Lab 5

This lab will focus on MATLAB as a tool to analyze and design filters.

Filtering in Matlab, the Easy Way

In this lab we will begin using the filter() command. It takes 3 parameters and they relate to the transfer function, so your first step is to convert the difference equation for your filter to a transfer function. The first parameter takes the numerator coefficients (in decreasing powers of z), and the second is the denominator coefficients (in decreasing powers of z). The third parameter is the signal that you want to filter. So, if your transfer function looks like

$$H(z) = (b_0 + b_1 z^{-1}) / (1 + a_1 z^{-1})$$

then your numerator coefficients are

```
B = [b0 b1];
```

and your denominator coefficients are

```
A = [1 a1]; % the first element of feedback coefficients must ALWAYS be 1!
```

and you can use them in `filter()` thusly:

```
signal = 2 * rand(1, 2*fs) - 1;
filteredsignal = filter(B,A,signal);
```

A Word of Warning

Matlab follows of the convention of B = numerator coefficients and A = denominator coefficients but this is not true everywhere else. Notice that in Lab 4 we used A to represent the numerator coefficients. One of the tricky things about DSP books, papers, software, etc, is that no one agrees on exactly what the A's and the B's represent. Which "religion" you belong to is entirely up to you, but you need to be aware of the possible confusion and always make sure you are doing the correct implementation!

Evaluating a Transfer Function in Matlab

One of the convenient things in Matlab is that not only can we easily perform operations on vectors and matrices, those vectors and matrices can hold complex numbers. In Problem 1, we will evaluate the transfer function directly. This is as easy as solving an algebraic polynomial if we remember that $z = e^{j\omega}$ with ω being the frequency in **radians per sample**. Once we evaluate H(z),

we will have a vector of complex numbers. The magnitude of each vector element is our magnitude response of the system, and the angle of each element is the phase response of the system.

Evaluating a Transfer Function in Matlab, the Easy(er) Way

It turns out that we can accomplish the same thing we did above by using the `freqz()` command. It takes three arguments, in the form `freqz(B, A, N)`. The coefficients B and A are handled in the same way as we did for `filter()`. N is the number of points we want to evaluate around the positive half of the unit circle. Note that we generally don't care about the negative half of the unit circle as it will be symmetric with the positive half. If we call freqz with no output arguments it will produce a plot for us (how convenient!), or we can take the return arguments H and W. H is the complex response of the system, and W is a vector of frequencies in radians per sample.

More Helpful Matlab Commands

(type "`help command_name`" to get more info about a command)

- `roots(C)` takes a polynomial and solves for its roots.
- `poly(A)` takes roots of a polynomial and creates a polynomial out of them.
- `zplane(Z,P)` takes the numerator (zeros) and denominator (poles) of a transfer function and plots its poles and zeros. Alternately, it can be called using the coefficients. See the documentation for more info.
- `abs()` takes the absolute value for real numbers, or takes the magnitude of complex numbers. You can give it a whole vector/matrix at once.
- `angle()` takes the angle of complex numbers.
- `real()` takes the real part of a complex number.
- `imag()` takes the imaginary part of a complex number.
- `disp()` display a variable in the command window.

Reson Filter Design:

For problem 2, we will be designing a two-pole resonant filter. Use the procedure discussed in class to specify the bandwidth B, the center frequency f_c,=ψ and find the pole radius R. Then, find the actual pole angle θ, and the amplitude A_0. With these parameters A_0, R, and θ you can generate your difference equation. Then, take the z-transform of that difference equation, and use the `filter()` function to filter the signal.

Assignment

Problem 1

Use the difference equation: y[n] = x[n] - 1.414*x[n-1] + x[n-2] + 1.273*y[n-1] - 0.81*y[n-2]

A) Write out the transfer function H(z) **(please use the equation editor in Word or similar!)**

B) Use the `roots()` command to determine the poles and zeros. Plot them using `zplane()` (include both the numeric values and your plot).

C) Convert the poles and zeros to polar coordinates (radius and angle). What do you suspect the magnitude response will look like? At what frequency will it have its greatest effect?

D) Starting with the code below, write a script to evaluate the transfer function. Plot the magnitude and phase in a 2 x 1 subplot (use a linear frequency scale). The frequency scale on the plot should be in Hz.

```
fs = 48000;  % note this is NOT 44100!
freq = linspace(0, fs/2, 1025);
freq = freq(1:(end-1));
w = % you will need to fill in the correct conversion
here
z = exp(1i*w);
Hz = % fill in the transfer function here
```

E) Use the `freqz()` command to calculate/plot the magnitude and phase at 1024 points (using the coefficients from the transfer function you found in part A). The frequency scale on the plot should be in Hz (NOTE: You will need to convert the output "W"!). Use a linear frequency scale.

F) Do your plots from D and E match? (Hint: yes, they should). What does this filter accomplish, and what might be a practical application?

Problem 2

Use MATLAB to design a 2nd order resonant filter to boost frequencies at 2000 Hz, with a bandwidth of 1000 Hz.

A) Plot the filter's magnitude response, phase response, and pole zero plot.

B) Use Matlab to calculate the poles and zeros in both cartesian and polar form.

C) Apply this filter to a wav file of your choice. Include both the original and filtered versions with your submission (please name them clearly).

BONUS: Use MATLAB to design a 2nd order resonant filter that sweeps the frequency it boosts from 200Hz to 4000Hz over the course of the length of the signal. Apply this to a white noise signal. Export the result to a wav file.

Lab 5 – IIR Filters and Pole-Zero Plots

Nate Paternoster

Part 1 – Using the difference equation:

$$y_n = x_n - 1.414x_{n-1} + x_{n-2} + 1.273y_{n-1} - 0.81y_{n-2}$$

a. *Write out the transfer function H(z).*

$$y_n - 1.273y_{n-1} + 0.81y_{n-2} = x_n - 1.414x_{n-1} + x_{n-2}$$

$$y(z)z^0 - 1.273y(z)z^{-1} + 0.81y(z)z^{-2} = x(z)z^0 - 1.414x(z)z^{-1} + x(z)z^{-2}$$

$$y(z)[1 - 1.273z^{-1} + 0.81z^{-2}] = x(z)[1 - 1.414z^{-1} + z^{-2}]$$

$$\frac{y(z)}{x(z)} = \frac{[1 - 1.414z^{-1} + z^{-2}]}{[1 - 1.273z^{-1} + 0.81z^{-2}]}$$

$$H(z) = \frac{1 - 1.414z^{-1} + z^{-2}}{1 - 1.273z^{-1} + 0.81z^{-2}}$$

b. *Use the roots() command to determine the poles and zeros. Plot them using zplane() (include both the numeric values and your plot).*

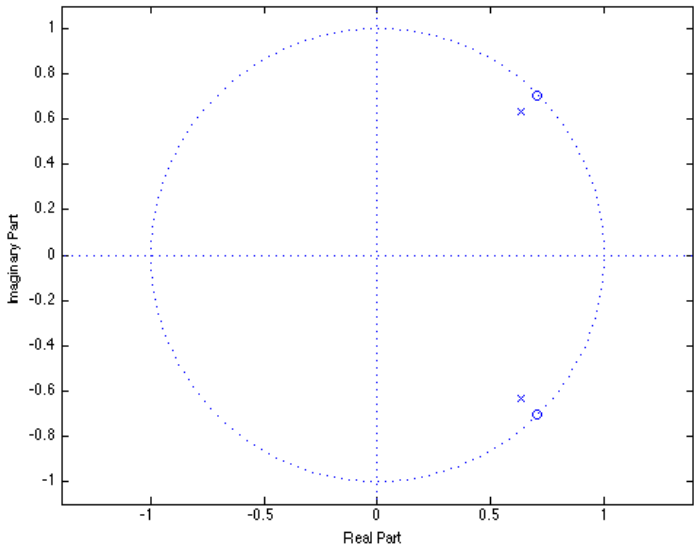

Figure 1a: The z-plane of the filter's transfer function. There are two poles and two zeros

The zeros are at:
0.7070 + j0.7072
0.7070 – j0.7072

The poles are at:
0.6365 + j0.6363
0.6365 - j0.6363

c. *Convert the poles and zeros to polar coordinates (radius and angle). What do you expect the magnitude response will look like? At what frequency will it have its greatest effect?*

In polar coordinates the zeros are at:
1∠+45.009°
1∠ -45.009°

In polar coordinates the poles are at:
0.9∠+44.991°
0.9∠ -44.991°

I expect that there will be some gain at the two poles. It will not be as great as it could be because there are also zeros in roughly the same location. The zeros' magnitudes are not as great as the poles so they will not cancel the poles. The greatest effect should occur at the poles (at an angle of +45°). This converts to (45/360)*Fs = 6000Hz. (Realistically, we will only see the pole within the nyquist frequency – the top one).

d. *Starting with the code below, write a script to evaluate the transfer function. Plot the magnitude and phase in a 2 x 1 subplot (use a linear frequency scale). The frequency scale on the plot should be in Hz.*

```
fs = 48000;  % note this is NOT 44100!
freq = linspace(0, fs/2, 1025);
freq = freq(1:(end-1));
w = % you will need to fill in the correct conversion here
z = exp(1i*w);
Hz = % fill in the transfer function here
```

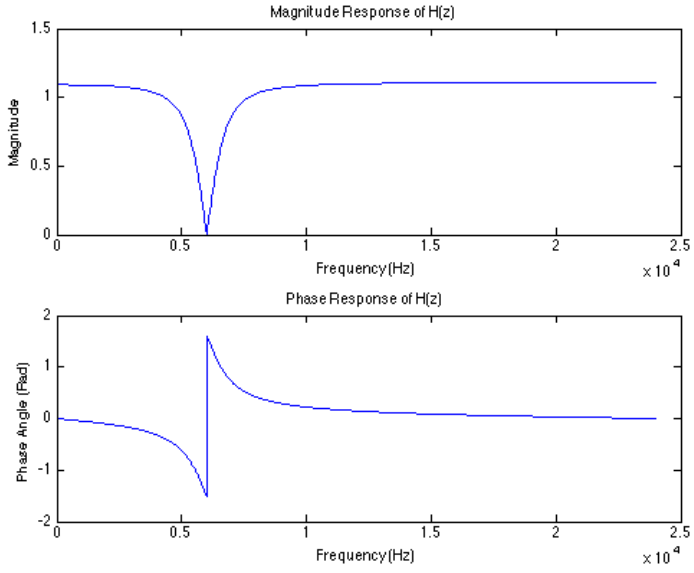

Figure 1b: The magnitude and phase response of the transfer function

The angular frequency w for this script is:

```
w = (freq*2*pi)/fs;
```

The transfer function Hz for this script is:

```
Hz = (1-1.414*z.^(-1)+z.^(-2))./(1-1.273*z.^(-1)+0.81*z.^(-2));
```

e. *Use the freqz() command to calculate/plot the magnitude and phase at 1024 points (using the coefficients from the transfer function you found in part A). The frequency scale on the plot should be in Hz (NOTE: You will need to convert the output "W"!). Use a linear frequency scale.*

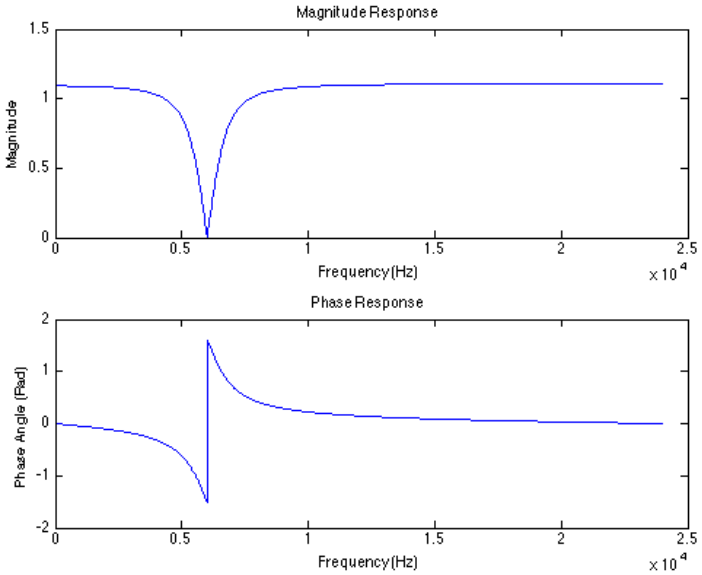

Figure 1c: The magnitude and phase response found using freqz()

e. *Do your plots from D and E match? (Hint: yes, they should). What does this filter accomplish, and what might be a practical application?*

Figure (1b) and (1c) both show the magnitude and phase response of the transfer function. They are identical and demonstrate that the magnitude and phase plots may either be calculated and plotted from the transfer function directly (by using the matlab commands *abs(Hz)* and *angle(Hz)*) or using just the coefficients of the transfer function and the matlab command *freqz()*.

This filter notches out a frequency around 6kHz. This resembles a very narrow bandstop filter (a high Q value). It may be used to remove an unwanted hiss from the recording of a cheaper video camera or other audio device.

Part 2 – Design of a 2nd-order resonant bandpass filter

Specifications:
- Center frequency = 2000Hz
- Bandwidth = 1000Hz

$$\boldsymbol{f_s = 44100Hz, B = bandwidth, \omega_c = center\ frequency}$$

$$B = \frac{1000Hz * 2\pi}{f_s} = \frac{1000Hz * 2\pi}{44100Hz} = 0.1425\ rad/sample$$

$$\boldsymbol{\varphi = 2000Hz}$$

$$\omega_c = \frac{2000Hz * 2\pi}{f_s} = \frac{2000Hz * 2\pi}{44100Hz} = 0.2850\ rad/sample$$

$$\boldsymbol{R = magnitude\ of\ the\ pole's\ phasor}$$

$$R = 1 - \frac{B}{2} = 1 - \frac{0.1425\ rad}{2} = 0.929$$

$$\boldsymbol{\theta = angle\ of\ the\ pole's\ phasor}$$

$$\theta = \cos^{-1}\left(\frac{2R}{1+R^2}cos\varphi\right) = \cos^{-1}\left(\frac{2(0.929)}{1+(0.929)^2}\cos\left(0.2850\right)\right) = 0.294\ rad$$

$$\boldsymbol{A_0 = gain\ of\ the\ filter}$$

$$A_0 = (1-R^2)sin\theta = (1-(0.929)^2)\sin(0.294\ rad) = 0.040$$

- The equation for this filter may be written in the form:

$$y_t = A_0 x_t + 2Rcos\theta y_{t-1} - R^2 y_{t-2}$$

$$y_t = 0.040x_t + 1.778y_{t-1} - 0.863y_{t-2}$$

- Now we must derive the transfer function H(z) of this resonant filter:

$$y_t = 0.040x_t + 1.778y_{t-1} - 0.863y_{t-2}$$

$$y_t - 1.778y_{t-1} + 0.863y_{t-2} = 0.040x_t$$

$$y(z)z^0 - 1.778y(z)z^{-1} + 0.863y(z)z^{-2} = 0.040x(z)z^0$$

$$y(z)[1 - 1.778z^{-1} + 0.863z^{-2}] = 0.040x(z)$$

$$\frac{y(z)}{x(z)} = \frac{0.040}{[1 - 1.778z^{-1} + 0.863z^{-2}]}$$

$$H(z) = \frac{0.040}{1 - 1.778z^{-1} + 0.863z^{-2}}$$

a. *Plot the filter's magnitude response, phase response, and pole zero plot.*

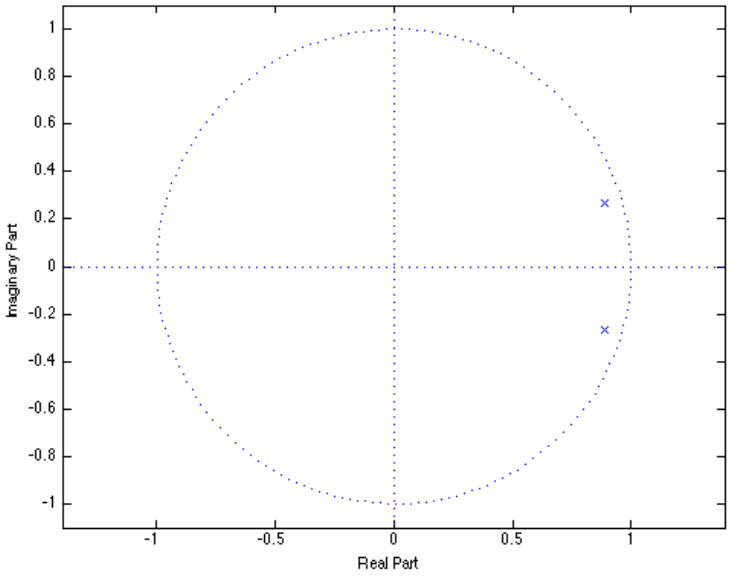

Figure 2a: The pole-zero plot of the resonant bandpass filter. There is a complex pair of poles.

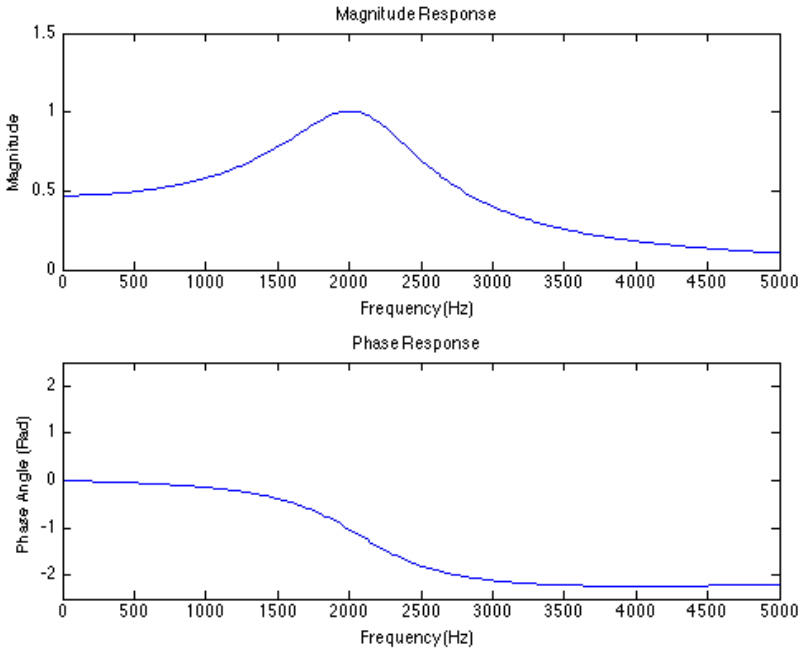

Figure 2b: The magnitude and phase plots of the resonant bandpass filter.

We can see that there is a gain at the center frequency of 2000Hz and a bandwidth of roughly 1000Hz. The low stop band is not as attenuated as the high stop band because the two poles are close enough to interfere with each other (as seen on the z-plane in figure (2a)). This could be fixed by implementing two zeros at +1 and -1 on the real axis (at the cost of a slower rolloff).

b. *Use Matlab to calculate the poles and zeros in both cartesian and polar form.*

Since I did not design this filter as a reson-z filter, there are no zeros.

The poles are at:

0.8890 + j0.2696
0.8890 - j0.2696

In polar coordinates these are:

0.9290∠+16.870°
0.9290∠ -16.870°

c. Apply this filter to a wav file of your choice. Include both the original and filtered versions with your submission (please name them clearly).

[Included]