

Matlab Project III

Fall 2004

Issued: Tuesday, November 23, 2004**Due:** Wednesday, December 9, 2004

The goal of this exercise is to design and implement a digital lowpass filter to remove high-frequency noise from an audio signal. You will design both FIR and IIR filters.

Each student must do her or his own work on this project, however you may ask other students for advice. As stated in the guidelines given in the ECE 410 course information packet, you should identify any students you collaborate with. Your writeup must include all of the analytical (*i.e.*, pencil/paper) work, Matlab plots and code, and relevant explanations. A list of guidelines for preparing the writeup of this project are given below.

- The report must be *neatly* handwritten or typed, and all pages must be numbered.
- All plots must be neatly annotated with x-axis and y-axis labels and a title. Any graph not labeled will be considered not handed in.
- I will not spend time trying to figure out which graphs are for which problems. When referring to plots in the text, I recommend doing at least one of the following:
 - use figure numbers, e.g., “Figure 1 is a plot of the signal $x[n]$.”
 - cite the page number they are on, e.g., “The figure at the top of page 4 is a plot of $x[n]$.”
- All Matlab code must be well-documented and should be included in an appendix at the end of the report.
- Please **do not** include answers to questions or other descriptions within your Matlab code. You must write a report separate from the Matlab code itself. You are encouraged to include comments in your Matlab code, but I will not consider the comments part of your official report.

Filtering Problem Definition:

An audio signal sampled at a rate of 8,192 Hz has been corrupted with highpass noise. The corrupted signal is stored in the file `noisysig.mat` (available from the course website). The file contains two variables: the sampling frequency `fs` and the noisy signal `xnoisy`. In this exercise you will design FIR and IIR lowpass filters to remove the noise so that you can listen to the signal. Your lowpass designs must meet the following specifications:

- The filter must pass all frequencies up to 2,000 Hz. The allowable amplitude distortion (ripple) in the passband is $\pm 2\%$, *i.e.*, $\delta_1 = 0.02$.
- Above 2,500 Hz, the filter must have an attenuation of at least 40 dB, *i.e.*, $20 \log_{10}(\delta_2) = -40$.

A. Determine and sketch the specifications of the discrete-time lowpass filter. In other words sketch the tolerances for $H(e^{j\omega})$ for $-\pi \leq \omega \leq +\pi$.

B. Window Design

Use the Matlab command `fir1` to design lowpass filters using the window method. Try to design the filter using each of the five standard windows (rectangular, Bartlett, Hanning, Hamming, and Blackman).

For each window, you should design the **minimum** length filter that meets the specifications. It may not be possible to achieve the specifications with all of the windows. If certain windows do not work, explain why.

The writeups for each filter that does meet specifications 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.*

C. Bilinear Transformation Design

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). Again, `freqz` will be helpful.*

- 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.

Notes on IIR design:

- The `buttord` and `ellipord` require you to specify the passband and stopband ripples in dB. These can be specified as follows:
 - $R_p = -20 \log_{10}(1 - \delta_1)$
 - $R_s = -20 \log_{10}(\delta_2)$where δ_1 and δ_2 are the passband and stopband ripples, respectively.
- Matlab's IIR filter design programs return filters with a maximum amplitude of 1. To obtain a passband ripple of 1 ± 0.02 , you may need to scale the elliptic filter specifications and then rescale the coefficients you obtain. See Basic Problem 7.3a in your textbook for help with doing this.

D. Application: Filtering the noisy audio signal.

- i. Load the noisy signal by typing `load noisysig.mat`. Play the noisy signal using the command `soundsc(xnoisy,fs)`, where `xnoisy` is the signal vector and `fs` is the sampling frequency in Hz.
- ii. Filter the signal with each of the filters you designed in Parts B and C. Use the `filter` command to implement the filtering operation. Can you identify the signal? What does it sound like?
- iii. How do your filters perform? Do they remove all of the noise? Does one filter perform better than the others or do they all produce identical results? Write up your comparison of the filter performance. If you notice differences, try to explain them based on what you know about the impulse and frequency responses of your filters.
- iv. Please include plots of the noisy signal and the filtered signals in your report. You should plot them versus an appropriate time vector (remember the signals are sampled at a rate of `fs` Hz).

E. Computational Complexity Comparisons

For the following comparisons, assume that the FIR filters are implemented in the direct form and that the elliptic and Butterworth filters are 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.

F. Summary Memo

Write a short memo that discusses the relative merits of each of your lowpass filter designs. Compare the computational complexity, storage requirements, maximum ripple, stopband attenuation, phase characteristics, etc. for each of the filters. Also discuss how well they performed for the noisy audio signal given in this project. Your memo should be concise, but complete. It should contain all the information that a busy boss would need to decide which filter to use in the final design.

Hints:

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

<code>soundsc</code>	Autoscales and plays a vector through the speakers
<code>filter</code>	Filters a data vector (using user-specified a and b coefficients)
<code>fir1</code>	Filter design using the window method
<code>boxcar</code>	Rectangular window
<code>bartlett</code>	Bartlett window
<code>hanning</code>	Hanning window
<code>hamming</code>	Hamming window
<code>blackman</code>	Blackman 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>	Computes the frequency response