Chittagong University of Engineering & Technology

EEE-496

DIGITAL SIGNAL PROCESSING SESSIONAL

Design Low Pass, High Pass and Band Pass Filter By Using Butterworth Filter.

Submitted by:
MD. Sayedul
ID: **1702079**
Section: **B**

Submitted to:
Dr. Muhammad Ahsan Ullah
Professor
Dept. of EEE

June 13, 2023

Contents

1 Objectives

The objectives of the experiment are:

1. To design a low pass Butterworth filter without using MATLAB predefined function.
2. To import voice data in MATLAB and adding noise to it.
3. To filter the noisy signal with a low pass filter to recover the original voice signal.
4. To design a high pass and band pass filter and filtered a signal from a harmonic signal.
5. To determine cross correlation between the recorded and filtered signal.

2 Filter a noisy voice signal with the help of low pass filter

2.1 Code

```
clc;
clear;
close all;
%% Read sppech wave file
[data, fs]=audioread('speech.wav');
t=(1:1:length(data))/fs;

plt=Plot(t, data);
plt.XLabel ='Time'
plt.YLabel ="Audio Signal";
plt.Title ="Speech signal with sampling rate 16000";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";

%% Frequency Specturm of Speech.wav
voice_fft = fft(data);
n = length(data);           % number of samples
y0 = fftshift(voice_fft);   % shift y values
f0 = (-n/2:n/2-1)*(fs/n);   % 0-centered frequency
  range
```

```
power0 = abs(y0).^2/n;     % 0-centered power
plt=Plot(f0, power0);
plt.Colors={[139/256 0 0]};
plt.XLabel ='Frequency'
plt.YLabel ="Power";
plt.Title ="Frequency Specturm of Voice Signal";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";

%% Add noise with speech.wav
DataWithNoise=data+0.1*randn(size(data));
plt=Plot(t, DataWithNoise);
plt.XLabel ='Time'
plt.YLabel ="Audio Signal With Noise";
plt.Title ="Speech & noise signal with sampling rate
   16000";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";
%% Frequency specturm of noisy speech.wav
y = fft(DataWithNoise);
n = length(DataWithNoise);          % number of
  samples
y0_WithNoise = fftshift(y);         % shift y values
f0_noise = (-n/2:n/2-1)*(fs/n); % 0-centered
  frequency range
PowerWithNoise = abs(y0_WithNoise).^2/n;    % 0-
  centered power
plt=Plot(f0_noise,PowerWithNoise)
plt.Colors={[139/256 0 0]};
plt.XLabel ='Frequency'
plt.YLabel ="Power";
plt.Title ="Frequency Specturm of Voice Signal With
  Noise";
plt.XGrid =" on ";
```

```
plt.YGrid =" on ";
plt.ShowBox ="off";
%% Butterworth filter design
Fs = fs;                              % Sampling Frequency
  (Hz)
Fn = Fs/2;                                     % Nyquist
  Frequency (Hz)
Wp = 0.04;                                  % Passband
  Frequency (Normalised)
Ws = .2;                                  % Stopband
  Frequency (Normalised)
Rp =15;                                      % Passband
  Ripple (dB)
Rs = 60;                                      % Stopband
  Ripple (dB)

[n,Wn] = buttord(Wp,Ws,Rp,Rs);
[b, a]= butter(n,Wn, "low");
[p, q]=freqz(b, a, 16000);
plt=Plot(q, abs(p));
plt.Colors="red";
plt.LineWidth=2;
plt.XLabel ='Frequency'
plt.YLabel ="Magnitude Plot";
plt.Title ="Low Pass Filter";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";

filtered_sound=filter(b, a, DataWithNoise);
fs2_d = [filtered_sound,filtered_sound];
sound(fs2_d, Fs)

plt=Plot(t, filtered_sound);
plt.XLabel ='Time'
plt.YLabel ="Audio Signal";
```

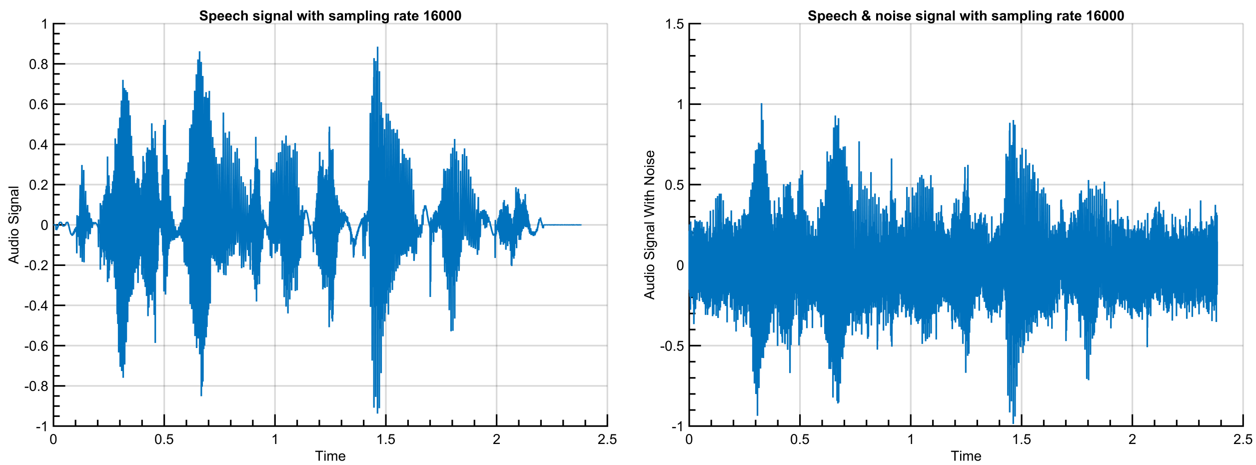
```
plt.Title ="Filtered Speech signal with sampling rate 16000";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";

filtered_fft = fft(filtered_sound);
n = length(filtered_sound);           % number of samples
f = (0:n-1)*(fs/n);      % frequency range
PowerAfterFilter = abs(filtered_fft).^2/n;     % power of the DFT
y0_filtered = fftshift(filtered_fft);          % shift y values
f0_filtered = (-n/2:n/2-1)*(fs/n); % 0-centered frequency range
PowerAfterFilter0 = abs(y0_filtered).^2/n;     % 0-centered power
plt=Plot(f0_noise,PowerAfterFilter0)
plt.Colors={[8/256 156/256 50/256]};
plt.XLabel ='Frequency'
plt.YLabel ="Power";
plt.Title ="Frequency Specturm of Filtered Voice Signal";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";

%% Cross correlation
[c, lags]=xcorr(data, filtered_sound);
plt=Plot(lags/fs, c);
plt.Colors={[62/256, 19/256, 191/256]};
plt.XLabel ='time(s)'
plt.YLabel ="Amplitude";
plt.Title ="Correlation between filtered signal & speech";
```

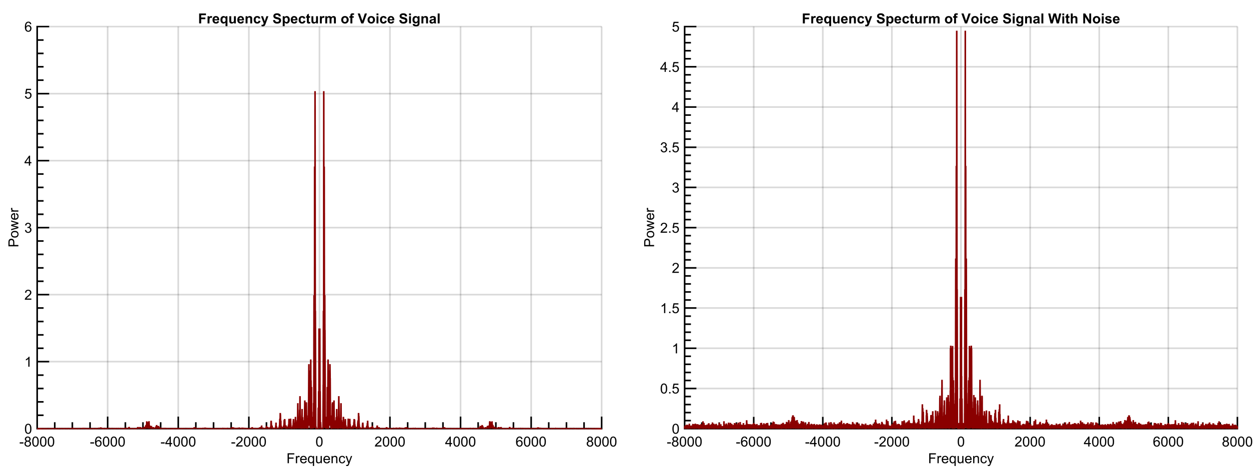
```
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";
```

2.2 Figure



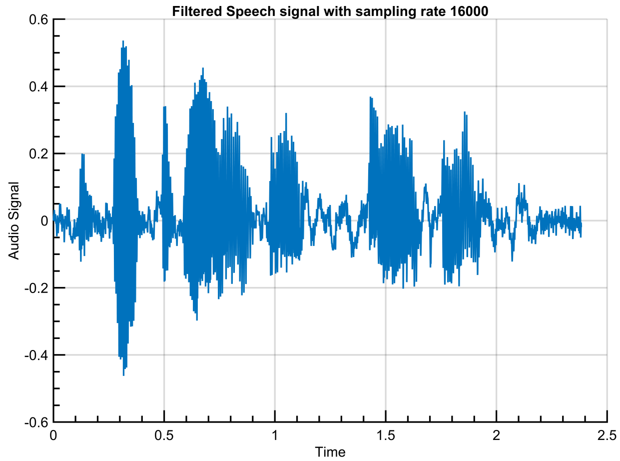
(a) (b)

Figure 1: (a) Recorded Voice signal at sampling frequency 16000 Hz. (b) Recorded Voice signal with noise at sampling frequency 16000 Hz.

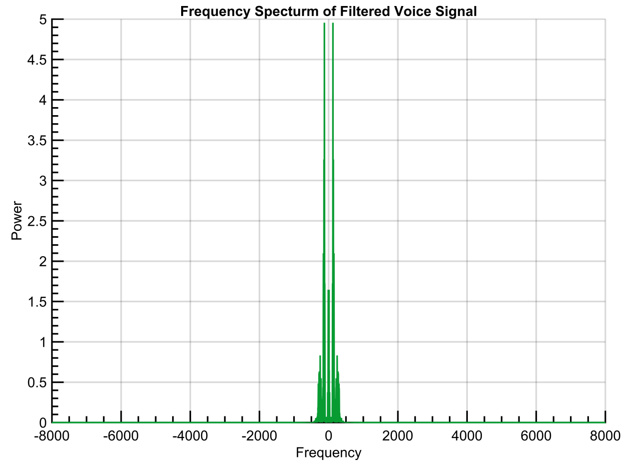


(a) (b)

Figure 2: (a) Frequency spectrum of recorded voice signal. (b) Frequency spectrum of recorded voice signal with noise.

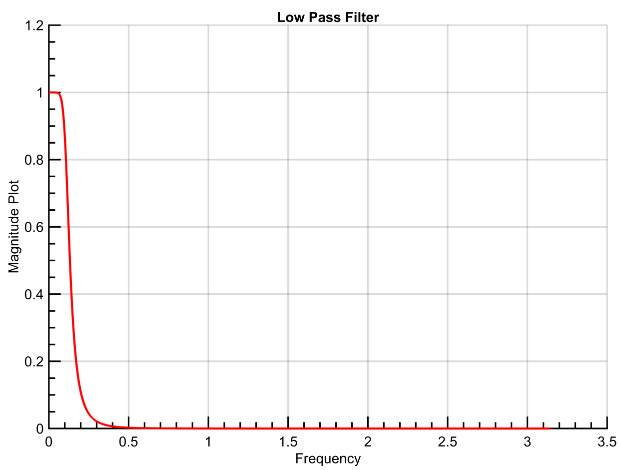


(a)

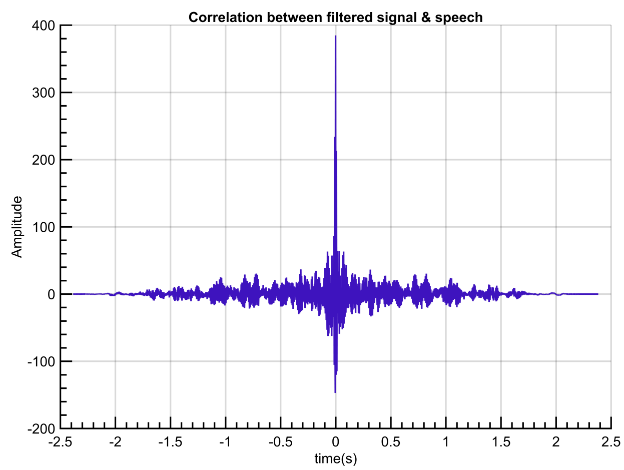


(b)

Figure 3: (a) Recovered voice signal from a noisy signal with the help of Butterworth low pass filter. (b) Frequency spectrum of filtered voice signal.



(a)



(b)

Figure 4: (a) Magnitude response of Botterworth low pass filter. (b) Cross correlation between original voice signal and filtered voice signal.

3 Design a high pass filter.

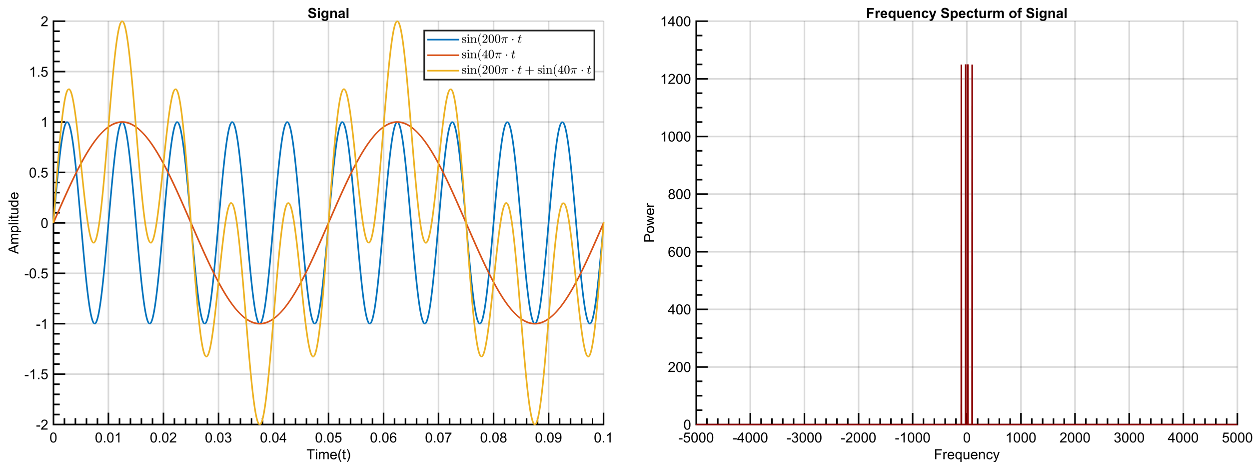
3.1 Code

```
clc;
clear;
close all;
```

```
fs = 10000;  % Sampling frequency
t = (0:1:5000) / fs;
signala = sin(2*pi*100*t);
signalb = sin(2*pi*20*t);
signalc = signala + signalb;
Fs = fs; % Sampling Frequency (Hz)
Fn = Fs/2;% Nyquist Frequency (Hz)
Wp = 80/Fn;% Passband Frequency (Normalised)
Ws = 30/Fn;% Stopband Frequency (Normalised)
Rp =3; % Passband Ripple (dB)
Rs = 20;
[n,Wn] = buttord(Wp,Ws,Rp,Rs);
[b, a]= butter(n,Wn, "high");
[p, q]=freqz(b, a, fs);
output = filter(b, a, signalc);
%% Signal plot
plt=Plot(t, signala, t, signalb, t, signalc);
leg=legend('$\sin(200\pi\cdot t$', '$\sin(40\pi\cdot
   t$',...
    '$\sin(200\pi\cdot t+\sin(40\pi\cdot t$');
set(leg , "Interpreter", "latex")
plt.XLabel ='Time(t)'
plt.YLabel ="Amplitude";
plt.Title ="Signal";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.XLim =[0 0.1 ];
plt.ShowBox ="off";
%setPlotProp(plt)
%% Frequency Specturm of signal
signal_fft = fft(signalc);
n = length(signalc);          % number of samples
y0 = fftshift(signal_fft);  % shift y values
f0 = (-n/2:n/2-1)*(fs/n);  % 0-centered frequency
  range
```

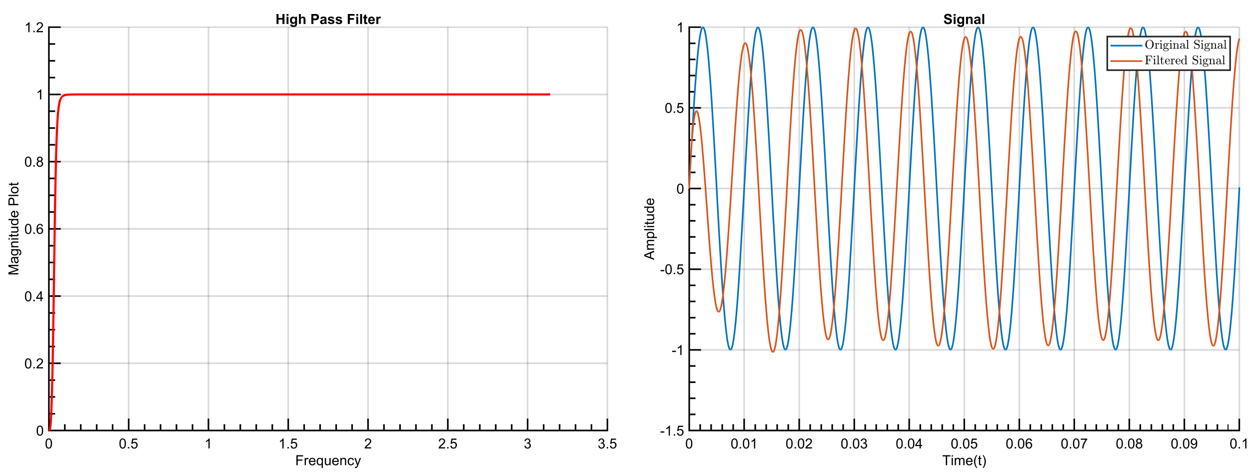
```
power0 = abs(y0).^2/n;     % 0-centered power
plt=Plot(f0, power0);
plt.Colors={[139/256 0 0]};
plt.XLabel ='Frequency'
plt.YLabel ="Power";
plt.Title ="Frequency Specturm of Signal";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";
%% Magnitude plot of a high pass filter
plt=Plot(q, abs(p));
plt.Colors="red";
plt.LineWidth=2;
plt.XLabel ='Frequency'
plt.YLabel ="Magnitude Plot";
plt.Title ="High Pass Filter";
plt.XGrid =" on ";
plt.YGrid =" on ";
plt.ShowBox ="off";
%% Filtered output
plt=Plot(t, signala, t, output)
leg=legend('Original Signal',...
    'Filtered Signal');
set(leg , "Interpreter", "latex")
plt.XLabel ='Time(t)'
plt.YLabel ="Amplitude";
plt.Title ="Signal";
plt.XGrid ="on";
plt.YGrid ="on";
plt.ShowBox ="off";
plt.XLim =[0 0.1 ];
%setPlotProp(plt)
```

3.2 Figure



(a) (b)

Figure 5: (a) A signal $\sin(200\pi \cdot t) + \sin(40\pi \cdot t)$ which is the the sum of two signal $\sin(200\pi \cdot t)$ and $\sin(40\pi \cdot t)$. (b) Frequency spectrum of signal $\sin(200\pi \cdot t) + \sin(40\pi \cdot t)$.



(a) (b)

Figure 6: (a) Frequency spectrum of filtered voice signal. (b) Recovered $\sin(200\pi \cdot t$ signal from $\sin(200\pi \cdot t + \sin(40\pi \cdot t$ signal with the help of Bitterworth high pass filter.

4 User defined low pass filter.

4.1 Code

```
function [order,w_c]]=ButterWorthFilter(w_p, w_s,
  G_p_db, G_s_db)
G_p_db=-3;
```

```
G_s_db=-25;
w_p=20;
w_s=50;
numerator_order=log((10^(-G_s_db/10)-1)/(10^(-G_p_db
  /10)-1));
denominator_order=2*log(w_s/w_p);
order=ceil(numerator_order/denominator_order);
w_c=w_p/(10^(-G_p_db/10)-1)^(1/(2*order));
```