Computational Astrophysics HW3

Yi-Hsiang Kuo, 110022506

(Dated: Oct. 27, 2022)

+0.25 I. EXERCISE 2

(1) The equation of **vector norm (L_p-norms)**:

$$||x||_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}} \tag{1}$$

After calculation, we can get first and second norms 3.3 and 2.7 respectively

for **infinite norm**, applying:

$$||x||_\infty = max|x_i|, \quad 1 \leq i \leq n \tag{2}$$

and get the answer: 2.6.

(2) For the matrix A, applying the **matrix norm** formula:

$$||A|| = max_{x\neq 0}\frac{||Ax||}{||x||} \tag{3}$$

$$||A||_1 = max_j \sum_{i=1}^{n} |a_{ij}| \tag{4}$$

$$||A||_\infty = max_i \sum_{j=1}^{n} |a_{ij}| \tag{5}$$

max(10 ; 6 ; 8)=10

we can get the L1-norm (maximum absolute column sum):~~78~~, and the L_∞-norm (maximum absolute row sum):~~83~~ max(12;7;5)=12

+0.7 II. EXERCISE 3

To evaluate the real root(s) of:

$$x^3 + 1.5x^2 - 5.75x + 4.37 = 0 \tag{6}$$

You could change the trial value or initial guess when using three methods since the convergence rate should be Newton-Raphson > Secant > Bisection

Please attach your code at the end of the file. Thanks

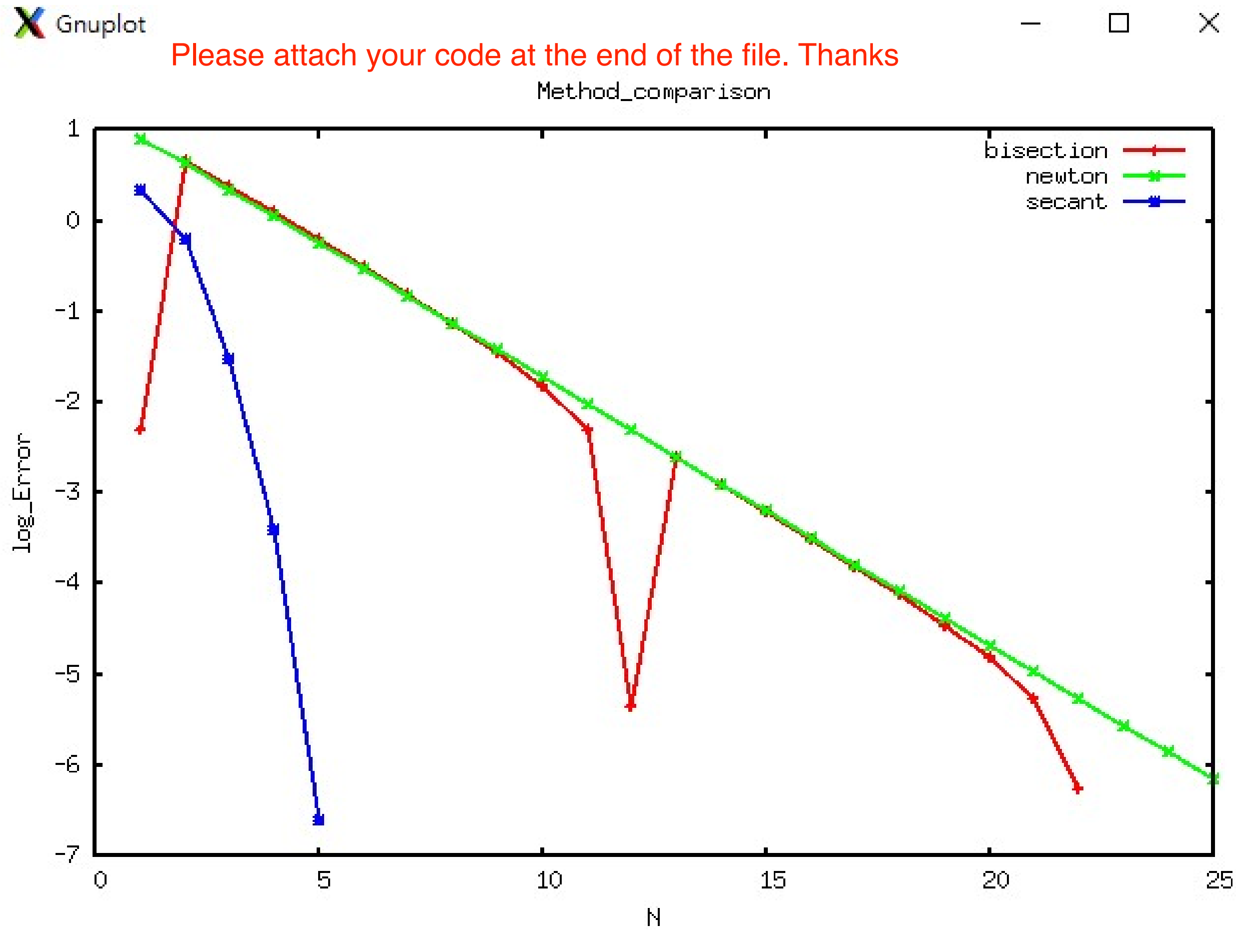

FIG. 1. Exercise3 Method comparison

using (a) bisection method with initial guess from -4.0 to -3.0; (b) Newton-Raphson's method with initial point $x = -3.0$ and (c) secant method with initial guess from -4.0 to -3.0, and FIG. 1. is the comparison result.

By the way, for Newton-Raphson's method, if our initial guess is $x = -3.5$, the iteration value N still need to approach $N = 14$, it seems that (c) secant method is the best way here.

+1.5 III. EXERCISE 4

(1)(2)(3) The analytical calculations wrote on FIG. 2. :

(1) $Ax = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 6 & 8 \\ 4 & 8 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 10 \end{bmatrix} = b$

step1 $\hookrightarrow$ $M_1 A x = M_1 b$, where $M_1 = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix}$

$M_1 A = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 4 & 6 & 8 \\ 4 & 8 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$, $M_1 b = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 6 \\ 10 \end{bmatrix} = \begin{bmatrix} 4 \\ -10 \\ -6 \end{bmatrix}$

reduced to : $Ux = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ -10 \\ -6 \end{bmatrix} \Rightarrow \begin{cases} x_1 = 0 \\ x_2 = 5 \\ x_3 = -3 \end{cases}$ #.

(2)

$L = M_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix}$, $U = \begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$

(3)

$LU = \begin{bmatrix} 1 & 0 & 0 \\ 4 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 2 \\ 4 & 6 & 8 \\ 4 & 8 & 10 \end{bmatrix} = A$ #

FIG. 2. Exercise4 Gaussian elimination

+2 IV. EXERCISE 5

(1)(2)(3) I have finished the subroutine *LU_decomposition* in *linalg.f*90 and the **git gub link is here**, put two different matrix A from in-class exercise and Exercise 4, the result is the following figures:

Next, further call *solve_lower&upper_triangular_matrix*, we can get the soltion x as the following figure, and the result indeed meets my answer in Exercise 4.

+2 V. EXERCISE 6

(1) I build up the code *Ex*6_1 **link here** to build up matrix A and using *linalg_lu.solve* to get the value x, see FIG. 6.

(2) using another method *scipy.linalg.solve*$_b$*anded* and the code *Ex*6_2 **link here**, we

```
A
   2.000    4.000   -2.000
   4.000    9.000   -3.000
  -2.000   -3.000    7.000


L
   1.000    0.000    0.000
   2.000    1.000    0.000
  -1.000    1.000    1.000


U
   2.000    4.000   -2.000
   0.000    1.000    1.000
   0.000    0.000    4.000
```

FIG. 3. Ex5-1 result

```
A
   1.000    2.000    2.000
   4.000    6.000    8.000
   4.000    8.000   10.000


L
   1.000    0.000    0.000
   4.000    1.000    0.000
   4.000   -0.000    1.000


U
   1.000    2.000    2.000
   0.000   -2.000    0.000
   0.000    0.000    2.000
```

FIG. 4. Ex5-2 result

```
A
  1.000   2.000   2.000
  4.000   6.000   8.000
  4.000   8.000  10.000

vector   b =    4.0000000000000000         6.0000000000000000         10.000000000000000
solution x =    0.0000000000000000         5.0000000000000000        -3.0000000000000000
```

FIG. 5. Ex5-3 result

can get same x as from Ex6-1.

For the computing time, I use *timeit* as my tool monitoring the running time and FIG. 7. is the result. LU decomposition costs about $1.74ms$, and bandmatrix method cost only about $542\mu s$

(3)By using *np.linalg.cond*, the condition number of matrix A is about 130661079.52047908

```
(compAstro) [yhkuo@fomalhaut EX6]$ python EX6_1.py
[[1.26250000e+03]
 [7.47500000e+03]
 [1.85385000e+04]
 [3.43550000e+04]
 [5.48275000e+04]
 [7.98600000e+04]
 [1.09357500e+05]
 [1.43226000e+05]
 [1.81372500e+05]
 [2.23705000e+05]
 [2.70132500e+05]
 [3.20565000e+05]
 [3.74913500e+05]
 [4.33090000e+05]
 [4.95007500e+05]
 [5.60580000e+05]
 [6.29722500e+05]
 [7.02351000e+05]
 [7.78382500e+05]
 [8.57735000e+05]
 [9.40327500e+05]
 [1.02608000e+06]
 [1.11491350e+06]
 [1.20675000e+06]
 [1.30151250e+06]
 [1.39912500e+06]
 [1.49951250e+06]
 [1.60260100e+06]
 [1.70831750e+06]
 [1.81659000e+06]
 [1.92734750e+06]
 [2.04052000e+06]
 [2.15603850e+06]
 [2.27383500e+06]
 [2.39384250e+06]
 [2.51599500e+06]
```

FIG. 6. Ex6-1 (a portion of) x

```
In [137]: runcell(0, 'C:/Users/user/untitled8.py')
542 µs ± 35.5 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
1.74 ms ± 73.6 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

FIG. 7. Ex6-2 Computing time