Project4

（一）本次作業學習目標：

實作 diagonalization 在簡化矩陣的冪運算的應用。

（二）作業要求（一項未符合標準作業成績-10）

1.壓縮檔案名稱：LA_project04_學號_version

2.ipynb 用我們所提供的檔案去撰寫，檔案名稱：LA_project04_學號_version

3.pdf 檔案名稱：report4_學號_version

4.心得禁止全篇只寫心路歷程，還要寫學到了什麼 ex：哪些函式、算法，以及思考。

5.pdf&ipynb 放在一個檔案夾裡壓縮成.zip 檔。

（三）作業繳交期限與更新

1.作業繳交 deadline：11/30（三）23：59 前。

2.期限之後繳交作業依天數打折 ex：遲交一天打 8 折，遲交兩天打 6 折，以此類推。

3.上傳作業後請確認規格及內容，不接受任何理由，遲交及錯誤皆按照規定扣分。

（四）作業配分說明

1.各小題配分如下標示。

2.照著檔案中引導所寫之答案滿分為 60，加上心得 40 分，作業總成績滿分為 100。

（五）作業題目內容

給定一個矩陣 $A=\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}$**，用 Diagonalization 的方法，計算**A^{100}**。**

請勿將矩陣 A 直接連乘 100 次（ $D = P^{-1}AP$ $\qquad$ $A^{100} = PD^{100}P^{-1}$）

Step1：使用 numpy 函式求出矩陣A的 eigenvalues、eigenvectors。(30pt)

```
λ =  2.8284271247461894
λ =  2.236067977499789
λ =  -2.828427124746192
λ =  -2.23606797749979
```

```
p1 =  [-0.15622986  0.68963195 -0.37717224 -0.59811462]
p2 =  [ 0.69455348 -0.13264791  0.42925766  0.56190557]
p3 =  [-0.37717224  0.59811462  0.15622986  0.68963195]
p4 =  [ 0.42925766 -0.56190557 -0.69455348 -0.13264791]

P =
 [[-0.15622986  0.69455348 -0.37717224  0.42925766]
 [ 0.68963195 -0.13264791  0.59811462 -0.56190557]
 [-0.37717224  0.42925766  0.15622986 -0.69455348]
 [-0.59811462  0.56190557  0.68963195 -0.13264791]]
```

Step2：使用 for 迴圈求出P^{-1}。(10pt)

```
P^(-1) =
 [[ 0.75434448  1.06680419 -0.31245971 -0.44188477]
 [ 1.25645904  0.56190557  0.29660975  0.13264791]
 [-0.31245971  0.44188477 -0.75434448  1.06680419]
 [ 0.29660975 -0.13264791 -1.25645904  0.56190557]]
```

Step3：使用 numpy 的.dot()做矩陣相乘$P^{-1}AP$，求出對角矩陣D。(10pt)

```
D =
 [[ 2.82842712  0.          0.         -0.        ]
 [-0.          2.23606798  0.         -0.        ]
 [-0.          0.         -2.82842712 -0.        ]
 [ 0.          0.          0.         -2.23606798]]
```

Step4：做矩陣相乘$PD^{100}P^{-1}$，求出A^{100}。(10pt)

（不強制一定用 for 迴圈，可以使用@或 np.dot，但方法步驟必須嚴格按照題目要求用對角化的方法解）

(根據同學們上面步驟 1~3 的計算方式不同，可能A^{100}的值會略有不同，與下方輸出的值四捨五入後誤差在+-0.001 好了即可)

```
A^100 =
 [[ 8.88173984e+34 -4.75749231e+44  4.75749231e+44 -4.75749231e+44]
 [ 4.75749231e+44  1.42724769e+45 -9.51498462e+44  4.75749231e+44]
 [-4.75749231e+44 -4.75749231e+44  8.88193948e+34  4.75749231e+44]
 [-9.51498462e+44 -4.75749231e+44 -4.75749231e+44  1.42724769e+45]]
```

（六）作業心得內容

1.矩陣相乘求 A 的 n 次方的時間複雜度是多少？diagonalization 的方法時間複雜度是多少？即相比矩陣乘法，使用對角化求矩陣的 n 次冪，節省了多少次乘法運算？(30pt)

2.解釋為什麼 P 是由 eigenvector 構成的 invertible matrix 時，得到的 similar matrix 是 diagonal matrix？(10pt)

(七) 作業語法補充

1.pow(x,y)：代表 x^y

pow(x,y,z)：代表 x^y/z 的餘數

```
print(pow(2,4))
print(pow(2,4,5))
```

```
16
1
```

（八）numpy 套件補充

1.numpy.identity(n, dtype=None, *, like=None)：建一個 n*n 的 identity 矩陣

```
    print(np.identity(4))
    print(np.identity(4,dtype=int))
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]
 [0 0 0 1]]
```

2.numpy.ndarray.T：為轉置矩陣的應用，然而若為一維矩陣輸出的並不會是真正的轉置矩陣，而是一個橫向向量。

```
    A = np.array([[1,2],[3,4]])
    print(A)
    print(A.T)
    print(A[0].T)
```

```
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
[1 2]
```

3.numpy.linalg.eig(A):取 A 矩陣的 eigValue, eigVector 值

eigValue[i]對應的 eigVector 為 eigVector[:,i]或 eigVector.T[i]

```
from numpy.linalg import eig
A = np.diag((1, 2, 3))
eigValue, eigVector = eig(A)
print(A)
print('\n')
print(eigValue)
print(eigVector)
print('\n')
print(eigValue[0])
print(eigVector[:,0])
print(eigVector.T[0])
```

```
[[1 0 0]
 [0 2 0]
 [0 0 3]]


[1. 2. 3.]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]


1.0
[1. 0. 0.]
[1. 0. 0.]
```

4.@：用法語 np.dot()的用法相同

```
A=np.array([[1,2],[3,4]])
B=np.array([[1,2],[3,4]])
print(np.dot(A,B))
print(A@B)
```

```
[[ 7 10]
 [15 22]]
[[ 7 10]
 [15 22]]
```

Ps.如遇到任何不了解的地方請詢問助教，預祝各位解題順利