APPLIED LINEAR ALGEBRA HOMEWORK 4

TOMMENIX YU
STAT 31430
DUE FRIDAY, NOV. 12, 3PM

Exercise 1. *(4.3)*

Let A and B be square matrices of $\mathcal{M}_n(\mathbb{R})$, and u a vector of $\mathbb{R}^n$.

(a) If A is a band matrix (see Definition 6.2.1), compute the computational complexity for computing Au (assuming n large) in terms of the half-bandwidth p and of n.
(b) If A and B are two band matrices, of equal half-bandwidths p, prove that the product AB is a band matrix. Find the computational complexity for computing AB.

Proof.

(a):

Let $Au =: (v_1, \dots, v^n)^T$ denote the output of the matrix vector computation. Then we know

$$v_i = \sum_{j=1}^{n} a_{ij} u_j$$

since u is dense, the only place where this could be simplified is when $a_{ij} = 0$. This means we only count the case when $a_{ij} \neq 0$, which means we've converted our problem into a problem of finding how many non-zero entries does A have, as each one corresponding to a multiplication with one entry of u. (Note that in the expression above no entry in A is counted repeatedly.)

So since there is a main diagonal with n entries and the $k-th$ offdiagonal

$$a_{1k}, a_{2,k+1}, \dots, a_{n-k,n}$$

has $n-k$ entries, and there are 2 of such for each $1 \geq k \leq p$, we get that there are in total

$$n + 2\left(\frac{[(n-1)+(n-p)]\,p}{2}\right) = (2p+1)n - p^2 - p$$

non-zero entries of A, hence the complexity is $(2p+1)n - p^2 - p = (2p+1)n + O(p^2)$ for $p \ll n$.

(b):

Let $C := AB = (c_{ij})$, then the matrix multiplication formula tells us

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}.$$

But $a_{ik} = 0$ if $|i-k| > p$ and $b_{kj} = 0$ if $|j-k| > p$, and thus for $n \gg p$, if $|i-j| > 2p$ then at least one of $|i-k|$ and $|j-k|$ is larger than p for all k, so

$$c_{ij} = 0 \quad \text{for } |i-j| > 2p$$

which means that C is a band matrix with half bandwidth $2p$.

In particular, even if A has half-bandwidth p_1, B has half-bandwidth p_2, $C = AB$ is still a band-matrix of half-bandwidth $p_c \leq 2\max\{p_1, p_2\}$.

Now back to the same half-bandwidth case, we notice that after eliminating all cases where $|i-k| > p$ or $|j-k| > p$, the expression of c_{ij} becomes

$$c_{ij} = \sum_{k \in S(i,j)} a_{ik} b_{kj}$$

where

$$S(i,j) := \{k : |i-k| \leq p \text{ or } |j-k| \leq p\}.$$

Under this expression, every multiplication is necessary. But we note that each a_{ik} corresponds to 1 multiplication in every c_{ij} where $k \in S(i,j)$, and that each multiplication is of some entry in A and some entry in B. Plus, there's no division.

Therefore, we know that for each $a_{ik} \neq 0$, it appeared in $|J(k)|$ terms of C where

$$J(k) = \{j : |j-k| \leq p, 1 \leq j \leq n\}.$$

Plus, for each k there's $|I(k)|$ non-zero entries a_{ik} in A, where

$$I(k) = \{i : |i-k| \leq p, 1 \leq i \leq n\}.$$

So the complexity of $C = AB$ is nothing but to count for each k, how many counted a_{ik} there were, and how much times each is multiplied, which gives:

$$\text{complexity}(C = AB) = \sum_{k=1}^{n} I(k)J(k)$$

where we note that

$$I(k) = J(k) = \begin{cases} p+k & 1 \leq k \leq p \\ 2p+1 & p+1 \leq k \leq n-p \\ p+(n-k) & n-p+1 \leq k \leq n \end{cases}$$

and plugging in we have

$$
\begin{aligned}
\text{complexity}(C = AB) &= \sum_{k=1}^{n} I(k)J(k) \\
&= 2\left((p+1)^2 + \cdots + (2p)^2\right) + (n-2p)(2p+1)^2 \\
&= 2\left(\sum_{i=1}^{2p} i^2 - \sum_{i=1}^{p} i^2\right) + (n-2p)(2p+1)^2 \\
&= 2\left(\frac{2p(2p+1)(4p+1)}{6} - \frac{p(p+1)(2p+1)}{6}\right) + (n-2p)(2p+1)^2 \\
&= \frac{14p^3 + 9p^2 + 1p}{3} + n(2p+1)^2 - 2p(2p+1)^2 \\
&= n(2p+1)^2 - \frac{10p^3 - p^2 - 3p}{3} = (2p+1)^2 n + O(p^3).
\end{aligned}
$$

□

Exercise 2. *(5.6)*

We define a matrix $A = [1 : 5; 5 : 9; 10 : 14]$.

(1) Compute a matrix Q whose columns form a basis of the null space of A^t .

(2)

(a) Consider $b = [5; 9; 4]$ and the vector $x \in \mathbb{R}^5$ defined by the instruction $x = A\backslash b$. Compute x, $Ax - b$, and $Q^t b$.

(b) Same question for $b = [1; 1; 1]$. Compare both cases.

(c) Justification. Let A be a real matrix of size mxn. Let $b \in \mathbb{R}^m$. Prove the equivalence

$$b \in \mathrm{Im}(A) \iff Q^T b = 0.$$

(d) Write a function InTheImage(A, b) whose input arguments are a matrix A and a vector b and whose output argument is "yes" if $b \in \mathrm{Im}(A)$ and "no" otherwise. Application:

$$A = [123; 456; 789], b = [1; 1; 1] \text{ and } b = [1; 2; 1].$$

Proof.

(1):

We can simply find Q by the code null(A.'), which yields:

```
Q =

    0.4527
   -0.8148
    0.3621
```

(2):

(a): Doing the test yields:

x	$Ax - b$	$Q^T b$

`x =`
 	
`-1.8443`
`0`
`0`
`0`
`1.6967`
	`Ax-b=`
	
	`ans =`
	
	`1.6393`
	`-2.9508`
	`1.3115`
		`Q^T b`
		
		`ans =`
		
		`-3.6214`

(b): Doing the test yields:

x	$Ax - b$	$Q^T b$

`x =`
 	
`-0.2500`
`0`
`0`
`0`
`0.2500`
	`Ax-b=`
	
	`ans =`
	
	`1.0e-15 *`
	
	`-0.2220`
	`-0.2220`
	`-0.4441`
		`Q^T b`
		
		`ans =`
		
		`8.8818e-16`

Note that the $Ax - b = 0$ already implies that a best fit x is found.

(c):

($\Rightarrow$): If $b \in \text{Im}(A)$, then $\exists x$ such that $b = Ax$, and hence

$$Q^T b = Q^T Ax = (x^T (A^T Q))^T = (x^T * O)^T = 0.$$

($\Leftarrow$): On Oct 19th's lecture we've covered the result $\ker(A^T) = \text{Col}(A)^{\perp}$, where $\text{Col}(A)$ is the image of the column space of A, which is exactly $\text{Im}(A)$.

Since the columns of Q^T spans the whole $\ker(A^T)$, and $Q^T b = 0$ means that b is perpendicular to each column of Q, since each row of $Q^T b$ is the inner product of b and a column of Q^T. So either $b = 0$ or $b \perp \ker(A^T) \Rightarrow b \in \text{Im}(A)$ by above discussion, but either way $b \in \text{Im}(A)$.

(d): The code is

```
function T = InTheImage(A, b)
%This returns whether b = Ax for some x using the above method.
Q = null(A.');
tol = 0.001;
if Q.'*b < tol
    T = "yes";
else
    T = "no";
end
end
```

And since I realized that even though $(1, 2, 1)$ is not in the image of A (not in kernel either), the error is 0.81 that is quite big. But that's a genuine result of matlab. I used $(1, -2, 1) \in \ker(A^T)$ to test the no case, which works well. Also, maybe I should use norm of $Q^T b <$ tolerance, but since the dimension of kernel is just 1, I didn't write that for this case.

```
b =

     1
     1
     1


answer =

    "yes"
```

```
b =

     1
     2
     1


answer =

    "yes"
```

```
b =

     1
    -2
     1


answer =

    "no"
```

□

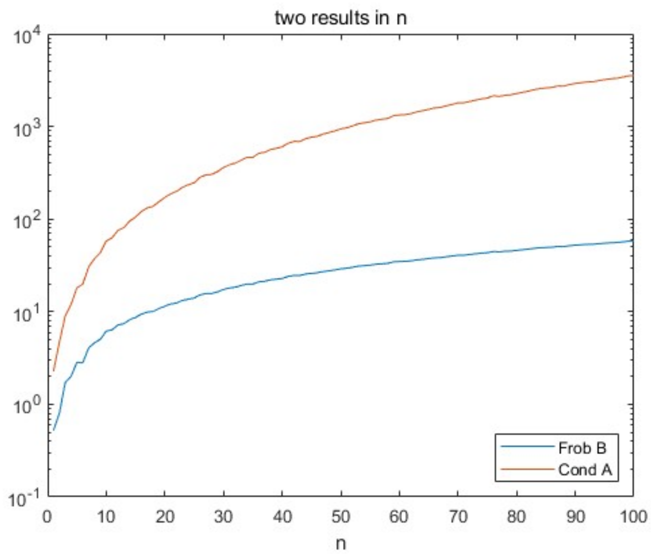
Exercise 3. *(5.8) Let A and B be two matrices defined by the instructions*

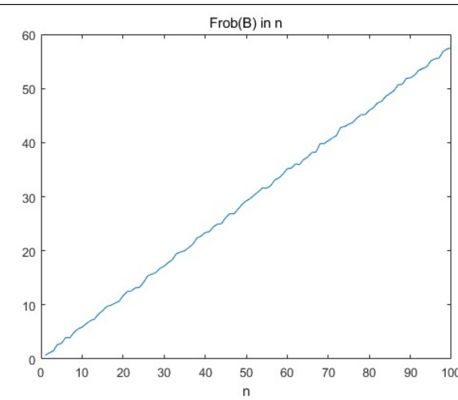
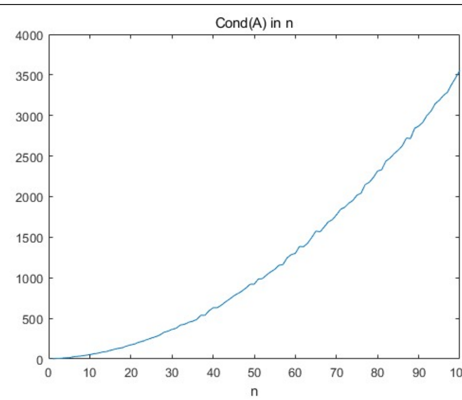
```
n=10;B=rand(n,n);A=[eye(size(B)) B; zeros(size(B)) eye(size(B))];
```

Compute the Frobenius norm of B as well as the condition number of A (in the Frobenius norm). Compare the two quantities for various values of n. Justify the observations.

Solution:

So the graph I get for various n is the following:



$\|\|B\|\|_{\mathcal{F}}$	Cond(A)
	

So as we can see, the Frobinius norm grow linearly while the conditional number grows quadratically.

This is natural since all entries of B is random, so assuming that each entry is iid we get

$$||B||_{\mathcal{F}} = \left(\sum_{i=1}^{n^2} X_i^2\right)^{\frac{1}{2}} = \left(n^2 X_1^2\right)^{\frac{1}{2}}$$

and by taking expectations on both sides we get

$$\mathbb{E}\left[||B||_{\mathcal{F}}\right] = \mathbb{E}\left[\left(n^2 X_1^2\right)^{\frac{1}{2}}\right] = n\mathbb{E}[X_1]$$

where $\mathbb{E}[X_1]$ is just a constant. So the result should be linear.

Now, let's look at A, the definition of it means

$$A = \begin{pmatrix} I & B \\ 0 & I \end{pmatrix} \quad \text{and} \quad A^{-1} = \begin{pmatrix} I & -B \\ 0 & I \end{pmatrix}$$

either by analogy to 2 by 2 square matrices or just by computation. Hence,

$$||A||_{\mathcal{F}} = \left(\sum_{i=1}^{n^2} X_i^2 + 2n\right)^{\frac{1}{2}} = ||B||_{\mathcal{F}} \left(1 + \frac{2n}{||B||_{\mathcal{F}}^2}\right)^{\frac{1}{2}} = ||B||_{\mathcal{F}} + ||B||_{\mathcal{F}} \frac{\frac{2n}{||B||_{\mathcal{F}}}}{2} + O()$$

where since by Taylor we have

$$(1+x)^{\frac{1}{2}} = 1 + \frac{x}{2} - \frac{x^2}{8} + O\left(x^3\right)$$

we get

$$||A||_{\mathcal{F}} = ||B||_{\mathcal{F}} + ||B||_{\mathcal{F}} \frac{\frac{2n}{||B||_{\mathcal{F}}^2}}{2} + O\left(\frac{1}{n}\right) = ||B||_{\mathcal{F}} + \frac{n}{||B||_{\mathcal{F}}} + O\left(\frac{1}{n}\right)$$

since $||B||_{\mathcal{F}} = O(n)$, which is also why we can use Taylor in the first place.

And since Frobinius norm doesn't care about the sign of each term, we get

$$||A^{-1}||_{\mathcal{F}} = ||A||_{\mathcal{F}}$$

and hence

$$\text{Cond}(A) = ||A||_{\mathcal{F}} ||A^{-1}||_{\mathcal{F}} = ||B||_{\mathcal{F}}^2 + O(||B||_{\mathcal{F}}) = O(n^2)$$

and the trend is indeed quadratic.

Exercise 4. *(5.9)*

The goal of this exercise is to empirically determine the asymptotic behavior of $\operatorname{cond}_2(H_n)$ *as* $n \to \infty$*, where* $H_n \in \mathcal{M}_n(\mathbb{R})$ *is the Hilbert matrix of order n, defined by its entries* $(H_n)_{ij} = 1/(i+j-1)$*. Compute* $\operatorname{cond}_2(H_5)$, $\operatorname{cond}_2(H_10)$*. What do you notice? For n varying from 2 to 10, plot the curve* $n \mapsto \ln(\operatorname{cond}_2(H_n))$*. Draw conclusions about the experimental asymptotic behavior.*

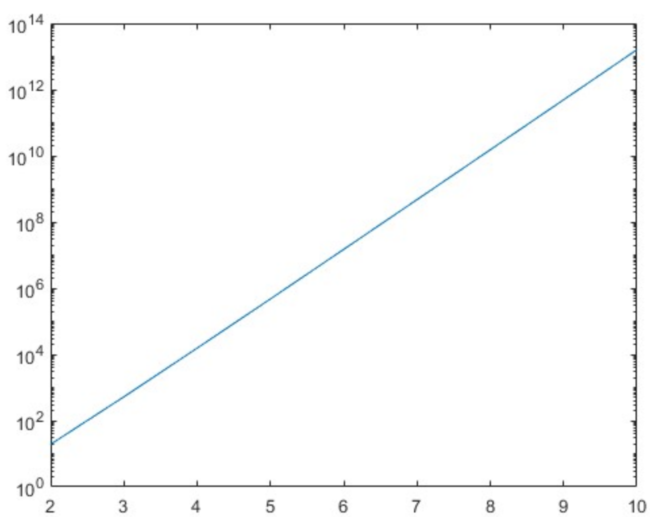
Solution:

Note that the question just asks us to draw conclusion of the empirical data, not deduce it, so I'll not.

The special values are

`n =`
 `5`
 `ans =`
 `4.7661e+05`
	`n =`
	 `10`
	 `ans =`
	 `1.6025e+13`

and the curve I get is



So OK I admit that this is not genuinely against $\ln(\operatorname{cond}_2(H_n))$, but I drew a semilogy plot so the genuine plot is just against the power on 10 on the y-label.

Thus, we know $\operatorname{cond}_2(H_n)$ grows exponentially in n from the graph.

Exercise 5. *(5.14)*

We define n by n matrices C, D, and E by

```
C=NonsingularMat(n);D=rand(m,n);E=D*inv(C)*D';
```

We also define $(n+m)\times(n+m)$ block matrices A and M by

```
A=[C D';D zeros(m,m)];M=[C zeros(n,m);zeros(m,n) E];
```

(1) For different values of n, compute the spectrum of $M^{-1}A$. What do you notice?
(2) What is the point in replacing system $Ax = b$ by the equivalent system $M^{-1}Ax = M^{-1}b$?
(3) We now want to give a rigorous explanation of the numerical results of the first question. We assume that $A \in \mathcal{M}_{m+n}(\mathbb{R})$ is a nonsingular matrix that admits the block structure $A = \begin{pmatrix} C & D^T \\ D & 0 \end{pmatrix}$ where $C \in \mathcal{M}_n(\mathbb{R})$ and $D \in \mathcal{M}_{m,n}(\mathbb{R})$ are such that C and $DC^{-1}D^T$ are non-singular too.
   (a) Show that the assumption "A is nonsingular" implies $m \le n$.
   (b) Show that for $m = n$, the matrix D is invertible.
(4) From now on, we assume $m < n$. Let $x = (x_1, x_2)^T$ be the solution of the system $Ax = b = (b_1, b_2)^T$. The matrix D is not assumed to be invertible, so that we cannot first compute x_1 by relation $Dx_1 = b_2$, then x_2 by $Cx_1 + D^T x_2 = b_1$. Therefore, the relation $Dx_1 = b_2$ has to be considered as a constraint to be satisfied by the solutions x_1, x_2 of the system $Cx_1 + D^T x_2 = b_1$. We study the preconditioning of the system $Ax = b$ by the matrix M^{-1} with $M = \begin{pmatrix} C & 0 \\ 0 & DC^{-1}D^T \end{pmatrix}$.
   (a) Let λ be an eigenvalue of $M^{-1}A$ and $(u, v)^T \in \mathbb{R}^{m+n}$ a corresponding eigenvector. Prove that $(\lambda^2 - \lambda - 1)Du = 0$.
   (b) Deduce the spectrum of the matrix $M^{-1}A$.
   (c) Compute the 2-norm conditioning of $M^{-1}A$, assuming that it is a symmetric matrix.

Solution:

(1):

I tested for $m = n$ and the results says that all eigenvalues of the matrix is either 1.6180 or -0.6180.

Then I tested for $m \neq n$ and the result is that in general

- If $m = n$, the eigenvalues of the matrix is either 1.6180 or -0.6180. Moreover there are m of each.

- If $m > n$, there's a warning of singular matrices, and the eigenvalues contains 0, and other numbers for which I've found no pattern.
- If $m < n$, there are $n - m$ eigenvalues that is 1, and the rest $2m$ eigenvalues have half being 1.6180 and half -0.6180.

```
n =

     5


E =

   1.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 - 0.0000i
  -0.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 - 0.0000i
   1.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
```

```
m =

     4


n =

     9


E =

  -0.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 - 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.0000 + 0.0000i
   1.0000 - 0.0000i
   1.0000 + 0.0000i
   1.0000 - 0.0000i
   1.0000 + 0.0000i
```

```
m =

     7


n =

     9


E =

  -0.6180 + 0.0000i
  -0.6180 - 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 - 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 - 0.0000i
```

```
m =

    10


n =

     5

警告: 矩阵接近奇异值，或者缩放
> 位置: g514 (第 9 行)


E =

   0.5000 + 2.5265i
   0.5000 - 2.5265i
   1.9913 + 0.0000i
   1.9009 + 0.0000i
   1.5652 + 0.3838i
   1.5652 - 0.3838i
  -0.9913 + 0.0000i
  -0.9009 + 0.0000i
  -0.5652 + 0.3838i
  -0.5652 - 0.3838i
  -0.0000 + 0.0000i
  -0.0000 - 0.0000i
  -0.0000 + 0.0000i
  -0.0000 + 0.0000i
   0.0000 + 0.0000i
```

```
m =

     5


n =

    10


E =

   1.6180 + 0.0000i
   1.6180 - 0.0000i
   1.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 - 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
  -0.6180 + 0.0000i
   1.6180 + 0.0000i
   1.6180 + 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
   1.0000 + 0.0000i
```

```
m =

     7


n =

     3

警告: 矩阵接近奇异值，或者缩放不[
> 位置: g514 (第 9 行)


E =

   2.6641 + 0.4423i
   2.6641 - 0.4423i
   1.9519 + 0.0000i
  -1.6641 + 0.4423i
  -1.6641 - 0.4423i
  -0.9519 + 0.0000i
  -0.0000 + 0.0000i
  -0.0000 - 0.0000i
   0.0000 + 0.0000i
  -0.0000 + 0.0000i
```

(2):

The main reason is to when A is bad (in the sense of ill-conditioned, too dense, etc) then it's possible to get a much easier result computationally to solve the preconditioned system.

(3):

(a): I show the contrapositive: $m > n \quad \Rightarrow A \quad$ is singular.

But this is easy. Since D is m by n and thus we can at least do Gaussian Elimination to let the first n rows span the whole $\mathbb{R}^n$, which means we can easily do row operations to D to eliminate the last $m-n$ lines.

But we can do the same row operation on the $n+1$ to $m+n$ lines on A to let the last $m-n$ lines equal to 0. So $\det(A) = 0$ and A is singular.

(b): If D is not invertible, then $\det(DC^{-1}D^T) = 0 \cdot \det(C) \cdot 0 = 0$, which contradicts the assumption that it is non-singular. So D is invertible.

Note that the condition $m = n$ is really just because we say only square matrices can be invertible.

(4):

(a): We've proven that for a block matrix with only diagonal terms non-zero, the inverse is just by inverting all block squares. So by definition of M and A, we can compute

$$M^{-1}A = \begin{pmatrix} C^{-1} & 0 \\ 0 & (DC^{-1}D^T)^{-1} \end{pmatrix} \begin{pmatrix} C & D^T \\ D & 0 \end{pmatrix} = \begin{pmatrix} I & C^{-1}D^T \\ (DC^{-1}D^T)^{-1}D & 0 \end{pmatrix}$$

and using the spectrum condition we get

$$\begin{pmatrix} I & C^{-1}D^T \\ (DC^{-1}D^T)^{-1}D & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u + C^{-1}D^T v \\ (DC^{-1}D^T)^{-1}Du \end{pmatrix} = \begin{pmatrix} \lambda u \\ \lambda v \end{pmatrix}$$

and from the second line we get

$$v = \frac{1}{\lambda}(DC^{-1}D^T)^{-1}Du$$

plugging into the first line we end with

$$(\lambda - 1)u = C^{-1}D^T v = C^{-1}D^T \frac{1}{\lambda}(DC^{-1}D^T)^{-1}Du$$

$$\Rightarrow (\lambda^2 - \lambda)u = C^{-1}D(DC^{-1}D^T)^{-1}Du$$

$$\Rightarrow (\lambda^2 - \lambda)Du = ((DC^{-1}D^T))(DC^{-1}D^T)^{-1}Du = Du$$

hence we get

$$(\lambda^2 - \lambda - 1)Du = 0.$$

(b): From last result we get that either $Du = 0$ or $(\lambda^2 - \lambda - 1) = 0$.

Suppose $Du = 0$, then $A \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} Cu + D^T v \\ Du \end{pmatrix} = \begin{pmatrix} Cu + D^T v \\ 0 \end{pmatrix}$ which, if we left multiply by M^{-1} it gives us $v = 0$ since M^{-1} is a diagonal matrix (so really by computation).

But then $(\lambda - 1)u = C^{-1}D^T v = 0$, so either $u = 0$ or $\lambda = 1$.

If $u = 0$, then the eigenvector is 0, so impossible. Thus, $\lambda = 1$.

Let's check that for all $u \in \ker(D)$, $(u, 0)^T$ is an eigenvector of $M^{-1}A$.

But this is by computation:

$$\begin{pmatrix} u + C^{-1}D^T v \\ (DC^{-1}D^T)^{-1}Du \end{pmatrix} = \begin{pmatrix} u \\ 0 \end{pmatrix}.$$

So we get that if $Du = 0$, then the eigenvalue is 1 with multiplicity dim(ker(D)) (since otherwise the eigenvalues are not linearly independent).

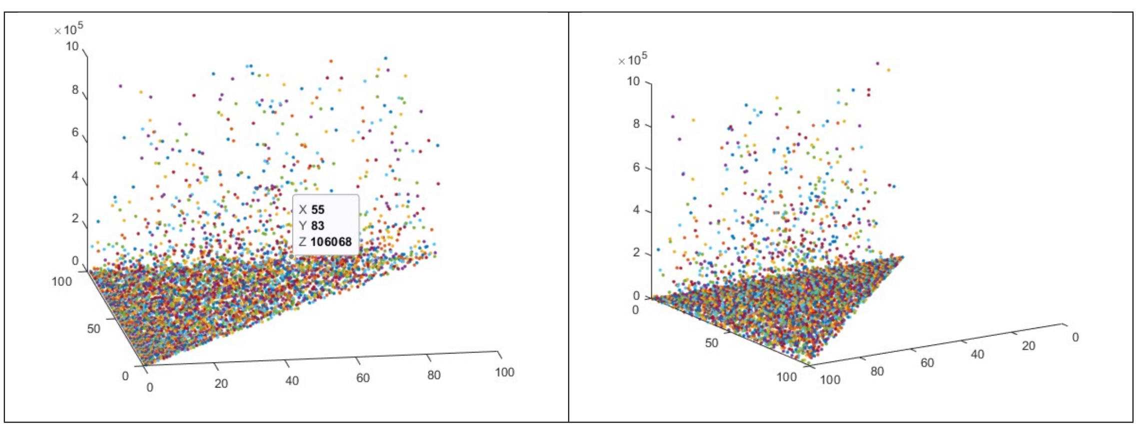
As for when $(\lambda^2 - \lambda - 1) = 0$, we simply get

$$\lambda_1 = \frac{1 - \sqrt{5}}{2} \approx -0.618; \quad \lambda_2 = \frac{1 + \sqrt{5}}{2} \approx 1.618.$$

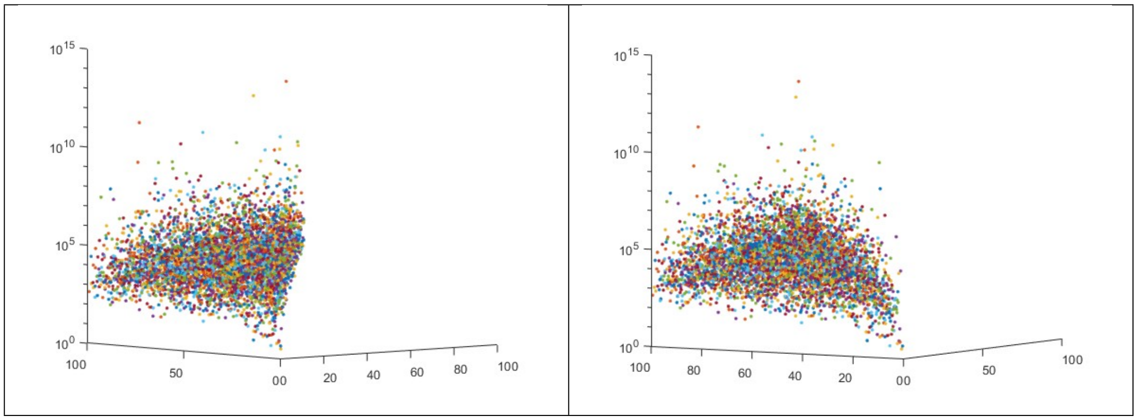
And that they are evenly distributed is simply because by our computation above, whenever we get one there's another solution of $(u, v)^T$ with the other eigenvalue.

Therefore, we've reached the exact same conclusion as our observation in (1).

(c): So I write up a little plot that reflects the behavior of the conditional numbers in m and n:

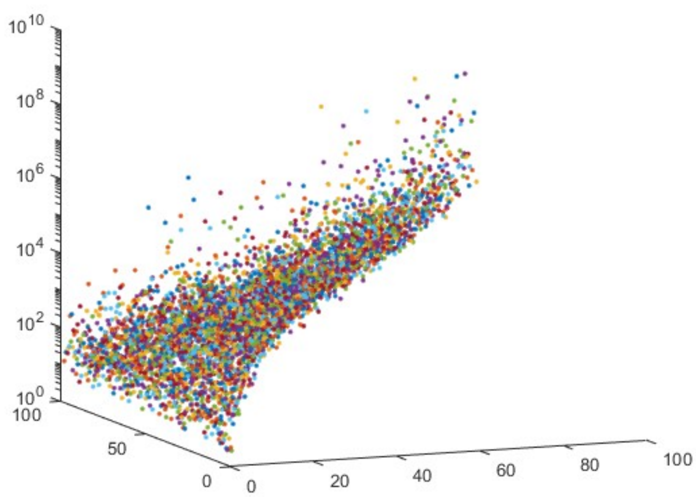


(I neglected outliers that's too big that the graph is not meaningful) Which I notices that it's quite flat comparatively, so I then plotted the log-scale graph:



So I guess most of the dsitribution is around 10^2 to 10^7.

And I also did for symmetric matrix, which gives this smoother surface I guess:



Exercise 6.

The goal of this exercise is to compare the performances of the LU and Cholesky methods.

(1) Write a function LUfacto returning the matrices L and U determined via Algorithm 6.1. If the algorithm cannot be executed (division by 0), return an error message.
(2) Write a function Cholesky returning the matrix B computed by Algorithm 6.2. If the algorithm cannot be executed (nonsymmetric matrix, division by 0, negative square root), return an error message. Compare with the Matlab function chol.
(3) For $n = 10, 20, \ldots, 100$, we define a matrix A =MatSdp(n) (see Exercise 2.20) and a vector $b = ones(n, 1)$. Compare:
   - On the one hand, the running time for computing the matrices L and U given by the function LUFacto, then the solution x of the system $Ax = b$. Use the functions BackSub and ForwSub defined in Exercise 5.2.
   - On the other hand, the running time for computing the matrix B given by the function Cholesky, then the solution x of the system $Ax = b$. Use the functions BackSub and ForwSub.

   Plot on the same graph the curves representing the running times in terms of n. Comment.

Solution:(1): The code is below:

```
function A = LUfacto(A)
%This function returns A containing U and L but its diagonal
sz = size(A);
tol = 0.000000000001;
n = sz(1);
for k = 1: n-1
    for i = k+1: n
        if abs(A(k,k)) < tol
            error("cannot divide by 0");
            return
        else
            A(i,k) = A(i,k)/A(k,k);
            for j = k+1: n
                A(i,j) = A(i,j)-A(i,k)*A(k,j);
            end
        end
    end
end
end
```

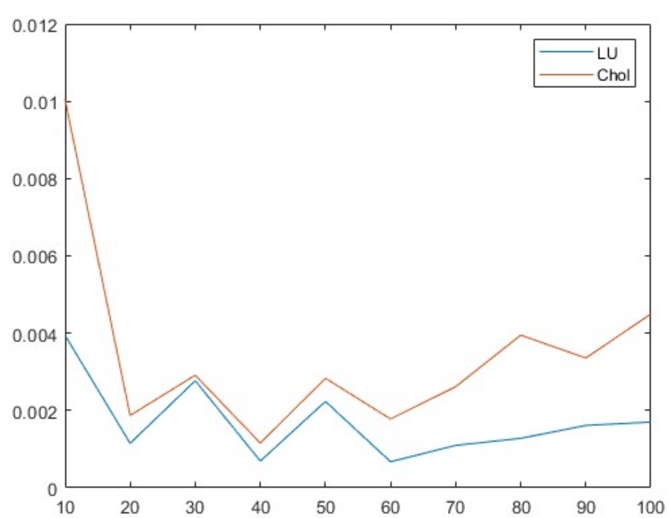
(2):

```
function A = Cholesky(A)
% A containing B in its lower triangular part
sz = size(A);
tol = 0.0000000001;
n = sz(1);
if norm(A-A.') > tol
    error("A is not symmetric");
    return
end

for j = 1:n
    for k = 1: j-1
        A(j,j) = A(j,j) - (A(j,k))^2;
    end
    if A(j,j)<0
        error("cannot take square root of negative numbers");
        return
    elseif A(j,j)< tol
        error("Cannot divide by 0");
        return
    end
    A(j,j) = sqrt(A(j,j));
    for i = j+1:n
        for k = 1:j-1
            A(i,j) = A(i,j) -A(j,k)*A(i,k);
        end
        A(i,j) = A(i,j)/A(j,j);
    end
end
end
```

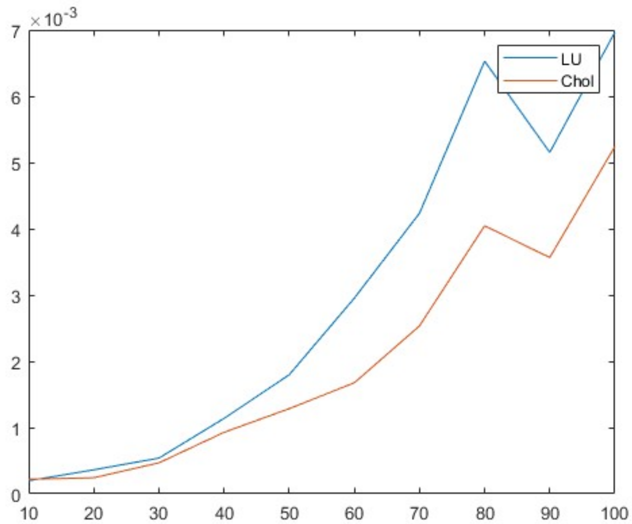
(3):

```
 A = [4 1 1; 1 2 3; 1 3 5];
 B = LUfacto(A);
C = Cholesky(A);
nl = [];
T_LU = [];
T_Chol = [];
for i = 1:10
    n = 10*i;
    nl(i) = n;
    b = ones(n,1);
    A = MatSdp(n);
    tic
    B = LUfacto(A);
    % This ForwSubL really uses the fact that L has diagonal 1
    x1 = ForwSubL(B,b);
    x2 = Backsub(B, x1);
    T_LU(i) = toc;
    tic
    C = Cholesky(A);
    x1 = ForwSub(C,b);
    x2 = Backsub(C.', x1);
    T_Chol(i) = toc;
end

plot(nl,T_LU)
hold all
plot(nl,T_Chol)
legend("LU","Chol")
%
% f = @() myComputeFunction(x,y); % handle to function
% timeit(f)

```

And the run time is:



This may seem a little weird because the beginning is quite large. But this is only because that was the first time I called the functions in this file. So I add a few code that does all the functions once, and afterwards the curve becomes



which makes a lot more sense.

We can see that For larger matrices, it's less painful to do Cholesky factorization.

Exercise 7.

Suppose that an $n \times n$ matrix $A \in \mathcal{M}_n(\mathbb{R})$is invertible and has an LU factorization (and that we have already computed and stored it). It is often practically important to be able to re-solve the system of equations when the matrix A is "updated" by a rank-one matrix uv^T (with $u, v \in \mathbb{R}^n$) to obtain a system

$$(A + uv^T)x = b.$$

(a) Recall that we have assumed that A is invertible. Show that $A + uv^T$ is invertible if and only if $\langle v, A^{-1}u\rangle \neq -1$.

(b) When $(A + uv^T)$ is invertible (i.e. when the condition in (a) is satisfied), one has

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1 + \langle v, A^{-1}u\rangle}$$

This identity is often attributed to Sherman and Morrison 1 and is related to more general matrix identities due to M. Woodbury (see also W. Hager, Updating the inverse of a matrix, SIAM Review 31 (1989), no. 2, 221–239). Verify this formula by computing the product of $A + uv^T$ with the given matrix.

(c) Suppose that all of the (principal) diagonal submatrices of A are nonsingular, and suppose that we are given an LU factorization for A. Use the Sherman–Morrison formula from (b) to formulate an efficient algorithm for solving

$$(A + uv^T)x = b.$$

Proof. (a): I follow the block matrix hint and will show that

Lemma 0.1.

$$\det(I + xy^T) = 1 + x^T y$$

Proof. This is done by block matrix operation where each $\sim$ represent some row/col operation, where if I use index i, I mean for all $i = 1, 2, \dots n$ **AND NOT FOR** $(\mathbf{n+1})$:

$$\begin{pmatrix} I + xy^T & 0 \\ 0 & 1 \end{pmatrix} \underbrace{\quad}_{r_i = r_i + x_i * r_{n+1}}^{r_i = r_i + x_i * r_{n+1}} \begin{pmatrix} I + xy^T & x \\ 0 & 1 \end{pmatrix} \overset{c_i = c_i - y_i * c_{n+1}}{\sim} \begin{pmatrix} I & x \\ -y^T & 1 \end{pmatrix}$$

$$\overset{c_{n+1} = c_{n+1} - x_i * c_i}{\sim} \begin{pmatrix} I & 0 \\ -y^T & 1 + x^T y \end{pmatrix}$$

and by taking the determinant on both sides we get

$$\det(I + xy^T) = 1 + \langle x, y\rangle$$

□

Now back to the question. Let $A + uv^T = A(I + (A^{-1}u)v^T)$ we know that

$$\det(I + (A^{-1}u)v^T) = 1 + \langle A^{-1}u, v\rangle$$

which is 0 if and only if $\langle v, A^{-1}u\rangle \neq -1$ by above lemma.

But since A invertible $\det(A) \neq 0$ so

$$A + uv^T \text{ invertible} \iff \det(I + (A^{-1}u)v^T) \neq 0 \iff \langle v, A^{-1}u\rangle \neq -1$$

and so we are done.

(b): This equality is actually encoded in the thrid matrix of the above deduction of lemma in (a), but still let's check it:

$$\begin{aligned}\left(A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + \langle v, A^{-1}u\rangle}\right)(A + uv^T) &= I + A^{-1}uv^T - \frac{A^{-1}uv^T + A^{-1}u(v^TA^{-1}u)v^T}{1 + \langle v, A^{-1}u\rangle} \\ &= I + A^{-1}uv^T\left(\frac{(1 + \langle v, A^{-1}u\rangle) - (1 + (v^TA^{-1}u))}{1 + \langle v, A^{-1}u\rangle}\right) \\ &= I + A^{-1}uv^T * 0 = I\end{aligned}$$

(c): We can just use $U^{-1}L^{-1}$ to compute A^{-1} and then plug in to the formula, then right multiply by b to solve the system. Since inverting triangular matrix requires $O(n^2)$ mult/div, the whole process is dominated by this procedure and thus the whole complexity is $O(n^2)$. The code is here:

```
function x = SM(L,U,u,v,b)
%This function uses the formula to get the solution x
Ainv = inv(U)*inv(L);
fml = Ainv-(Ainv*u*v.'*Ainv)/(1+v.'*Ainv*u);
x = fml*b;
end
```

□