PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

APRIL 2022: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE20MA251- LINEAR ALGEBRA

MATLAB Assignment

Session: Jan-May 2022

Name of the Student : _____________Sriram R_________________

SRN : __________PES1UG20CS435__________________

Branch : Computer Science and Engineering

Semester & Section : Semester IV Section H

FOR OFFICE USE ONLY:

Marks Allotted : / 05

Name of the Course Instructor :**Prof. Jyothi R**

Signature of the Course Instructor : ______________________________

Gaussian Elimination

```
C = [1 2 -1; 2 1 -2; -3 1 1]
b= [3 3 -6]'
A = [C b];
n= size(A,1);
x = zeros(n,1); %variable matrix [x1 x2
... xn] column
for i=1:n-1
for j=i+1:n
m = A(j,i)/A(i,i)
A(j,:) = A(j,:) - m*A(i,:)
end
end
x(n) = A(n,n+1)/A(n,n)
for i=n-1:-1:1
summ = 0
for j=i+1:n
summ = summ + A(i,j)*x(j,:)
x(i,:) = (A(i,n+1) - summ)/A(i,i)
end
end
```

Output:

x = 3, y = 1, z = 2

```
x =

     1
     1
     2

summ =

     0

x =

     3
     1
     2
```

Gauss Jordan Method

```
A =[1,1,1;4,3,-1;3,5,3];
n =length(A(1,:));
Aug =[A,eye(n,n)]
for j=1:n-1
for i=j+1:n
Aug(i,j:2*n)=Aug(i,j:2*n)-Aug(i,j)/Aug(j,j)*Aug(j,j:2*n)
end
end
for j=n:-1:2
Aug(1:j-1,:)=Aug(1:j-1,:)-Aug(1:j-1,j)/Aug(j,j)*Aug(j,:)
end
for j=1:n
Aug(j,:)=Aug(j,:)/Aug(j,j)
end
B=Aug(:,n+1:2*n)
```

Output :

```
Aug =

    1.0000         0         0    1.4000    0.2000   -0.4000
         0    1.0000         0   -1.5000         0    0.5000
         0         0  -10.0000  -11.0000    2.0000    1.0000

Aug =

    1.0000         0         0    1.4000    0.2000   -0.4000
         0    1.0000         0   -1.5000         0    0.5000
         0         0    1.0000    1.1000   -0.2000   -0.1000

B =

    1.4000    0.2000   -0.4000
   -1.5000         0    0.5000
    1.1000   -0.2000   -0.1000
```

LU Decomposition

```
%LU Decomposition
Ab = [1 1 1;1 2 2;1 2 3];
%% Forward Elimination
n= length(A);
L = eye(n);
% With A(1,1) as pivot Element
for i =2:3
alpha = Ab(i,1)/Ab(1,1);
L(i,1) = alpha;
Ab(i,:) = Ab(i,:) - alpha*Ab(1,:);
end
% With A(2,2) as pivot Element
i=3;
alpha = Ab(i,2)/Ab(2,2);
L(i,2) = alpha
Ab(i,:) = Ab(i,:) - alpha*Ab(2,:);
U = Ab(1:n,1:n)
```

Output :

```
L =

     1     0     0
     1     1     0
     1     1     1


U =

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

Four Fundamental Subspaces

```
% Bases of four fundamental vector spaces of matrix A.
A=[1,2,3;2,-1,1];
% Row Reduced Echelon Form
[R, pivot] = rref(A)
% Rank
rank = length(pivot)
% basis of the column space of A
columnsp = A(:,pivot)
% basis of the nullspace of A
nullsp = null(A,'r')
% basis of the row space of A
rowsp = R(1:rank,:)'
% basis of the left nullspace of A
leftnullsp = null(A','r')
```

Output :

```
columnsp =

     1     2
     2    -1

nullsp =

    -1
    -1
     1

rowsp =

     1     0
     0     1
     1     1

leftnullsp =

  2×0 empty double matrix
```

QR Factorisation

1.

Command: >> [Q,R]=qr([1,1,0;1,0,1;0,1,1])

Output:

```
Q =

   -0.7071    0.4082   -0.5774
   -0.7071   -0.4082    0.5774
         0    0.8165    0.5774


R =

   -1.4142   -0.7071   -0.7071
         0    1.2247    0.4082
         0         0    1.1547
```

2.

Command: >> [Q,R]=qr([1,1,1,1;-1,4,4,-1;4,-2,2,0])

Output:

```
Q =

   -0.2357   -0.4264   -0.8733
    0.2357   -0.8969    0.3743
   -0.9428   -0.1176    0.3119


R =

   -4.2426    2.5927   -1.1785   -0.4714
         0   -3.7786   -4.2491    0.4705
         0         0    1.2476   -1.2476
```

3.

Command: >> [Q,R]=qr([3,2,4;2,0,2;4,2,3])

Output:

```
Q =

   -0.5571    0.4952   -0.6667
   -0.3714   -0.8666   -0.3333
   -0.7428    0.0619    0.6667


R =

   -5.3852   -2.5997   -5.1995
         0    1.1142    0.4333
         0         0   -1.3333
```

Projection Matrices and Least Square

1.

Command:

>> A=[1,0;0,1;1,1]

>> b=[1;3;4]

>> x=lsqr(A,b)

Output:

```
lsqr converged at iteration 2 to a solution with relative residual 4.3e-17.

x =

     1
     3
```

2.

Command:

>> A=[1,0;0,2;3,1]

>> b=[1;0;4]

>> x=lsqr(A,b)

Output:

```
lsqr converged at iteration 2 to a solution with relative residual 0.076.

x =

    1.2927
    0.0244
```

3.

Command:

```
>>  A=[1,2;3,1;4,1]

>> b=[1;5;4]

>> x=lsqr(A,b)
```

Output:

```
lsqr converged at iteration 2 to a solution with relative residual 0.25.

x =

    1.2400
   -0.0267
```

Grams Schmidt Organisation

1.

Command:

```
>> A=[1,1,2;0,0,1;1,0,0]

>> Q=zeros(3)

>> R=zeros(3)

>> for j=1:3

>> v=A(: , j)

>> for i=1:j-1

>> R(i,j)=Q(:,i)'*A(:,j)

>> v=v-R(i,j)*Q(:,i)

>> end

>> R(j,j)=norm(v)

>> Q(:,j)=v/R(j,j)

>> end
```

Output:

```
v =

   -0.0000
    1.0000
    0.0000


R =

    1.4142    0.7071    1.4142
         0    0.7071    1.4142
         0         0    1.0000


Q =

    0.7071    0.7071   -0.0000
         0         0    1.0000
    0.7071   -0.7071    0.0000
```

2. Command

>> A=[0,1,1;1,1,0;1,-1,2;1,0,-1]

>> Q=zeros(4,3)

>> R=zeros(3)

>> for j=1:3

>> v=A(: , j);

>> For i=1:j-1

>> R(i,j)=Q(:,i)'*A(:,j)

>> v=v-R(i,j)*Q(:,i)

>> end

>> R(j,j)=norm(v)

>> Q(:,j)=v/R(j,j)

>> end

Output:

```
v =

    1.3333
         0
    1.3333
   -1.3333


R =

    1.7321         0    0.5774
         0    1.7321   -0.5774
         0         0    2.3094


Q =

         0    0.5774    0.5774
    0.5774    0.5774         0
    0.5774   -0.5774    0.5774
    0.5774         0   -0.5774
```

3.

Command:

```
>> A=[0,2,3;1,1,0;3,-4,2;1,5,-1]
>> Q=zeros(4,3)
>> R=zeros(3)
>> for j=1:3
>> v=A(: , j)
>> for i=1:j-1
>> R(i,j)=Q(:,i)'*A(:,j)
>> v=v-R(i,j)*Q(:,i)
>> end
>> R(j,j)=norm(v)
>> Q(:,j)=v/R(j,j)
>> end
```

Output:

```
v =

    3.2000
   -0.3000
    0.4000
   -0.9000


R =

    3.3166   -1.8091    1.5076
         0    6.5366   -0.6537
         0         0    3.3615


Q =

         0    0.3060    0.9519
    0.3015    0.2364   -0.0892
    0.9045   -0.3616    0.1190
    0.3015    0.8484   -0.2677
```

GAUSS JORDAN INVERSE

1.

Command:

```
A =[1,1,1;4,3,-1;3,5,3];
n =length(A(1,:));
Aug =[A,eye(n,n)]
for j=1:n-1
for i=j+1:n
Aug(i,j:2*n)=Aug(i,j:2*n)-Aug(i,j)/Aug(j,j)*Aug(j,j:2*n)
end
end
```

```
for j=n:-1:2
Aug(1:j-1,:)=Aug(1:j-1,:)-Aug(1:j-1,j)/Aug(j,j)*Aug(j,:)
end
for j=1:n
Aug(j,:)=Aug(j,:)/Aug(j,j)
end
B=Aug(:,n+1:2*n)
```

Output:

```
Aug =

     1     1     1     1     0     0
     4     3    -1     0     1     0
     3     5     3     0     0     1


Aug =

     1     1     1     1     0     0
     0    -1    -5    -4     1     0
     3     5     3     0     0     1


Aug =

     1     1     1     1     0     0
     0    -1    -5    -4     1     0
     0     2     0    -3     0     1
```

```
Aug =

     1     1     1     1     0     0
     0    -1    -5    -4     1     0
     0     0   -10   -11     2     1


Aug =

    1.0000    1.0000         0   -0.1000    0.2000    0.1000
         0   -1.0000         0    1.5000         0   -0.5000
         0         0  -10.0000  -11.0000    2.0000    1.0000


Aug =

    1.0000         0         0    1.4000    0.2000   -0.4000
         0   -1.0000         0    1.5000         0   -0.5000
         0         0  -10.0000  -11.0000    2.0000    1.0000
```

```
Aug =

    1.0000         0         0    1.4000    0.2000   -0.4000
         0    1.0000         0   -1.5000         0    0.5000
         0         0  -10.0000  -11.0000    2.0000    1.0000


Aug =

    1.0000         0         0    1.4000    0.2000   -0.4000
         0    1.0000         0   -1.5000         0    0.5000
         0         0    1.0000    1.1000   -0.2000   -0.1000


B =

    1.4000    0.2000   -0.4000
   -1.5000         0    0.5000
    1.1000   -0.2000   -0.1000
```

LU DECOMPOSITION:

1.

Command:

```
Ab = [1 1 -1;3 5 6;7 8 9];
n= length(A);
L = eye(n);
for i =2:3
alpha = Ab(i,1)/Ab(1,1);
L(i,1) = alpha;
Ab(i,:) = Ab(i,:) - alpha*Ab(1,:);
end
i=3;
alpha = Ab(i,2)/Ab(2,2);
L(i,2) = alpha
Ab(i,:) = Ab(i,:) - alpha*Ab(2,:);
U = Ab(1:n,1:n)
```

Output:

```
L =

    1.0000         0         0
    3.0000    1.0000         0
    7.0000    0.5000    1.0000


U =

    1.0000    1.0000   -1.0000
         0    2.0000    9.0000
         0         0   11.5000
```

2.

Command:

```
Ab = [1 1 -2;3 4 6;7 -8 9];
```

```
n= length(A);
L = eye(n);
for i =2:3
alpha = Ab(i,1)/Ab(1,1);
L(i,1) = alpha;
Ab(i,:) = Ab(i,:) - alpha*Ab(1,:);
end
i=3;
alpha = Ab(i,2)/Ab(2,2);
L(i,2) = alpha
Ab(i,:) = Ab(i,:) - alpha*Ab(2,:);
U = Ab(1:n,1:n)
```

Output:

```
L =

     1     0     0
     3     1     0
     7   -15     1


U =

     1     1    -2
     0     1    12
     0     0   203
```

Gauss Jordan Elimination

1.

Command:

```
C = [1 2 -1; 2 1 -2; -3 1 1]
b= [3 3 -6]'
```

```
A = [C b];
n= size(A,1);
x = zeros(n,1); %variable matrix [x1 x2
 ... xn] column
for i=1:n-1
for j=i+1:n
m = A(j,i)/A(i,i)
A(j,:) = A(j,:) - m*A(i,:)
end
end
x(n) = A(n,n+1)/A(n,n)
for i=n-1:-1:1
summ = 0
for j=i+1:n
summ = summ + A(i,j)*x(j,:)
x(i,:) = (A(i,n+1) - summ)/A(i,i)
end
end
```

Output:

```
x =

     3
     1
     2
```

2.

Command:

```
C = [2 1 -1; 2 5 7; 1 1 1]
b= [0 52 9]'
```

```
A = [C b];

n= size(A,1);

x = zeros(n,1); %variable matrix [x1 x2

 ... xn] column

for i=1:n-1

for j=i+1:n

m = A(j,i)/A(i,i)

A(j,:) = A(j,:) - m*A(i,:)

end

end

x(n) = A(n,n+1)/A(n,n)

for i=n-1:-1:1

summ = 0

for j=i+1:n

summ = summ + A(i,j)*x(j,:)

x(i,:) = (A(i,n+1) - summ)/A(i,i)

end

end
```

Output:

```
x =

     1
     3
     5
```

Eigen Values and Eigen Vectors

1.

Command:

>> A=[1,1,3;1,5,1;3,1,1]

>> e=eig(A)

>> [V,D]=eig(A)

Output:

```
e =

   -2.0000
    3.0000
    6.0000

>> [V,D]=eig(A)

V =

   -0.7071    0.5774    0.4082
   -0.0000   -0.5774    0.8165
    0.7071    0.5774    0.4082


D =

   -2.0000         0         0
         0    3.0000         0
         0         0    6.0000
```

2.

Command:

>> A=[1,3,1;4,1,3;2,1,3]

>> e=eig(A)

>> [V,D]=eig(A)

Output:

```
e =

    6.1970
   -2.3132
    1.1162

>> [V,D]=eig(A)

V =

   -0.4986   -0.6863   -0.5816
   -0.6881    0.7168   -0.2774
   -0.5272    0.1234    0.7647


D =

    6.1970         0         0
         0   -2.3132         0
         0         0    1.1162
```

Four Fundamental Subspaces

1.

Command:

clc;

clear all;

close all;

% Bases of four fundamental vector spaces of matrix A.

A=[1,2,3;2,-1,1];

% Row Reduced Echelon Form

[R, pivot] = rref(A)

% Rank

rank = length(pivot)

% basis of the column space of A

columnsp = A(:,pivot)

% basis of the nullspace of A

nullsp = null(A,'r')

% basis of the row space of A

rowsp = R(1:rank,:)'

% basis of the left nullspace of A

leftnullsp = null(A','r')

Output:

```
R =

     1     0     1
     0     1     1


pivot =

     1     2


rank =

     2


columnsp =

     1     2
     2    -1
```

```
nullsp =

    -1
    -1
     1


rowsp =

     1     0
     0     1
     1     1


leftnullsp =

  2×0 empty double matrix
```

2.

Command:

```
clc;

clear all;

close all;

% Bases of four fundamental vector spaces of matrix A.

A=[1,2,3;4,-2,1];

% Row Reduced Echelon Form

[R, pivot] = rref(A)

% Rank

rank = length(pivot)

% basis of the column space of A

columnsp = A(:,pivot)

% basis of the nullspace of A

nullsp = null(A,'r')

% basis of the row space of A

rowsp = R(1:rank,:)'

% basis of the left nullspace of A

leftnullsp = null(A','r')
```

Output:

```
R =

    1.0000         0    0.8000
         0    1.0000    1.1000


pivot =

     1     2


rank =

     2


columnsp =

     1     2
     4    -2


nullsp =

   -0.8000
   -1.1000
    1.0000


rowsp =

    1.0000         0
         0    1.0000
    0.8000    1.1000


leftnullsp =

  2×0 empty double matrix
```