Audio Filter

EE23BTECH11040 - Manoj kumar Ambatipudi*

I. Digital Filter

I.1 The sound file used for this code is obtained from the below link

```
https://github.com/Manojkumar−ambatipudi/EE1205/blob/mainmain/Audio_Filter/codes/Manoj_Singing.wav
```

I.2 A Python Code is written to achieve Audio Filtering

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

#sampling frequency of Input signal
sampl_freq=fs


#order of the filter
order=4

#cutoff frquency
cutoff_freq=4000.0

#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 audio file is analyzed using Librosa Library in Python3.
It is a contour plot depicting the Intensity as a function of frequency and time of the signal. The darker region depicts no sound and the yellow region depicts high sound region.

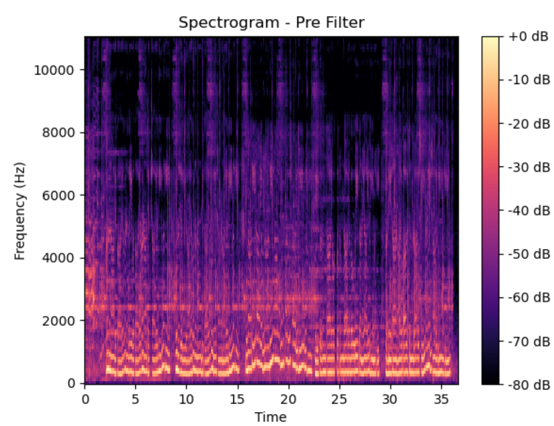

Fig. 1. Spectrogram of the audio file before Filtering

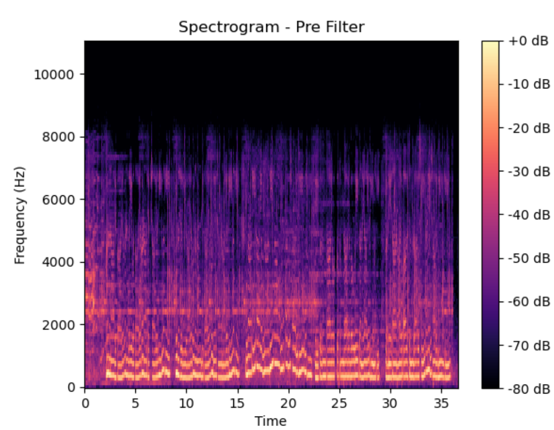

Fig. 2. Spectrogram of the audio file after Filtering

II. Difference Equation

II.1 Let

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

Sketch $x(n)$.

II.2 Let

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

Sketch $y(n)$.
Solve
Solution: The C code calculates $y(n)$ and generates values in a text file.

```
https://github.com/Manojkumar−ambatipudi/
    EE1205/blob/mainmain/Audio_Filter/
    codes/2_2.c
```

The following code plots (1) and (2)

```
https://github.com/Manojkumar−ambatipudi/
    EE1205/blob/mainmain/Audio_Filter/
    codes/2.2.py
```

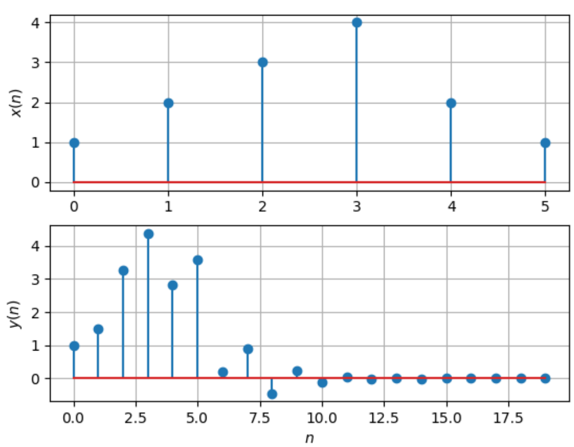

Fig. 3. Plot of $x(n)$ and $y(n)$

III. Z-Transform

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

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

Show that

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

and find

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

Solution: From (3),

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

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

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

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

III.2 Find

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

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

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

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

III.3 Find the Z transform of

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

and show that the Z-transform of

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

is

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

Solution: It is easy to show that

$$\delta(n) \overset{\mathcal{Z}}{\longleftrightarrow} 1 \quad (15)$$

and from (13),

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

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

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

III.4 Show that

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

Solution:

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

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

III.5 Let

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

Plot $|H(e^{j\omega})|$. Comment. $H(e^{j\omega})$ is known as the *Discret Time Fourier Transform* (DTFT) of $x(n)$.
Solution: The following code plots the magnitude of transfer function.

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/3.5.
    py
```

Substituting $z = e^{j\omega}$ in (11), we get

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

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

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

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

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

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

Therefore its fundamental period is 2π , which verifies that DTFT of a signal is always periodic.

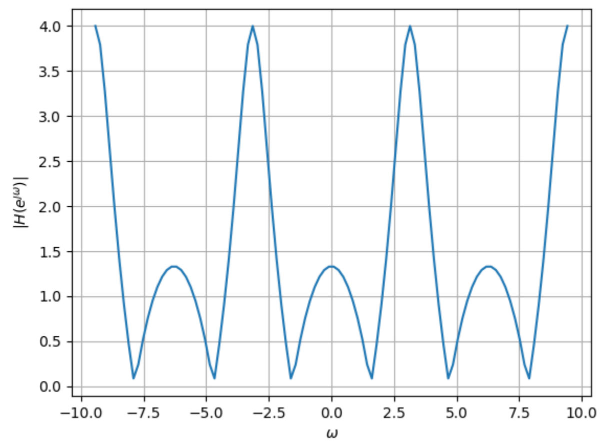

Fig. 4. $\left|H\left(e^{j\omega}\right)\right|$

IV. Impulse Response

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

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

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 (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 $h(n)$

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/4.2.
    py
```

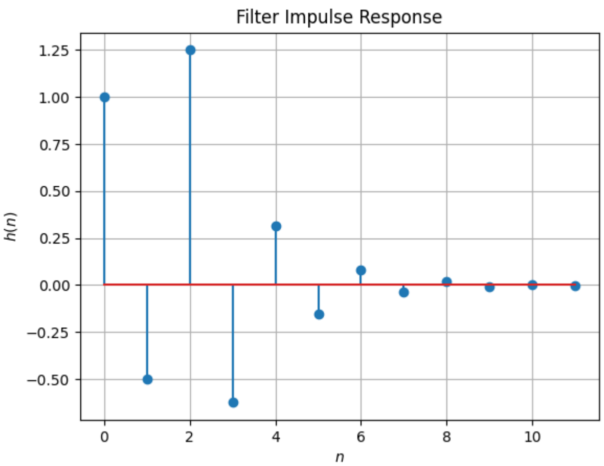

Fig. 5. $h(n)$ as the inverse of $H(z)$

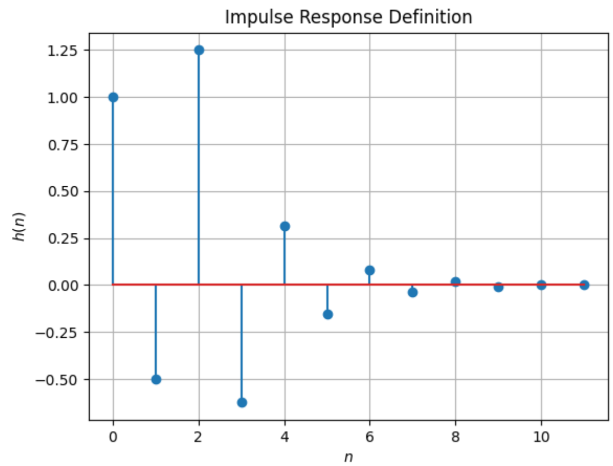

Fig. 6. $h(n)$ from the definition is same as Fig. 5

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: For stable system (31) should converge.
By using ratio test

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

$$\tag{33}$$

For large n

$$u(n) = u(n-2) = 1 \tag{34}$$

$$\lim_{n\to\infty} \left(\frac{h(n+1)}{h(n)}\right) = 1/2 < 1 \tag{35}$$

Therefore it converges. Hence it is stable.

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

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2), \tag{36}$$

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. 6. Note that this is the same as Fig. 5.

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/4.2.
    py
```

IV.5 Compute

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

Comment. The operation in (37) is known as *convolution*.
Solution: The following code plots Fig. 7. Note that this is the same as $y(n)$ in Fig. 3.

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/4.5.
    py
```

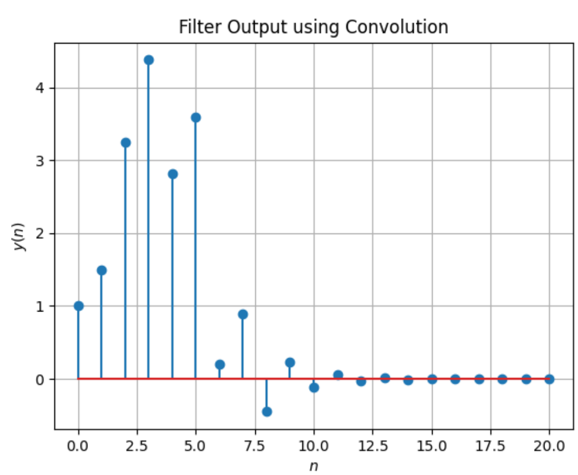

Fig. 7. $y(n)$ from the definition of convolution

IV.6 Show that

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

Solution: In (37), we substitute $k = n - k$ to get

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

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

$$= \sum_{k=-\infty}^{\infty} x(n-k)\, h(k) \tag{41}$$

V. DFT and FFT

V.1 Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n)e^{-\jmath 2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \tag{42}$$

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

V.2 Compute

$$Y(k) = X(k)H(k) \tag{43}$$

V.3 Compute

$$y(n) = \frac{1}{N}\sum_{k=0}^{N-1} Y(k)\cdot e^{\jmath 2\pi kn/N}, \quad n = 0, 1, \dots, N-1 \tag{44}$$

Solution: The above three questions are solved using the code below.

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/5
    _sol.py
```

V.4 Repeat the previous exercise by computing $X(k), H(k)$ and $y(n)$ through FFT and IFFT.
Solution: The solution of this question can be found in the code below.

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/5.4.
    py
```

This code verifies the result by plotting the obtained result with the result obtained by IDFT.

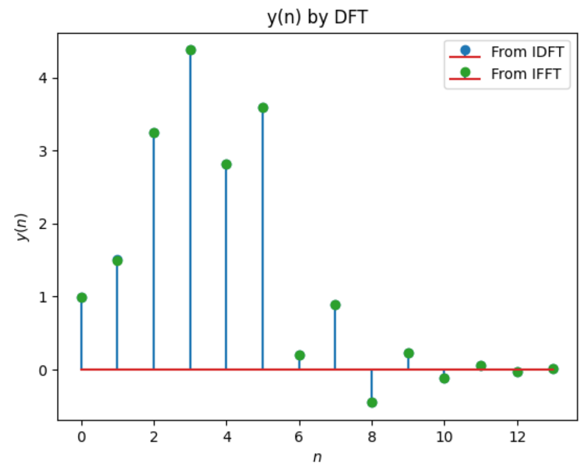

Fig. 8. $y(n)$ obtained from IDFT and IFFT is plotted and verified

V.5 Wherever possible, express all the above equations as matrix equations.
Solution: The DFT matrix is defined as :

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \tag{45}$$

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

$$\mathbf{X} = \mathbf{W}\mathbf{x} \tag{46}$$

where

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

$$\mathbf{X} = \begin{pmatrix} X(0) \\ X(1) \\ \vdots \\ X(n-1) \end{pmatrix} \tag{48}$$

Thus we can rewrite (43) as:

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{H} = (\mathbf{W}\mathbf{x}) \odot (\mathbf{W}\mathbf{h}) \tag{49}$$

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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/5.5.py
```

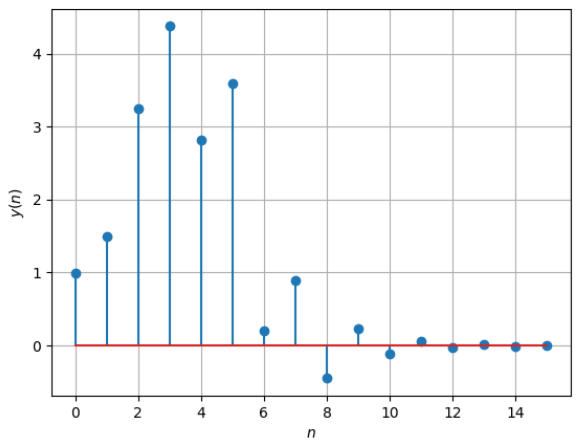

Fig. 9. $y(n)$ obtained from DFT Matrix

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

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \quad (50)$$

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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/6.1.
    py
```

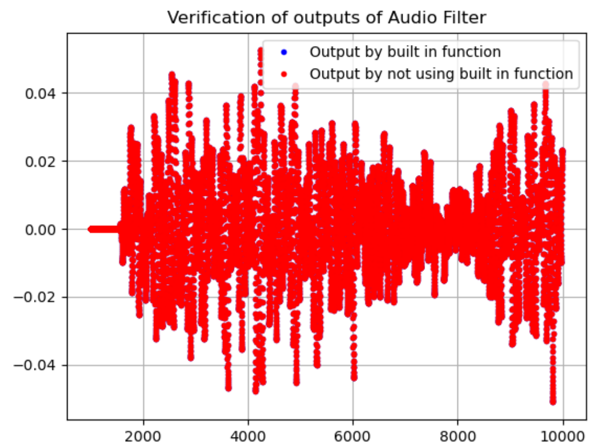

Fig. 10. Both the outputs using and without using function overlap

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,

$$M = 5 \quad (51)$$

$$N = 5 \quad (52)$$

From 50

$$a(0)\, y(n) + a(1)\, y(n-1) + a(2)\, y(n-2) + a(3) \quad (53)$$

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

Difference Equation is given by :

$$\begin{aligned} &y(n) - (3.66)\, y(n-1) + (5.05)\, y(n-2) \\ &- (3.099)\, y(n-3) + (0.715)\, y(n-4) \\ &= \left(1.45 \times 10^{-5}\right) x(n) + \left(5.74 \times 10^{-5}\right) x(n-1) \\ &+ \left(8.62 \times 10^{-5}\right) x(n-2) + \left(5.74 \times 10^{-5}\right) x(n-3) \\ &+ \left(1.43 \times 10^{-5}\right) x(n-4) \end{aligned} \quad (54)$$

From (50)

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-N}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-M}} \quad (55)$$

$$H(z) = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{\sum_{k=0}^{M} a(k) z^{-k}} \quad (56)$$

Partial fraction on (56) can be generalised as:

$$H(z) = \sum_i \frac{r(i)}{1 - p(i) z^{-1}} + \sum_j k(j) z^{-j} \quad (57)$$

Now,

$$a^n u(n) \overset{\mathcal{Z}}{\longleftrightarrow} \frac{1}{1 - az^{-1}} \tag{58}$$

$$\delta(n-k) \overset{\mathcal{Z}}{\longleftrightarrow} z^{-k} \tag{59}$$

Taking inverse z transform of (57) by using (58) and (59)

$$h(n) = \sum_i r(i)[p(i)]^n u(n) + \sum_j k(j)\delta(n-j) \tag{60}$$

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

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_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)

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} \tag{61}$$

$$H(1) = \sum_{n=0}^{\infty} h(n) = \frac{\sum_{k=0}^{N} b(k)}{\sum_{k=0}^{M} a(k)} < \infty \tag{62}$$

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

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/6
    _filter_response.py
```

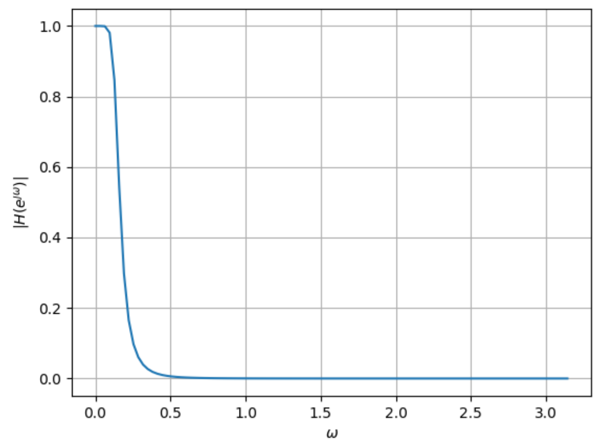

Fig. 11. Frequency Response of Audio Filter

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

$$z = \frac{1 + sT/2}{1 - sT/2} \tag{63}$$

```
https://github.com/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/
    analog_filt.py
```

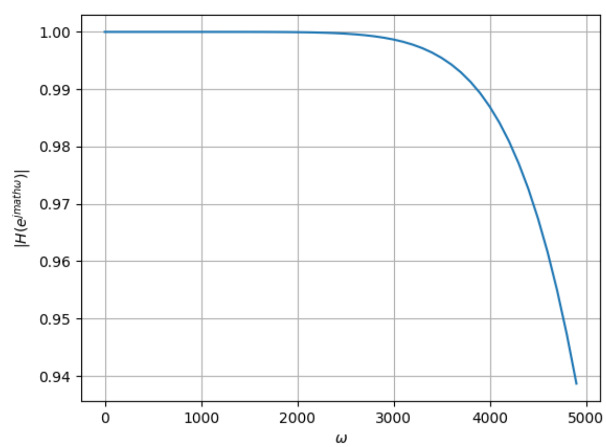

Fig. 12. Butterworth Filter Frequency response in analog domain

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

```
https://github.com/dhanushnayakh03/EE1205/
    tree/main/Audio_Filter/codes/6.2_pole−
    zero.py
```

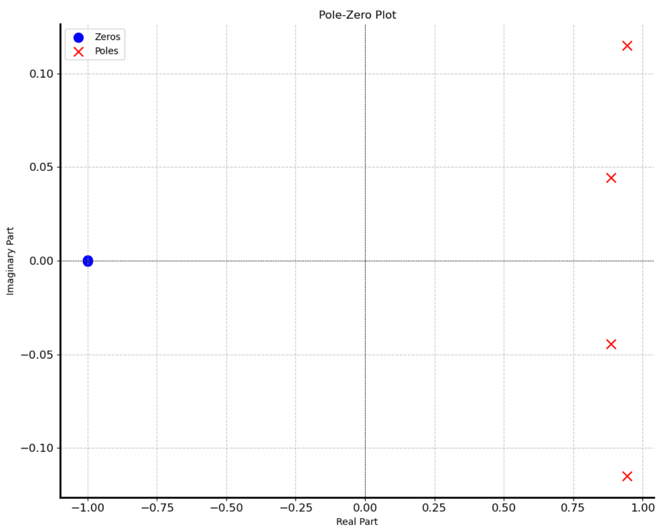

Fig. 13. There are complex poles. So $h(n)$ should be damped sinusoid.

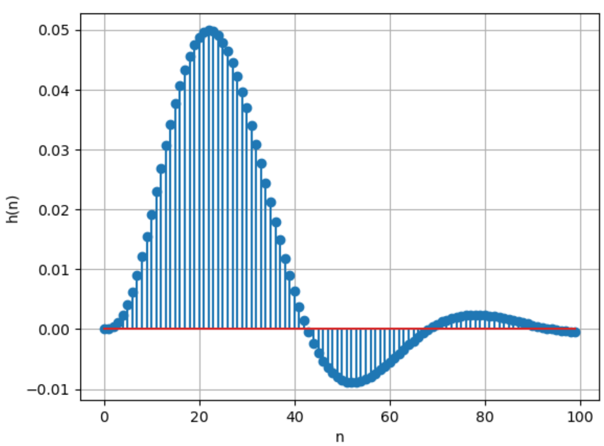

Fig. 14. h(n) of Audio Filter.It is a damped sinusoid.

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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_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/Manojkumar−amabtipudi/
    EE1205/blob/mainAudio_Filter/codes/6.4
    _plot.py
```

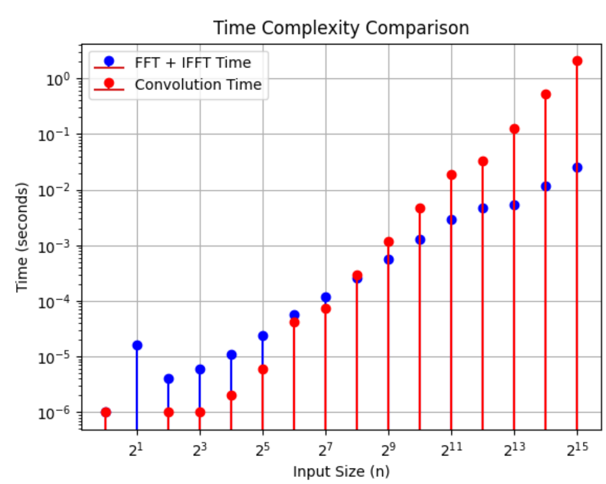

Fig. 15. The Complexity of FFT+IFFT method is $O(nlogn)$ where as by convolution is $O(n^2)$

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.