Lab 1 – Filter Design and Evaluation in MATLAB

S0002E – DSP Systems In Practice

Andre Lundkvist
nadlun-5@student.ltu.se

Rikard Qvarnström
rikqva-5@student.ltu.se

Luleå, 6th October 2009
Abstract

This report is for a lab in DSP - Systems in Practice (S0002E), that gives a elementary introduction to digital filter design in MATLAB. The focus of this report will be placed on the notch filter and the low-pass filter with different implementations.

The first part is to remove a sinusoid disturbance from a song. Because the frequency is fixed a notch filter will be used to remove the disturbance. The notch filter used here will be implemented using the input output difference equation implemented in MATLAB. The filter used will be of first order because the only requirement is to have at least a attenuation of \(-50\) dB at the center frequency. Since there are no requirements on the Q-factor of the filter, the only thing to take in consideration is the attenuation of the center frequency.

The low-pass filter will be implemented in several ways, to demonstrate that there are more then one way to implement a filter and get equivalent results. The low-pass filter is first implemented as an Infinite Impulse Response (IIR) filter, which means that the output depends on both the input and the previous outputs.

The second implementation is using a Finite Impulse Response (FIR). This differs from an IIR filter by only using the input signal, and no feedback from the output signal.
## Contents

1 Introduction 3
   1.1 Low Pass Filter 3
   1.2 Notch Filter 3

2 Assignments 4
   2.1 Assignment 1 4
   2.2 Assignment 2 6
   2.3 Assignment 3 7
   2.4 Assignment 4 10
   2.5 Assignment 5 12

A MATLAB Code 14
   A.1 Lab 1 - Main Program 14
   A.2 Lab 1 - Assignment 1 15
   A.3 Lab 1 - Assignment 2 16
   A.4 Lab 1 - Assignment 3 17
   A.5 Lab 1 - Assignment 4 19
   A.6 Lab 1 - Assignment 5 20
List of Figures

1  Bode plot of the notch filter ........................................ 4
2  Pole/Zero plot of notch filter system function ...................... 5
3  Power Spectral Density .................................................. 5
4  Bode plot of the \texttt{myfilter} ........................................... 7
5  Pole/Zero plot of the system ............................................ 8
6  Impulse Response of the \texttt{myfilter} implementation. Input impulse is delayed 5 samples .................................................. 8
7  Top: Unfiltered input signal. Bottom: Filtered output signal ........ 9
8  Bode plot of the IIR lowpass filter .................................... 10
9  Pole/Zero plot of IIR lowpass filter system function ............... 11
10 The impulse response of the IIR lowpass filter ....................... 11
11 Bode plot of the FIR lowpass filter .................................. 12
12 Pole/Zero plot of FIR filter system .................................. 13
13 The impulse response of the FIR filter ................................ 13
1 Introduction

This lab will introduce some elementary filters and their design in MATLAB. MATLAB will be used as a tool for designing and evaluating the digital filters. The filters used in this lab are the lowpass filter and the notch filter. These will be implemented using several different MATLAB functions.

In the general case for high- and low-pass filters the slope increase or decrease by 6 dB/octave or 20 dB/decade respectively.

1.1 Low Pass Filter

The general low pass filter (LP-filter) is characterized by the attenuation of the higher frequencies and passing the low, sometimes with a gain. The phase characteristics depends on the order of the filter and the implementation.

1.2 Notch Filter

The notch filter is characterized by the sharp attenuation dip in frequency domain. This filter type is generally used to remove a disturbance of a known narrow band frequency. The width of the attenuation dip and the maximum attenuation depends on the order and implantation of the filter. The maximum gradient of the phase slope is at the center of the attenuation frequency.
2 Assignments

2.1 Assignment 1

In assignment 1, a song was given but distorted by a pure sinusoid. The simplest way to remove this type of noise is by using a notch filter tuned to the specific frequency. Since the song is already a sampled wavefile, a digital filter is appropriate to utilize. MATLAB will be used to implement the filter.

A notch filter is characterized with one or several sharp anti-resonances at the tuned frequencies. See the magnitude curve in figure 1. The response can also be described using a pole/zero plot in the complex $\mathbb{Z}$-Domain (see figure 2).

The function created will locate the most dominant frequency and tune the notch filter to remove it. If the frequency of the disturbance would be displaced, the program will find and kill it. Analysis of the wavefile can be seen in figure 3.

The code for this program can be found in appendix A.2 at page 15.

![Assignment 1 – Bode plot](image1)

![Assignment 1 – Bode plot](image2)

Figure 1: Bode plot of the notch filter
Figure 2: Pole/Zero plot of notch filter system function

Figure 3: Power Spectral Density
2.2 Assignment 2

The function `myfilter` works as a real-time filter, by utilizing the definition of the input-output difference equation (1).

\[
\sum_{k=0}^{N} a_k y[n - k] = \sum_{r=0}^{M} b_r x[n - r]
\]  (1)

The MATLAB implementation of the filter can be found in appendix A.3 on page 16. The function takes the arguments:

**A_coeffs** Vector containing the polynomial \(a_k\) coefficients, which correspond to the system poles.

**B_coeffs** Vector containing the polynomial \(b_r\) coefficients, which correspond to the system zeros.

**output_state** The system’s previous output states

**input_state** The system’s current and previous inputs

The function `myfilter` returns the current output sample \(y[n]\).
2.3 Assignment 3

To implement the function `myfilter` in MATLAB a for loop is used, to simulate the real-time behavior of the input signal.

The filter’s impulse response can be seen in figure 6. The input impulse is delayed to location \( n = 5 \), and it’s Bode plot in figure 4.

The power spectral density of the input signal and the filtered output signal is displayed in figure 7. The top plot shows the unfiltered input signal. The bottom plot shows the output signal filtered using the `myfilter` function. The disturbance has clearly been removed from the signal. Unfortunately the filter also removes some of the signal content near the disturbance in the frequency domain.

The MATLAB code to the filter implementation can be found in appendix A.4 on page 17.

![Figure 4: Bode plot of the myfilter](image)
Figure 5: Pole/Zero plot of the system

Figure 6: Impulse Response of the \texttt{myfilter} implementation. Input impulse is delayed 5 samples
Figure 7: Top: Unfiltered input signal. Bottom: Filtered output signal
2.4 Assignment 4

The design criteria for the IIR lowpass filter is:

**Cutoff Frequency** $\Omega_p = \pi/10$

**Stop Band Frequency** $\Omega_s = \pi/5$

**Minimum Pass Band Magnitude** $PB_{min} = -6$ dB

**Maximum Pass Band Magnitude** $PB_{max} = 3$ dB

**Maximum Stop Band Magnitude** $SB_{max} = -50$ dB

The final IIR filter design (see code in appendix A.5 on page 19) results in:

**Filter Order** 8

**Cutoff Frequency** $f_{-3\, dB} = 2355.5$ Hz

**Stop Band Frequency** $f_{-50\, dB} = 4410$ Hz

The Bode plot in figure 8 shows the magnitude and phase response of the system. In the lower frequencies the response is fairly smooth where the signal goes through the filter without much attenuation. In the higher frequencies where the attenuation is high, their exact response is usually of less importance since the signal will be extremely attenuated.

The pole/zero plot of the filter is shown in figure 9.

The impulse response of the filter is shown in figure 10 resembles a dampened sinusoid.

![Figure 8: Bode plot of the IIR lowpass filter](image)
Figure 9: Pole/Zero plot of IIR lowpass filter system function

Figure 10: The impulse response of the IIR lowpass filter
2.5 Assignment 5

The design criteria for the FIR lowpass filter is:

**Cutoff Frequency** $\Omega_p = \pi/10$

**Stop Band Frequency** $\Omega_s = \pi/5$

**Minimum Pass Band Magnitude** $PB_{min} = -6$ dB

**Maximum Pass Band Magnitude** $PB_{max} = 3$ dB

**Maximum Stop Band Magnitude** $SB_{max} = -50$ dB

The final FIR filter design (see code in appendix A.6 on page 20) results in:

**Filter Order** 34

**Cutoff Frequency** $f_{-3\,dB} = 2204.5$ Hz

**Stop Band Frequency** $f_{-50\,dB} = 4379.5$ Hz

The Bode plot in figure 11 shows the magnitude and phase response of the system. The magnitude of the system is smooth for the lower frequencies. The signal will be distorted by the combfilter effect seen in the higher frequencies, but because the average high attenuation it won’t be as noticeable on the output signal. The phase is linear in the pass band, so the effect on the output signal is not so problematic. On the higher frequencies the phase suffers from the combfilter effect. Due to the higher attenuation this should not affect the overall response much.

The pole/zero plot of the filter is shown in figure 12.

The impulse response of the filter is shown in figure 13.

![Figure 11: Bode plot of the FIR lowpass filter](image-url)
Figure 12: Pole/Zero plot of FIR filter system

Figure 13: The impulse response of the FIR filter
A MATLAB Code

A.1 Lab 1 - Main Program

% Andre Lundkvist and Rikard Qvarnstrm
% Lab 1
% Filter Design and Evaluation in MATLAB

function lab1(assignment)
clc;
close all;
if nargin < 1
    assignment = 3;
end

% Note to self:
% use profile to check performance
prof = false;
if prof == true
    profile on;
tic;
end

[x, fs] = wavread('boyNY.wav');

% Chose which assignment to run:
switch assignment
    case 1
        % Code for Assignment 1 <here>
    case 2
        % See the myfilter() function
    case 3
        % Code for Assignment 3 <here>
    case 4
        % Code for Assignment 4 <here>
    case 5
        % Code for Assignment 5 <here>
    otherwise
        % If input does not match assignment:
        fprintf('Wrong input\n');
end

if prof == true
    profile off;
    profile viewer;
    trun = toc;
    fprintf('Total time elapsed = %g\n', trun);
end
fprintf('\n');
A.2 Lab 1 - Assignment 1

% Andre Lundkvist and Rikard Qvarnström
% Lab 1, Assignment 1
% Notch filter design

h = spectrum.periodogram;
hpds = psd(h,x,'Fs',fs);
figure; % fig 1
plot(hpds);
title('Assignment 1 - Periodogram Power Spectral Density Estimate');
print -depsc -tiff -r300 Lab_1_Ass_1_PSD

fx = real(fft(x,fs));
freq = find(fx == max(fx)); % find the most dominating frequencies
freq(2) = fs - freq(2);
meanfreq = mean(freq); % take the mean of the 2 detected peaks
figure;
axis([0 fs/2 min(fx) max(fx)]);

% Design
angle = 2*pi*meanfreq/fs; % the angle for the zeroes placement
B = [1 -2*cos(angle) cos(angle)^2 + sin(angle)^2]; % polynomial coefficients
A = [1 0 0]; % two poles in the center of the unit circle
[hplot, fplot] = freqz(B,A,fs,fs);
figure; % fig 2
zplane(B,A); % the pole and zeroes plot
print -depsc -tiff -r300 Lab_1_Ass_1_PZ

figure; % fig 3
freqzplot(hplot, fplot, 'Hz'); % bode plot (frequency and phase)
title('Assignment 1 - Bode plot');
print -depsc -tiff -r300 Lab_1_Ass_1_Bode

% Filter the input
xf = filter(B, A, x);
figure;

% Play the music files
% sound(x, fs); % original file
% sound(xf, fs); % After filtering
% Andre Lundkvist and Rikard Qvarnström
% Lab 1, Assignment 2
% Filter Design and Evaluation in MATLAB

% Filter Routine Implementation
function output = myfilter(A_coeffs, B_coeffs, output_state, input_state)

% To be able to divide by a0 and multiply with the other coefficients later
a0 = A_coeffs(1);
A_coeffs = A_coeffs(2:end);

% To calculate the lengths only once
L_A = length(A_coeffs);
L_B = length(B_coeffs);
L_OS = length(output_state);
L_IS = length(input_state);

% Fault handling, in case that the size of the input vectors is different
% buffer the smaller one with zeros so they become the same length
if (L_IS < L_B)
    input_state = [zeros(1, L_B-L_IS) input_state];
else if (L_IS > L_B)
    B_coeffs = [zeros(1, L_IS-L_B) B_coeffs];
    input_state = input_state(end-L_B+1:end);
end
if (L_OS < L_A)
    output_state = [zeros(1, L_A-L_OS) output_state];
else if (L_OS > L_A)
    A_coeffs = [zeros(1, L_OS-L_A) A_coeffs];
    output_state = output_state(end-L_A+1:end);
end

xsum = sum(B_coeffs.*input_state);
ysum = sum(A_coeffs.*output_state);
output = (xsum-ysum)/a0;
end
A.4 Lab 1 - Assignment 3

1 \% Andre Lundkvist and Rikard Qvarnström
2 \%
3 \% Lab 1, Assignment 3
4 \% Filter Design and Evaluation in MATLAB
5
6 %f = real(fft(x,fs));
7 freq = find(fx == max(fx)); % find the most dominating frequencies
8 freq(2) = fs-freq(2); % meanfreq = mean(freq); % take the mean of the 2 detected peaks
9
10 scale = 1; % Unused when set to 1
11 angle = 2*pi*meanfreq/fs; % the angle for the zeroes placement
12 B = [1 -2*cos(angle)*scale ((cos(angle)*scale)^2+(sin(angle)*scale)^2)] % polynomial coefficients
13 A = [1 0 0]; % two poles in the center of the unit circle
14
15 zplane(B,A); % the pole and zeroes plot
16 title('Assignment 3 - Pole/Zero plot'); %
17 print -depsc -tiff -r300 Lab_1_Ass_3_PZ
18 figure;
19 [hplot,fplot]=freqz(B,A,fs,fs);
20 freqzplot(hplot,fplot,'Hz'); % bode plot (frequency and phase)
21 title('Assignment 3 - Bode plot');
22 print -depsc -tiff -r300 Lab_1_Ass_3_Bode
23
24 xlength = length(x);
25 xf = zeros(1,xlength+1); % memory allocation
26
27 % Length for time optimization
28 Alength = length(A);
29 Blength = length(B);
30 fprintf('0 %%
31 for n = Blength+1:xlength
32 xe(n) = myfilter(A, B, xe(n-Alength+1:n-1), x(n-Blength+1:n))';
33 \% ~time left
34 if mod(n,10000) == 0
35 fprintf('%i %%
36 end
37 end
38 end
39 wavwrite(xe,fs,'process3.wav');
40 h = spectrum.periodogram;
41 hpsd = psd(h,x,'Fs',fs);
42 hpsd2 = psd(h,xe,'Fs',fs);
43 figure;
44 subplot(2,1,1);plot(hpsd);
45 title('Assignment 3 - Periodogram Power Spectral Density Estimate, unfiltered');
46 subplot(2,1,2);plot(hpsd2);
47 title('Assignment 3 - Periodogram Power Spectral Density Estimate, filtered');
48 print -depsc -tiff -r300 Lab_1_Ass_3_PSD
49
50 % Send only an unit impulse to get % impulse response of filter.
51 x = zeros(1,10)'
52 x(5) = 1;
53
54 xlength = length(x);
55 xf = zeros(1,xlength+1); % memory allocation
56
57 % Length for time optimization
58 Alength = length(A);
\textbf{Blength} = \textbf{length}(B); \\
\textbf{fprintf}('0 \\
'); \\
\textbf{for} \ n = \textbf{Blength}+1:\textbf{xlength} \\
\hspace{1em} \textbf{xf}(n) = \textbf{myfilter}(A, B, \textbf{xf}(n-\textbf{Blength}+1:n-1), x(n-\textbf{Blength}+1:n)'); \\
\hspace{1em} \% \textit{~time left} \\
\hspace{1em} \textbf{if} \ \textbf{mod}(n,10000) == 0 \\
\hspace{2em} \textbf{fprintf}('\%i \\
', \textbf{round}((n/\textbf{xlength})*100)); \\
\hspace{1em} \textbf{end} \\
\textbf{end} \\
\textbf{fprintf}('\n 100 \\
\n'); \\
\textbf{figure}(); \\
\textbf{stem}(	extbf{xf}); \\
\textbf{xlabel}('Sample n'); \\
\textbf{ylabel}('Amplitude'); \\
\textbf{title}('Assignment 3 - Impulse Response'); \\
\textbf{print} -\textbf{depsc} -\textbf{tiff} -r300 \textbf{Lab_1_Ass_3_Impulse_Response} \\
\% \textit{Play the music files} \\
\%\textbf{sound}(x, fs); \% \textit{original file} \\
\%\textbf{sound}(xf, fs); \% \textit{After filtering}
A.5 Lab 1 - Assignment 4

1 % Andre Lundkvist and Rikard Qvarnström
2 % Lab 1, Assignment 4
3 % Filter Design and Evaluation in MATLAB
4
5 % Cutoff frequency \( wp = \pi/10 \),
6 % Stop band frequency \( ws = \pi/5 \),
7 % Minimum pass band magnitude \(-6\)dB,
8 % Maximum pass band magnitude \(3\)dB,
9 % Maximum stop band magnitude \(-50\)dB.
10
11 wp = \pi/10;
12 ws = \pi/5;
13 minpb = 6;
14 maxpb = 3;
15 maxsb = 50;
16
17 % normalized
18 wpn = wp/\pi;
19 wsn = ws/\pi;
20
21 [N, Wn] = buttord(wpn, wsn, minpb, maxsb);
22 [B, A] = butter(N, Wn, \textit{\textquoteleft}low\textquoteright);  
23 [hplot, fplot] = freqz(B, A, fs, fs);
24 hp = 20 * log10(abs(hplot)); % Magnitude in dB
25
26 % Find the \(-6\)dB and \(-50\)dB limits
27 f6dB = 1; f50dB = 1; % fail safe
28 for \(i = 1:\textit{length}(hp)\)
29     if hp(i) > -6
30         f6dB = i;
31         elseif hp(i) < -50
32             f50dB = i;
33         break;
34     end;
35 end;
36
37 fprintf(\textit{\textquoteleft}Order = \textbf{\%g}\textbf{\textquoteright}, N);
38 fprintf(\textit{\textquoteleft} -3 dB frequency = \textbf{\%g \textit{Hz}}\textbf{\textquoteright}, fplot(f6dB));
39 fprintf(\textit{\textquoteleft} -50 dB frequency = \textbf{\%g \textit{Hz}}\textbf{\textquoteright}, fplot(f50dB));
40
41 figure;
42 zplane(B, A); % the pole and zeroes plot
43 title(\textit{\textquoteleft}Assignment 4 - Pole/Zero plot\textquoteright);
44 print -depsc -tiff -r300 Lab_1_Ass_4_PZ
45
46 figure;
47 freqzplot(hplot, fplot, \textit{\textquoteleft}Hz\textquoteright); % bode plot (frequency and phase)
48 title(\textit{\textquoteleft}Assignment 4 - Bode plot\textquoteright);
49 print -depsc -tiff -r300 Lab_1_Ass_4_Bode
50
51 % Send only an unit impulse to get
52 % impulse response of filter.
53 x = zeros(120,1);
54 x(1) = 1;
55
56 xf = filter(B, A, x);
57
58 figure;
59 stem(xf);
60 xlabel(\textit{\textquoteleft}Sample n\textquoteright);
61 ylabel(\textit{\textquoteleft}Amplitude\textquoteright);
62 title(\textit{\textquoteleft}Assignment 4 - Impulse Response\textquoteright);
63 print -depsc -tiff -r300 Lab_1_Ass_4_Impulse_Response
% Andre Lundkvist and Rikard Qvarnstrm
% Lab 1, Assignment 5
% Filter Design and Evaluation in MATLAB

% Cutoff frequency \( wp = \pi/10 \),
% Stop band frequency \( ws = \pi/5 \),
% Minimum pass band magnitude \(-6\)dB,
% Maximum pass band magnitude \(3\)dB,
% Maximum stop band magnitude \(-50\)dB.
wp = \pi/10;
ws = \pi/5;

% Educated guess of \( N \)
N = 34;

% normalized
wpn = wp/\pi;
wsn = ws/\pi;

B = fir1(N,wpn,'low');

% Create the pole vector \( A \) and place all poles in origo
A = zeros(1,length(B));
A(1) = 1;

[hplot, fplot] = freqz(B,A,fs,fs);

hp = 20*\log10(abs(hplot)); % Magnitude in dB
f6dB = 1; f50dB = 1; % fail safe
for i = 1:length(hp)
    if hp(i) > -6
        f6dB = i;
    elseif hp(i) < -50
        f50dB = i;
    break;
end

fprintf('Order = %g
', N);
fprintf(' -6 dB frequency = %g Hz
', fplot(f6dB));
fprintf(' -50 dB frequency = %g Hz
', fplot(f50dB));

figure;
zplane(B,A); % the pole and zeroes plot
title('Assignment 5 - Pole/Zero plot');
print -depsc -tiff -r300 Lab_1_Ass_5_PZ

figure;
freqzplot(hplot,fplot,'Hz'); % bode plot (frequency and phase)
%plot(fplot/(fs/2),hp);
title('Assignment 5 - Bode plot');
print -depsc -tiff -r300 Lab_1_Ass_5_Bode
%axis([20 20e3 -200 10]);

% Send only an unit impulse to get
% impulse response of filter.
x = zeros(40,1);
x(1) = 1;

xf = filter(B,A,x);

figure;
stem(xf);
xlabel('Sample n');
ylabel('Amplitude');
title('Assignment 5 - Impulse Response');
print -depsc -tiff -r300 Lab_1_Ass_5_Impulse_Response