omega)$, and the impulse response, $h[n]$.

Methods

First we will develop some tools:

- **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 $H(z)$;
- **IPlot**: a function to plot the impulse response, $h[n]$, given $H(z)$

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} ,

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + \dots + b(nb+1)z^{-nb}}{a(1) + a(2)z^{-1} + \dots + a(na+1)z^{-na}}$$

You determine the poles and zeros from the vectors a and b . Your first task is to write a Matlab function that displays the pole-zero plot given vectors for 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.
% Each zero is represented with a blue 'o' and each pole with a red
'x' on the plot.
```

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

$$H(z) = \frac{z^2 + 2z + 2}{z + .8} = \frac{1 + 2 * z^{-1} + 2 * z^{-2}}{z^{-1} + .8 * z^{-2}}$$

Then we would call `zplot([1 2 2], [0 1 .8])` to create the following pole-zero plot::

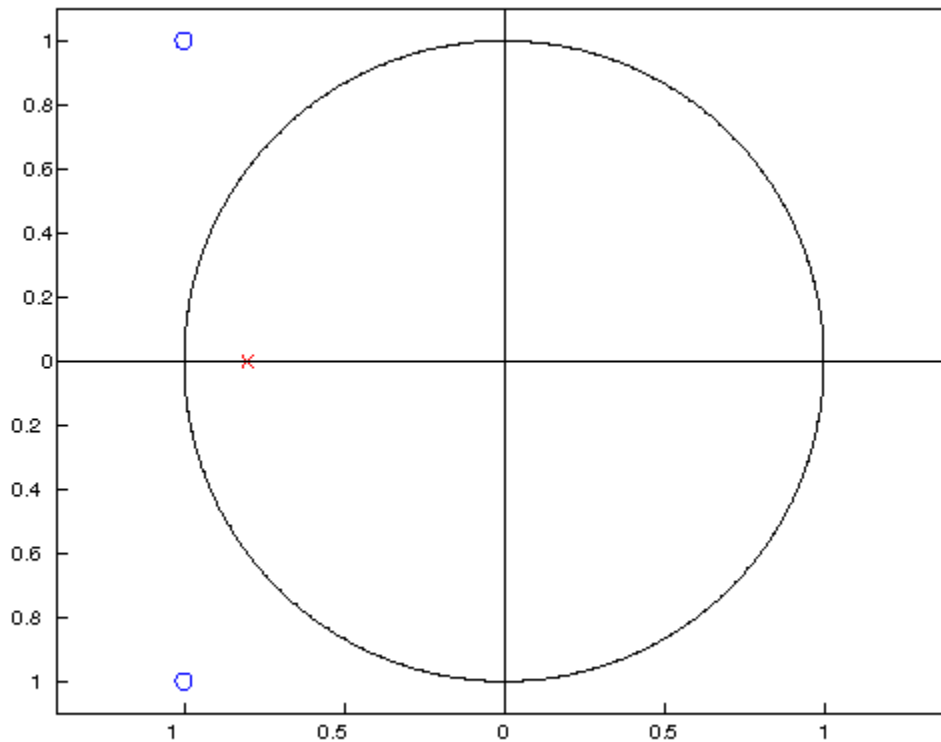


Figure 1

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.

- The size of the 'x' and 'o' have been changed using the 'MarkerSize' and 'LineWidth' attributes of the `plot` command. To read more about this, and thus to understand the `plot` command, use the `helpdesk` command and then type in 'plot'. Then read...
- 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 appropriate Matlab `axis` command to set the equal axis ratio. Then I figured out what Matlab set as the limits of the x- and y-axes (using `xlim` and `ylim`). Finally, I redrew the axes so they would just fit into the plot.

Make sure your function can correctly handle things like:

```
zplot([0 1 1], [0 1]);
zplot([0 1 1], [0 0 1]);
```

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.

FPLOTT: 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)
% FPLOTT Z-transform frequency response.
%   FPLOTT(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
%
%   given numerator and denominator coefficients in vectors B and A.
The
%   frequency response is evaluated at 256 points equally spaced
around the
%   upper half of the unit circle.
%   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 responses for us. To understand what `fft` does, realize that the `fft` is just computational way to compute the

DTFT at a bunch of frequencies spaced linearly between 0 and 2π . For example, consider a simple sequence of three impulses:

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

Obviously,

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

and so

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

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

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 $\omega = 2\pi k/16, 0 \leq k < 16$. 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.
- In this simple version, we're ignoring the poles at $z = 0$. How do we include them?
- In this simple example, we are plotting from $H(0)$ almost to $H(2\pi)$. Remember that point #1 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 get it to 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 $\angle H(\omega)$?

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

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

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 2 2], [0 1 .8]);
```

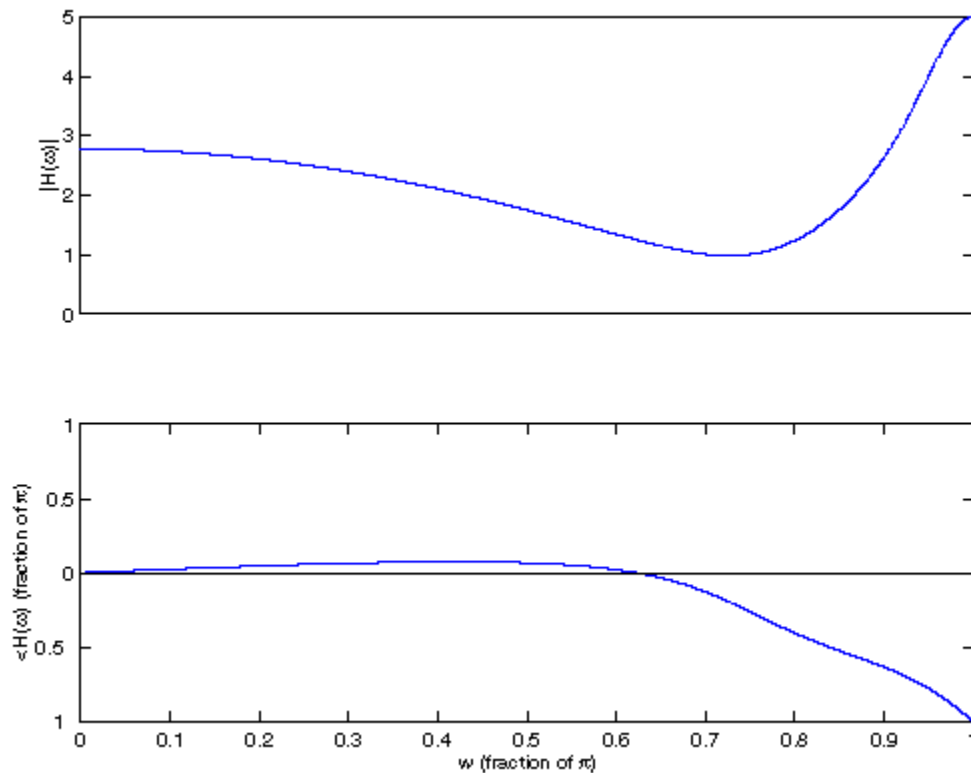


Figure 2

This plot displays points corresponding to frequencies 0 to π *inclusive*. Note that the magnitude of this plot peaks at $\omega = \pi$. The magnitude is also minimum near $\omega = 3\pi/4$, which is reasonable, since that's where the zeros are closest to the unit circle.

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, we'll always set the lower y-limit of the magnitude plot is 0. Also, we'll set the y-limits of the phase plot to $[-1 \ 1]$ as a fraction of π .
- Matlab sometimes has a [problem](#) plotting phase. We're going to ignore this, unless you feel ambitious.

IPLLOT: The relation between $H(z)$ and the impulse response, $h[n]$

This is what we want:

```
function iplot(b, a)
% IPLLOT Impulse response of system.
```

```

% IPLOT(B,A,N) is the N-point impulse response
% of the filter B/A:
%
%          -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
%
% given numerator and denominator coefficients in vectors B and A.
% N is specified according to the following rule:
%   If h[n] FIR, then N = length(h);
%   If h[n] is IIR and increasing (i.e. |h[n]|-->inf as n-->inf),
%       then N = 20;
%   If h[n] is IIR and decreasing (i.e. |h[n]|-->0 as n-->inf),
%       then the maximum N is determined such that
%       rms value of h(1:N) = 0.999 * rms value of h(1:1000).
%       However, in this case, N must also be chosen such that 10 <=
N <= 100

```

You should use Matlab's `filter` command to do the work of computing the impulse response. The input to `filter` should be an impulse array, which is 1 followed by 999 zeros. The challenging thing here is that we need to figure out the length of the impulse response we wish to plot. Obviously, we don't want to plot 1000 points. Instead, we will determine the length to be displayed according to the following rules:

- If $h[n]$ is FIR, we will display the entire impulse response;
- If $h[n]$ is IIR and increasing (i.e. $|h[n]|$ is increasing), we will limit the displayed length of the impulse response to 20 points. How do you know that $h[n]$ is increasing, just based on $H(z)$, specifically, on the denominator? Hint: what has to be true about the position of the poles with respect to the unit circle for a system to be stable
- If $h[n]$ is IIR and decreasing (i.e. $|h[n]|$ is decreasing), we will choose the length to display as follows:
 - we compute the impulse response for 1000 points (effectively infinite for our purposes)
 - we calculate the "total energy", defined as the sum of the energy of $h(1:1000)$.
 - we display only N points, such that the sum of the energy of $h(1:N)$ has 99.9% of the total energy. You definitely do not want to do this with a `for` loop! There are other ways in Matlab to compute a cumulative sum (i.e. `cumsum`) and to check for the value that is 99.9% of the maximum. Remember that you want the cumulative sum of the energy.
 - we limit the number of points in this case to $10 \leq N \leq 100$. That is, we always display at least 10 points and no more than 100 points.

Your `ipplot` command should correctly handle the case of non-causal systems. For example, if you do `ipplot([1 2 2], [0 1 .8])`, you should get:

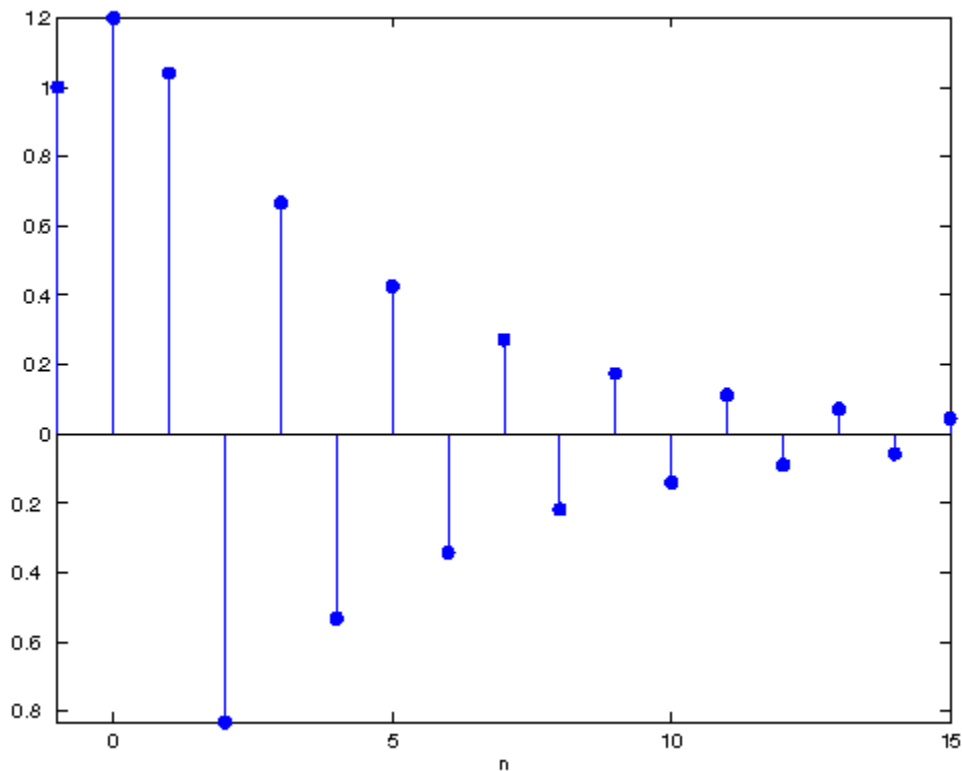


Figure 3

This is not the response of a causal system (how could you tell just from the pole-zero plot?), so the sample number won't start at zero! You'll notice that if you try to do `h = filter([1 2 2], [0 1 .8], x)`, where `x` is the impulse array, Matlab will give an error. It doesn't like leading zeros in the denominator polynomial, which is the characteristic of non-causal systems (why?). Accordingly, you need to keep track of how many there are and strip them out. You use that number to compute the starting value of the time axis (e.g., -1 in the previous example). Note also that in this example, we only display 17 points, because that's all that is necessary to show 99.9% of the maximum energy in the impulse response.

Your function should properly handle impulse responses that start at $n < 0$, $n = 0$ and $n > 0$; things like:

```
ipplot([0 0 1 1 1], [0 1]);
ipplot([0 0 1 1 1], [0 0 1]);
ipplot([0 0 1 1 1], [0 0 0 1]);
```

At what value of n should each of these plots start?

Assignment

Your assignment is to implement the three functions, ZPLOT, FPLOT and IPLOT.

- Your functions should produce plots as close as possible to those shown here.
- 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:
 - `fizplot.m` will basically call your functions `zplot`, `fplot` and `ipplot` to place the frequency response, plot-zero plot and impulse response on a single figure:

```
function fizplot(b, a)
% FIZPLOT Plot Frequency response, pole-zero plot and
impulse response of the filter H(z)
```

Just put `fizplot` in the same directory as your functions. Here's what `fizplot([1 2 2], [0 1 .8])` should look like:

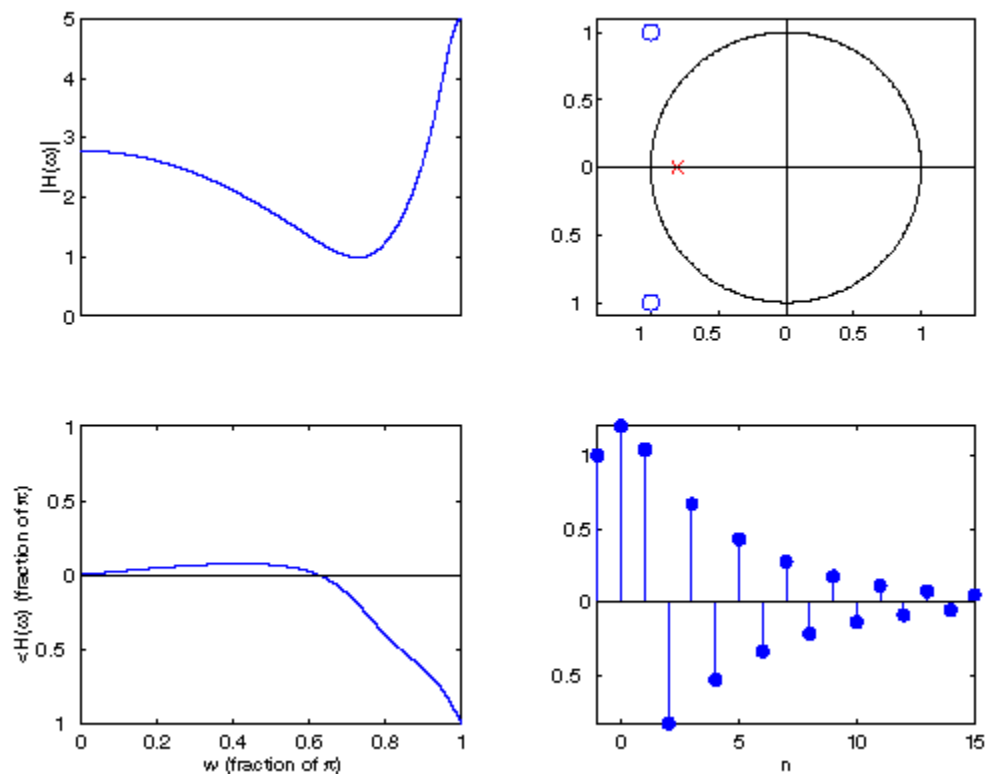


Figure 4

- Look at some interesting functions:

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

```
fizplot(ones(1, 11), [zeros(1, 5) 1]);
```

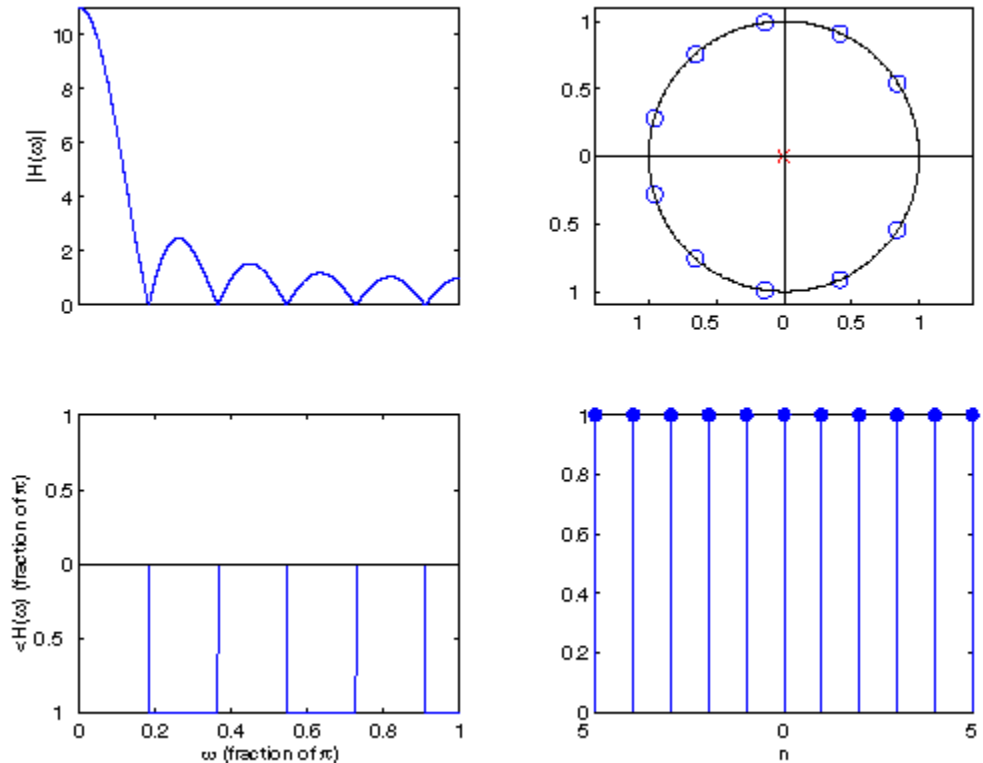


Figure 5

Can you explain the arguments in the `fizplot` command that generated this window (i.e. what the story with the `...[zeros(1, 5) 1]`)? How did I produce a window centered at $n = 0$? How many poles are at the origin?

This is a FIR filter (of course). $|H(\omega)|$ is a sinc function, and $\angle H(\omega)$ is not only linear, it is only either 0 or $-\pi$. 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 (such as we used in the [previous](#) lab), we get:

```
fizplot(kaiser(1, 11), 1);
```

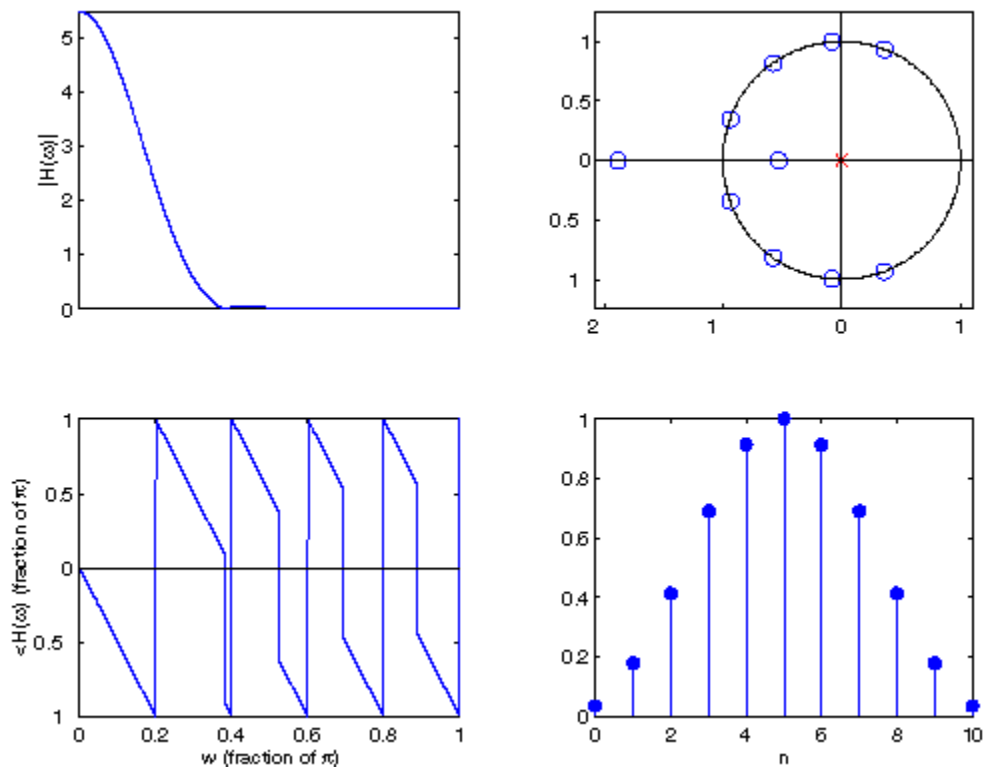


Figure 6

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.

- [lab8.m](#) is (most) of your assignment. Run it and you should get a bunch of pages. Submit them along with printouts of your code.
- Final question: The first filter in your assignment is

```
fizplot([1 2 2], [0 1 .8]);
```

This should just give you the same as Figure 2 of this write-up. This is a non-causal, maximum-phase bandstop filter. Your final task is to find and plot a causal, minimum-phase filter which has the *same* frequency-response magnitude as this filter.