Audio Filtering

EE23BTECH11013 - Chedurtipati Avyaaz

Contents

1 Digital Filter

1.1 Download the sound file from

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/
    Input_audio.wav
```

1.2 Below is the Python Code to perform the Audio Filtering:

```
import matplotlib.pyplot as plt
import numpy as np
import soundfile as sf
from scipy import signal

# Read .wav file
input_signal, fs = sf.read('Input_audio.wav')

# Order of the filter
order = 3

# Cutoff frequency 4kHz
cutoff_freq = 4000.0

# Digital frequency
Wn = 2 * cutoff_freq / fs
b, a = signal.butter(order, Wn, 'low')

# Filter the input signal with a Butterworth
    filter
output_signal = signal.filtfilt(b, a,
    input_signal, method="gust")

sf.write('ReducedNoise_s181.wav',
    output_signal, fs)
```

1.3 The audio file is analyzed using spectrogram using the online platform https://academo.org/demos/spectrum-analyzer.

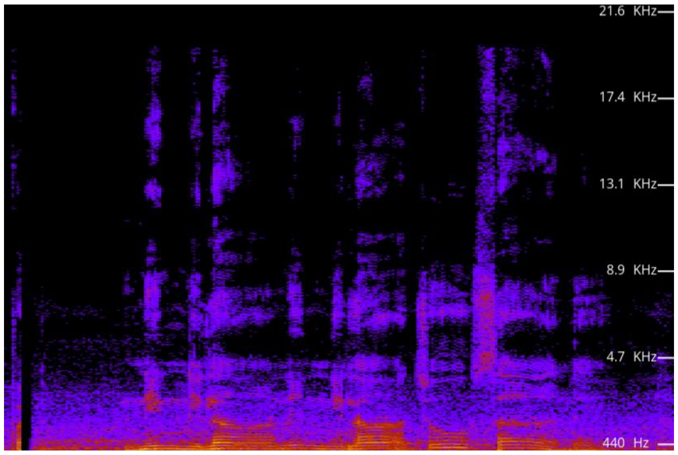

Fig. 1: Spectrogram of Input Audio

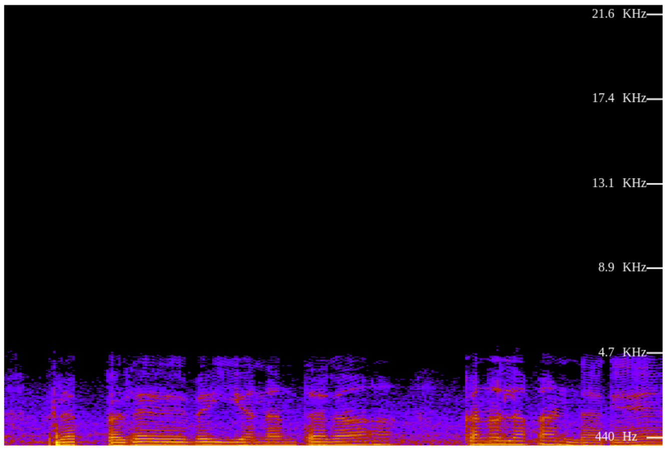

Fig. 2: Spectrogram of Filtered Input Audio

1.4 The output of the python script in Problem 1.2 is the audio file ReducedNoise_s181.wav. Play the file in the spectrogram in Problem 1.3.

What do you observe?
Solution: The orange and yellow areas represent frequencies that have high intensities in the sound. The key strokes as well as background noise is subdued in the audio. Also, the signal is blank for frequencies above 5.1 kHz.

2 Difference Equation

2.1 Let

$$x(n) = \left\{\underset{\uparrow}{1}, 2, 3, 4, 2, 1\right\} \tag{2.0.1}$$

Sketch $x(n)$.

2.2 Let

$$y(n) + \frac{1}{2}y(n-1) = x(n) + x(n-2), \quad y(n) = 0, n < 0 \tag{2.0.2}$$

Sketch $y(n)$. Solve
Solution: The C code calculates $y(n)$ and Python plots the graph.

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/2.2
    _xnyn.c
```

Below are the plots of the $x(n)$ and $y(n)$:

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/2.2
    _xnyn.py
```

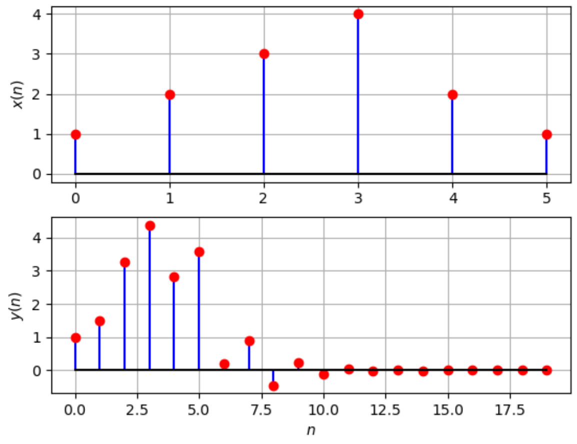

Fig. 3: Plot of $x(n)$ and $y(n)$

3 Z-Transform

3.1 The Z-transform of $x(n)$ is defined as

$$X(z) = \mathcal{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{3.0.1}$$

Show that

$$\mathcal{Z}\{x(n-1)\} = z^{-1}X(z) \tag{3.0.2}$$

and find

$$\mathcal{Z}\{x(n-k)\} \tag{3.0.3}$$

Solution: From (3.0.1),

$$\mathcal{Z}\{x(n-1)\} = \sum_{n=-\infty}^{\infty} x(n-1)z^{-n} \tag{3.0.4}$$

$$= \sum_{n=-\infty}^{\infty} x(n)z^{-n-1} = z^{-1}\sum_{n=-\infty}^{\infty} x(n)z^{-n} \tag{3.0.5}$$

resulting in (3.0.2). Similarly, it can be shown that

$$\mathcal{Z}\{x(n-k)\} = z^{-k}X(z) \tag{3.0.6}$$

3.2 Find

$$H(z) = \frac{Y(z)}{X(z)} \tag{3.0.7}$$

from (2.0.2) assuming that the Z-transform is a linear operation.
Solution: Applying (3.0.6) in (2.0.2),

$$Y(z) + \frac{1}{2}z^{-1}Y(z) = X(z) + z^{-2}X(z) \tag{3.0.8}$$

$$\implies \frac{Y(z)}{X(z)} = \frac{1 + z^{-2}}{1 + \frac{1}{2}z^{-1}} \tag{3.0.9}$$

3.3 Find the Z transform of

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.0.10}$$

and show that the Z-transform of

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.0.11}$$

is

$$U(z) = \frac{1}{1 - z^{-1}}, \quad |z| > 1 \tag{3.0.12}$$

Solution:

$$\delta(n) \overset{\mathcal{Z}}{\longleftrightarrow} 1 \tag{3.0.13}$$

and from (3.0.11),

$$U(z) = \sum_{n=0}^{\infty} z^{-n} \tag{3.0.14}$$

$$= \frac{1}{1 - z^{-1}}, \quad |z| > 1 \tag{3.0.15}$$

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

3.4 Show that

$$a^n u(n) \overset{\mathcal{Z}}{\longleftrightarrow} \frac{1}{1 - az^{-1}} \quad |z| > |a| \tag{3.0.16}$$

Solution:

$$a^n u(n) \overset{\mathcal{Z}}{\longleftrightarrow} \sum_{n=0}^{\infty} \left(az^{-1}\right)^n \tag{3.0.17}$$

$$= \frac{1}{1 - az^{-1}} \quad |z| > |a| \tag{3.0.18}$$

3.5 Let

$$H\left(e^{j\omega}\right) = H\left(z = e^{j\omega}\right). \tag{3.0.19}$$

Plot $\left|H\left(e^{j\omega}\right)\right|$. Comment. $H(e^{j\omega})$ is known as the *Discrete Time Fourier Transform* (DTFT) of $h(n)$.

Solution: Below is the code which plots the magnitude of Transfer Function:

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/3.5
    _H.py
```

The DTFT of a sequence $x(n)$ is given by:

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n}$$

Now, consider $H(e^{j(\omega+2\pi)})$:

$$H(e^{j(\omega+2\pi)}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j(\omega+2\pi)n}$$

Using Euler's formula,

$$e^{-j(\omega+2\pi)n} = e^{-j\omega n}e^{-j2\pi n} = e^{-j\omega n} \tag{3.0.20}$$

$$\because e^{-j2\pi n} = 1 \quad \forall n \tag{3.0.21}$$

$$H(e^{j(\omega+2\pi)}) = H(e^{j\omega}) \tag{3.0.22}$$

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

Substituting $z = e^{j\omega}$ in (3.0.9), we get

$$\left|H\left(e^{j\omega}\right)\right| = \left|\frac{1 + e^{-2j\omega}}{1 + \frac{1}{2}e^{-j\omega}}\right| \tag{3.0.23}$$

$$= \sqrt{\frac{(1 + \cos 2\omega)^2 + (\sin 2\omega)^2}{\left(1 + \frac{1}{2}\cos\omega\right)^2 + \left(\frac{1}{2}\sin\omega\right)^2}} \tag{3.0.24}$$

$$= \frac{4|\cos\omega|}{\sqrt{5 + 4\cos\omega}} \tag{3.0.25}$$

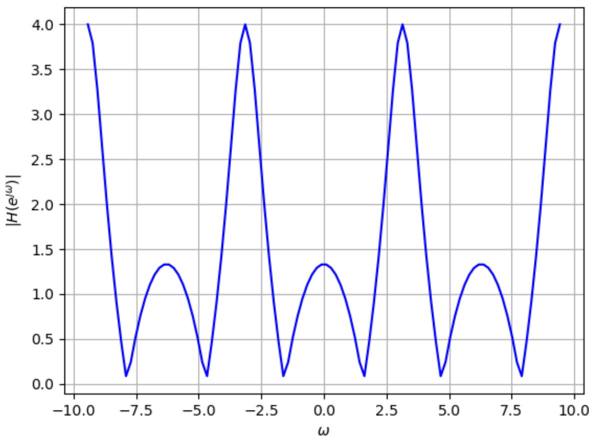

Fig. 4: $\left|H\left(e^{j\omega}\right)\right|$ vs ω

4 Impulse Response

4.1 Find an expression for $h(n)$ using $H(z)$, given that

$$h(n) \overset{\mathcal{Z}}{\longleftrightarrow} H(z) \tag{4.0.1}$$

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 (2.0.2).

Solution: From (3.0.9),

$$H(z) = \frac{1}{1 + \frac{1}{2}z^{-1}} + \frac{z^{-2}}{1 + \frac{1}{2}z^{-1}} \tag{4.0.2}$$

$$\implies h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \tag{4.0.3}$$

using (3.0.16) and (3.0.6).

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

Solution: The following code plots $h(n)$

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/4.2
    _h.py
```

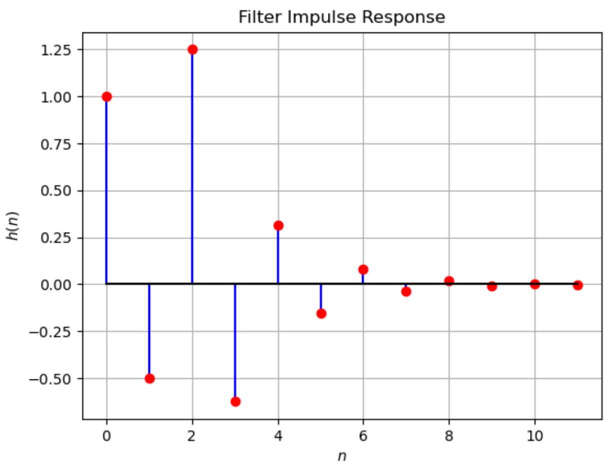

Fig. 5: $h(n)$ vs n

4.3 The system with $h(n)$ is defined to be stable if

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \tag{4.0.4}$$

Is the system defined by (2.0.2) stable for the impulse response in (4.0.1)?
Solution: For stable system (4.0.4) should converge.
By using ratio test for convergence:

$$\lim_{n\to\infty}\left|\frac{h(n+1)}{h(n)}\right| < 1 \tag{4.0.5}$$

$$\tag{4.0.6}$$

For large n

$$u(n) = u(n-2) = 1 \tag{4.0.7}$$

$$\lim_{n\to\infty}\left(\frac{h(n+1)}{h(n)}\right) = \frac{1}{2} < 1 \tag{4.0.8}$$

Hence it is stable.

4.4 Compute and sketch $h(n)$ using

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2), \tag{4.0.9}$$

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

Below code plots Fig. 6 which is same as the Fig. 5.

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/4.4
    _hndef.py
```

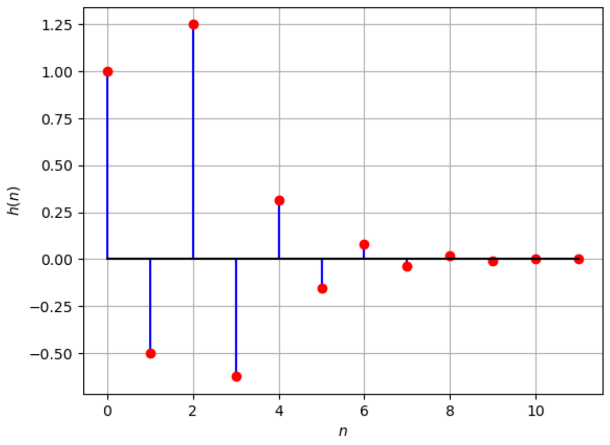

Fig. 6: $h(n)$ vs n using definition

4.5 Compute

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \tag{4.0.10}$$

Comment. The operation in (4.0.10) is known as *convolution*.
Solution: Below code plots Fig. 7 which is same as $y(n)$ in Fig. 3.

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/4.5
    _ynconv.py
```

4.6 Show that

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \tag{4.0.11}$$

Solution: In (6.0.11), we substitute $k = n - k$

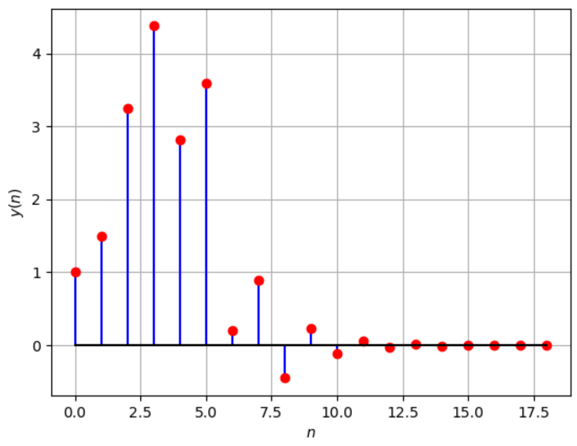

Fig. 7: $y(n)$ from the definition of convolution

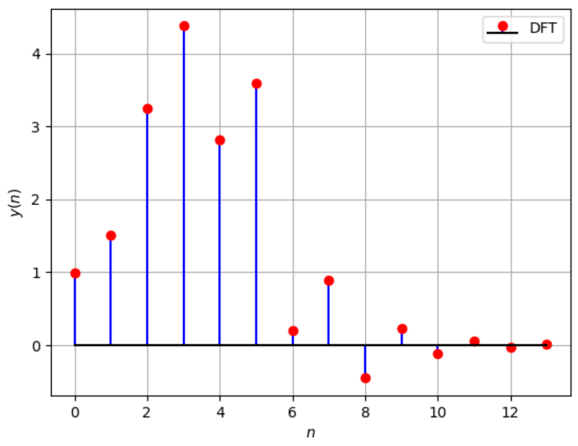

Fig. 8: $y(n)$ obtained from DFT

to get

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\, h(n-k) \tag{4.0.12}$$

$$= \sum_{n-k=-\infty}^{\infty} x(n-k)\, h(k) \tag{4.0.13}$$

$$= \sum_{k=-\infty}^{\infty} x(n-k)\, h(k) \tag{4.0.14}$$

5 DFT and FFT

5.1 Compute

$$X(k) \overset{\triangle}{=} \sum_{n=0}^{N-1} x(n) e^{-\jmath 2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \tag{5.0.1}$$

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

5.2 Compute

$$Y(k) = X(k)H(k) \tag{5.0.2}$$

5.3 Compute

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k) \cdot e^{\jmath 2\pi kn/N}, \quad n = 0, 1, \dots, N-1 \tag{5.0.3}$$

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

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/5.1
    _2_3.py
```

5.4 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/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/5.4
    _FFT.py
```

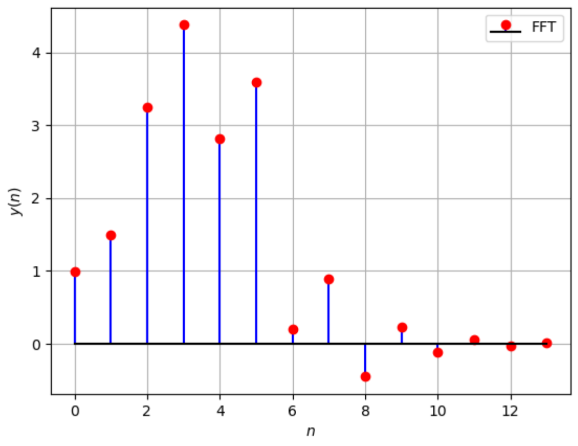

Fig. 9: $y(n)$ obtained using IFFT

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

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \tag{5.0.4}$$

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

$$\mathbf{X} = \mathbf{W}\mathbf{x} \tag{5.0.5}$$

where

$$\mathbf{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix} \tag{5.0.6}$$

$$\mathbf{X} = \begin{pmatrix} X(0) \\ X(1) \\ \vdots \\ X(n-1) \end{pmatrix} \tag{5.0.7}$$

Thus we can rewrite (5.0.2) as:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{H} = (\mathbf{W}\mathbf{x}) \cdot (\mathbf{W}\mathbf{h}) \tag{5.0.8}$$

The below code computes $y(n)$ by DFT Matrix and then plots it.

```
https://github.com/Avyaaz13/Audio−Filtering/blob/
    main/Audio%20Filtering/codes/5.5_matrix.py
```

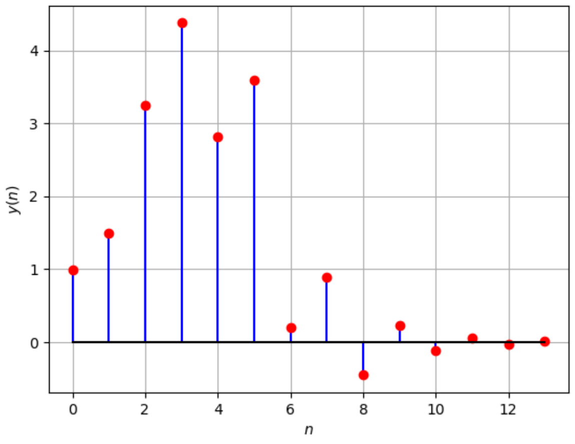

Fig. 10: $y(n)$ from DFT Matrix

6 EXERCISES

Answer the following questions by looking at the python code in Problem 1.2.

6.1 The command

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

in Problem 1.2 is executed through the following difference equation

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \tag{6.0.1}$$

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.filtfilt.

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/6.1
    _filtfilt.py
```

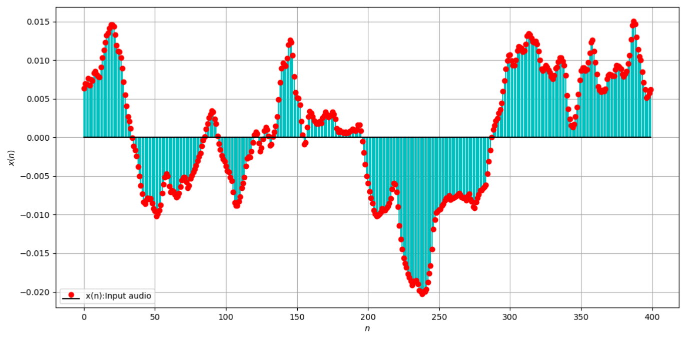

Fig. 11: $x(n)$ vs n

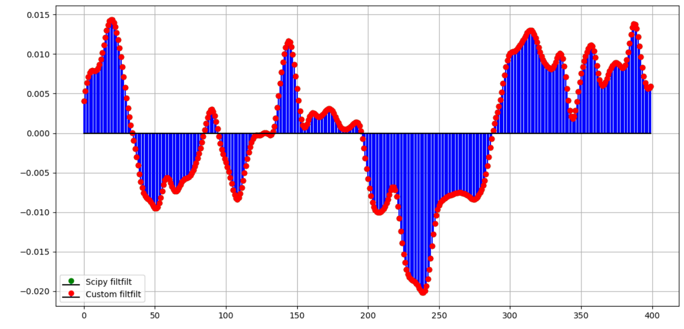

Fig. 12: Verifying the output using and without using *signal.filtfilt*

Solution: The below code gives the output of an Audio Filter without using the built in function signal.lfilter.

```
https://github.com/Avyaaz13/Audio−Filtering/
      blob/main/Audio%20Filtering/codes/lfilt.
      py
```

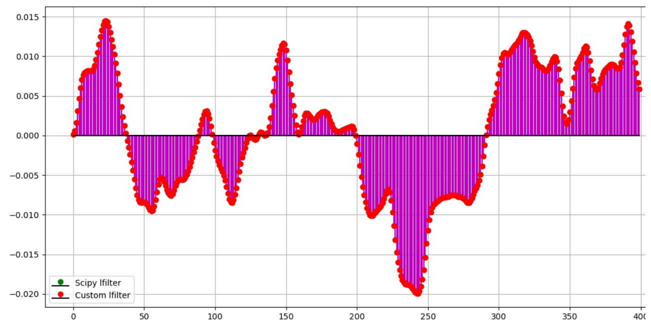

Fig. 13: Verifying the output using and without using *signal.lfilter*

6.2 Repeat all the exercises in the previous sections for the above a and b.

Solution: The code in 1.2 generates the values of a and b which can be used to generate a difference equation.

$$a = \begin{bmatrix} 1 & -1.87302725 & 1.30032695 & -0.31450204 \end{bmatrix}$$

$$b = \begin{bmatrix} 0.0140997 & 0.0422991 & 0.0422991 & 0.0140997 \end{bmatrix}$$

And,

$$M = 3 \tag{6.0.2}$$

$$N = 3 \tag{6.0.3}$$

From 6.0.1

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

Difference Equation is given by :

$$y(n) - 1.87y(n-1) + 1.3y(n-2) - 0.31y(n-3) = 0.014x(n) + 0.042x(n-1) + 0.042x(n-2) + 0.014x(n-3)$$

From (6.0.1)

$$H(z) = \frac{b(0) + b(1)z^{-1} + b(2)z^{-2} + \ldots + b(N)z^{-N}}{a(0) + a(1)z^{-1} + a(2)z^{-2} + \ldots + a(M)z^{-M}} \tag{6.0.4}$$

$$H(z) = \frac{\sum_{k=0}^{N} b(k)z^{-k}}{\sum_{k=0}^{M} a(k)z^{-k}} \tag{6.0.5}$$

Below code plotsFig. 14 Frequency response:

```
https://github.com/Avyaaz13/Audio−Filtering/
      blob/main/Audio%20Filtering/codes/6.2
      _Hz_custom.py
```

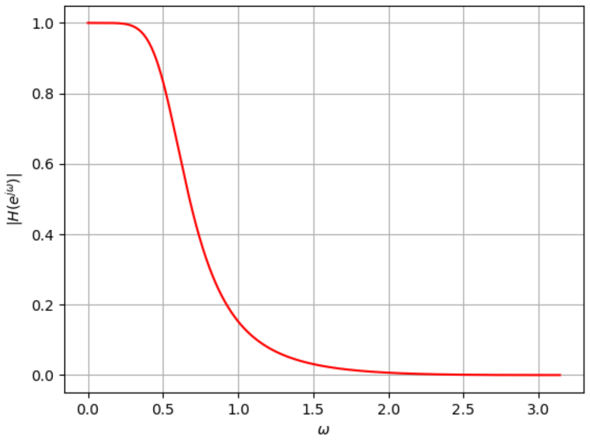

Fig. 14: Frequency Response of Audio Filter

Below code plots Fig. 15 Zero-Pole graph:

```
https://github.com/Avyaaz13/Audio−Filtering/
      blob/main/Audio%20Filtering/codes/zero−
      pole.py
```

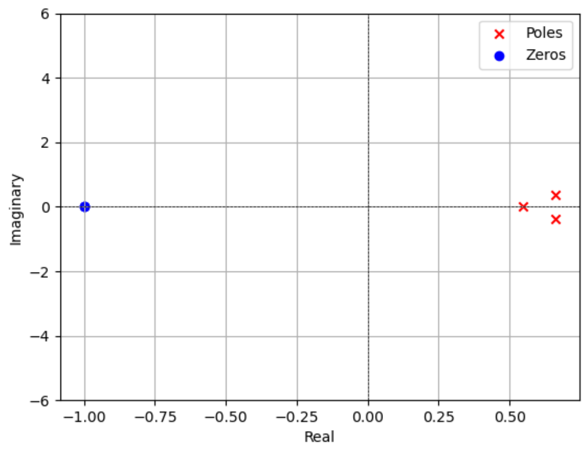

Fig. 15: Zero-Pole plot

$$\because \text{Zeroes} \approx \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}$$

$$\text{Poles} = \begin{bmatrix} 0.663 + 0.368j & 0.663 - 0.368j & 0.547 \end{bmatrix}$$

Now,

$$\delta(n-k) \overset{\mathcal{Z}}{\longleftrightarrow} z^{-k} \tag{6.0.6}$$

Let us assume that a causal sequence is to be obtained using **Long Division Method:**

$$H(z) = 0.014 + 0.069z^{-1} + 0.153z^{-2} + 0.215z^{-3} + 0.226z^{-4} + 0.192z^{-5} + ... \quad (6.0.7)$$

Taking inverse z transform of (6.0.5) by using (6.0.6):

$$h(n) = 0.014\delta(n) + 0.069\delta(n-1) + 0.153\delta(n-2) + 0.215\delta(n-3) + 0.226\delta(n-4) + 0.192\delta(n-5) + ... \quad (6.0.$$

Below is the code which plots the *Impulse response:*

```
https://github.com/Avyaaz13/Audio−Filtering
    /blob/main/Audio%20Filtering/codes/6.2
    _hn.py
```

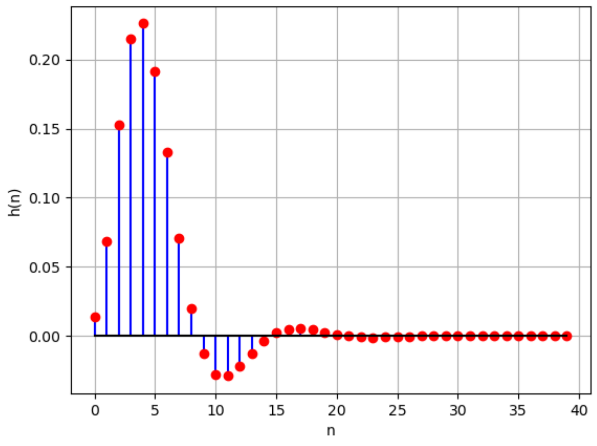
Fig. 16: $h(n)$ vs n

Stability of h(n):
According to (4.0.4)

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} \quad (6.0.9)$$

$$H(1) = \sum_{n=0}^{\infty} h(n) = \frac{\sum_{k=0}^{N} b(k)}{\sum_{k=0}^{M} a(k)} < \infty \quad (6.0.10)$$

As both $a(k)$ and $b(k)$ are finite length sequences they converge.

6.3 Compute

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (6.0.11)$$

Comment. The operation in (6.0.11) is known as *convolution*.
Solution: Below code plots Fig. 17

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/6.3
    _ynconv.py
```

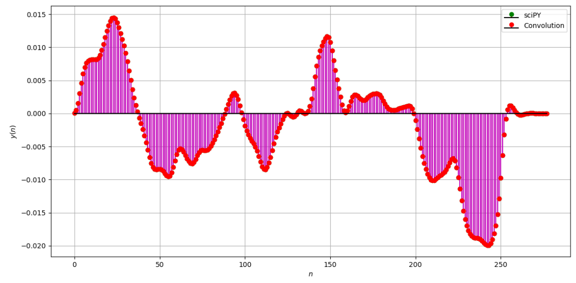

Fig. 17: $y(n)$ from the definition of convolution

6.4 Compute $y(n)$ using DFT and FFT.
Below code plots Fig. 18 $y(n)$ using DFT:

```
https://github.com/Avyaaz13/Audio−Filtering/
    blob/main/Audio%20Filtering/codes/6.4
    _yndft.py
```

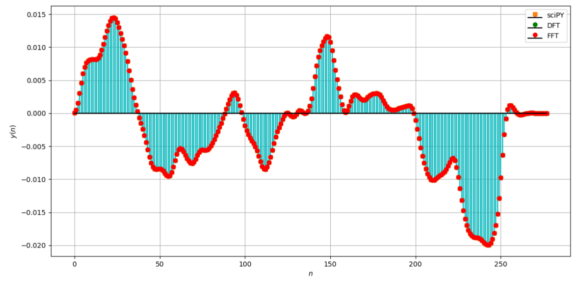

Fig. 18: $y(n)$ obtained from DFT & FFT

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

6.6 What is type, order and cutoff-frequency of the above butterworth filter
Solution: The given butterworth filter is low-pass with order = 3 and cutoff-frequency = $4kHz$.

6.7 Modify the code with different input parameters and get the best possible output.
Solution: A better filtering was found when order of the filter is 4.