

Audio Filtering

EE23BTECH11009 - AROSHISH PRADHAN*

CONTENTS

1	Software Installation	1
2	Digital Filter	1
3	Difference Equation	2
4	Z-transform	2
5	Impulse Response	4
6	DFT and FFT	6
7	Exercises	7

Abstract—This manual attempts digital signal processing of an audio file.

1 SOFTWARE INSTALLATION

Run the following commands

```
sudo apt-get update
sudo apt-get install libffi-dev libsndfile1 python3
-sciPy python3-numpy python3-matplotlib
sudo pip install cffi pysoundfile
```

2 DIGITAL FILTER

2.1 Download the sound file from

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/2.wav
```

2.2 You will find a spectrogram at <https://academo.org/demos/spectrum-analyzer>. Upload the sound file that you downloaded in Problem 2.1 in the spectrogram and play. Observe the spectrogram. What do you find?

Solution: The purple areas of the spectrogram represent frequencies with low intensities (noise) while the red-yellow regions represent frequencies with high intensities (voice).

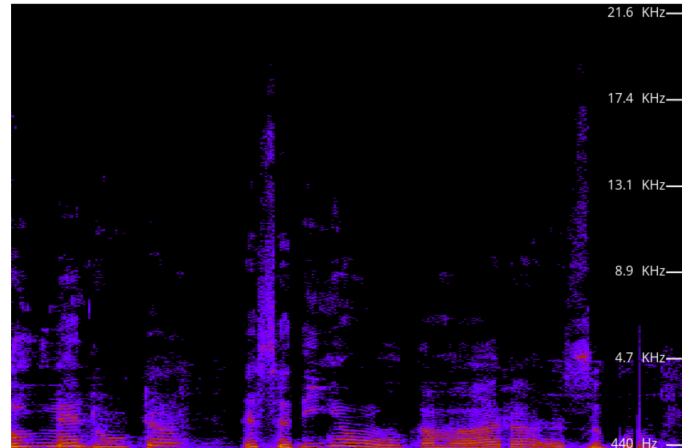


Fig. 2.2: Spectrogram before filtering

2.3 Write the python code for removal of out of band noise and execute the code.

Solution: Noise in the audio is filtered out using the following python code:

```
import soundfile as sf
from scipy import signal

# Read .wav file
input_signal, fs = sf.read('2.wav')

# Order of the filter
order = 4

# Cutoff frequency 6kHz
cutoff_freq = 6000.0

# Digital frequency
Wn = 2 * cutoff_freq / fs

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

print(a)
print(b)
```

```

output_signal = signal.lfilter(b,a,
    input_signal)
# Write the output signal into a .wav file
sf.write('2_fil.wav', output_signal, fs)

```

2.4 The output of the python script in Problem 2.3 is the audio file 2_fil.wav. Play the file in the spectrogram in Problem 2.2. What do you observe?

Solution:

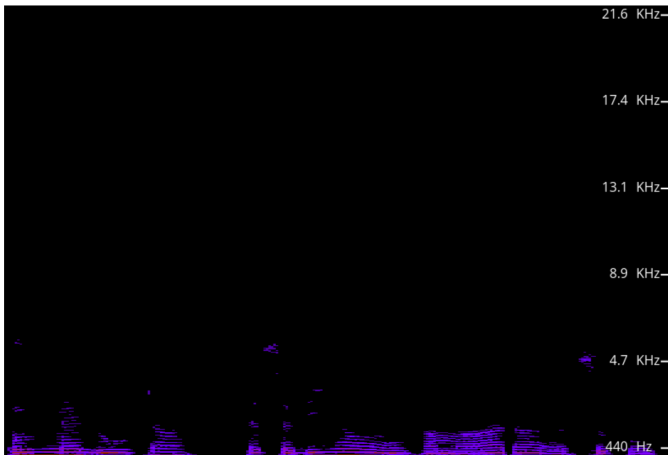


Fig. 2.4: Spectrogram after filtering

The background noise (low intensities) is subdued in the audio. Also, the signal is blank for frequencies above 6 kHz.

3 DIFFERENCE EQUATION

3.1 Let

$$x(n) = \left\{ \underset{\uparrow}{1}, 2, 3, 4, 2, 1 \right\} \quad (3.1)$$

Sketch $x(n)$.

Solution: The following code yields Fig. 3.1.

```

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/3.1.py

```

3.2 Let

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

$$y(n) = 0, n < 0 \quad (3.2)$$

Sketch $y(n)$.

Solution: The following codes yield Fig. 3.2.

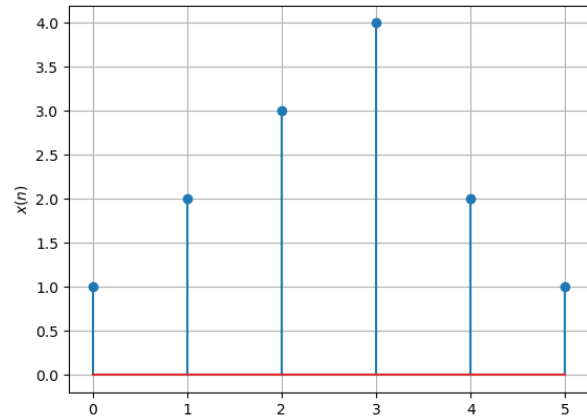


Fig. 3.1: Digital Filter Input $x(n)$

```

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/3.2.c

```

```

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/3.2.py

```

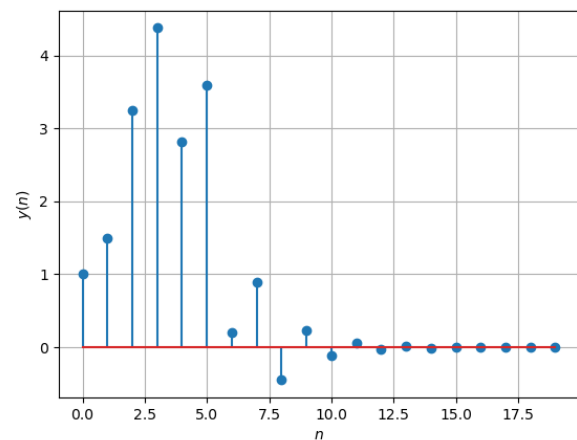


Fig. 3.2: Digital Filter Output $y(n)$

4 Z-TRANSFORM

4.1 The Z-transform of $x(n]$ is defined as

$$X(z) = \mathcal{Z}\{x(n)\} = \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (4.1)$$

Show that

$$\mathcal{Z}\{x(n-1)\} = z^{-1}X(z) \quad (4.2)$$

and find

$$\mathcal{Z}\{x(n-k)\} \quad (4.3)$$

Solution: From (4.1),

$$\mathcal{Z}\{x(n-1)\} = \sum_{n=-\infty}^{\infty} x(n-1)z^{-n} \quad (4.4)$$

$$n \longrightarrow n+1 \quad (4.5)$$

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

$$= z^{-1} \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (4.6)$$

$$= z^{-1}X(z) \quad (4.7)$$

resulting in (4.2). Similarly, it can be shown that

$$\mathcal{Z}\{x(n-k)\} = \sum_{n=-\infty}^{\infty} x(n-k)z^{-n} \quad (4.8)$$

$$n \longrightarrow n+k$$

$$= \sum_{n=-\infty}^{\infty} x(n)z^{-n-k} \quad (4.9)$$

$$= z^{-k} \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (4.10)$$

$$= z^{-k}X(z) \quad (4.11)$$

4.2 Find

$$H(z) = \frac{Y(z)}{X(z)} \quad (4.12)$$

from (3.2) assuming that the Z-transform is a linear operation.

Solution: Using (4.11) in (3.2),

$$Y(z) + \frac{1}{2}z^{-1}Y(z) = X(z) + z^{-2}X(z) \quad (4.13)$$

$$\Rightarrow \frac{Y(z)}{X(z)} = \frac{1+z^{-2}}{1+\frac{1}{2}z^{-1}} \quad (4.14)$$

4.3 Find the Z transform of

$$\delta(n) = \begin{cases} 1 & n=0 \\ 0 & \text{otherwise} \end{cases} \quad (4.15)$$

and show that the Z-transform of

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.16)$$

is

$$U(z) = \frac{1}{1-z^{-1}}, \quad |z| > 1 \quad (4.17)$$

Solution:

$$\mathcal{Z}\{\delta(n)\} = \sum_{n=-\infty}^{\infty} \delta(n)z^{-n} \quad (4.18)$$

$$= \delta(0)z^{-0} \quad (4.19)$$

$$= 1 \quad (4.20)$$

and from (4.16),

$$U(z) = \sum_{n=0}^{\infty} z^{-n} \quad (4.21)$$

$$= \frac{1}{1-z^{-1}}, \quad |z| > 1 \quad (4.22)$$

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

4.4 Show that

$$a^n u(n) \longleftrightarrow \frac{1}{1-az^{-1}} \quad |z| > |a| \quad (4.23)$$

Solution:

$$\mathcal{Z}\{a^n u(n)\} = \sum_{n=-\infty}^{\infty} a^n u(n)z^{-n} \quad (4.24)$$

$$= \sum_{n=0}^{\infty} (az^{-1})^n \quad (4.25)$$

$$= \frac{1}{1-az^{-1}} \quad |z| > |a| \quad (4.26)$$

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

4.5 Let

$$H(e^{j\omega}) = H(z = e^{j\omega}). \quad (4.27)$$

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

Solution: The following codes plot Fig. 4.5.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/4.5.c
```

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/4.5.py
```

Substituting $z = e^{j\omega}$ in (4.14),

$$|H(e^{j\omega})| = \left| \frac{1+e^{-2j\omega}}{1+\frac{1}{2}e^{-j\omega}} \right| \quad (4.28)$$

$$= \frac{\sqrt{(1+\cos 2\omega)^2 + (\sin 2\omega)^2}}{\sqrt{\left(1+\frac{\cos \omega}{2}\right)^2 + \left(\frac{\sin \omega}{2}\right)^2}} \quad (4.29)$$

$$= \frac{4|\cos \omega|}{\sqrt{5+4\cos \omega}} \quad (4.30)$$

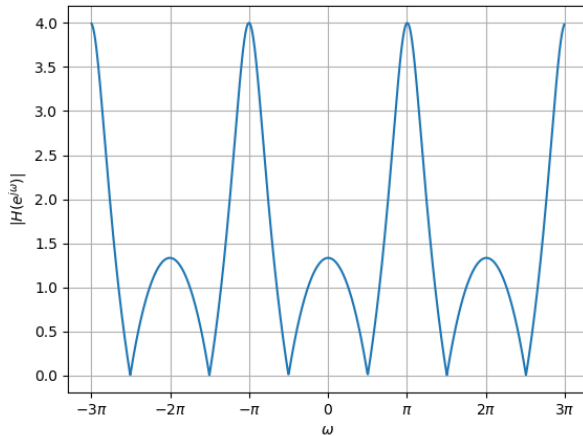


Fig. 4.5: Plot of $|H(e^{j\omega})|$

which has a fundamental period of 2π :

$$|H(e^{j(\omega+2\pi)})| = \frac{4|\cos(\omega+2\pi)|}{\sqrt{5+4\cos(\omega+2\pi)}} \quad (4.31)$$

$$= \frac{4|\cos \omega|}{\sqrt{5+4\cos \omega}} \quad (4.32)$$

$$= |H(e^{j\omega})| \quad (4.33)$$

This can be verified from the graph too.

The plot verifies the property of DTFT of a signal that it is continuous and periodic.

5 IMPULSE RESPONSE

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

$$h(n) \longleftrightarrow H(z) \quad (5.1)$$

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 (3.2).

Solution: From (4.14),

$$H(z) = \frac{1}{1 + \frac{1}{2}z^{-1}} + \frac{z^{-2}}{1 + \frac{1}{2}z^{-1}} \quad (5.2)$$

$$\Rightarrow h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \quad (5.3)$$

using (4.23) and (4.11).

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

Solution: The following code plots Fig. 5.2.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.2.c
```

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.2.py
```

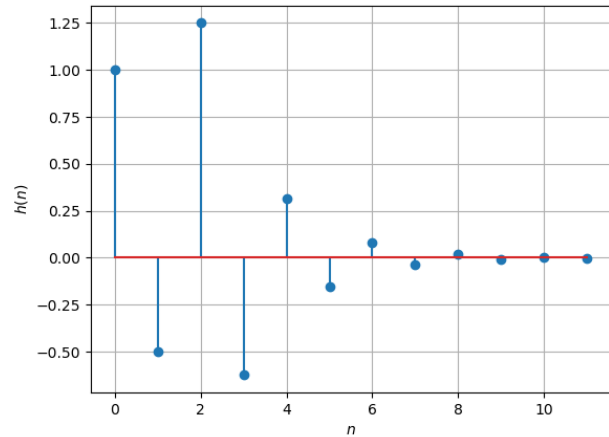


Fig. 5.2: Plot of $h(n)$

From graph, it is visible that $h(n)$ is bounded. To check for convergence we can use the ratio test:

$$\lim_{n \rightarrow \infty} \left| \frac{h(n+1)}{h(n)} \right| = \left| \frac{\left(-\frac{1}{2}\right)^{n+1} + \left(-\frac{1}{2}\right)^{n-1}}{\left(-\frac{1}{2}\right)^n + \left(-\frac{1}{2}\right)^{n-2}} \right| \quad (5.4)$$

$$= \frac{1}{2} < 1 \quad (5.5)$$

Hence, $h(n)$ is convergent.

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

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \quad (5.6)$$

Is the system defined by (3.2) stable for the impulse response in (5.1)?

Solution: Sum of infinite terms of a convergent series is finite. From (5.5), we proved that $h(n)$ was convergent therefore

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \quad (5.7)$$

Hence, the system with the impulse response $h(n)$ is a stable system.

5.4 Compute and sketch $h(n)$ using

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2), \quad (5.8)$$

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

Solution: The following code plots Fig. 5.4. Note that this is the same as Fig. 5.2.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.4.c

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.4.py
```

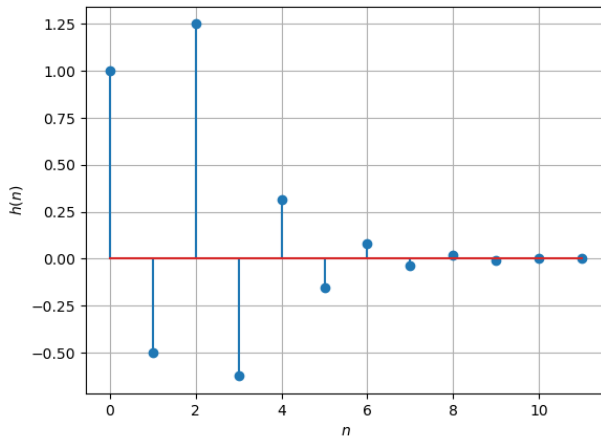


Fig. 5.4: Plot of $h(n)$ using definition

5.5 Compute

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (5.9)$$

Comment. The operation in (5.9) is known as *convolution*.

Solution: The following codes plot Fig. 5.5. Note that this is the same as $y(n)$ in Fig. 3.2.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.5.c

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.5.py
```

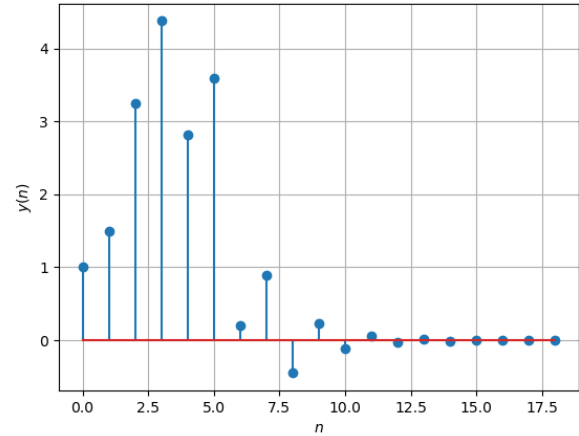


Fig. 5.5: $y(n)$ using convolution

5.6 Show that

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \quad (5.10)$$

Solution: From (5.9),

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (5.11)$$

Substitute $k \rightarrow n-k$

$$= \sum_{n-k=-\infty}^{\infty} x(n-k)h(k) \quad (5.12)$$

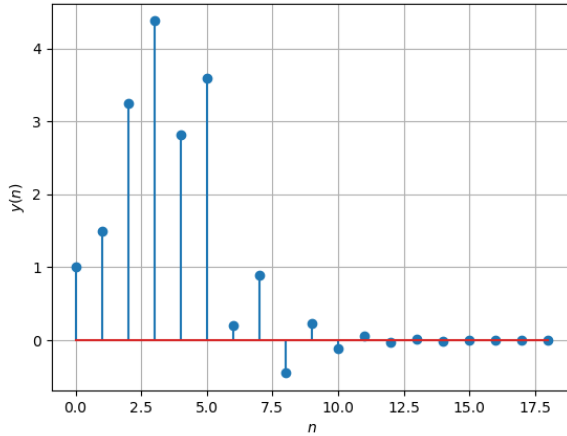
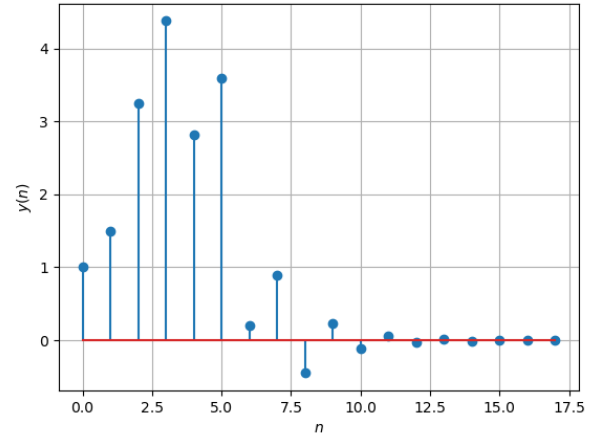
$$= \sum_{k=-\infty}^{\infty} x(n-k)h(k) \quad (5.13)$$

as flipping limits does not change sum.

The following codes plot Fig. 5.6. Note that this is the same as $y(n)$ in Fig. 5.5.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.6.c

wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/5.6.py
```

Fig. 5.6: Plot of $y(n)$ using (5.13)Fig. 6.3: $y(n)$ from IDFT

6 DFT AND FFT

6.1 Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \quad (6.1)$$

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

6.2 Compute

$$Y(k) = X(k)H(k) \quad (6.2)$$

6.3 Compute

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k) \cdot e^{j2\pi kn/N}, \quad n = 0, 1, \dots, N-1 \quad (6.3)$$

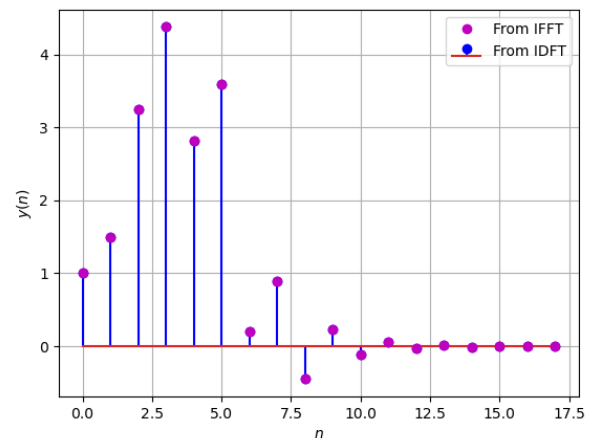
Solution: The following code plots Fig. 6.3 by taking *Inverse Discrete Fourier Transform* (IDFT) of $Y(k)$. Note that this is also the same as $y(n)$ in Fig. 3.2. It also prints out the values of $X(k)$, $H(k)$, $Y(k)$, and $y(n)$.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/6.123.
py
```

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

Solution: The code below calculates $X(k)$, $H(k)$ using FFT and plots the graph of $y(n)$ using IFFT and IDFT both (to compare).

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/6.4.py
```

Fig. 6.4: Plot of $y(n)$ from IDFT and IFFT

6.5 Wherever possible, express all the above equations as matrix equations.

Solution: The DFT matrix is given by:

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \quad (6.4)$$

where $\omega = e^{-\frac{j2\pi}{N}}$. General DFT equation is given by:

$$\mathbf{X} = \mathbf{W}\mathbf{x} \quad (6.5)$$

where

$$\mathbf{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix} \quad (6.6)$$

$$\mathbf{X} = \begin{pmatrix} X(0) \\ X(1) \\ \vdots \\ X(n-1) \end{pmatrix} \quad (6.7)$$

Then from (6.2):

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{H} = (\mathbf{W}\mathbf{x}) \odot (\mathbf{W}\mathbf{h}) \quad (6.8)$$

where \odot represents the Hadamard product which multiplies corresponding elements of matrices of same size.

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

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/6.5.py
```

7 EXERCISES

Answer the following questions by looking at the python code in Problem 2.3.

7.1 The command

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

in Problem 2.3 is executed through the following difference equation

$$\sum_{m=0}^M a(m) y(n-m) = \sum_{k=0}^N b(k) x(n-k) \quad (7.1)$$

where the input signal is $x(n)$ and the output signal is $y(n)$ with initial values all 0. Replace

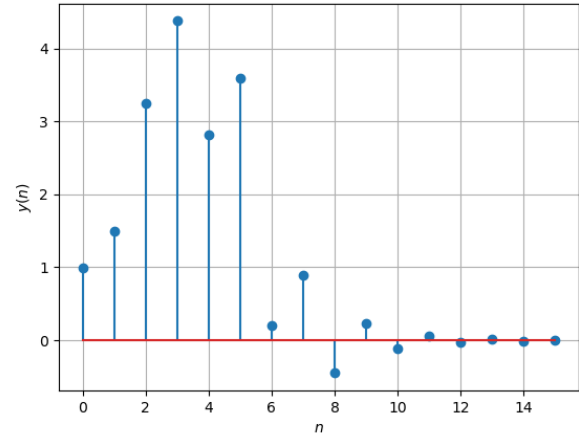


Fig. 6.5: Plot of $y(n)$ using matrix method

signal.filtfilt with your own routine and verify.

Solution: The code below plots the output of `scipy.signal.lfilter` and the output of custom function on the same graph.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/lfilter.py
```

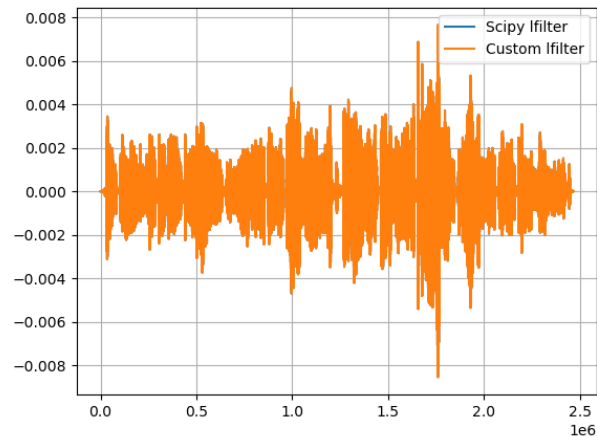


Fig. 7.1: Output of `signal.lfilter` and custom routine

Both the plots overlap, indicating that the custom filter code does the same function as `scipy.signal.lfilter`.

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

Solution: The code in 2.3 calculates values of a and b and prints them:

$a = [1.000 \quad -1.792 \quad 1.518 \quad -0.608 \quad 0.098]$
 $b = [0.013 \quad 0.054 \quad 0.081 \quad 0.054 \quad 0.013]$
 Now, using (7.1), difference equation:

$$\begin{aligned}
 & a(0)y(n) + a(1)y(n-1) + a(2)y(n-2) \\
 & + a(3)y(n-3) + a(4)y(n-4) = b(0)x(n) \\
 & + b(1)x(n-1) + b(2)x(n-2) + b(3)x(n-3) \\
 & + b(4)x(n-4) \quad (7.2)
 \end{aligned}$$

Substituting,

$$\begin{aligned}
 & y(n) - 1.792y(n-1) + 1.518y(n-2) \\
 & - 0.608y(n-3) + 0.098y(n-4) = 0.013x(n) \\
 & + 0.054x(n-1) + 0.081x(n-2) + 0.054x(n-3) \\
 & + 0.013x(n-4) \quad (7.3)
 \end{aligned}$$

The rational transfer function describing this filter in the z-transform domain is:

$$H(z) = \frac{b(0) + b(1)z^{-1} + b(2)z^{-2} + \dots + b(N)z^{-N}}{a(0) + a(1)z^{-1} + a(2)z^{-2} + \dots + a(M)z^{-M}} \quad (7.4)$$

$$= \frac{\sum_{k=0}^N b(k)z^{-k}}{\sum_{k=0}^M a(k)z^{-k}} \quad (7.5)$$

In our case, $M = N = 4$.

Now, the partial fraction of (7.5) is given by:

$$H(z) = \sum_i \frac{r(i)}{1 - p(i)z^{-1}} + \sum_j k(j)z^{-j} \quad (7.6)$$

Values of $r(i)$, $p(i)$ and $k(j)$ are calculated using `scipy.signal.residuez`, which returns the above mentioned series:

$r(i)$	$p(i)$	$k(i)$
$0.28018185 - 1.23886252j$	$0.38674749 + 0.17013423j$	0.13702919
$0.28018185 + 1.23886252j$	$0.38674749 - 0.17013423j$	—
$-0.3419458 + 0.19576406j$	$0.50928445 + 0.54087922j$	—
$-0.3419458 - 0.19576406j$	$0.50928445 - 0.54087922j$	—

TABLE 1: Values of $r(i)$, $p(i)$, $k(i)$

Inverse of (7.6) is given using:

$$a^n u(n) \longleftrightarrow \frac{1}{1 - az^{-1}} \quad (7.7)$$

$$\delta(n - k) \longleftrightarrow z^{-k} \quad (7.8)$$

$$\Rightarrow h(n) = \sum_i r(i)[p(i)]^n u(n) + \sum_j k(j)\delta(n - j) \quad (7.9)$$

Code to plot $h(n)$:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.2hn.
py
```

Plot of $h(n)$:

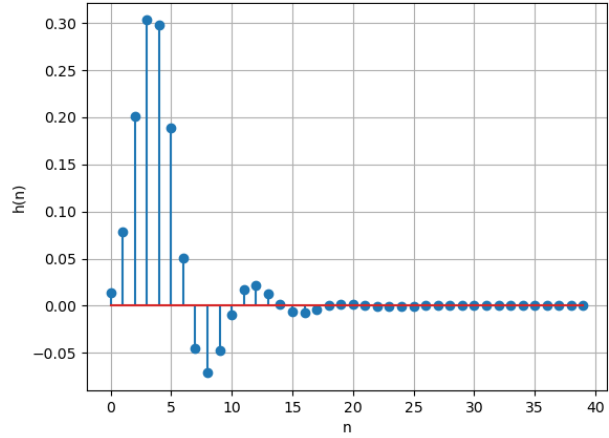


Fig. 7.2: Plot of $h(n)$

Code to plot Pole-Zero Plot:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.2pole.
py
```

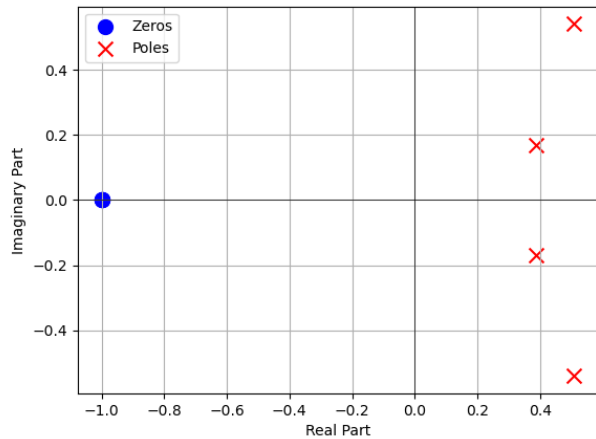


Fig. 7.2: Pole-Zero Plot

There are complex poles, so $h(n)$ has a damped sinusoidal form.

Stability of System

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} \quad (7.10)$$

$$\Rightarrow H(1) = \sum_{n=0}^{\infty} h(n) \quad (7.11)$$

$$= \frac{\sum_{k=0}^N b(k)}{\sum_{k=0}^M a(k)} < \infty \quad (7.12)$$

as both $a(k)$ and $b(k)$ are finite length sequences.

Then, (5.6) implies $h(n)$ is impulse response of a stable system.

Code to plot Frequency Response:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.2hw.
py
```

Frequency Response of Butterworth Filter

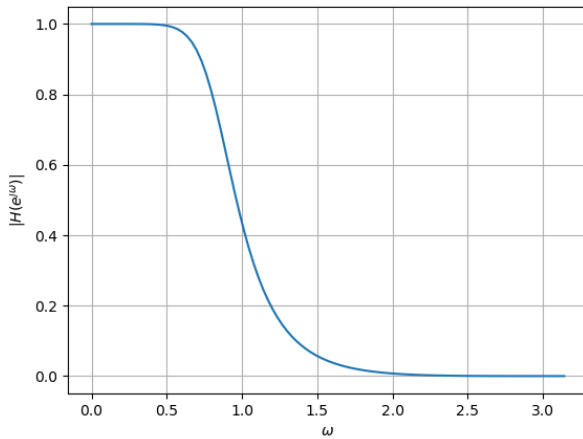


Fig. 7.2: Plot of Frequency Response

Frequency Response of Butterworth Filter in Analog Domain

To convert to analog domain, we can use the Bilinear Transform where we substitute:

$$z = \frac{1 + \frac{sT}{2}}{1 - \frac{sT}{2}} \quad (7.13)$$

Code to plot Frequency Response in Analog Domain:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.2bt.py
```

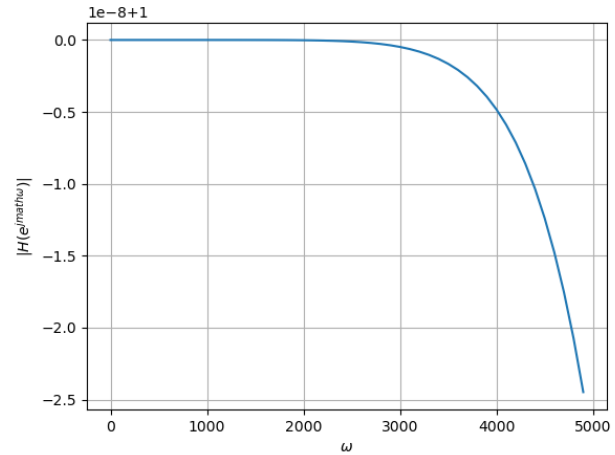


Fig. 7.2: Plot of Frequency Response in Analog Domain

7.3 Implement your own FFT routine in C and call this FFT in python

Solution: The below C code implements the FFT algorithm:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.3.c
```

Run the following command to generate a shared library '7.3.so':

```
gcc -shared -o 7.3.so -fPIC 7.3.c
```

The C code is called in the following Python code and output is printed. It can be seen that the same output is printed through both codes.

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.3.py
```

7.4 Find the Time Complexities of computing $y(n)$ using FFT/IFFT and convolution and compare.

Solution: The below codes generate and plot the time complexities of computing $y(n)$ using FFT/IFFT and Convolution:

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.4.c
```

```
wget https://github.com/aroshishp/EE1205/
blob/main/Audio_Filtering/codes/7.4.py
```

Time complexity of FFT/IFFT method is $O(n \log(n))$ and that of Convolution method is $O(n^2)$.

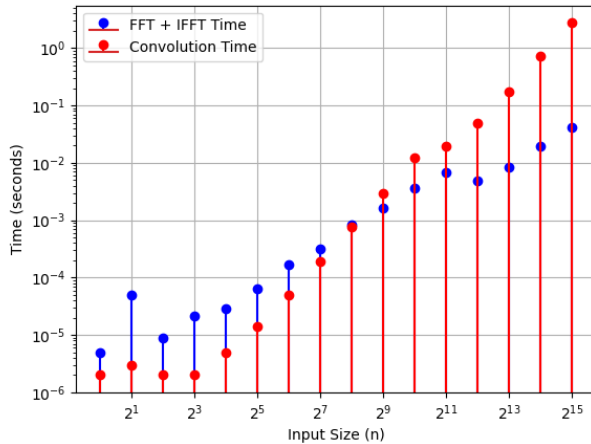


Fig. 7.4: Comparison of Time Complexities

7.5 What is the sampling frequency of the input signal?

Solution: Sampling frequency (f_s) = 44100 Hz. It can be printed from 2.3.

7.6 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 = 6kHz.

7.7 Modifying the code with different input parameters and to get the best possible output.

Solution: The best filtered audio output was obtained by setting order to 4 and keeping cutoff frequency at 6000 Hz. These parameters were used for the spectrogram output as well as for this section.