Digital Signal Processing

Lab 3 FIR Filters

Objectives:

- Use filter design and analysis tools to create FIR filters based on general filter specifications.
- Create a simulation with Simulink.

1. Background

Digital filters are used in a wide variety of signal processing applications, such as spectrum analysis, digital image processing, and pattern recognition. Digital filters eliminate a number of problems associated with their classical analog counterparts and thus are often used in place of analog filters. The most common digital filters belong to the class of discrete-time LTI (linear time invariant) systems, which are characterized by the properties of causality, recursibility, and stability. They can be characterized in the time domain by the unit-impulse response and in the z-transform domain by the transfer function. A unit-impulse response sequence of a causal LTI system can be either finite or infinite in duration. This property determines their classification as either a finite impulse response (FIR) or an infinite impulse response (IIR) systems. To illustrate this, consider the most general case of a discrete time LTI system with the input sequence denoted by $x(kT)$ and the resulting output sequence $y(kT)$ given by:

$$y(kT) = \sum_{m=0}^{M-1} b_m x((k-m)T) - \sum_{n=1}^{N-1} a_n y((k-n)T) \qquad (1)$$

The corresponding transfer function in the Z-domain is given by:

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \frac{\sum_{m=0}^{M-1} b_m z^{-m}}{1 + \sum_{n=1}^{N-1} a_n z^{-n}} \qquad (2)$$

If at least one denominator coefficient a_n is not zero, then system is recursive (its current output depends on previous output values), and as a result its impulse response is of infinite duration (IIR system). If all denominator coefficients are zero (polynomial of order 0), the corresponding system is non-recursive (FIR system), and its impulse response is of finite duration. The transfer function of Eq. (2) in this case becomes a polynomial of finite order M-1:

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \sum_{m=0}^{M-1} b_m z^{-m} \qquad (2)$$

The corresponding FIR difference equation in time domain is:

$$y(kT) = \sum_{m=0}^{M-1} b_m x((k-m)T) \tag{3}$$

As with analog filter design, the general shape of the frequency response is the main criteria in discrete filter design. Recall the frequency response for continuous-time systems was obtained by substituting evaluating the transfer function on the $j\omega$ axis, similarly for the discrete case the transfer function in z is evaluated over the unit circle. In this case substitute $z = \exp(j\omega/\omega_s)$ where ω_s is the sampling frequency in radians per second. Therefore, the frequency response of an FIR filter is given by:

$$\hat{H}(z)\Big|_{z=\exp\left(j\frac{\omega}{\omega_s}\right)} = \sum_{m=0}^{M-1} b_m \exp\left(-j\frac{m\omega}{\omega_s}\right) \tag{4}$$

Note that even though the time domain is discrete, the frequency response is continuous (defined for all ω); however, it is periodic with period ω_s due to the periodic behavior of the complex exponential and consistent with the concept of aliasing.

The design of digital filters involves determining the filter order (M) and computing the values of the coefficients (b_i's in the above equations) to achieve the desired filter response. The desired response can be specified in the frequency domain in terms of the magnitude response and/or the phase response. It can also be specified in terms the impulse response. Once filter coefficients are computed, the filter performance must be analyzed to determine the filter meets specification.

In this lab you will design FIR filters using 2 popular methods – impulse response windowing and the Parks-McClellan algorithm. See help files in Matlab for *fir1()* and *firpm*() for a more detailed explanation of the 2 methods and algorithms used in each of these approaches. For the analysis of the filter, see help on transfer function evaluations tools like *fft()* (this takes the DFT of the signal) and *freqz()* (this implements the frequency response computation of Eq. (4)). There are 2 useful scripts posted on the class web site, which are *winlook* and *firlook* that show examples of using these functions.

To better understand the behavior of the filters and the design process, it is useful to be have a working knowledge of the sinc function, as well as tapering window functions. These are used in the impulse response windowing method.

2. Pre-Laboratory Assignment

1. Sketch a rectangular function with height 1 and width A seconds that is symmetrically distributed about 0. Sketch its Fourier transform (sinc function) and label the axis to identify the width of its mainlobe (distance between the first null points on the frequency axis), height of the mainlobe, and height of the first sidelobe (absolute value of the peak between first and second null points on either the positive or negative frequency axis).

2. For the tapering window functions listed below, write a script to plot each window function on the same graph (use different line styles for each window function). Create another graph and plot its DFT magnitude on a linear scale, and finally create another graph and plot its DFT magnitude on a dB scale. Comment on how the general window

shape (steepness of taper) affects the spectral magnitude (impact on width of mainlobe and height of sidelobes)

a) Boxcar
b) Triangular
c) Hamming

3. The Kaiser window is also very popular because the degree of the taper can be adjusted parametrically with parameterβ. Repeat number 2 for a Kaiser window of length 128 and with β values of 2, 4, and 8. (see help *kaiser* in Matlab).
4. Become familiar with the template scripts *winlook.m* and *firlook.m*, posted on the course web site. Read through the comments so you know how they relate to the laboratory exercises. There is nothing the hand in on the prelab for this exercise.

3. Laboratory Assignment

(Assume sampling frequency of 44.1kHz, unless otherwise specified)

1. Design a 127th order linear-phase FIR low-pass filter with a cut-off at 12kHz using the windowing method (*fir1*). Design a filter using each of the following windows:

   a. rectangular (`boxcar`)
   b. triangular (`triang`)
   c. Hamming (`hamming`)

   On a single graph, plot the all the impulse responses together, and on another graph plot all the (frequency) magnitude responses together. In individual graphs plot the pole-zero locations of the 3 filters. **In the discussion section, compare the impact of the window shape on the filter characteristics (impulse response, Spectral magnitude, and zero placements).** Refer to the window plots of the prelab (in discussion use language such as mainlobe width, sidelobe, taper …) and make a general statement about the impact of window characteristics on the filter characteristics.

2. Repeat Exercise 1 for a 127th order linear-phase FIR low-pass filter using a Kaiser window with β = 2, 4, and 8 and pass-band cut off of 12kHz. Plot the impulse response, magnitude response, and zero-pole locations. **In the discussion section compare the characteristics of the magnitude response with each other and with the filters from the first exercise. Discuss how the trade-off between transition bandwidth and ripple varies with β.**

3. Repeat Exercise 2 for a 31^{at} order linear-phase FIR low-pass filter. Plot the impulse response, magnitude response, and zero-pole locations. **In the discussion section compare the characteristics of the magnitude response with those from the previous exercise. Discuss the impact of filter order.**

4. Design an optimal 31-order low-pass FIR filter using the Parks-McClellan algorithm (*firpm*). Use a pass-band cutoff of 10.5kHz and a stop-band cutoff of 14kHz. Plot the impulse response, magnitude response, and zero-pole locations. **In the discussion section compare the characteristics of the magnitude response to the other designs in prior exercises and comment on differences.**

5. Generate a square wave signal of 423.5 Hz and determine its approximate bandwidth (i.e. the frequency range that contains significant energy) by plotting its spectral magnitude. Use the Parks-McClellan algorithm (*firpm*) to design a 151-order FIR low-pass filter, such that frequencies starting with the 3rd harmonic (i.e. at 3*423.5=1270.5Hz) are suppressed by 10 dB or more with minimal impact on the fundamental and 2nd harmonic. Plot the magnitude response of the filter and also the filtered output. Produce a graph to demonstrates how well the filter achieved the specification. Use *soundsc* to play the sound before and after filtering and plot a few cycles of the input and output waveforms. **In the discussion section describe the differences (both aurally and graphically) that result from suppressing the higher harmonics.**

6. Repeat Exercise 5 with an order of 199. **In the discussion section describe the impact of filter order from differences observed in the magnitude responses.**

7. Generate the signal *x* using the following Matlab code (note the sampling rate in this problem is 8kHz):

```
>> Fs=8e3;  %Specify Sampling Frequency
>> Ts=1/Fs; %Sampling period.
>> Ns=512;  %Nr of time samples to be plotted.
>> t=[0:Ts:Ts*(Ns-1)];  %Make time array that contains Ns
>>              % elements t = [0, Ts, 2Ts, 3Ts,..., (Ns-1)Ts]
>> f1=500;
>> f2=1414;
>> f3=2000;
>> f4=3400;
>> x1=sin(2*pi*f1*t); %create sampled sinusoids at different
                     %frequencies
>> x2=sin(2*pi*f2*t);
>> x3=sin(2*pi*f3*t);
>> x4=sin(2*pi*f4*t);
>> x=x1+x2+x3+x4;  %Calculate samples for a 4-tone input signal
```

With the FIR windowing method, design a 16 order band-stop filter with stop-band between 1000 Hz and 2000 Hz using the hamming window. Plot the magnitude of the frequency response of the filter. **Is this what you expected?** Apply the filter to signal (*x*), and plot the spectrum magnitude of the input and output signal on the same figure. **Is this output as expected?** For spectrum plot, the following code is helpful:

```
>> Nfft = 2*length(x);
>>  xfftmag=(abs(fft(x,Nfft)));        % Compute spectrum of input
                                  % signal.
>> xfftmagh=xfftmag(1:floor(length(xfftmag)/2));
% Plot only the first half of FFT, since second half is
```

```
% mirror image the first half represents the positive
% frequencies from 0 to Fs/2, the Nyquist frequency.
f=Fs*[1:length(xfftmagh)]/(length(xfftmagh)*2);    %Make frequency
                    % array that varies from 0 Hz to Fs/2 Hz.
plot(f,xfftmagh,':k'); %Plot frequency spectrum of input signal
%(  Now you can hold the figure and repeat for the output signal,
%   using different line styles/colors - see "help plot" for details)
```

8. Using Simulink, create the block diagram shown in Figure 1.

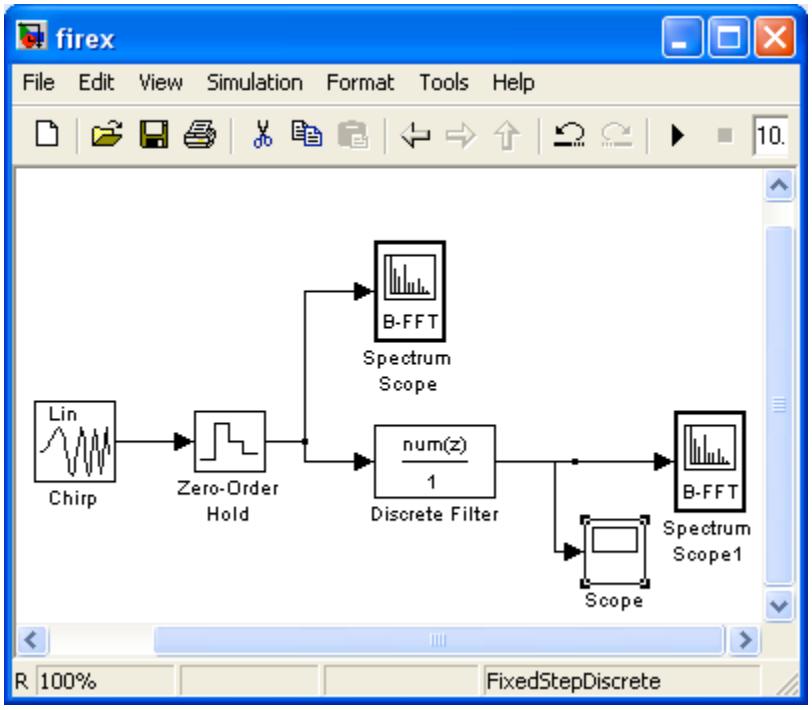
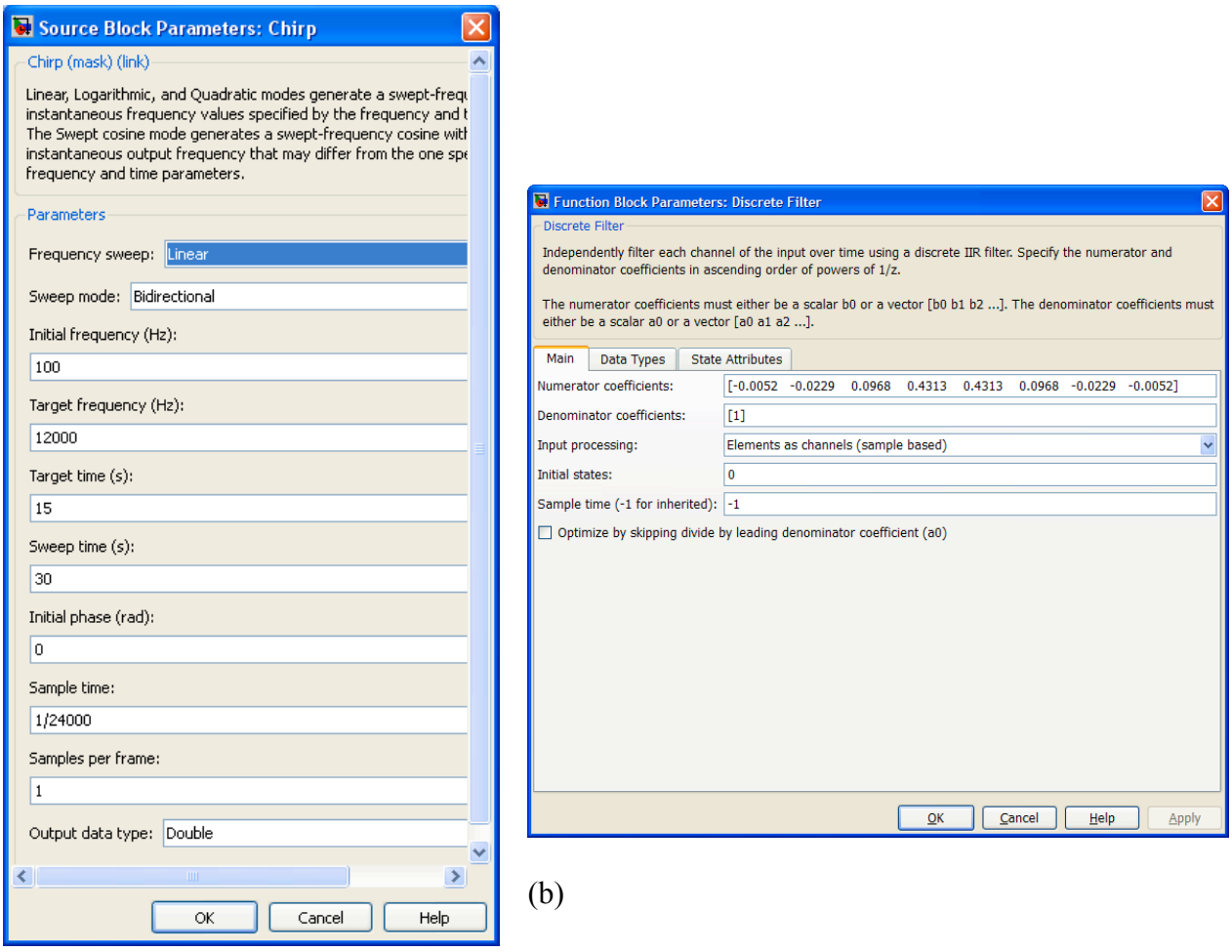


Figure 1. Simulink block diagram for FIR filter design

Generate a linear bidirectional sweep from 100Hz-12kHz. The signal generator for this can be found in the signal processing block under "Signal Processing Sources" and labeled as chirp. The signal generator block can be configured with the dialog box similar to that shown in Fig 2a. Set the parameters so the sweep will make 1 round trip in 30 seconds. Set the sample time to 1/24000 (i.e. sampling rate to twice the highest frequency). For the zero-order hold parameters (block found under Simulink blocks labeled "Discrete") simply have it inherit the sampling rate of the simulation. On the Spectrum Scopes select buffer input, and set the buffer size to 512 samples, buffer overlap to 256, likewise set the FFT lengths to 512 and the spectral averages to 1. You should come back to this Spectrum scope block during your experiment to change the display properties to better observe what is going on (i.e. change scales between magnitude and dB, auto scale …). The initial plots may be clipped but you can adjust the y-limits under the axis tab. Use the *fir1* command in Matlab to create coefficients for a 7^{th} order low-pass filter with a cut-off at 6kHz. Double click on the digital filter block (found under the signal processing block set) to indicate an FIR filter. The dialog box for this block is shown in Fig. 2b. Adjust the settings to get desired filter

and enter the coefficients (can cut and paste from Matlab workspace, but must keep the square brackets on the ends). Don't forget to set the simulation parameter to a fixed-time step, make the solver discrete (no continuous states), and have the simulation run for 30 seconds. Run the simulation and observe the dynamic spectrum of the output and input signal while the frequency is swept over time. Also observe the signal amplitude in the time scope and describe what you observe (you may have to use auto scale and disable the setting that limits the display to the last 5000 points for a better view of the time signal, this tends to be the default and you will need to see all points plotted over the simulation to draw the necessary conclusions). For the results section, copy (probably need to use a screen/window copy in windows for this) and paste the time scope in the results section.

Now repeat this simulation using the boxcar window (square window) for the *fir1*() algorithm. The default window is the Hamming window, which was use previously. What would you expect would be the main differences between these 2 windows? **In the discussion section explain how your observations either agree or disagree with what was expected, and agreement between the time and spectral scopes for the filter output. Also compare the 2 filters. Point out the key differences in their behavior (site differences from the figures in your results section) and provide and explanation for the differences.**



(b)

Figure 2. Dialog boxes for a) signal generator b) Example 7th order FIR filter.