ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
School of Computer and Communication Sciences

Handout 17 Signal Processing for Communications
Graded Homework 8. Due: May 4 April 27, 2009

Matlab has comprehensive documentation. Don't forget to use it whenever you need help!

PROBLEM 1. In this problem, we are going to design FIR filters using the windowing method. Consider the ideal low-pass filter with cut-off frequency $\omega_c \in [0, \pi]$,

$$H(e^{i\omega}) = \begin{cases} 1 & |\omega| < \omega_c \\ 0 & \text{else} \end{cases}.$$

1. What is the impulse response $h[n]$ of this filter?
2. What is the best FIR approximation of length $2K+1$ for $H(e^{j\omega})$? That is, an FIR filter $h_K[n]$ for which $h_K[n] = 0$ for $|n| > K$, and minimizes $\|H(e^{j\omega}) - H_K(e^{j\omega})\|_2$.
3. Write a Matlab m-file `rectangular_window.m` that produces a vector of length $2K+1$ that holds the coefficients of the filter $h_K[n]$ and then takes the Fourier transform of the produced vector. (You can either write your own code to implement Fourier transform or you can use the `fft` command. Make sure that you compensate the shift in frequency if you use `fft` command).
4. Plot the frequency response of your filter for $\omega_c = \pi/4$, $K = 10$, 50, 100, 1000 on top of each other (use `plot`, `hold on`, `plot` with different colors and line styles (don't forget to use Matlab help). What happens as K increases in the passband, stopband and transition band? Can we get rid of the peaks near the transition region? How well is the approximation in comparison to the ideal filter? What happens at the cut-off frequency? How can you explain this phenomenon?

   We will now use different windows to design the low-pass filter. Recall that

   $$\hat{h}[n] = h[n]w[n],$$

   where $h[n]$ is the ideal impulse response and $w[n]$ is the window.
5. Plot the rectangular, triangular, hamming and blackman windows on top of each other. Set the window length to 25.
6. Write Matlab m-files (`triangular_window.m`, `hamming_window.m` and `blackman_window.m`) to compute the frequency response $\hat{H}(e^{j\omega})$.
7. Plot the frequency response of all of the above filters on top of each other in normal scale and in *log* scale ($20\log_{10}(\hat{H})$) to compare different windows. Comment on the results. What are the effects of different windows?

PROBLEM 2.

1. Download the file `corrupt_1.wav` from the course website (under 'Additional Reading Material'). Read the samples of the .wav file using the Matlab function `wavread`. Play the file using `soundsc`. You should be able to notice a high frequency noise on top of the original music.

   Now plot the DFT magnitude of the file, using `fft` (don't forget to shift the frequencies in order to better view the response: use `fftshift`). In the DFT plot, indicate the parts corresponding to the original signal and the noise. What is the bandwidth of the original music (that is, the bandwidth where the DFT magnitude is not negligibly small)? What about the bandwidth of the noise? What kind of filter would you need to design in order to remove the noise from the original signal? What is an acceptable passband for this filter?

2. You will implement the (ideal) filter in the previous part using windowing. Use your favorite window to truncate the ideal filter. In order to be able to keep the transition band to a minimum, you might have to use a wide window (possibly in the order of hundreds of taps). Plot the DFT of the resulting filter. Remove the noise from the file using your filter (either in the time domain using the convolution function `conv` or in the frequency domain by simple multiplication). Plot the DFT of the resulting file. Play the file using `soundsc`.

3. We will now implement the same filter using Parks-McClellan algorithm. Matlab has a built-in function, `firpm`, that does this. The usage is as follows:

   ```
   >>[hpm, err] = firpm(40, [0 0.6 0.75 1], [0 0 1 1], [10 1]);
   ```

   The first argument of `firpm` specifies that the resulting filter should be of length 41. The second argument says that there is one frequency band between 0 and 0.6π and a second band between 0.75π and π. The third argument says that the filter frequency response should be close to 0 in the first frequency band and close to 1 in the second frequency band. Between the two bands, we do not care. The fourth argument says that the precision in the first frequency band is 10 times more important than in the second frequency band.

   Use the function `firpm` to construct a filter that gets rid of the noise in the original file. The return value 'err' gives you the error (how far the filter is from the ideal filter). Play the filtered file. You can play a bit with the filter-length parameter to find a good filter. Plot the DFT of your filter, and the resulting sound after filtering.

4. Which of the above methods is better in terms of the resulting sound file? Briefly explain the differences (e.g., which methods is good for what error criterion etc.) in the above methods, and comment on the causes of the possibly different results you obtained above. Comparing the DFTs of the filters you used above might be of help.

5. We are now interested in recovering the noise. How can you do this using only the original and the filtered files? What kind of filter does this operation correspond to?

6. The noise, $y[n]$, you extracted above was created as follows:

   $$y[n] = w[n]\cos(0.2\pi n),$$

   where $w[n]$ is the samples of an audio file. You will recover $w[n]$ from $y[n]$ as follows: First shift $y[n]$ to baseband. Plot the resulting signal's DFT. You should be able to

observe unwanted frequency content. Now filter these out to obtain $w[n]$. Play the result. What do you hear?