National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman Dated: 24/03/2023

Semester: 6th Section: BEE 12C

EE-330 Digital Signal Processing

Lab 10: IIR Filter Design using Analog Prototypes

Group Members

		PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 IIR Filter Design using Analog Prototypes

2.1 Objectives

The objective is to design digital filters with an infinite impulse response (IIR). IIR filters can be designed either directly, by means of a clever placement of poles and zeros in the z-plane, or "indirectly" by first designing a corresponding analog filter, which is then transformed to the digital domain. In this lab we will only consider the later design procedure.

- Familiarization with IIR filter design procedure
- Familiarization with frequency transformation.
- Bilinear frequency transformation
- Impulse invariance frequency transformation

2.2 Introduction

IIR filters can be designed either directly, by means of a clever placement of poles and zeros in the z-plane, or "indirectly" by first designing a corresponding analog filter, which is then transformed to the digital domain. Within this practical we will only consider the latter design procedure. Note that the described principle of using analog prototypes for IIR filter design originally stems from the idea to use digital filters as a substitute for analog systems. This allows us to use the well-constructed theory of analog filter design and simply transfer it to the digital domain.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data. The objective of this lab is to provide a hands-on experience with the A-to-D sampling and the D-to-A reconstruction processes that are essential for digital image processing. We will also demonstrate a commonly used method of image zooming (reconstruction) that produces "poor" results, which will help illustrate the importance of understanding the underlying principles of digital image processing.

2.4 Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Conclusio

3 Lab Procedure

3.1 Lab Task 1

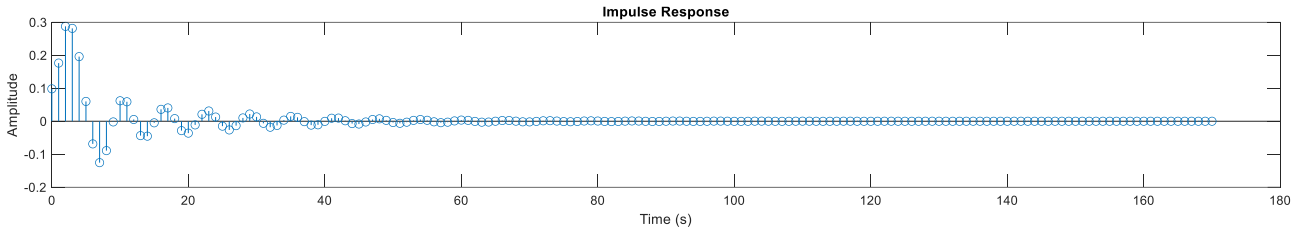
The N-th order analog Butterworth lowpass filter with cutoff frequency Ωc is given by:

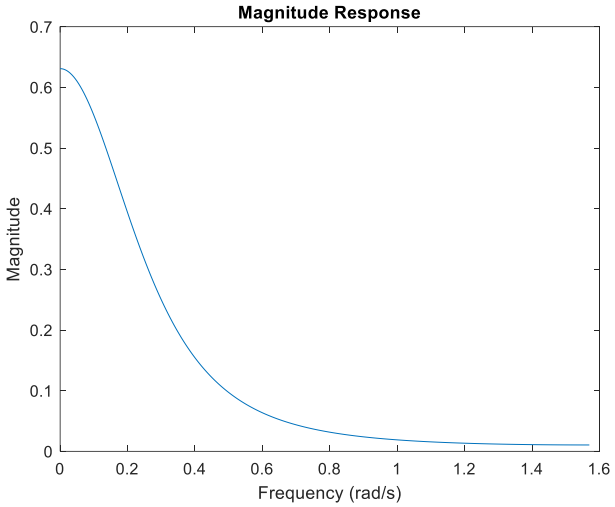
$$H_c(s) = \frac{\Omega_c^N}{\prod_{k=0}^{N-1}(s - s_k)} \quad \text{with} \quad s_k = \Omega_c e^{j\frac{\pi}{2N}(2k+N+1)} \tag{5}$$

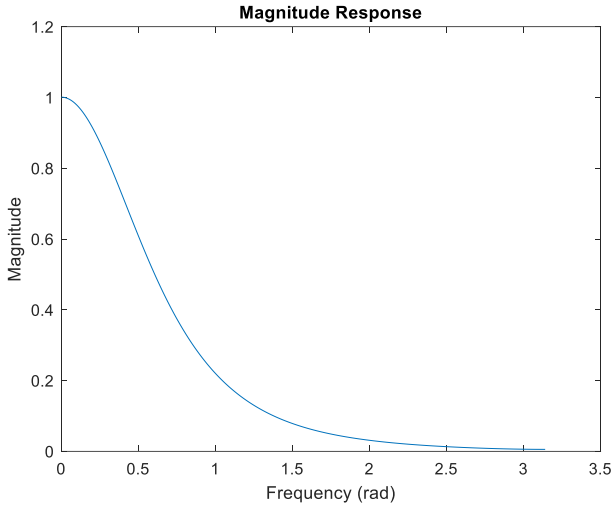
1. Design a Butterworth filter with N = 4, ωc = 0.3π and Td = 2 using the impulse invariance method. Proceed as follows:
   a) Determine Ωc and use (5) to calculate the analog filtering coefficients
   b) In the analog domain, plot hc(t) and |Hc(jΩ)|. For the analog domain plots in MATLAB use the commands zp2tf, tf, impulse and freqs.
   c) Transform your filter coefficients from the analog to the discrete time domain.
   d) In the discrete-time domain, plot h(n) and |H(ejω)| . Compare your results with (b).
2. Repeat 1. but with the bilinear transformation.
3. Further, compare both methods: what are their strengths and weaknesses?

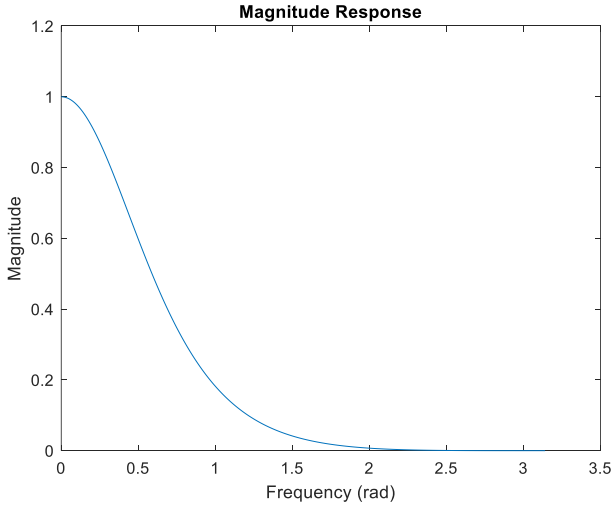
```
Td = 2;
w_c = 0.3 * pi;
omega_c = w_c / Td;
N = 4;
k = 0:(N - 1);
s_k = omega_c * exp(1i * (pi / 2 * N) * (2 * k + N - 1));
H_c = zpk([], s_k, omega_c ^ N);
[num, den] = zp2tf([], s_k, omega_c ^ N);

num = real(num); den = real(den);

H = tf(num, den);

% impulse response
figure(1);
impulse(H);
title('Impulse Response');
xlabel('Time (s)');
ylabel('Amplitude');

% magnitude response
w = 0:0.01:pi;
figure(2);
[h, w] = freqz(num, den, w);
plot(w/Td, abs(h));
title('Magnitude Response');
xlabel('Frequency (rad/s)');
ylabel('Magnitude');

% convert to digital filter using impulse invariance
```

```
figure(3);
[bz, az] = impinvar(num, den, 1 / Td);
[h, w] = freqz(bz, az, w);

plot(w, abs(h));
title('Magnitude Response');
xlabel('Frequency (rad)');
ylabel('Magnitude');
```









3.2 Lab Task 2

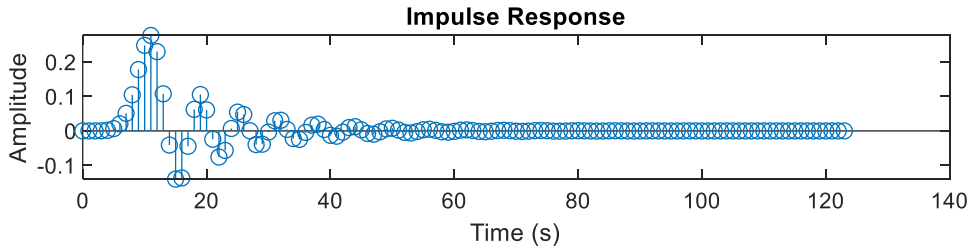
For simplicity, we now use MATLAB functions which automatically perform all digital IIR filter design steps, mentioned in the introduction. The commands are e.g. butter, cheby1 or ellip. Use the help command for more details. Say the filtering specifications for a particular job are:

Sampling frequency 200 Hz
Passband ripple _ 1 dB
Passband edge 32 Hz
Stopband edge 38 Hz
Stopband attenuation > 25 dB

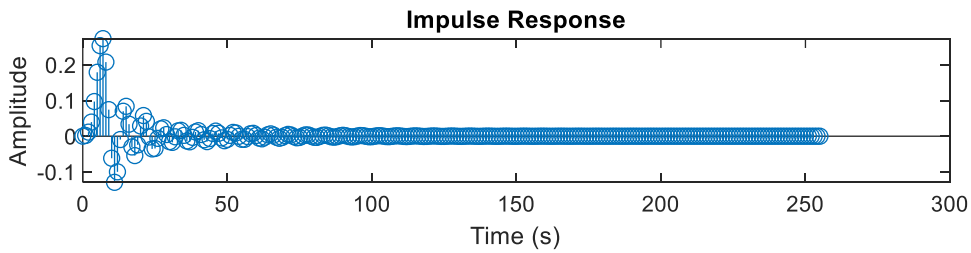
1. Determine the order of a Butterworth, Chebyshev Type I and Elliptic filters that meet these specifications? Use the buttord, cheb1ord and ellipord commands.
2. Use butter, cheby1 and ellip to design the specified filters. For each, plot the magnitude of the frequency response, the pole-zero location diagram and the significant part of the impulse response.
3. For comparison, design a FIR filter using fir1 that meets the specifications and compare results.

```
fs = 200;
Rp = 1;
Wp = 32/(fs/2);
Ws = 38/(fs/2);
Rs = 25;

% get order and cutoff
[n, Wn] = buttord(Wp, Ws, Rp, Rs);
disp(['Order: ', num2str(n)]);

% define filter
[b, a] = butter(n, Wn);

% show impulse response
[h, t] = impz(b, a);
figure(1);
subplot(311);
stem(t, h);
title('Impulse Response');
xlabel('Time (s)');
ylabel('Amplitude');

% show magnitude response
w = 0:0.01:pi;
[h, w] = freqz(b, a, w);
subplot(312);
plot(w/pi, abs(h));
title('Magnitude Response');
xlabel('Frequency (rad)');
ylabel('Magnitude');

% define discrete filter transfer function
```

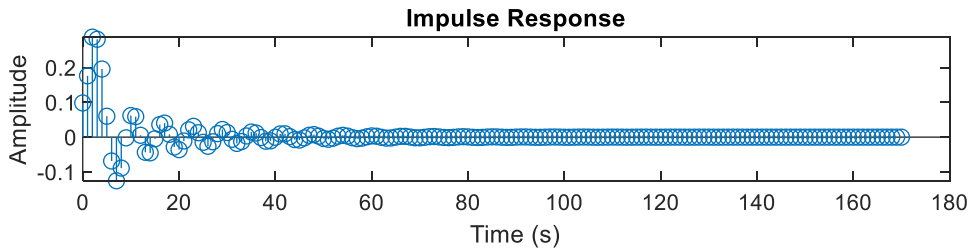
```
H = tf(b, a, -1, 'variable', 'z^-1');
subplot(313);
pzmap(H);
```




```
fs = 200;
Rp = 1;
Wp = 32/(fs/2);
Ws = 38/(fs/2);
Rs = 25;

% get order and cutoff
[n, ~] = cheb1ord(Wp, Ws, Rp, Rs);
disp(['Order: ', num2str(n)]);

% define filter
[b, a] = cheby1(n, Rp, Wp);

% show impulse response
[h, t] = impz(b, a);
figure(1);
subplot(311);
stem(t, h);
title('Impulse Response');
xlabel('Time (s)');
ylabel('Amplitude');

w = 0:0.01:pi;
[h, w] = freqz(b, a, w);
subplot(312);
plot(w/pi, abs(h));
title('Magnitude Response');
xlabel('Frequency (rad)');
ylabel('Magnitude');

% define discrete filter transfer function
H = tf(b, a, -1, 'variable', 'z^-1');
subplot(313);
pzmap(H);
```



```
%% Lab Task 2c
fs = 200;
Rp = 1;
Wp = 32/(fs/2);
Ws = 38/(fs/2);
Rs = 25;

% get order and cutoff
[n, Wn] = ellipord(Wp, Ws, Rp, Rs);
disp(['Order: ', num2str(n)]);

% define filter
[b, a] = ellip(n, Rp, Rs, Wp);

% show impulse response
[h, t] = impz(b, a);
figure(1);
subplot(311);
stem(t, h);
title('Impulse Response');
xlabel('Time (s)');
ylabel('Amplitude');

% show magnitude response
w = 0:0.01:pi;
[h, w] = freqz(b, a, w);
subplot(312);
plot(w/pi, abs(h));
title('Magnitude Response');
xlabel('Frequency (rad)');
ylabel('Magnitude');

% define discrete filter transfer function
H = tf(b, a, -1, 'variable', 'z^-1');
subplot(313);
pzmap(H);
```



4 Conclusion

In this lab, we have demonstrated the design of FIR filters using windowing. We have seen that windowing is a simple and effective method for designing FIR filters. We have also seen that linear phase filters can be designed using windowing. The main advantage of windowing is that it is a very simple method to implement. The main disadvantage of windowing is that it can introduce some distortion into the frequency response. This distortion can be minimized by using a window function with a good stopband attenuation. In general, windowing is a good choice for designing FIR filters when simplicity and speed are important considerations. However, if high performance is required, other methods of designing FIR filters, such as least squares design, may be a better choice.