Numerical Method

National Cheng Kung University Department of Engineering Science

Instructor: Chi-Hua Yu

Lab 4

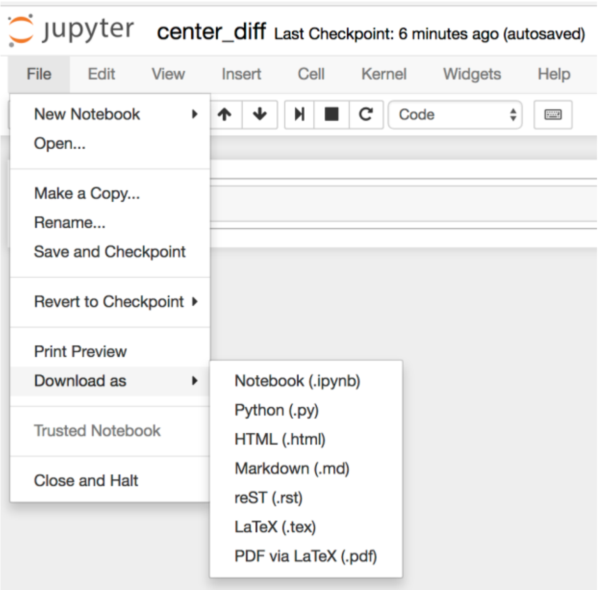
Programming, Due 10:00, Wednesday, March 30th , 2022

注意事項：

1. Lab 的時間為授課結束(Lab 當天 10:00)。
2. Lab 的分數分配：出席 20%，Lab 分數 100%，Bonus 20%。
3. 請儘量於 Lab 時段完成練習，完成後請找助教檢查，經助教檢查後沒問題者請用你的學號與 Lab number 做一個檔案夾 (e.g., N96091350_Lab24, 將你的全部 ipynb 檔放入檔案夾，壓縮後上傳至課程網站 (e.g., N96091350_Lab4.zip)。
4. 上傳後即可離開。
5. 未完成者可於隔日 11:55 pm 前上傳至 Moodle，惟補交的分數將乘以 0.8 計，超過期限後不予補交。
6. Bouns 只需要在每週四的 11:55 pm 上傳即可。

Lab Submission Procedure (請仔細閱讀)

1. You should submit your Jupyter notebook and Python script (`*.py`, in Jupyter, click File, Download as, Python (`*.py`)).



2. Name a folder using your student id and lab number (e.g., n96081494_lab1), put all the python scripts into the folder and zip the folder (e.g., n96081494_lab1.zip).

3. Submit your lab directly through the course website.

1. **(100%)** Name your Jupyter notebook `gauss_elimination.ipynb` and Python script `gauss_elimination.py`. Write a Python program to solve the equations by using the Gauss elimination method.

$$Ax = y$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ a_{2,1} & a_{2,2} & a_{2,3} & a_{2,4} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,4} \\ a_{4,1} & a_{4,2} & a_{4,3} & a_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

The Gauss elimination method is a procedure that turns the matrix *A* into an upper-triangular form to solve the system of equations.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & a'_{2,2} & a'_{2,3} & a'_{2,4} \\ 0 & 0 & a'_{3,3} & a'_{3,4} \\ 0 & 0 & 0 & a'_{4,4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix}$$

Below is the running example

Sample 1

```
a = np.array([[1, 2, 3],[3, 4, 5],[3, 5, 5]])
y = np.array([2, 2, 5])

upper triangular matrix:
[[ 1.  2.  3.  2.]
 [ 0. -2. -4. -4.]
 [ 0.  0. -2.  1.]]

x:
[-2.5,  3. , -0.5]
```

Sample 2

```
a = np.array([[1, 2, 3, 4],[5, 4, 3, 2],[2, 1, 2, 4],[2, 1, 3, 4]])
y = np.array([4, 8, 5, 6])

upper triangular matrix:
[[  1.    2.    3.    4.    4. ]
 [  0.   -6.  -12.  -18.  -12. ]
 [  0.    0.    2.    5.    3. ]
 [  0.    0.    0.   -2.5  -0.5]]

x:
[ 1.4, -0.6,  1. ,  0.2]
```

Bonus. (20%): Name your Jupyter notebook `inverse_to_solve` and Python script `inverse_to_solve.py`. When A is square and invertible. We can solve equation $Ax = y$ by multiplying each side of thex matrix by A^{-1}.

$$A^{-1} = \frac{\text{abj}(A)}{\det(A)}$$

$$A^{-1}Ax = A^{-1}y$$

Please write a program to calculate adjugate matrix and use `np.linalg.det` to get A^{-1}. Finally, we can use $x = A^{-1}y$ to compute x.

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \qquad \text{abj}(A) = \begin{bmatrix} a_{2,2} & -a_{2,1} \\ -a_{1,2} & a_{1,1} \end{bmatrix}^T$$

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \qquad \text{abj}(A) = \begin{bmatrix} +\begin{vmatrix} a_{2,2} & a_{2,3} \\ a_{3,2} & a_{3,3} \end{vmatrix} & -\begin{vmatrix} a_{2,1} & a_{2,3} \\ a_{3,1} & a_{3,3} \end{vmatrix} & +\begin{vmatrix} a_{2,1} & a_{2,2} \\ a_{3,1} & a_{3,2} \end{vmatrix} \\ -\begin{vmatrix} a_{1,2} & a_{1,3} \\ a_{3,2} & a_{3,3} \end{vmatrix} & +\begin{vmatrix} a_{1,1} & a_{1,3} \\ a_{3,1} & a_{3,3} \end{vmatrix} & -\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{3,1} & a_{3,2} \end{vmatrix} \\ +\begin{vmatrix} a_{1,2} & a_{1,3} \\ a_{2,2} & a_{2,3} \end{vmatrix} & -\begin{vmatrix} a_{1,1} & a_{1,3} \\ a_{2,1} & a_{2,3} \end{vmatrix} & +\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix} \end{bmatrix}^T$$

Below is a sample output:

Sample 1

```
a = np.array([[1,2],[3,4]])
y = [2, 5]

adjugate matrix:
array([[ 4., -2.],
       [-3.,  1.]])

x:
[1. , 0.5]
```

Sample 2

```
a = np.array([[1,2,3],[3,4,5],[3,5,5]])
y = [2, 2, 5]

adjugate matrix:
array([[-5.,  5., -2.],
       [-0., -4.,  4.],
       [ 3.,  1., -2.]])

x:
[-2.5,  3. , -0.5]
```