

Audio Filter

EE23BTECH11037 - M Esha *

I. SOFTWARE INSTALLATION

I.1 Run the following commands(for laptop)

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

I.2 Run the following commands(for termex(mobile))

```
apt-get update
apt-get install libffi-dev libsndfile1 python3-
scipy python3-numpy python3-
matplotlib
apt install python3-cffi python3-soundfile
```

II. DIGITAL FILTER

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

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/Esha1.wav>

II.2 You will find a spectrogram at <https://academo.org/demos/spectrum-analyzer>. Upload the sound file that you downloaded in Problem in the spectrogram and play. Observe the spectrogram. What do you find?

Solution: The audio file is analyzed using spectrogram using the online platform <https://academo.org/demos/spectrum-analyzer>. There are a lot of yellow lines between 440 Hz to 5KHz. These represent the synthesizer key tones. Also, the key strokes are audible along with background noise.

II.3 A Python Code is written to achieve Audio Noise Filtering

```
import soundfile as sf
from scipy import signal
```

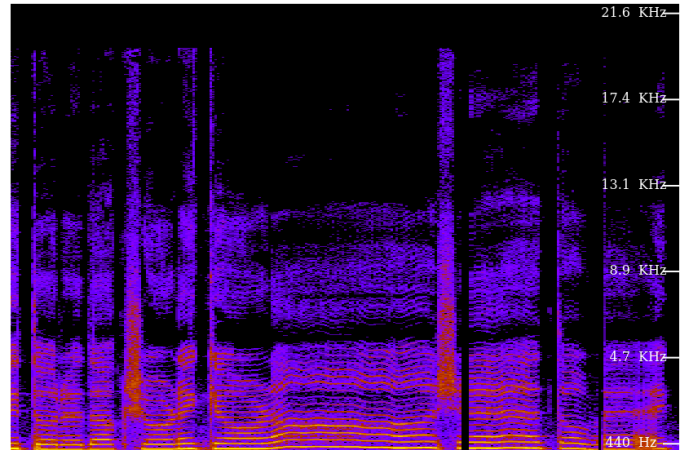


Fig. 1. Spectrogram of the audio file before Filtering

```
#read .wav file
input_signal,fs = sf.read('Esha1.wav')

#sampling frequency of Input signal
sampl_freq=fs

#order of the filter
order=4

#cutoff frequency 4kHz
cutoff_freq=5000.0

#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(a);
print(b);

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

$$H(z) = \frac{Y(z)}{X(z)} \quad (12)$$

from (2) assuming that the Z-transform is a linear operation.

Solution: Applying (11) in (2),

$$Y(z) + \frac{1}{2}z^{-1}Y(z) = X(z) + z^{-2}X(z) \quad (13)$$

$$\Rightarrow \frac{Y(z)}{X(z)} = \frac{1 + z^{-2}}{1 + \frac{1}{2}z^{-1}} \quad (14)$$

IV.3 Find the Z transform of

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

and show that the Z-transform of

$$u(n) = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

is

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

Solution: It is easy to show that

$$\delta(n) \xleftrightarrow{Z} 1 \quad (18)$$

and from (16),

$$U(z) = \sum_{n=0}^{\infty} z^{-n} \quad (19)$$

$$= \frac{1}{1 - z^{-1}}, \quad |z| > 1 \quad (20)$$

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

IV.4 Show that

$$a^n u(n) \xleftrightarrow{Z} \frac{1}{1 - az^{-1}} \quad |z| > |a| \quad (21)$$

Solution:

$$a^n u(n) \xleftrightarrow{Z} \sum_{n=-\infty}^{\infty} a^n u(n) z^{-n} \quad (22)$$

$$= \sum_{n=0}^{\infty} a^n z^{-n} \quad (23)$$

$$= 1 + az^{-1} + a^2 z^{-2} + \dots \quad (24)$$

$$= \frac{1}{1 - az^{-1}} \quad |z| > |a| \quad (25)$$

IV.5 Let

$$H(e^{j\omega}) = H(z = e^{j\omega}). \quad (26)$$

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/Eshamankar/Audio-Filter/tree/main/codes/m.py>

Substituting $z = e^{j\omega}$ in (14), we get

$$|H(e^{j\omega})| = \left| \frac{1 + e^{-2j\omega}}{1 + \frac{1}{2}e^{-j\omega}} \right| \quad (27)$$

$$= \sqrt{\frac{(1 + \cos 2\omega)^2 + (\sin 2\omega)^2}{\left(1 + \frac{1}{2}\cos \omega\right)^2 + \left(\frac{1}{2}\sin \omega\right)^2}} \quad (28)$$

$$= \frac{4|\cos \omega|}{\sqrt{5 + 4\cos \omega}} \quad (29)$$

$$|H(e^{j(\omega+2\pi)})| = \frac{4|\cos(\omega + 2\pi)|}{\sqrt{5 + 4\cos(\omega + 2\pi)}} \quad (30)$$

$$= \frac{4|\cos \omega|}{\sqrt{5 + 4\cos \omega}} \quad (31)$$

$$= |H(e^{j\omega})| \quad (32)$$

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

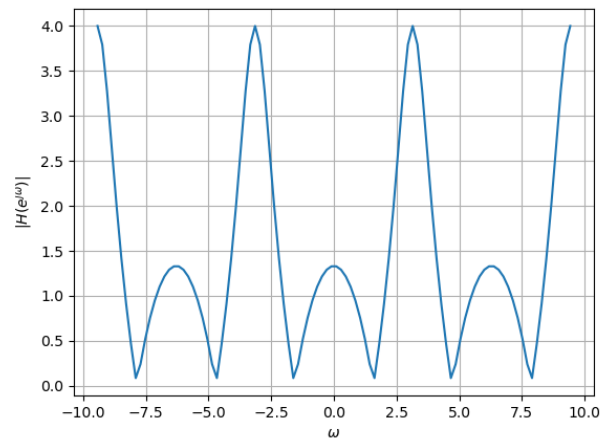


Fig. 4. $|H(e^{j\omega})|$

V. IMPULSE RESPONSE

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

$$h(n) \xleftrightarrow{Z} H(z) \quad (33)$$

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 (14),

$$H(z) = \frac{1}{1 + \frac{1}{2}z^{-1}} + \frac{z^{-2}}{1 + \frac{1}{2}z^{-1}} \quad (34)$$

$$\Rightarrow h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \quad (35)$$

using (21) and (11).

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

Solution:

$$h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \quad (36)$$

$$h(n) \text{ is convergent equation} \quad (37)$$

$$\left(-\frac{1}{2}\right)^n \rightarrow 0, \text{ when } n \rightarrow \infty \text{ So,} \quad (38)$$

$$h(n) \rightarrow 0, \text{ when } n \rightarrow \infty \quad (39)$$

The following code plots $h(n)$

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/l.py>

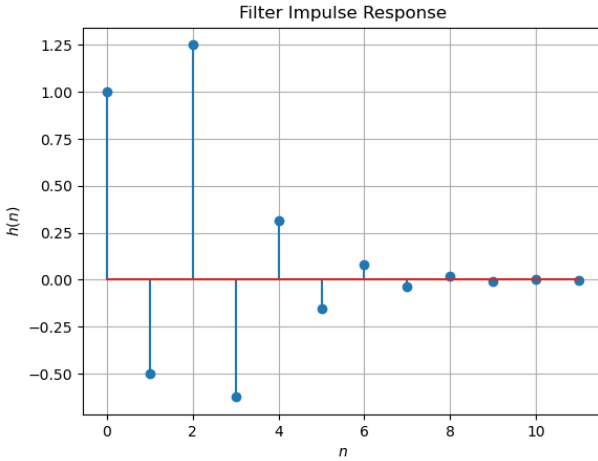


Fig. 5. $h(n)$ as the inverse of $H(z)$

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

$$\sum_{n=-\infty}^{\infty} h(n) < \infty \quad (40)$$

Is the system defined by (2) stable for the impulse response in (33)?

Solution: For stable system (40) should converge.

By using ratio test for convergence:

$$\lim_{n \rightarrow \infty} \left| \frac{h(n+1)}{h(n)} \right| < 1 \quad (41)$$

$$(42)$$

For large n

$$u(n) = u(n-2) = 1 \quad (43)$$

$$\lim_{n \rightarrow \infty} \left(\frac{h(n+1)}{h(n)} \right) = 1/2 < 1 \quad (44)$$

Hence it is stable.

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

$$h(n) + \frac{1}{2}h(n-1) = \delta(n) + \delta(n-2), \quad (45)$$

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/Eshamankar/Audio-Filter/tree/main/codes/k.py>

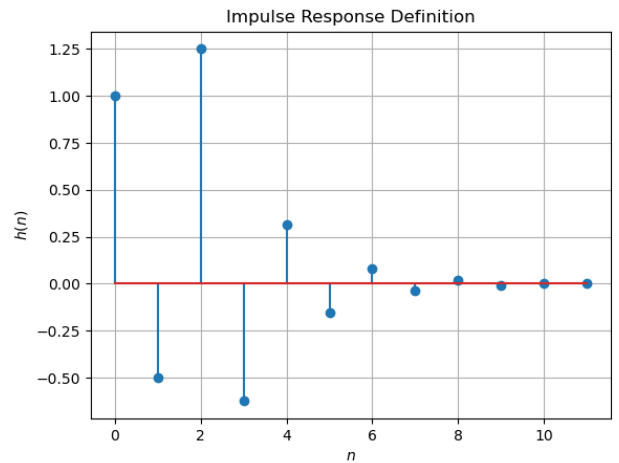


Fig. 6. $h(n)$ from the definition is same as Fig. 5

V.5 Compute

$$y(n) = x(n) * h(n) = \sum_{n=-\infty}^{\infty} x(k)h(n-k) \quad (46)$$

Comment. The operation in (46) 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/Eshamankar/Audio-Filter/tree/main/codes/p.py>

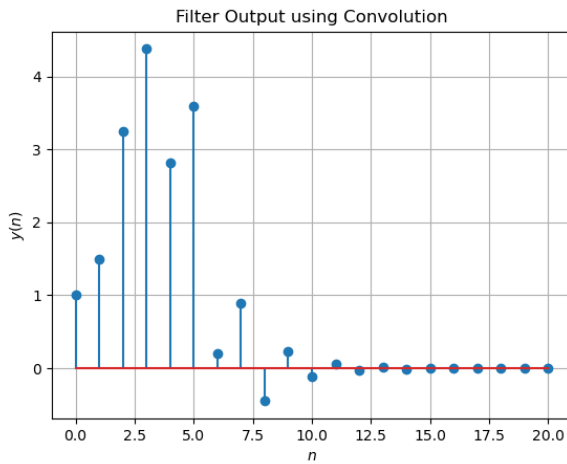


Fig. 7. $y(n)$ from the definition of convolution

V.6 Show that

$$y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \quad (47)$$

Solution: In (46), we substitute $k = n - k$ to get

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (48)$$

$$= \sum_{n-k=-\infty}^{\infty} x(n-k)h(k) \quad (49)$$

$$\Rightarrow y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \quad (50)$$

Hence, proved

VI. DFT AND FFT

VI.1 Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n)e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \quad (51)$$

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

Solution:

$$h(n) = \left(-\frac{1}{2}\right)^n u(n) + \left(-\frac{1}{2}\right)^{n-2} u(n-2) \quad (52)$$

$$H(k) = \sum_{n=0}^{N-1} h(n)e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \quad (53)$$

VI.2 Compute

$$Y(k) = X(k)H(k) \quad (54)$$

VI.3 Compute

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k) \cdot e^{j2\pi kn/N}, \quad n = 0, 1, \dots, N-1 \quad (55)$$

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

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/q.py>

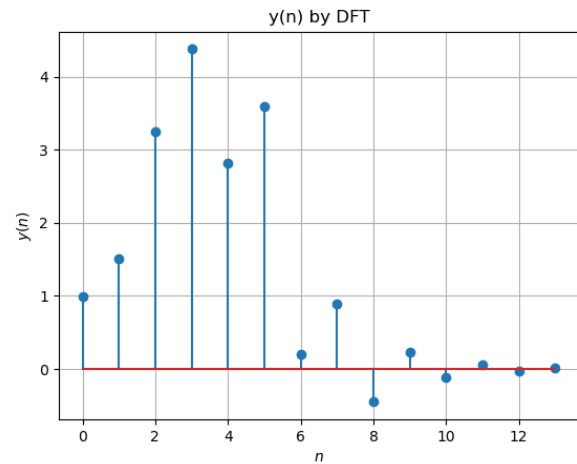


Fig. 8. $y(n)$ obtained from IDFT is plotted

VI.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/Eshamankar/Audio-Filter/tree/main/codes/r.py>

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



Fig. 9. $y(n)$ obtained from IDFT and IFFT is plotted and verified



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

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

Solution: The DFT matrix is defined as :

$$\mathbf{W} = \begin{pmatrix} \omega^0 & \omega^0 & \dots & \omega^0 \\ \omega^0 & \omega^1 & \dots & \omega^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^0 & \omega^{N-1} & \dots & \omega^{(N-1)(N-1)} \end{pmatrix} \quad (56)$$

where $\omega = e^{-j\frac{2\pi}{N}}$. Now any DFT equation can be written as

$$\mathbf{X} = \mathbf{W}\mathbf{x} \quad (57)$$

where

$$\mathbf{x} = \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(n-1) \end{pmatrix} \quad (58)$$

$$\mathbf{X} = \begin{pmatrix} X(0) \\ X(1) \\ \vdots \\ X(n-1) \end{pmatrix} \quad (59)$$

Thus we can rewrite (54) as:

$$\mathbf{Y} = \mathbf{X} \odot \mathbf{H} = (\mathbf{W}\mathbf{x}) \odot (\mathbf{W}\mathbf{h}) \quad (60)$$

where the \odot represents the Hadamard product which performs element-wise multiplication.

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/s.py>

VII. EXERCISES

Answer the following questions by looking at the python code in Problem II.3.

VII.1 The command

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

in Problem II.3 is executed through the following difference equation

$$\sum_{m=0}^M a(m)y(n-m) = \sum_{k=0}^N b(k)x(n-k) \quad (61)$$

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/Eshamankar/Audio-Filter/tree/main/codes/t.py>

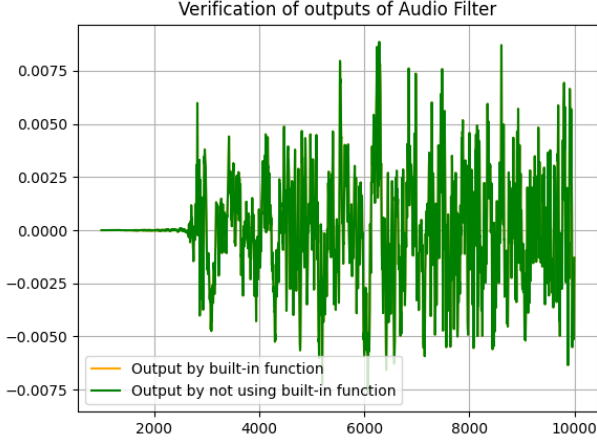


Fig. 11. Both the outputs using and without using function overlap

VII.2 Repeat all the exercises in the previous sections for the above a and b .

Solution: The code in II.3 generates the values of a and b which can be used to generate a difference equation.

And,

$$M = 5 \quad (62)$$

$$N = 5 \quad (63)$$

From 61

$$a(0)y(n) + a(1)y(n-1) + a(2)y(n-2) + a(3)y(n-3) + a(4)y(n-4) = b(0)x(n) + b(1)x(n-1) + b(2)x(n-2) + b(3)x(n-3) + b(4)x(n-4) \quad (64)$$

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

Difference Equation is given by :

$$\begin{aligned} & y(n) - (2.15448443)y(n-1) + (1.98942448)y(n-2) \\ & - (0.86509739)y(n-3) + (0.14814218)y(n-4) \\ & = (0.00737405 \times 10^{-5})x(n) + (0.02949621 \times 10^{-5})x(n-1) \\ & + (0.04424432 \times 10^{-5})x(n-2) + (0.02949621 \times 10^{-5})x(n-3) \\ & + (0.00737405 \times 10^{-5})x(n-4) \end{aligned} \quad (65)$$

From (61)

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \dots + b_Mz^{-M}}{a_0 + a_1z^{-1} + a_2z^{-2} + \dots + a_Nz^{-N}} \quad (66)$$

$$H(z) = \frac{\sum_{k=0}^N b(k)z^{-k}}{\sum_{k=0}^M a(k)z^{-k}} \quad (67)$$

Partial fraction on (67) can be generalised as:

$$H(z) = \sum_i \frac{r(i)}{1 - p(i)z^{-1}} + \sum_j k(j)z^{-j} \quad (68)$$

Now,

$$a^n u(n) \xleftrightarrow{z} \frac{1}{1 - az^{-1}} \quad (69)$$

$$\delta(n - k) \xleftrightarrow{z} z^{-k} \quad (70)$$

Taking inverse z transform of (68) by using (69) and (70)

$$h(n) = \sum_i r(i)[p(i)]^n u(n) + \sum_j k(j)\delta(n - j) \quad (71)$$

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

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/u.py>

$r(i)$	$p(i)$	$k(i)$
$0.27794949 - 0.95005714j$	$0.47186091 + 0.15595655j$	0.04977686
$0.27794949 + 0.95005714j$	$0.47186091 - 0.15595655j$	$-$
$-0.2991509 + 0.15635781j$	$0.60538131 + 0.48305248j$	$-$
$-0.2991509 - 0.15635781j$	$0.60538131 - 0.48305248j$	$-$

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

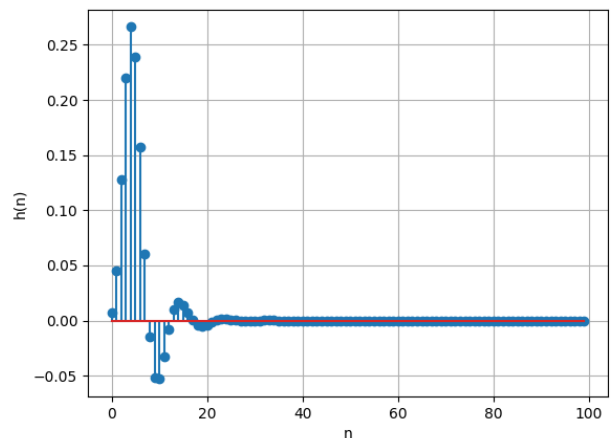


Fig. 12. $h(n)$ of Audio Filter

Stability of $h(n)$:

According to (40)

$$H(z) = \sum_{n=0}^{\infty} h(n) z^{-n} \quad (72)$$

$$H(1) = \sum_{n=0}^{\infty} h(n) = \frac{\sum_{k=0}^N b(k)}{\sum_{k=0}^M a(k)} < \infty \quad (73)$$

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

The below code plots Filter frequency response

<https://github.com/Eshamankar/Audio-Filter/tree/main/codes/v.py>

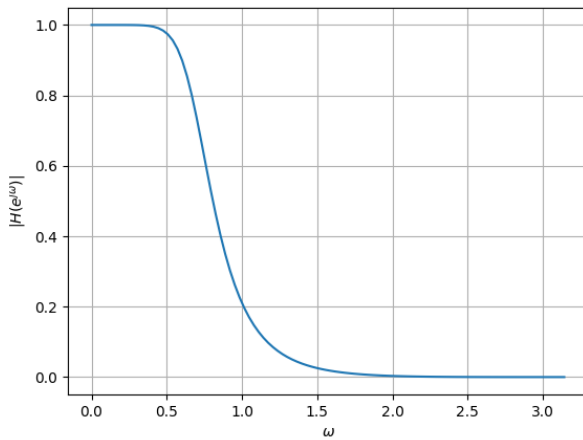


Fig. 13. Frequency Response of Audio Filter

VII.3 What is the sampling frequency of the input signal?

Solution: The Sampling Frequency is 48.0KHz

VII.4 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=5kHz.

VII.5 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.