ECEn 380 Signals and Systems Laboratory

Lab 5
Discrete Time Filtering

Due Dates

This is a three-week lab. Neither section will meet during the week of Thanksgiving. Completion and electronic submission of the lab for both sections will be due Wednesday, December 7th by 11:59pm. (Note that this is immediately after your final lab sessions, which are December 5th for Section 1 and December 6th for Section 2.) Please ensure that you get all TA check-off completed by your final lab sessions.

The first week (Task 1) of this lab consists entirely of a reading assignment. Tasks 2 and 3 consist in the design (in Matlab) of discrete time filters that will feed directly into your ECEn 390 laser tag project next semester.

Objectives

The purpose of this assignment is to introduce you to the design and testing of discrete-time filters in Matlab, and to apply these skills to designing the discrete-time filters that will be needed for the ECEn 390 laser tag project next semester.

The laser tag system supports ten different players, each assigned a different laser modulation frequency ranging from approximately 1.4 kHz to 4.2 kHz. In order to analyze the data in discrete time, we must use a sampling frequency that is greater than twice the highest player frequency. However, we don't want to use too high of a sampling frequency since we have limited processing power in the laser tag system. A sampling frequency of 10 kHz is a good choice, since it is above the Nyquist rate (2*4.2 kHz = 8.4 kHz).

However, the boards that you will be using in 390 capture data at a sampling frequency of 100 kHz (100,000 samples/sec). We will need to decimate, or down-sample, this data to 10 kHz (10,000 samples/sec). In Task 2, you will design a **discrete time** anti-aliasing (low pass) filter with a cut-off frequency of 4.6 kHz. You will apply this filter to data captured at a 100 kHz sampling frequency, so that you mostly eliminate frequencies over about 5 kHz (remember the cut-off will not be perfect at 4.6 kHz). You are then ready to take the filtered 100 kHz data and throw away 9 out of every 10 samples, yielding data sampled at 10 kHz. As previously mentioned, this process is called "decimation."

Once we have the laser tag signal data decimated to 10 kHz, we will pass the decimated signal through 10 different discrete-time **band pass filters**, each with a passband centered around one of the player frequencies. In Task 3, you will design and test these filters. By comparing the **signal energy** of the 10 outputs from these filters, you should be able to determine whether there was a player "hit" in the signal, and which player it was.

Reading Assignment

This lab brings together a number of different concepts, and as such is a nice way to end the class. The material needed from the text is listed below.

Topic	Reading Assignment
Analog Butterworth filters	Section 6-8, pp. 279–287
Sampling	Section 6-12, pp. 306–321
Discrete-time convolution	Section 7-5, pp. 348–350
The z-transform	Section 7-6, pp. 350–354
Stability	Section 7-11, pp. 365–367
System frequency response	Section 7-12, pp. 367–372
DTFT	Section 7-14, pp. 377–383
DFT	Section 7-15, pp, 383–387
Windowing	Section 7-15.2, pp. 384–385
Basic discrete-time filter concepts	Section 8-1, pp. 397–402

A handout entitled "An Introduction to Discrete-Time Filter Design" applies the reading assignments from the text to filter design.

The following Matlab commands may be helpful for this lab: `load, :, length, log10, exp, conv, fft, freqz, fftshift, zplane, filter, pwelch, rectwin, butter, ellip, cheby2, cheb2ord`.

In addition, the following MATLAB plotting commands will be used: `plot, subplot, grid, xlabel, ylabel, axis.`

Task 1. Introduction to Discrete-Time Filtering: FIR and IIR Filters

Read the handout entitled "An Introduction to Discrete-Time Filter Design." Please pay special attention to the design of finite impulse response (FIR) filters (ending on page 18 of the handout). You should also read the remainder of the handout (pages 19 and beyond), which talks about the recursive, or infinite impulse response (IIR) filters. However, we are not going to emphasize much of what is covered for IIR filters, and are going to allow you to use the "butter" command in Matlab to design your bank of 10 IIR bandpass filters.

Task 2. Design of a Discrete-Time Low-Pass Antialiasing FIR Filter

In this task you will design your discrete-time antialiasing filter that will allow you to filter the 100 kHz sampled data as part of the decimation process to take it down to 10 kHz sampled data. The filter should have a length of 81 samples and a corner frequency of 4.6 kHz.

Procedure

As mentioned, all of this lab will be conducted using Matlab.

1. Design several versions of an FIR filter with total length of 81 and corner frequency of 4.6 kHz.

    a. Review the FIR filter design covered on the first 18 pages of the handout that you read in Task 1. Note that in the reading, they actually show you how to design a band-pass filter (not a low-pass filter). All we need to do to use similar code to produce a low-pass instead of a band-pass filter is specify a center frequency of 0 (which makes the cosine in the derived filter impulse response always equal to 1, so it goes away and we are left with a simple sinc function as our impulse response, which we would expect).

    b. First, design an "ideal" filter with no additional windowing, using the following code:

```
N = 81;                 % total number of samples in the filter
L = (N-1)/2;            % the filter will go from –L to L
n = (-L:L);             % this is our sample index

f_corner = 4600;        % corner frequency of our low-pass filter in Hz
f_s = 100000;           % our sampling frequency in Hz

h_ideal_FIR = 2*f_corner/f_s*sinc(n*2*f_corner/f_s);  % ideal sinc
            % IMPORTANT: in the code above we convert f_corner from
            % Hz to cycles/sample by dividing by f_s
stem(n, h_ideal_FIR);
```

    Now look at the frequency response of this "ideal" filter over a frequency range from 0 to 10 kHz:

```
f_axis = linspace(0, 10000, 2000);  % frequency axis with 2000 points
H_ideal_FIR = freqz(h_ideal_FIR, 1, f_axis, f_s);
            % IMPORTANT: in the code above, the first two parameters
            % to freqz are the b and a coefficients for the filter.
            % Remember that the b coefficients for an FIR filter
            % are just the impulse response values, and a_0 = 1 while
            % all of the other a coefficients are zero.
plot(f_axis, 20*log10(abs(H_ideal_FIR)));
xlabel('frequency (Hz)');
ylabel('magnitude (dB)');
```

    c. Show **both the frequency response AND a stem plot of the impulse response** of this filter in your lab notebook. Comment in your lab notebook on how "ideal" the frequency response of this low-pass FIR filter is. Does your corner frequency match the -3dB point? Is there much "ripple" in the filter?

    d. Now add to your code above to design 3 additional filters using any 3 of the different windows shown on page 15 of the filter design handout. **Note: You will need to**

transpose the window function to get it to match the size of h_ideal_FIR, which is 1x81. By default, the window functions will produce something that is 81x1.

e. Plot **both the stem plots AND the frequency response** of each of these filters and include in your lab notebook. **You can make use of the Matlab "hold on" command if you wish to reduce the total number of graphs.** Comment on how the frequency response is improved when you use a window. Which of your filters do you think will perform the best?

f. IMPORTANT: Save your Matlab code! You will want it for ECEn 390 next semester.

2. Test all four of your filters using input sampled data with sinusoids at a variety of frequencies.

   a. Create a signal sampled at 100 kHz that is 500 msec in duration and contains sinusoids at frequencies 4 kHz, 4.6 kHz, 6 kHz, 25 kHz, and 40 kHz using the following code:

```
t = 0:1/f_s:(0.5-1/f_s);      % time axis in seconds
sin1 = sin(2*pi*4000*t);      % sinusoid at 4 kHz
sin2 = sin(2*pi*4600*t);      % sinusoid at 4.6 kHz
sin3 = sin(2*pi*6000*t);      % sinusoid at 6 kHz
sin4 = sin(2*pi*25000*t);     % sinusoid at 25 kHz
sin5 = sin(2*pi*40000*t);     % sinusoid at 40 kHz
s1 = sin1+sin2+sin3+sin4+sin5;      % create the overall signal
```

   b. Plot this signal in the frequency domain using the following code:

```
S1 = fft(s1);                 % take the FFT of the signal s1
f1 = linspace(0, 50000, length(S1)/2);    % freq axis from 0 to 50 kHz
plot(f1, abs(S1(1:length(S1)/2)));
```

   Show this plot in your lab notebook. Does it look like you would expect?

   c. Now filter this signal through **each** of your four filters using the Matlab "filter" command, and then plot each of the filtered signals in the frequency domain (using code similar to that in (b) above. Include each of these plots in your lab notebook, and comment on how well each filter performed.

Sign-off: *Show the TAs your code, and the stem plots and the frequency response of all four of your filters. Also show the TAs the frequency domain plots of the signal s1, and the frequency domain plots of the four filtered versions of s1.*

Task 3. Design and Test Ten Band Pass IIR Filters

In this task, you will design 10 band-pass IIR filters, each with a center frequency around one of the 10 player frequencies. The filters should each have a full bandwidth of 50 Hz (plus or minus 25 Hz from the center frequency), and the filters will have a total length of 11. Use the following 10 center frequencies for players 1-10 respectively:

1471, 1724, 2000, 2273, 2632, 2941, 3333, 3571, 3846, 4167 (all in Hz)

You will then test your bank of 10 filters on some sample data and see if you can identify a player hit.

Procedure: (Create a new .m file for this task, and clear your Matlab workspace!)

1. Design your ten IIR filters with appropriate center frequencies. When we say "design your filters", we mean determine the *a* and *b* coefficients for each filter. There will be 11 of each coefficient for each filter. Below we show you how to create the first filter at Player 1 frequency. (You should be clever in coding this into a loop to create all ten of your filter. You might think about producing arrays of *a* and *b* coefficients that are 11x10 to store the values for all of your filters. You will need to modify some of the variables in the code below to do this.)

```
f_players = [1471, 1724, 2000, 2273, 2632, 2941, 3333, 3571, 3846, 4167];
                  % Player center frequencies in Hz
f_s = 10000;      % sampling frequency (remember we have decimated!)
BW = 50;          % Full bandwidth of filters in Hz
f_lower = f_players(1) - BW/2;      % Lower corner frequency of BPF
f_upper = f_players(1) + BW/2;      % Upper corner frequency of BPF
[b1, a1] = butter(5, [f_lower*2/f_s f_upper*2/f_s]);
% IMPORTANT: "butter" takes frequencies in half cycles/sample, not Hz, hence
the division by f_s and multiplication by 2


f_axis = linspace(1000, 4500, 2000); % create a frequency axis for plotting
from 0 to 5 KHz
H1 = freqz(b1, a1, f_axis, f_s); % calculates the frequency response H at the
frequencies in f_axis
plot(f_axis, abs(H1));  % frequency response
```

2. Plot the frequency response of all 10 of your filters on a single graph (using "hold on") and include this in your lab write-up. Keep the vertical axis in absolute units (not dB), as shown in the code above. Do they look correct?

3. **IMPORTANT: Save your Matlab code so you can quickly generate the *a* and *b* coefficients for all 10 filters for 390 next semester!**

4. Load the provided sample data into Matlab, which includes 10 sample signals. Play around with filtering each of these 10 sample signals through each of your 10 filters (again using the Matlab "filter" command). For each sample signal, calculate the total energy (the sum of the absolute values squared for each signal) for each of the 10 filtered signals, and create a bar graph showing these energies. **Include this bar graph for two of the sample signals in your lab notebook, one of the easy ones and one of the hard ones.** Can you identify a player hit in the sample data? If so, what player in each case?

Sign-off: *Show the TAs your code, the frequency response of all 10 of your IIR bandpass filters, and the bar plots showing the filtered signal energies for each of the 10 filters for an easy signal and a hard signal.*