

Audio Filter

ee23btech11223 - Soham Prabhakar More

I. DIGITAL FILTER

I.1 The Audio file is taken from:

https://github.com/Soham-More/EE1205/blob/main/audio_filter/audio/alt.wav

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

```
import soundfile as sf
from scipy import signal

#read .wav file
input_signal,fs = sf.read('../audio/alt.wav')

#sampling frequency of Input signal
sampl_freq=fs

print(fs)

#order of the filter
order=4

#cutoff frequency 4kHz
cutoff_freq=6000.0

#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 =', a)
print('b =', b)

#filter the input signal with butterworth filter
#Specify axis, since input is stereo
output_signal = signal.filtfilt(b, a,
    input_signal, axis=0)
#output_signal = signal.lfilter(b, a,
    input_signal)

#write the output signal into .wav file
```

```
sf.write('../audio/alt_filtered.wav',
    output_signal, fs)
```

I.3 The audio file is analyzed using spectrogram using the online platform <https://academo.org/demos/spectrum-analyzer>.

The orange, yellow regions have high amplitudes and purple small amplitudes.

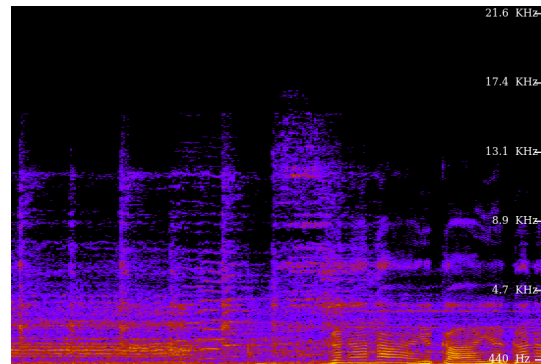


Fig. I.3: Spectrogram of the audio file before Filtering

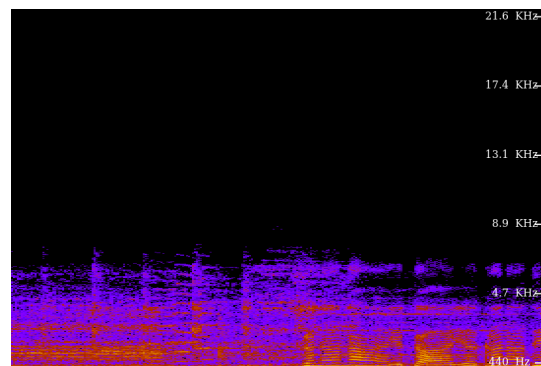


Fig. I.3: Spectrogram of the audio file after Filtering

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 $y(n)$.

Solution: The following C code generates $y(n)$:

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/gen.c

This python code plots $x(n)$ and $y(n)$:

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/x_n.py

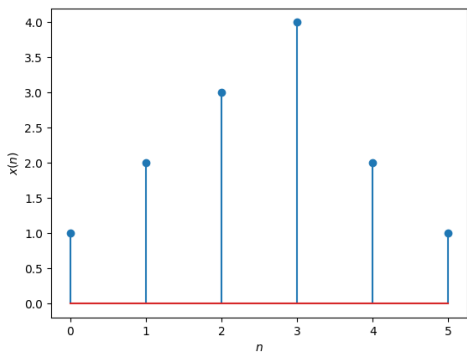


Fig. II.2: Plot of $x(n)$

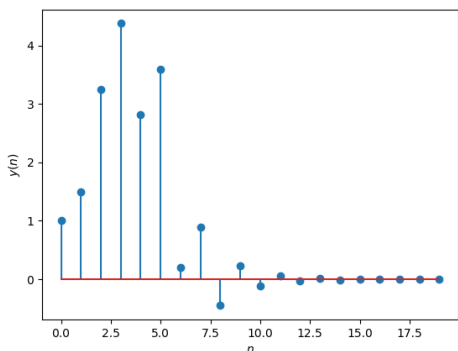


Fig. II.2: Plot of $y(n)$

III. Z-TRANSFORM

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

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

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),

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

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

$$= z^{-k} \sum_{n=-\infty}^{\infty} x(n)z^{-n} \quad (8)$$

Substituting $k = 1$ we get (4).

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),

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

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

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

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

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

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

Solution:

$$a^n u(n) \xleftrightarrow{Z} \sum_{n=0}^{\infty} (az^{-1})^n \quad (19)$$

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

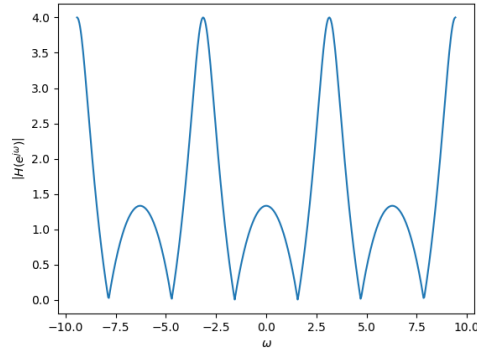


Fig. III.5: Plot of $H(e^{j\omega})$

III.5 Let

$$H(e^{j\omega}) = H(z = e^{j\omega}). \quad (21)$$

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/Soham-More/EE1205/blob/main/audio_filter/codes/H_jw.py

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

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

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

$$\therefore |H(e^{j\omega})| = \frac{4|\cos \omega|}{\sqrt{5 + 4 \cos \omega}} \quad (24)$$

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

$$= \frac{4|\cos \omega|}{\sqrt{5 + 4 \cos \omega}} \quad (26)$$

$$= |H(e^{j\omega})| \quad (27)$$

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

IV. IMPULSE RESPONSE

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

$$h(n) \xleftrightarrow{Z} H(z) \quad (28)$$

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),

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

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

using (18) and (8).

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

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

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/H_jw.py

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

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

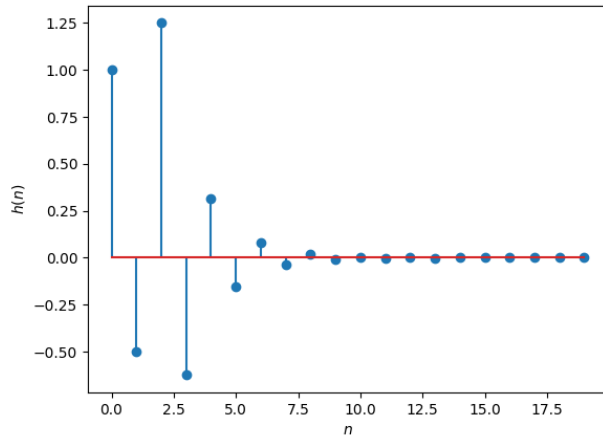
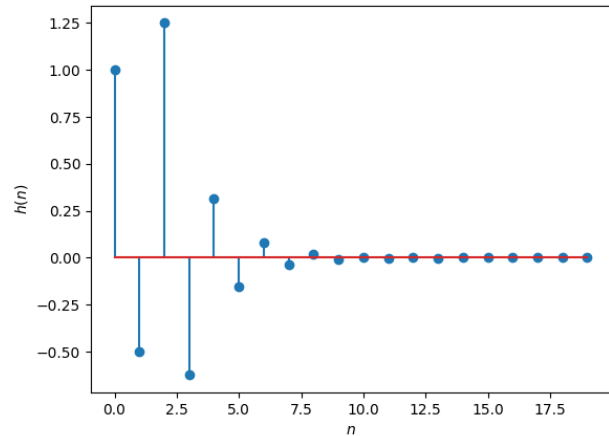
Is the system defined by (2) stable for the impulse response in (28)?

Solution: For stable system (31) should converge.

By using ratio test

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

$$(33)$$

Fig. IV.2: Plot of $h(n)$ as the inverse of $H(z)$ Fig. IV.4: $h(n)$ from the definition is same as Fig. IV.2

For large n

$$u(n) = u(n-2) = 1 \quad (34)$$

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

Therefore it converges. Hence it is stable.

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

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

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. IV.4. Note that this is the same as Fig. IV.2.

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/h_ir.py

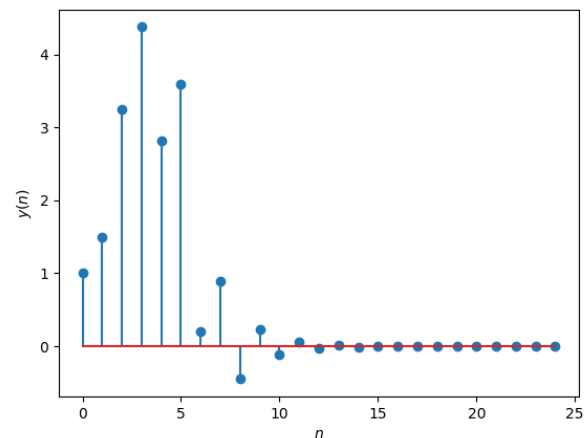
IV.5 Compute

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

Comment. The operation in (37) is known as *convolution*.

Solution: The following code plots Fig. IV.5. Note that this is the same as $y(n)$ in Fig. ??.

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/h_ir.py

Fig. IV.5: $y(n)$ from the definition of convolution

IV.6 Show that

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

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

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

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

$$\therefore y(n) = \sum_{k=-\infty}^{\infty} x(n-k)h(k) \quad (41)$$

V. DFT AND FFT

V.1 Compute

$$X(k) \triangleq \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}} \quad k = 0, 1, \dots, N-1 \quad (42)$$

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

V.2 Compute

$$Y(k) = X(k)H(k) \quad (43)$$

V.3 Compute

$$y(n) = \frac{1}{N} \sum_{k=0}^{N-1} Y(k) \cdot e^{\frac{j2\pi kn}{N}} \quad n = 0, 1, \dots, N-1 \quad (44)$$

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

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/dft.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/Soham-More/EE1205/blob/main/audio_filter/codes/fft.py

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

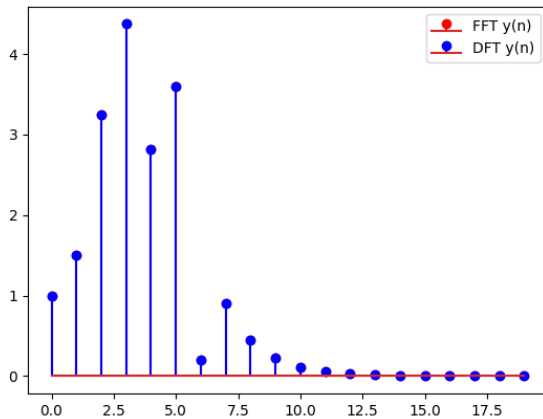


Fig. V.4: $y(n)$ obtained from IDFT and IFFT is plotted and verified

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

Solution: The DFT matrix is defined as :

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

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

$$\mathbf{X} = \mathbf{W}\mathbf{x} \quad (46)$$

where

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

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

Thus we can rewrite (43) as:

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

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

VI. EXERCISES

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

VI.1 The command

```
output_signal = signal.filtfilt(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/Soham-More/EE1205/blob/main/audio_filter/codes/lfilt.py

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/poly.py

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

$$a(0)y(n) + a(1)y(n-1) + a(2)y(n-2) + a(3)y(n-3) + a(4)y(n-4) = b(0)x(n) + b(1)x(n-1) + b(2)x(n-2) + b(3)x(n-3) + b(4)x(n-4) \quad (53)$$

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

Difference Equation is given by :

$$\begin{aligned} & y(n) - (1.968)y(n-1) + (1.735)y(n-2) \\ & - (0.724)y(n-3) + (0.120)y(n-4) \\ & = (0.01020)x(n) + (0.0408)x(n-1) \\ & + (0.06125)x(n-2) + (0.0408)x(n-3) \\ & + (0.01020)x(n-4) \end{aligned} \quad (54)$$

From (50)

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2} + \dots + b_Mz^{-M}}{a_0 + a_1z^{-1} + a_2z^{-2} + \dots + a_Nz^{-N}} \quad (55)$$

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

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,

$$a^n u(n) \xleftrightarrow{Z} \frac{1}{1 - az^{-1}} \quad (58)$$

$$\delta(n-k) \xleftrightarrow{Z} z^{-k} \quad (59)$$

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

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

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

i	$r(i)$	$p(i)$	$k(i)$
0	$(0.285 - 1.091j)$	$(0.428 + 0.164j)$	0.085
1	$(0.285 + 1.091j)$	$(0.428 - 0.164j)$	0.0
2	$(-0.322 + 0.176j)$	$(0.557 + 0.514j)$	0.0
3	$(-0.322 - 0.176j)$	$(0.557 - 0.514j)$	0.0

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

Stability of $h(n)$:

According to (31)

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

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

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

The below code plots Filter frequency response

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/response.py

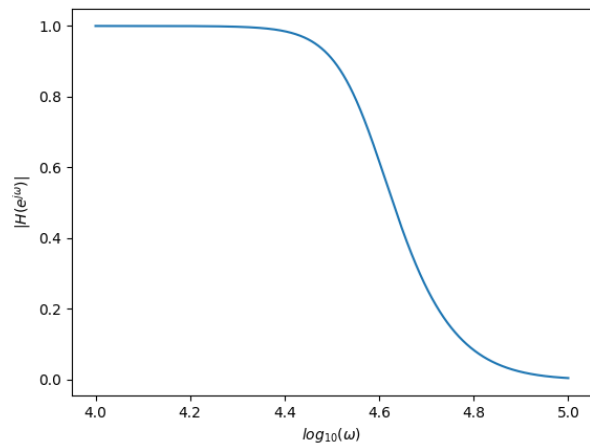


Fig. VI.2: Frequency Response of Digital Filter

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

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/pole.py

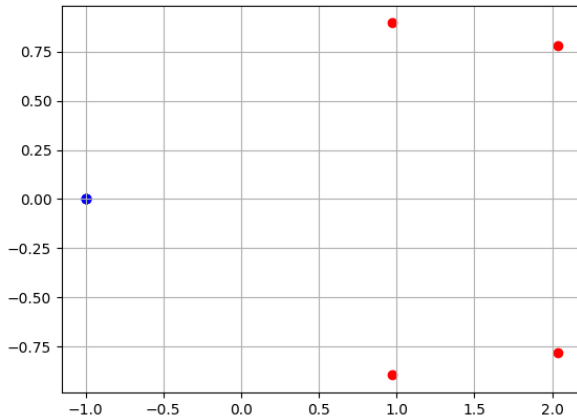


Fig. VI.2: There are complex poles. So $h(n)$ should be damped sinusoid.

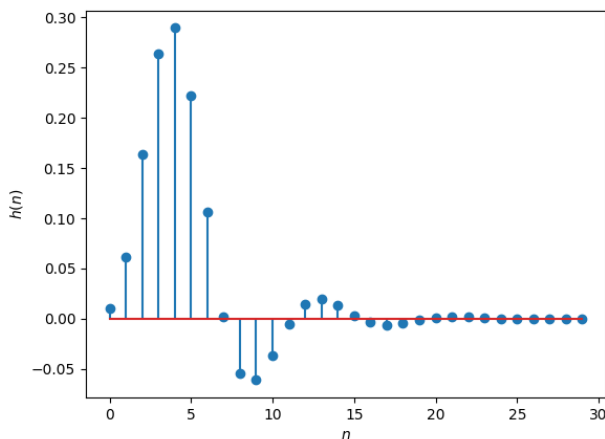


Fig. VI.2: $h(n)$ of Audio Filter. It is a damped sinusoid.

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/Soham-More/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/Soham-More/EE1205/blob/main/audio_filter/codes/c_fft.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/Soham-More/EE1205/blob/main/audio_filter/codes/fft_time.c

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

https://github.com/Soham-More/EE1205/blob/main/audio_filter/codes/time_plot.py

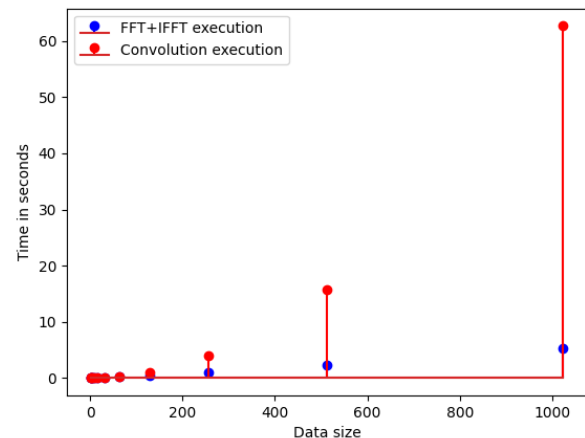


Fig. VI.4: The Complexity of FFT+IFFT method is $O(n \log n)$ whereas by convolution is $O(n^2)$

VI.5 What is the sampling frequency of the input signal?

Solution: The Sampling Frequency is 44.1 KHz

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=1 kHz.

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.