EE386 Digital Signal Processing Lab **Jul-Dec 2021**

Experiment 6 Report

Author: J Karan Tejas *Email:* karantejas.191ee126@nitk.edu.in

The calculated value of α for the Roll Number : 191EE126 is

$$\alpha = 1 + mod(126, 3) = 1$$

Problem 1

(Butterworth filter design)

(Solution)

The specifications of the butterworth lowpass filter is given by :

- Passband ripple gain, δ_p = -α dB = -1 dB
- Passband edge frequency, $\Omega_p^{'} = \Omega_c^{'} = 10$ Hz
- Stopband ripple Attenuation, δ_s = -40 dB
- Stopband edge frequency $\Omega_s^{'} = 20$ Hz
- Sampling Frequency, $F_s = 720\ \frac{samples}{sec}$

To design the Butterworth filter using Bi-Linear Transformation, we find the pre-warping analog frequencies :

- Desired discrete time passband edge frequency, $\omega_p^{'} = \frac{\Omega_p^{'}}{F_s} = \frac{\pi}{36}\frac{rad}{sample}$
- Desired discrete time stopband edge frequency, $\omega_s^{'} = \frac{\Omega_s^{'}}{F_s} = \frac{\pi}{18}\frac{rad}{sample}$
- Pre-Warped Passband edge frequency, $\Omega_p^{''} = \frac{2}{T}tan(\frac{\omega_p^{'}}{2}) = 62.87175\frac{rad}{sec}$
- Pre-Warped Stopband edge frequency, $\Omega_s^{''} = \frac{2}{T}tan(\frac{\omega_s^{'}}{2}) = 125.9836\frac{rad}{sec}$

For a Butterworth filter,

$$\epsilon = \sqrt{\frac{1-\delta_p^2}{\delta_p^2}} = 0.7175$$

The order of the filter is given by,

$$N = \frac{log(\frac{1}{\epsilon}\sqrt{\frac{1-\delta_s^2}{\delta_s^2}})}{log(\frac{\Omega_s}{\Omega_p})} = 8$$

Transfer Function H(s) for the order 8 is given by :

$$H(s) = \frac{1}{(s^2 + 0.390181s + 1)(s^2 + 1.111140s + 1)(s^2 + 1.663s + 1)(s^2 + 1.961571s + 1)}$$

H(s') is found by the following linear transformation of H(s) :

$$s' = s\frac{\Omega_p}{\Omega'_p}$$

The *Buttord()* function of the scipy library returns the order and critical frequemcy of the filter. The *butter()* of the same library is used to find the transfer function using the critical frequency and order of the filter. Bilinear transformation of H(s') to H(z) is applied using *bilinear()* function.

$$s' \rightarrow \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}$$

The Final Transfer Function is given by

$$\frac{2.034e\text{-}11\ z^8 + 1.627e\text{-}10\ z^7 + 5.696e\text{-}10\ z^6 + 1.139e\text{-}09\ z^5 + 1.424e\text{-}09\ z^4 + 1.139e\text{-}09\ z^3 + 5.696e\text{-}10\ z^2 + 1.627e\text{-}10\ z + 2.034e\text{-}11}{z^8 - 7.513\ z^7 + 24.71\ z^6 - 46.47\ z^5 + 54.64\ z^4 - 41.14\ z^3 + 19.37\ z^2 - 5.215\ z + 0.6145}$$

Figure 1: Transfer Function

(Pole Zero Plot)

(Solution)

The poles and zeros of the transfer function are found using *tf2zpk()* from the scipy library.

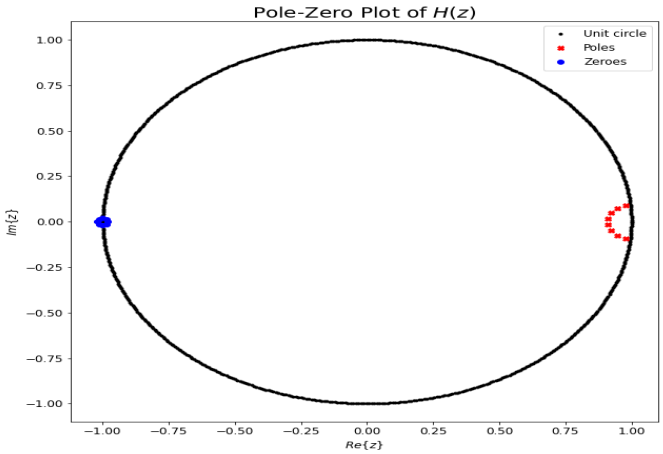

Figure 2: Pole Zero Plot

All the 8 poles of the transfer function lie inside the unit circle. Hence, we can conclude that the system is stable.

(Bode Plots)

(Solution)

The Bode Plot is plotted using *freqz()* of the scipy library which takes the transfer function as the input and returns the frequency response. The magnitude and phase is plotted against frequency. The scales are logarithmic using *semilogx()*.

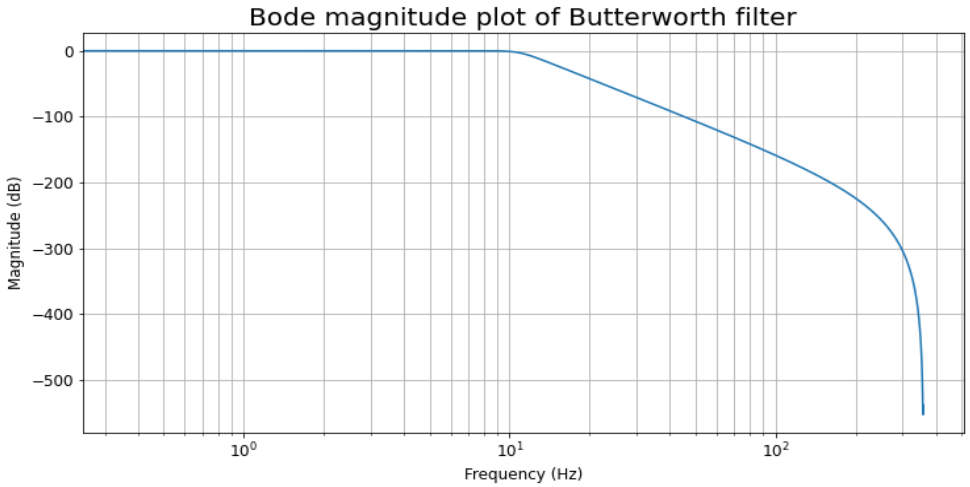

Figure 3: Bode Magnitude Plot

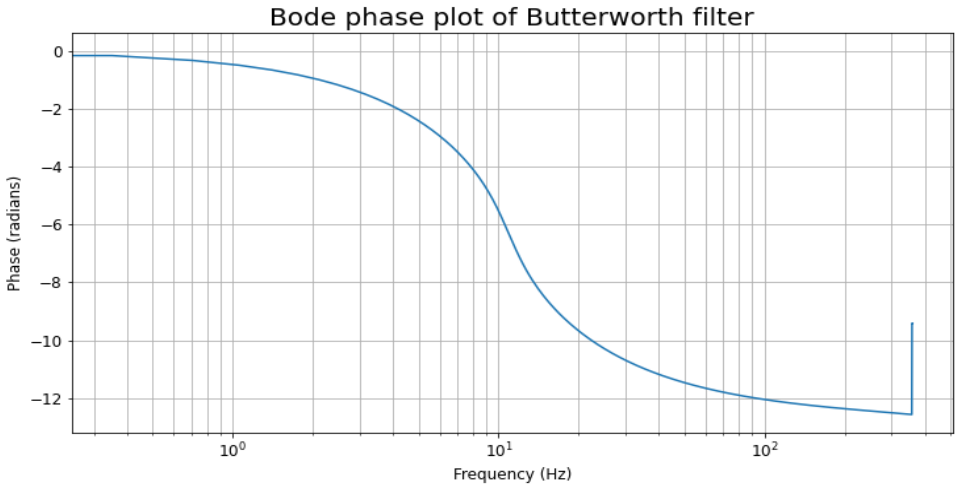

Figure 4: Bode Phase Plot

(Impulse and Step Response)

(Solution)

The impulse response and step response is found using *lfilter()* function from the scipy library which intakes the input signal and the transfer function and returns the output of the given system.

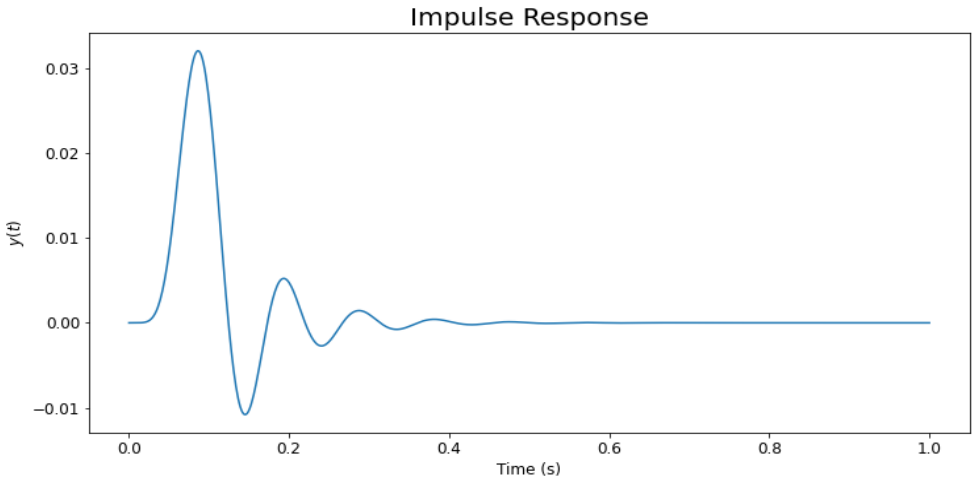

Figure 5: Impulse Response of Butterworth Filter

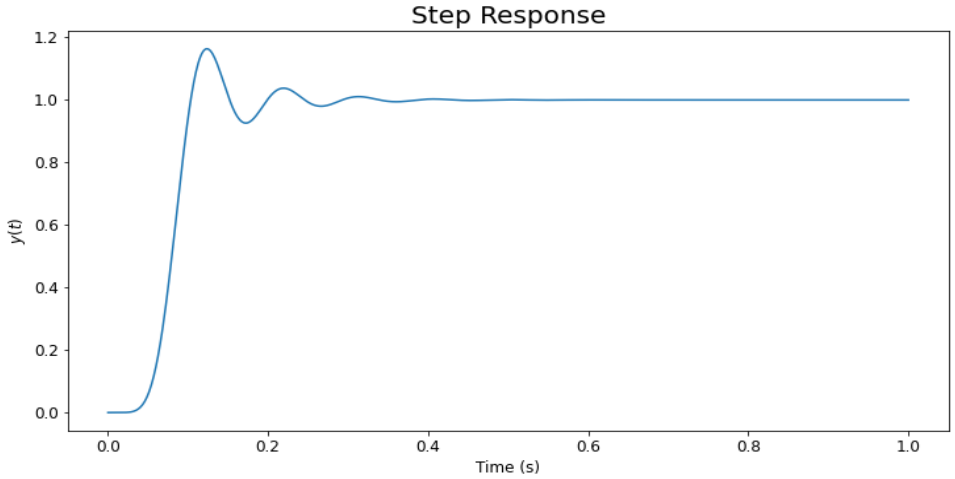

Figure 6: Step Response of Butterworth Filter

Problem 2

(Filtering Given Signal)

(Solution)

The given signal is read from the *ECG_Data.txt* file. This signal is filter using the *lfilter()* function from the scipy library using the our previous Butterworth Low Pass Filter and it is plotted against time.

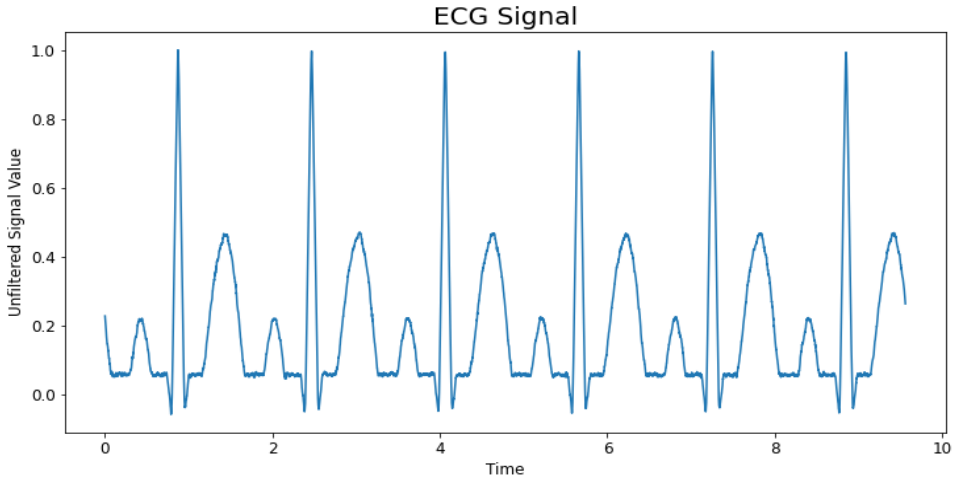

Figure 7: Original ECG Signal

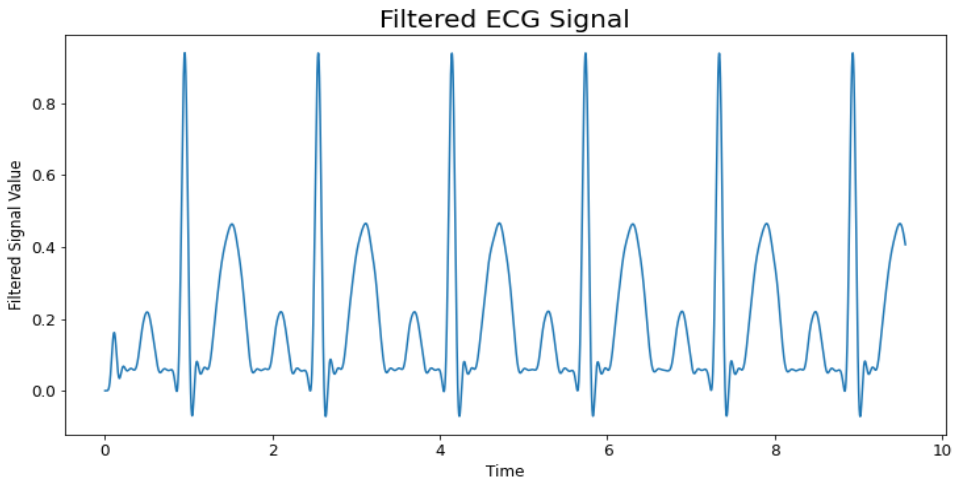

Figure 8: Filtered ECG Signal

From the figures, we can see that the high frequency components of the signal are removed. It is evident from fact the rapid changes of small magnitude is removed and hence the plot is

more smooth.

(Fourier Transform)

(Solution)

The same can be verified using fourier transform of the ECG signal and filtered signal using the emphfft() function from the scipy library.

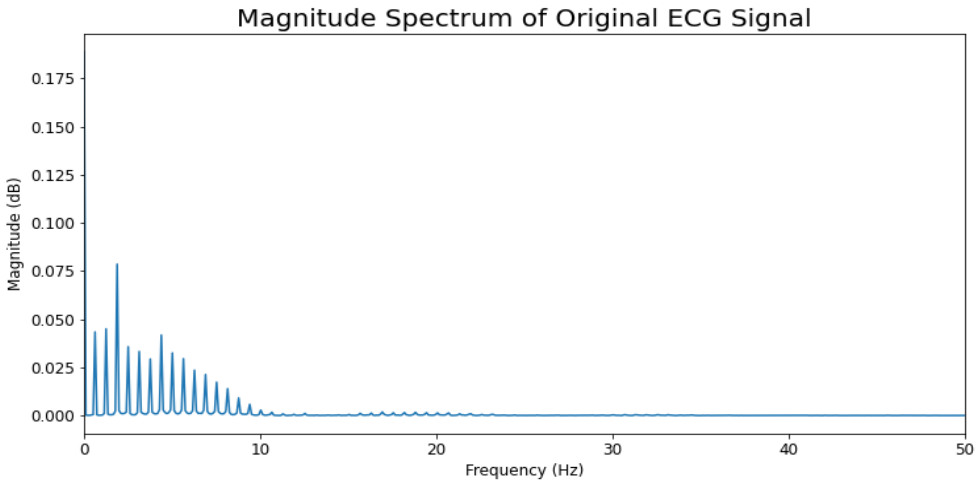

Figure 9: FFT of Original ECG Signal

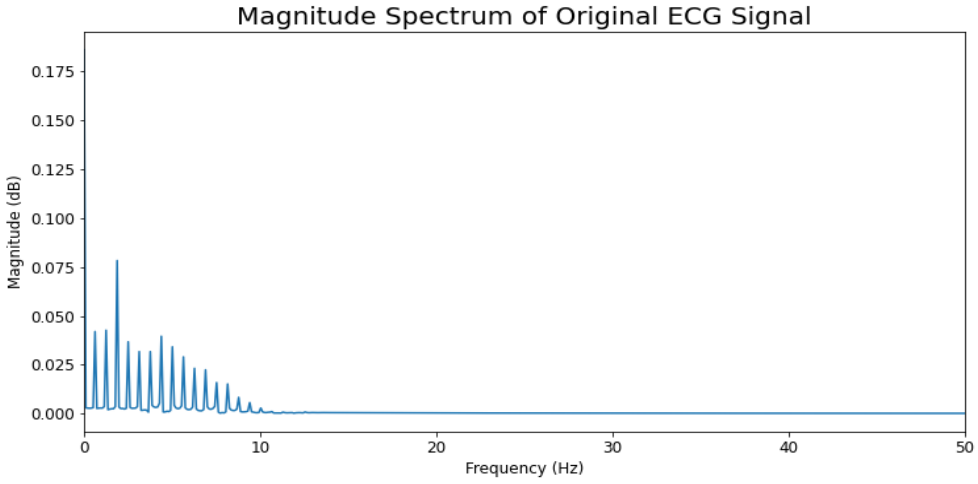

Figure 10: FFT of Filtered ECG Signal

In the magnitude spectrum it can be clearly seen that the high frequency components are removed. The frequencies before 10Hz are unaffected.

Problem 3

(Spectogram of Audio Signal)

(Solution)

The audio signal is read form the *intru1.wav* file. The spectrogram of the signal is plotted using *specgram()* function in matplotlib library using hamming window.

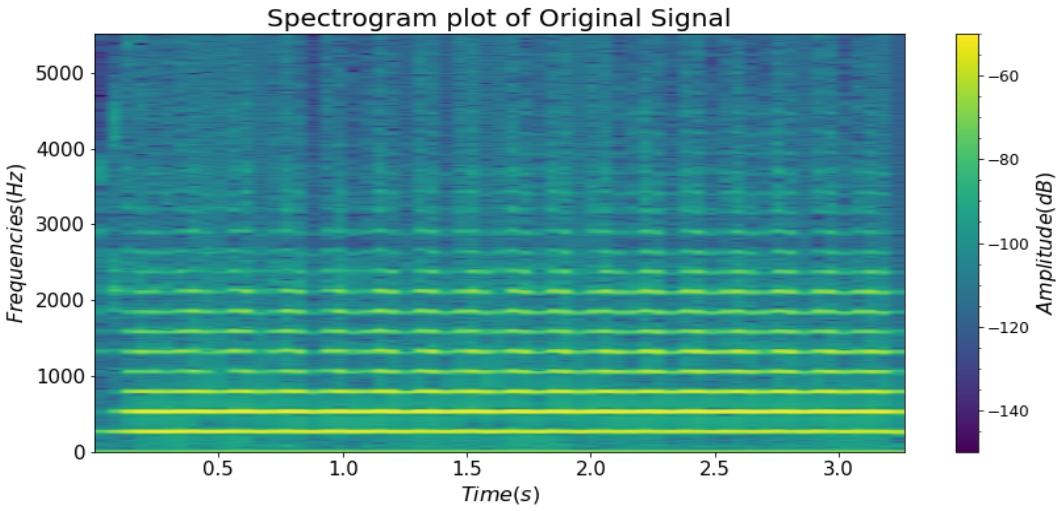

Figure 11: Spectrogram of Instru1 file

The fundamental frequency of the signal is found to be 256.6924 Hz. The Butterworth Bandpass Filter is defined using the following specifications :

- δ_p = -α dB = -1 dB
- δ_s = -40 dB
- $\Omega_l^{'} = 400\pi \frac{rad}{sec}$
- $\Omega_u^{'} = 600\pi \frac{rad}{sec}$
- $\Omega_{s1}^{'} = 200\pi \frac{rad}{sec}$
- $\Omega_{s2}^{'} = 800\pi \frac{rad}{sec}$

The given signal is filtered using the *lfilter()* function from the scipy library with the designed filter. The filtered signal is stored in *instrument_filtered.wav* file. The spectrogram of the new signal is plotted with the same window.

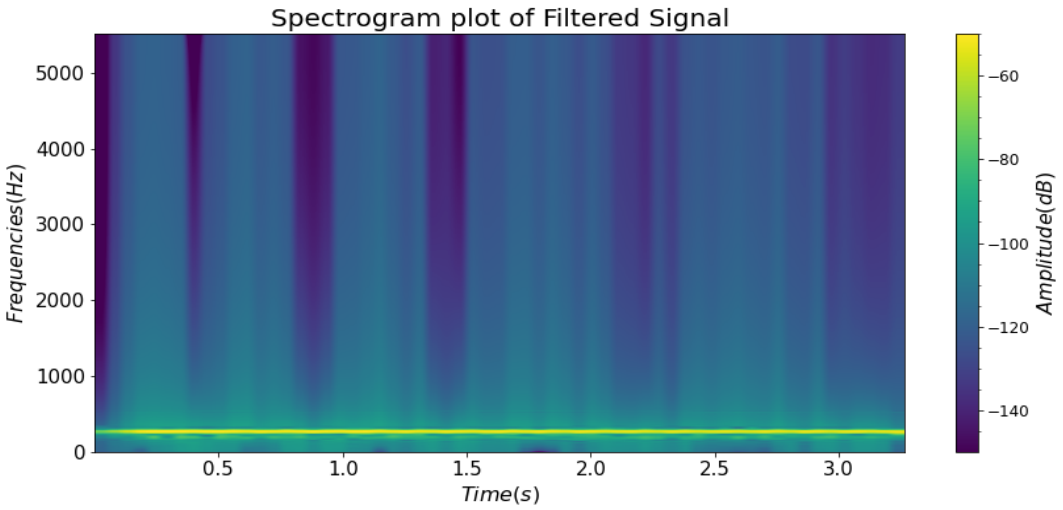

Figure 12: Spectrogram of Filtered Audio file

In the plotted spectogram we can see that all the harmonics except the first harmonic is removed. The frequency corresponding to this harmonic is 256.6924 Hz which verifies that the given filter retains the fundamental frequency.

Problem 4

(Chebyshev filter design)

(Solution)

We follow the same procedure as the Butterworth Filter to design Chebyshev Filter, but we use a few different functions. We use *cheb1ord()* and *cheby1()* instead of *buttord()* and *butter()* functions. The order of the Chebyshev Filter is given by :

$$N = \frac{cosh^{-1}(\frac{1}{\epsilon}\sqrt{\frac{1-\delta_s^2}{\delta_s^2}})}{cosh^{-1}(\frac{\Omega_s}{\Omega_p})} = 5$$

The bilinear transformation of the transfer functin is done using *bilinear()* function in scipy library.

(Poles Zeros Plot)

(Solution)

The poles and zeros of the transfer function are found using *tf2zpk()* from the scipy library.

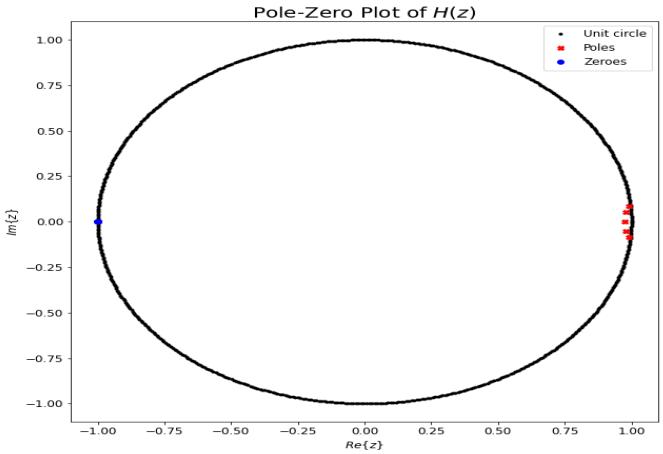

Figure 13: Pole Zero Plot

(Bode Plot)

(Solution)

The Bode Plot is plotted using *freqz()* of the scipy library. The scales are logarithmic using *semilogx()*.

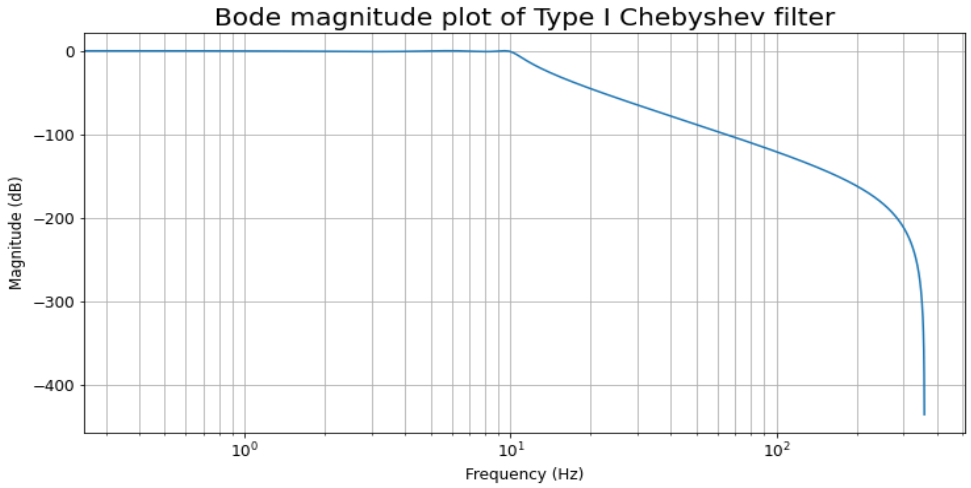

Figure 14: Bode Magnitude Plot

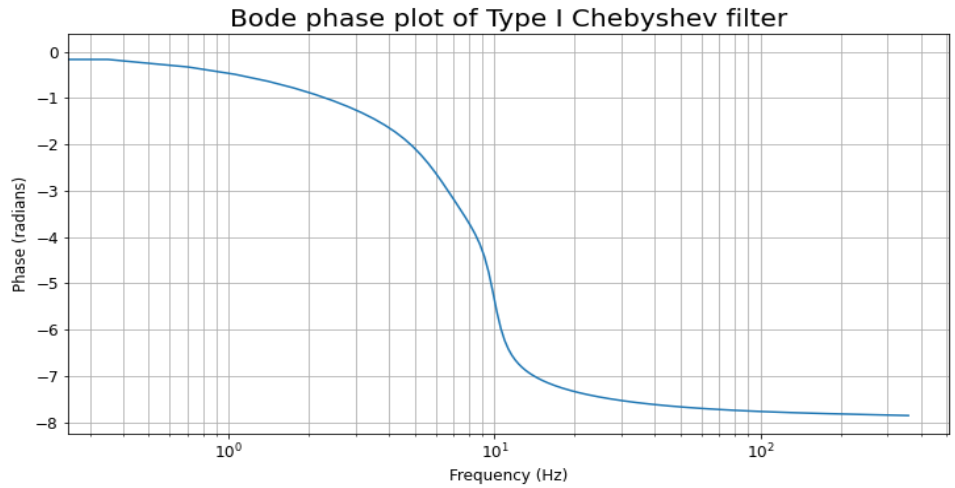

Figure 15: Bode Phase Plot

(Impulse and Step Response)

(Solution)

The impulse and step signals are filtered using *lfilter()* function from the scipy library.

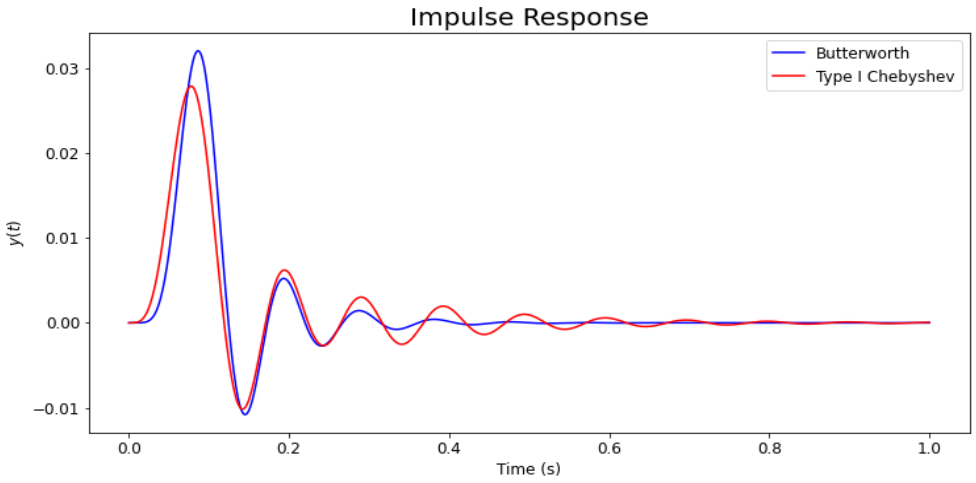

Figure 16: Impulse Response of both filters

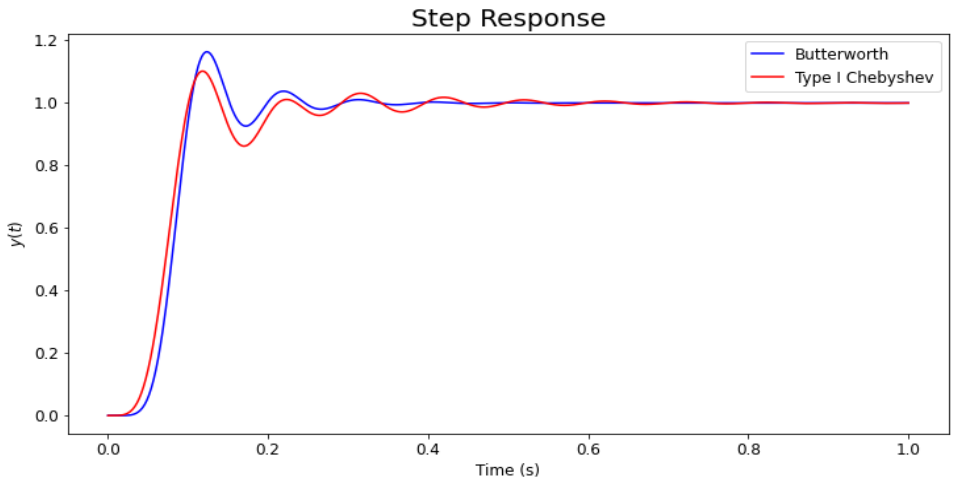

Figure 17: Step Response of both filters

In both the plots there is an overshoot, but the overshoot in the Butterworth filter is higher than that of the Chebyshev filter. But when we consider the settling time, the Butterworth takes lesser time when compared the Chebyshev filter.

Code Repository

The code, input and output of all the problems is in the following repository : `https://github.com/KaranTejas/DSP-Lab/tree/main/Experiment6`.