Homework 3

Name: Mohamed Khaled Ahmed Atyaa

Group No.: 3 Alex

Question 1

P1=1/4*0+1/4*0+1/4*P2+1/4*P3

P2=1/5*0+1/5*P1+1/5*P3+1/5*P4+1/5*P5

P3=1/5*0+1/5*P1+1/5*P2+1/5*P5+1/5*P6

P4=1/5*0+1/5*P2+1/5*P5+1/5*P7+1/5*P8

P5=1/6*P2+1/6*P3+1/6*P4+1/6*P6+1/6*P8+1/6*P9

P6=1/5*0+1/5*P3+1/5*P5+1/5*P9+1/5*P10

P7=1/4*0+1/4*1+1/4*P4+1/4*P8

P8=1/5*1+1/5*P4+1/5*P5+1/5*P7+1/5*P9

P9=1/5*1+1/5*P5+1/5*P6+1/5*P8+1/5*P10

P10=1/4*0+1/4*1+1/4*P6+1/4*P9

The matrix form of equations is

$$\begin{matrix} 4 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 5 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 5 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 5 & -1 & 0 & -1 & -1 & 0 & 0 \\ 0 & -1 & -1 & -1 & 6 & -1 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 5 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 & -1 & 5 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & -1 & 5 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 4 \end{matrix}$$

Jacobi's Method code

```
import numpy as np
A=np.array([
 [4,-1,-1,0,0,0,0,0,0,0],
 [-1,5,-1,-1,-1,0,0,0,0,0],
 [-1,-1,5,0,-1,-1,0,0,0,0],
 [0,-1,0,5,-1,0,-1,-1,0,0],
 [0,-1,-1,-1,6,-1,0,-1,-1,0],
 [0,0,-1,0,-1,5,0,0,-1,-1],
 [0,0,0,-1,0,0,4,-1,0,0],
 [0,0,0,-1,-1,0,-1,5,-1,0],
 [0,0,0,0,-1,-1,0,-1,5,-1],
 [0,0,0,0,0,-1,0,0,-1,4]
])
```

```
b = np.array([0,0,0,0,0,0,1,1,1,1])
from numpy.linalg import inv, solve, norm


def jacobi(A, b, tolerance):
    xk_1 = np.zeros_like(b)
    D = np.diag(A)
    count=0
    LplusU = A - np.diag(D)
    x_k = (b - (LplusU @ xk_1)) / D
    while (norm(x_k - xk_1, 2) / norm(x_k, 2)) > tolerance:
        xk_1 = x_k
        x_k = (b - (LplusU @ xk_1)) / D
        count+=1
    return x_k,count

prob,iter=jacobi(A, b, 1e-8)
print("the values of probabilties are:",prob," and the # of
iterations=",iter)
```

solution

the values of probabilties are: [0.09019607 0.18039214 0.18039214 0.2980392 0.33333332 0.2980392

0.45490195 0.52156861 0.52156861 0.45490195] and the # of iterations= 69

P1=*0.09019607*

P2=*0.18039214*

P3=0.18039214

P4=0.2980392

P5=0.33333332

P6=0.2980392

P7=0.45490195

P8=0.52156861

P9=0.52156861

P10=0.45490195

Gauss-Siedel Method

```
import numpy as np
A=np.array([
 [4,-1,-1,0,0,0,0,0,0,0],
 [-1,5,-1,-1,-1,0,0,0,0,0],
 [-1,-1,5,0,-1,-1,0,0,0,0],
 [0,-1,0,5,-1,0,-1,-1,0,0],
 [0,-1,-1,-1,6,-1,0,-1,-1,0],
 [0,0,-1,0,-1,5,0,0,-1,-1],
 [0,0,0,-1,0,0,4,-1,0,0],
 [0,0,0,-1,-1,0,-1,5,-1,0],
 [0,0,0,0,-1,-1,0,-1,5,-1],
 [0,0,0,0,0,-1,0,0,-1,4]
])

b = np.array([0,0,0,0,0,0,1,1,1,1])
from numpy.linalg import inv


def siedel(A, b, N):
    x = np.zeros_like(b)  # initial solution (zeros)
    LD = np.tril(A)
    U = A - LD
    LDinv = inv(LD)
    for i in range(N):
        x = LDinv @ (b - U @ x)
    return x,N



prob,N=(siedel(A, b, 18))
print("the values of probabilties are:",prob," and the # of iterations=",N)
```

solution

the values of probabilties are: [0.09011191 0.1802798 0.18029671 0.29794377 0.33322943 0.29796987

0.45485001 0.52149928 0.52150972 0.4548699] and the # of iterations= 18

P1=*0.09019607*

P2=*0.18039214*

P3=0.18039214

P4=0.2980392

P5=0.33333332

P6=0.2980392

P7=0.45490195

P8=0.52156861

P9=0.52156861

P10=0.45490195

We see that Gauss-Siedel Method converge to same solution of Jacobi's Method after 18 iteration but Jacobi's Method code converge after 69 iteration so Gauss-Siedel Method is more efficient

Question 2 (1)

```
import numpy as np
from numpy.linalg import eig,norm
A = np.array([
    [0, 1, 2.],
    [0.5, 0, 0],
    [0, 0.25, 0]
])
l, v = eig(A)
print("The eigen values are :",l)
print("The eigen vector matrix is:",v)
x = np.random.rand(3)
for i in range(50):
    x = (A @ x) / norm(x,2)
lmbda = ((A@x) / x)[0]
v = x / lmbda
print("The first eigen value by power method is :",lmbda)
print("The first eigen vector is :",v)



solution
The eigen values are : [ 0.88464618+0.j        -0.44232309+0.2948714j -
0.44232309-0.2948714j]
The eigen vector matrix is: [[-0.86227396+0.j          0.69332593+0.j
0.69332593-0.j        ]
 [-0.48735527+0.j         -0.54259608-0.36171765j -0.54259608+0.36171765j]
 [-0.13772604+0.j          0.11796101+0.28307982j  0.11796101-0.28307982j]]
The first eigen value by power method is : 0.8846461770951362
The first eigen vector is : [0.86227396 0.48735527 0.13772604]
```

We see the eigenvector and the eigenvalue using built in function (eig) is the same as power method

Question 2 (2)

```
import numpy as np
from numpy.linalg import eig,norm
A = np.array([
    [0, 6, 8],
    [0.5, 0, 0],
    [0, 0.5, 0]
])
l, v = eig(A)
print("The eigen values are :",l)
print("The eigen vector matrix is:",v)
x = np.random.rand(3)
for i in range(50):
    x = (A @ x) / norm(x,2)
lmbda = ((A@x) / x)[0]
v = x / lmbda
print("The first eigen value by power method is :",lmbda)
print("The first eigen vector is :",v)

solution
The eigen values are : [ 2.         -0.99999998 -1.00000002]
The eigen vector matrix is: [[-0.96836405  0.87287156 -0.87287156]
 [-0.24209101 -0.43643579  0.43643578]
 [-0.06052275  0.2182179  -0.21821788]]
The first eigen value by power method is : 2.0000000000000133
The first eigen vector is : [0.96836405 0.24209101 0.06052275]
```

We see the eigenvector and the eigenvalue using built in function (eig) is the same as power method