

# Project2

## (一) 本次作業學習目標：

1. 算法：gaussian elimination, back substitution method and forward substitution method
2. 結構：for 迴圈，向量運算，二维矩陣的應用

## (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\_project02\_學號\_version
2. ipynb 用我們所提供的檔案去撰寫，檔案名稱：LA\_project02\_學號\_version
3. pdf 檔案名稱：report2\_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：10/18（二）00：00 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 70，加上心得 10 分，基本總作業成績滿分為 80。
3. 第三題的高斯消去法中，由自己編寫之程式碼答案，並且符合該題要求之條件，再於旁邊註記或心得中說明解題思路，得額外加分 20 分。

## (五) 作業題目限制說明

1. Q1 及 Q2 只能使用一層 for 迴圈撰寫，Q3 最多只能使用兩層 for 迴圈撰寫，試著使用向量計算的方式減少 for 迴圈的使用（向量計算方法詳請洽臉書社團）

## (六) 作業題目內容

Q1 : Triangular systems-upper triangular system method. (20pt)

$$\text{Backward substitution: } A = \begin{bmatrix} -2 & 1 & 2 \\ 0 & 3 & -2 \\ 0 & 0 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 9 \\ -1 \\ 8 \end{bmatrix}$$

$$\text{Output: } x = [-2 \quad 1 \quad 2]$$

The Algorithm of [backward substitution](#) for your reference :

(backward substitution 是針對 upper triangular 從下至上迭代消除的方法)

Given u, b

$$x_n = b_n / u_{nn}$$

for i= n-1 . . . 1

$$s = b_i$$

for j = i + 1 . . . n

$$s = s - u_{i,j} * x_j$$

end

$$x_i = s/u_{i,i}$$

end

Q2 : Triangular systems-lower triangular system method. (20pt)

$$\text{Forward substitution: } A = \begin{bmatrix} -2 & 0 & 0 \\ 1 & 3 & 0 \\ 2 & -2 & 4 \end{bmatrix} \quad B = \begin{bmatrix} -2 \\ 7 \\ -6 \end{bmatrix}$$

$$\text{Output: } x = [1 \quad 2 \quad -1]$$

The Algorithm of [forward substitution](#) for your reference :

(forward substitution 是針對 lower triangular 從上至下迭代消除的方法)

given F, b

$$x_1 = b_1/f_{11}$$

for i = 2 . . . n

$$s = b_i$$

for j = 1 . . . i-1

```

        s = s-f_i, j * x_j
    end
xi = s/f_i, i
end

```

**Q3 : Guassion elimination method to solve linear system. (30pt)**

Ax=b, for x

$$A = \begin{bmatrix} 2 & 4 & -2 & -2 \\ 1 & 2 & 4 & -3 \\ -3 & -3 & 8 & -2 \\ -1 & 1 & 6 & -3 \end{bmatrix} \quad b = \begin{bmatrix} -4 \\ 5 \\ 7 \\ 7 \end{bmatrix}$$

The Algorithm of Guassion elimination for your reference :

(這裡提供的只是最簡單的 function，高斯消去實際上要考慮到 trace=0，以及找 pivot 等情況，繳交的答案有做到其中一個且註記或是附上心得就額外加 20 分)

```

Function Ab = demoGE(A,b)

[m,n] = size(A);

Ab = [A b];

for i = 1:n-1
    for k = i+1:n
        Ab(k,i:n+1) = Ab([k, i:n+1])
            -Ab(i,i:n+1)*Ab(k,i)/Ab(i,i);
    end
end

step1 : Guassion elimination to upper triangular matrix.
step2 : Backward method to solve Ab.

```

### (七) for 迴圈語法補充說明

1. `range`(起始值, 結束值, 遞增(減)值)。

```
for i in range(10,20,3):  
    print(i)
```

10  
13  
16  
19

2. 可以針對 Iterable(可迭代的)物件來進行讀取

Python 內建幾個常用的 Iterable 物件。Ex: `String`(字串)、`List`(串列)、`Tuples`(元組)、`Dictionary`(字典)。

```
for i in 'apple':  
    print(i)
```

a  
p  
p  
l  
e

3. `Nested Loops`(巢狀迴圈)，即迴圈內又有迴圈。

```
for i in range(3):  
    for j in range(3):  
        print(i,j)
```

0 0  
0 1  
0 2  
1 0  
1 1  
1 2  
2 0  
2 1  
2 2

## (七) numpy 套件語法補充教學

1. `numpy.arange(start, stop, step, dtype=None, like=None)`: 產生一個等差數列的陣列。

`start` : 從某數開始, 預設為 0

`stop` : 到某數停止(不包含)

`step` : 間隔多少

`dtype` : `bool`(布林值), `int`(整數), `float`(浮點數), `complex`(複數)

```
print(np.arange(1,10))
print(np.arange(1,10,2,float))
print(np.arange(1,11,2,float))
print(type(np.arange(1,11,2,float)))

[1 2 3 4 5 6 7 8 9]
[1. 3. 5. 7. 9.]
[1. 3. 5. 7. 9.]
<class 'numpy.ndarray'>
```

2. `numpy.triu(m, k=0)`: 取得上三角矩陣, 返回值其中元素 `k-th` 對角線包含以下歸零。

```
print(np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]]),'\\n')
print(np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]],-1),'\\n')
print(np.triu([[1,2,3],[4,5,6],[7,8,9],[10,11,12]],1))

[[1 2 3]
 [0 5 6]
 [0 0 9]
 [0 0 0]]

[[ 1  2  3]
 [ 4  5  6]
 [ 0  8  9]
 [ 0  0 12]]

[[0 2 3]
 [0 0 6]
 [0 0 0]
 [0 0 0]]
```

3. **numpy.tril**(m, k=0):取得下三角矩陣，返回的值其中元素 k-th 對角線包含以上歸零 (k 默認為 0)。

```
print(np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]]),'\n')
print(np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]],-1),'\n')
print(np.tril([[1,2,3],[4,5,6],[7,8,9],[10,11,12]],1))
```

Python

```
[[ 1  0  0]
 [ 4  5  0]
 [ 7  8  9]
 [10 11 12]]

[[ 0  0  0]
 [ 4  0  0]
 [ 7  8  0]
 [10 11 12]]

[[ 1  2  0]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

4. python **slice** 用法:[起點 : 終點(不包含) : 間隔]

預設值為 [0: 提取到最後一個:1]，[:-1]表示為到最後一個元素(不包含)。

```
U=np.array([-2, 1, 2, 0, 3, -2, 0, 0, 4])
print(U[1:6])
print(U[1:6:2])
print(U[:-1],'\n')
U=U.reshape(3,3)
print(U,'\n')
print(U[2,1:3])
print(type(U[2,1:3]))
```

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

```
[[-2  1  2]
 [ 0  3 -2]
 [ 0  0  4]]

[0 4]
<class 'numpy.ndarray'>
```

5. `numpy.concatenate((a1, a2, ...), axis=0, out=None, dtype=None, casting="same_kind")`: 用來**拼接 array**，可為多組或多維。

```
a = np.array([1, 2])
b = np.array([3, 4])
c = np.array([5, 6])
np.concatenate((a,b,c))

array([1, 2, 3, 4, 5, 6])
```

6. 一維陣列：axis 的值不影響結果

二維陣列：axis=0 按照行拼接，axis=1 按照列拼接，axis=None 使變成一維陣列，  
( axis=0 為預設值)

```
a = np.array([[1, 2], [3, 4]])
b = np.array([[5, 6]])
print(a,b,'\n')
print(np.concatenate((a, b), axis=0),'\n')
print(np.concatenate((a, b.T), axis=1),'\n')
print(np.concatenate((a, b), axis=None))

[[1 2]
 [3 4]] [[5 6]]

[[1 2]
 [3 4]
 [5 6]]

[[1 2 5]
 [3 4 6]]

[1 2 3 4 5 6]
```

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