

ECE 320 Signals and Systems II

Matlab Project 3: Discrete-Time Filter Design

Issued: Monday, November 27, 2000

Due: Wednesday, December 6, 2000

The analog signal $x(t)$ consists of a sinusoidal component and a random noise component. The frequency of the sinusoidal component is unknown, but it is guaranteed to be less than 100 Hz. Noise is concentrated at frequencies higher than 100 Hz. After anti-alias filtering with a cutoff of 500 Hz, this signal is sampled at a rate of 1000 Hz and stored in the vector $x[n]$.

The goal of this exercise is to design and implement a lowpass filter to remove noise from $x(t)$. As a part of this process, you will experiment with several different FIR and IIR discrete-time filter design techniques. The lowpass filters you design should meet the follow specifications:

- The filter must pass all frequencies up to 100 Hz. Allowable amplitude distortion (ripple) is $\pm 2\%$, i.e., $\delta_1 = 0.02$.
- Above 175 Hz, the filter must have an attenuation of at least 40 dB, i.e., $20 * \log_{10}(\delta_2) = -40$.

Part I below leads you through the design process for four different filters (two FIR and two IIR). In the second part, you will implement these filters in Matlab and use them to filter a “mystery” signal that is downloadable from the course website.

I. Lowpass Filter Design

A. Determine and sketch the specifications of the discrete-time filter.

B. FIR Designs: Parks-McClellan vs. Window Method

In parts (i) and (ii) below, you will design two different FIR lowpass filters using the optimal Parks-McClellan method and window method. The writeups for each filter should include the names of the Matlab functions used in the design, the values of the input parameters to those functions, and plots of the magnitude and phase characteristics of the filter. In particular, there should be a zoomed plot of both passband and stopband, clearly showing amount of ripple and confirming that the design meets the specification at the pass/stop band edges. *It is important that the magnitude and phase characteristics are plotted as a function of continuous-time frequency (in Hz) so that proper comparisons can be made to the design specifications (which are also given in Hz). Note that the `freqz` command will return a frequency vector in Hz if you specify the sampling frequency.*

- (i) Use the Parks-McClellan algorithm to design the minimum-length filter that meets the specifications. You may want to use the approximate filter length formula provided in class as a starting point for your design.
- (ii) Using same length (M) obtained in part (i), design an FIR lowpass filter using the window method with a Hamming window. Does this filter meet the specifications? If not, find the smallest length Hamming window design that has the desired characteristics.

C. IIR Designs via the Bilinear Transformation

In this part, you will use the Bilinear Transformation on a Butterworth filter and an elliptic filter to obtain two IIR lowpass filters that meet the specifications provided above. Again, the writeups for

each filter should include the names of the Matlab functions used in the design, the values of the input parameters to those functions, and plots of the magnitude and phase characteristics of the filter (including zoomed pass/stopbands). *It is important that the magnitude and phase characteristics are plotted as a function of continuous-time frequency (in Hz) so that proper comparisons can be made to the design specifications (which are also given in Hz). Note that the `freqz` command will return a frequency vector in Hz if you specify the sampling frequency.*

- (i) Using the Matlab function `butter`, design the minimum order Butterworth filter that meets the given amplitude specifications. You may use the command `buttord` to compute the required filter order and other input parameters to the `butter` function.
- (ii) Using the Matlab function `ellip`, design the minimum order elliptic filter that meets the given amplitude specifications. You may use the command `ellipord` to compute the required filter order and other input parameters to the `ellip` function.

Note: Both the IIR filter design programs return filters with a maximum amplitude of 1. To obtain a passband ripple of 1 ± 0.02 , you will need to scale the filter specifications and then rescale the coefficients you obtain (as discussed in class).

D. Comparisons

For the following comparisons, assume that the FIR filters are implemented in the direct form and that the elliptic filter is implemented as a cascade of first- and second-order sections. It may be helpful to sketch these forms before answering the questions below.

- (i) Determine the number of storage registers required to implement each of your designs.
- (ii) Determine the number of multiplies and adds required per output point for each of the filters. You will need to sketch the appropriate block diagram (or equivalently look at the difference equation) for each filter to determine these numbers.
- (iii) Briefly discuss the relative merits of each of your designs, e.g., the computational complexity (number of multiplies/adds), storage requirements, maximum ripple, etc.

II. Removing Noise From a Sinusoidal Signal

In this part, you will test that your lowpass filters are working properly and use them to remove noise from a “mystery” sinusoidal signal.

A. Test Signals

Create several sinusoidal test signals using the `samp_sine` function that you developed for Project 2. Process these signals using the lowpass filters you designed in part I. Use Matlab’s `filter` command to implement the processing. Verify that your filters are passing sinusoids with frequencies less than 100 Hz and attenuating sinusoids with frequencies higher than 175 Hz. Include two or three test plots, along with a short explanation of your findings, in your writeup.

B. Mystery Signal

Download the file `mystery.bin` from the course website. Change the extension from `.bin` to `.mat`. Type `load mystery` to load the contents of this binary file into your workspace. The noisy signal is contained in the vector `x` and the time indices corresponding to the samples are contained in the vector `t`. Plot the noisy signal as a function of time to see what it looks like.

- (i) Process the noisy signal using each of the filters designed in Part I. Again, use Matlab's `filter` command.
- (ii) Plot the processed output for each of the filters.
- (iii) From your time series plots, estimate the amplitude of the mystery sinusoid.
- (iv) Compute the frequency response of the filtered signals and use that to estimate the frequency of the mystery sinusoid. Draw on the experience you gained in Project 2 to explain the resolution of your frequency estimate.
- (v) Compare/contrast the results for each of the 4 filters, i.e., how are the filtered outputs similar? how are they different? Can you explain these differences?

Hints:

You will probably want to make use of the following Matlab commands:

<code>remez</code>	Parks-McClellan optimal equiripple FIR filter design
<code>fir1</code>	Filter design using the window method
<code>hamming</code>	Hamming window
<code>ellip</code>	Elliptic (Cauer) digital/analog filter design
<code>ellipord</code>	Elliptic filter order selection
<code>butter</code>	Butterworth digital/analog filter design
<code>buttord</code>	Butterworth filter order selection
<code>freqz</code>	Frequency response
<code>type</code>	Types out a listing of an M-file
<code>load</code>	Loads workspace variables from disk