Assignment 4 of Computational Astrophysics in NTHU

Wei-Hsiang Yu 游惟翔

March 31, 2021

1 Written Assignments

Q1 : Vector Norms.

For the vector x $x = \begin{bmatrix} -1.6, & 1.2 \end{bmatrix}^T$ From the lecture we know that:

$$First\ norm : ||x||_1 = \sum_{i=1}^{n} |x_i| \qquad \text{(sum of each element)=2.8}$$

$$Second\ norm : ||x||_2 = (\sum_{i=1}^{n} |x_i|^2)^{\frac{1}{2}} \qquad \text{(square root the sum of each element does square)=2}$$

$$Infinity\ norm : ||x||_\infty = \max_{1 \le i \le n} |x_i| \qquad \text{(maximum element)=1.6}$$

Q2 : Matrix Norms.

For the matrix A

$$A = \begin{bmatrix} 7 & -3 & 2 \\ 1 & 1 & 5 \\ 2 & -2 & 1 \end{bmatrix}$$

From the lecture we know that:

$$First\ norm : ||A||_1 = \max_j \sum_{i=1}^{n} |a_{ij}| \qquad \text{(the maximum absolute **column** sum)=10}$$

$$Infinity\ norm : ||A||_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \qquad \text{(the maximum absolute **row** sum)=12}$$

Q3 : Matrix calculation.

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

3a. Gaussian elimination

Eliminate matrix A, we can use $4 \times first\ row$ to subtract the *third row*(step1), and also subtract the *second row* by $4 \times first\ row$ (step2).

$$\begin{bmatrix} 1 & 2 & 2 \\ 4 & 6 & 8 \\ 4 & 8 & 10 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 2 & 2 \\ 4 & 6 & 8 \\ 0 & 0 & 2 \end{bmatrix}_{step1} \Rightarrow \begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix}_{step2} = U$$

Finally, we can solve this equation.

$$\begin{bmatrix} 1 & 2 & 2 \\ 0 & -2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 10 \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 5 \end{bmatrix}$$

3b.&3c. LU factorization

See 3a. we have already gotten the Upper triangular matrix U. The lower triangular matrix L is described by the steps in 3a.(The subtracted steps using the first row can be viewed as improving of the unit matrix I, but opposite sign) And we also see the lecture that $A = LU$ So we finally get the LU factorization:

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

Q4 : Cholesky factorization.

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix}$$

Cholesky factorization has the feature that it can decompose a matrix into two triangular matrix L & U, which are transpose to each other($U = L^T$) Let's derive the solution of doing Cholesky factorization [1]:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = GG^T = \begin{bmatrix} g_{11} & 0 & 0 \\ g_{21} & g_{22} & g \\ g_{31} & g_{32} & g_{33} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} & g_{13} \\ 0 & g_{22} & g_{23} \\ 0 & 0 & g_{33} \end{bmatrix}$$

$$= \begin{bmatrix} g_{11}^2 & g_{21}g_{11} & g_{31}g_{11} \\ g_{21}g_{11} & g_{21}^2 + g_{22}^2 & g_{31}g_{21} + g_{32}g_{22} \\ g_{31}g_{11} & g_{31}g_{21} + g_{32}g_{22} & g_{11}^2 + g_{22}^2 + g_{33}^2 \end{bmatrix}$$

So, we can use this solution to find out the Cholesky factorization of the problem:

$$\begin{bmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ -\frac{1}{2} & \frac{\sqrt{15}}{2} & 0 \\ -\frac{1}{2} & -\frac{5}{2\sqrt{15}} & \sqrt{\frac{10}{3}} \end{bmatrix} \begin{bmatrix} 2 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{15}}{2} & -\frac{5}{2\sqrt{15}} \\ 0 & 0 & \sqrt{\frac{10}{3}} \end{bmatrix}$$

2 Programming Assignments

Q1 : Error of solving the system.

$$\begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1+\epsilon+\epsilon^2 \\ 1 \end{bmatrix}$$

(a) error when $\epsilon = 1e-11$

```
vector   b =    1.000000000010000000000099999999991          1.000000000000000000000000000000000
solution x =  0.99999999999940738530586462830973884          1.059261469412944554308469137117461936-0011
```

(b) error when $\epsilon = 1e-12$

```
vector   b =    1.000000000001000000000000999999999999          1.000000000000000000000000000000000000
solution x =    1.000000000001000000000000099999999999          0.000000000000000000000000000000000000
```

(c) error when $\epsilon = 1e-6$

```
vector   b =    1.000001000000999999999999999999999986          1.000000000000000000000000000000000000
solution x =  0.99999999999999999999999928457505109825          1.000000000000000071542423347655430806-0006
error of x0(1) =    7.154249489017507502432799330471779916-0023
error of x1(epsilon) =    7.154242334765543087167775983430516596-0017
```

Figure 1: error of ϵ

We change the matrix A to become the equation.4. in the assignment.
Noticed that when we declare the variable ϵ, we should make the precision of the variable type become **real*16**, then we can get a better result of our computer's calculation error.

We know the definition of condition number:

$$\text{cond(A)} = ||A||\dot{|}|A^{-1}|| \tag{1}$$

[1] Cholesky factorization:https://ccjou.wordpress.com/2010/09/16/cholesky-%E5%88%86%E8%A7%A3/

And we get $det(A) = \epsilon^2 \quad det(A^{-1}) = 1$, so the condition number of this system is ϵ^2. For my computer, the value of ϵ can't show when it's equal to 1e-12.

We are interested in $\sqrt{\epsilon_{mach}}$, I choose 1e-6 be the value of ϵ, and get the relative error of solution x in component 1 and ϵ(Fig.1(c)).

We know:

$$\frac{||\triangle x||}{||x||} \leq \text{cond(A)}\epsilon_{mach} \tag{2}$$

but here the relative error about ϵ is not obey Eq.2. I guess the max ϵ_{mach} is not equal value 1e-12 (maybe 5e-11...), so my computer can't get good result.

Q2 : Cholesky factorization.

In Fig.2(a)., the pseudo code supplied by the lecture is translated into the fortran code. Noticed that the output variable L don't need to be assigned at the second do loop, because it doesn't finish the process yet when program go to the second loop, the first loop or the moment when we calculate k,k position on the matrix will then get the output result of Cholesky factorization.

(a) Modification of linalg.f90

```
As=A
do k=1,N
    As(k,k)=As(k,k)**0.5
    L(k,k)=As(k,k)
    do i=k+1,N
        As(i,k)=As(i,k)/As(k,k)
        L(i,k)=As(i,k)
    enddo

    do j=k+1,N
        do i=k+1,N
            As(i,j)=As(i,j)-As(i,k)*As(j,k)
        enddo
    enddo
enddo
```

(b) the result of writing assignment problem 4

```
(base) kevinyu@惟翔の電腦:/mnt/d/Google可備份的檔案/清
r$ ./matrixinear

 cholesky_factorization of pro4 in writing assignment
   2.000   0.000   0.000
  -0.500   1.936   0.000
  -0.500  -0.645   1.826
```

Figure 2: Cholesky factorization exercise

The test of my program is showed in Fig.2(b). The output is the same as the result we get in the written assignments problem 4.

Q3 : Banded matrix.

The package of *scipy.linalg.solve_banded* need to change the banded matrix into the diagonal baned form. The following is the example of transforming a 7×7 banded matrix into the diagonal banded form:

$$\begin{bmatrix} 9 & -4 & 1 & 0 & 0 & 0 & 0 \\ -4 & 6 & -4 & 1 & 0 & 0 & 0 \\ 1 & -4 & 6 & -4 & 1 & 0 & 0 \\ 0 & 1 & -4 & 6 & -4 & 1 & 0 \\ 0 & 0 & 1 & -4 & 6 & -4 & 1 \\ 0 & 0 & 0 & 1 & -4 & 5 & -2 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} * & * & 1 & 1 & 1 & 1 \\ * & -4 & -4 & -4 & -4 & -2 \\ 9 & 6 & 6 & 6 & 5 & 1 \\ -4 & -4 & -4 & -4 & -2 & * \\ 1 & 1 & 1 & 1 & * & * \end{bmatrix}$$

In this example, there are 2 nonzero diagonal below and above the main diagonal, so we can set the variables l & u which package need to 2 (Fig.3(a). & Fig.3(b).).

And I also decide two methods to assign the diagonal banded form matrix. Fig.3(a). judge loop number every time to decide whether the variable is need to be assigned. Fig.3(b). will continue to execute next loop without doing another if-else judgement.

The time consumption of those three method is showed in Fig.4.

Q4 : UL factorization.

Verify UL factorization

In Fig.5., we first given a matrix R, and transpose it into matrix R^T, then we inner product these two matrix ,and judge whether it is equal to matrix A in writing assignment problem 4.

(a) Using solve_banded package-1

```python
def solver_bandmatrix1(A,b,N):
    ab=np.full((5,N), 0)
    for i in np.arange(N):
        if i+2<N:
            ab[0][i+2]=A[i][i+2]
            ab[4][i]=A[i+2][i]
        if i+1<N:
            ab[1][i+1]=A[i][i+1]
            ab[3][i]=A[i+1][i]
        ab[2][i]=A[i][i]

    x = scipy.linalg.solve_banded((2, 2), ab, b)

    return x
```

(b) Using solve_banded package-2

```python
def solver_bandmatrix2(A,b,N):
    ab=np.full((5,N), 0)
    for i in np.arange(N):
        if i<N-2:
            ab[0][i+2]=A[i][i+2]
            ab[1][i+1]=A[i][i+1]
            ab[2][i]=A[i][i]
            ab[3][i]=A[i+1][i]
            ab[4][i]=A[i+2][i]
            continue

        if i==N-2:
            ab[1][i+1]=A[i][i+1]
            ab[2][i]=A[i][i]
            ab[3][i]=A[i+1][i]
            continue

        elif i==N-1:
            ab[2][i]=A[i][i]
            continue

    x = scipy.linalg.solve_banded((2, 2), ab, b)

    return x
```

(c) Using LU decomposition package

```python
def LU_decompositon(A,b):
    lu, piv = scipy.linalg.lu_factor(A)
    x = lu_solve((lu, piv), b)
```

Figure 3: The three method to solve Banded matrix

```
%timeit solver_bandmatrix1(A,b,N)
575 µs ± 34.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
%timeit solver_bandmatrix2(A,b,N)
499 µs ± 27.7 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
%timeit LU_decompositon(A,b)
74.2 µs ± 813 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

Figure 4: Time spend by different package in python.

Solving the system

In this part, I use the method in problem3.(solver_bandmatrix2 Fig.2(b).) be the solution of using banded system solver.

And add the comparison: Cholesky factorization was applied by the package in scipy called *linalg.cho_factor*

The algorithm of my code is used by the code in fortran to transform, but noticed that: **the UL factorization has different order to solve compared to LU factorization!**

$$Ax = RR^T x = ULx = b \tag{3}$$
$$Uy = b$$
$$Lx = y$$

The error I derive by the following steps:
1.Inner product the matrix A with the solution x to get the experiment b.
2.Compare to expected b, get the gap between expected and experiment.
3.Finally, sum those error to get the displacement of the calculative error.

And we can see that the error generates by UL factorization has even no error.(Fig.6) but spend much more time(Fig.7.) The option of using package or self-coding seem a kind of trade-off.

Condition number of matrix A

I use the instruction of numpy called *linalg.cond* to find out the condition number. The output is 1.29e12, it means that this matrix can get pretty accurate solution.(Since larger value of condition number represents nearly singular.)

```
R=matrixR(N)

RT=R.transpose()

A4=R.dot(RT)
print(np.array_equal(A4,A))
True
```

Figure 5: Verify UL factorization.

```
X1=UL_factorization(R,RT,b,N)
b=np.full(N, 1)
X2=cho_factorization(A,b)
b=np.full(N, 1)
X3=solver_bandmatrix2(A,b,N)

acc1=np.dot(A, X1)
acc2=np.dot(A, X2)
acc3=np.dot(A, X3)

error1=np.abs(acc1-1)
error2=np.abs(acc2-1)
error3=np.abs(acc3-1)

accurate1=error1.sum()
accurate2=error2.sum()
accurate3=error3.sum()

print ("The error of using UL_factorization: %f" %accurate1)
print ("The error of using cho_factorization: %f" %accurate2)
print ("The error of using solver_bandmatrix2: %f" %accurate3)
The error of using UL_factorization: 0.000000
The error of using cho_factorization: 0.019947
The error of using solver_bandmatrix2: 0.015274
```

Figure 6: The accuracy of doing UL factorization by different package.

```
%timeit UL_factorization(R,RT,b,N)

/home/kevinyu/miniconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:18: RuntimeWarning: overflow encountered in long_scalars
/home/kevinyu/miniconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:10: RuntimeWarning: overflow encountered in long_scalars
  # Remove the CWD from sys.path while we load stuff.
750 ms ± 14.9 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

%timeit cho_factorization(A,b)

9.93 ms ± 112 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)

%timeit solver_bandmatrix2(A,b,N)

4.65 ms ± 200 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Figure 7: The time consumption of each algorithm.

```
np.linalg.cond(A)

1295526023618.2551
```

Figure 8: condition number of matrix A.