EGR 111
Filters

This lab introduces digital filters which are used in many different systems to remove noise from signals, separate signals, and to perform other processing.

New MATLAB commands: filter, fir1, freqz

1. Background

Most engines have an air filter that allows air to pass through the filter and into the engine, but the filter blocks larger particles like dust that would be harmful to the engine. A digital filter is a device that is used to block signals based not on their size, but on their frequency. For example, the antenna in an FM radio receives all of the radio stations added together. Since each station is assigned a different frequency, a filter can be used to block all of the stations except one, which is then played to the speakers. Without the filter, the output of a radio would be corrupted by all of the other radio stations and would be completely unusable.

An amplifier such as a stereo takes a relatively small input signal from a radio or MP3 player and generates a larger output signal that drives the speakers. The output signal y(t) is the input signal x(t) multiplied by a number that we call the gain g: y(t) = g * x(t). For amplifiers, the gain is the same for all frequencies. Filters, however, are more general in that the gain can be different for different frequencies. If the gain is near 1, we say that the filter passes that frequency because the output is almost the same as the input. However, if the gain is near zero, we say that the filter blocks that frequency because the output is close to zero.

Filters are classified according to which frequencies they pass. A lowpass filter, for example, allows low frequency signals to pass and blocks high frequency signals. On the other hand, a highpass filter allows high frequency signals to pass and blocks low frequency signals.

Using a filter in MATLAB involves two steps. First a command like `fir1` is used to design the filter, and then the output of the filter is computed using the `filter` command.

2. Lowpass Filter

Let's create a lowpass filter and see how it works.

Exercise 1: Type the following commands into a new script file.

```
fs = 44100;        % sampling frequency (in Hz)
b = fir1(300, 1000/(fs/2)); % design lowpass filter
```

By default, the `fir1` function designs lowpass filters (the last character in the command name is the number 1, not the letter l). The first argument is the order of the filter. Using a higher order filter results in a better filter, but requires more computation. The second argument is the cutoff frequency normalized to one half the sampling rate. Lowpass filters pass frequencies below the cutoff frequency and block signals above the cutoff frequency. So the command above designs a 300th order lowpass filter with a cutoff frequency of 1000 Hz for a sampling rate of 44100 Hz.

Next, let's generate a 500 Hz cosine, pass it through the filter, and plot the input and output of the filter. Add the following lines to your script file.

```
Ts = 1/fs;        % sampling interval (in seconds)
t = 0:Ts:2;     % sample every Ts seconds for 2 seconds
x = cos(2*pi*500*t);    % generate a 500 Hz cosine
y = filter(b,1,x);      % filter the 500 Hz cosine
plot(t,x,t,y)           % plot the input and output signals
legend('x[n] - Input','y[n] - Output')
```

Zoom in on the graph so you can see just one or two periods as shown below.

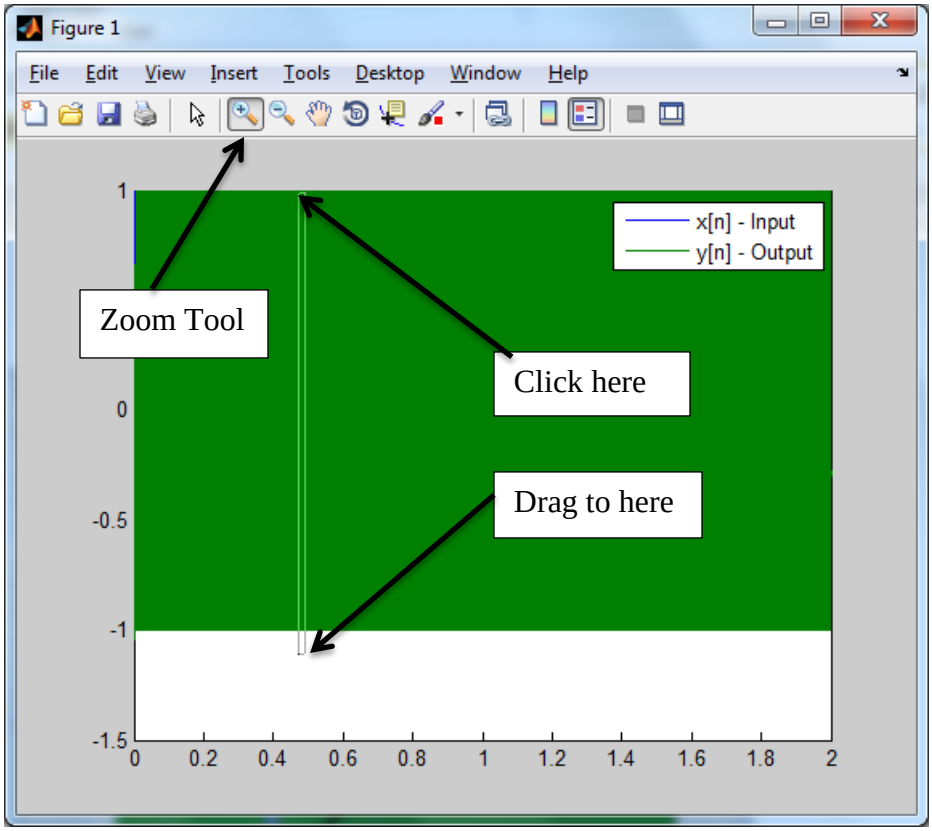
Figure 1
File Edit View Insert Tools Desktop Window Help
x[n] - Input
y[n] - Output
Zoom Tool
Click here
Drag to here
1
0
-0.5
-1
-1.5
0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6 1.8 2

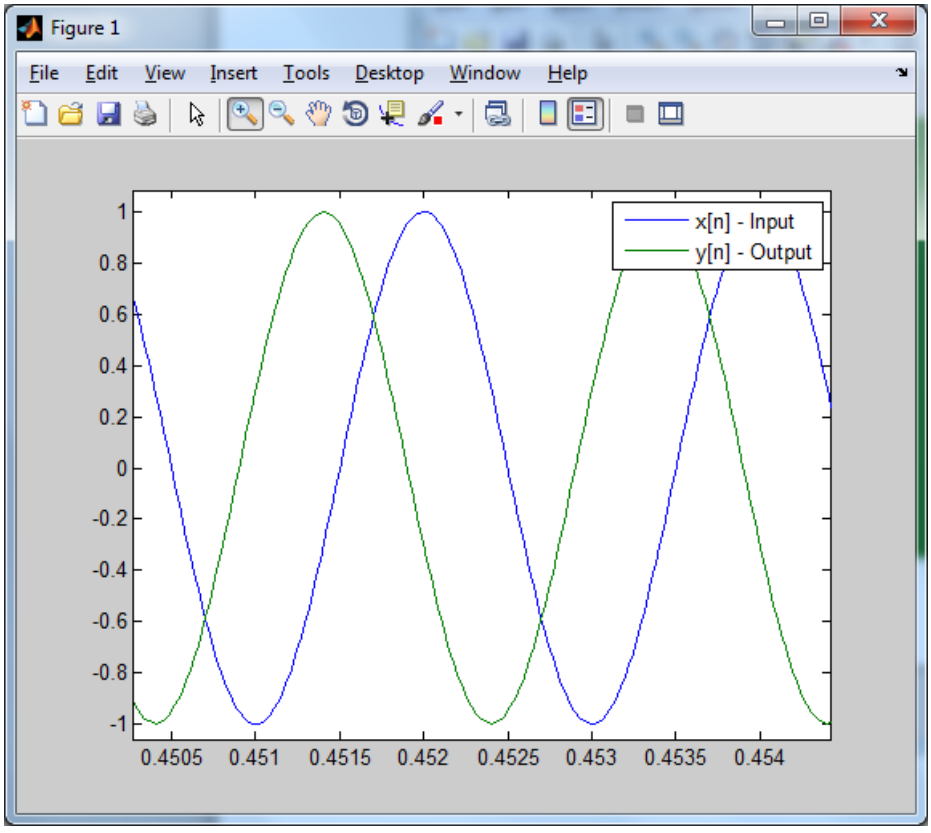
Figure 1
File Edit View Insert Tools Desktop Window Help
x[n] - Input
y[n] - Output
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0.4505 0.451 0.4515 0.452 0.4525 0.453 0.4535 0.454

Use the data cursor to measure the peak amplitude of the input and output signals, and calculate the gain (at 500 Hz) using the following equation:

$$gain = \frac{Peak\ Output\ Value}{Peak\ Input\ Value}$$

Hint: The peak output value should be nearly the same as the peak input value, so the gain should be close to 1. Since the gain is close to 1, we say that the filter passes the 500 Hz cosine.

Listen to the input and output signals by typing the following command into MATLAB's command window:

```
sound(x,fs)
sound(y,fs)
```

The output should sound the same as the input because the filter passes the 500 Hz cosine.

Now change your script file so that the input is a 1500 Hz cosine and measure the gain. Also listen to the input and output signals.

What gain did you measure at frequency 500 Hz? _____
What gain did you measure at frequency 1500 Hz? _____

Does the filter pass the 1500 Hz cosine?

Δ

Exercise 2: It would be nice if there were an easy way to make a graph of the gain for <u>all</u> frequencies, and the `freqz` command does just that. It generates a graph of gain versus frequency which is called the frequency response. Call `freqz` with the coefficients from the filter as shown below.

```
[H, f] = freqz(b,1,2^16,fs);
plot(f,abs(H));
```

Use the data cursor to measure the gain from the graph at 500 and 1500 Hz and compare your results to those you found above.

What gain did you measure at frequency 500 Hz? _____
What gain did you measure at frequency 1500 Hz? _____

Δ

<u>Checkpoint 1:</u> Show the instructor your results from Exercises 1 and 2.

3. Highpass Filter

The `fir1` function can also be used to design highpass filters that pass high frequencies. To demonstrate, download the file "Take Five excerpt.wav" from the course website and save it to your P:\MATLAB folder. Make a new script file and copy (or type) the following commands that load the file and send it to the speakers.

```
[x,fs] = wavread('Take Five excerpt.wav');
soundsc(x,fs)
```

Run the script file and listen for the various instruments in the music (Take Five by Dave Brubeck). Next add the following commands to design a 300^{th} order highpass filter with a cutoff frequency of 3000 Hz and plot the gain.

```
b = fir1(300,3000/(fs/2),'high');
[H, f] = freqz(b,1,2^16,fs);
plot(f,abs(H));
```

Verify that the gain is close to zero for low frequencies and close to one for frequencies above 3000 Hz, so we say that the filter passes high frequencies. Finally pass the signal thru the filter and listen to the output.

```
y = filter(b,1,x);
soundsc(y,fs)
```

Notice that the high pass filter only passes the sound of the cymbal which generates high frequencies; the other instruments are blocked by the highpass filter.

4. Noise Removal

One common use for filters is to remove unwanted noise from a signal.

Exercise 3: Download the file "birds_jet_noise.wav" from the course website. Load the file into MATLAB and listen to it. Notice that it is a recording of bird chirps with jet engine noise in the background. The jet engine noise is at frequencies below about 1800 Hz, and the bird chirps are above 1800 Hz. Design a filter that passes the bird chirps and blocks the jet engine noise. Plot the gain. Pass the signal through the filter. Did the filter pass the bird chirps and remove the noise?

Exercise 4: Design a filter that passes the jet engine noise and blocks the bird chirps. Plot the gain. Pass the signal through the filter. Did the filter pass the noise and block the bird chirps?

Checkpoint 2: Show the instructor your results from Exercise 3 and 4.