| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

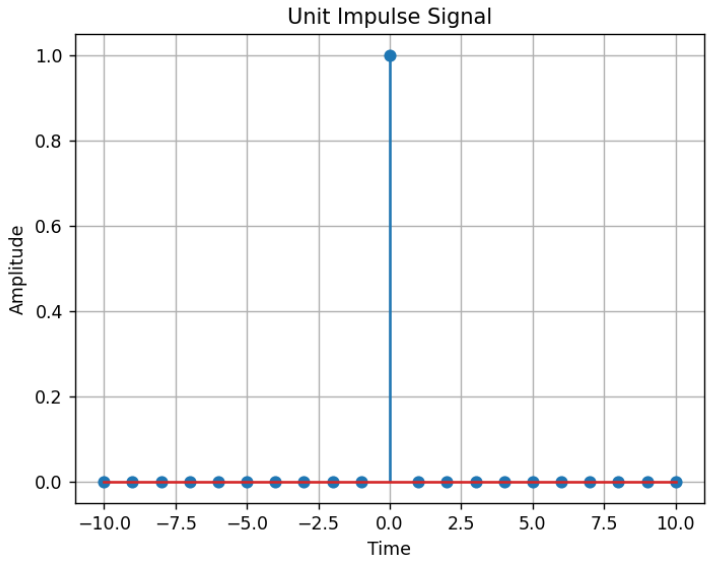
**AIM:** To simulate and analyze fundamental signals such as unit impulse, impulse train, unit step, and ramp signals in both continuous and discrete time domains, and to understand their properties, behavior, and importance in digital signal processing.

1. **Unit impulse signal:-**

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

def unit_impulse(length, position):
    signal = np.zeros(length)
    signal[position] = 1
    return signal

# Parameters
start = -10  # Start value of the x-axis range
stop = 10  # Stop value of the x-axis range
step = 1  # Step size

# Generate x-axis values
x = np.arange(start, stop+step, step)

# Generate unit impulse signal
impulse_signal = unit_impulse(len(x), abs(start)//step)

# Plot the signal
plt.stem(x, impulse_signal)
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Unit Impulse Signal')
plt.grid(True)
plt.show()
```

**Output:-**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |



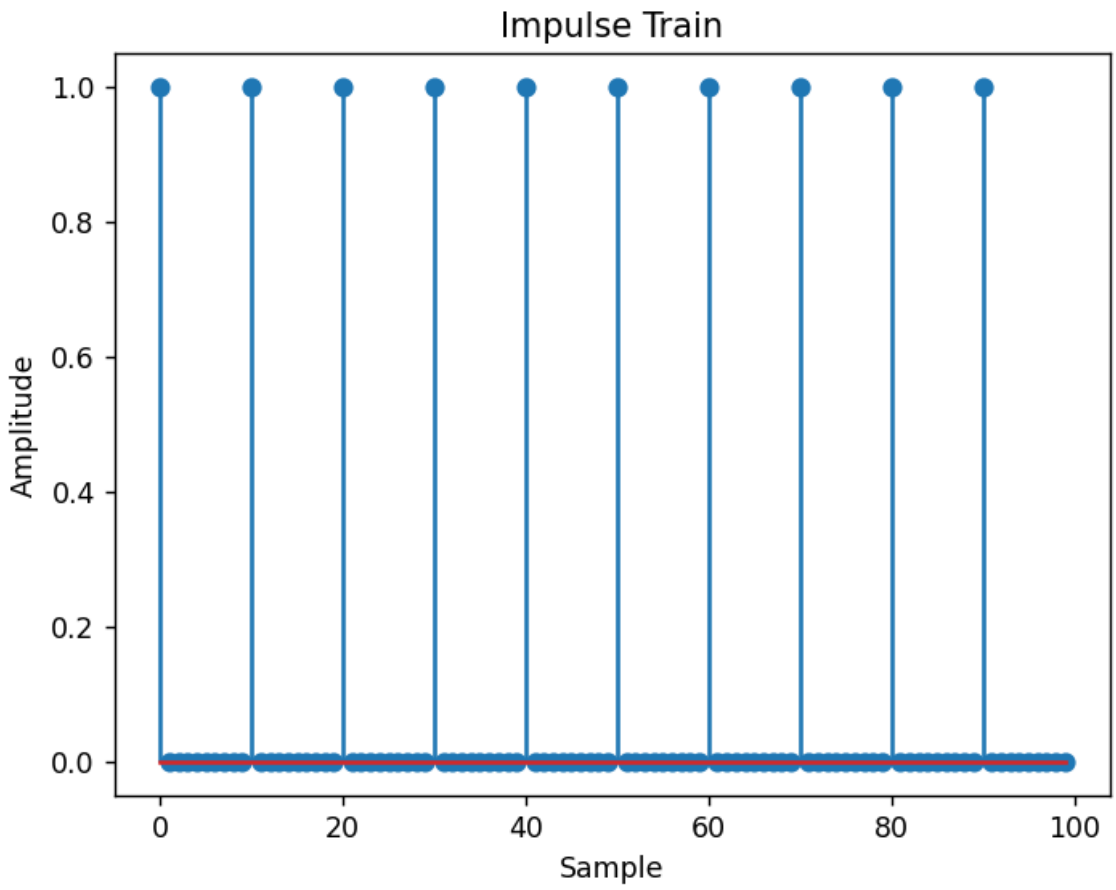

## 2. Impulse train signal:-

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

def simulate_impulse_train(signal_length, period):
    impulse_train = np.zeros(signal_length)
    for n in range(signal_length):
        if n % period == 0:
            impulse_train[n] = 1
    return impulse_train

# Define the parameters for the impulse train
signal_length = 100  # Length of the impulse train
period = 10  # Period of the impulse train

# Simulate the impulse train
impulse_train = simulate_impulse_train(signal_length, period)
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
# Plot and display the impulse train
plt.stem(impulse_train)
plt.title('Impulse Train')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.show()

# Save the impulse train array (optional)
# np.savetxt('impulse_train.txt', impulse_train, delimiter=',')
```

**Output:-**




3. **Continuous and Discrete unit step signal's:-**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group<br>NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

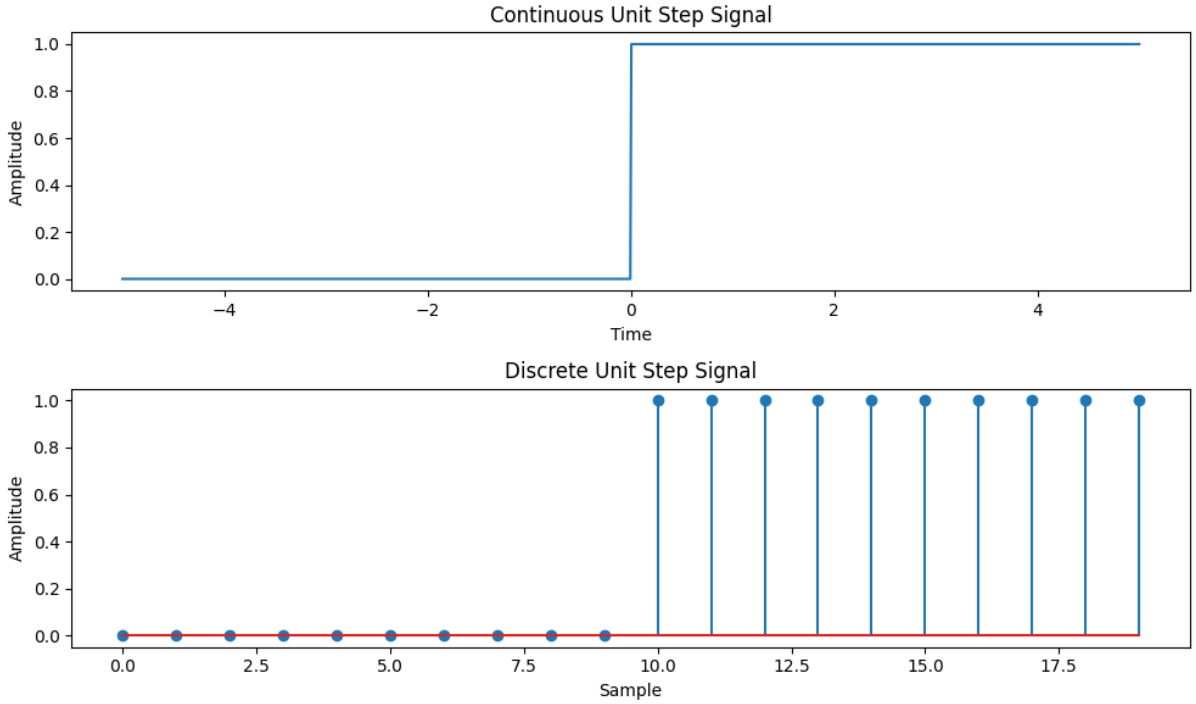
```
import numpy as np
import matplotlib.pyplot as plt

def simulate_continuous_unit_step(time):
    unit_step = np.zeros_like(time)
    unit_step[time >= 0] = 1
    return unit_step

def simulate_discrete_unit_step(num_samples):
    unit_step = np.zeros(num_samples)
    unit_step[num_samples // 2:] = 1
    return unit_step

# Define the time range for the continuous unit step signal
time = np.linspace(-5, 5, 1000)  # Time range from -5 to 5

# Simulate the continuous unit step signal
continuous_unit_step = simulate_continuous_unit_step(time)

# Define the number of samples for the discrete unit step signal
num_samples = 20  # Number of samples

# Simulate the discrete unit step signal
discrete_unit_step = simulate_discrete_unit_step(num_samples)

# Plot and display the continuous and discrete unit step signals
plt.figure(figsize=(10, 6))
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group<br>NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
plt.subplot(2, 1, 1)
plt.plot(time, continuous_unit_step)
plt.title('Continuous Unit Step Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.stem(discrete_unit_step)
plt.title('Discrete Unit Step Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

# Save the unit step signal arrays (optional)
# np.savetxt('continuous_unit_step.txt', continuous_unit_step, delimiter=',')
# np.savetxt('discrete_unit_step.txt', discrete_unit_step, delimiter=',')
```

**Output:-**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |




**4. Continuous and Discrete ramp signal:-**

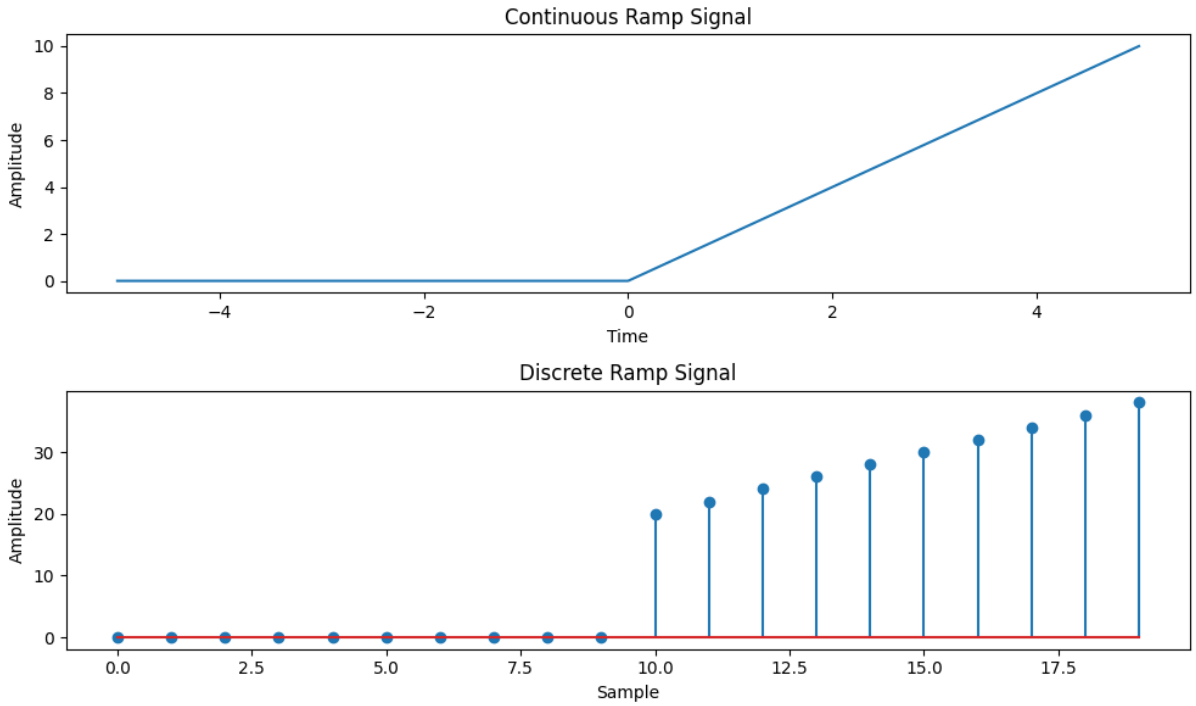
```
import numpy as np
import matplotlib.pyplot as plt


def simulate_continuous_ramp(time, slope):
    ramp = np.zeros_like(time)
    ramp[time >= 0] = slope * time[time >= 0]
    return ramp


def simulate_discrete_ramp(num_samples, slope):
    ramp = np.zeros(num_samples)
    ramp[num_samples // 2:] = slope * np.arange(num_samples // 2, num_samples)
    return ramp
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
# Define the time range for the continuous ramp signal
time = np.linspace(-5, 5, 1000)  # Time range from -5 to 5

# Define the number of samples and slope for the discrete ramp signal
num_samples = 20  # Number of samples
slope = 2  # Slope of the ramp

# Simulate the continuous ramp signal
continuous_ramp = simulate_continuous_ramp(time, slope)

# Simulate the discrete ramp signal
discrete_ramp = simulate_discrete_ramp(num_samples, slope)

# Plot and display the continuous and discrete ramp signals
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(time, continuous_ramp)
plt.title('Continuous Ramp Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.stem(discrete_ramp)
plt.title('Discrete Ramp Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
```

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

# Save the ramp signal arrays (optional)
# np.savetxt('continuous_ramp.txt', continuous_ramp, delimiter=',')
# np.savetxt('discrete_ramp.txt', discrete_ramp, delimiter=',')
```

**OUTPUT:-**




## 5. Continuous and Discrete exponential signal's:-

| | | |
|---|---|---|
| **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

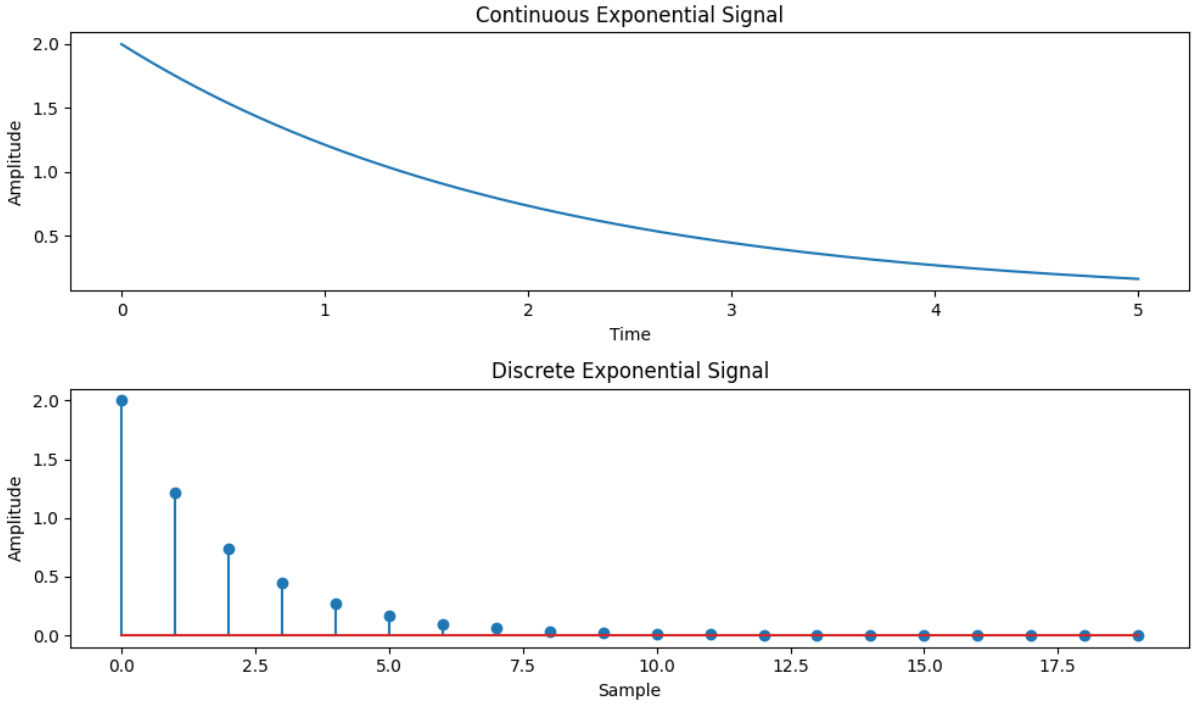
```
import numpy as np
import matplotlib.pyplot as plt

def simulate_continuous_exponential(time, amplitude, coefficient):
    exponential_signal = amplitude * np.exp(coefficient * time)
    return exponential_signal

def simulate_discrete_exponential(num_samples, amplitude, coefficient):
    exponential_signal = amplitude * np.exp(coefficient * np.arange(num_samples))
    return exponential_signal

# Define the time range for the continuous exponential signal
time = np.linspace(0, 5, 1000)  # Time range from 0 to 5

# Define the number of samples, initial amplitude, and coefficient for the discrete exponential signal
num_samples = 20  # Number of samples
amplitude = 2  # Initial amplitude
coefficient = -0.5  # Exponential coefficient

# Simulate the continuous exponential signal
continuous_exponential = simulate_continuous_exponential(time, amplitude, coefficient)

# Simulate the discrete exponential signal
discrete_exponential = simulate_discrete_exponential(num_samples, amplitude, coefficient)
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
# Plot and display the continuous and discrete exponential signals
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(time, continuous_exponential)
plt.title('Continuous Exponential Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.stem(discrete_exponential)
plt.title('Discrete Exponential Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

# Save the exponential signal arrays (optional)
# np.savetxt('continuous_exponential.txt', continuous_exponential, delimiter=',')
# np.savetxt('discrete_exponential.txt', discrete_exponential, delimiter=',')
```

**Output:-**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |




## 6. Continuous and Discrete parabolic signal's:-

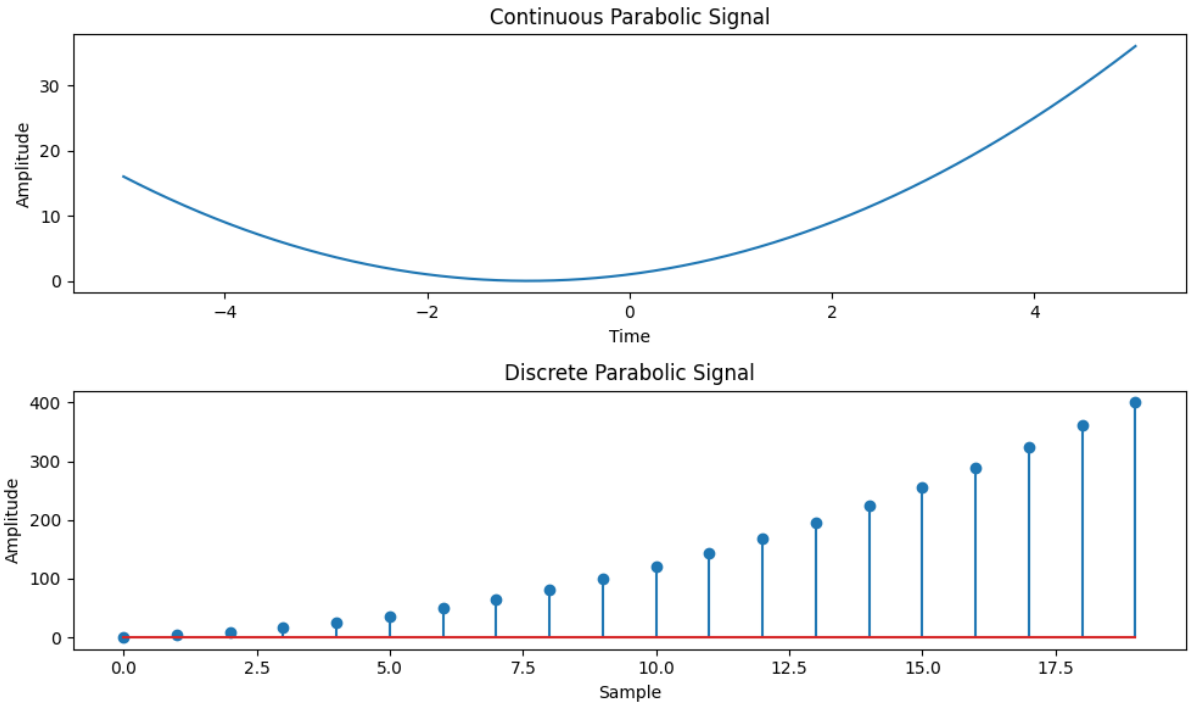
```
import numpy as np
import matplotlib.pyplot as plt

def simulate_continuous_parabolic(time, coefficients):
    parabolic_signal = np.polyval(coefficients, time)
    return parabolic_signal

def simulate_discrete_parabolic(num_samples, coefficients):
    parabolic_signal = np.polyval(coefficients, np.arange(num_samples))
    return parabolic_signal

# Define the time range for the continuous parabolic signal
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
time = np.linspace(-5, 5, 1000)  # Time range from -5 to 5

# Define the number of samples and coefficients for the discrete parabolic signal
num_samples = 20  # Number of samples
coefficients = [1, 2, 1]  # Coefficients of the parabolic signal

# Simulate the continuous parabolic signal
continuous_parabolic = simulate_continuous_parabolic(time, coefficients)

# Simulate the discrete parabolic signal
discrete_parabolic = simulate_discrete_parabolic(num_samples, coefficients)

# Plot and display the continuous and discrete parabolic signals
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(time, continuous_parabolic)
plt.title('Continuous Parabolic Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.stem(discrete_parabolic)
plt.title('Discrete Parabolic Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

plt.tight_layout()
```

```
plt.show()


# Save the parabolic signal arrays (optional)
# np.savetxt('continuous_parabolic.txt', continuous_parabolic, delimiter=',')
# np.savetxt('discrete_parabolic.txt', discrete_parabolic, delimiter=',')
```

**Output:-**




## 7. Continuous and Discrete sine wave signal:-

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


def simulate_continuous_sine_wave(time, amplitude, frequency, phase):
    sine_wave = amplitude * np.sin(2 * np.pi * frequency * time + phase)
    return sine_wave
```

| | | |
|---|---|---|
| **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

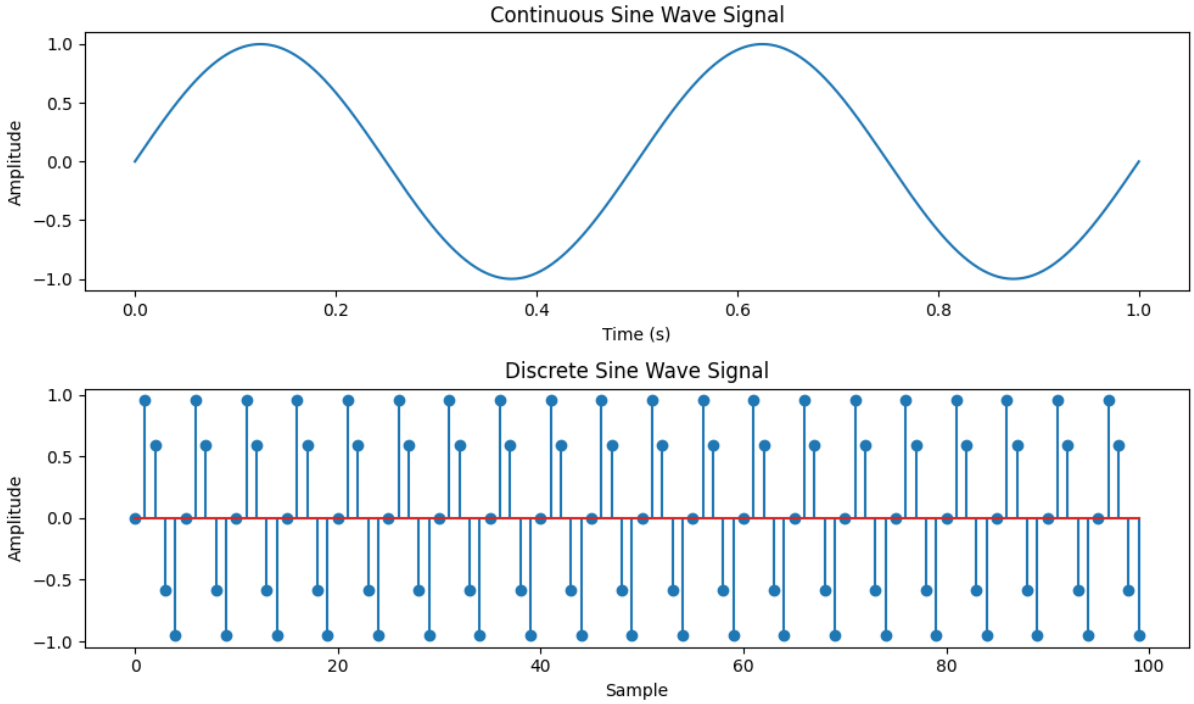
```
def simulate_discrete_sine_wave(num_samples, sampling_frequency, amplitude,
frequency, phase):
    time = np.arange(num_samples) / sampling_frequency
    sine_wave = amplitude * np.sin(2 * np.pi * frequency * time + phase)
    return sine_wave


# Define the time range for the continuous sine wave signal
time = np.linspace(0, 1, 1000)  # Time range from 0 to 1 second


# Define the number of samples, sampling frequency, and parameters for the discrete
sine wave signal
num_samples = 100  # Number of samples
sampling_frequency = 10  # Sampling frequency in Hz
amplitude = 1  # Amplitude of the sine wave
frequency = 2  # Frequency of the sine wave in Hz
phase = 0  # Phase angle of the sine wave in radians


# Simulate the continuous sine wave signal
continuous_sine_wave = simulate_continuous_sine_wave(time, amplitude, frequency,
phase)


# Simulate the discrete sine wave signal
discrete_sine_wave = simulate_discrete_sine_wave(num_samples,
sampling_frequency, amplitude, frequency, phase)


# Plot and display the continuous and discrete sine wave signals
plt.figure(figsize=(10, 6))
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group<br>NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
plt.subplot(2, 1, 1)
plt.plot(time, continuous_sine_wave)
plt.title('Continuous Sine Wave Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')

plt.subplot(2, 1, 2)
plt.stem(discrete_sine_wave)
plt.title('Discrete Sine Wave Signal')
plt.xlabel('Sample')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

# Save the sine wave signal arrays (optional)
# np.savetxt('continuous_sine_wave.txt', continuous_sine_wave, delimiter=',')
# np.savetxt('discrete_sine_wave.txt', discrete_sine_wave, delimiter=',')
```

**Output:-**

| | | |
|---|---|---|
| **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |



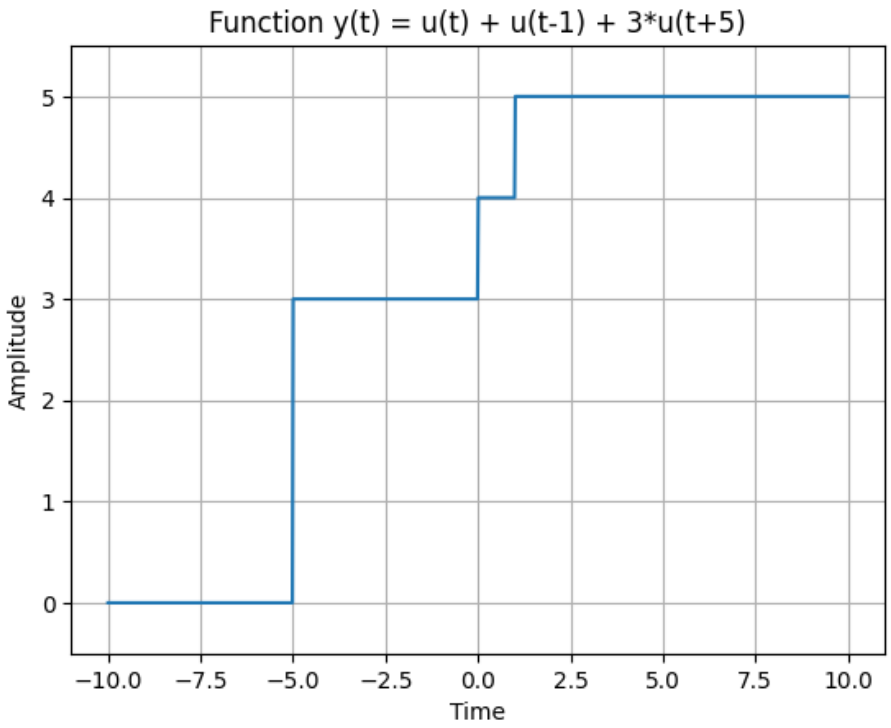

## 8. Simulating the signal of given function y(t)=u(t)+u(t-1)+3u(t+5).

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

def simulate_function(time):
    y = np.zeros_like(time)
    y[time >= 0] = 1
    y[time >= 1] += 1
    y[time >= -5] += 3
    return y

# Define the time range
time = np.linspace(-10, 10, 1000)
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

```
# Simulate the function
function_values = simulate_function(time)


# Plot and display the function
plt.plot(time, function_values)
plt.title('Function y(t) = u(t) + u(t-1) + 3*u(t+5)')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.ylim([-0.5, 5.5])
plt.grid(True)
plt.show()
```

**Output:-**



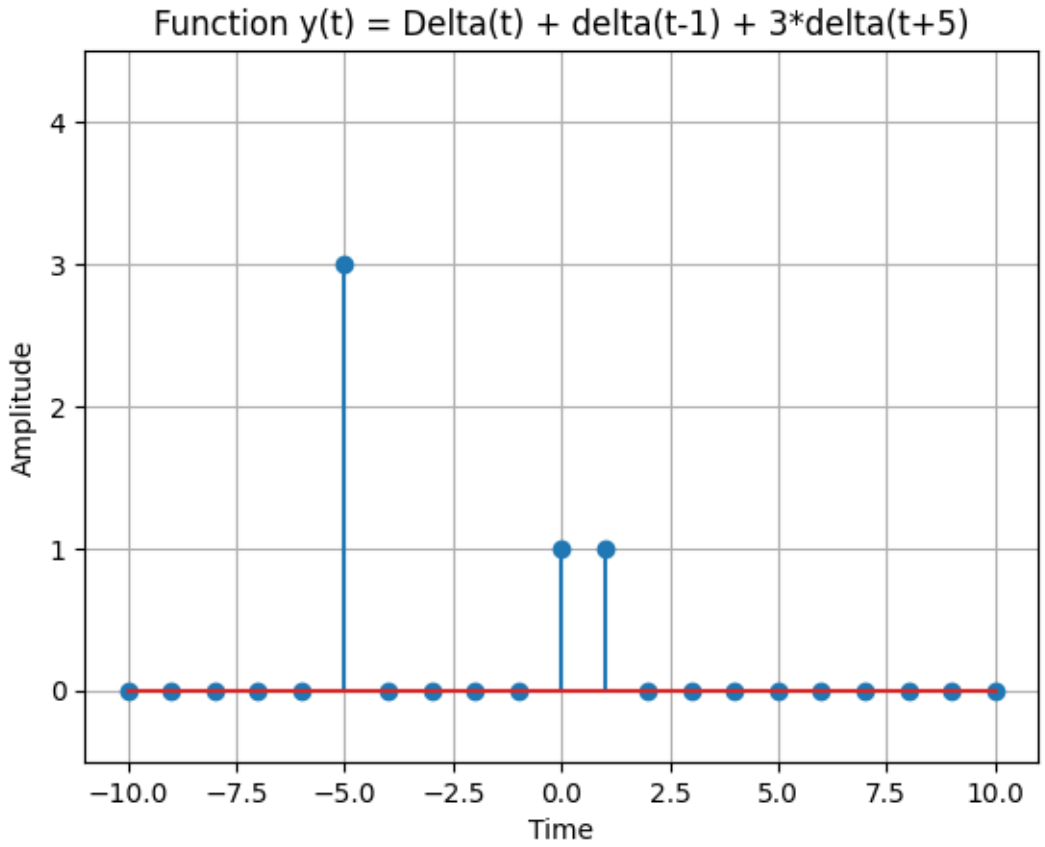

9. **Simulate given function → y(t)=Delta(t)+delta(t-1)+3*delta(t+5).**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group<br>NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |

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

def simulate_function(time):
    y = np.zeros_like(time)
    y[time == 0] = 1
    y[time == 1] += 1
    y[time == -5] += 3
    return y

# Define the time range
time = np.arange(-10, 11)

# Simulate the function
function_values = simulate_function(time)

# Plot and display the function
plt.stem(time, function_values)
plt.title('Function y(t) = Delta(t) + delta(t-1) + 3*delta(t+5)')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.ylim([-0.5, 4.5])
plt.grid(True)
plt.show()
```

**Output:-**

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group NAAC A+ | **Marwadi University**<br>**Faculty of Engineering & Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: DSIP (01CT0513)** | **Aim:-** Simulate discrete time sequence | |
| **Experiment:-** 1 | **Date:-** | **Enrollment No:-** 92301733041 |




## Conclusion:-

In this experiment, we created and studied some basic signals used in signal processing, such as unit impulse, impulse train, unit step, and ramp signals, in both continuous and discrete forms. By changing their parameters and observing their behavior, we understood their mathematical properties, how they change over time, and why they are important in analyzing and designng systems. This hands-on work made the theory clearer and showed how these basic signals act as the foundation for more advanced signal processing.