Assignment 1

成子浩 2018212327

1 Matrix Norm

a) **Norm-1** $||A||_1 = sup_x \frac{||Ax||_1}{||x||_1}$

$\because ||Ax||_1 = \sum_{i=1}^{m} |\sum_{j=1}^{n} a_{ij}x_j| \leqslant \sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}||x_j| \leqslant$
$(\sum_{j=1}^{n} |x_j|)(\max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{m} |a_{ij}|) = ||x||_1(\max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{m} |a_{ij}|)$

$\therefore ||A||_1 \leqslant \max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{m} |a_{ij}|$

Now we want to show that there exists some ξ such that

$$\frac{||A\xi||_1}{||\xi||_1} = \max_{1\leqslant j\leqslant n} \sum_{i=1}^{m} |a_{ij}|$$

Suppose $p = \arg\max\limits_{j} \sum\limits_{i=1}^{m} |a_{ij}|$. Let $\xi = (\xi_1, \xi_2, ..., \xi_n)^T$ where $\xi_j =$
$\begin{cases} 1, & j = p \\ 0, & otherwise \end{cases}$

Thus we have $||A||_1 = \frac{||A\xi||_1}{||\xi||_1} = \max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{m} |a_{ij}|$

Norm-Infinity $||A||_\infty = sup_x \frac{||Ax||_\infty}{||x||_\infty}$

$\because ||Ax||_\infty = \max\limits_{1\leqslant i\leqslant m} |\sum\limits_{j=1}^{n} a_{ij}x_j| \leqslant \max\limits_{1\leqslant i\leqslant m} \sum\limits_{j=1}^{n} |a_{ij}||x_j| \leqslant$

$\max\limits_{1\leqslant j\leqslant n} |x_j| \max\limits_{1\leqslant i\leqslant m} \sum\limits_{j=1}^{n} |a_{ij}| = ||x||_\infty \max\limits_{1\leqslant i\leqslant m} \sum\limits_{j=1}^{n} |a_{ij}|$

$\therefore ||A||_\infty \leqslant \max\limits_{1\leqslant i\leqslant m} \sum\limits_{j=1}^{n} |a_{ij}|$

Now we want to show that there exists some ξ such that

$$\frac{||A\xi||_\infty}{||\xi||_\infty} = \max_{1\leqslant i\leqslant m} \sum_{j=1}^{n} |a_{ij}|$$

Suppose $q = \arg\max_i \sum_{j=1}^{n} |a_{ij}|$. Let $\xi = (\xi_1, \xi_2, ..., \xi_n)^T$ where $\xi_j = \begin{cases} 1, & a_{qj} > 0 \\ -1, & otherwise \end{cases}$

Thus we have $||A||_\infty = \frac{||A\xi||_\infty}{||\xi||_\infty} = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{n} |a_{ij}|$.

b) **Frobenius norm and 2 norm**

Suppose $A = V\Sigma U^T$, $A^T A = U\Sigma^2 U^T$ where Σ is the eigen value matrix of A while U and V are orthogonal matrices. Let $\sigma(k)$ denote the k_{th} largest eigen value of A.

$$\|A\|_F = (\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2)^{\frac{1}{2}} = (tr(A^T A))^{\frac{1}{2}} = (tr(U\Sigma^2 U^T))^{\frac{1}{2}}$$

$$= (tr(\Sigma^2))^{\frac{1}{2}} = (\sum_{i=1}^{p} \sigma^2(i))^{\frac{1}{2}}$$

where $p = min(m, n)$

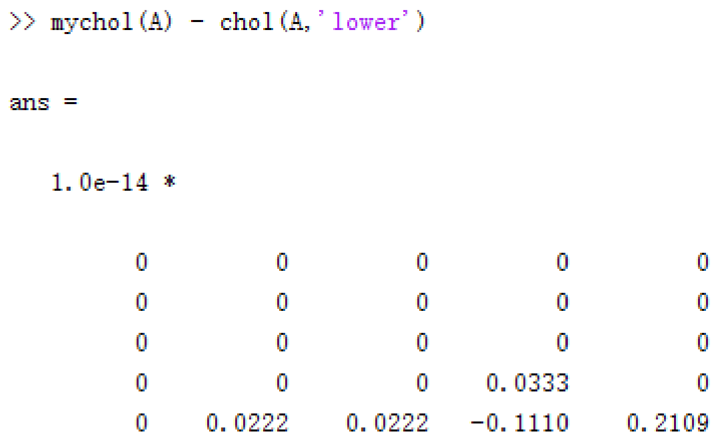
While $\|A\|_2 = \sigma(1)$ according to the definition. It is obvious that $\|A\|_2 \leqslant \|A\|_F \leqslant \sqrt{p}\|A\|_2$.

2 norm and 1 norm

Suppose $x \in \mathbb{R}^n$. $\|x\|_1 = \sum_{i=1}^n |x_i|$, $\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}$.
Obviously, $\|x\|_2^2 = \sum_{i=1}^n |x_i|^2 \leqslant (\sum_{i=1}^n |x_i|)^2 = \|x\|_1^2$. Thus we have $\|x\|_2 \leqslant \|x\|_1$
Since $f(x) = x^2$ is a convex function, according to $Jenson's\ inequality$,
$(\frac{1}{n}\sum_{i=1}^n |x_i|)^2 \leqslant \frac{1}{n}\sum_{i=1}^n |x_i|^2 \Rightarrow \|x\|_1 \leqslant \sqrt{n}\|x\|_2$
To sum up, we have $\|x\|_2 \leqslant \|x\|_1 \leqslant \sqrt{n}\|x\|_2$

2 Cholesky Decomposition

a) Code is in mychol.m, the result is as following:



```
>> mychol(A) - chol(A,'lower')

ans =

   1.0e-14 *

         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0    0.0333         0
         0    0.0222    0.0222   -0.1110    0.2109
```

There exists some differences. I guess they may be caused by Round-off error since I didn't take error analysis into consideration. Instead, I just implement the basic algorithm at this step.

b) To solve this problem, I designed a 'randomly generating' process. More specifically, we run $n!$ times (Because there are totally $n!$ kinds of line translation of a $n * n$ matrix) line translation randomly upon the Unit Matrix and take every of them as the pivoting matrix. Then we try to find the most sparse result after implementing those pivoting matrix upon the origin matrix. Although we didn't run through every line translation (Similar to boosting method in Machine Learning when selecting training samples from datasets randomly), we can find a relatively sparse matrix using this method. The result is as following:

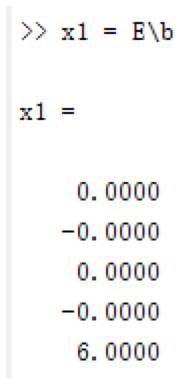
```
>> [L,P]=myspchol(A);
>> L

L =

    2.0000         0         0         0
         0    2.0000         0         0
         0         0    2.0000         0
    0.5000    0.5000    0.5000    1.8028

>> P

P =

     0     1     0     0
     0     0     0     1
     0     0     1     0
     1     0     0     0
```

3 Linear System

Firstly, we generate matrix E. Then we solve the problem $Ex = b$ using different methods respectively.

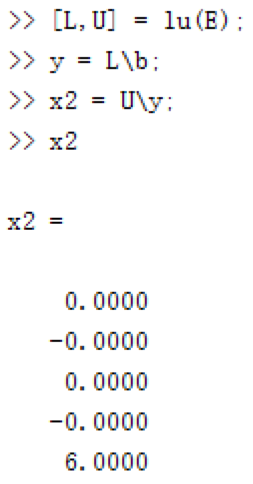
```
E =

    0.8333    0.6667    0.5000    0.3333    0.1667
    0.6667    1.3333    1.0000    0.6667    0.3333
    0.5000    1.0000    1.5000    1.0000    0.5000
    0.3333    0.6667    1.0000    1.3333    0.6667
    0.1667    0.3333    0.5000    0.6667    0.8333
```

a) Using \ operator



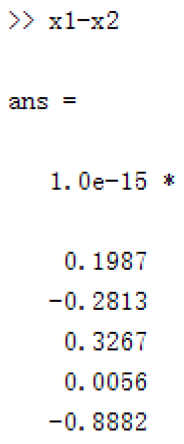
```
>> x1 = E\b

x1 =

    0.0000
   -0.0000
    0.0000
   -0.0000
    6.0000
```

b) Using LU decomposition



```
>> [L,U] = lu(E);
>> y = L\b;
>> x2 = U\y;
>> x2

x2 =

    0.0000
   -0.0000
    0.0000
   -0.0000
    6.0000
```

Difference Between 2 Methods



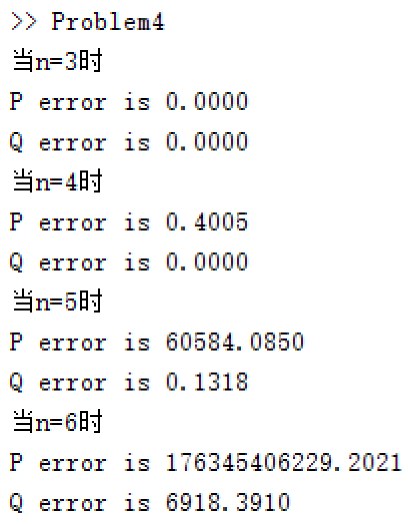
```
>> x1-x2

ans =

   1.0e-15 *

    0.1987
   -0.2813
    0.3267
    0.0056
   -0.8882
```

4 Inverse

According to the question, I computed the exact inverse of $A^T A$ as following way:

$$R = invhilb(k) * invhilb(k)'$$

where k denotes the dimension of the matrix. And the result is as following:

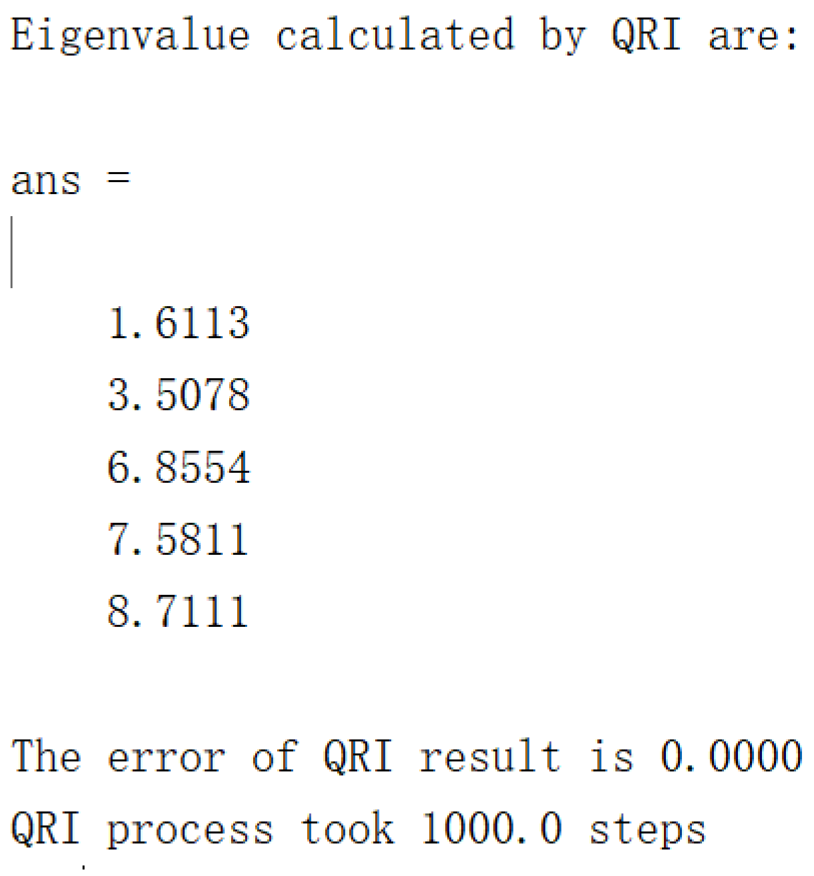


```
>> Problem4
当n=3时
P error is 0.0000
Q error is 0.0000
当n=4时
P error is 0.4005
Q error is 0.0000
当n=5时
P error is 60584.0850
Q error is 0.1318
当n=6时
P error is 176345406229.2021
Q error is 6918.3910
```

5 QR Iteration and Orthogonal Iteration

We try to find p leading eigenvalues using 2 iteration algorithms on the same randomly generated matrix separately, the result is as follwing:

```
创建方阵维度:5
Eigenvalue randomly generated are:

lambda =

    1.6113
    3.5078
    6.8554
    7.5811
    8.7111



 Eigenvalue calculate by OI are:

 ans =

    1.6113
    3.5078
    6.8554
    7.5811
    8.7111

 The error of OI result is 0.0000
 OI process took 166.0 steps
```



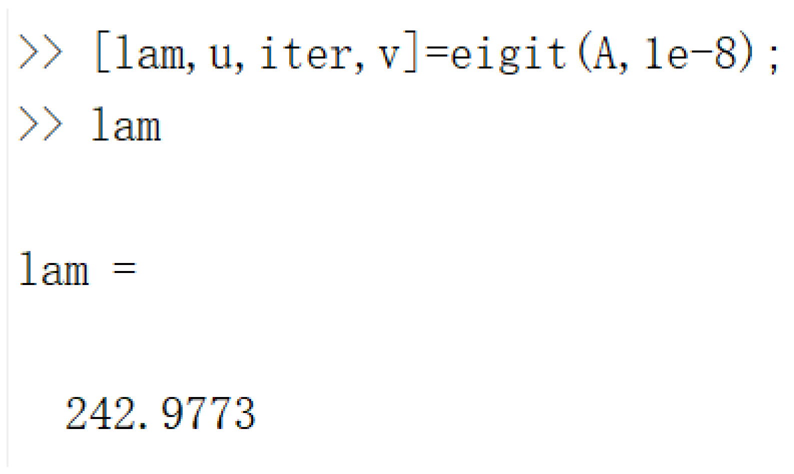
```
Eigenvalue calculated by QRI are:

ans =

    1.6113
    3.5078
    6.8554
    7.5811
    8.7111

The error of QRI result is 0.0000
QRI process took 1000.0 steps
```

The result shows that the 2 algorithms produce the same answer, thus they are equivalent when $p = n$.

6 Eigen Problems

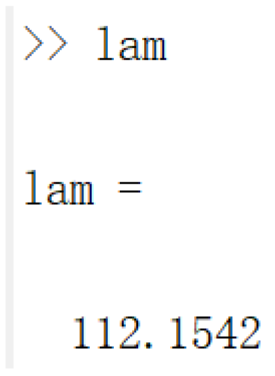
In this problem, we let $tol = 1e - 8$, which is a parameter of the function to control the accuracy of the error. And the result is as following.

a) We use 'eigit' function to find the biggest eigen value. The biggest eigenvalue is 242.9773



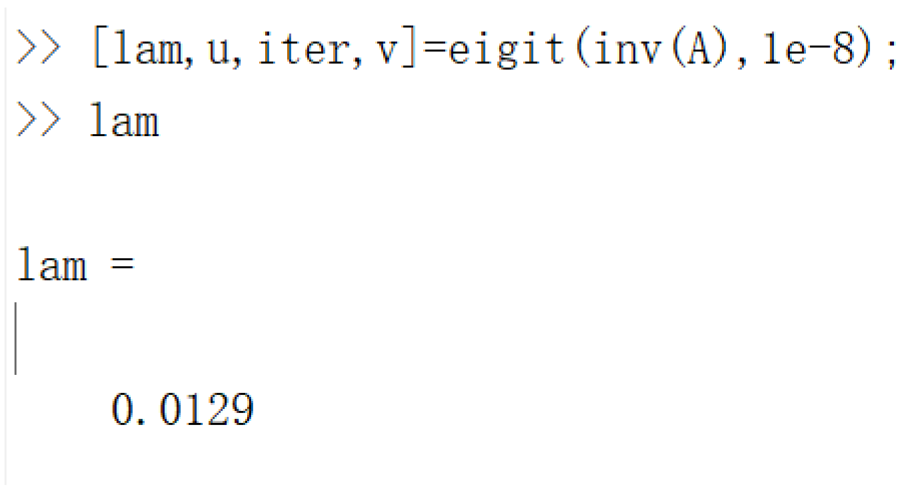
```
>> [lam,u,iter,v]=eigit(A,1e-8);
>> lam

lam =

  242.9773
```

b) We use 'eiginv' function to find the eigenvalue nearest 100. The eigenvalue nearest 100 is 112.1542



```
>> lam

lam =

  112.1542
```

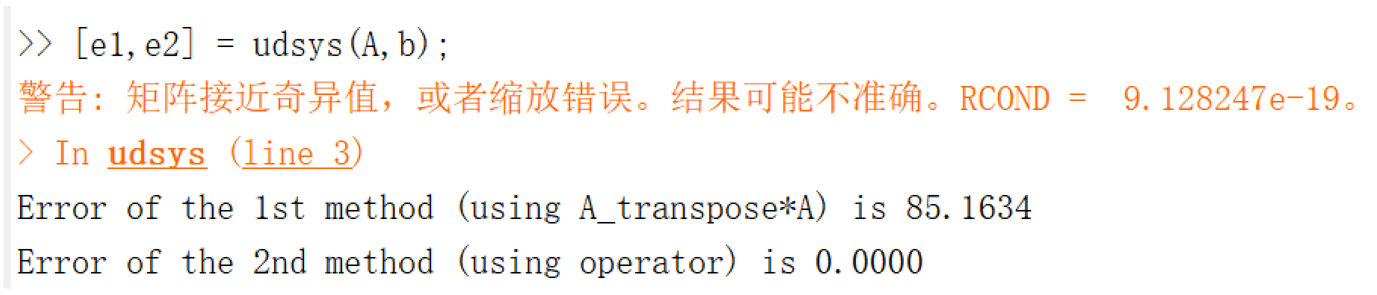
c) We use 'eigit' function to find the smallest eigen value while we use the inverse of A as the input. The smallest eigenvalue is 0.0129



```
>> [lam,u,iter,v]=eigit(inv(A),1e-8);
>> lam

lam =

    0.0129
```

7 Underdetermined Systems

Given A as the question mentioned, $A^T A$ is not a positive definite matrix, which means that it is not invertible. So use *inv* method may result in wrong answer as following:



```
>> [e1,e2] = udsys(A,b);
警告: 矩阵接近奇异值，或者缩放错误。结果可能不准确。RCOND =  9.128247e-19。
> In udsys (line 3)
Error of the 1st method (using A_transpose*A) is 85.1634
Error of the 2nd method (using operator) is 0.0000
```