# Lab Assignment - 1

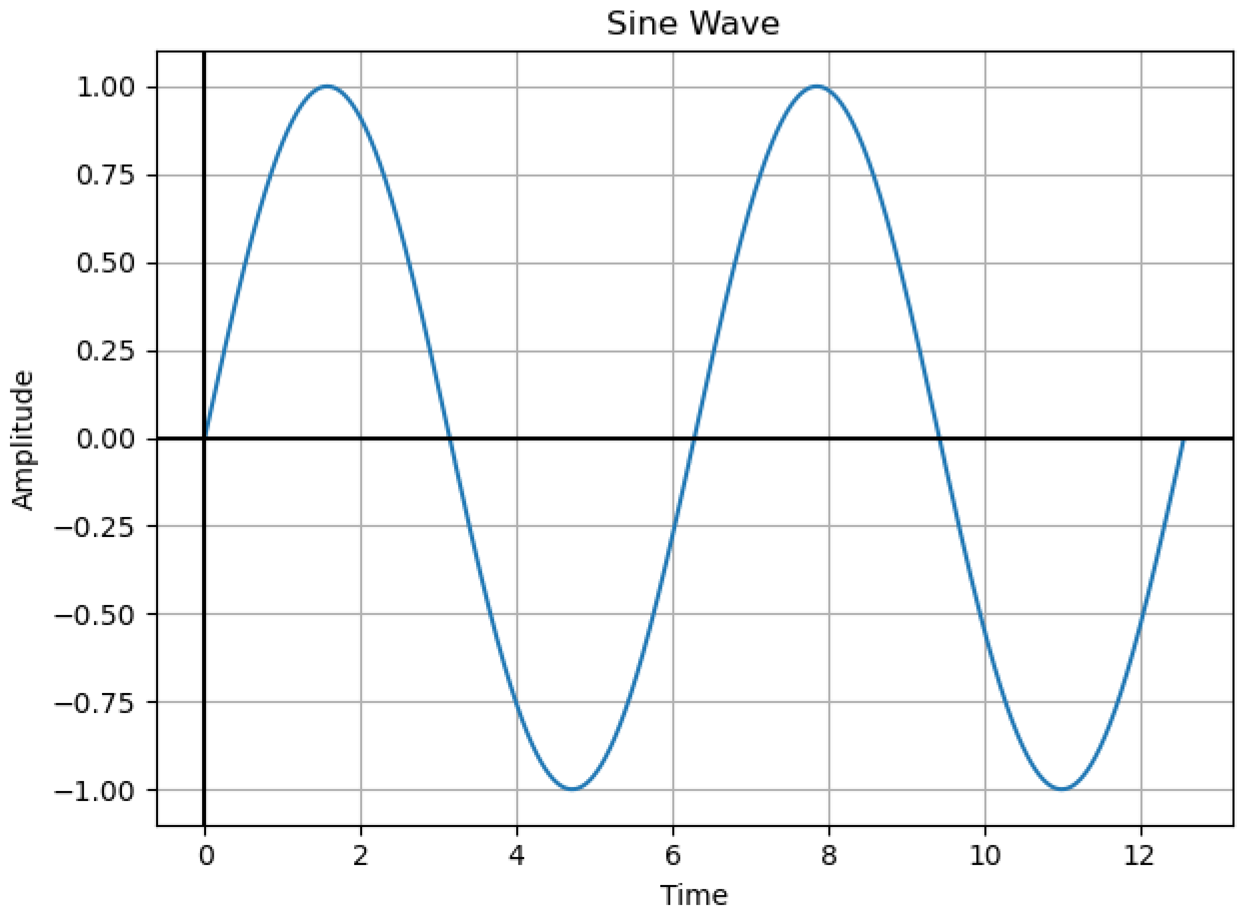
# Nithin S

# 221IT085

# Experiment 1 : Introduction to Signal Types

## a) Continuous Time Sinosodial Signal

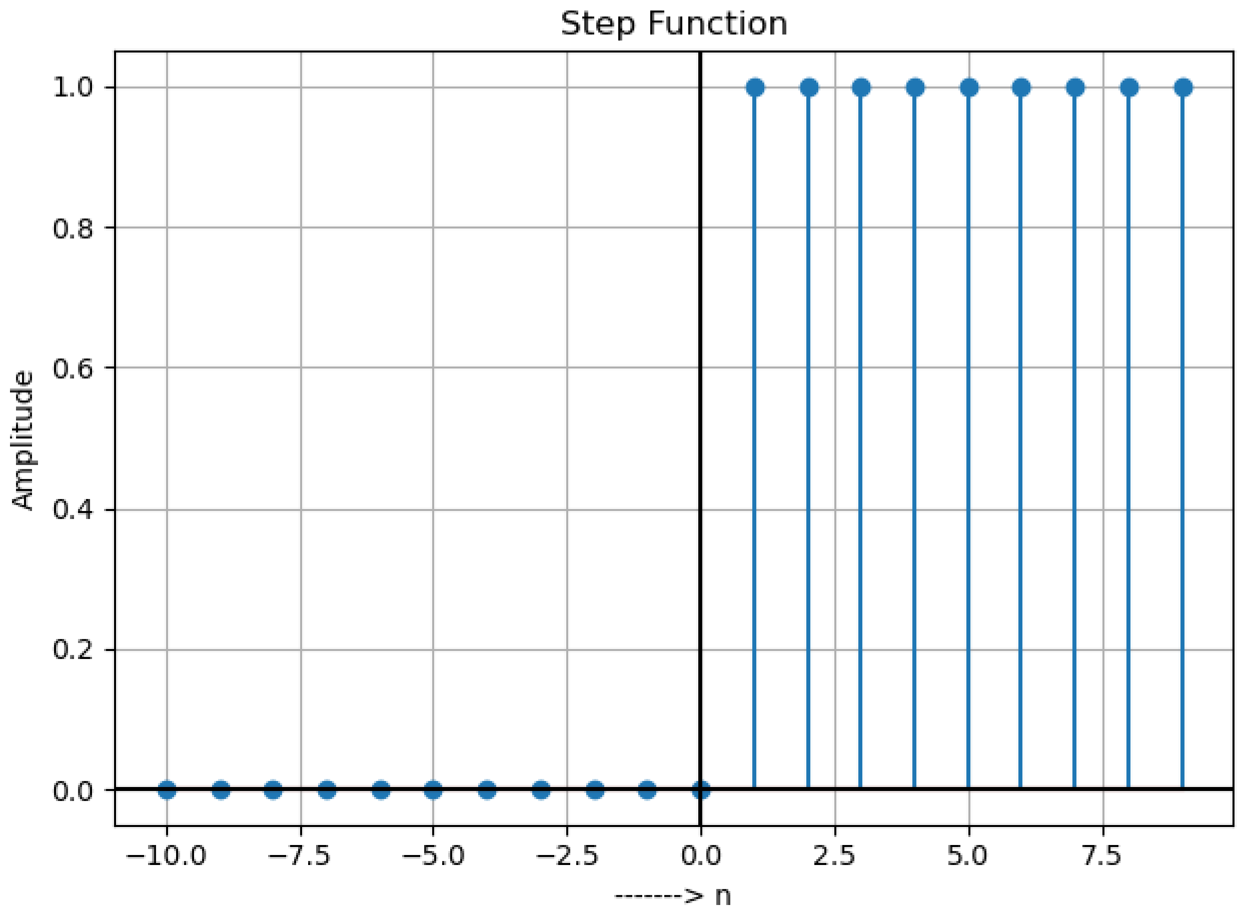
In [134…
```
import numpy as np
import matplotlib.pyplot as plt

t=np.arange(0,4*np.pi,0.01)
xt=np.sin(t)
plt.plot(t,xt)
plt.title("Sine Wave")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.tight_layout()
plt.show()
```




## b) Discrete-Time Unit Step Signal

In [135…

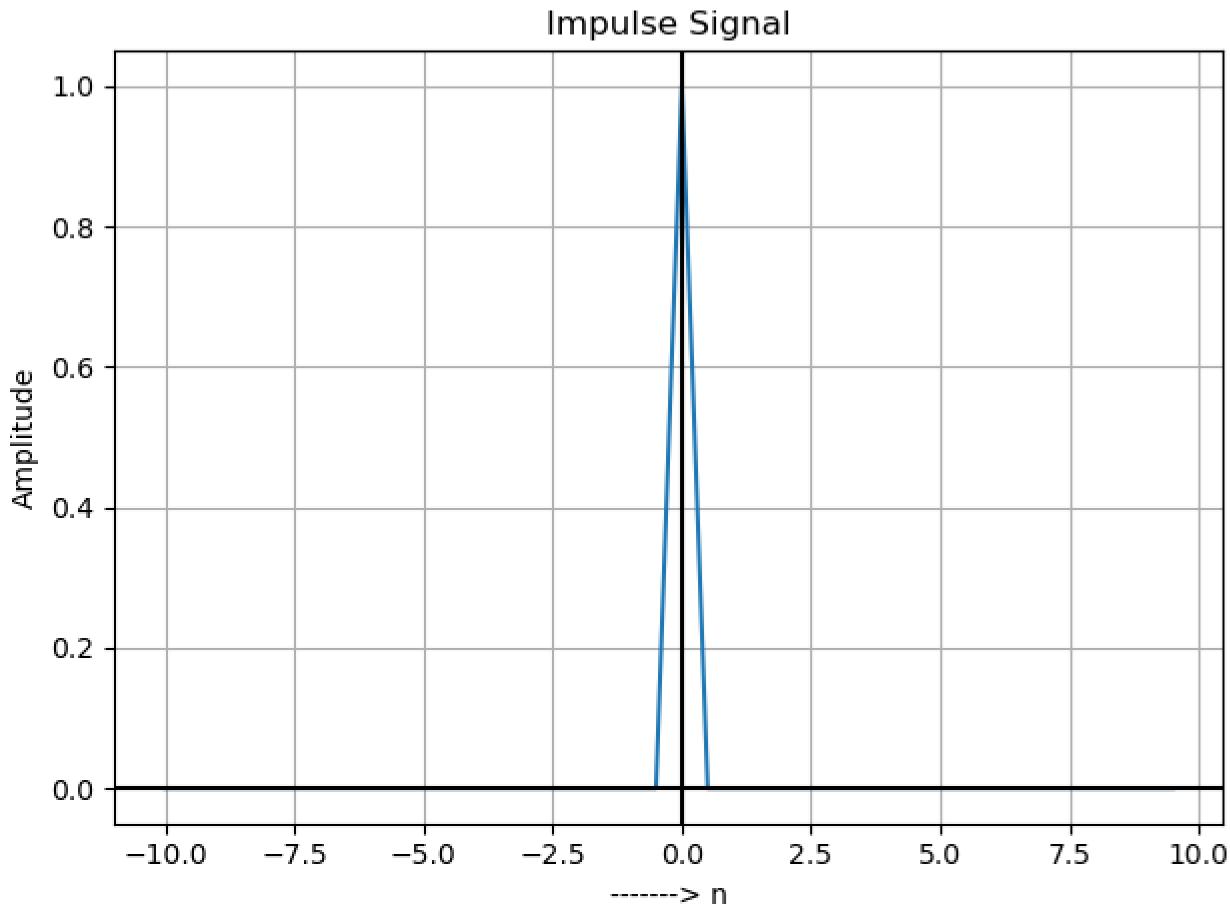
```
import numpy as np
import matplotlib.pyplot as plt

n=[x for x in range(-10,10)]
un=[1 if x>0 else 0 for x in n]
plt.stem(n,un)
plt.title("Step Function")
plt.xlabel("-------> n")
plt.ylabel("Amplitude")
plt.axhline(y=0,color='k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.tight_layout()
plt.show()
```




## c) Continuous Time Impulse Signal

In [136…

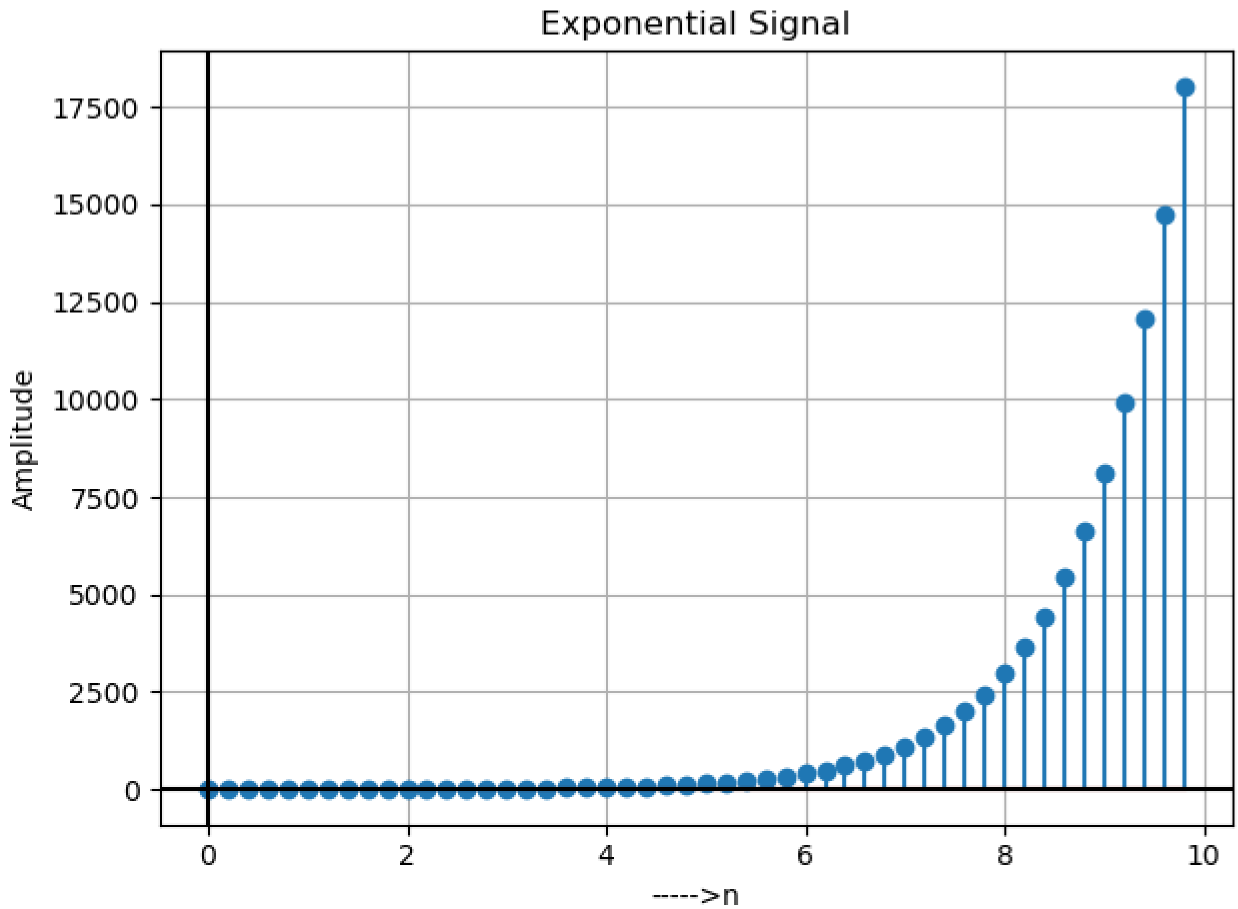
```
import numpy as np
import matplotlib.pyplot as plt

n=np.arange(-10,10,0.5)
delta=[1 if x==0 else 0 for x in n]
plt.plot(n,delta)
plt.axhline(y=0,color='k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.title("Impulse Signal")
plt.xlabel("-------> n")
plt.ylabel("Amplitude")
plt.tight_layout()
plt.show()
```




## d) Discrete Time Exponential Signal

In [137…

```
n=np.arange(0,10,0.2)
xn1=np.exp(n)

plt.stem(n,xn1)
plt.axhline(y=0,color='k')
plt.axvline(x=0,color = 'k')
plt.title("Exponential Signal")
plt.xlabel("----->n")
plt.ylabel("Amplitude")
plt.grid()
plt.tight_layout()
plt.show()
```



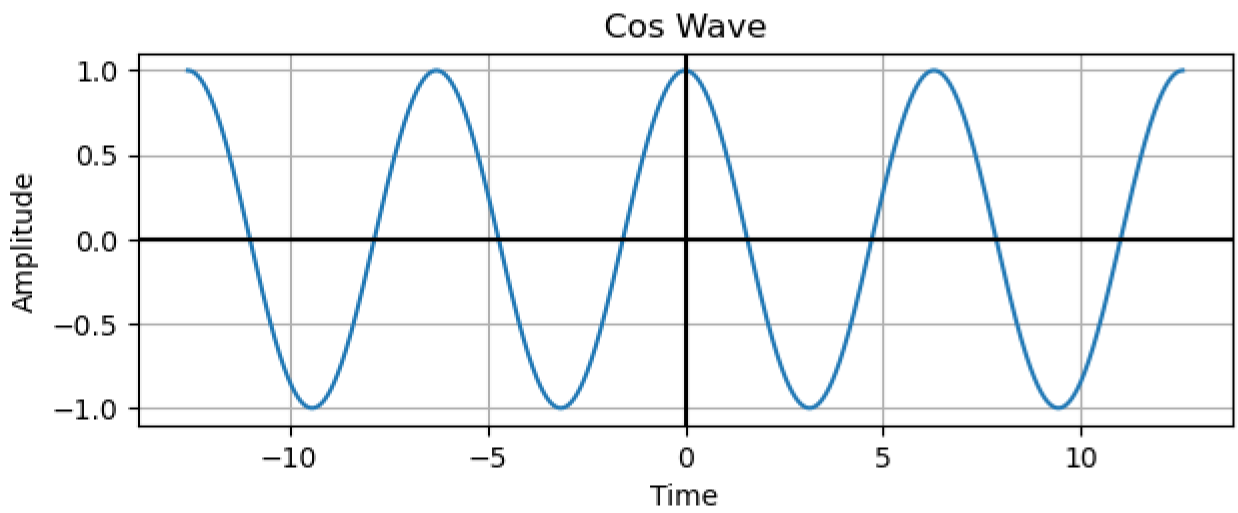

# Experiment 2: Signal Transformation

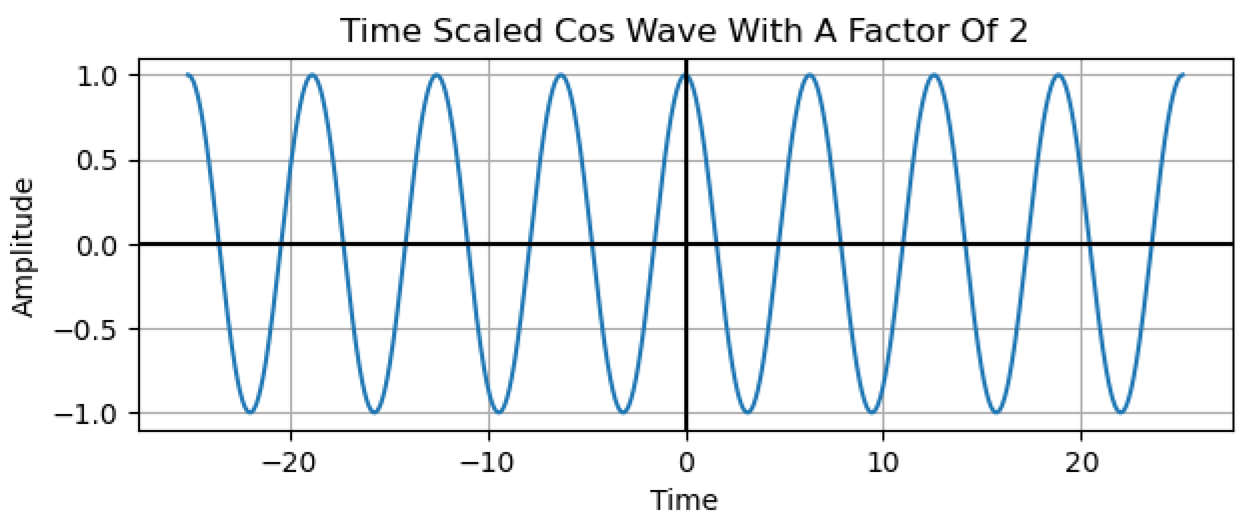
## a) Time Scaling

In [138…

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

t=np.arange(-4*np.pi,4*np.pi,0.01)
xt=np.cos(t)

plt.subplot(2,1,1)
plt.plot(t,xt)
plt.title("Cos Wave")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.tight_layout()
plt.show()

plt.subplot(2,1,2)
scaling_factor=2
t_scaled=scaling_factor*t
xt=np.cos(t_scaled)
plt.plot(t_scaled,xt)
plt.title(f"Time Scaled Cos Wave With A Factor Of {scaling_factor}")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')
```

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



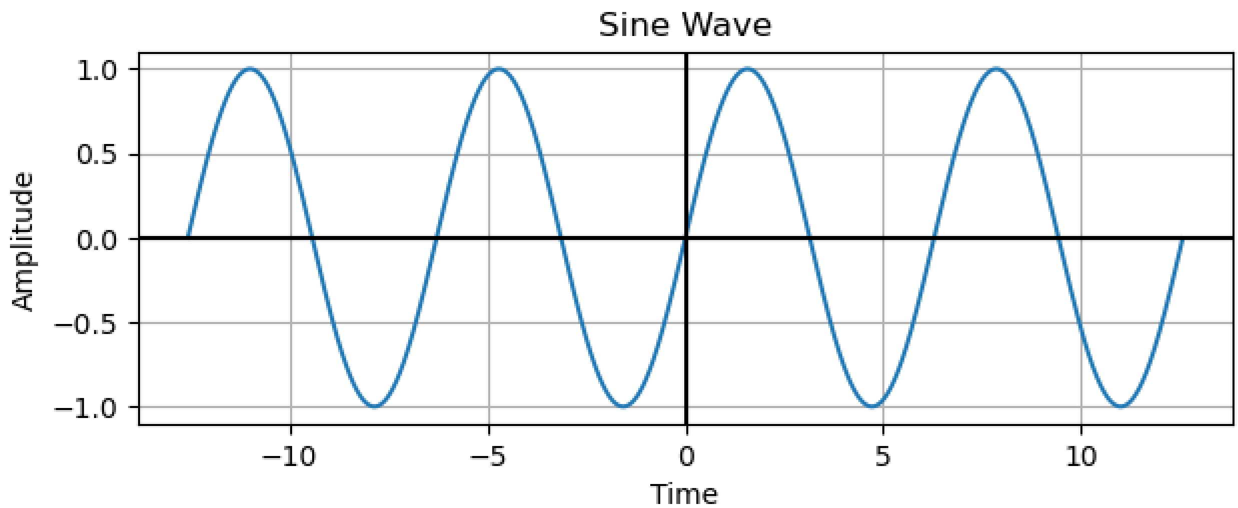



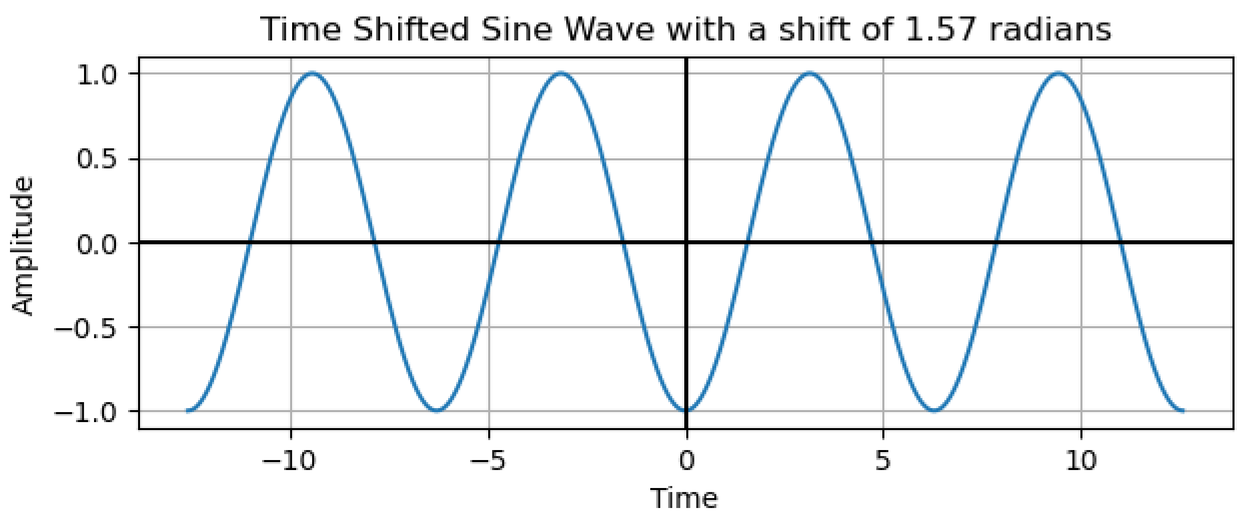

## b) Time Shifting

In [139…

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

t=np.arange(-4*np.pi,4*np.pi,0.01)
xt=np.sin(t)

plt.subplot(2,1,1)
plt.plot(t,xt)
plt.title("Sine Wave")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.tight_layout()
plt.show()

plt.subplot(2,1,2)
t_shift=(np.pi)/2    #The Time Shift
xt=np.sin(t-t_shift)
plt.plot(t,xt)
plt.title(f"Time Shifted Sine Wave with a shift of {round(t_shift,2)} radians")
plt.xlabel("Time")
```

```
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')

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

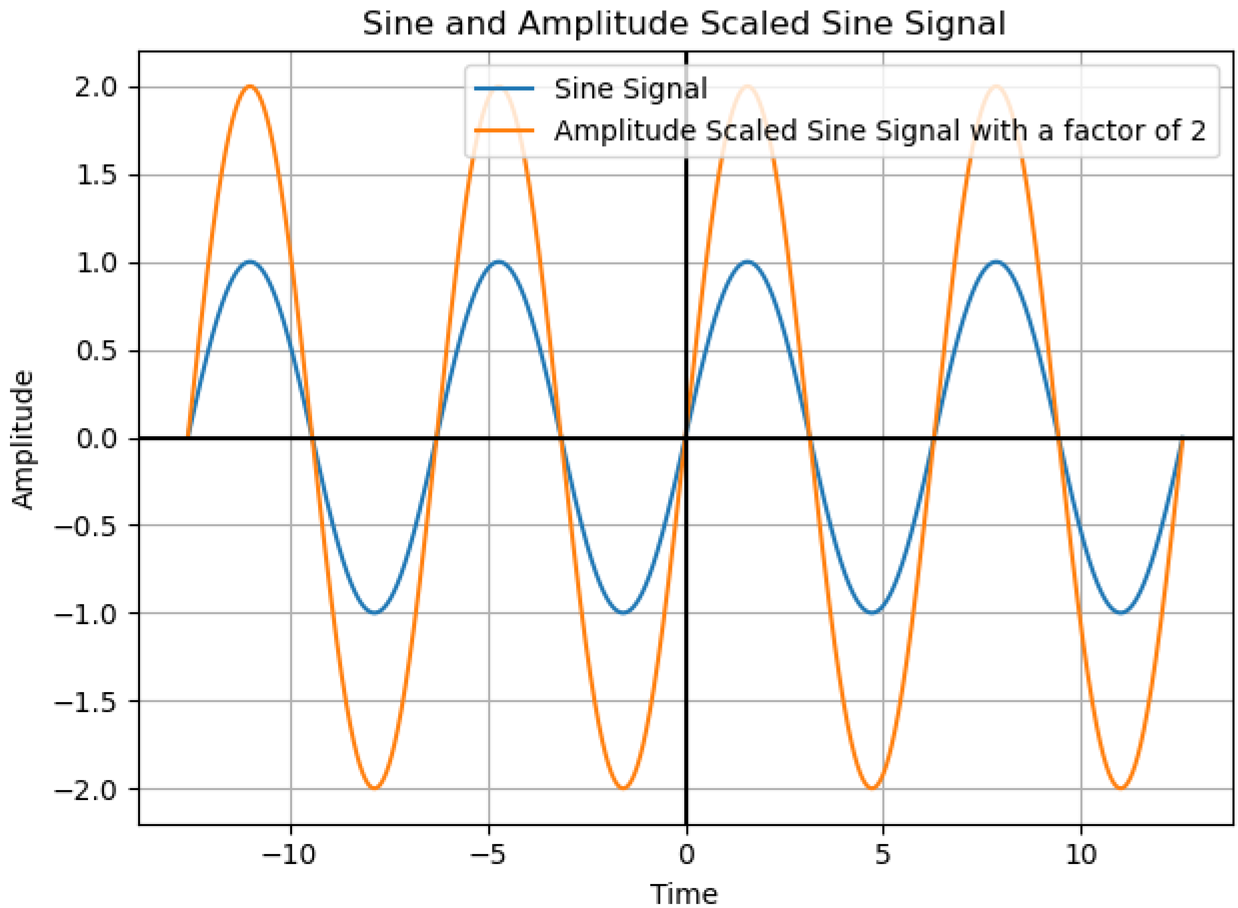



## c) Amplitude Scaling

In [140…

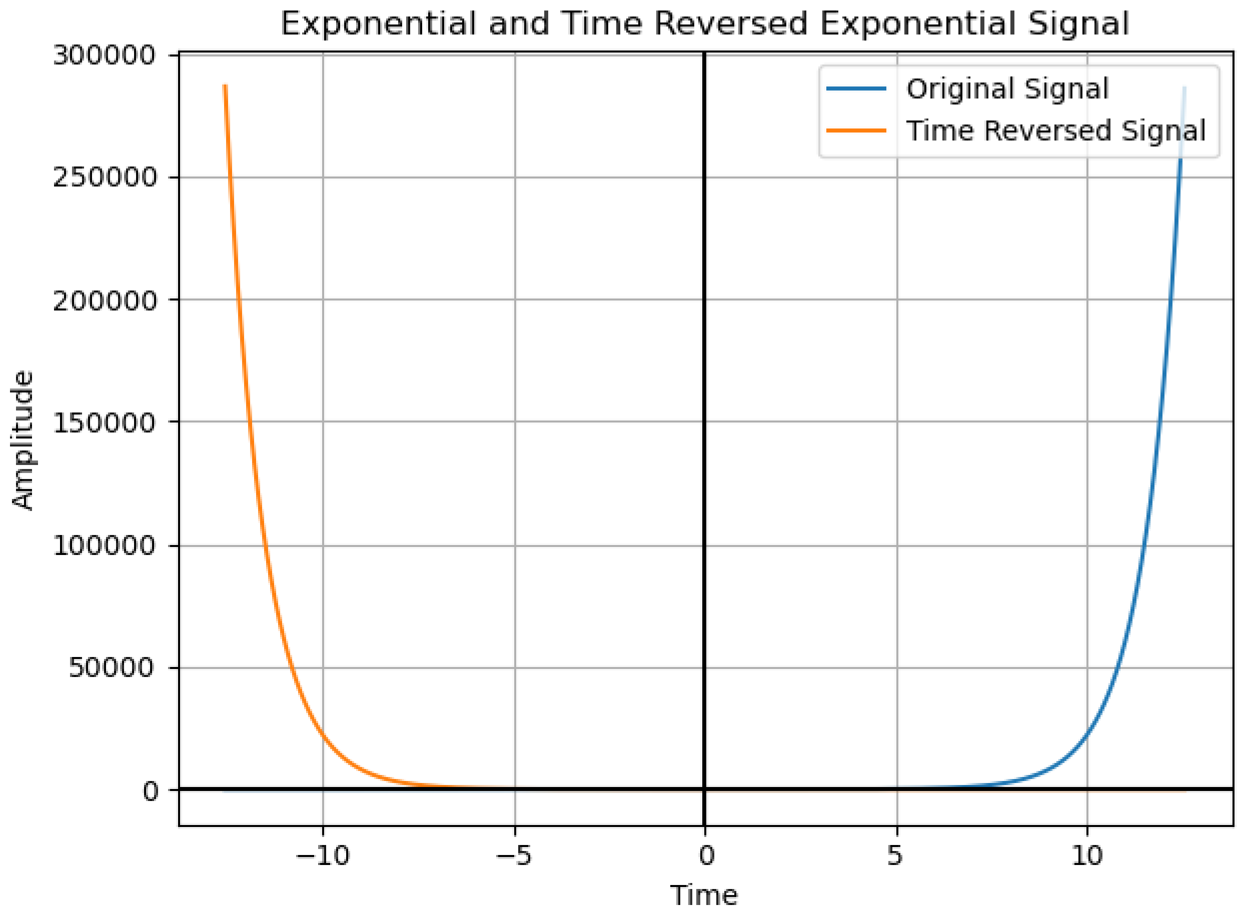
```
import numpy as np
import matplotlib.pyplot as plt

t=np.arange(-4*np.pi,4*np.pi,0.01)
xt=np.sin(t)
scaling_factor=2
xt_scaled=np.sin(t)*scaling_factor

plt.plot(t,xt,label="Sine Signal")
plt.plot(t,xt_scaled,label=f"Amplitude Scaled Sine Signal with a factor of {scaling_fact
plt.title("Sine and Amplitude Scaled Sine Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0,color = 'k')
plt.axvline(x=0,color = 'k')
plt.grid()
plt.legend()
plt.tight_layout()
plt.show()
```




## d) Time Reversal

In [141…

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

t=np.arange(-4*np.pi,4*np.pi,0.01)
xt=np.exp(t)
xt_reversed=np.exp(-t)

plt.plot(t, xt, label="Original Signal")
plt.plot(t, xt_reversed, label="Time Reversed Signal")
plt.title("Exponential and Time Reversed Exponential Signal")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.axhline(y=0, color='k')
plt.axvline(x=0, color='k')
plt.grid()
plt.legend(loc="upper right")

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

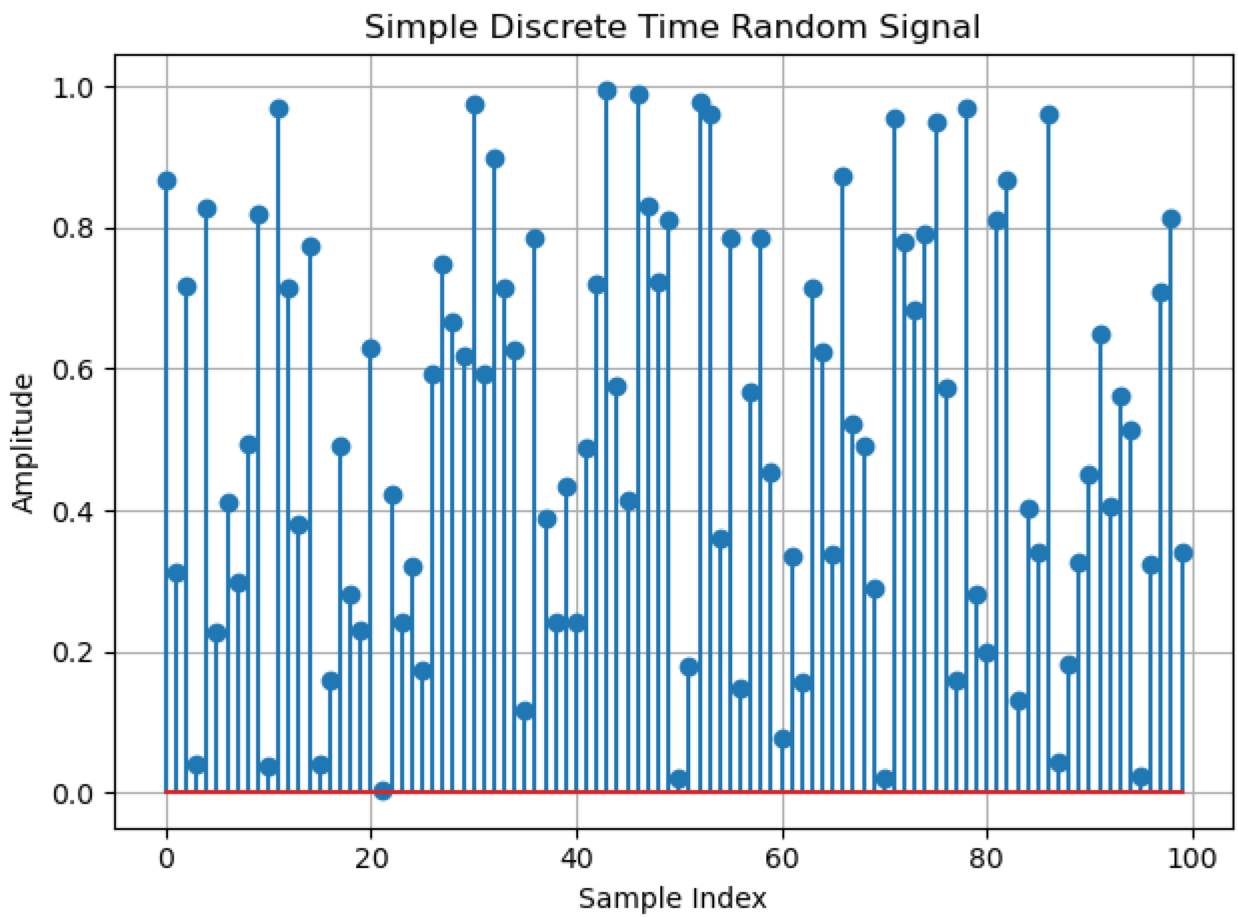



# Classification of Signals

## a) Periodic & Aperiodic Signals

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np

# Number of samples in the signal
num_samples = 100
# Generate random values for the signal
random_signal = np.random.rand(num_samples)  # Using random values between 0 and 1

plt.stem(random_signal)
plt.xlabel('Sample Index')
plt.ylabel('Amplitude')
plt.title('Simple Discrete Time Random Signal')
plt.grid(True)
plt.tight_layout()
plt.show()

# Analyze the signal for periodicity
is_periodic = False
period = None

#np.allclose(a, b) is a function in NumPy that checks if all elements of two arrays are
#np.roll(a, shift) is a function in NumPy that circularly shifts the elements of an arra
for shift in range(1, num_samples):
    if np.allclose(random_signal, np.roll(random_signal, shift)):
        is_periodic = True
        period = shift
        break
```

```
if is_periodic:
    print(f"The signal is periodic with a period of {period} samples.Since the Signal re

else:
    print("The signal is aperiodic.Signal with no repeating pattern at regular intervals
```



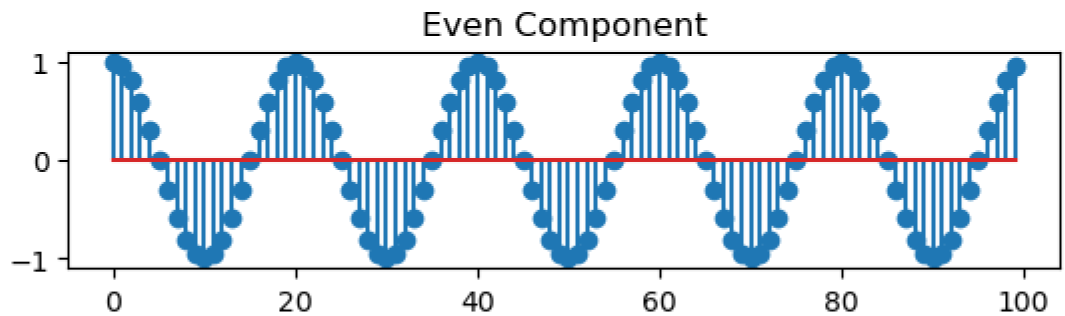

```
The signal is aperiodic.Signal with no repeating pattern at regular intervals in its beh
avior.
```

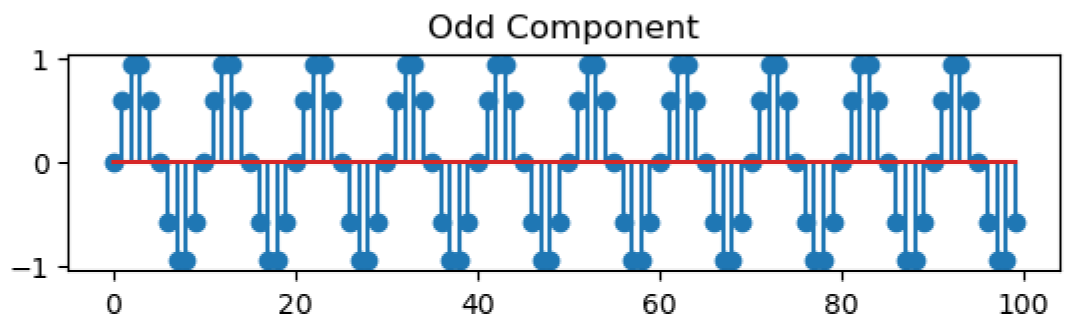
## b) Even and Odd Signal

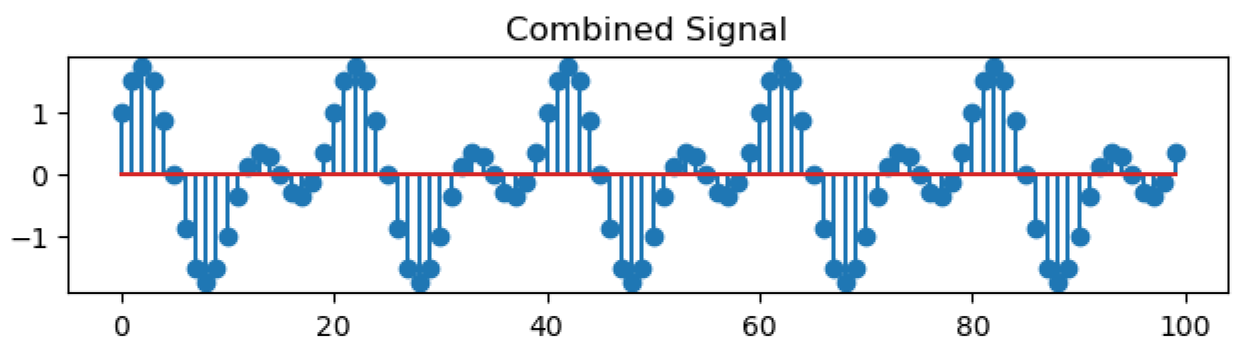
In [3]:

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

num_samples = 100
n = np.arange(num_samples)

even_component = np.cos(0.1 * np.pi * n)  # Even component Considered
odd_component = np.sin(0.2 * np.pi * n)  # Odd component Considered

# Combine the components
combined_signal = even_component + odd_component

# Plot the original components and the combined signal

plt.subplot(3,1,1)
plt.stem(n, even_component)
plt.title('Even Component')
plt.show()

plt.subplot(3,1,2)
plt.stem(n, odd_component)
plt.title('Odd Component')
plt.show()
```

```python
plt.subplot(3,1,3)
plt.stem(n, combined_signal)
plt.title('Combined Signal')

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

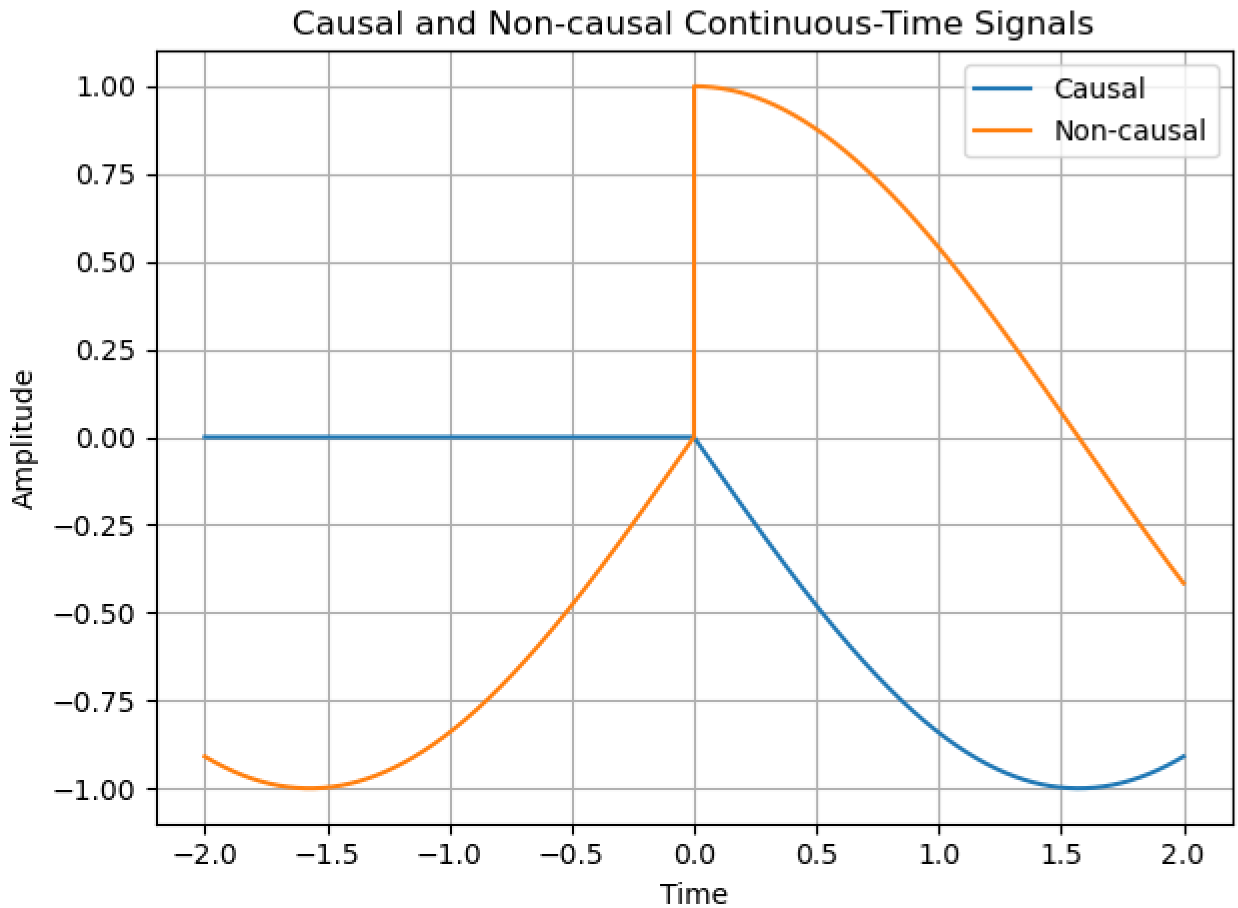



## c) Causal and Non-Causal Signal

In [4]:
```python
import numpy as np
import matplotlib.pyplot as plt

t_continuous = np.arange(-2, 2.001, 0.001)  # Time values from -2 to 2 with step of 0.00

x_causal = np.where(t_continuous >= 0, np.sin(-t_continuous), 0)
x_noncausal = np.where(t_continuous >= 0, np.cos(t_continuous), np.sin(t_continuous))

plt.plot(t_continuous, x_causal, label='Causal')
plt.plot(t_continuous, x_noncausal, label='Non-causal')
plt.title('Causal and Non-causal Continuous-Time Signals')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()
plt.grid()
plt.tight_layout()
plt.show()
```


Causal and Non-causal Continuous-Time Signals
Causal
Non-causal
Amplitude
Time
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0