Audio Filter

EE23BTECH11038 - Rohith Madhani*

I. Digital Filter

I.1 The audio file is obtained from the link given below

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/Rohith_Singing.wav

I.2 A python code is written to filter the audio.

```
import soundfile as sf
import numpy as np
from scipy import signal
#read .wav file
input_signal,fs = sf.read('Rohith_Singing.wav')

#sampling frequency of Input signal
sampl_freq=fs


#order of the filter
order=4

#cutoff frquency
cutoff_freq=1000.0

#digital frequency
Wn=2*cutoff_freq/sampl_freq


# b and a are numerator and denominator polynomials respectively
b, a = signal.butter(order, Wn, 'low')
print(b)
print(a)

#filter the input signal with butterworth filter
output_signal = signal.filtfilt(b, a, input_signal,padlen=1)
#output_signal = signal.lfilter(b, a, input_signal)

#write the output signal into .wav file
sf.write('Sound_With_ReducedNoise.wav', output_signal, fs)
```

I.3 The original audio file and the filtered audio file are analyzed using the spectrogram at https://academo.org/demos/spectrum-analyzer
The yellow and orange regions respresent sound having high intensities and the black region represents sound having very low frequencies.

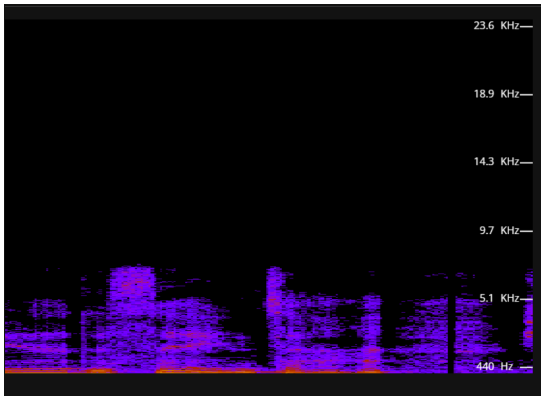

Fig. 1. Spectrogram of the original audio file

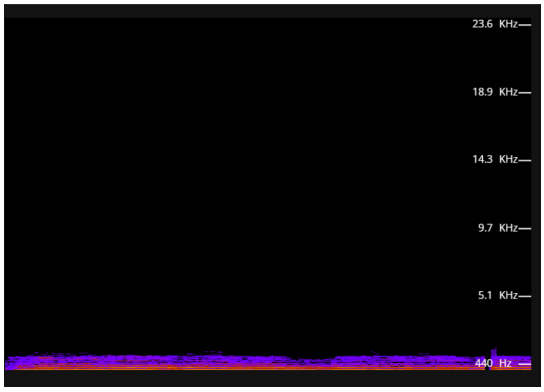

Fig. 2. Spectrogram of the audio file after Filtering

II. Difference Equation

II.1 Let

$$x(n) = \left\{ \underset{\uparrow}{1}, 2, 3, 4, 2, 1 \right\} \tag{1}$$

Sketch $x(n)$.
Solution: The python code given below plots $x(n)$

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/2_1.py

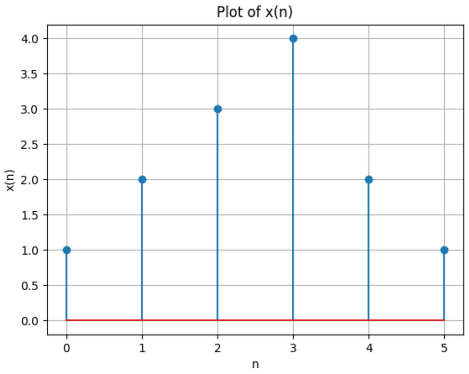

Fig. 3. Plot of $x(n)$

II.2 Let

$$y(n) + \frac{1}{2}y(n-1) = x(n) + x(n-2), \quad y(n) = 0, n < 0 \tag{2}$$

Sketch $y(n)$.
Solution: The C code given below generates the data points into a text file.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/2_2.c

The above text file generated is used to plot in the python code given below.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/2_2.py

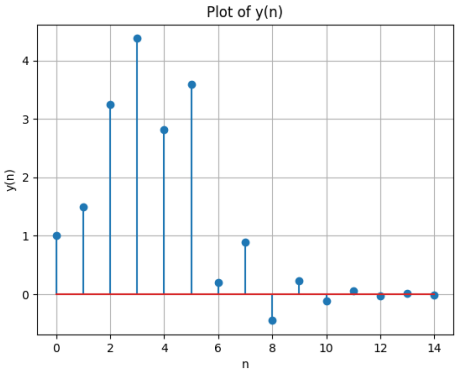

Fig. 4. Plot of $y(n)$

III. Z-TRANSFORM

III.1 The Z-transform of $x(n)$ is defined as

$$X(z) = \mathcal{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{3}$$

Show that

$$\mathcal{Z}\{x(n-1)\} = z^{-1}X(z) \tag{4}$$

and find

$$\mathcal{Z}\{x(n-k)\} \tag{5}$$

Solution: From (3),

$$\mathcal{Z}\{x(n-k)\} = \sum_{n=-\infty}^{\infty} x(n-1)z^{-n} \tag{6}$$

$$= \sum_{n=-\infty}^{\infty} x(n)z^{-n-1} = z^{-1}\sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{7}$$

resulting in (4). Similarly, it can be shown that

$$\mathcal{Z}\{x(n-k)\} = z^{-k}X(z) \tag{8}$$

III.2 Find

$$H(z) = \frac{Y(z)}{X(z)} \tag{9}$$

from (2) assuming that the Z-transform is a linear operation.
Solution: Applying (8) in (2),

$$Y(z) + \frac{1}{2}z^{-1}Y(z) = X(z) + z^{-2}X(z) \tag{10}$$

$$\implies \frac{Y(z)}{X(z)} = \frac{1+z^{-2}}{1+\frac{1}{2}z^{-1}} \tag{11}$$

III.3 Find the Z transform of

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

and show that the Z-transform of

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

is

$$U(z) = \frac{1}{1 - z^{-1}}, \quad |z| > 1 \tag{14}$$

Solution: It is easy to show that

$$\delta(n) \overset{\mathcal{Z}}{\leftrightarrow} 1 \tag{15}$$

and from (13),

$$U(z) = \sum_{n=0}^{\infty} z^{-n} \tag{16}$$

$$= \frac{1}{1 - z^{-1}}, \quad |z| > 1 \tag{17}$$

using the fomula for the sum of an infinite geometric progression.

III.4 Show that

$$a^n u(n) \overset{\mathcal{Z}}{\leftrightarrow} \frac{1}{1 - az^{-1}} \quad |z| > |a| \tag{18}$$

Solution:

$$a^n u(n) \overset{\mathcal{Z}}{\leftrightarrow} \sum_{n=0}^{\infty} \left(az^{-1}\right)^n \tag{19}$$

$$= \frac{1}{1 - az^{-1}} \quad |z| > |a| \tag{20}$$

III.5 Let

$$H\left(e^{\mathrm{J}\omega}\right) = H\left(z = e^{\mathrm{J}\omega}\right). \tag{21}$$

Plot $|H\left(e^{\mathrm{J}\omega}\right)|$. Comment. $H(e^{\mathrm{J}\omega})$ is known as the *Discret Time Fourier Transform* (DTFT) of $x(n)$.

Solution: The following code plots $|H\left(e^{\mathrm{J}\omega}\right)|$.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/3_5.py

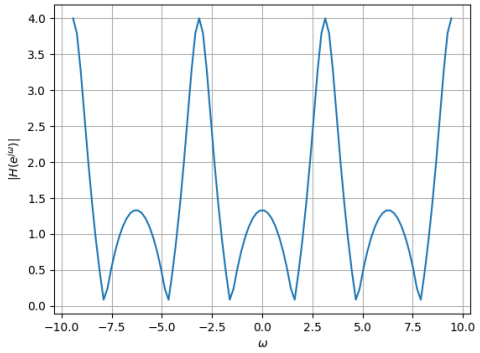

Fig. 5. $|H\left(e^{\mathrm{J}\omega}\right)|$

By substituting $z = e^{\mathrm{J}\omega}$ in (11), we get

$$\left|H\left(e^{j\omega}\right)\right| = \left|\frac{1 + e^{-2j\omega}}{1 + \frac{1}{2}e^{-j\omega}}\right| \tag{22}$$

$$= \sqrt{\frac{(1 + \cos 2\omega)^2 + (\sin 2\omega)^2}{\left(1 + \frac{1}{2}\cos\omega\right)^2 + \left(\frac{1}{2}\sin\omega\right)^2}} \tag{23}$$

$$= \frac{4|\cos\omega|}{\sqrt{5 + 4\cos\omega}} \tag{24}$$

By substituting $\omega + 2\pi$ in place of ω

$$\left|H\left(e^{j(\omega+2\pi)}\right)\right| = \frac{4|\cos(\omega + 2\pi)|}{\sqrt{5 + 4\cos(\omega + 2\pi)}} \tag{25}$$

$$= \frac{4|\cos\omega|}{\sqrt{5 + 4\cos\omega}} \tag{26}$$

$$= \left|H\left(e^{j\omega}\right)\right| \tag{27}$$

Therefore its fundamental period is 2π , which verifies that *Discret Time Fourier Transform* (DTFT) of a signal is always periodic.

IV. Impulse Response

IV.1 Find an expression for $h(n)$ using $H(z)$, given that

$$h(n) \overset{\mathcal{Z}}{\leftrightarrow} H(z) \tag{28}$$

and there is a one to one relationship between $h(n)$ and $H(z)$. $h(n)$ is known as the *impulse response* of the system defined by (I.2).

Solution: From (11),

$$H(z) = \frac{1}{1 + \frac{1}{2}z^{-1}} + \frac{z^{-2}}{1 + \frac{1}{2}z^{-1}} \tag{29}$$

$$\implies h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \tag{30}$$

using (18) and (8).

IV.2 Sketch $h(n)$. Is it bounded? Convergent?

Solution: The following code plots $h\left(n\right)$

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/4_2.py

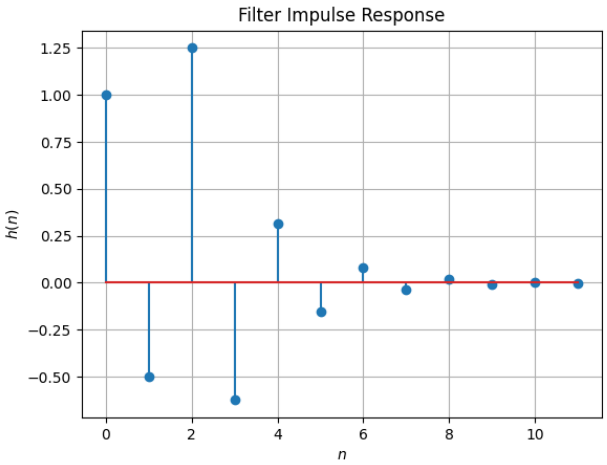

Fig. 6. $h(n)$ as the inverse of $H(z)$

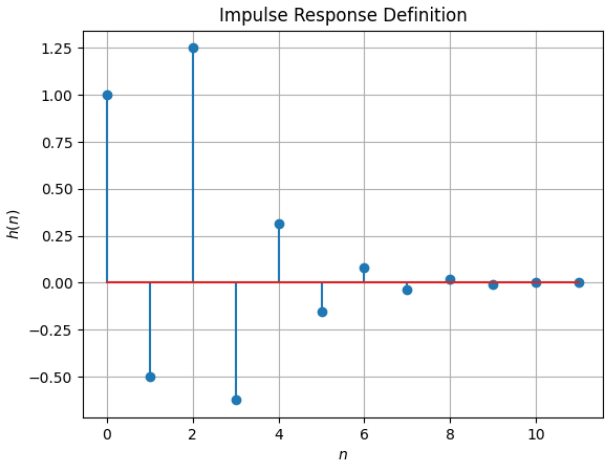

Fig. 7. $h(n)$ from the definition is same as Fig. 6

$h(n)$ is bounded and convergent.

IV.3 The system with $h(n)$ is defined to be stable if

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \tag{31}$$

Is the system defined by (2) stable for the impulse response in (28)?

Solution: To check if the system is stable,we will use ratio test of convergence

$$\lim_{n\to\infty} \left| \frac{h(n+1)}{h(n)} \right| < 1 \tag{32}$$

For $n \to \infty$,

$$u(n) = u(n-2) = 1 \tag{33}$$

$$\lim_{n\to\infty} \left(\frac{h(n+1)}{h(n)} \right) = 1/2 < 1 \tag{34}$$

Therefore $h(n)$ coneverges.Hence it is stable.

IV.4 Compute and sketch $h(n)$ using

$$h(n) + \frac{1}{2} h(n-1) = \delta(n) + \delta(n-2), \tag{35}$$

This is the definition of $h(n)$.

Solution:

Definition of $h(n)$: The output of the system when $\delta(n)$ is given as input.

The following code plots Fig. 7. Note that this is the same as Fig. 6.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/4_4.py

IV.5 Compute

$$y(n) = x(n) * h(n) = \sum_{n=-\infty}^{\infty} x(k)h(n-k) \tag{36}$$

Comment. The operation in (36) is known as *convolution*.

Solution: The following code plots Fig. 8. Note that this is the same as $y(n)$ in Fig. 4.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/4_5.py

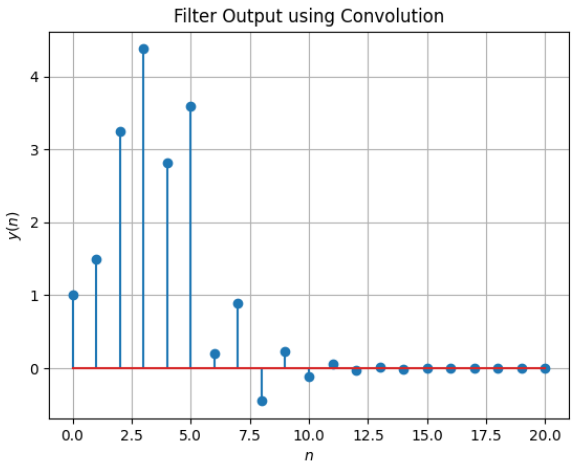

Fig. 8. $y(n)$ from the definition of convolution

IV.6 Show that

$$y(n) = \sum_{n=-\infty}^{\infty} x(n-k)h(k) \tag{37}$$

Solution: In (36),we replace k with $n-k$

$$y(n) = \sum_{n=-\infty}^{\infty} x(k)h(n-k) \tag{38}$$

$$y(n) = \sum_{n-k=-\infty}^{\infty} x(n-k)h(k) \tag{39}$$

$$\implies y(n) = \sum_{n=-\infty}^{\infty} x(n-k)h(k) \tag{40}$$

V. DFT AND FFT

V.1 Compute

$$X(k) \stackrel{\triangle}{=} \sum_{n=0}^{N-1} x(n)e^{-\mathrm{j}2\pi kn/N}, \quad k = 0, 1, \ldots, N-1 \tag{41}$$

and $H(k)$ using $h(n)$.

V.2 Compute

$$Y(k) = X(k)H(k) \tag{42}$$

V.3 Compute

$$y\,(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y\,(k)\cdot e^{\mathrm{j}2\pi kn/N}, \quad n = 0, 1, \ldots, N-1 \tag{43}$$

Solution: The following code plots Fig. 8. Note that this is the same as $y(n)$ in Fig. 4. The above three questions are solved by the code given below

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/5.py

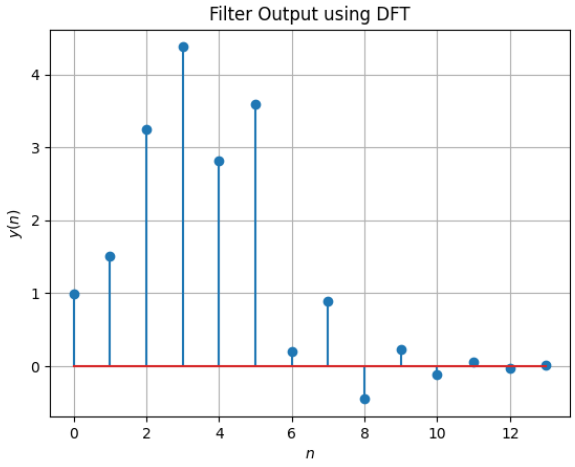

Fig. 9. $y(n)$ from the DFT

V.4 Repeat the previous exercise by computing $X(k), H(k)$ and $y(n)$ through FFT and IFFT.

Solution: The solution is given in the code below.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/5_4.py

The above code given verifies the result by plotting the obtained result with the one obtained from IDFT.

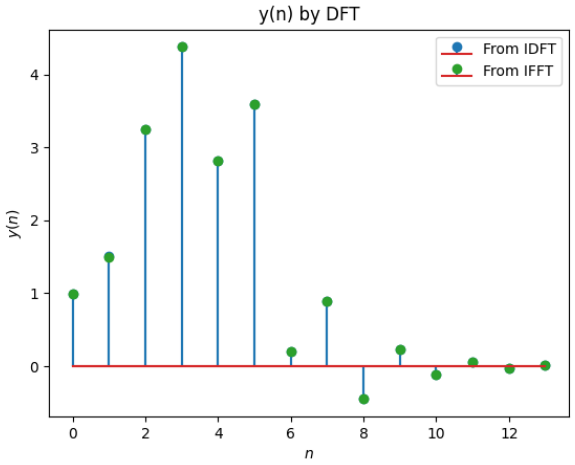

Fig. 10. $y(n)$ from the IDFT and IFFT are plotted and verified

V.5 Wherever possible, express all the above equations as matrix equations.

Solution: The DFT matrix is defined as :

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \ldots & \omega^0 \\ \omega^0 & \omega^1 & \ldots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \ldots & \omega^{(N-1)(N-1)} \end{pmatrix} \tag{44}$$

where $\omega = e^{-\frac{j2\pi}{N}}$. Now any DFT equation can be written as

$$\mathbf{X} = \mathbf{W}\mathbf{x} \tag{45}$$

where

$$\mathbf{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix} \tag{46}$$

$$\mathbf{X} = \begin{pmatrix} X(0) \\ X(1) \\ \vdots \\ X(n-1) \end{pmatrix} \tag{47}$$

Thus we can rewrite (42) as:

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{H} = (\mathbf{Wx}) \odot (\mathbf{Wh}) \tag{48}$$

where the $\odot$ represents the Hadamard product which performs element-wise multiplication. The below code computes $y(n)$ by DFT Matrix and then plots it.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/5_5.py

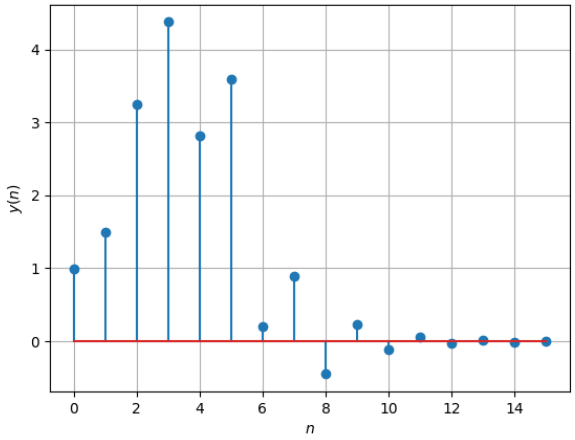

Fig. 11. $y(n)$ obtained from DFT Matrix

VI. Exercises

Answer the following questions by looking at the python code in Problem I.2.

VI.1 The command

```
output_signal = signal.lfilter(b, a, input_signal)
```

in Problem I.2 is executed through the following difference equation

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \tag{49}$$

where the input signal is $x(n)$ and the output signal is $y(n)$ with initial values all 0. Replace **signal.filtfilt** with your own routine and verify.
Solution: The below code gives the output of an Audio Filter without using the built in function signal.lfilter.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6_1.py

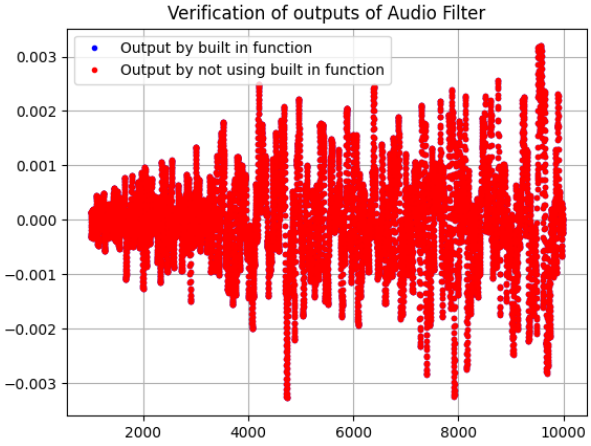

Fig. 12. Both the outputs using and without using function overlap

VI.2 Repeat all the exercises in the previous sections for the above a and b.
Solution: The code in I.2 generates the values of a and b which can be used to generate a difference equation.
And,

$$M = 5 \tag{50}$$
$$N = 5 \tag{51}$$

From 49

$$a(0)\, y(n) + a(1)\, y(n-1) + a(2)\, y(n-2) + a(3) \tag{52}$$
$$y(n-3) + a(4)\, y(n-4) = b(0)\, x(n) + b(1)\, x(n-1)$$
$$+ b(2)\, x(n-2) + b(3)\, x(n-3) + b(4)\, x(n-4)$$

Difference Equation is given by :

$$y(n) - (3.66)\, y(n-1) + (5.05)\, y(n-2)$$
$$- (3.099)\, y(n-3) + (0.715)\, y(n-4)$$
$$= \left(1.45 \times 10^{-5}\right) x(n) + \left(5.74 \times 10^{-5}\right) x(n-1)$$
$$+ \left(8.62 \times 10^{-5}\right) x(n-2) + \left(5.74 \times 10^{-5}\right) x(n-3)$$
$$+ \left(1.43 \times 10^{-5}\right) x(n-4) \tag{53}$$

From (49)

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-N}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-M}} \quad (54)$$

$$H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{\sum_{k=0}^{M} a(k) z^{-k}} \quad (55)$$

Partial fraction on (55) can be generalised as:

$$H(z) = \sum_i \frac{r(i)}{1 - p(i) z^{-1}} + \sum_j k(j) z^{-j} \quad (56)$$

Now,

$$a^n u(n) \longleftrightarrow Z \frac{1}{1 - a z^{-1}} \quad (57)$$

$$\delta(n-k) \longleftrightarrow Z z^{-k} \quad (58)$$

Taking inverse z transform of (56) by using (57) and (58)

$$h(n) = \sum_i r(i)[p(i)]^n u(n) + \sum_j k(j)\delta(n-j) \quad (59)$$

The below code computes the values of $r(i), p(i), k(i)$ and plots $h(n)$

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6_2.py

$r(i)$	$p(i)$	$k(i)$
$0.0590681 - 0.14379042j$	0.8869974 +0.04376584j	2.00×10^{-5}
$0.0590681 + 0.14379042j$	0.8869974 -0.04376584j	–
$-0.05907096 + 0.02466936j$	0.94551962+0.11263131j	–
$-0.05907096 - 0.02466936j$	0.94551962-0.11263131j	–

TABLE 1
VALUES OF $r(i), p(i), k(i)$

Stability of h(n):
According to (31)

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} \quad (60)$$

$$H(1) = \sum_{n=0}^{\infty} h(n) = \frac{\sum_{k=0}^{N} b(k)}{\sum_{k=0}^{M} a(k)} < \infty \quad (61)$$

As both $a(k)$ and $b(k)$ are finite length sequences they converge.
The below code plots Filter frequency response

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6_filter_response.py

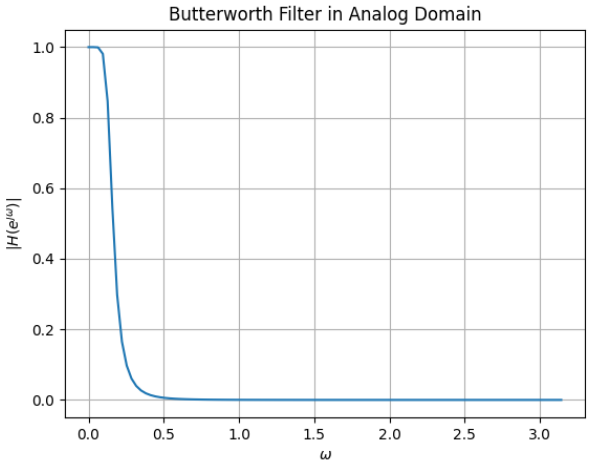

Fig. 13. Frequency Response of Audio Filter

The below code plots the Butterworth Filter in analog domain by using bilinear transform.

$$z = \frac{1 + sT/2}{1 - sT/2} \quad (62)$$

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/analog_filt.py

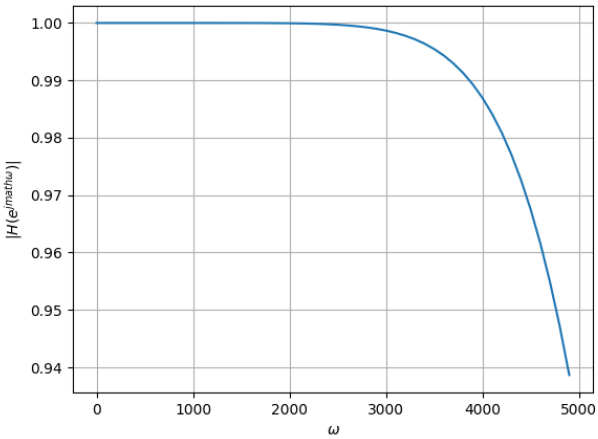

Fig. 14. Butterworth Filter Frequency response in analog domain

The below code plots the Pole-Zero Plot of the frequency response.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6_2_pole-zero.py

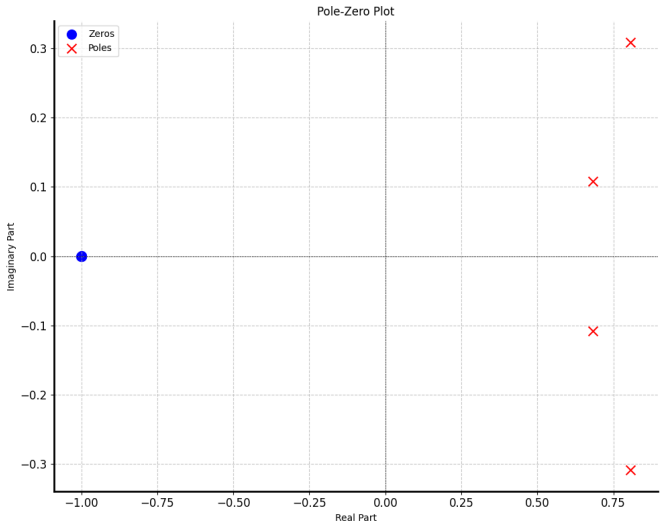

Fig. 15. There are complex poles. So $h(n)$ should be damped sinusoid.

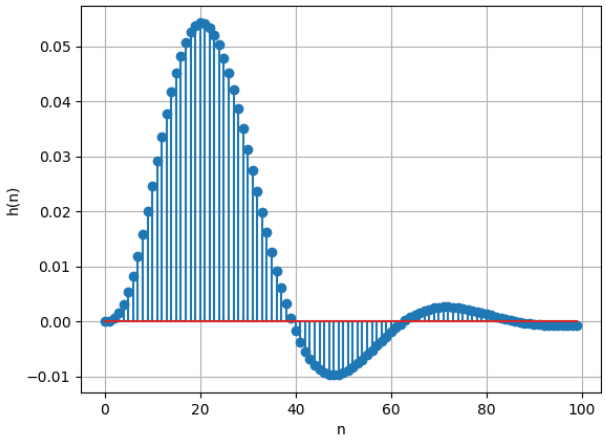

Fig. 16. h(n) of Audio Filter.It is a damped sinusoid.

VI.3 Implement your own fft routine in C and call this fft in python.

Solution: The below C code computes FFT of a given sequence.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/fft.c

The C function involved in computing the FFT is called in the below python code and the result is computed.

Before executing the python code. Execute the following command.

```
gcc −shared −o fft.so −fPIC fft.c
```

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/fft_python.py

VI.4 Find the time complexities of computing y(n) using FFT/IFFT and convolution and Compare.
Solution: The time required to compute y(n) using these two methods is calculated and the data is stored in a text file using the below C code.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6.4_time.c

The below python code extracts the data from these text files and plots Time vs n for comparison.

https://github.com/RohithMadhani/EE1205/blob/main/Audio_Filter/codes/6.4_plot.py

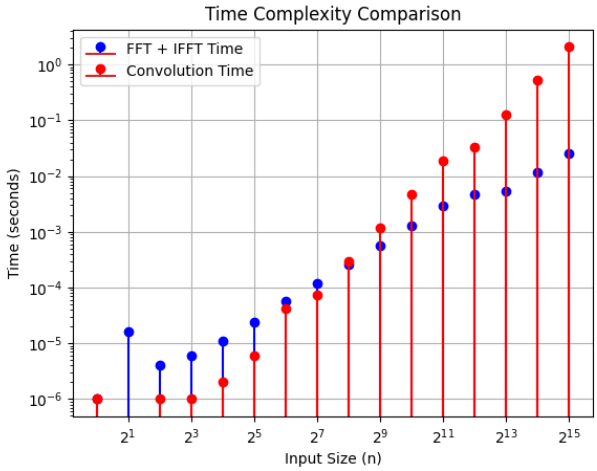

Fig. 17. The Complexity of FFT+IFFT method is $O(nlogn)$ where as by convolution is $O(n^2)$

VI.5 What is the sampling frequency of the input signal?
Solution: The Sampling Frequency is 44.1KHz

VI.6 What is type, order and cutoff-frequency of the above butterworth filter
Solution: The given butterworth filter is low-pass with order=4 and cutoff-frequency=1kHz.

VI.7 Modify the code with different input parameters and get the best possible output.
Solution: A better filtering was found on setting the order of the filter to be 5.