Muhideen Ogunlowo Math 426 Hw5

Exercise 2.6.1

Assuming A is an nxn matrix and b is a column vector of compatible length. Given the function [L,U]= lu(A), we can find the LU factorization of the matrix A. However:

x= $U\backslash(L\backslash b)$ solves the code because Matlab solves $Lz = b$ in parentheses first using forward substitution and it solves the upper triangular Ux=z system using backward substitution. However if we used $U\backslash L\backslash b$ instead, MATLAB solves $U\backslash L$ first which does not provide a correct solution as they are both triangular matrices. That incorrect result will then be used to complete the operation, leading to incorrect solutions.

Exercise 2.6.8 a)

$p = e_3e_1^T + e_2e_2^T + e_4e_3^T + e_1e_4^T$

$$= \begin{bmatrix}0\\0\\1\\0\end{bmatrix}[1 \quad 0 \quad 0 \quad 0] + \begin{bmatrix}0\\1\\0\\0\end{bmatrix}[0 \quad 1 \quad 0 \quad 0] + \begin{bmatrix}0\\0\\0\\1\end{bmatrix}[0 \quad 0 \quad 1 \quad 0] + \begin{bmatrix}1\\0\\0\\0\end{bmatrix}[0 \quad 0 \quad 0 \quad 1]$$

$$= \begin{bmatrix}0 & 0 & 0 & 1\\0 & 1 & 0 & 0\\1 & 0 & 0 & 0\\0 & 0 & 1 & 0\end{bmatrix}$$

Exercise 2.6.8b

p^{-1} moves to $i_1 \to 1, i_2 \to 2, i_3 \to 3 \ldots\ldots i_n \to n$

Using the general formula,

$p^{-1} = e_1e_{i_1}^T + e_2e_{i_2}^T + v + e_3e_{i_3}^T\ldots\ldots e_ne_{i_n}^T$

$= \left(e_1e_{i_1}^T\right)^T + \left(e_2e_{i_2}^T\right)^T + \left(e_3e_{i_3}^T.\right)^T\ldots\ldots\left(e_ne_{i_n}^T\right)^T$

$= \left(e_1e_{i_1}^T + e_2e_{i_2}^T + v + e_3e_{i_3}^T\ldots\ldots e_ne_{i_n}^T\right)^T = p^T$

Exercise 2.6.101

```
A=magic(5); %magic 5 matrix
y=ones(5,1); %makes the diagonals 1
b=A*y; %b as a product of matrix A and matrix y
[L,U,P]=lu(A); %LU factorization
x = U \ (L \ (P * b)); %solves system of Linear Equations for x using LU
factorization
```

```
resError = norm(A * x - b) %residual error
```

```
resError =
     1.421085471520200e-14
```

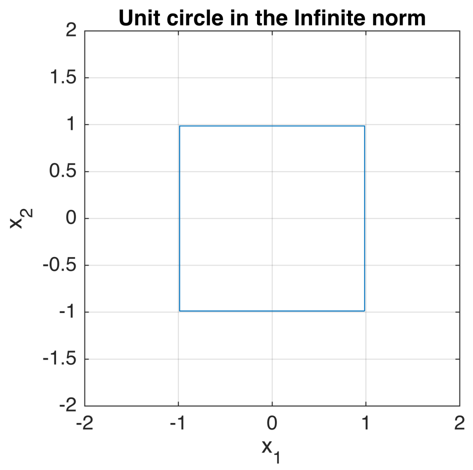
```
absError = norm(x - y) %absolute error
```

```
absError =
     5.978733960281817e-16
```

```
%both absolute error and residual error are very small so we can say the
answer is good
%row switching happened because P is not an identity matrix, rather a
permutation matrix
```

Exercise 2.7.2a

```
p = 2^100000 ; %large values of p that are close to infinity, representing
the norm
for j = 1:length(p) %for loop
    f = @(x,y) abs(x).^p(j) + abs(y).^p(j) - 1; %equation of unit circle
    fimplicit(f, [-2, 2, -2, 2]) %plots unit circle in specified range
end
grid on, %grid added
xlabel('x_1'), ylabel('x_2')
axis square %Square axis
title({'Unit circle in the Infinite norm'})% Title
```



Exercise 2.7.3a

Given a matrix A of any size, we can prove that the infinite-norm is always less than or equal to it's two-norm. The infinite norm of a matrix, is the value of the highest element in the matrix while the two-norm can be defined as the magnitude or the square root of the sum of all squares of every element in the matrix. We can prove this by doing the folowing examples:

```
v=[1,-2,3]; %Sample matrix V
twonorm= norm(v) %2-norm of v
```

```
twonorm =
   3.741657386773941
```

```
infnorm= norm(v,inf) %Infinite norm of v
```

```
infnorm =
     3
```

```
q=[2,3,5] %sample matrix q
```

```
q = 1×3
     2     3     5
```

```
twonorm2= norm(q)%Two-norm
```

```
twonorm2 =
   6.164414002968977
```

```
infnorm2= norm(q,inf) %Infinite-norm
```

```
infnorm2 =
     5
```

```
p=[3,0;1,2]; %sample matrix p
twonorm3=norm(p) %two-norm p
```

```
twonorm3 =
   3.256616537982941
```

```
infnorm3= norm(p,inf)% infinite norm p
```

```
infnorm3 =
     3
```

By using the following examples we can confirm that the infinite norm is smaller than or equal to the two norm given $x \in \mathbb{R}^n$

Exercise 2.7.4 a

Assume X is a 2 matrix [2,0

```
X=[2,0;2,3];%sample matrix X and Y
Y=[1,2;3,4];
```

```
B= X'.*Y %B= xtranspose * y
```

```
B = 2×2
     2     4
     0    12
```

```
C= norm(X,1)*norm(Y,inf) %C=product of 1-norm of X and infinite norm of Y
```

```
C = 
    28
```

```
Q=norm(B) %norm of xtranspose * y
```

```
Q = 
  12.665274696415024
```

By comparison, we can see that Q(norm of the product of X transpose and Y) is smaller than or equal to the C (product of one-norm of X and infinite norm of Y)

This holds true for another set of compatible 2X2 matrix examples

```
X=[4,0;3,4]; %sample matrix X and Y
Y=[3,5;7,9];
J= norm(X'*Y) %J= norm(xtranspose * y)
```

```
J = 
  73.314062679756191
```

```
U= norm(X,1)*norm(Y,inf) %U=product of 1-norm of X and infinite norm of Y
```

```
U = 
   112
```

By comparison, we can see that J(norm of the product of X transpose and Y) is smaller than or equal to U (product of one-norm of X and infinite norm of Y)

Therefore, for any vectors $x, y \in \mathbb{R}^n$ product of one-norm of X and infinite norm of Y is smaller than or equal to product of one-norm of X and infinite norm of Y

Exercise 2.7.8

A permutation matrix is a square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere. I will find the two norm of each nXn permutation matrix using matlab and determine it is 1

```
A=[0,1,0;1,0,0;0,0,1] % 3 x 3 permutation matrix
```

```
A = 3×3
     0     1     0
     1     0     0
     0     0     1
```

```
B=norm(A) %norm of permutation matrix
```

```
B = 
     1
```

```
R=[0,0,0,1;0,0,1,0;0,1,0,0;1,0,0,0] % 4 x 4 permutation matrix
```

```
R = 4×4
     0     0     0     1
     0     0     1     0
     0     1     0     0
     1     0     0     0
```

```
U=norm(R) %norm of permutation matrix
```

```
U =
     1
```

Using the matlab function, we are able to determine that the two-norm of a permutation matrix is = 1. This is because after changing rows around (which changes the sign of the determinant) a permutation matrix becomes I, whose determinant is one.

Exercise 2.8.2

```
n=(10:10:70)'; %list number in interval of 10 from 10 to 70
%RelError, T, Error B are initialized to store the relative error,
condition number and error bounds for each matrix n
RelError = zeros(length(n),1);
T= zeros(length(n),1);
ErrorB= zeros(length(n),1);

for i=1:length(n) %For loop iterating over values of n
    A= gallery('prolate',n(i),0.4); %generating matrix A with a prolate
structure and parameter of 0.4
    x=(1/n(i)).*(1:n(i))'; %This vectorrepresents exact solution to the
linear system
    b= A*x; %computes right hand side vector b
    x_approximate= A\b; %computes an approximate solution
    RelError(i)= norm(x-x_approximate)/norm(x);
    T(i)=cond(A); %computes condition number for matrix A
    ErrorB(i)=T(i)*eps; %computes error bound where eps = machine epsilon
end
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
1.568682e-16.
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
3.743965e-17.
```

```
K= table(n,RelError,T,ErrorB) %Table of n, Relative Error, Condition
numbers, and error bounds
```

K = 7×4 table

	n	RelError	T	ErrorB
1	10	0.0000000000...	56.612333970793...	0.0000000000...
2	20	0.0000000000...	2.078733333015710e+04	0.0000000000...
3	30	0.0000000000...	9.801735508756941e+06	0.0000000021...

	n	RelError	T	ErrorB
4	40	0.0000001001...	4.983408555512708e+09	0.0000011065...
5	50	0.0000344796...	2.632994479469430e+12	0.0005846422...
6	60	0.0014116973...	2.294371469823370e+15	0.5094528065...
7	70	0.4385586085...	1.350955009964403e+16	2.9997227145...

Exercise 2.8.3

```
beta = logspace(1, 12, 12)'; %array spaced logarithmically between 10 and
10^12
alpha = 0.1; %set constant value of alpha = 0.1
b = [alpha; 0; 0; 0; 1];
x_backsub_diff = zeros(size(beta));
x_bslash_diff = zeros(size(beta));
infnormA = zeros(size(beta));
condN= zeros(size(beta)); %condition number
%matrix set up
for i = 1:length(beta)
    beta_el = beta(i);
    A = eye(5) - diag(ones(4, 1), 1);
    A(1, [4, 5]) = [alpha - beta_el, beta_el];


    infnormA(i) = norm(A,inf); %infinite norm of each matrix for beta
    condN(i)= cond(A,inf); %condition number of infinite norm for each
matrix of beta

    xx = backsub(A, b);
    xxx = A\b;

    x_backsub_diff(i) = abs(xx(1) - 1);
    x_bslash_diff(i) = abs(xxx(1) - 1);
end
table(beta,infnormA, condN) %table
```

```
ans = 12×3 table
```

	beta	infnormA	condN
1	10	21.899999999999...	3.241200000000000e+02
2	100	2.019000000000000e+02	2.115912000000000e+04
3	1000	2.001900000000000e+03	2.011509120000000e+06
4	10000	2.000190000000000e+04	2.001150091200000e+08
5	100000	2.000019000000000e+05	2.000115000912000e+10
6	1000000	2.000001900000000e+06	2.000011500009120e+12
7	10000000	2.000000190000000e+07	2.000001150000091e+14

	beta	infnormA	condN
8	100000000	2.000000019000000e+08	2.000000115000001e+16
9	1.000000000000000e+09	2.000000001900000e+09	2.000000011500000e+18
10	1.000000000000000e+10	2.000000000190000e+10	2.000000001150000e+20
11	1.000000000000000e+11	2.000000000019000e+11	2.000000000115000e+22
12	1.000000000000000e+12	2.000000000001900e+12	2.000000000011500e+24

Exercise 2.9.1

A matrix is said to be diagonally dominant if the diagonal element of every row is greater or equal to the sum of the non-diagonal elements of the same row

```
A=[3,1,0,1;0,-2,0,1;-1,0,4,1;0,0,0,6]; %Matrix from textbook
B=[1,0,0,0,0;0,1,0,0,0;0,0,1,0,0;0,0,0,1,0;0,0,0,0,0];%Matrix from textbook
C=[2,-1,0,0;-1,2,-1,0;0,-1,2,-1;0,0,-1,2];%Matrix from textbook
diag_main=diag(A,0)%Using diagonalization function
```

```
diag_main = 4×1
     3
    -2
     4
     6
```

For A, each diagonal element is greater than or equal to the sum of the non-diagonal elements of the same row

```
diag_main2=diag(B,0)%Using diagonalization function
```

```
diag_main2 = 5×1
     1
     1
     1
     1
     0
```

For B, each diagonal element is greater than or equal to the sum of the non-diagonal elements of the same row

```
diag_main3=diag(C,0)%Using diagonalization function
```

```
diag_main3 = 4×1
     2
     2
     2
     2
```

For C, each diagonal element is greater than or equal to the sum of the non-diagonal elements of the same row

Question 2.9.2

```
A=[1,0,-1; 0,4,5; -1,5,10];
B=[1,0,1; 0,4,5; -1,5,10];
C=[1,0,1; 0,4,5; 1,5,1];
try chol(A) %Uses Cholesky method to determine if it is SPD
```

```
    disp('Matrix is symmetric positive definite.') % displays 'Matrix is
symmetric positive definite.' if true
catch ME %Otherwise displays 'Matrix is not symmetric positive definite' if
false
    disp('Matrix is not symmetric positive definite')
end
```

```
ans = 3×3
   1.000000000000000                   0  -1.000000000000000
                   0   2.000000000000000   2.500000000000000
                   0                   0   1.658312395177700
Matrix is symmetric positive definite.
```

```
try chol(B) %Uses Cholesky method to determine if it is SPD
    disp('Matrix is symmetric positive definite.') % displays 'Matrix is
symmetric positive definite.' if true
catch ME %Otherwise displays 'Matrix is not symmetric positive definite' if
false
    disp('Matrix is not symmetric positive definite')
end
```

```
ans = 3×3
   1.000000000000000                   0   1.000000000000000
                   0   2.000000000000000   2.500000000000000
                   0                   0   1.658312395177700
Matrix is symmetric positive definite.
```

```
try chol(C) %Uses Cholesky method to determine if it is SPD
    disp('Matrix is symmetric positive definite.')% displays 'Matrix is
symmetric positive definite.' if true
catch ME %Otherwise displays 'Matrix is not symmetric positive definite' if
false
    disp('Matrix is not symmetric positive definite')
end
```

```
Matrix is not symmetric positive definite
```