ELEN E3084: Signals and Systems Lab
Lab VI: Analog Filter Design

1 Introduction

This lab corresponds to Section 7 of the Lathi course textbook, and here we'll design four basic types of filters: the lowpass, bandpass, highpass and bandstop filters. Filters are used to eliminate unwanted signal components such as noise and distortion. For example, a lowpass filter is used to suppress high frequency components of a signal. Also recall from Section 5 of the textbook that an anti-aliasing filter (an analog lowpass filter) is used prior to sampling of a continuous signal in order to get rid of high frequency components of the signal. This is essential in order to maintain the integrity of the low frequency components during sampling. Figure 1 shows ideal filter characteristics for lowpass, bandpass, highpass, and bandstop filters.

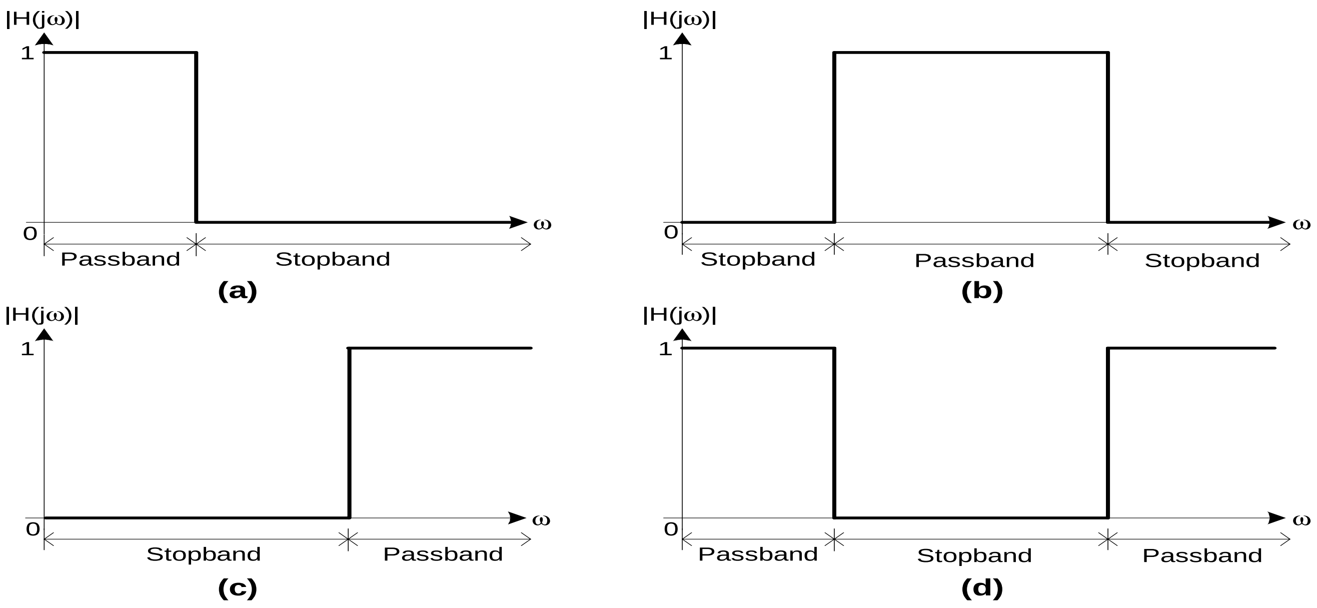

Figure 1: Ideal filter characteristics of various types of filters. (a) Lowpass filter. (b) Bandpass filter. (c) Highpass filter. (d) Bandstop filter.

As can be seen from Figure 1, an ideal filter has a passband (unity gain) and a stopband (zero gain) with a sudden transition from the passband to the stopband. There is no transition band. However, for practical filters (see Figure 2), the transition takes place over a finite band of frequencies. Moreover, for realizable filters, the gain cannot be zero over a finite band (Paley-Wiener condition). We therefore define a stopband to be a band over which the gain is below some small number G_s (maximum stopband gain), as illustrated in Figure 2. Similarly, we define a passband to be a band over which the the gain is between 1 and some number G_p (minimum passband gain). In our design procedure, G_p, G_s, ω_p, and ω_s will be specified for lowpass and highpass filters. For bandpass and bandstop filters, G_p, G_s, ω_{p1}, ω_{s1}, ω_{p2}, and ω_{s2} will be specified. In this lab, the gains (G_p and G_s) are specified in terms of decibels and the frequencies are specified in radians.

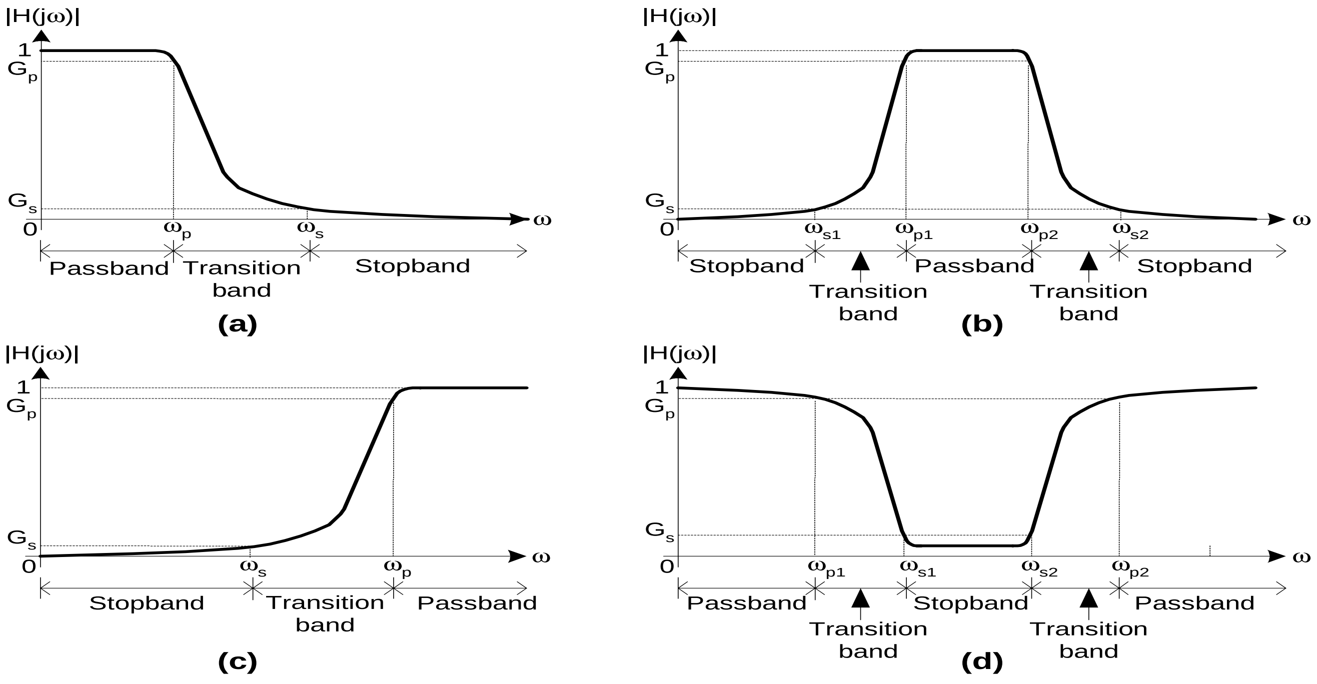

Figure 2: Passband, stopband, and transitionband in various types of filters. (a) Lowpass filter. (b) Bandpass filter. (c) Highpass filter. (d) Bandstop filter.

Before we begin, first go to the `lab6` directory and start the `diary` function. This creates a diary file that must be submitted to the LA or TA at the end of the lab session. Don't forget to turn off the function at the end of the session.

2 Lowpass Filter

In this section, we'll design different classes of lowpass filters (Butterworth, Chebyshev, and elliptic filters) for a given set of specifications. The students are highly encouraged to read Sections 7.5 and 7.6 to fully understand the steps involved in designing a lowpass filter.

To Do 1

For each filter we design, we'll test the filter with the musical signal used in Lab IV. A

MATLAB function `wavinfo` that reads a WAV file and plots the signal in both time and frequency domain is provided. See how the input signal looks like by typing the following command.

```
> wavinfo('neneh32.wav');
```

What is the duration of the signal?
What is the bandwidth of the signal?

Now, we'll design a lowpass filter so that we can hear only the low frequency components of the given speech signal. We'll design 4 different classes of lowpass filters and compare them.

To Do 2

Design a Butterworth lowpass filter that meets the following specifications: $G_p = -2$dB and $G_s = -20$dB. You should determine your own passband and stopband edge frequencies (ω_p and ω_s). You can look at the spectrum of the given signal and determine the frequency range of the signal components that you want to filter. If time permits, the filter can be tested with different values of edge frequencies.

A MATLAB function `makefilter`, which can be used to create a filter that meets a set of given specifications, is provided. Note that the gains are specified in decibels and the frequencies in radians.

```
> % wp and ws are the edge frequencies.
> [num, den] = makefilter('lowpass', 'butter', -2, -20, wp, ws);
```

What is the filter order and the transfer function of the lowpass filter?
Now test the filter using the provided function, `runfilter`.

```
> runfilter(num, den, 'neneh32.wav');
```

Does your filter behave as expected?

Design a Chebyshev (Type I) lowpass filter that meets the same set of specifications. Write down the filter order and the transfer function of the lowpass filter. For this, we can again use the `makefilter` function with the second parameter set to `'cheby1'`.

What is the difference between the amplitude response of the Butterworth filter and that of the Chebyshev filter? Which filter is better? Why?

Design an inverse Chebyshev (Chebyshev Type II) lowpass filter with the same specifications. What is the difference between this filter and the Chebyshev Type I filter?

Design an elliptic lowpass filter with the same specifications.

Which lowpass filter would you prefer? Why?

---

3 Bandpass, Highpass and Bandstop Filters

Using certain frequency tranformations, the transfer functions of bandpass, highpass and bandstop filters can be obtained from a basic lowpass filter design. The interested students should read Section 7.7 of the textbook for detailed information about designing the filters using frequency transformations. In this section, we'll simply use the `makefilter` function to create the filters.

3.1 Bandpass Filter

To Do 3

Design four bandpass filters (Butterworth, Chebyshev Type I, Chebyshev Type II, and elliptic filters) that meet the following specifications: $G_p = -2$dB and $G_s = -20$dB. Again, you should determine your own passband and stopband edge frequencies. In this case, the edge frequencies (ω_p and ω_s) are vectors with two elements. Make sure the frequencies are chosen appropriately so that you can see the low and high frequency components of the given signal are missing after it gets filtered.

```
> [num, den] = makefilter('bandpass', 'butter', -2, -20, wp, ws);
> runfilter(num, den, 'neneh32.wav');
> % Repeat the above commands for the other three classes of filters.
```

What is the filter order and the transfer function for each bandpass filter?
Do your filters behave as expected?

3.2 Highpass Filter

To Do 4

Design four highpass filters (Butterworth, Chebyshev Type I, Chebyshev Type II, and elliptic filters) and test them with the given speech signal.

What is the filter order and the transfer function for each filter?
Do your filters behave as expected?

3.3 Bandstop Filter

To Do 5

Design four bandstop filters (Butterworth, Chebyshev Type I, Chebyshev Type II, and elliptic filters) and test them with the given speech signal. Again, make sure that the frequencies are chosen appropriately so that you only see the low and high frequency components of the given signal after it gets filtered.

What is the filter order and the transfer function for each filter?
Do your filters behave as expected?

4 Lab Problems

In this section, using just lowpass and highpass filters, we'll design bandpass and bandstop filters. Assume two programmable black boxes are given as shown in Figure 3. Figure 3(a) represents a lowpass filter whose edge frequencies (ω_p and ω_s) can be controlled, and Figure 3(b) represents a highpass filter. Using the given blocks in any way, draw a block diagram representation of a system that filters the components of an input signal with frequencies in the range [1K, 3K] Hz. In your diagram, clearly indicate the edge frequency values used in the lowpass and highpass filters. Then, obtain the transfer function of the system you designed and test the system with the audio signal we studied in this lab. Does your system behave as expected?

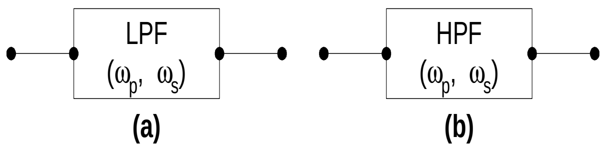

Figure 3: Black box representation of programmable lowpass and highpass filters. (a) Lowpass filter. (b) Highpass filter.

Repeat the above procedure, but now the system should suppresses the components with frequencies in the range [1K, 3K] Hz.

Hint:

1. *First the transfer function representing each block of the system should be obtained.*
2. *You may find some of the MATLAB functions* - `series`, `feedback`, `parallel`, *and* `append` - *useful when finding the transfer function of the whole system.*

5 Appendix

This appendix lists the MATLAB code of the functions used in the lab.

wavinfo

```
function wavinfo(wav)
% WAVINFO Reads a WAV file and plots the signal both in time and frequency domain.
%   wavinfo(wav)
%
% wav: name of a WAV file

% Read the input signal
[m,fs] = wavread(wav);
% Keep only a small part of it to speed up calculations
m = m(110000:230000);
t = linspace(0, length(m)/fs, length(m));
f = linspace(-fs/2, fs/2, length(m));
% Listen to it
soundsc(m, fs);

figure;
% Look at the original time signal
subplot(2,1,1); plot(t,m); grid on;
axis([min(t), max(t), min(m), max(m)]);
title('The original time signal');
% Look at the spectrum of the original signal
M = am_spectrum(m);
subplot(2,1,2); plot(f, M); grid on;
axis([min(f), max(f), min(M), max(M)]);
title('The spectrum of the original signal');
```

makefilter

```
function [num, den] = makefilter(type, class, Gp, Gs, wp, ws)
% MAKEFILTER Creates an analog filter that satisfies the given criteria.
%   makefilter(type, class, gp, gs, wp, ws)
%
% type: type of the filter - 'lowpass', 'bandpass', 'highpass', or 'bandstop'
% class: filter class: 'butter', 'cheby1', 'cheby2' or 'elliptic'
% Gp: minimum passband gain in dB
% Gs: maximum stopband gain in dB
% wp: passband edge frequencies in radians
% ws: stopband edge frequencies in radians
% num: numerator of the transfer function of the filter
% den: denominator of the transfer function of the filter

if ~strcmp(type,'lowpass')  & ~strcmp(type,'bandpass') & ~strcmp(type,'highpass') & ~strcmp(type,'bandstop')
    error('Invalid option for the type of the filter.');
end
if ~strcmp(class,'butter')  & ~strcmp(class,'cheby1') & ~strcmp(class,'cheby2') & ~strcmp(class,'elliptic')
    error('Invalid option for the filter class.');
end

if strcmp(type,'lowpass')
    if strcmp(class,'butter')
        % Butterworth
        [n, wn] = buttord(wp, ws, -Gp, -Gs, 's');
        [num, den] = butter(n, wn, 's');
    elseif strcmp(class,'cheby1')
        % Chebyshev Type I
        [n wn] = cheb1ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby1(n, -Gp, wn, 's');
    elseif strcmp(class,'cheby2')
        % Chebyshev Type II
        [n wn] = cheb2ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby2(n, -Gs, wn, 's');
    else
        % Elliptic
        [n wn] = ellipord(wp, ws, -Gp, -Gs, 's');
        [num, den] = ellip(n, -Gp, -Gs, wn, 's');
    end
elseif strcmp(type, 'bandpass')
    if strcmp(class,'butter')
        % Butterworth
        [n, wn] = buttord(wp, ws, -Gp, -Gs, 's');
        [num, den] = butter(n, wn, 's');
    elseif strcmp(class,'cheby1')
        % Chebyshev Type I
        [n wn] = cheb1ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby1(n, -Gp, wn, 's');
    elseif strcmp(class,'cheby2')
        % Chebyshev Type II
        [n wn] = cheb2ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby2(n, -Gs, wn, 's');
    else
        % Elliptic
        [n wn] = ellipord(wp, ws, -Gp, -Gs, 's');
        [num, den] = ellip(n, -Gp, -Gs, wn, 's');
    end
elseif strcmp(type, 'highpass')
    if strcmp(class,'butter')
        % Butterworth
        [n, wn] = buttord(wp, ws, -Gp, -Gs, 's');
        [num, den] = butter(n, wn, 'high', 's');
    elseif strcmp(class,'cheby1')
        % Chebyshev Type I
        [n wn] = cheb1ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby1(n, -Gp, wn, 'high', 's');
    elseif strcmp(class,'cheby2')
        % Chebyshev Type II
        [n wn] = cheb2ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby2(n, -Gs, wn, 'high', 's');
    else
        % Elliptic
        [n wn] = ellipord(wp, ws, -Gp, -Gs, 's');
        [num, den] = ellip(n, -Gp, -Gs, wn, 'high', 's');
    end
else
    if strcmp(class,'butter')
        % Butterworth
        [n, wn] = buttord(wp, ws, -Gp, -Gs, 's');
        [num, den] = butter(n, wn, 'stop', 's');
    elseif strcmp(class,'cheby1')
        % Chebyshev Type I
        [n wn] = cheb1ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby1(n, -Gp, wn, 'stop', 's');
    elseif strcmp(class,'cheby2')
        % Chebyshev Type II
        [n wn] = cheb2ord(wp, ws, -Gp, -Gs, 's');
        [num, den] = cheby2(n, -Gs, wn, 'stop', 's');
    else
        % Elliptic
        [n wn] = ellipord(wp, ws, -Gp, -Gs, 's');
        [num, den] = ellip(n, -Gp, -Gs, wn, 'stop', 's');
    end
end
```

```
fprintf('The filter order n = %i\n', n);
fprintf('The transfer function of the filter is:\n');
printsys(num, den);

% Plot the magnitude response of the filter
w = 0:1:2*pi*4500;
w = w';
[mag, phase, w] = bode(num, den, w);
figure;
plot(w, mag);
title(['The ' type ' filter (' class ')']);
```

runfilter

```
function runfilter(num, den, wav)
% RUNFILTER Tests a given filter
%   runfilter(num, den, m)
%
% num: numerator of the transfer function of the filter
% den: denominator of the transfer function of the filter
% wav: name of a WAV file

% Read the input signal
[m,fs] = wavread(wav);
% Keep only a small part of it to speed up calculations
m = m(110000:230000);
t = linspace(0, length(m)/fs, length(m));
f = linspace(-fs/2, fs/2, length(m));
M = am_spectrum(m);

% Filter
sys = tf(num, den);
newm = lsim(sys, m, t);
% Listen to the filtered signal
soundsc(newm, fs);

figure;
% Look at the filtered time signal
subplot(2,1,1); plot(t, newm); grid on;
axis([min(t), max(t), min(m), max(m)]);
title('The filtered time signal');
% Look at the spectrum of the filtered signal
subplot(2,1,2); plot(f, am_spectrum(newm)); grid on;
axis([min(f), max(f), min(M), max(M)]);
title('The spectrum of the filtered signal');
```