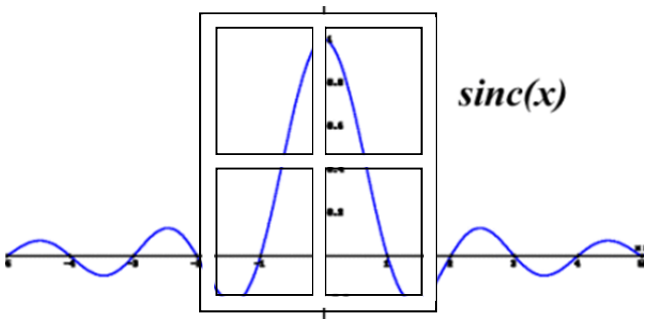
EE 419 - Project 7
FIR Filter Design by Windowing



REMINDERS:
1) When you are finished, be sure to always copy your m-files and "C" source code files to a place that you can retrieve them again in the future.
2) Always read the "reminders" just in case something new shows up. Make sure especially to read the *fine print*. You never know what might be lurking there.

1) FIR Filter Design by Windowing Method - Manual Computations

Design by hand calculations an FIR low pass filter via the method of windowing. The filter should have a -6 dB cutoff frequency, f_c=3200 Hz (for a sampling rate of f_s=16 KHz). Use a filter length M=9 and a **Hamming** window.

a) First compute the filter coefficients using a calculator or spreadsheet, based on an ideal low-pass filter impulse response. (Do not rescale your filter coefficients to achieve unity-gain). **Show your computation results for the ideal LP filter impulse response values, the window sample values, and the resulting Difference Equation from the windowed impulse response in your report**.

b) Using your Matlab filter analysis program, **plot the frequency response (linear scale magnitude and phase vs. digital freq.) and pole/zero diagram** for your filter. Provide these with your report.

c) **Based on the pole/zero diagram** and pole/zero locations (complex values), **explain** how can you tell that this filter is:
   - **i. FIR**
   - ii. **Linear Phase** (report the zero locations in polar form: |magnitude| ∠ angle and explain how these confirm the linear phase behavior.)
   - **iii. Low-Pass**

d) From the Matlab filter analysis plots, report the following **actual performance measures** for your filter:
   - i. Magnitude response at DC.
   - ii. Stop-band attenuation (in dB) at the Nyquist Frequency
   - iii. Magnitude response at the designed Cutoff Frequency

2) FIR Filter Design by Windowing - Using Matlab

Using Matlab, create the following programs to design and analyze FIR filters using the Method of Windowing:

a) Create a function m-file that computes and returns the impulse response values for an ideal low-pass filter that has been windowed. The length of the filter (M), the -6 dB cutoff frequency (Fc), and the window function values should be specified by parameters that are passed to the function.

Your function should have the following format:

```
function hn_lp = FIR_Filter_By_Window (M,Fc,window)
        M = the filter length (odd)
        Fc = filter cutoff digital frequency (-6dB) (cycles/sample)
        window = the Matlab window function values to multiply h[n] by
        hn_lp = windowed impulse response values for the Low-pass FIR filter
```

In order to use your choice of window function, you can incorporate the calling of a valid Matlab window function right in the function call to your program. Matlab provides a number of standard windowing functions (as listed below). The window length is a parameter passed to these window functions. The returned window sample values are a column array; so you typically need to transpose these to a row vector using: .'

rectwin( )	- Rectangular window.
triang( )	- Triangular window.
bartlett( )	- Bartlett window (Triangle Window with '0's at both ends)
hann( )	- von Hann window (Hanning Window with '0's at both ends)
hanning( )	- Hanning window (With non-zero end samples).
hamming ( )	- Hamming window.
blackman( )	- Blackman window.
kaiser( )	- Kaiser window (with b parameter)
tukeywin( )	- Tukey window.
Barthannwin( )	- Modified Bartlett-Hanning window.
bohmanwin( )	- Bohman window.
chebwin( )	- Chebyshev window.
flattopwin( )	- Flat Top window.
gausswin( )	- Gaussian window.
blackmanharris( )	- Minimum 4-term Blackman-Harris window
nuttallwin( )	- Nuttall defined minimum 4-term Blackman-Harris window.
parzenwin( )	- Parzen (de la Valle-Poussin) window.

For example, to use a rectangular window for a length 5 filter (M=5) with digital cutoff frequency Fc=0.3 cyc./sample, you can call your program as follows:

```
hn_lp = FIR_Filter_By_Window(5,0.3,rectwin(5));
```

This way, the correct values for the window (with the correct length) are provided to your program. It is up to your m-file to transpose and multiply these values by the ideal LPF impulse response.

Do not scale your impulse response for unity gain at DC (as any ripple in the frequency response will confuse this scaling).

For your report, test your program using the same length=9 lowpass filter with an analog -6dB cutoff frequency of 3200 Hz for a sampling rate of 16 KHz; created with a Hamming window. Report the results using your Matlab program from Part 2a, and compare them to your hand calculations from Part 1.

Turn in:

- **A listing of your Matlab function m-file**
- **The exact function call used for your test case given.**
- **The filter coefficients determined by your program for the test case given.**

b) Create a second Matlab function to transform a low-pass filter impulse response into a high-pass filter impulse response (with the same length and same cutoff frequency). Do not scale your impulse responses for unity passband gain.

```
function hnhp = transform_lp2hp(hnlp)
  hnlp - low-pass filter impulse response values (any odd length)
  hnhp - high-pass filter impulse response, with same cutoff freq Fc
```

Test your program by transforming the impulse response from the M=9 Hamming windowed low-pass filter with fc= 3200 Hz for a sampling rate of 16 KHz designed using your program in Part 2a, into a high-pass filter with the same cutoff frequency.

Provide the following for your test case:

- **A listing of your Matlab function m-file**
- **The exact function call used for transforming the test case low-pass filter to a high-pass filter.**
- **Filter coefficients for the high-pass filter determined by your function**
- **Pole/Zero Diagram of the high-pass filter**
- **Frequency response plot of the high-pass filter (linear magnitude scale)**

3) Implementing an FIR Filter Designed by Windowing in Software (C Language)

Implement a linear-phase FIR **low pass filter** of arbitrary length designed via the method of windowing using the Dev C++ compiler. The Dev C++ compiler can be downloaded from the course PolyLearn site, if needed. Use the WAV project template provided with the Dev C++ program: C://Dev-C++/CP/CP-Projects/WAV Project/WAV Project.dev

Write your program to use an **arbitrary (odd) value of filter length M** and a specified **-6 dB cutoff frequency;** both of which are #defined in the code. Have the program compute the filter coefficients based on the ideal LPF impulse response (appropriately shifted and truncated) multiplied by a **Hamming window** function to improve the stop-band attenuation. (Do not rescale your filter coefficients to achieve a unity-gain passband.)

Use an efficient symmetric FIR filter implementation as a basis for your program to minimize the computations required to implement your filter. Use **floating-point arithmetic and variables** to simplify your software implementation (rather than trying to implement a scaled integer version).

a) Implement your filter on the Dev C++ Compiler. **Turn in a listing of your filter "C" code**. Your C program should incorporate ALL of the following features:
   - Filter coefficients computed and initialized by the C program, based on **an ideal low-pass filter** impulse response, with a **Hamming window** function applied.
   - **Arbitrary Filter Length (M), set by a #define compiler directive at the top of the code.** This way, any filter length can be implemented with only one number change to the program. Your program should then compute the ideal lowpass impulse response and window values needed for that particular length filter. **Only odd filter length (M) values** need to be handled by your program, since we are creating a Type 1 linear phase filter. This will also probably require you to use **arrays** for your Bk coefficients and for keeping track of the previous input sample values x[n-__].
   - **Arbitrary cut-off frequency (Fc)** for the design, set by another #define
   - **Minimized # of multiplications** by taking advantage of the symmetric Bk's.

b) Test and verify that your program works using the same length M=9 lowpass filter design with fc= 3200 Hz, created previously. Measure and report the magnitude response for your software filter implementation using the SIPTool to generate a frequency response (Spectrum-All) plot using a noise input file noise.wav (included with the and the chirp_0_8K.wav file provided on PolyLearn:
   a. First, observe the "Wav Input" and then the "Spectrum-Short" plots in the SIPTool, as you *playback* (left-click on the plot in the SIPTool) the
      - unfiltered noise input signal
      - filtered noise output signal
      - unfiltered chirp input signal
      - filtered chirp output signal
   b. Save an image (and **include a copy in your report**) of **the SIPTool frequency spectrum plots** ("Spectrum-All" plots) for the unfiltered noise input signal and filtered noise output signal, and the unfiltered chirp input signal and filtered chirp output signal.
   c. **Estimate the -6dB cutoff frequency** from the filtered noise output plot (show the point on the SIPTool frequency spectrum plot that you estimated to be the -6dB point, and show the corresponding amplitude and frequency axis values for that point on the same graph.)
   d. **Describe any audible differences** you can hear during playback with the SIPTool, between the unfiltered chirp input signal and the filtered output chirp signal. Compare also the unfiltered noise input signal and the filtered noise output signal.

c) Increase the filter length M to 31 in your C program. Test and verify that the frequency response of the C code implementation matches the expected response found from Matlab analysis of the same filtered designed by your Matlab FIR_Filter_By_Window() function.

d) **Report the results for your M=31 filter** running on Dev C++ for**:**
   a. Expected Frequency response plot (from Matlab)
   b. SIPTool actual Magnitude response (Spectrum All) using a noise input
   c. -6dB frequency (estimated from SIPTool spectrum)
   d. Compare the M=31 filter frequency response to the M=9 filter's response. What is the most significant change in the performance of the filter between these two designs?

4) At Last(!), a "Real" FIR Filter Design Problem Using Windowing

Using the Matlab tools you have developed, design an FIR **low-pass** filter via the method of windowing. The filter should meet the following specifications:

Sampling Rate= 44.1 KHz (CD Audio)
-6 dB Cutoff Frequency = 3750 Hz
Transition Band Widths < 2500 Hz
Stopband Attenuations: > 60 dB
No Phase Distortion
Filter Length: As short as possible

a) Indicate your initial design parameter selections, and show how you arrived at each one:
   i) Window selected
   ii) Filter Length (M)
   iii) Low-Pass Filter prototype Cutoff Frequency (Fc)

b) Generate the Low-Pass Filter windowed impulse response, using your Matlab programs. Analyze the resulting Low-Pass filter (using your Matlab analysis programs), and determine from the plots the following performance metrics:
   i) Minimum Stopband Attenuation
   ii) Stopband Edge Frequency Fs
   - Frequency where |H(F)| 1st crosses the level of minimum allowed stopband attenuation of 60 dB
   iii) Passband Edge Frequency Fp - Frequency where |H(F)| crosses (1-δ_p), assuming $\delta_p=\delta_s=10^{-As/20}$ (using A_s= 60 dB allowed)
   iv) Transition Band Width

   Provide the results of your measurements for the above metrics. Also provide a **frequency response plot** of your LP filter, **using a dB scale** for the magnitude.

c) If your initial design exceeds the provided specifications, adjust your design parameters to try to minimize the filter length while still meeting all the specifications (within $\pm$2%). Provide the following results for your final Low-Pass filter design:
   i) Window selected
   ii) Filter Length (M)
   iii) Low-Pass Filter prototype Cutoff Frequency (Fc)
   iv) Minimum Stopband Attenuation
   v) Stopband Edge Frequency Fs (same definition as above)
   vi) Passband Edge Frequency Fp (same definition)
   vii) Transition Band Width

   Provide the results of your measurements for the above metrics, taken from you Matlab filter analysis programs and plots. Also provide another **frequency response plot** of your final LP filter, **using a dB scale** for the magnitude.