Contents

4. Integral in MATLAB
   a. Example: integration in MATLAB
   b. Example: convolution of two signals using ‘trapz’
   c. Example: convolution of two signals using ‘conv’
   d. Assignment 4: Convolution

5. Exponential Fourier Series
   a. Exponential Fourier series coefficients
   b. Signal reconstruction using the exponential Fourier series coefficients
   c. Side note: summation of vectors
   d. Assignment 5: exponential Fourier series in MATLAB

6. Introduction to Simulink
   a. Example: Fourier series coefficient in Simulink
   b. Example: square wave signal reconstruction in Simulink
   c. Assignment 6: exponential Fourier series coefficients in Simulink

7. Fourier Transform
   a. Example: Fourier transform of a rectangular function
   b. Assignment 7: Fourier transform

8. Audio Signals in MATLAB
   a. Example: audio signal plotting in MATLAB
   b. Assignment 8: convolution and the Fourier analysis

9. Filters in LTI Systems
   a. Example: lowpass filtering of a noisy EKG signal
   b. Assignment 9: Lowpass, Highpass, and Bandstop Filters

10. Audio Signals
    a. Example: audio signal analysis in MATLAB
    b. Assignment 10: Notch Filter for Interference Removal

11. Laplace Transform
    a. Example: Laplace transform visualization in MATLAB
    b. Assignment 11: Visualizing the Laplace Transform

12. Image Signals
    a. Example: image signal analysis in MATLAB
    b. Assignment 12: Image Signal Enhancement and Resizing
9. Filters in LTI Systems

There are several types of practical filters which can be generally categorized in four standard kinds: ‘lowpass’, ‘highpass’, ‘bandpass’, and ‘bandstop’. The lowpass filter (LPF) is defined as a filter which preserves the low frequency components while attenuating the high frequency components. In contrast, a highpass filter (HPF) keeps the high frequency components while attenuating the low frequency components. Figure 9-1 shows the frequency response of ideal lowpass and highpass filters, respectively.

Figure 9-1. (a) ideal lowpass filter, (b) ideal highpass filter

As can be seen, the left side filter passes only the frequencies lower than \( \Omega_c \) while filtering out the frequencies higher than \( \Omega_c \). Therefore, it is called a lowpass filter. The right side filter behaves oppositely, called a highpass filter. The frequency response of ideal bandpass and bandstop filters are also displayed in Figure 9-2.

Figure 9-2. (a) ideal bandpass filter, (b) ideal bandstop filter

It is clear that the left side filter, bandpass, passes the middle-band frequencies between \( \Omega_{c1} \) and \( \Omega_{c2} \), while the bandstop filter functions inversely by removing the middle-band frequency components of the input signal.

The frequency response of a filter or system is usually defined as a ratio of polynomials

\[
H(j\omega) = \frac{b_q(j\omega)^q + b_{q-1}(j\omega)^{q-1} + \cdots + b_1(j\omega) + b_0}{a_p(j\omega)^p + a_{p-1}(j\omega)^{p-1} + \cdots + a_1(j\omega) + a_0} = \frac{\sum_{m=0}^{q} b_{q-m}(j\omega)^m}{\sum_{n=0}^{p} a_{p-n}(j\omega)^n}
\]  (9-1)
One approach to designing a filter is to calculate the frequency response coefficients, \(a_i\) and \(b_i\), of the Equation (9-1) such that the frequency response tends to the desired response, which is typically among one of the ideal responses described above.

There are several commands in MATLAB to design the filter coefficients, the most common being the ‘Butter’ command. This command creates any of the four filter types mentioned above and returns two vectors, \(b = [b_0, b_1, \ldots, b_q]\) and \(a = [a_0, a_1, \ldots, a_p]\), which include the coefficients of the numerator and denominator polynomials in the Equation (9-1). The general syntax for this command is as follows

\[
[b, a] = \text{butter}(n, Wn, ‘ftype’)
\]

and

\[
[b, a] = \text{butter}(n, Wn, ‘ftype’, ’s’)
\]

where \(n\) is the filter order specifying how many coefficients are to be returned in vectors \(a\) and \(b\). The first command returns the coefficients for a digital filter while the second one is used for an analog filter. \(Wn\) is the cutoff frequency which is normalized for the digital filter, but for an analog filter, it is the actual cutoff frequency in rad/sec. ‘ftype’ also determines the filter type, whether it is a lowpass, highpass, bandpass, or bandstop

- ‘high’ for a highpass filter with cutoff frequency \(Wn\)
- ‘low’ for a lowpass digital filter with cutoff frequency \(Wn\)
- ‘stop’ for an order 2\(\times\)n bandstop filter if \(Wn\) is a two-element vector, \(Wn = [\omega_1 \omega_2]\). The stopband is \(\omega_1 < \omega < \omega_2\).
- For a bandpass filter, \(Wn\) is a two-element vector, \(Wn = [\omega_1 \omega_2]\), with the passband being \(\omega_1 < \omega < \omega_2\). Here, nothing is placed for ‘ftype’.

When the filter coefficients are calculated and returned by ‘butter’ command in the vectors \(b\) and \(a\), the filter corresponding to the expression (9-1) is to be applied to the input signal. This is performed by using either ‘filter’ or ‘lsim’ commands. If the first syntax of ‘butter’ is used to design a digital filter, the corresponding coefficients ‘\(a\)’ and ‘\(b\)’ are employed as the input arguments to ‘filter’, while ‘lsim’ employs the coefficients resulting from the second syntax of ‘butter’ used to create an analog filter. The general syntax for ‘filter’ command is as follows

\[
y = \text{filter}(b, a, x)
\]

where ‘\(x\)’ is the input signal vector to be filtered by the digital filter specified by the coefficients ‘\(a\)’ and ‘\(b\)’. The general syntax for ‘lsim’ is as follows

\[
y = \text{lsim}(b, a, x, t)
\]
where ‘x’ is the input signal vector to be filtered by the analog filter specified by the coefficients ‘a’ and ‘b’, and ‘t’ is a vector with the same length as ‘x’. Each element of the vector ‘t’ corresponds to the time of occurrence of each element of the vector ‘x’.

The frequency response of the designed filter, including magnitude and phase, can be plotted using ‘freqs’ and ‘freqz’ commands for the analog and digital filters, respectively. The syntax for these commands is as follows

\[ freqs(b,a) \]

and

\[ freqz(b,a) \]

where ‘a’ and ‘b’ are the numerator and denominator polynomial coefficients of the designed filter. Note that the plots of the frequency response generated by these commands are in logarithmic scale.

**Example: lowpass filtering of a noisy EKG signal**

Suppose that the signal to be filtered is an EKG signal degraded by a fast changing noise, a noise with high frequency components. The signal sample values x and the corresponding recording time t are stored in a MATLAB file ‘EKG.mat’ and is loaded by MATLAB code using ‘load’ command. Then an analog lowpass filter is designed as well as a digital one by the command ‘butter’ and their frequency responses are plotted in Figures 9-4 and 9-6. The filters are applied to the signal using the ‘lsim’ and ‘filter’ commands. The following MATLAB code shows how it is implemented. The noisy EKG signal as well as the filtered signals are plotted in Figures 9-5 and 9-7.
clc; clear; close all;
load EKG.mat; %load t and x vectors

%-----------------Lowpass Filter Design (Analog)-----------------------------
[b,a] = butter(8,100,'low','s'); %LPF for frequencies less than 100rad/s
figure; freqs(b,a); %plot the frequency response of analog filter
%-----------------Lowpass Filtering-----------------------------------------
y_a = lsim(b,a,x,t); %Apply the filter to the signal x to get the output y

figure; subplot(2,1,1);
plot(t,x,'b');
xlim([t(1) t(end)]);
xlabel('Time (sec)'); ylabel('Noisy EKG Signal');
subplot(2,1,2); plot(t,y_a,'r');
xlim([t(1) t(end)]);
xlabel('Time (sec)'); ylabel('Lowpass-filtered EKG');

%-----------------Lowpass Filter Design (Digital)---------------------------
fs = 1/t(2); %sampling frequency in Hz
w_s = 2*pi*fs; %sampling frequency in rad/sec
[b,a]=butter(8,100/(w_s/2),'low');%LPF for frequencies less than 100rad/s
figure; freqz(b,a); %plot the frequency response of digital filter
%-----------------Lowpass Filtering-----------------------------------------
y_d = filter(b,a,x); %Apply the filter to the signal x to get the output y

figure; subplot(2,1,1);
plot(t,x,'b');
xlim([t(1) t(end)]);
xlabel('Time (sec)'); ylabel('Noisy EKG Signal');
subplot(2,1,2); plot(t,y_d,'r');
xlim([t(1) t(end)]);
xlabel('Time (sec)'); ylabel('Lowpass-filtered EKG');

Figure 9-3. MATLAB implementation of lowpass filtering an EKG signal
Figure 9-4. Frequency response of the designed lowpass analog filter with cutoff=100\text{rad/sec}

Figure 9-5. The noisy EKG signal and the corresponding lowpass-filtered EKG
Figure 9-6. Frequency response of the designed lowpass digital filter with cutoff=100rad/sec

Figure 9-7. The noisy EKG signal and the corresponding lowpass-filtered EKG
Assignment 9: Lowpass, Highpass, and Bandstop Filters

In this section, you are going to use the filter concepts to process a real signal. Please use the MATLAB m-file editor to write your code and evaluate the results in different mfiles for each section of this question.

1. Using the ‘wavread’ command, load the samples of the recorded digital signal (‘Sentence.wav’) as well as the sampling frequency ($f_s$) and listen to it using the ‘sound’ command.

   a. Plot the waveform of the signal versus time (in seconds) using subplot(3,1,1). Make sure to have proper labels and limit bounds on both $x$ and $y$ axes.

   b. Using ‘butter’ command, design a lowpass filter with cutoff frequency of 1 kHz and order of 10. Then using the ‘filter’ command, apply the designed filter to the signal. Plot the output waveform of the signal versus time (in seconds) using subplot(3,1,2). Make sure to have proper labels and limit bounds on both $x$ and $y$ axes. Listen to the output signal and explain any difference with the original one (in both the waveforms and what you hear). Also, report the phonemes (in terms of consonants and vowels) that can or cannot be heard.

   c. Using ‘butter’ command, design a highpass filter with cutoff frequency of 6 kHz and order of 10. Then using the ‘filter’ command, apply the designed filter to the signal. Plot the output waveform of the signal versus time (in seconds) using subplot(3,1,3). Make sure to have proper labels and limit bounds on both $x$ and $y$ axes. Listen to the output signal and explain any difference with the original one (in both the waveforms and what you hear). Also, report the phonemes (in terms of consonants and vowels) that can or cannot be heard.

   d. Summarize your observations by making a statement about the nature of the phonemes whether they are of low or high frequency components.

2. Using the ‘wavread’ command, load the samples of the recorded digital signal (‘NoisySentence1.wav’) as well as the sampling frequency ($f_s$) and listen to it using the ‘sound’ command.

   a. Plot the original (‘Sentence.wav’) and the noisy (‘NoisySentence1.wav’) signal versus time (in seconds) using subplot(4,1,1) and subplot(4,1,2), respectively. Make sure to have proper labels and limit bounds on both $x$ and $y$ axes.

   b. Based on the filtering concept, design a proper filter (highpass, lowpass, bandpass, or bandstop filter with proper cutoff frequencies) to reduce the noise. To do so, listen to the noise carefully and indicate whether it sounds like high frequency or low frequency voices, and then find and appropriate filter for it. Apply the filter to the noisy signal, listen to the output signal, and modify the cutoff frequency by trial and
error if needed. Plot your best result of denoising using subplot(4,1,3) (use proper axes labels and limit bounds) and compare the denoised waveform with both the original and noisy waveforms.

c. Based on instructor’s explanations in the lab and using the ‘spectrogram’ command, find the best choice for the filter. Apply the filter to the noisy signal and listen to your signal. Plot your best result of denoising using subplot(4,1,4) (use proper axes labels and limit bounds) and compare the denoised waveform with both the original and noisy waveforms.

3. Using what you learned in section 2, design a proper filter to denoise the signal (‘NoisySentence2.wav’). Plot the original, noisy, and denoised signal in the same figure with respect to time (sec). Also include the spectrogram of the clean, noisy, and denoised signals (three spectrograms in separate figures) in your report. Explain your observations.