Digital signal processing lab

Name: Tasfia Tasneem Annesha

ID: 190041220

Course Code:CSE 4632

```
import numpy as np
```

```
import math
from decimal import *
```

Task 1

a.Create a row vector x of 5 equally spaced elements between 2 and 3 add 1 to the second element

The numpy function linspace() can create equally distant arrays of whatever size is required

```
x = np.linspace(2,3,5)
x[1] = x[1]+1
x
```

```
     array([2.  , 3.25, 2.5 , 2.75, 3.  ])
```

b.Create a second-row vector y of same dimension with elements equal to the successive even integers starting with 4.

```
y = np.arange(4,13,2)
print(y)
```

```
     [ 4  6  8 10 12]
```

c.Create the matrix A, whose first row is equal to x, whose second row is a line of ones, and whose third row is equal to y.

The numpy function ones() can create a vector of 1's of whatever size is required

```
A = [list(x), np.ones(5), list(y)]
A = np.array(A)
A
```

```
     array([[ 2.  ,  3.25,  2.5 ,  2.75,  3.  ],
            [ 1.  ,  1.  ,  1.  ,  1.  ,  1.  ],
            [ 4.  ,  6.  ,  8.  , 10.  , 12.  ]])
```

d.Define a row vector z, whose elements are equal to the mean value of the columns of A.

The numpy function mean() calculates the mean of the given array along any given axis

```
z = A.mean(axis=0)
z
```

```
     array([2.33333333, 3.41666667, 3.83333333, 4.58333333, 5.33333333])
```

e.Define a column vector zz, whose elements are the sum of the elements in each rows of A.

The numpy function sum() calculates the sum of the given array along any given axis.

```
zz = A.sum(axis=1)
zz
```

```
     array([13.5,  5. , 40. ])
```

Task 2----Create two matrices A and B:

$$A = \begin{pmatrix} 1 & 2 \\ 4 & -1 \end{pmatrix}, \qquad B = \begin{pmatrix} 4 & -2 \\ -6 & 3 \end{pmatrix}$$

a. Compute C1 = A+B and C2 = A-B

```
A = np.array([[1, 2], [4, -1]])
B = np.array([[4, -2], [-6, 3]])
C1 = A + B
print(C1)
```

```
     [[ 5  0]
```

```
      [-2  2]]
```

```
C2=A-B
print(C2)

     [[-3  4]
      [10 -4]]
```

b.Compute the matrix products D1 = AB and D2 = BA

The numpy function matmul() computes the matrix product given two matrices as params

```
D1=np.matmul(A,B)
D1

     array([[ -8,   4],
            [ 22, -11]])
```

```
D2=np.matmul(B,A)
D2

     array([[ -4,  10],
            [  6, -15]])
```

c. Using element wise operations, compute the matrix F whose elements are computed as follows:

$$f_{ij} = b_{ij} + a_{ij}b_{ij}^{1/4}$$

To enable complex numbers as answers in python we added +0j with the expression.

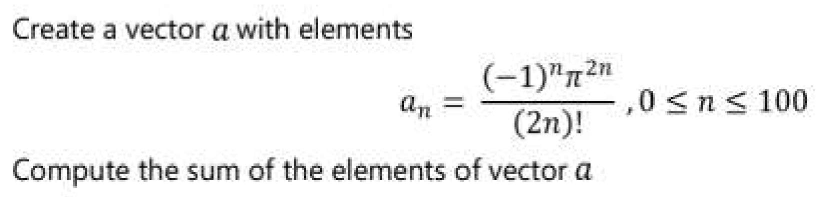
```
F = []
for i in range(0, 2):
    F.append([])
    for j in range(0, 2):
        f = B[i, j] + A[i, j] * (B[i, j]+0j) ** (1/4)
        F[i].append(f)
F = np.array(F)
F

     array([[ 5.41421356+0.j        , -0.31820717+1.68179283j],
            [-1.57327232+4.42672768j,  1.68392599+0.j        ]])
```

d. In A, subtract from the second row, the first row multiplied by 4

```
TempA = A
TempA[1] = A[1] - (A[0]*4)
TempA
```

```
     array([[ 1,  2],
            [ 0, -9]])
```

Task 3



Create a vector a with elements

$$a_n = \frac{(-1)^n \pi^{2n}}{(2n)!}, 0 \leq n \leq 100$$

Compute the sum of the elements of vector a

The Decimal class is able to handle very large calculations in python which is required for solving this problem.

```
a = []
for n in range(101):
    an = Decimal(((-1)**n) * (math.pi ** (2*n)))
    an = an/Decimal(math.factorial(2*n))
    a.append(an)
a = np.array(a)
print(a.sum())
```

```
     -1.000000000000000442826750528
```

4. Given x = [7 6 1 2 0 -1 4 3 -2 0], what are the commands that will execute the following operations?

a. Sets the negative values of x to zero.

The numpy function clip() limits the value of a vector to a specified min/max value.

```
import numpy as np
x = np.array([7, 6, 1, 2, 0, -1, 4, 3, -2, 0])
x
```

```
     array([ 7,  6,  1,  2,  0, -1,  4,  3, -2,  0])
```

```
x1 = x.clip(min=0)
x1

     array([7, 6, 1, 2, 0, 0, 4, 3, 0, 0])
```

b. Extract the values of x greater than 3 in a vector y.

```
x2 = x[x > 3]
x2

     array([7, 6, 4])
```

c. Add 3 to the values of x that are even.

```
x3 = np.array(x)
for i in range(x3.shape[0]):
    if x3[i] % 2 == 0:
        x3[i] = x3[i] + 3
x3

     array([ 7,  9,  1,  5,  3, -1,  7,  3,  1,  3])
```

d. Set the values of x that are less than the mean to zero.

```
x4 = np.array(x)
mean_x4 = x4.mean()
print(f'The mean is {mean_x4}')
for i in range(x4.shape[0]):
    if x4[i] < mean_x4:
        x4[i] = 0
x4

     The mean is 2.0
     array([7, 6, 0, 2, 0, 0, 4, 3, 0, 0])
```

e. Set the values of x those are greater than the mean to their difference with the mean.

At first the mean was found using the mean() function. Then the array was looped over and all values greater than the mean was set to their difference with the mean. The np.abs() function finds the absolute value of the given parameter.

```
x5 = np.array(x)
mean_x5= x5.mean()
```

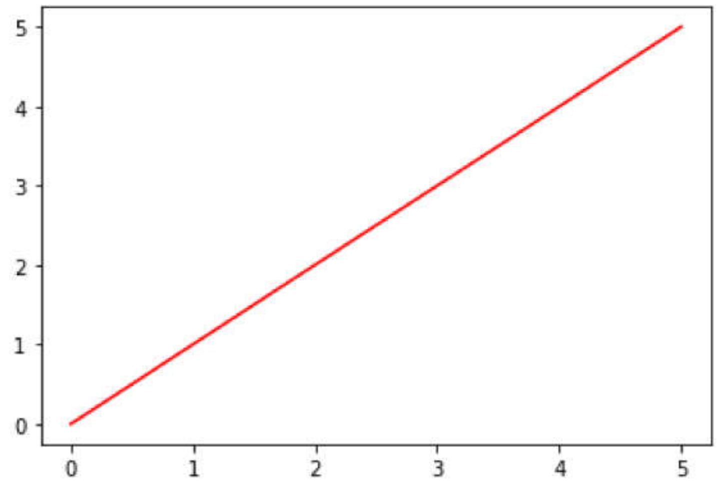
```
print(f'The mean is {mean_x5}')
for i in range(x5.shape[0]):
    if x5[i] > mean_x5:
        x5[i] = np.abs(x5[i] - mean_x5)
x5
```

```
     The mean is 2.0
     array([ 5,  4,  1,  2,  0, -1,  2,  1, -2,  0])
```

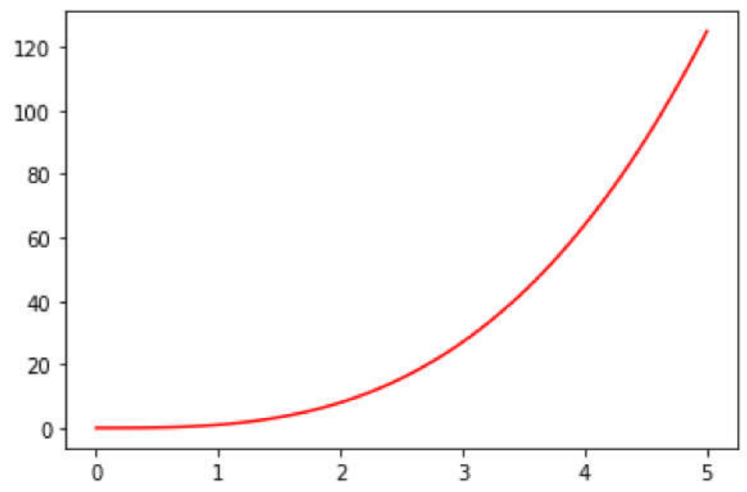
5. In MATLAB, plot the functions x, x3, and over the interval 0 < x < 5. Learn about the command hold and apply it. Use labels and titles for your plot.

The plot_function() takes the values of the x and y axes and plots then in a graph. The graphs are plotted using the matplotlib library.

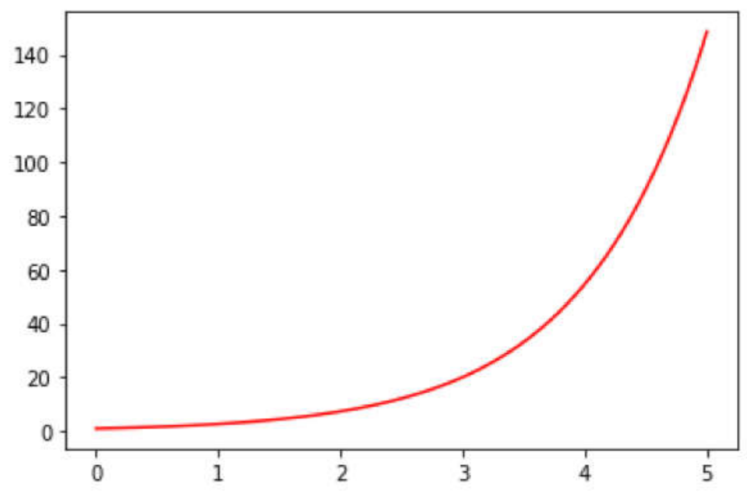
```
import matplotlib.pyplot as plt


x = np.linspace(0, 5, 100)
def plot_function(x, y):
    fig = plt.figure()
    ax = fig.add_subplot(1, 1, 1)
    plt.plot(x, y, 'r')
    plt.show()



import matplotlib.pyplot as plt
y = x
plot_function(x, y)
```



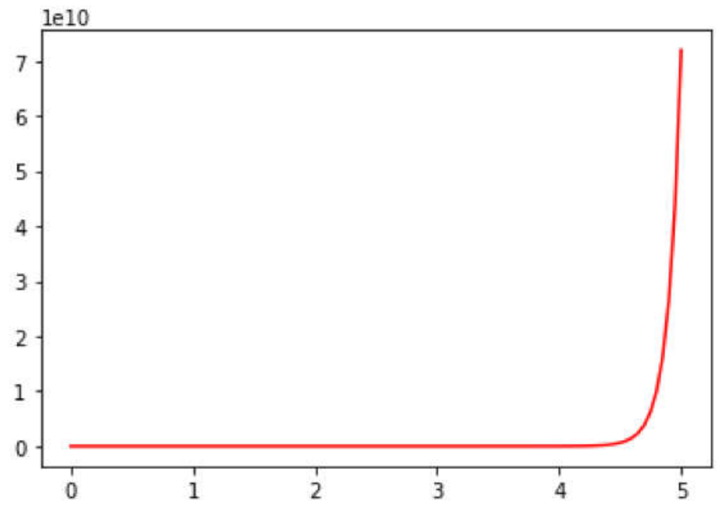
```
y = x**3
plot_function(x, y)
```



```
y = np.exp(x)
plot_function(x, y)
```



```
y = np.exp(x**2)
plot_function(x, y)
```



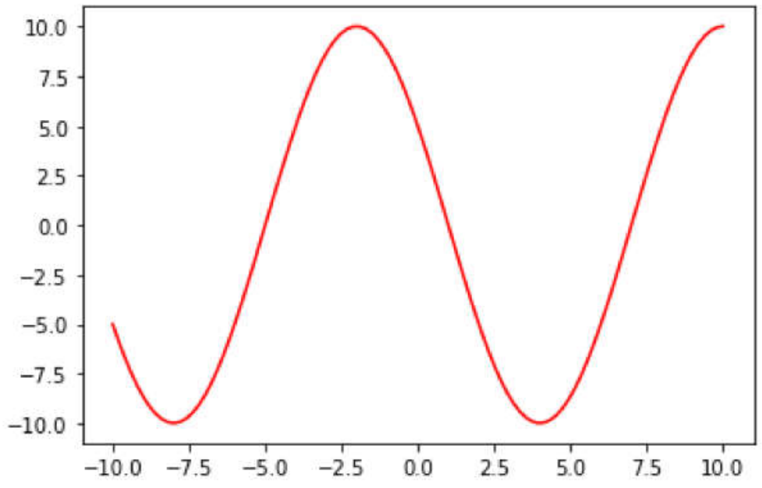
6. Plot a discrete-time sinusoidal signal with $\omega = \frac{\pi}{6}, \theta = \frac{\pi}{3}, A = 10$ where $-10 < n < 10$.

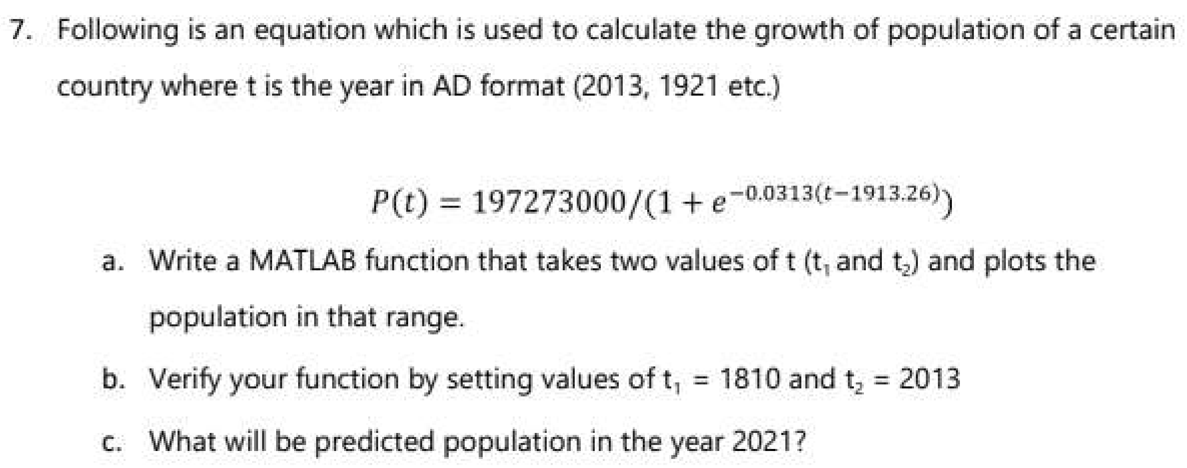
Here's a little bit of help for you, this is the general equation for a discrete-time sinusoidal signal.

$$x(n) = A\cos(\omega n + \theta)$$

In this problem math.pi was used to get the value of pi. The numpy function np.cos() does element wise cos for a given array. Later the plot_function() that was defined earlier was used to plot the graph of the function

```
n = np.linspace(-10, 10, 100)
A = 10
y = (math.pi/6)*n + (math.pi/3)
y·=·A*np.cos(y)
plot_function(n,·y)
```





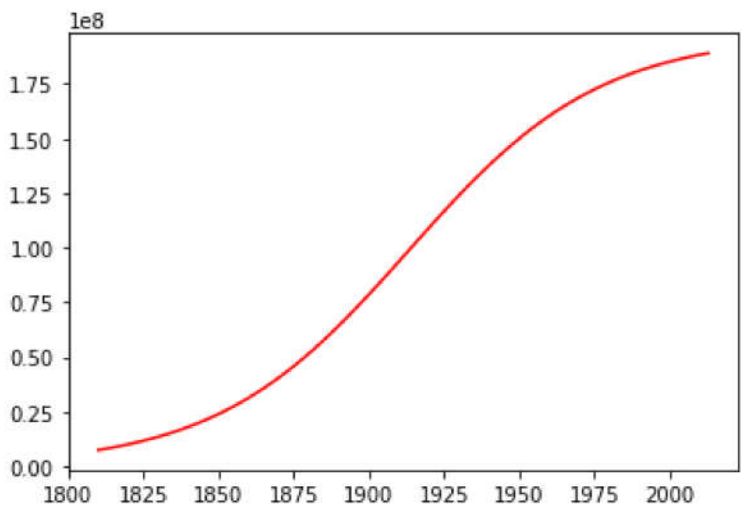
7. Following is an equation which is used to calculate the growth of population of a certain country where t is the year in AD format (2013, 1921 etc.)

$$P(t) = 197273000/(1 + e^{-0.0313(t-1913.26)})$$

   a. Write a MATLAB function that takes two values of t (t_1 and t_2) and plots the population in that range.
   b. Verify your function by setting values of t_1 = 1810 and t_2 = 2013
   c. What will be predicted population in the year 2021?

A function p(t1, t2, n_samples=100) is defined which takes two years and plot the population in that range using the given formula. The default number of samples to take in between is 100.

```
def p(t1, t2, n_samples=100):
    t = np.linspace(t1, t2, n_samples)
    y = 197273000/(1 + np.exp(-0.0313*(t-1913.26)))
    plot_function(t, y)
p(1810, 2013)
```



```
population_2021 = 197273000/(1 + np.exp(-0.0313*(2021-1913.26)))
population_2021

     190728728.107411
```

8. Check the help for "diag" and use it (may be more than once) to build the following 16x16 matrix:

$$D = \begin{bmatrix} -2 & 1 & 0 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & \cdots & 0 & 1 & -2 \end{bmatrix}$$

```
d1 = np.diag([-2 for i in range(16)])
d2 = np.diag([1 for i in range(15)], k=1)
d3 = np.diag([1 for i in range(15)], k=-1)
D = d1 + d2 + d3
D[0][15] = 1
D[15][0] = 1
D
```

```
array([[-2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1],
       [ 1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2,  1],
       [ 1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  1, -2]])
```

✓ 0s completed at 7:49 PM