29/08/2025

https://github.com/Manikanta199-vlsi/MANIPAL/tree/main/Analog/Assign1

Analog Assignment -1 Codes link

https://github.com/Manikanta199-vlsi/MANIPAL/tree/Shell_script/Shell

Shell Scripting Assignment Codes Link

all codes can be easily accessed from the above link

Analog Assignment:- Done in PYTHON

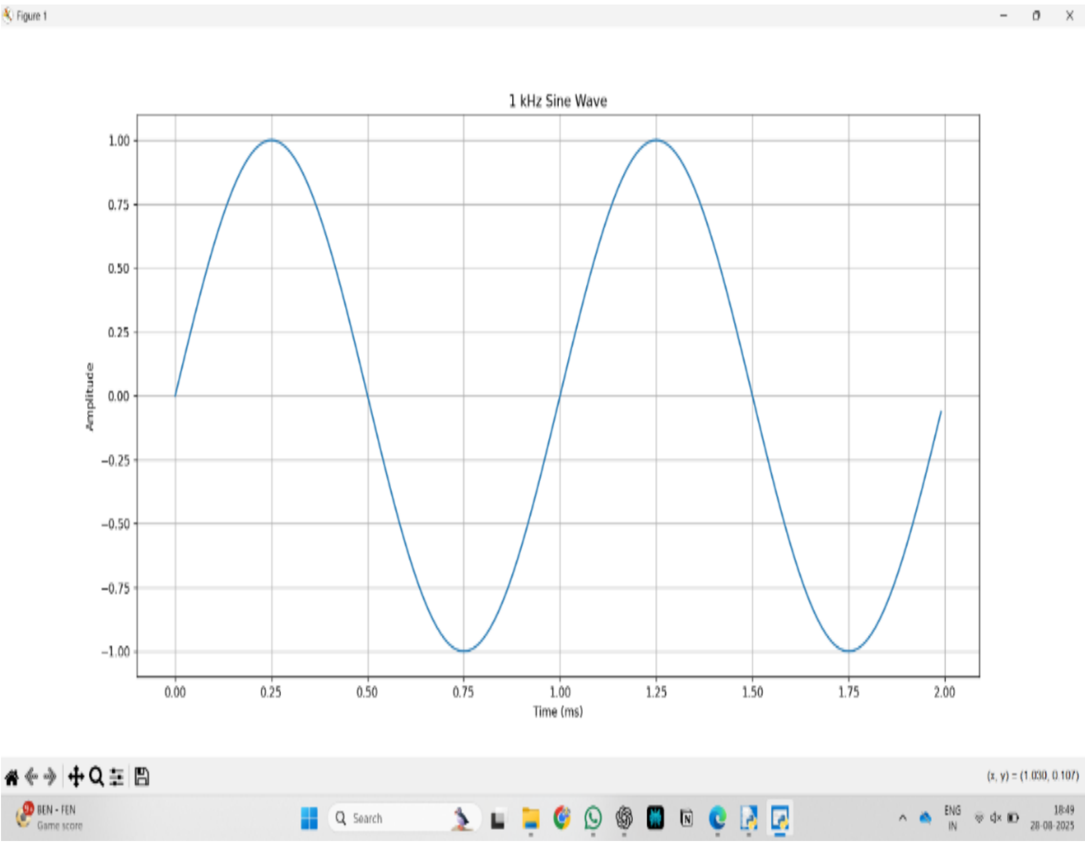
Question 1)

1.a)

```
import numpy as np

import matplotlib.pyplot as plt

# Parameters

f = 1000        # frequency = 1 kHz

Fs = 100000      # sampling rate = 100 kHz (100 samples per cycle)

T = 1/Fs        # sampling interval

t = np.arange(0, 2e-3, T)   # 2 ms duration (enough for 2 cycles)

# Generate sine wave

A = 1           # amplitude = 1

y = A * np.sin(2 * np.pi * f * t)

# Plot
```

```python
plt.figure(figsize=(8,4))
plt.plot(t*1000, y)   # time in milliseconds
plt.title("1 kHz Sine Wave")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.show()
```

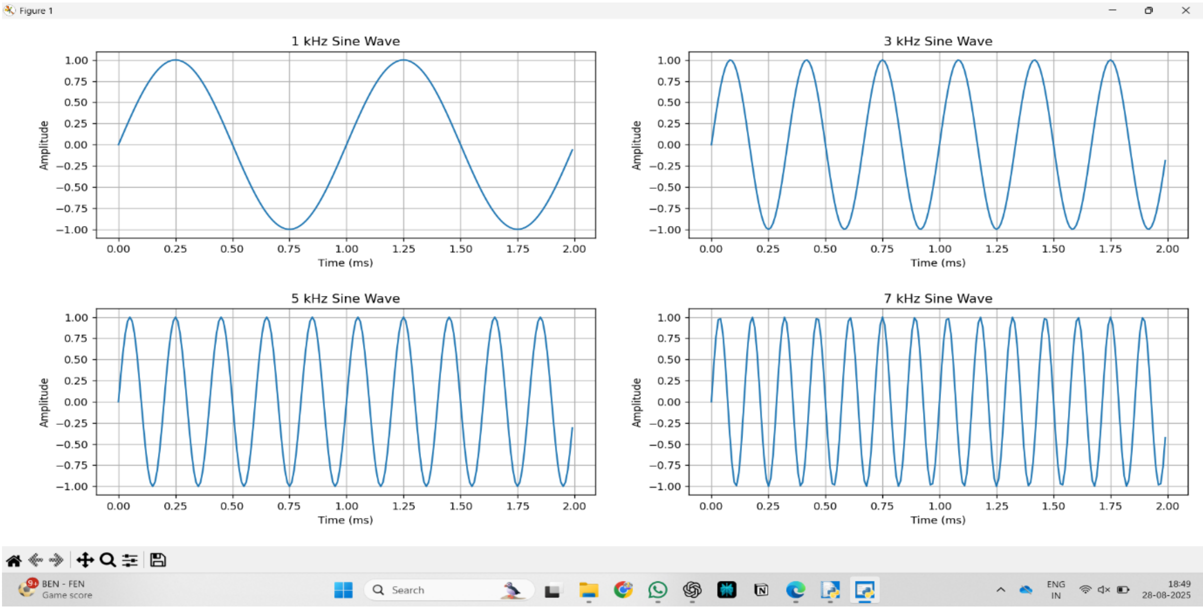
OUTPUT :- 1KHZ WAVE

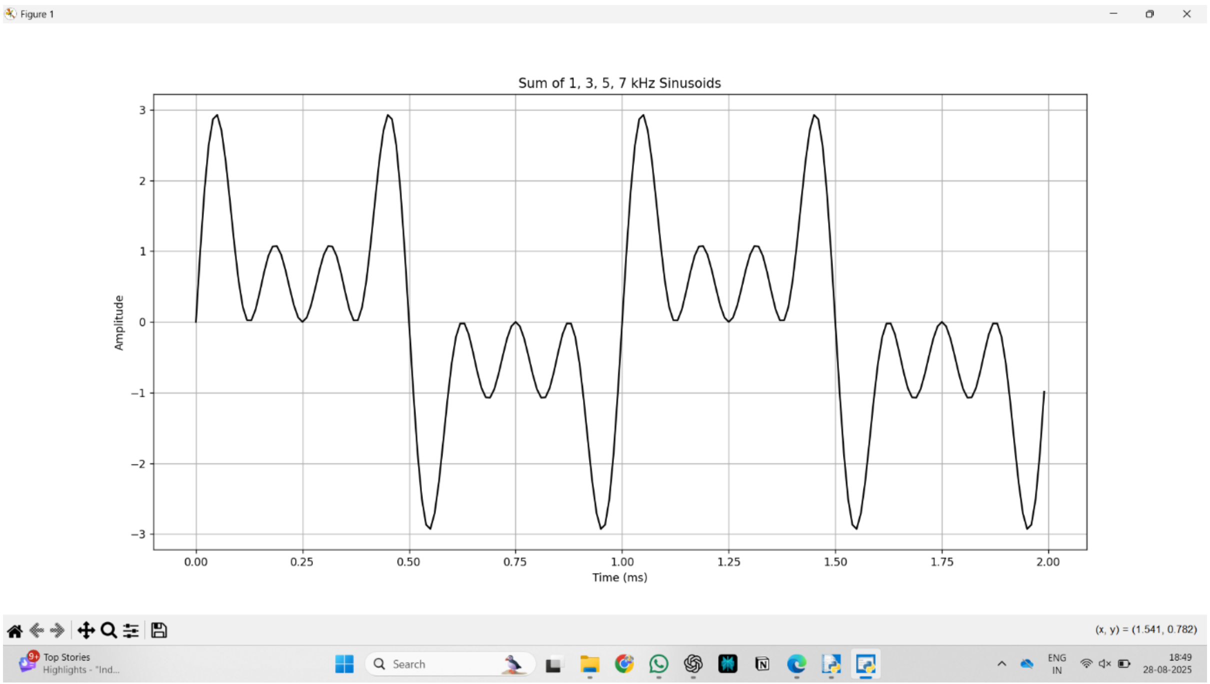


1 .b)

```python
import numpy as np

import matplotlib.pyplot as plt


# Sampling parameters

Fs = 100000      # sampling rate (100 kHz)

T = 1/Fs        # sampling interval

t = np.arange(0, 2e-3, T)   # 2 ms duration


# Frequencies

freqs = [1000, 3000, 5000, 7000]  # Hz


# ---------------- PAGE 1: Individual signals ----------------

fig, axs = plt.subplots(2, 2, figsize=(10,6))

axs = axs.ravel()


for i, f in enumerate(freqs):

   y = np.sin(2*np.pi*f*t)

   axs[i].plot(t*1000, y)

   axs[i].set_title(f"{f/1000:.0f} kHz Sine Wave")

   axs[i].set_xlabel("Time (ms)")

   axs[i].set_ylabel("Amplitude")

   axs[i].grid(True)


plt.tight_layout()

plt.show()


# ---------------- PAGE 2: Sum of signals ----------------

y_sum = np.zeros_like(t)

for f in freqs:

   y_sum += np.sin(2*np.pi*f*t)
```

```
plt.figure(figsize=(10,4))
plt.plot(t*1000, y_sum, color='black')
plt.title("Sum of 1, 3, 5, 7 kHz Sinusoids")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.grid(True)
plt.show()
```

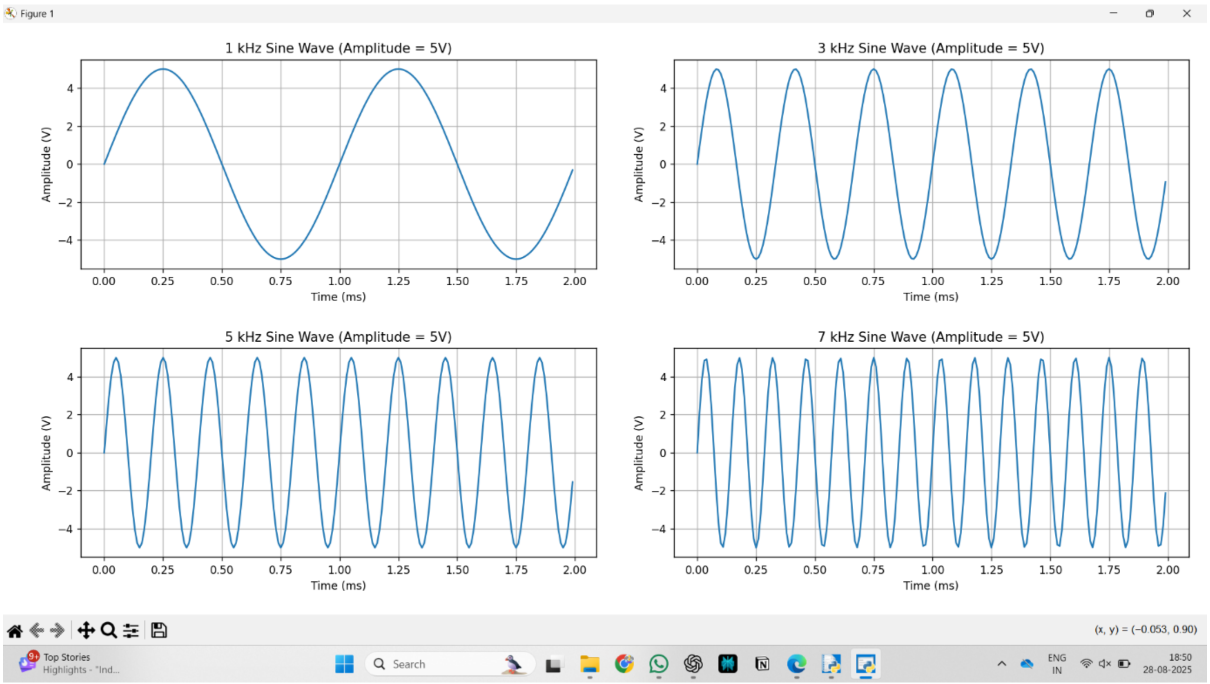




1.c)

```
import numpy as np
import matplotlib.pyplot as plt

# Sampling parameters
Fs = 100000      # sampling rate (100 kHz)
T = 1/Fs         # sampling interval
t = np.arange(0, 2e-3, T)   # 2 ms duration

# Frequencies
freqs = [1000, 3000, 5000, 7000]  # Hz
amplitude = 5   # 5V amplitude

# ---------------- PAGE 1: Individual signals ----------------
fig, axs = plt.subplots(2, 2, figsize=(10,6))
axs = axs.ravel()

for i, f in enumerate(freqs):
   y = amplitude * np.sin(2*np.pi*f*t)
   axs[i].plot(t*1000, y)
   axs[i].set_title(f"{f/1000:.0f} kHz Sine Wave (Amplitude = {amplitude}V)")
   axs[i].set_xlabel("Time (ms)")
   axs[i].set_ylabel("Amplitude (V)")
   axs[i].grid(True)

plt.tight_layout()
plt.show()
```

```
# ---------------- PAGE 2: Sum of signals ----------------
y_sum = np.zeros_like(t)
for f in freqs:
    y_sum += amplitude * np.sin(2*np.pi*f*t)

plt.figure(figsize=(10,4))
plt.plot(t*1000, y_sum, color='black')
plt.title("Sum of 1, 3, 5, 7 kHz Sinusoids (Amplitude = 5V each)")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude (V)")
plt.grid(True)
plt.show()
```



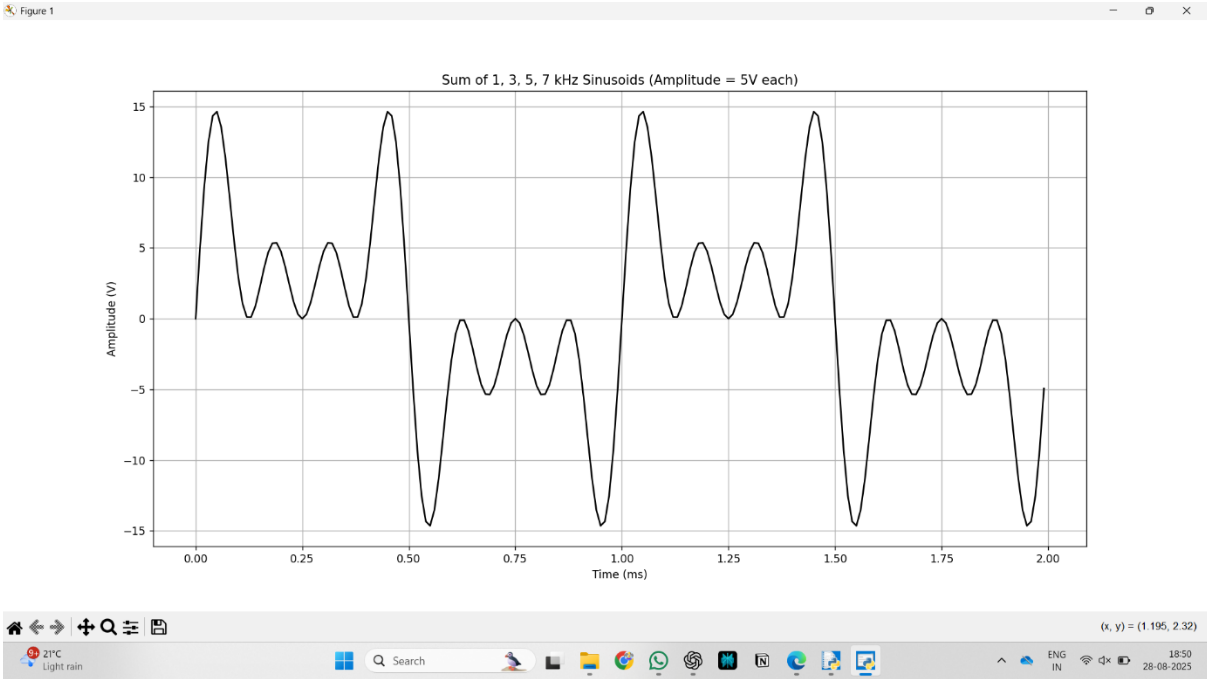
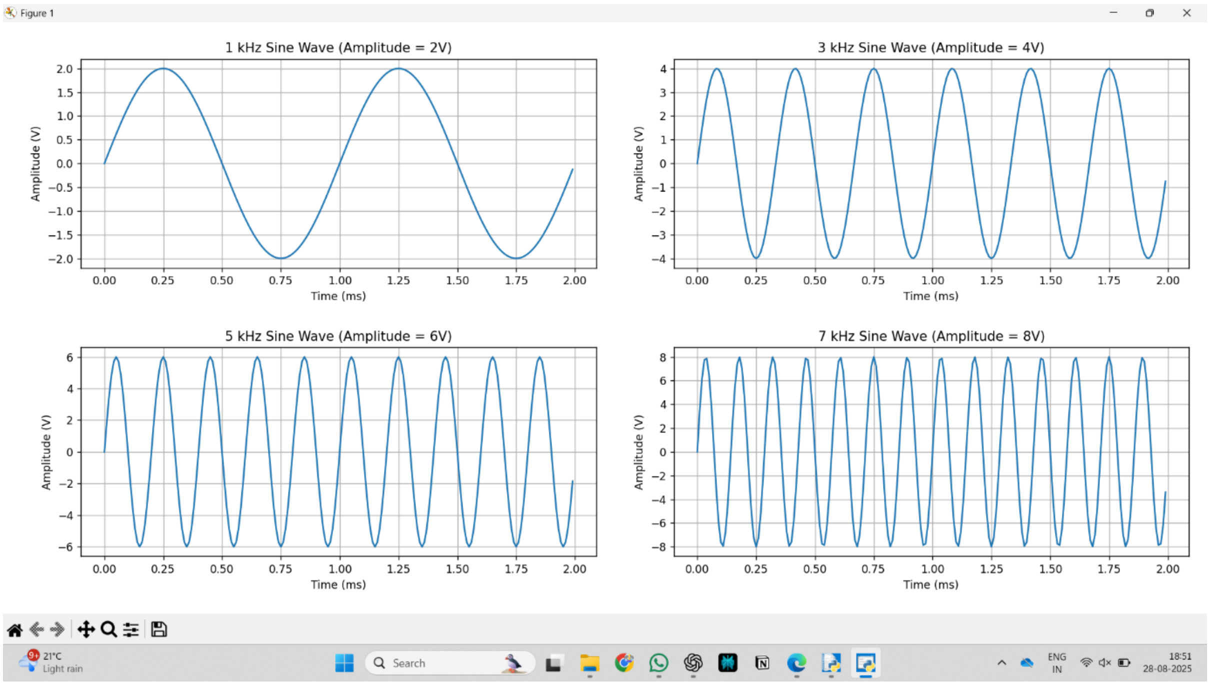


Figure 1
Sum of 1, 3, 5, 7 kHz Sinusoids (Amplitude = 5V each)
Amplitude (V)
Time (ms)
15
10
5
0
−5
−10
−15
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
(x, y) = (1.195, 2.32)
21°C
Light rain
Search
ENG
IN
18:50
28-08-2025

1.d)

```
import numpy as np

import matplotlib.pyplot as plt


# Sampling parameters

Fs = 100000      # sampling rate (100 kHz)

T = 1/Fs        # sampling interval

t = np.arange(0, 2e-3, T)   # 2 ms duration


# Frequencies and corresponding amplitudes

freqs = [1000, 3000, 5000, 7000]      # Hz

amps  = [2, 4, 6, 8]                # Volts


# ---------------- PAGE 1: Individual signals ----------------

fig, axs = plt.subplots(2, 2, figsize=(10,6))

axs = axs.ravel()


for i, (f, A) in enumerate(zip(freqs, amps)):

    y = A * np.sin(2*np.pi*f*t)

    axs[i].plot(t*1000, y)

    axs[i].set_title(f"{f/1000:.0f} kHz Sine Wave (Amplitude = {A}V)")

    axs[i].set_xlabel("Time (ms)")

    axs[i].set_ylabel("Amplitude (V)")

    axs[i].grid(True)


plt.tight_layout()

plt.show()


# ---------------- PAGE 2: Sum of signals ----------------
```

```
y_sum = np.zeros_like(t)
for f, A in zip(freqs, amps):
  y_sum += A * np.sin(2*np.pi*f*t)


plt.figure(figsize=(10,4))
plt.plot(t*1000, y_sum, color='black')
plt.title("Sum of 1, 3, 5, 7 kHz Sinusoids with Different Amplitudes")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude (V)")
plt.grid(True)
plt.show()
```



OBSERVATIONS :- The output wave form begins to take the shape of the SQUARE wave

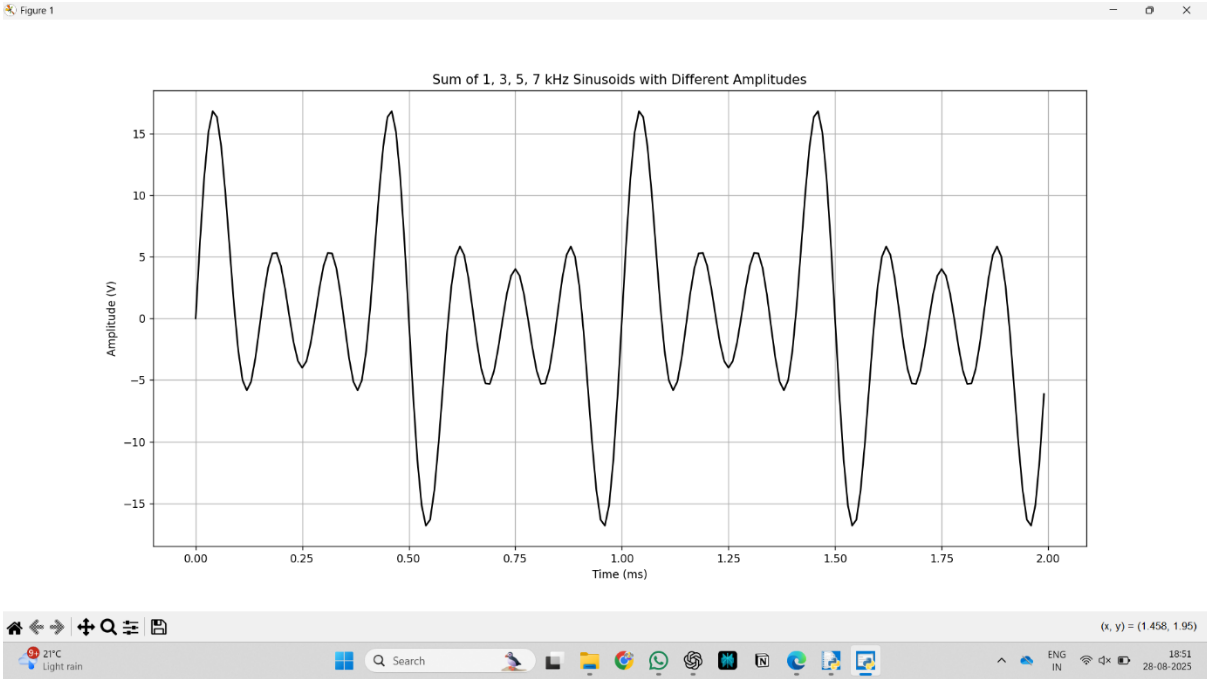


Figure 1
Sum of 1, 3, 5, 7 kHz Sinusoids with Different Amplitudes
Amplitude (V)
Time (ms)
15
10
5
0
−5
−10
−15
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
(x, y) = (1.458, 1.95)
21°C
Light rain
Search
ENG
IN
18:51
28-08-2025

Q – 2)

apsara

Date: __-__-____

(2) (I) e^x ; x is a real number

$$f(x) = e^x$$

Derrivative

$$\frac{d}{dx}(e^x) = e^x > 0 \text{ for all real } x.$$

$\rightarrow$ the function is always increasing

Behaviour Depends on sign of x.

* if $x > 0$; e^x grows rapidly $\rightarrow$ Rising exponent
* if $x < 0$; e^x ~~grows~~ decays $\rightarrow$ decaying exponent

(II) $e^{i\theta}$ where θ is real (imaginary exponent)

Let θ be real and $i = \sqrt{-1}$; Using Eulers formula

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$

The magnitude of $e^{i\theta}$ is : $|e^{i\theta}| = \sqrt{\cos^2\theta + \sin^2\theta} = 1$

Interpretation :-

The real part $= \cos(\theta) \rightarrow$ oscillates b/w -1 and 1
The img part $= \sin(\theta) \rightarrow$ " " -1 & 1

$\Rightarrow$ So $e^{i\theta}$ is a sinusoidal Function in both real & imaginary parts.

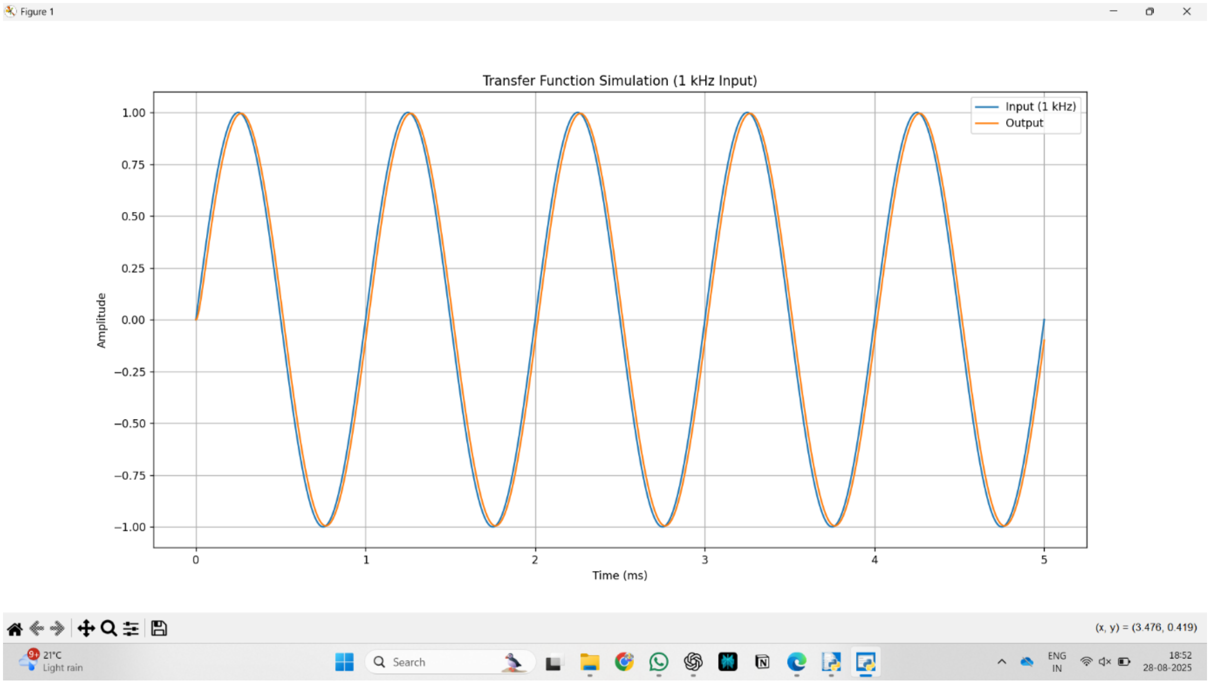
Page No.: ____

Q – 3)

3.a)

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import signal


# Cutoff frequency

fc = 10000

wc = 2 * np.pi * fc


# Transfer function H(s) = wc / (s + wc)

num = [wc]

den = [1, wc]

system = signal.TransferFunction(num, den)


# Input: 1 kHz sinusoid

fsig = 1000

wsig = 2 * np.pi * fsig

t = np.linspace(0, 0.005, 5000)   # 5 ms duration

x = np.sin(wsig * t)


# Simulate response

t_out, y, _ = signal.lsim(system, U=x, T=t)


# Plot input vs output

plt.figure(figsize=(10,5))

plt.plot(t*1000, x, label="Input (1 kHz)")

plt.plot(t*1000, y, label="Output")

plt.xlabel("Time (ms)")

plt.ylabel("Amplitude")

plt.title("Transfer Function Simulation (1 kHz Input)")
```

```
plt.legend()
plt.grid(True)
plt.show()
```



3.b)

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

# Cutoff frequency
fc = 10000
wc = 2 * np.pi * fc
```

```
# Transfer function H(s) = wc / (s + wc)
num = [wc]
den = [1, wc]
system = signal.TransferFunction(num, den)

# Input: 1 kHz sinusoid
fsig = 1000
wsig = 2 * np.pi * fsig
t = np.linspace(0, 0.005, 5000)   # 5 ms duration
x = np.sin(wsig * t)

# Simulate response
t_out, y, _ = signal.lsim(system, U=x, T=t)

# Plot input vs output
plt.figure(figsize=(10,5))
plt.plot(t*1000, x, label="Input (1 kHz)")
plt.plot(t*1000, y, label="Output")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.title("Transfer Function Simulation (1 kHz Input)")
plt.legend()
plt.grid(True)
plt.show()
```

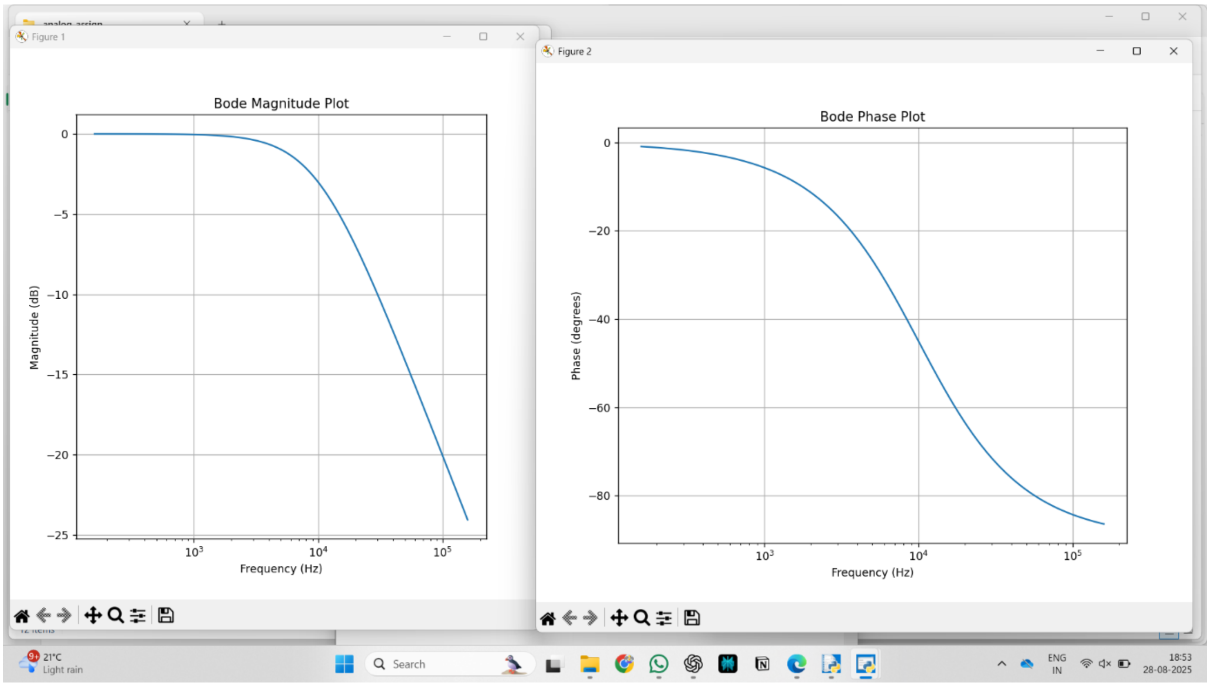


Figure 1
Bode Magnitude Plot
Magnitude (dB)
0
−5
−10
−15
−20
−25
10^3
10^4
10^5
Frequency (Hz)
Figure 2
Bode Phase Plot
Phase (degrees)
0
−20
−40
−60
−80
10^3
10^4
10^5
Frequency (Hz)
21°C
Light rain
Search
ENG
IN
18:53
28-08-2025

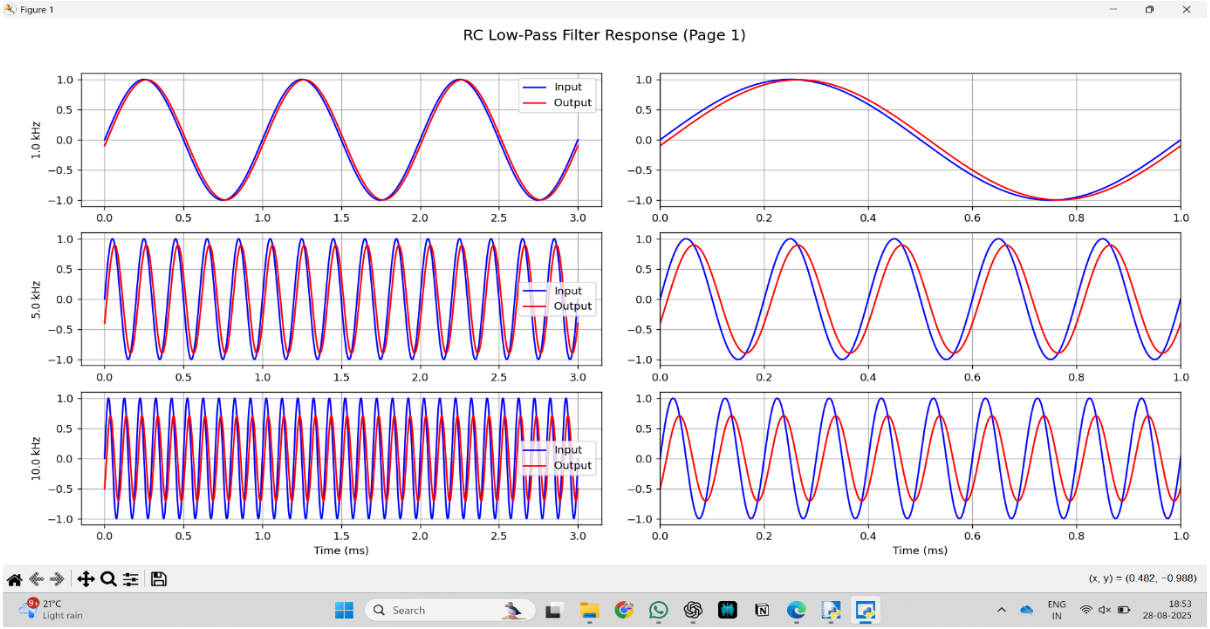
3.c)

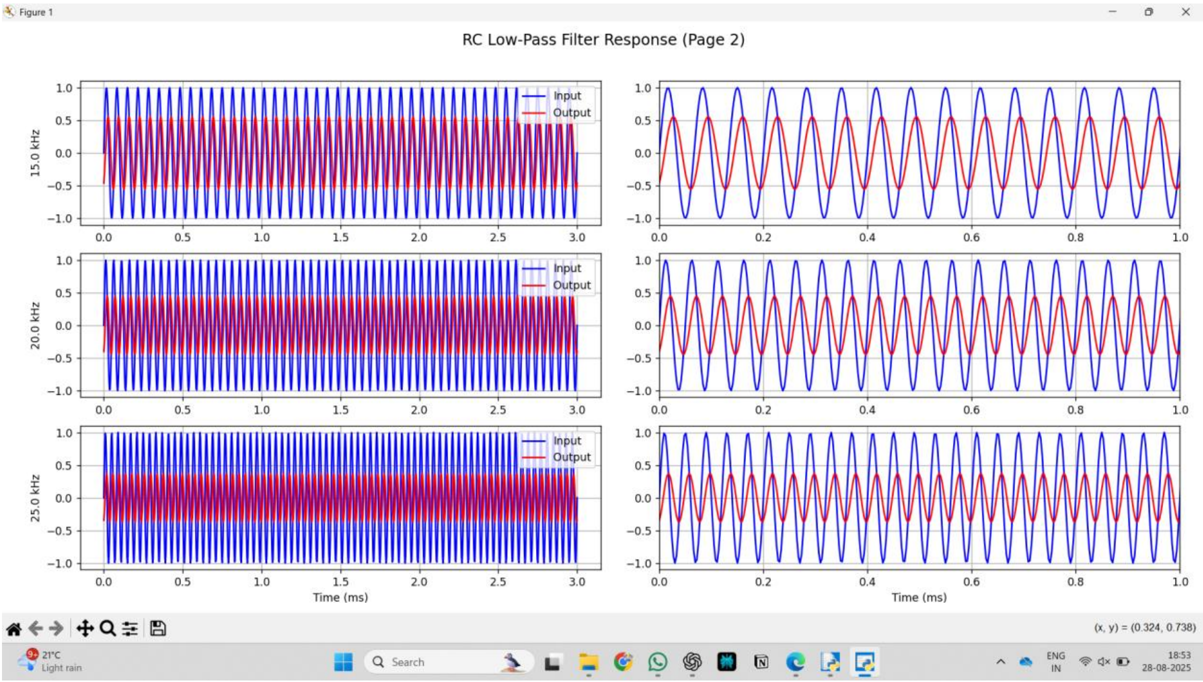
```
import numpy as np

import matplotlib.pyplot as plt


# Cutoff frequency

fc = 10000

wc = 2 * np.pi * fc


# Test frequencies

freqs = [1000, 5000, 10000, 15000, 20000, 25000]


# Time vector (long enough to show cycles)

t = np.linspace(0, 3e-3, 1000)  # 3 ms


# Store signals

inputs = []

outputs = []


for f in freqs:

    w = 2 * np.pi * f

    # Transfer function H(jw)

    H = wc / (1j*w + wc)

    mag = abs(H)

    phase = np.angle(H)


    # Input = 1 V amplitude

    x = np.sin(w*t)

    # Output = attenuated + shifted
```

```
    y = mag * np.sin(w*t + phase)

    inputs.append(x)
    outputs.append(y)

# --- Page 1 (first 3 freqs) ---
fig, axs = plt.subplots(3, 2, figsize=(12, 8))
fig.suptitle("RC Low-Pass Filter Response (Page 1)", fontsize=14)

for i, f in enumerate(freqs[:3]):
    # Left = magnitude comparison
    axs[i, 0].plot(t*1000, inputs[i], 'b', label="Input")
    axs[i, 0].plot(t*1000, outputs[i], 'r', label="Output")
    axs[i, 0].set_ylabel(f"{f/1000:.1f} kHz")
    axs[i, 0].legend()
    axs[i, 0].grid(True)

    # Right = phase shift view (zoom to few cycles)
    axs[i, 1].plot(t*1000, inputs[i], 'b')
    axs[i, 1].plot(t*1000, outputs[i], 'r')
    axs[i, 1].set_xlim(0, 1.0)   # zoom in (1 ms window)
    axs[i, 1].grid(True)

axs[2, 0].set_xlabel("Time (ms)")
axs[2, 1].set_xlabel("Time (ms)")

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
# --- Page 2 (next 3 freqs) ---
fig, axs = plt.subplots(3, 2, figsize=(12, 8))
fig.suptitle("RC Low-Pass Filter Response (Page 2)", fontsize=14)

for i, f in enumerate(freqs[3:]):
  idx = i + 3
  # Left = magnitude comparison
  axs[i, 0].plot(t*1000, inputs[idx], 'b', label="Input")
  axs[i, 0].plot(t*1000, outputs[idx], 'r', label="Output")
  axs[i, 0].set_ylabel(f"{f/1000:.1f} kHz")
  axs[i, 0].legend()
  axs[i, 0].grid(True)

  # Right = phase shift view
  axs[i, 1].plot(t*1000, inputs[idx], 'b')
  axs[i, 1].plot(t*1000, outputs[idx], 'r')
  axs[i, 1].set_xlim(0, 1.0)   # zoom for clear phase lag
  axs[i, 1].grid(True)

axs[2, 0].set_xlabel("Time (ms)")
axs[2, 1].set_xlabel("Time (ms)")

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

```
======== RESTART: C:/Users/KrishnArjun/Documents/analog_assign/3ctry.py ========
Frequency (Hz)  Magnitude (dB)  Phase (deg)
1000            -0.04           -5.71
5000            -0.97           -26.57
10000           -3.01           -45.00
15000           -5.12           -56.31
20000           -6.99           -63.43
25000           -8.60           -68.20
>>
```





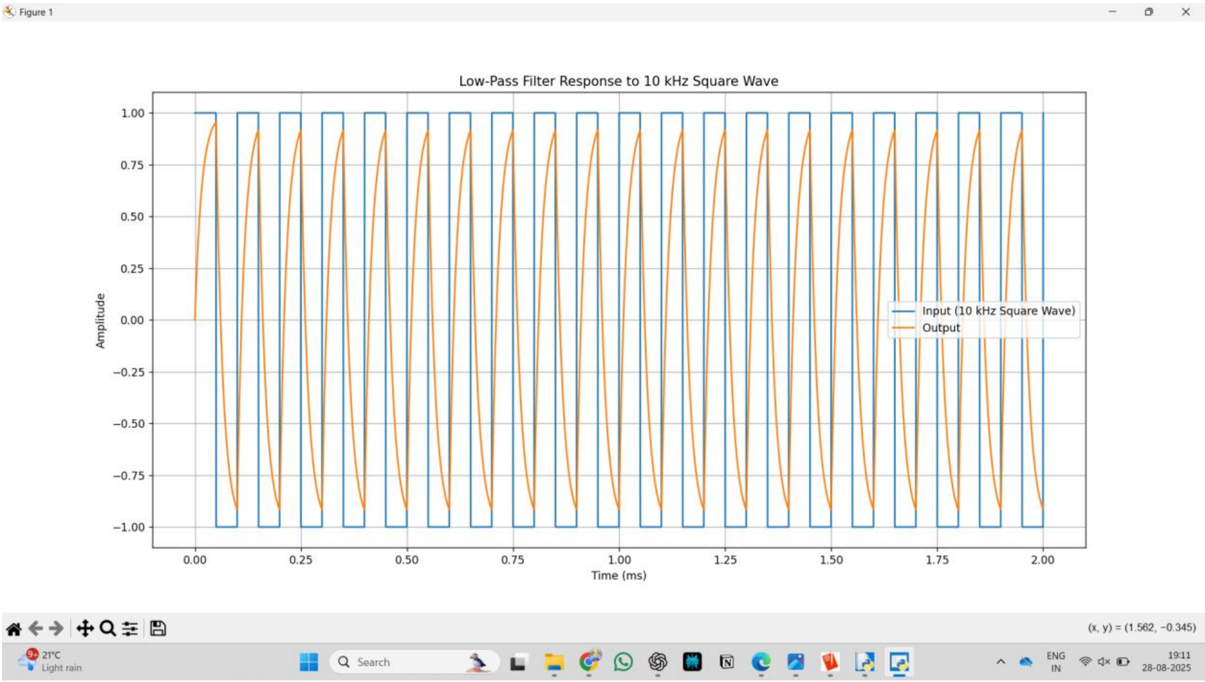
Observations :- for frequencies below 10KHZ , the output wave is almost similar to input wave in magnitude and phase , but later , attenution and phase difference happened

3.d)

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import signal


# Cutoff frequency

fc = 10000      # 10 kHz

wc = 2 * np.pi * fc


# Transfer function H(s) = wc / (s + wc)

num = [wc]

den = [1, wc]

system = signal.TransferFunction(num, den)


# Input: 10 kHz square wave

fsig = 10000     # 10 kHz

wsig = 2 * np.pi * fsig

t = np.linspace(0, 0.002, 5000)  # 2 ms duration to show a few cycles

x = signal.square(wsig * t)     # Square wave input


# Simulate response

t_out, y, _ = signal.lsim(system, U=x, T=t)


# Plot input vs output

plt.figure(figsize=(10,5))

plt.plot(t*1000, x, label="Input (10 kHz Square Wave)")

plt.plot(t*1000, y, label="Output")

plt.xlabel("Time (ms)")

plt.ylabel("Amplitude")

plt.title("Low-Pass Filter Response to 10 kHz Square Wave")

plt.legend()
```

plt.grid(True)

plt.show()



Observations :- The input Square wave is almost presented in output in one or other way of some sinusoidal signal representing it..

Q – 4)

4.a.i)

```
import numpy as np

import matplotlib.pyplot as plt


# Time axis

t = np.linspace(0, 2, 2000)   # simulate 2 seconds, 2000 samples


# Input signal x(t)

x = np.sin(2 * np.pi * 1 * t)   # 1 Hz sine wave


# Function Y

y = 2*x + (x**2)/4 + (x**3)/16


# Plot

plt.figure(figsize=(10,5))

plt.plot(t, x, label="x(t) = sin(2πt)", color='blue')

plt.plot(t, y, label="y(t)", color='red')

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.title("Simulation of Y = 2x + (x^2)/4 + (x^3)/16")

plt.legend()

plt.grid(True)

plt.show()
```

Output Y

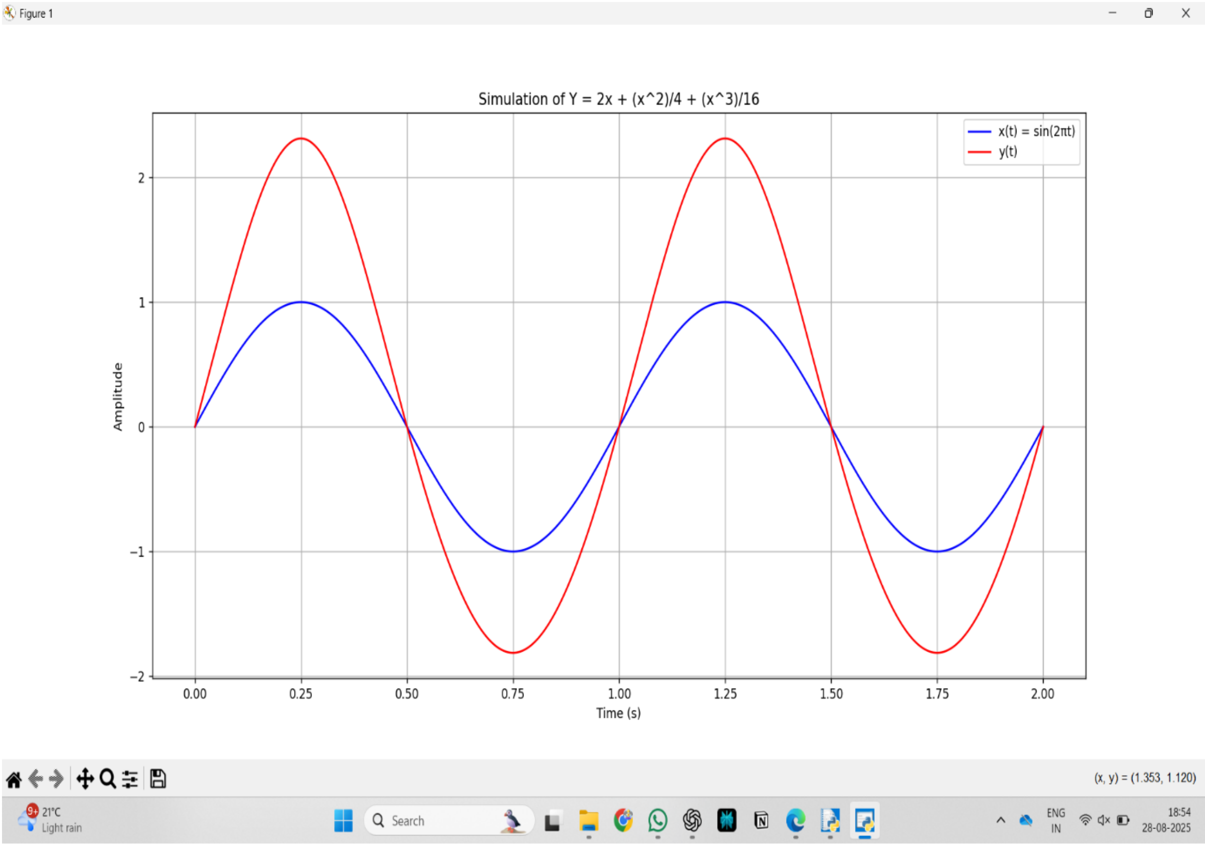


Figure 1
Simulation of Y = 2x + (x^2)/4 + (x^3)/16
x(t) = sin(2πt)
y(t)
Amplitude
Time (s)
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2
1
0
−1
−2
(x, y) = (1.353, 1.120)
21°C
Light rain
Search
ENG
IN
18:54
28-08-2025

4.a.ii) FFT

```
import numpy as np

import matplotlib.pyplot as plt


# Time settings

fs = 1000      # Sampling frequency in Hz

T = 1          # Duration in seconds

t = np.linspace(0, T, int(fs*T), endpoint=False)  # Time vector


# Input signal

x = np.sin(2 * np.pi * 1 * t)  # 1 Hz sine wave


# Output function

Y = 2*x + (x**2)/4 + (x**3)/16


# FFT

Y_fft = np.fft.fft(Y)

freq = np.fft.fftfreq(len(Y), d=1/fs)


# Take only the positive frequencies

idx = np.arange(len(freq)//2)

freq = freq[idx]

Y_fft_magnitude = np.abs(Y_fft[idx]) / len(Y)  # Normalize amplitude


# --- Plot time-domain signal ---

plt.figure(figsize=(12,5))

plt.plot(t, Y)

plt.xlabel('Time (s)')

plt.ylabel('Y(t)')

plt.title('Time-Domain Signal')
```

```python
plt.grid(True)
plt.show()

# --- Plot FFT with highlighted harmonics ---
plt.figure(figsize=(12,5))
plt.stem(freq, Y_fft_magnitude, basefmt=" ")  # Removed use_line_collection
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.title('FFT of Y(t) with Harmonics Highlighted')
plt.grid(True)

# Highlight harmonics
harmonics = [1, 2, 3, 4, 5]  # Theoretical harmonics
for h in harmonics:
    if h < fs/2:  # Only plot within Nyquist
        idx_h = np.argmin(np.abs(freq - h))
        plt.plot(h, Y_fft_magnitude[idx_h], 'ro')  # red dot
        plt.text(h, Y_fft_magnitude[idx_h]+0.01, f'{h} Hz', color='red', ha='center')

plt.show()
```

Observations :- The harmonics are distributed evenly along the x- axis

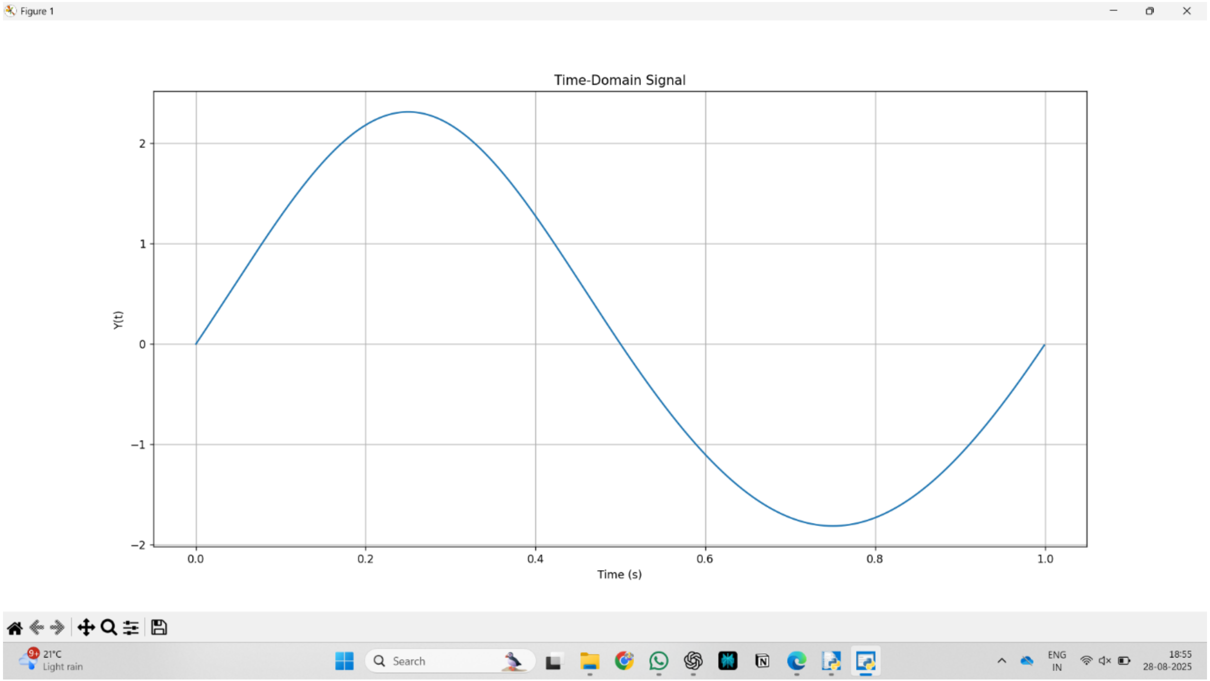


Figure 1
Time-Domain Signal
Y(t)
Time (s)
0.0
0.2
0.4
0.6
0.8
1.0
2
1
0
−1
−2
21°C
Light rain
Search
ENG
IN
18:55
28-08-2025

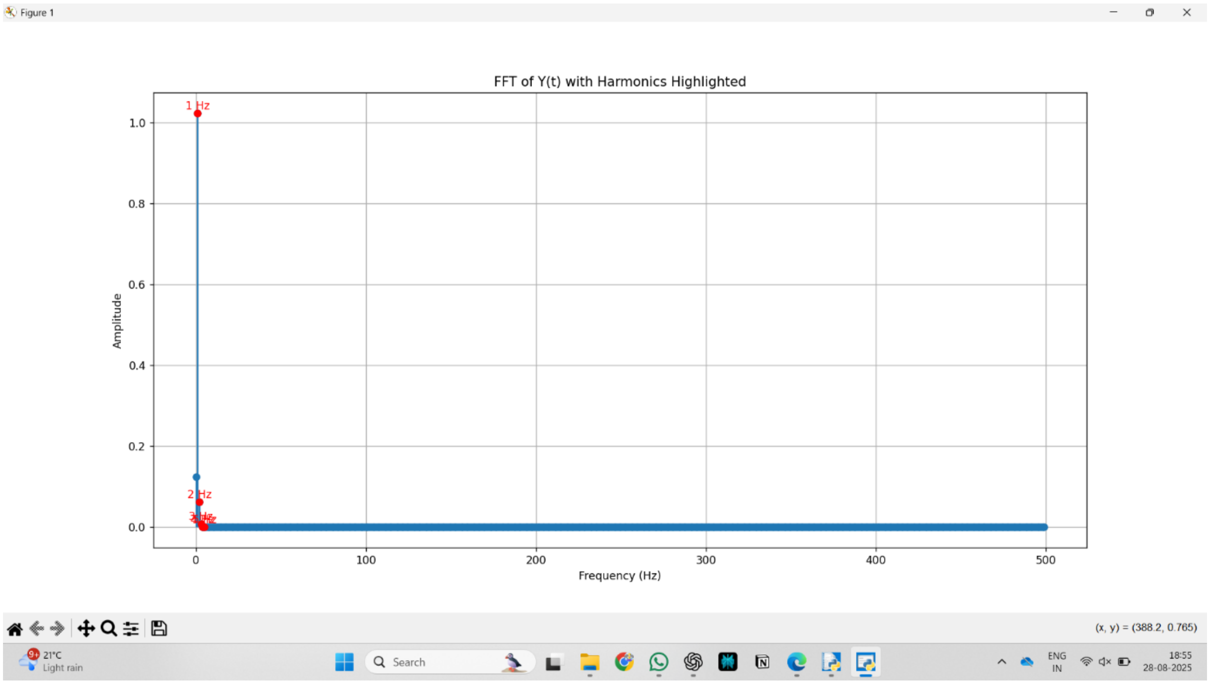


Figure 1
FFT of Y(t) with Harmonics Highlighted
1 Hz
2 Hz
Amplitude
Frequency (Hz)
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
(x, y) = (388.2, 0.765)
21°C
Light rain
Search
ENG
IN
18:55
28-08-2025

4.b.i)

```
import numpy as np

import matplotlib.pyplot as plt


# Time axis

t = np.linspace(0, 2, 2000)   # simulate 2 seconds, 2000 samples


# Input signal x(t)

x = np.sin(2 * np.pi * 1 * t)   # 1 Hz sine wave


# Function Y

y = 2*x + (x**2)/8 + (x**3)/32


# Plot

plt.figure(figsize=(10,5))

plt.plot(t, x, label="x(t) = sin(2πt)", color='blue')

plt.plot(t, y, label="y(t)", color='red')

plt.xlabel("Time (s)")

plt.ylabel("Amplitude")

plt.title("Simulation of Y = 2x + (x^2)/8 + (x^3)/32")

plt.legend()

plt.grid(True)

plt.show()
```

Output Y

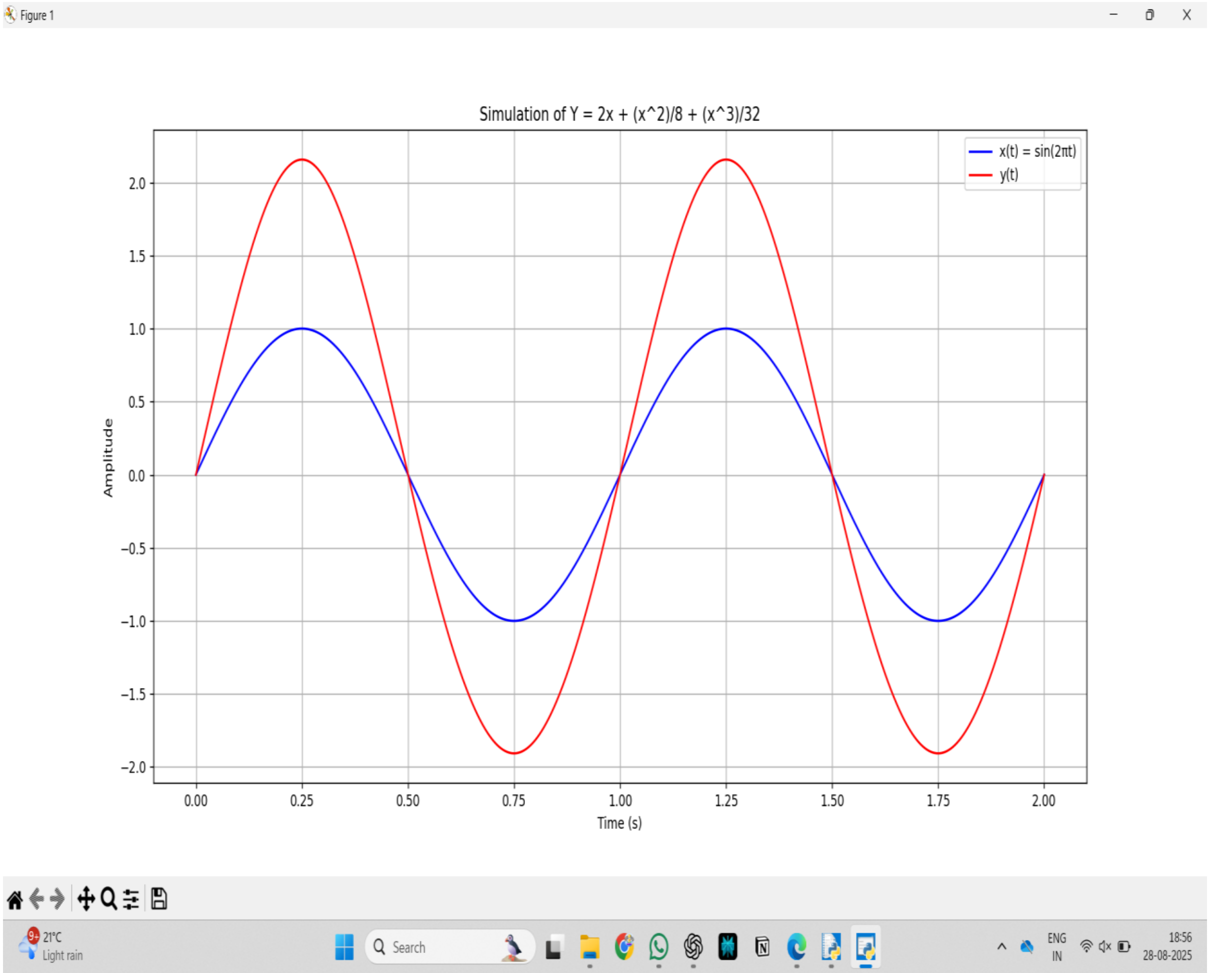


Figure 1
Simulation of Y = 2x + (x^2)/8 + (x^3)/32
x(t) = sin(2πt)
y(t)
Amplitude
Time (s)
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
21°C
Light rain
Search
ENG
IN
18:56
28-08-2025

4.b.ii) FFT

```
import numpy as np

import matplotlib.pyplot as plt


# Time settings

fs = 1000      # Sampling frequency in Hz

T = 1          # Duration in seconds

t = np.linspace(0, T, int(fs*T), endpoint=False)  # Time vector


# Input signal

x = np.sin(2 * np.pi * 1 * t)  # 1 Hz sine wave


# Updated Output function

Y = 2*x + (x**2)/8 + (x**3)/32


# --- Plot time-domain signal ---

plt.figure(figsize=(12,5))

plt.plot(t, Y)

plt.xlabel('Time (s)')

plt.ylabel('Y(t)')

plt.title('Time-Domain Signal of Updated Y(t)')

plt.grid(True)

plt.show()


# FFT

Y_fft = np.fft.fft(Y)

freq = np.fft.fftfreq(len(Y), d=1/fs)
```

```
# Take only the positive frequencies
idx = np.arange(len(freq)//2)
freq = freq[idx]
Y_fft_magnitude = np.abs(Y_fft[idx]) / len(Y)  # Normalize amplitude

# --- Plot FFT with highlighted harmonics ---
plt.figure(figsize=(12,5))
plt.stem(freq, Y_fft_magnitude, basefmt=" ")  # Removed use_line_collection
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.title('FFT of Updated Y(t) with Harmonics Highlighted')
plt.grid(True)

# Highlight harmonics
harmonics = [1, 2, 3, 4, 5]  # Expected harmonics from nonlinear terms
for h in harmonics:
    if h < fs/2:  # Only plot within Nyquist
        idx_h = np.argmin(np.abs(freq - h))
        plt.plot(h, Y_fft_magnitude[idx_h], 'ro')  # red dot
        plt.text(h, Y_fft_magnitude[idx_h]+0.01, f'{h} Hz', color='red', ha='center')

plt.show()
```

Observations :- The harmonics are distributed evenly along the x- axis

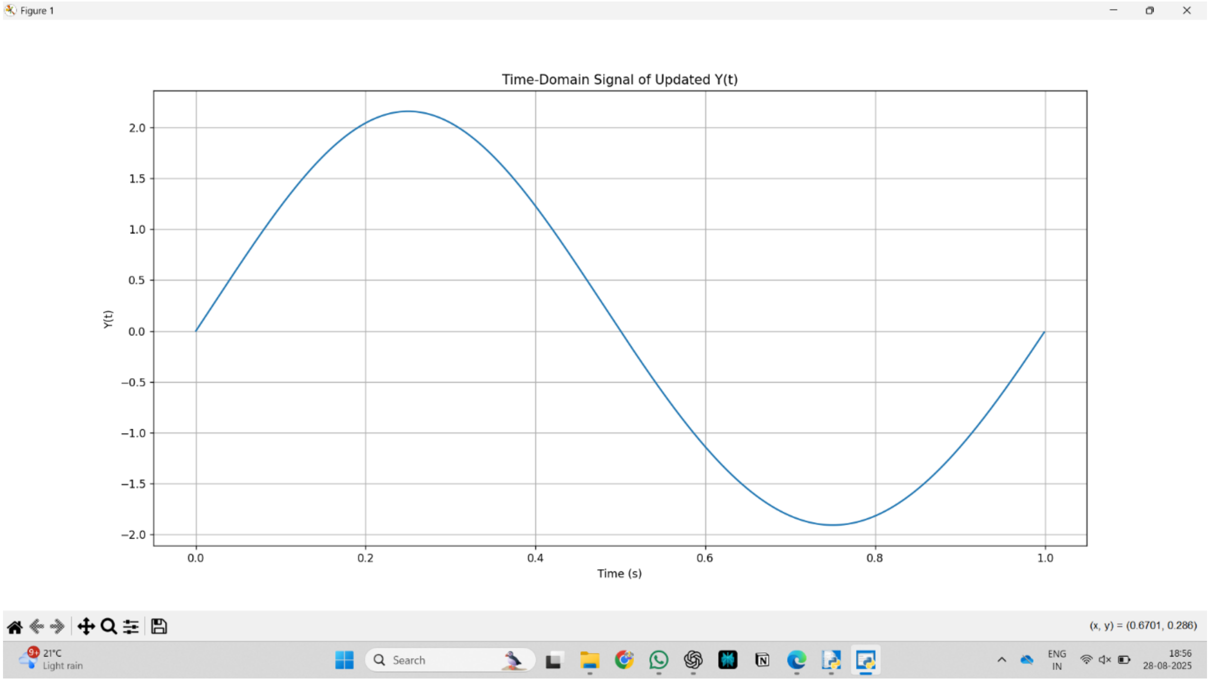


Figure 1
Time-Domain Signal of Updated Y(t)
Y(t)
Time (s)
(x, y) = (0.6701, 0.286)

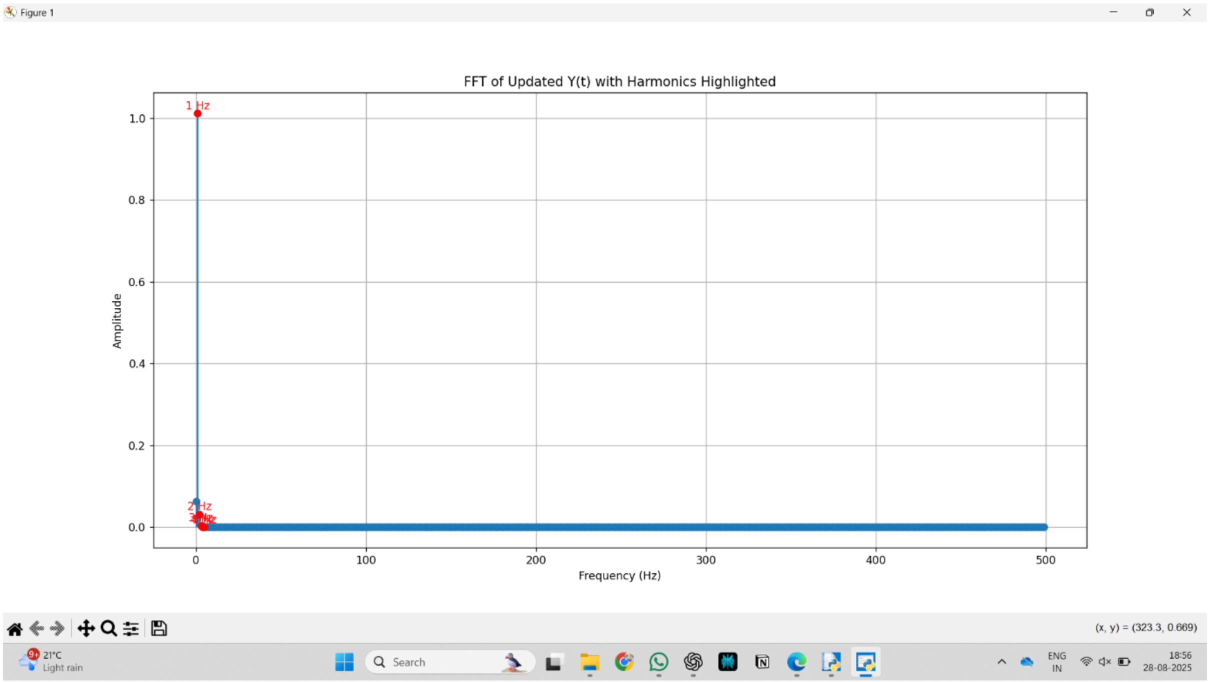


Figure 1
FFT of Updated Y(t) with Harmonics Highlighted
1 Hz
2 Hz
Amplitude
Frequency (Hz)
(x, y) = (323.3, 0.669)

Q – 5)

5 Cutoff frequency $f_c = 10\ kHz$.

$$f_c = \frac{1}{2\pi RC}$$

Lets take $C = 1nF = 1 \times 10^{-9} F$.

$$R = \frac{1}{2 \times \pi \times f_c \times C}$$

$$= \underline{\underline{15.9\ k\Omega}}$$

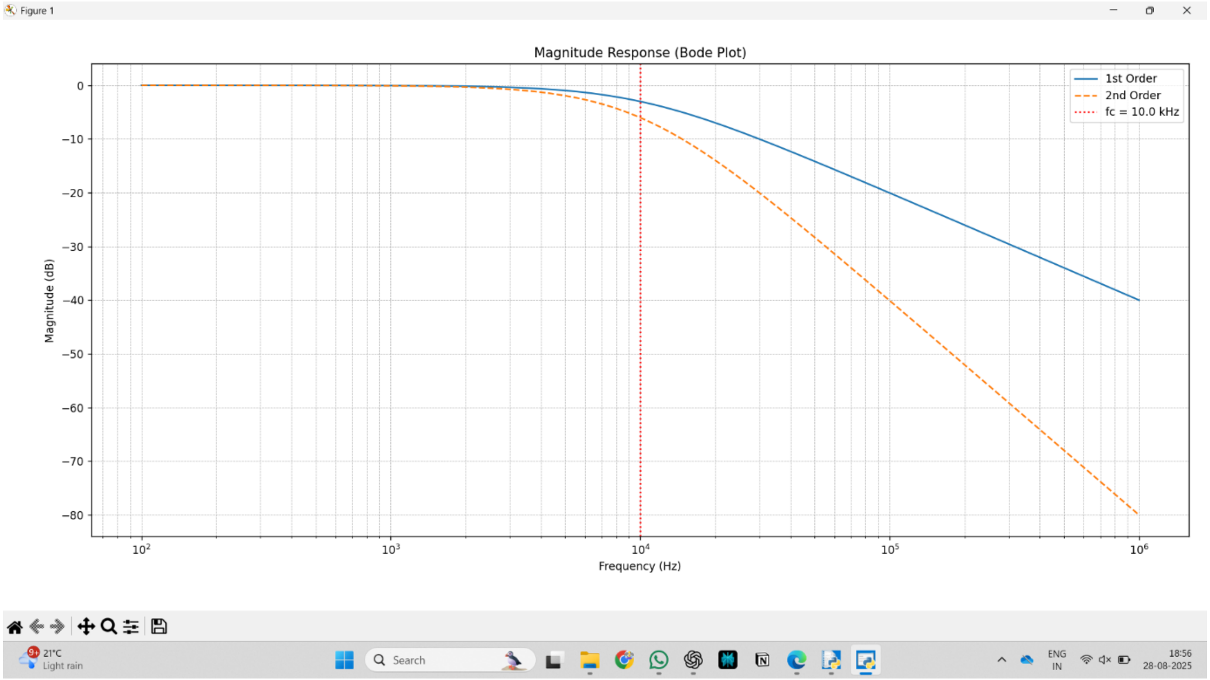


Figure 1
Magnitude Response (Bode Plot)
Magnitude (dB)
Frequency (Hz)
1st Order
2nd Order
fc = 10.0 kHz
0
−10
−20
−30
−40
−50
−60
−70
−80
10^2
10^3
10^4
10^5
10^6
21°C
Light rain
Search
ENG
IN
18:56
28-08-2025

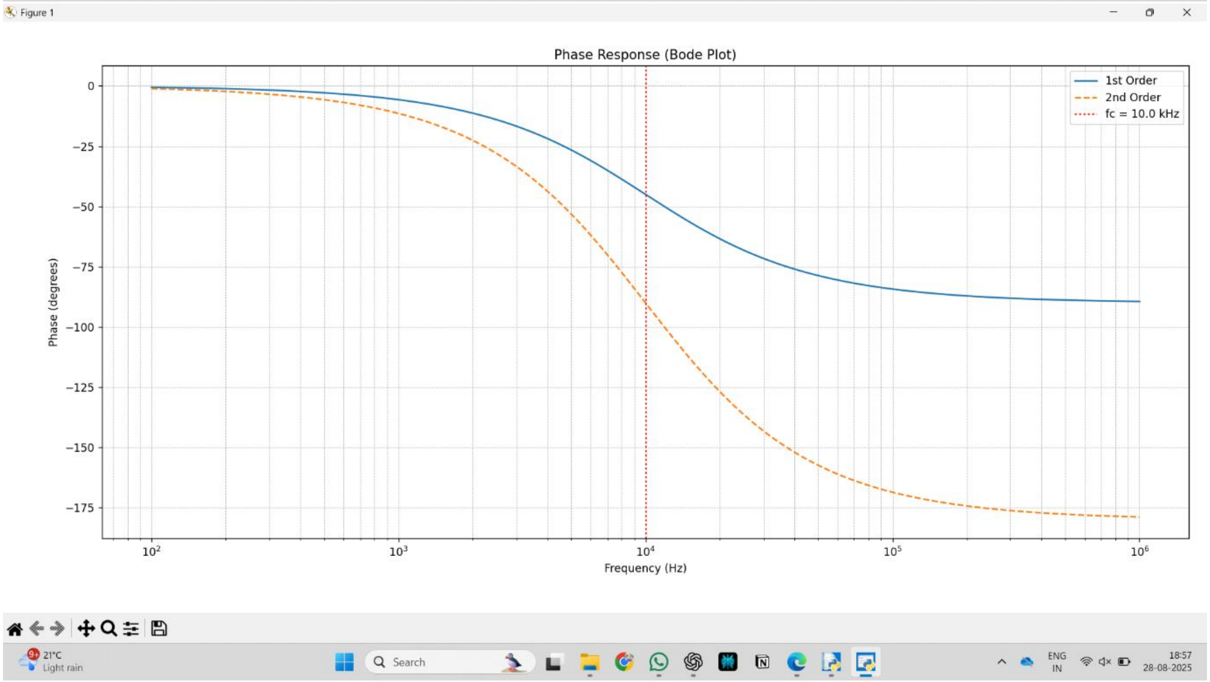


Figure 1
Phase Response (Bode Plot)
Phase (degrees)
Frequency (Hz)
1st Order
2nd Order
fc = 10.0 kHz
0
−25
−50
−75
−100
−125
−150
−175
10^2
10^3
10^4
10^5
10^6
21°C
Light rain
Search
ENG
IN
18:57
28-08-2025