ME 594 - Numerical Methods – HW03

Viral Panchal | Due Date: 10/12

‘I pledge my honor that I have abided by the stevens honor system’

VIRAL PANCHAL
HW03.

Q.1) Inverse using Gaussian elimination:

Process:

4x4 (Given)

1. Augmented matrix $[A|I]$
2. $A_{4\times4} \rightarrow$ upper triangular form
   ↳ Partial pivoting to be considered
3. Back substitution to get the solution

(Program attached)

Q1. Programs

- ***Program to compute inverse of a square matrix:***

```
% Making a function to get inverse of a matrix using Gaussian elimination
(considering pivoting)

function A_inv = get_inv(A)

[p,q] = size(A);

if (p ~= q)
    fprintf('Error: Given matrix is not a suqare matrix \n');
    return
end

A_aug = [A eye(p)];         % Augmented matrix

for i = 1:p-1
    [max_value, r] = max(abs(A_aug(i:p,i)));
    if (r>1)                            % pivoting condition
        temp_row = A_aug(i,:);
        A_aug(i,:) = A_aug(i+r-1,:);
        A_aug(i+r-1,:) = temp_row;
    end

    for j = i+1:p
        pivot = A_aug(j,i)/A_aug(i,i);
        A_aug(j,:) = A_aug(j,:)-pivot.*(A_aug(i,:));
    end
end

% Checking if matrix in not singular

if (norm(A_aug(p,1:p),'inf')< eps)
    fprintf('Error: Given matrix is singular \n');
    return
end

for k=1:p
    A_inv(p,k) = A_aug(p,p+k)/A_aug(p,p);
    for l=p-1:-1:1
        A_inv(l,k) = (A_aug(l,p+k) - A_aug(l,l+1:p) *
A_inv(l+1:p,k))/A_aug(l,l);
    end
end

error = norm(A*A_inv - eye(p));
if (error < 10^(-10))
    fprintf('Inverse matrix achieved');
end
```

- ***Driver to run the above function:***

```
% Driver Q1

% Part A - 4x4 matrix
fprintf('Part A \n');
A = [1 2 0 0
    1 3 -1 0
    0 -1 1 3
    0 0 2 3]

A_inv = get_inv(A)
fprintf('Confirming result: A*A_inv = \n');
A*A_inv

% Part B - 10x10 matrix
fprintf('Part B \n');
A = [1.000000   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000
0.111111   0.100000
   0.500000   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111
0.100000   0.090909
   0.333333   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000
0.090909   0.083333
   0.250000   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909
0.083333   0.076923
   0.200000   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909   0.083333
0.076923   0.071429
   0.166667   0.142857   0.125000   0.111111   0.100000   0.090909   0.083333   0.076923
0.071429   0.066667
   0.142857   0.125000   0.111111   0.100000   0.090909   0.083333   0.076923   0.071429
0.066667   0.062500
   0.125000   0.111111   0.100000   0.090909   0.083333   0.076923   0.071429   0.066667
0.062500   0.058824
   0.111111   0.100000   0.090909   0.083333   0.076923   0.071429   0.066667   0.062500
0.058824   0.055556
   0.100000   0.090909   0.083333   0.076923   0.071429   0.066667   0.062500   0.058824
0.055556   0.052632]

A_inv = get_inv(A)
fprintf('Confirming result: A*A_inv = \n');
A*A_inv
```

- ***Matlab output:***

- Part A

```
A =

     1     2     0     0
     1     3    -1     0
     0    -1     1     3
     0     0     2     3

Inverse matrix achieved
A_inv =

    2.0000   -1.0000    1.0000   -1.0000
   -0.5000    0.5000   -0.5000    0.5000
    0.5000   -0.5000   -0.5000    0.5000
   -0.3333    0.3333    0.3333         0

Confirming result: A*A_inv =

ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1

Part B

A =

  Columns 1 through 7

    1.0000    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429
    0.5000    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250
    0.3333    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111
    0.2500    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000
    0.2000    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909
    0.1667    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833
    0.1429    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769
    0.1250    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714
    0.1111    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667
    0.1000    0.0909    0.0833    0.0769    0.0714    0.0667    0.0625

  Columns 8 through 10

    0.1250    0.1111    0.1000
    0.1111    0.1000    0.0909
    0.1000    0.0909    0.0833
    0.0909    0.0833    0.0769
    0.0833    0.0769    0.0714
    0.0769    0.0714    0.0667
    0.0714    0.0667    0.0625
```

```
    0.0667    0.0625    0.0588
    0.0625    0.0588    0.0556
    0.0588    0.0556    0.0526


A_inv =

   1.0e+06 *

  Columns 1 through 7

    0.0000   -0.0002    0.0000    0.0020   -0.0051    0.0057   -0.0046
   -0.0002    0.0014    0.0056   -0.0543    0.1316   -0.1544    0.1270
    0.0000    0.0056   -0.0855    0.3983   -0.8356    0.9632   -0.7794
    0.0020   -0.0543    0.3983   -1.2578    2.0448   -1.9935    1.4756
   -0.0051    0.1316   -0.8356    2.0448   -2.0202    0.5374    0.1014
    0.0057   -0.1544    0.9632   -1.9935    0.5374    2.5822   -2.6077
   -0.0046    0.1270   -0.7794    1.4756    0.1014   -2.6077    1.8736
    0.0033   -0.0845    0.4825   -0.8152   -0.1963    1.4445   -1.0701
   -0.0006    0.0202   -0.1188    0.1075    0.6221   -1.7442    2.1819
   -0.0005    0.0076   -0.0308    0.0931   -0.3799    0.9676   -1.2997

  Columns 8 through 10

    0.0033   -0.0006   -0.0005
   -0.0845    0.0202    0.0076
    0.4825   -0.1188   -0.0308
   -0.8152    0.1075    0.0931
   -0.1963    0.6221   -0.3799
    1.4445   -1.7442    0.9676
   -1.0701    2.1819   -1.2997
    1.3844   -2.2440    1.0960
   -2.2440    1.8692   -0.6938
    1.0960   -0.6938    0.2412

Confirming result: A*A_inv =

ans =

  Columns 1 through 7

    1.0000   -0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0000
    0.0000    1.0000   -0.0000    0.0000    0.0000    0.0000    0.0000
   -0.0000    0.0000    1.0000    0.0000   -0.0000    0.0000    0.0000
    0.0000   -0.0000    0.0000    1.0000   -0.0000    0.0000    0.0000
    0.0000   -0.0000   -0.0000    0.0000    1.0000    0.0000   -0.0000
   -0.0000    0.0000   -0.0000    0.0000    0.0000    1.0000    0.0000
    0.0000   -0.0000   -0.0000    0.0000         0    0.0000    1.0000
   -0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
   -0.0000         0         0         0    0.0000         0         0
    0.0000         0    0.0000    0.0000   -0.0000    0.0000   -0.0000

  Columns 8 through 10
```

```
  -0.0000   -0.0000    0.0000
   0.0000   -0.0000    0.0000
   0.0000   -0.0000   -0.0000
   0.0000    0.0000   -0.0000
   0.0000   -0.0000    0.0000
  -0.0000    0.0000   -0.0000
  -0.0000         0   -0.0000
   1.0000    0.0000   -0.0000
        0    1.0000         0
        0         0    1.0000
```

- *Published with MATLAB® R2021a*

Q2.

Q.2) LU decomposition → No pivoting

$A_{4\times4}$ → Given

Process: for L & U

L → lower triangular form matrix

U → Upper triangular form matrix

Q2. Programs

- ***Program to perform LU decomposition without pivoting:***

```
% Function for LU decomposition
% No pivoting in this case

function [L,U]  = LU_decompose(A)

[p,p] = size(A);
% lower triangle matrix
L = zeros(p,p);
L(1,1) = 1;
% Upper triangle matrix
U = A;

for i = 1:p-1
     if U(i,i) == 0
         fprintf('Pivoting required to reduce A to U(upper traingular matrix 
form)\n');
         break
     end

    L(i+1,i+1) = 1;
    for j = i+1:p
        L(j,i) = U(j,i)/U(i,i);
        U(j,i:p) = U(j,i:p) - L(j,i) * U(i,i:p);
    end
end

k = norm(L*U-A);
if (k < (10^(-10)))
    fprintf('No issues in performing LU decomposition');
end
```

- ***Driver to run the above function:***

```
% Q2 driver

% Matrix A given
A = [4 -1 3 2
    -8 0 -3 -3.5
    2 -3.5 10 3.75
    -8 -4 1 -0.5]

[L,U] = LU_decompose(A)
```

- ***Matlab output:***

```
A =

    4.0000   -1.0000    3.0000    2.0000
   -8.0000         0   -3.0000   -3.5000
    2.0000   -3.5000   10.0000    3.7500
   -8.0000   -4.0000    1.0000   -0.5000

No issues in performing LU decomposition
L =

    1.0000         0         0         0
   -2.0000    1.0000         0         0
    0.5000    1.5000    1.0000         0
   -2.0000    3.0000   -0.5000    1.0000


U =

    4.0000   -1.0000    3.0000    2.0000
         0   -2.0000    3.0000    0.5000
         0         0    4.0000    2.0000
         0         0         0    3.0000
```

Published with MATLAB® R2021a

Q3.

Q.3) Truss :

6 points/pins where forces are acting

Resolving forces in horizontal & vertical directions

$x(1) = D + F\cos\theta_1 = 0$ $\quad y(1) = A + F\sin\theta_1 = 0$

$x(2) = E - D - G\cos\theta_2 = 0$ $\quad y(2) = G\sin\theta_2 - 2000 = 0$

$x(3) = I + L\cos\theta_4 - H\cos\theta_3 - E = 0$

$y(3) = L\sin\theta_4 + H\sin\theta_3 - 2500 = 0$

$x(4) = K + H\cos\theta_3 + G\cos\theta_2 - F\cos\theta_1 = 0$

$y(4) = -H\sin\theta_4 - G\sin\theta_2 - F\sin\theta_1 = 0$

$x(5) = B - I = 0$ $\quad y(5) = C + J = 0$

$x(6) = -K - L\cos\theta_4 = 0$ $\quad y(6) = +J + L\sin\theta_4 = 0.$

given: $x = \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ I \\ J \\ K \\ L \end{bmatrix}$

$Ax = B$

$\downarrow$

13x13 Matrix (shown in Matlab).

Q3. Programs

- ***Truss problem to compute the required forces:***

```
% Making a function to compute the forces necessary to hold the trusses in
% quilibirum

function x = truss(A,B)

[p,q] = size(A);
r = length(B);
x = zeros(q,1);

if (p ~= q)
    fprintf('Given matrix is not square \n');
    return
end

if (r ~= q)
    fprintf('Given matrix and vector is not compatible \n');
    return
end

A_aug = [A B];
for i=1:q-1
    [max_value,j] = max(abs(A_aug(i:q,i)));
    if (j>1)
        temp_row = A_aug(i,:);
        A_aug(i,:) = A_aug(i+j-1,:);
        A_aug(i+j-1,:) = temp_row;
    end

    for k = i+1:q
        pivot = A_aug(k,i)/A_aug(i,i);
        A_aug(k,:) = A_aug(k,:) - pivot.*A_aug(i,:);
    end
end

if (norm(A_aug(q,1:q),'inf') < eps)
    fprintf('Given matrix in not singluar');
    return
end

x(q) = A_aug(q,q+1)/A_aug(q,q);
for l = q-1:-1:1
    x(l) = (A_aug(l,q+1)-A_aug(l,l+1:q)*x(l+1:q))/A_aug(l,l);
end
```

- ***Driver to run the above function:***

```
% Q3 driver

theta_1 = 48.4 * (pi/180);
theta_2 = 66.0 * (pi/180);
theta_3 = 26.6 * (pi/180);
```

```
theta_4 = 56.3 * (pi/180);

A = [0 0 0 1 0 cos(theta_1) 0 0 0 0 0 0
     1 0 0 0 0 sin(theta_1) 0 0 0 0 0 0
     0 0 0 -1 1 0 -cos(theta_2) 0 0 0 0 0
     0 0 0 0 0 0 sin(theta_2) 0 0 0 0 0
     0 0 0 0 -1 0 0 -cos(theta_3) 1 0 0 cos(theta_4)
     0 0 0 0 0 0 0 sin(theta_3) 0 0 0 sin(theta_4)
     0 0 0 0 0 -cos(theta_1) cos(theta_2) cos(theta_3) 0 0 1 0
     0 0 0 0 0 -sin(theta_1) -sin(theta_2) -sin(theta_3) 0 0 0 0
     0 1 0 0 0 0 0 0 -1 0 0 0
     0 0 1 0 0 0 0 0 0 1 0 0
     0 0 0 0 0 0 0 0 0 0 -1 -cos(theta_4)
     0 0 0 0 0 0 0 0 0 -1 0 -sin(theta_4)]

B = [0
    0
    0
    2000
    0
    2500
    0
    0
    0
    0
    0
    0]

x = truss(A,B)
```

- ***Matlab output:***

```
A =

  Columns 1 through 7

         0         0         0    1.0000         0    0.6639         0
    1.0000         0         0         0         0    0.7478         0
         0         0         0   -1.0000    1.0000         0   -0.4067
         0         0         0         0         0         0    0.9135
         0         0         0         0   -1.0000         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0   -0.6639    0.4067
         0         0         0         0         0   -0.7478   -0.9135
         0    1.0000         0         0         0         0         0
         0         0    1.0000         0         0         0         0
         0         0         0         0         0         0         0
         0         0         0         0         0         0         0

  Columns 8 through 12
```

```
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
         0         0         0         0         0
   -0.8942    1.0000         0         0    0.5548
    0.4478         0         0         0    0.8320
    0.8942         0         0    1.0000         0
   -0.4478         0         0         0         0
         0   -1.0000         0         0         0
         0         0    1.0000         0         0
         0         0         0   -1.0000   -0.5548
         0         0   -1.0000         0   -0.8320


B =

           0
           0
           0
        2000
           0
        2500
           0
           0
           0
           0
           0
           0


x =

   1.0e+03 *

    1.7188
         0
    2.7812
    1.5260
    2.4165
   -2.2984
    2.1893
   -0.6281
         0
   -2.7812
   -1.8548
    3.3430
```

Published with MATLAB® R2021a

Q4.

Q.4) Thomas algorithm:

$$x_1 + x_2 = 5$$
$$2x_1 - x_2 + 5x_3 = -9$$
$$3x_2 - 4x_3 + 2x_4 = 19$$
$$2x_3 + 6x_4 = 2$$

$$\begin{array}{cc} & \begin{array}{cccc} & c & & \end{array} \\ \begin{array}{c} b \\ a \\ \\ \end{array} & \begin{bmatrix} 1 & 1 & 0 & 0 \\ 2 & -1 & 5 & 0 \\ 0 & 3 & -4 & 2 \\ 0 & 0 & 2 & 6 \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \overset{\text{rhs}}{\begin{bmatrix} 5 \\ -9 \\ 19 \\ 2 \end{bmatrix}}$$

$$a = [2 \quad 3 \quad 2]$$
$$b = [1 \quad -1 \quad -4 \quad 6]$$
$$c = [1 \quad 5 \quad 2]$$

Matlab program attached

Q4. Programs

- ***Script for Thomas algorithm:***

```
% Making a function for Thomas algorithm

function x = thomas_alg(a,b,c,rhs)

p = length(rhs);

c(1) = c(1)/b(1);
rhs(1) = rhs(1)/b(1);
b(1) = 1;

for i = 2:p
    b(i) = b(i) - a(i-1)*c(i-1);
    rhs(i) = rhs(i) - (a(i-1) * rhs(i-1));
    if (i < p)
        c(i) = c(i)/b(i);
    end
    rhs(i) = rhs(i)/b(i);
    b(i) = 1;
end

% Gaussian elimination done.
% performing back substitution now

x(p) = rhs(p);
for j = p-1:-1:1
    x(j) = (rhs(j)-(c(j)*x(j+1)));
end
x = x';
```

- ***Driver to run the above function***

```
% Q4 driver

a = [2 3 2];
b = [1 -1 -4 6];
c = [1 5 2];
rhs = [5 -9 19 2];

x = thomas_alg(a,b,c,rhs)
```

- ***Matlab Output:***

```
x =

    2.0000
    3.0000
   -2.0000
    1.0000
```

Published with MATLAB® R2021a