SVKM's D J Sanghvi College of Engineering

Vile Parle (W), Mumbai – 400 056

(Autonomous College Permanently Affiliated to the University of Mumbai)

NAAC Accredited with "A" grade (CGPA: 3.18)

DEPARTMENT OF ARIFICIAL INTELLIGENCE AND MACHINE LEARNING

Sem/Branch: V/AIML   Course Code: Digital Signal and Image Processing – Laboratory

Experiment No 1 **Date:** 14-09-2023

Name: Abhay Mathur **SAPID:** 60017210016 **Batch:** A1

Title: Plot of Discrete Time Signals

Aim: To Plot the Discrete Time Signals

Plot the following discrete time signals using folding and shifting property:

- Delta Function
- Unit Step Function
- Ramp Function
- $2^{(n)}$ u(n)
- $2^{(n)}$ u(n+1)
- $2^{(n)}$ u(n-1)
- $2^{(n)}$ u(-n+1)
- $2^{(n)}$ u(-n-1)

Learning Objectives: At the end of this experiment, students will be able to:

- To Plot different signals
- To learn speech processing

Apparatus: Python - Jupyter

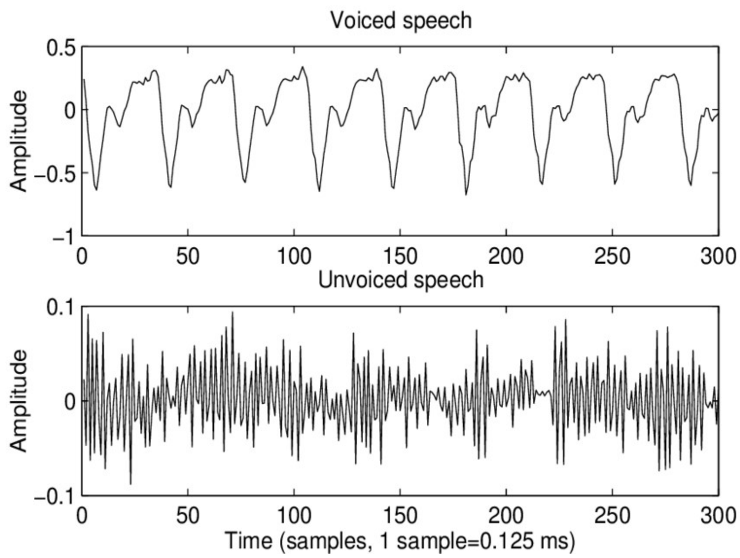
Theory: A signal is defined as any physical quantity that varies with time, space, or any other independent variable or variables. Mathematically, we describe a signal as a function of one or more independent variables. For example, the functions

$$S_1(t) = 5t$$

$$S_2(t) = 20t^2$$

describe two signals, one that varies linearly with the independent variable (time) and a second that varies quadratically with t. Signals can be further classified into four different categories depending on the characteristics of the time (independent) variable and the values they take.

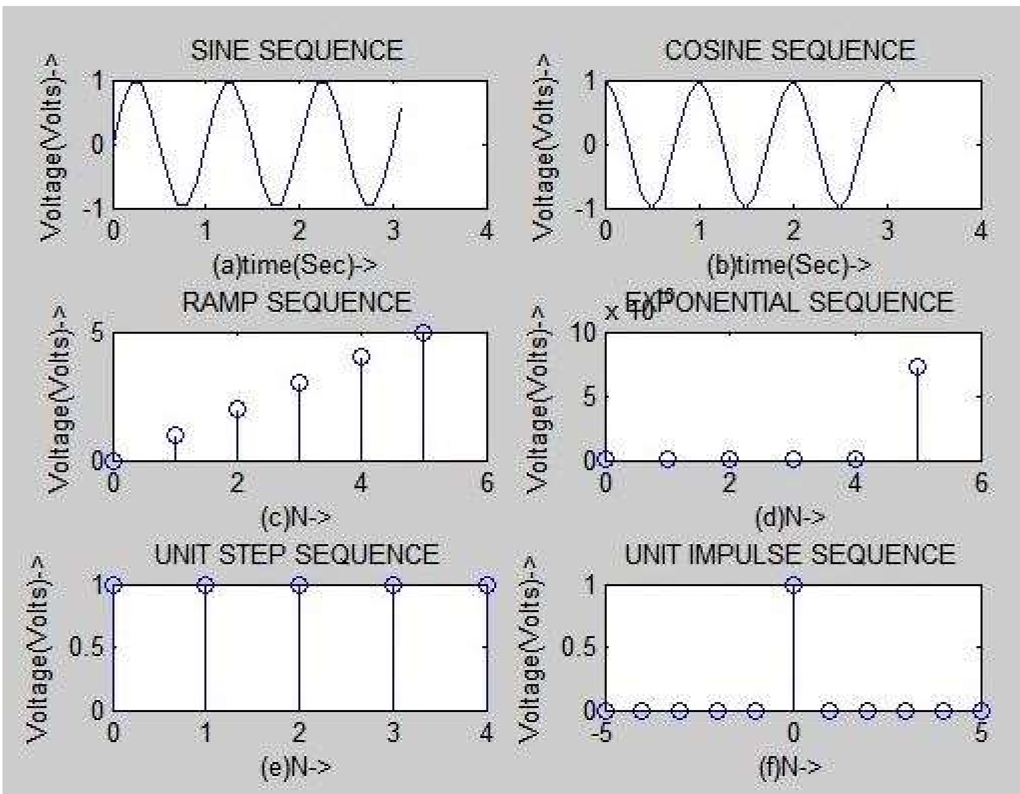
Continuous-time signals or analog signals are defined for every value of time and they take on values in the continuous interval (a, b) w here a can be -∞ and b can be ∞. Mathematically, these signals can be described by functions of a continuous variable. The speech waveforms are shown in figure below



Discrete-time signals are defined only at certain specific values of time. These time instants need not be equidistant, but in practice they a usually taken at equally spaced intervals for computational convenient and mathematical tractability. The signal x (n) = 0.8^n n = 0, ±1,±2 ... provides an example of a discrete -time signal. If we use the index n of the discrete-time instants as the independent variable, the sign value becomes a function of an integer variable (i.e., a square: numbers). Thus a mathematically by a discrete-time signal can be represented sequence of real or complex numbers. T emphasize the discrete-time nature of a signal, we shall denote such signal as x (n) instead of x (t). If the time instants are equally space (i.e., t = nT), the notation is also used. For the following examples of

- Step function
- Impulse function

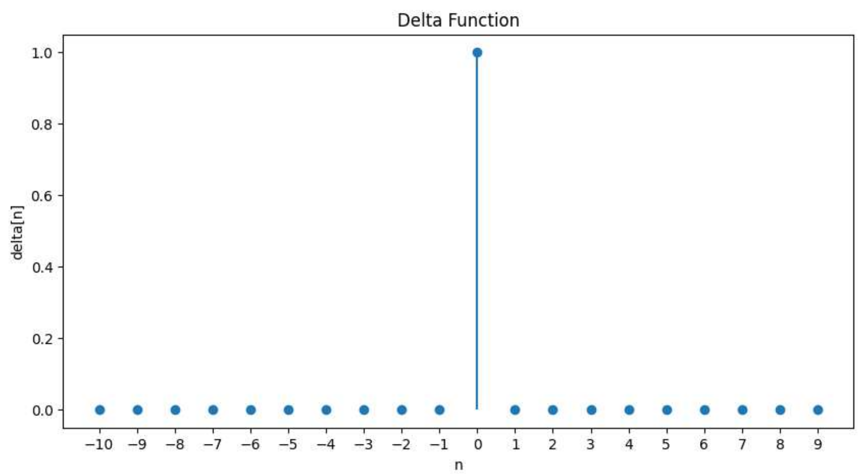
- Exponential function
- Ramp function
- Sine function
- Cosine function



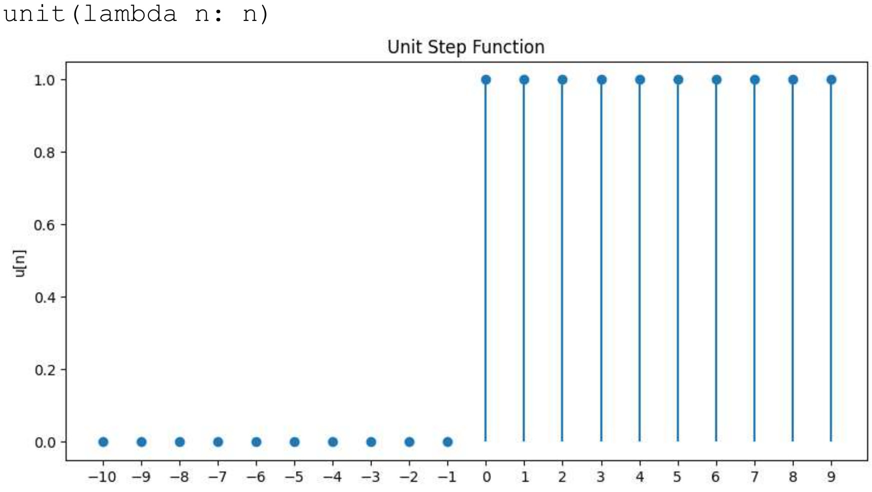
Code:

```
import matplotlib.pyplot as plt
#Delta Function
def delta(d):
    plt.figure(figsize=(10, 5))
    for n in range(-10, 10):
        if d(n) == 0:
            delta = 1
        else:
            delta = 0
        plt.stem(n, delta)

    plt.title('Delta Function')
    plt.xlabel('n')
    plt.ylabel('delta[n]')
    plt.xticks(range(-10, 10))


# Unit Step Function
def unit(d):
    plt.figure(figsize=(10, 5))
    for n in range(-10, 10):
        if d(n) < 0:
            u = 0
        else:
            u = 1
        plt.stem(n, u)

    plt.title('Unit Step Function')
    plt.xlabel('n')
    plt.ylabel('u[n]')
    plt.xticks(range(-10, 10))

# Ramp Function
def ramp(d):
    plt.figure(figsize=(10, 5))
    for n in range(-10, 10):
        if d(n) < 0:
            p = 0
        else:
            p = n
        plt.stem(n, p)

    plt.title('Ramp Function')
    plt.xlabel('n')
    plt.ylabel('p[n]')
    plt.xticks(range(-10, 10))

# (2^n)*u[n]
def exp(d):
```

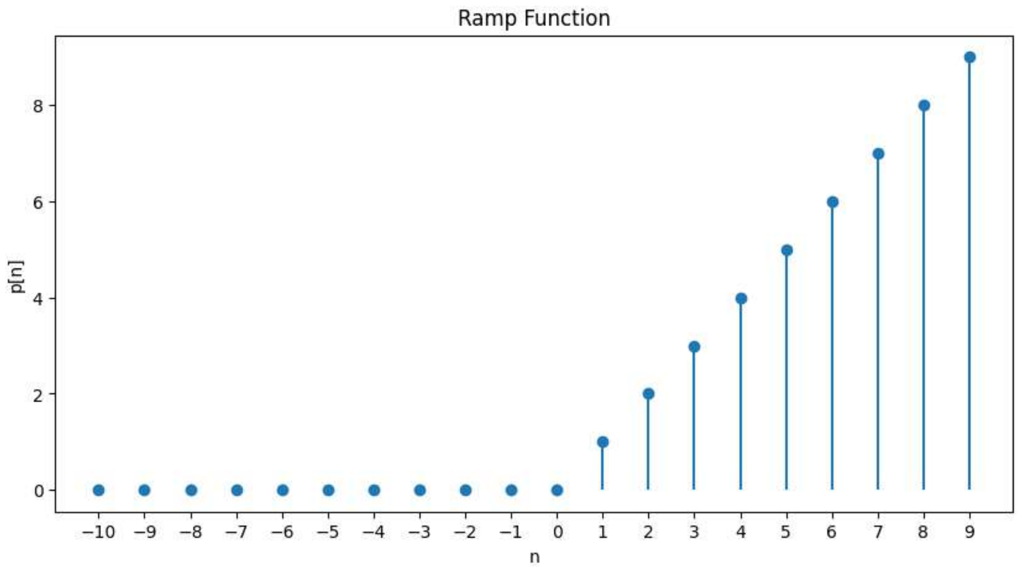
```
plt.figure(figsize=(10, 5))
    for n in range(-10, 10):
        if d(n) < 0:
            exp = 0
        else:
            exp = 2**n
        plt.stem(n, exp)

    plt.title('Exponential Function')
    plt.xlabel('n')
    plt.ylabel('2^n u[n]')
    plt.xticks(range(-10, 10))

#Delta Function Graph
delta(lambda n: n)
```



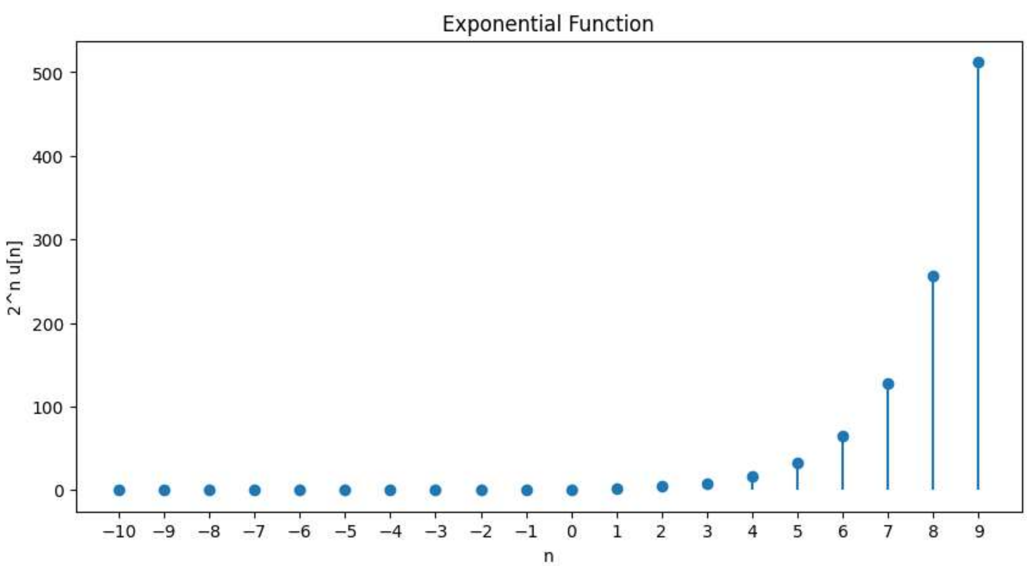

```
#Unit Step Function Graph
unit(lambda n: n)
```



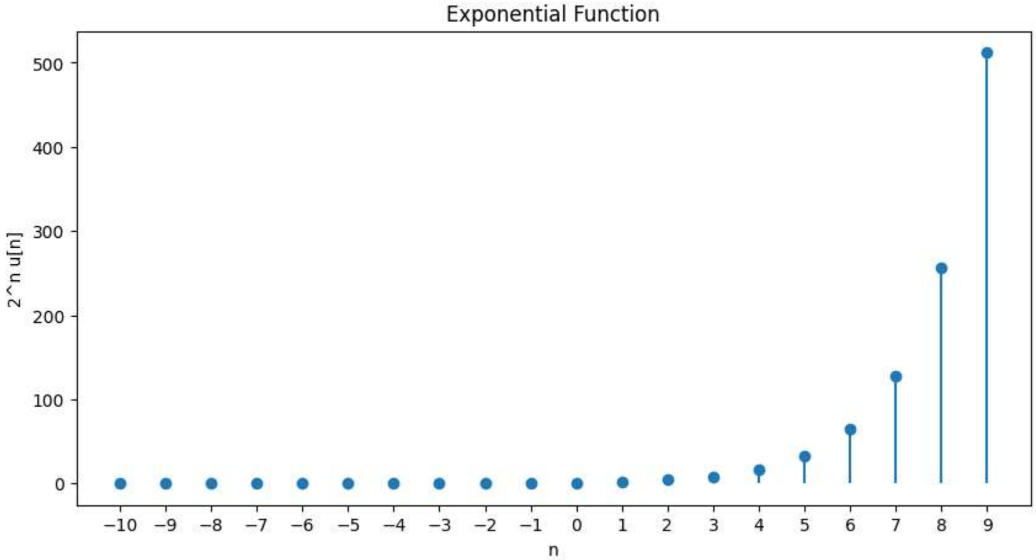
```
#Ramp Function Graph
ramp(lambda n: n)
```



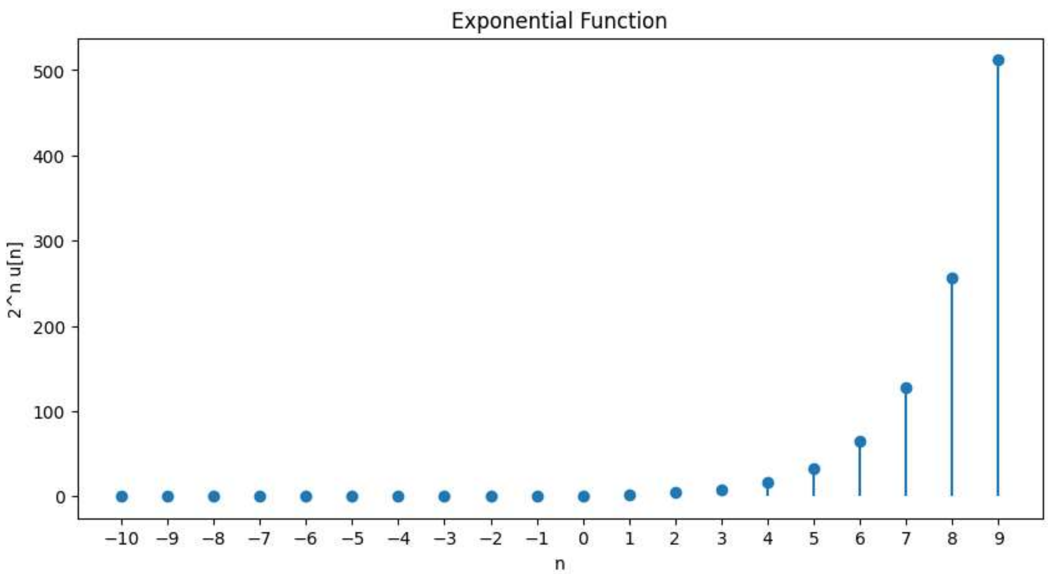

```
# (2^n)*u[n] Function Graph
exp(lambda n: n)
```



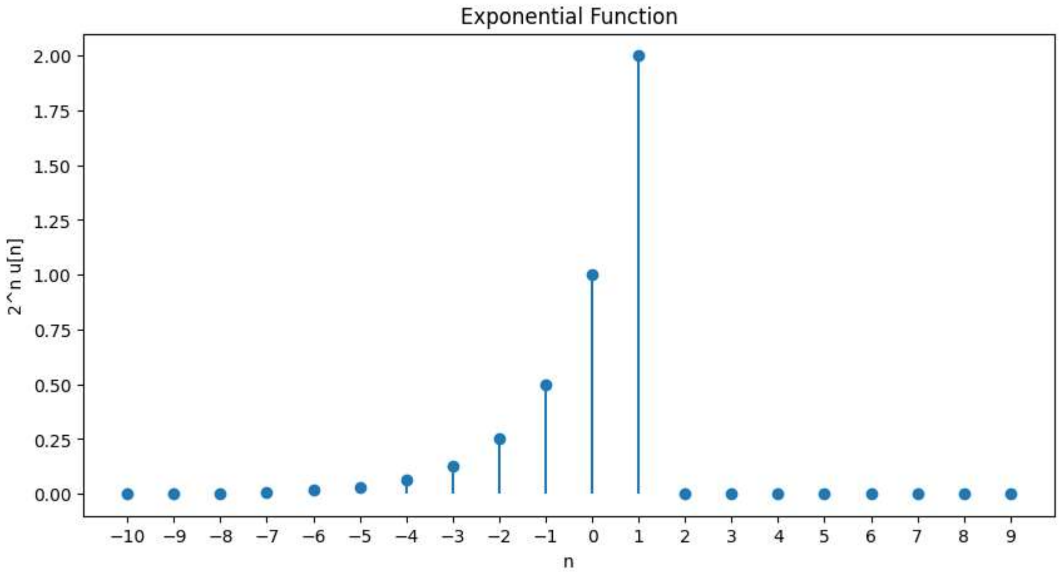
```
# (2^n)*u[n+1] Function Graph
exp(lambda n: n+1)
```




```
# (2^n)*u[n-1] Function Graph
exp(lambda n: n-1)
```



```
# (2^n)*u[-n+1] Function Graph
exp(lambda n: -n+1)
```




```
# (2^n)*u[-n-1] Function Graph
exp(lambda n: -n-1)
```