Lab 3 FIR Filters

Objectives:

- Use filter design and analysis tools to create FIR filters based on general filter specifications.
- Implement a real-time graphic equalizer in LabVIEW

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, recursive operation, 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 nonzero, then system is recursive (its current output depends on previous output values), and as a result, its impulse response has an 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:

$$\hat{H}(z) = \frac{\hat{Y}(z)}{\hat{X}(z)} = \sum_{m=0}^{M-1} b_m z^{-m} \tag{3}$$

The corresponding FIR difference equation in time domain is:

$$y(kT) = \sum_{m=0}^{M-1} b_m x((k-m)T) \tag{4}$$

As with analog filter design, the shape of the magnitude frequency response is often the criteria in discrete filter design. Recall the frequency response for continuous-time systems was obtained by evaluating its 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:

$$\hat{H}(z)\Big|_{z=\exp\left(j\frac{\omega}{\omega_s}\right)} = \sum_{m=0}^{M-1} b_m \exp\left(-j\frac{m\omega}{\omega_s}\right) \tag{5}$$

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

The design of digital filters involves determining the filter order (*M*) and computing coefficient values (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 of the impulse response in the time domain. Once filter coefficients are computed, the filter performance is analyzed to verify the filter meets the specifications.

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. (5)). 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 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 *A* and width *T* seconds symmetric 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) in terms of *A* and *T*.

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 plot). Create another graph and plot their DFT magnitudes on a linear scale, and finally create another graph and plot their DFT magnitudes 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, 6, and 10. (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 to hand in on the prelab for this exercise.

3. Laboratory Assignment

FIR Filter Design Methods

(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 10kHz 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 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. **Show these to the TA and indicate the impact window shape has 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 using a Kaiser window with β = 2, 6, and 10 a cut off of 10kHz. Plot the impulse response, magnitude response, and zero-pole locations. **Show the TA and point out any differences with the windows in Exercise 1. How does the trade-off between transition bandwidth and ripple vary with β.**

3. Repeat Exercise 2 for a 31at order linear-phase FIR low-pass filter. Plot impulse response, magnitude response, and zero-pole locations. **Show the TA and describe 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 9kHz and a stop-band cutoff of 13kHz. Plot the impulse response, magnitude response, and zero-pole locations. Examine the plots to ensure the filter meets these specifications. Then generate a square wave signal of 528.2 Hz and determine its approximate bandwidth (i.e. the frequency range that contains significant energy) by using the *fft*() command and 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*528.2 =1584.6Hz) 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 demonstrate 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. **Present the final result for the TA (both aurally and graphically) and demonstrate the effect of suppressing the higher harmonics.**

LabVIEW Real-Time Graphic Equalizer

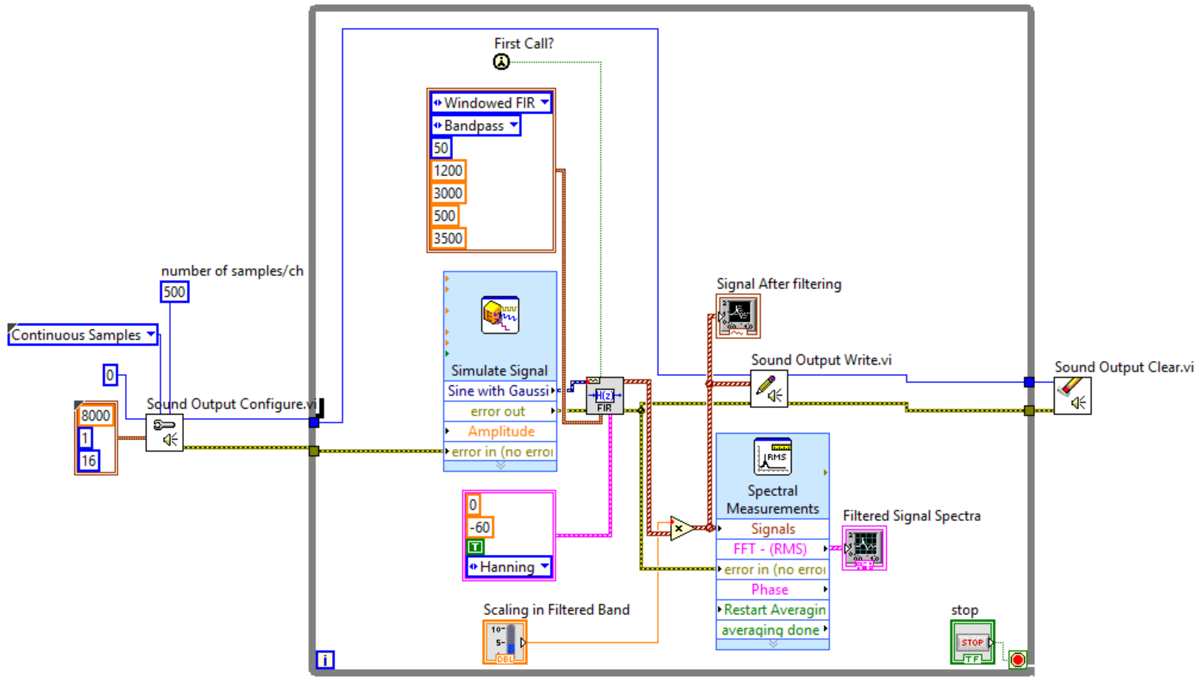
5. Create a real-time 3-band graphic equalizer for an audio waveform sampled at 8kHz. The program should display the average spectrum (PSD) and time waveform. Set the 3 bands to cover the following frequency ranges as shown in Table 1.

Table 1. Frequency ranges for 3 band equalizer

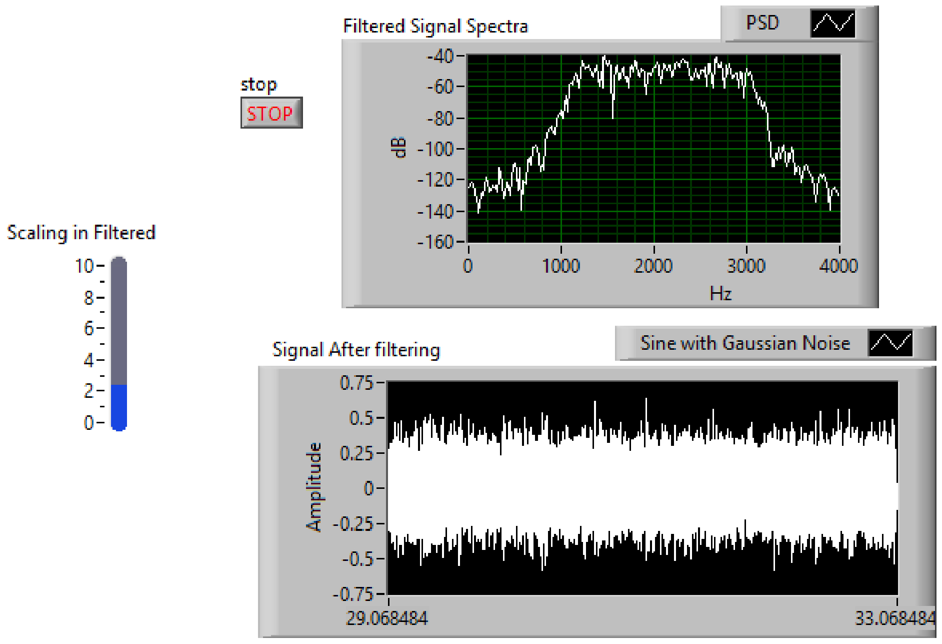
Low	25 to 450 Hz
Mid	450 to 1400 Hz
High	1400 to 3200 Hz

Demonstrate the filters and displays are working properly by first sending a simulated white noise signal through the system. When activating one filter (zero gain on the other 2 channels), the resulting spectrum should have the same shape as the desired spectrum for the filter. A sample VI is presented in Fig. 1. This uses LabVIEW's *Digital FIR Filter.vi* and Express VI to simulate white noise and show the spectrum. The dynamic data format from these express vi's will be consistent with the DAQ Assistant. So it should be minimal effort to replace the simulation vi with the actual DAQ acquiring real-time data from the microphone. As a first step build the system in Fig. 1. The express VI settings are shown in Fig. 2a for the *Simulate Signal* VI. In this case the sampling rate is set and the sample read/block size. While the VI is intended to simulate a sinusoid in additive white Gaussian noise (AWGN), the amplitude of the sine wave is set to 0 so only white noise is produced. Figure 2b shows the settings for the express vi that computes the PSD of the filtered signal. Note that different types of averaging and units are available. You can play around with these to see how they affect the spectral display. Once built, add 2 more FIR channels with variable gains, and set the pass-band parameters to meet the requirements. When completed, show the TA that energy in each of the filtered bands can be controlled in real-time with your gain controllers.

Now plug in the USB DAQ and connected the microphone to the high-impedance input of channel 1 on the amplifier (no phantom power). Connect the output of the amplifier channel to AI0 on the DAQ. Power up the amplifier with the gain set low to begin with (~ ¼ up from zero). Remove the *signal simulation* VI, and replace with the *DAQ assistant* VI. Configure for acquiring a signal as done for the simulations (*i.e.* same sampling rate and samples read values). Speak through the mic with various gains on the 3 bands. Make sure the time waveform appears and is consistent when you speak. You may need to increase the gain on the amp if the signal looks too weak. Demonstrate to the TA that it works.

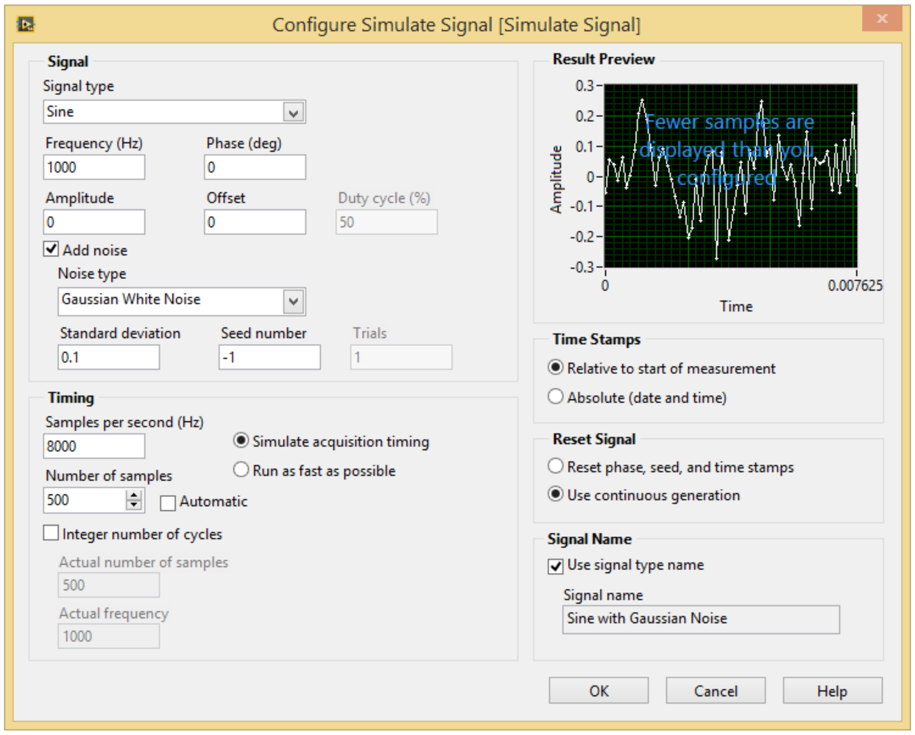


(a)



(b)

Figure 1. Sample VI implementing a Digital FIR filter VI and testing with white noise (a) Block diagram, (b) Front panel. FIR band-pass filter from 1200 Hz to 3000 Hz.



Configure Simulate Signal [Simulate Signal]

Signal

Signal type: Sine

Frequency (Hz): 1000 Phase (deg): 0

Amplitude: 0 Offset: 0 Duty cycle (%): 50

[✓ Add noise]

Noise type: Gaussian White Noise

Standard deviation: 0.1 Seed number: -1 Trials: 1

Timing

Samples per second (Hz): 8000

[✓ Simulate acquisition timing] Run as fast as possible

Number of samples: 500 Automatic

Integer number of cycles

Actual number of samples: 500

Actual frequency: 1000

Result Preview

Time Stamps

[✓ Relative to start of measurement] Absolute (date and time)

Reset Signal

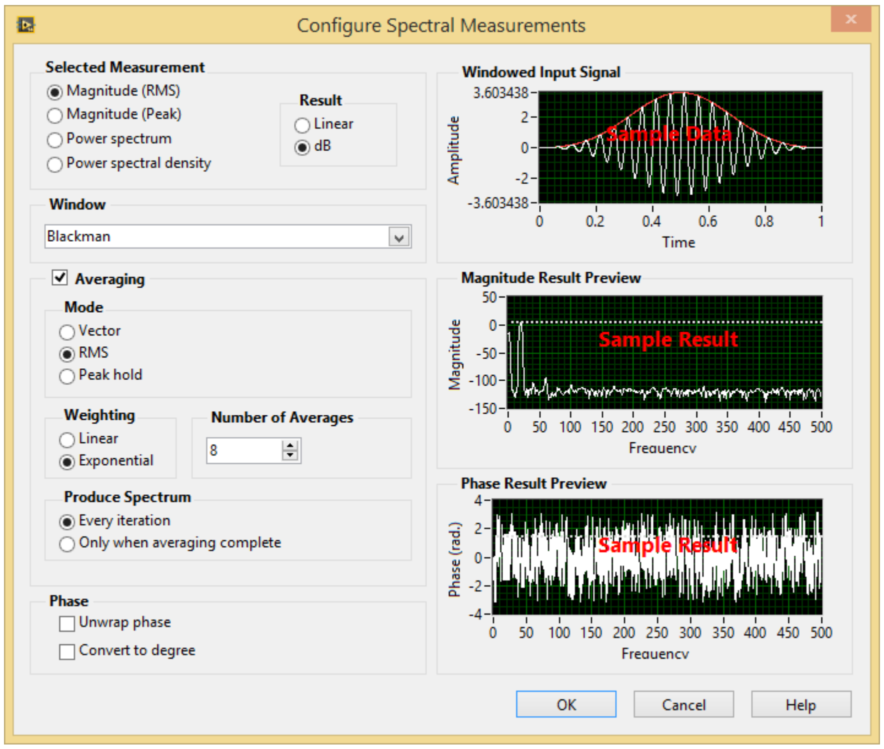
Reset phase, seed, and time stamps [✓ Use continuous generation]

Signal Name

[✓ Use signal type name]

Signal name: Sine with Gaussian Noise

(a)



Configure Spectral Measurements

Selected Measurement

[✓ Magnitude (RMS)] Magnitude (Peak) Power spectrum Power spectral density

Result

Linear [✓ dB]

Window

Blackman

[✓ Averaging]

Mode

Vector [✓ RMS] Peak hold

Weighting

Linear [✓ Exponential]

Number of Averages: 8

Produce Spectrum

[✓ Every iteration] Only when averaging complete

Phase

Unwrap phase

Convert to degree

Windowed Input Signal

Magnitude Result Preview

Phase Result Preview

(b)

Figure 2. Properties window with setting for the (a) *Simulate Signal* Express VI, and (b) for *Spectral Measurements* Express VI.