PES UNIVERSITY

ELECTRONIC CITY CAMPUS

(Established under Karnataka Act no. 16 of 2013)

Hosur Road, Near Electronic City,Bangalore-100

MAT LAB

Subject: Linear Algebra and its Applications

Subject Code: UE20MA251

Name: Vishwa Mehul Mehta
SRN: PES2UG20CS389
Section: F
Branch: CSE

1. Gaussian Elimination:

a) $A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & 1 & -2 \\ -3 & 1 & 1 \end{bmatrix}$, b= $\begin{bmatrix} 3 \\ 3 \\ -6 \end{bmatrix}$

Code:

```
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);
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
    s = 0
    for j=i+1:n
        s = s + A(i,j) * x(j,:)
        x(i,:) = (A(i,n+1) - s) / A(i,i)
    end
end
```

```
      0      0     -2     -4

x = 3x1
      0
      0
      2

s = 0
s = 0
x = 3x1
      0
      1
      2

s = 0
s = 2
x = 3x1
      1
      1
      2

s = 0
x = 3x1
      3
      1
      2
```

Output:

```
C = 3x3
     1     2    -1
     2     1    -2
    -3     1     1
b = 3x1
     3
     3
    -6
m = 2
A = 3x4
     1     2    -1     3
     0    -3     0    -3
    -3     1     1    -6
m = -3
A = 3x4
     1     2    -1     3
     0    -3     0    -3
     0     7    -2     3
m = -2.3333
A = 3x4
     1     2    -1     3
     0    -3     0    -3
     0     0    -2    -4
x = 3x1
     0
     0
```

```
      2
s = 0
s = 0
x = 3×1
      0
      1
      2
s = 0
s = 2
x = 3×1
      1
      1
      2
s = 0
x = 3×1
      3
      1
      2
```

b) $A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & -6 & -1 \\ 3 & 4 & 2 \end{bmatrix}$, b= $\begin{bmatrix} 11 \\ 0 \\ 0 \end{bmatrix}$

Code:

```
C = [1 1 1; 2 -6 -1; 3 4 2]
b = [11 0 0]'
A = [C b];
n = size(A,1);
x = zeros(n,1);
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
    s = 0
    for j=i+1:n
        s = s + A(i,j) * x(j,:)
        x(i,:) = (A(i,n+1) - s) / A(i,i)
    end
end
```

```
x = 3×1
     0
     0
    26

s = 0
s = -78
x = 3×1
     0
    -7
    26

s = 0
s = -7
x = 3×1
    18
    -7
    26

s = 19
x = 3×1
    -8
    -7
    26
```

Output:

```
C = 3×3
      1     1     1
      2    -6    -1
      3     4     2
b = 3×1
     11
      0
      0
m = 2
```

```
A = 3×4
     1     1     1    11
     0    -8    -3   -22
     3     4     2     0
m = 3
A = 3×4
     1     1     1    11
     0    -8    -3   -22
     0     1    -1   -33
m = -0.1250
A = 3×4
    1.0000    1.0000    1.0000   11.0000
         0   -8.0000   -3.0000  -22.0000
         0         0   -1.3750  -35.7500
x = 3×1
     0
     0
    26
s = 0
s = -78
x = 3×1
     0
    -7
    26
s = 0
s = -7
x = 3×1
    18
    -7
    26
s = 19
x = 3×1
    -8
    -7
    26
```

c) $A = \begin{bmatrix} 2 & 1 & -1 \\ 2 & 5 & 7 \\ 1 & 1 & 1 \end{bmatrix}$, b= $\begin{bmatrix} 0 \\ 52 \\ 9 \end{bmatrix}$

Code:

```
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);
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
    s = 0
    for j=i+1:n
        s = s + A(i,j) * x(j,:)
        x(i,:) = (A(i,n+1) - s) / A(i,i)
    end
end
```

```
x = 3x1
     0
     0
     5

s = 0
s = 40
x = 3x1
     0
     3
     5

s = 0
s = 3
x = 3x1
   -1.5000
    3.0000
    5.0000

s = -2
x = 3x1
     1
     3
     5
```

Output:

```
C = 3×3
     2     1    -1
     2     5     7
     1     1     1
b = 3×1
     0
    52
     9
m = 1
A = 3×4
     2     1    -1     0
     0     4     8    52
     1     1     1     9
m = 0.5000
A = 3×4
    2.0000    1.0000   -1.0000         0
         0    4.0000    8.0000   52.0000
         0    0.5000    1.5000    9.0000
m = 0.1250
A = 3×4
    2.0000    1.0000   -1.0000         0
         0    4.0000    8.0000   52.0000
         0         0    0.5000    2.5000
x = 3×1
     0
     0
     5
s = 0
s = 40
x = 3×1
     0
     3
     5
s = 0
s = 3
x = 3×1
```

```
    -1.5000
     3.0000
     5.0000
s = -2
x = 3×1
       1
       3
       5
```

2. Find inverse by Gauss Jordan method:

a) $A = \begin{bmatrix} 1 & 1 & 1 \\ 4 & 3 & -1 \\ 3 & 5 & 3 \end{bmatrix}$

Code:

```
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(i:j-1,:) = Aug(i:j-1,:) - Aug(i: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)
```

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

Aug = 3×6
       1       1       1       1       0       0
       0      -1      -5      -4       1       0
       0       0     -10     -11       2       1

Aug = 3×6
       1       1       1       1       0       0
       0       1       5       4      -1       0
       0       0     -10     -11       2       1

Aug = 3×6
    1.0000    1.0000    1.0000    1.0000 ...
         0    1.0000    5.0000    4.0000
         0         0    1.0000    1.1000

B = 3×3
    1.0000         0         0
    4.0000   -1.0000         0
    1.1000   -0.2000   -0.1000
```

Output:

```
Aug = 3×6
       1       1       1       1       0       0
       4       3      -1       0       1       0
       3       5       3       0       0       1
Aug = 3×6
       1       1       1       1       0       0
       0      -1      -5      -4       1       0
       3       5       3       0       0       1
Aug = 3×6
       1       1       1       1       0       0
       0      -1      -5      -4       1       0
       0       2       0      -3       0       1
Aug = 3×6
       1       1       1       1       0       0
       0      -1      -5      -4       1       0
       0       0     -10     -11       2       1
Aug = 3×6
       1       1       1       1       0       0
       0      -1      -5      -4       1       0
       0       0     -10     -11       2       1
Aug = 3×6
```

```
     1      1      1      1      0      0
     0     -1     -5     -4      1      0
     0      0    -10    -11      2      1
Aug = 3×6
     1      1      1      1      0      0
     0     -1     -5     -4      1      0
     0      0    -10    -11      2      1
Aug = 3×6
     1      1      1      1      0      0
     0      1      5      4     -1      0
     0      0    -10    -11      2      1
Aug = 3×6
    1.0000    1.0000    1.0000    1.0000         0         0
         0    1.0000    5.0000    4.0000   -1.0000         0
         0         0    1.0000    1.1000   -0.2000   -0.1000
B = 3×3
    1.0000         0         0
    4.0000   -1.0000         0
    1.1000   -0.2000   -0.1000
```

b) $A = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 4 \\ 1 & 1 & 5 \end{bmatrix}$

Code:

```
A = [1, 2, 3; 1, 7, 4; 0, -1, 5];
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(i:j-1,:) = Aug(i:j-1,:) - Aug(i: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)
```

```
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000 ...
         0    5.0000    1.0000   -1.0000
         0         0    5.2000   -0.2000

Aug = 3×6
    1.0000    2.0000    3.0000    1.0000 ...
         0    5.0000    1.0000   -1.0000
         0         0    5.2000   -0.2000

Aug = 3×6
    1.0000    2.0000    3.0000    1.0000 ...
         0    1.0000    0.2000   -0.2000
         0         0    5.2000   -0.2000

Aug = 3×6
    1.0000    2.0000    3.0000    1.0000 ...
         0    1.0000    0.2000   -0.2000
         0         0    1.0000   -0.0385

B = 3×3
    1.0000         0         0
   -0.2000    0.2000         0
   -0.0385    0.0385    0.1923
```

Output:

```
Aug = 3×6
     1      2      3      1      0      0
     1      7      4      0      1      0
     0     -1      5      0      0      1
Aug = 3×6
     1      2      3      1      0      0
     0      5      1     -1      1      0
     0     -1      5      0      0      1
```

```
Aug = 3×6
     1     2     3     1     0     0
     0     5     1    -1     1     0
     0    -1     5     0     0     1
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    5.0000    1.0000   -1.0000    1.0000         0
         0         0    5.2000   -0.2000    0.2000    1.0000
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    5.0000    1.0000   -1.0000    1.0000         0
         0         0    5.2000   -0.2000    0.2000    1.0000
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    5.0000    1.0000   -1.0000    1.0000         0
         0         0    5.2000   -0.2000    0.2000    1.0000
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    5.0000    1.0000   -1.0000    1.0000         0
         0         0    5.2000   -0.2000    0.2000    1.0000
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    1.0000    0.2000   -0.2000    0.2000         0
         0         0    5.2000   -0.2000    0.2000    1.0000
Aug = 3×6
    1.0000    2.0000    3.0000    1.0000         0         0
         0    1.0000    0.2000   -0.2000    0.2000         0
         0         0    1.0000   -0.0385    0.0385    0.1923
B = 3×3
    1.0000         0         0
   -0.2000    0.2000         0
   -0.0385    0.0385    0.1923
```

c) $A = \begin{bmatrix} -1 & 2 & 6 \\ -1 & -2 & 4 \\ -1 & 1 & 5 \end{bmatrix}$

Code:

```
A = [-1, 2, 6; -1, -2, 4; -1, 1, 5];
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(i:j-1,:) = Aug(i:j-1,:) - Aug(i: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)
```

```
Aug = 3x6
   -1.0000    2.0000    6.0000    1.0000 ...
         0   -4.0000   -2.0000   -1.0000
         0         0   -0.5000   -0.7500

Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000 ...
         0   -4.0000   -2.0000   -1.0000
         0         0   -0.5000   -0.7500

Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000 ...
         0    1.0000    0.5000    0.2500
         0         0   -0.5000   -0.7500

Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000 ...
         0    1.0000    0.5000    0.2500
         0         0    1.0000    1.5000

B = 3x3
   -1.0000         0         0
    0.2500   -0.2500         0
    1.5000    0.5000   -2.0000
```

Output:

```
Aug = 3x6
    -1     2     6     1     0     0
    -1    -2     4     0     1     0
    -1     1     5     0     0     1
Aug = 3x6
    -1     2     6     1     0     0
     0    -4    -2    -1     1     0
    -1     1     5     0     0     1
Aug = 3x6
    -1     2     6     1     0     0
     0    -4    -2    -1     1     0
     0    -1    -1    -1     0     1
Aug = 3x6
   -1.0000    2.0000    6.0000    1.0000         0         0
         0   -4.0000   -2.0000   -1.0000    1.0000         0
         0         0   -0.5000   -0.7500   -0.2500    1.0000
Aug = 3x6
   -1.0000    2.0000    6.0000    1.0000         0         0
         0   -4.0000   -2.0000   -1.0000    1.0000         0
         0         0   -0.5000   -0.7500   -0.2500    1.0000
Aug = 3x6
   -1.0000    2.0000    6.0000    1.0000         0         0
         0   -4.0000   -2.0000   -1.0000    1.0000         0
         0         0   -0.5000   -0.7500   -0.2500    1.0000
Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000         0         0
         0   -4.0000   -2.0000   -1.0000    1.0000         0
         0         0   -0.5000   -0.7500   -0.2500    1.0000
Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000         0         0
         0    1.0000    0.5000    0.2500   -0.2500         0
         0         0   -0.5000   -0.7500   -0.2500    1.0000
Aug = 3x6
    1.0000   -2.0000   -6.0000   -1.0000         0         0
         0    1.0000    0.5000    0.2500   -0.2500         0
         0         0    1.0000    1.5000    0.5000   -2.0000
B = 3x3
```

```
   -1.0000         0         0
    0.2500   -0.2500         0
    1.5000    0.5000   -2.0000
```

3. LU Decomposition Method:

a) $A = \begin{bmatrix} 1 & 1 & -1 \\ 3 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Code:

```
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)
```

```
L = 3x3
    1.0000         0         0
    3.0000    1.0000         0
    7.0000    0.5000    1.0000

U = 3x3
    1.0000    1.0000   -1.0000
         0    2.0000    9.0000
         0         0   11.5000
```

Output:

```
L = 3x3
    1.0000         0         0
    3.0000    1.0000         0
    7.0000    0.5000    1.0000
U = 3x3
    1.0000    1.0000   -1.0000
         0    2.0000    9.0000
         0         0   11.5000
```

b) $A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 2 & 4 \\ 1 & 1 & 5 \end{bmatrix}$

Code:

```
Ab = [1 1 3; 1 2 4; 1 1 5];
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)
```

```
L = 3x3
     1     0     0
     1     1     0
     1     0     1

U = 3x3
     1     1     3
     0     1     1
     0     0     2
```

Output:

```
L = 3x3
     1     0     0
     0     1     0
     0     0     1
U = 3x3
    1.0000    1.0000   -1.0000
         0    2.0000    9.0000
         0         0   11.5000
```

c) $A = \begin{bmatrix} -1 & 4 & 6 \\ 0 & -2 & 4 \\ 0 & 0 & 5 \end{bmatrix}$

Code:

```
Ab = [-1 4 6; 0 -2 4; 0 0 5];
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)
```

```
L = 3x3
     1     0     0
     0     1     0
     0     0     1

U = 3x3
    -1     4     6
     0    -2     4
     0     0     5
```

Output:

```
L = 3×3
     1     0     0
     0     1     0
     0     0     1
U = 3×3
    -1     4     6
     0    -2     4
     0     0     5
```

4. Grams-Schmidt Orthogonalisation:

a) $A = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$

Code:

```
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:-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
```

R = 3×3

	1	2	3
1	1.4142	0	0
2	0	1.0000	0
3	0	0	0

```
Q = 3×3
    0.7071    1.0000         0
         0         0         0
    0.7071         0         0

v = 3×1
     2
     1
     0

R = 3×3
    1.4142         0         0
         0    1.0000         0
         0         0    2.2361

Q = 3×3
    0.7071    1.0000    0.8944
         0         0    0.4472
    0.7071         0         0
```

Output:

```
A = 3×3
     1     1     2
     0     0     1
     1     0     0
Q = 3×3
     0     0     0
     0     0     0
     0     0     0
R = 3×3
     0     0     0
     0     0     0
     0     0     0
v = 3×1
     1
```

```
       0
       1
R = 3×3
    1.4142         0         0
         0         0         0
         0         0         0
Q = 3×3
    0.7071         0         0
         0         0         0
    0.7071         0         0
v = 3×1
       1
       0
       0
R = 3×3
    1.4142         0         0
         0    1.0000         0
         0         0         0
Q = 3×3
    0.7071    1.0000         0
         0         0         0
    0.7071         0         0
v = 3×1
       2
       1
       0
R = 3×3
    1.4142         0         0
         0    1.0000         0
         0         0    2.2361
Q = 3×3
    0.7071    1.0000    0.8944
         0         0    0.4472
    0.7071         0         0
```

b) $A = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & -1 & 2 \\ 1 & 0 & -1 \end{bmatrix}$

Code:

```
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:-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
```

```
     1.7321          0          0
          0     1.7321          0
          0          0          0

Q = 4x3
          0     0.5774          0
     0.5774     0.5774          0
     0.5774    -0.5774          0
     0.5774          0          0

v = 4x1
      1
      0
      2
     -1

R = 3x3
     1.7321          0          0
          0     1.7321          0
          0          0     2.4495

Q = 4x3
          0     0.5774     0.4082
     0.5774     0.5774          0
     0.5774    -0.5774     0.8165
     0.5774          0    -0.4082
```

Output:

```
A = 4×3
     0     1     1
     1     1     0
     1    -1     2
     1     0    -1
Q = 4×3
     0     0     0
     0     0     0
     0     0     0
     0     0     0
R = 3×3
     0     0     0
     0     0     0
     0     0     0
v = 4×1
     0
     1
     1
     1
R = 3×3
    1.7321         0         0
         0         0         0
         0         0         0
Q = 4×3
         0         0         0
    0.5774         0         0
    0.5774         0         0
    0.5774         0         0
v = 4×1
     1
     1
    -1
     0
R = 3×3
    1.7321         0         0
```

```
          0     1.7321          0
          0          0          0
Q =  4×3
          0     0.5774          0
     0.5774     0.5774          0
     0.5774    -0.5774          0
     0.5774          0          0
v =  4×1
      1
      0
      2
     -1
R =  3×3
     1.7321          0          0
          0     1.7321          0
          0          0     2.4495
Q =  4×3
          0     0.5774     0.4082
     0.5774     0.5774          0
     0.5774    -0.5774     0.8165
     0.5774          0    -0.4082
```

c) $A = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 1 & 5 & 0 \end{bmatrix}$

Code:

```
A = [1,0,2; 0,1,1; 1,5,0]
Q = zeros(3)
R = zeros(3)
for j=1:3
    v=A(:,j)
    for i=1:-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
```

```
      1
      5

R = 3×3
     1.4142          0          0
          0     5.0990          0
          0          0          0

Q = 3×3
     0.7071          0          0
          0     0.1961          0
     0.7071     0.9806          0

v = 3×1
      2
      1
      0

R = 3×3
     1.4142          0          0
          0     5.0990          0
          0          0     2.2361

Q = 3×3
     0.7071          0     0.8944
          0     0.1961     0.4472
     0.7071     0.9806          0
```

Output:

```
A =  3×3
      1      0      2
      0      1      1
      1      5      0
Q =  3×3
      0      0      0
```

```
     0     0     0
     0     0     0
R = 3×3
     0     0     0
     0     0     0
     0     0     0
v = 3×1
     1
     0
     1
R = 3×3
    1.4142         0         0
         0         0         0
         0         0         0
Q = 3×3
    0.7071         0         0
         0         0         0
    0.7071         0         0
v = 3×1
     0
     1
     5
R = 3×3
    1.4142         0         0
         0    5.0990         0
         0         0         0
Q = 3×3
    0.7071         0         0
         0    0.1961         0
    0.7071    0.9806         0
v = 3×1
     2
     1
     0
R = 3×3
    1.4142         0         0
         0    5.0990         0
         0         0    2.2361
Q = 3×3
    0.7071         0    0.8944
         0    0.1961    0.4472
    0.7071    0.9806         0
```

5. Fundamental Spaces:

a) $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \end{bmatrix}$

Code:

```
A=[1,2,3;2,-1,1];
[R, pivot] = rref(A)
rank = length(pivot)
columnsp = A(:,pivot)
nullsp = null(A,'r')
rowsp = R(1:rank,:)'
leftnullsp = null(A','r')
```

```
R = 2×3
     1     0     1
     0     1     1

pivot = 1×2
     1     2

rank = 2
columnsp = 2×2
     1     2
     2    -1

nullsp = 3×1
    -1
    -1
     1

rowsp = 3×2
     1     0
     0     1
     1     1

leftnullsp =

  2×0 empty double matrix
```

Output:

```
R = 2×3
     1     0     1
     0     1     1
pivot = 1×2
     1     2
rank = 2
columnsp = 2×2
     1     2
     2    -1
nullsp = 3×1
    -1
    -1
     1
rowsp = 3×2
     1     0
     0     1
     1     1
leftnullsp =

  2×0 empty double matrix
```

b) $A = \begin{bmatrix} 2 & 5 & 9 \\ 1 & -1 & 0 \end{bmatrix}$

Code:

```
A=[2,5,9;1,-1,0];
[R, pivot] = rref(A)
rank = length(pivot)
columnsp = A(:,pivot)
nullsp = null(A,'r')
rowsp = R(1:rank,:)'
leftnullsp = null(A','r')
```

```
R = 2×3
      1.0000         0    1.2857
           0    1.0000    1.2857

pivot = 1×2
       1     2

rank = 2
columnsp = 2×2
       2     5
       1    -1

nullsp = 3×1
     -1.2857
     -1.2857
      1.0000

rowsp = 3×2
      1.0000         0
           0    1.0000
      1.2857    1.2857

leftnullsp =

  2×0 empty double matrix
```

Output:

```
R = 2×3
    1.0000         0    1.2857
         0    1.0000    1.2857
pivot = 1×2
     1     2
rank = 2
columnsp = 2×2
     2     5
     1    -1
nullsp = 3×1
   -1.2857
   -1.2857
    1.0000
rowsp = 3×2
    1.0000         0
         0    1.0000
    1.2857    1.2857
leftnullsp =

  2×0 empty double matrix
```

c) $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 4 \end{bmatrix}$

Code:

```
A=[1,0,0;1,0,4];
[R, pivot] = rref(A)
rank = length(pivot)
columnsp = A(:,pivot)
nullsp = null(A,'r')
rowsp = R(1:rank,:)'
leftnullsp = null(A','r')
```

```
R = 2x3
       1     0     0
       0     0     1

pivot = 1x2
       1     3

rank = 2
columnsp = 2x2
       1     0
       1     4

nullsp = 3x1
       0
       1
       0

rowsp = 3x2
       1     0
       0     0
       0     1

leftnullsp =

  2x0 empty double matrix
```

Output:

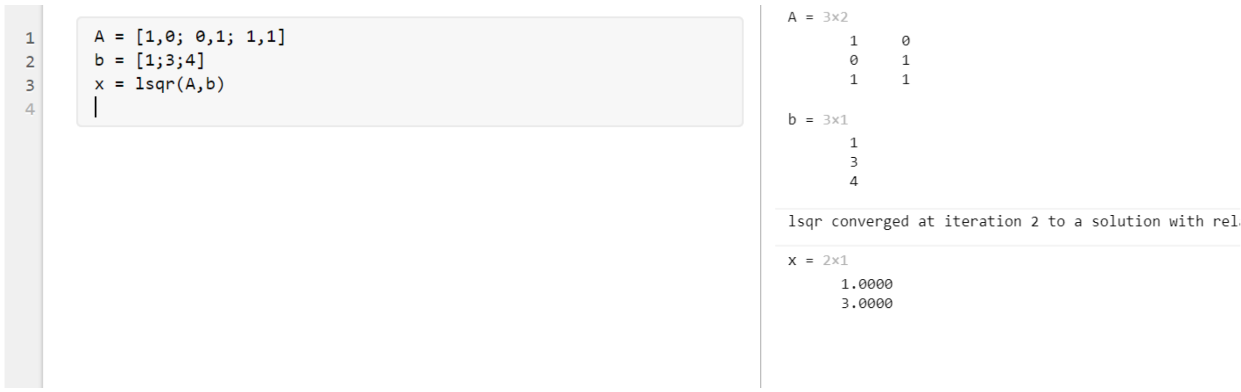
```
R = 2×3
     1     0     0
     0     0     1
pivot = 1×2
     1     3
rank = 2
columnsp = 2×2
     1     0
     1     4
nullsp = 3×1
     0
     1
     0
rowsp = 3×2
     1     0
     0     0
     0     1
leftnullsp =

  2×0 empty double matrix
```

6. Projection Matrices and least squares:

a) $A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$

Code:



```
A = [1,0; 0,1; 1,1]
b = [1;3;4]
x = lsqr(A,b)
```


Output:

```
A = 3×2
      1       0
      0       1
      1       1
b = 3×1
      1
      3
      4
lsqr converged at iteration 2 to a solution with relative residual 6.7e-17.
x = 2×1
    1.0000
    3.0000
```

b) $A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix}$

Code:

```
A = [1,0; 0,2; 3,1]
b = [1;0;4]
x = lsqr(A,b)
```


Output:

```
A = 3×2
      1       0
      0       2
```

```
     3      1
b = 3×1
     1
     0
     4
lsqr converged at iteration 2 to a solution with relative residual 0.076.
x = 2×1
    1.2927
    0.0244
```

c) $u = \begin{bmatrix} 1 \\ 7 \end{bmatrix}, v = \begin{bmatrix} -4 \\ 2 \end{bmatrix}$

Code:

```
u = [1;7]
v = [-4;2]
P = (v*transpose(v))/(transpose(v)*v)
P*u
```

```
P = 2×2
    0.8000   -0.4000
   -0.4000    0.2000
ans = 2×1
    -2
     1
```

Output:

```
u = 2×1
     1
     7
v = 2×1
    -4
     2
P = 2×2
    0.8000   -0.4000
   -0.4000    0.2000
ans = 2×1
    -2
     1
```

d) $A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \\ 1 & 1 \end{bmatrix}, \; b = \begin{bmatrix} 3 \\ 5 \\ 2.09 \end{bmatrix}$

Code:

```
A = [1,2; 3,2; 1,1]
b = [3;5;2.09]
x = lsqr(A,b)
```

```
A = 3×2
     1     2
     3     2
     1     1

b = 3×1
    3.0000
    5.0000
    2.0900

lsqr converged at iteration 2 to a solution with r

x = 2×1
    1.0000
    1.0100
```

Output:

```
A = 3×2
     1      2
     3      2
     1      1
b = 3×1
    3.0000
    5.0000
    2.0900
lsqr converged at iteration 2 to a solution with relative residual 0.014.
x = 2×1
    1.0000
    1.0100
```

e) $A = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 2 & -1 \\ 1 & 1 & 1 \end{bmatrix}$, $b = \begin{bmatrix} 3 \\ 4 \\ 6 \end{bmatrix}$

Code:

```
A = [1,2,1; 3,2,-1; 1,1,1]
b = [3;4;6]
x = lsqr(A,b)
```

```
A = 3×3
     1     2     1
     3     2    -1
     1     1     1

b = 3×1
     3
     4
     6

lsqr converged at iteration 3 to a solution with r

x = 3×1
    4.7500
   -3.0000
    4.2500
```

Output:

```
A = 3×3
     1      2      1
     3      2     -1
     1      1      1
b = 3×1
```

```
      3
      4
      6
lsqr converged at iteration 3 to a solution with relative residual 1.1e-14.
x = 3×1
    4.7500
   -3.0000
    4.2500
```

7. QR Decomposition with Gram-Schmidt:

a) $A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$

Code:

```
A = [1,1,0; 1,0,1; 0,1,1]
[Q,R] = qr(A)
```

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

Q = 3×3
   -0.7071    0.4082   -0.5774
   -0.7071   -0.4082    0.5774
         0    0.8165    0.5774

R = 3×3
   -1.4142   -0.7071   -0.7071
         0    1.2247    0.4082
         0         0    1.1547
```

Output:

```
A = 3×3
     1     1     0
     1     0     1
     0     1     1
Q = 3×3
   -0.7071    0.4082   -0.5774
   -0.7071   -0.4082    0.5774
         0    0.8165    0.5774
R = 3×3
   -1.4142   -0.7071   -0.7071
         0    1.2247    0.4082
         0         0    1.1547
```

b) $A = \begin{bmatrix} 1 & -1 & 4 \\ 1 & 4 & -2 \\ 1 & 4 & 2 \\ 1 & -1 & 0 \end{bmatrix}$

Code:

```
A = [1,-1,4; 1,4,-2; 1,4,2; 1,-1,0]
[Q,R] = qr(A)
```

```
A = 4x3
     1    -1     4
     1     4    -2
     1     4     2
     1    -1     0

Q = 4x4
   -0.5000    0.5000   -0.5000   -0.5000
   -0.5000   -0.5000    0.5000   -0.5000
   -0.5000   -0.5000   -0.5000    0.5000
   -0.5000    0.5000    0.5000    0.5000

R = 4x3
   -2.0000   -3.0000   -2.0000
         0   -5.0000    2.0000
         0         0   -4.0000
         0         0         0
```

Output:

```
A = 4x3
     1    -1     4
     1     4    -2
     1     4     2
     1    -1     0
Q = 4x4
   -0.5000    0.5000   -0.5000   -0.5000
   -0.5000   -0.5000    0.5000   -0.5000
   -0.5000   -0.5000   -0.5000    0.5000
   -0.5000    0.5000    0.5000    0.5000
R = 4x3
   -2.0000   -3.0000   -2.0000
         0   -5.0000    2.0000
         0         0   -4.0000
         0         0         0
```

c) $A = \begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix}$

Code:

```
A = [3,2,4; 2,0,2; 4,2,3]
[Q,R] = qr(A)
```

```
A = 4x3
     1    -1     4
     1     4    -2
     1     4     2
     1    -1     0

Q = 4x4
   -0.5000    0.5000   -0.5000   -0.5000
   -0.5000   -0.5000    0.5000   -0.5000
   -0.5000   -0.5000   -0.5000    0.5000
   -0.5000    0.5000    0.5000    0.5000

R = 4x3
   -2.0000   -3.0000   -2.0000
         0   -5.0000    2.0000
         0         0   -4.0000
         0         0         0
```

Output:

```
A = 4×3
     1    -1     4
     1     4    -2
     1     4     2
     1    -1     0
Q = 4×4
   -0.5000    0.5000   -0.5000   -0.5000
   -0.5000   -0.5000    0.5000   -0.5000
   -0.5000   -0.5000   -0.5000    0.5000
   -0.5000    0.5000    0.5000    0.5000
R = 4×3
   -2.0000   -3.0000   -2.0000
         0   -5.0000    2.0000
         0         0   -4.0000
         0         0         0
```

8. Eigen Values and Eigen Values:

a) $A = \begin{bmatrix} 1 & 1 & 3 \\ 1 & 5 & 1 \\ 3 & 1 & 1 \end{bmatrix}$

Code:

```
A = [1,1,3; 1,5,1; 3,1,1]
e = eig(A)
d = det(A)
p = prod(eig(A))
% Therefore det(A) == prod(eig(A))
s = sum(eig(A))
t = trace(A)
% Therefore sum(eig(A)) == trace(A)
[V,D] = eig(A)
```

```
A = 3×3
     1     1     3
     1     5     1
     3     1     1

e = 3×1
   -2.0000
    3.0000
    6.0000

d = -36
p = -36.0000
s = 7
t = 7
V = 3×3
   -0.7071    0.5774    0.4082
    0.0000   -0.5774    0.8165
    0.7071    0.5774    0.4082

D = 3×3
   -2.0000         0         0
         0    3.0000         0
         0         0    6.0000
```

Output:

```
A = 3×3
     1     1     3
     1     5     1
     3     1     1
e = 3×1
   -2.0000
    3.0000
    6.0000
d = -36
p = -36.0000
s = 7
```

```
t = 7
V = 3×3
   -0.7071    0.5774    0.4082
    0.0000   -0.5774    0.8165
    0.7071    0.5774    0.4082
D = 3×3
   -2.0000         0         0
         0    3.0000         0
         0         0    6.0000
```

b) $A = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ -1 & 1 & -1 \end{bmatrix}$

Code:

```
A = [1,-1,1; 1,0,0; -1,1,-1]
e = eig(A)
[V,D] = eig(A)
```

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

e = 3×1 complex
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.0000 + 0.0000i

V = 3×3 complex
   0.0000 + 0.5774i   0.0000 - 0.5774i ...
   0.5774 + 0.0000i   0.5774 + 0.0000i
  -0.0000 - 0.5774i  -0.0000 + 0.5774i

D = 3×3 complex
   0.0000 + 1.0000i   0.0000 + 0.0000i ...
   0.0000 + 0.0000i   0.0000 - 1.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i
```

Output:

```
A = 3×3
     1    -1     1
     1     0     0
    -1     1    -1
e = 3×1 complex
   0.0000 + 1.0000i
   0.0000 - 1.0000i
   0.0000 + 0.0000i
V = 3×3 complex
   0.0000 + 0.5774i   0.0000 - 0.5774i   0.0000 + 0.0000i
   0.5774 + 0.0000i   0.5774 + 0.0000i   0.7071 + 0.0000i
  -0.0000 - 0.5774i  -0.0000 + 0.5774i   0.7071 + 0.0000i
D = 3×3 complex
   0.0000 + 1.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 - 1.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
```

c) $A = \begin{bmatrix} 1 & 3 & 1 \\ 4 & 1 & 3 \\ 2 & 1 & 3 \end{bmatrix}$

Code:

```
A = [1,3,1; 4,1,3; 2,1,3]
e = eig(A)
[V,D] = eig(A)
```

```
A = 3×3
     1     3     1
     4     1     3
     2     1     3

e = 3×1
    6.1970
   -2.3132
    1.1162

V = 3×3
   -0.4986   -0.6863   -0.5816
   -0.6881    0.7168   -0.2774
   -0.5272    0.1234    0.7647

D = 3×3
    6.1970         0         0
         0   -2.3132         0
         0         0    1.1162
```

Output:

```
A = 3×3
     1     3     1
     4     1     3
     2     1     3
e = 3×1
    6.1970
   -2.3132
    1.1162
V = 3×3
   -0.4986   -0.6863   -0.5816
   -0.6881    0.7168   -0.2774
   -0.5272    0.1234    0.7647
D = 3×3
    6.1970         0         0
         0   -2.3132         0
         0         0    1.1162
```

d) $A = \begin{bmatrix} 2 & 3 & 4 \\ 5 & 3 & 2 \\ 1 & 2 & 2 \end{bmatrix}$

Code:

```
A = [2,3,4; 5,3,2; 1,2,2]
e = eig(A)
[V,D] = eig(A)
```

```
A = 3×3
     2     3     4
     5     3     2
     1     2     2

e = 3×1 complex
   8.0000 + 0.0000i
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i

V = 3×3 complex
   0.5926 + 0.0000i  -0.3873 + 0.2236i  -0.3873 - 0.2236i
   0.7293 + 0.0000i   0.7746 + 0.0000i   0.7746 + 0.0000i
   0.3419 + 0.0000i  -0.3873 - 0.2236i  -0.3873 + 0.2236i

D = 3×3 complex
   8.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.5000 + 0.8660i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.5000 - 0.8660i
```

Output:

```
A = 3×3
     2     3     4
     5     3     2
     1     2     2
e = 3×1 complex
   8.0000 + 0.0000i
  -0.5000 + 0.8660i
  -0.5000 - 0.8660i
V = 3×3 complex
   0.5926 + 0.0000i  -0.3873 + 0.2236i  -0.3873 - 0.2236i
   0.7293 + 0.0000i   0.7746 + 0.0000i   0.7746 + 0.0000i
   0.3419 + 0.0000i  -0.3873 - 0.2236i  -0.3873 + 0.2236i
D = 3×3 complex
   8.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.5000 + 0.8660i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.5000 - 0.8660i
```