Linear Algebra and its Applications

UE20MA251

Name: Naman Choudhary	SRN: PES2UG20CS209	Section: D

Gauss Elimination

Program Number	1
Program Qn	Solve the matrix by using gaussian elimination: x+2y+z=3 , 2x+y-2z=3 , -3x+y+z=-6

Source 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
    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 Screenshot

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



m = 2
A = 3×4
      1
      0
     -3
m = -3
A = 3×4
      1
      0
      0
m = -2.3333
A = 3×4
     1.0000
          0   -
          0
x = 3×1



summ = 0
summ = -3.0000
x = 3×1



summ = 0
summ = 2.9606e-16
x = 3×1



summ = 0.7500
x = 3×1
```

Program Number	1 a

Source Code

```
C=[1 5 1;1 1 -2;7 1 4]
b=[-8 2 -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
    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
```

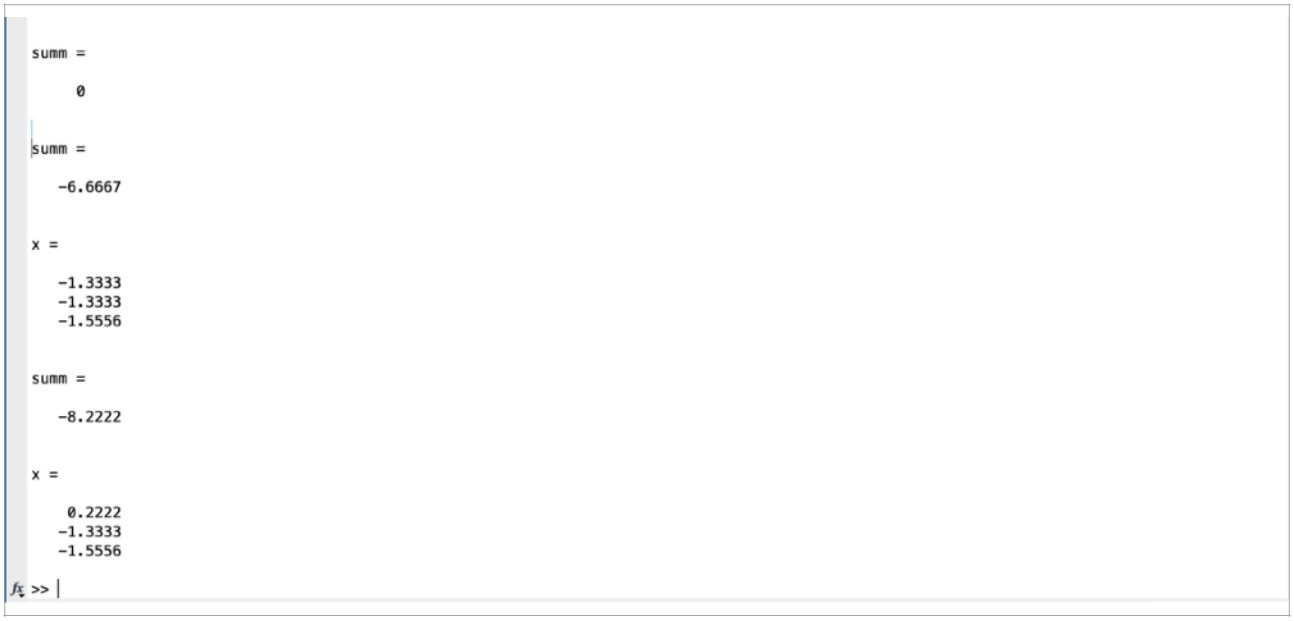
Output Screenshot

Command Window

```
>> P1a

C =

     1     5     1
     1     1    -2
     7     1     4


b =

    -8
     2
    -6


m =

     1


A =

     1     5     1    -8
     0    -4    -3    10
     7     1     4    -6


m =

     7


A =

     1     5     1    -8
     0    -4    -3    10
```

Command Window

```
     0   -34    -3    50


m =

    8.5000


A =

    1.0000    5.0000    1.0000   -8.0000
         0   -4.0000   -3.0000   10.0000
         0         0   22.5000  -35.0000


x =

         0
         0
   -1.5556


summ =

     0


summ =

    4.6667


x =

         0
   -1.3333
   -1.5556
```



Program Number	1 b

Source Code

```
C=[4 5 8;1 5 -2;-3 5 6]
b=[2 1 -2]'
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
    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 Screenshot

```
C = 3×3
     4     5     8
     1     5    -2
    -3     5     6

b = 3×1
     2
     1
    -2

m = 0.2500
A = 3×4
    4.0000    5.0000    8.0000    2.0000
         0    3.7500   -4.0000    0.5000
   -3.0000    5.0000    6.0000   -2.0000

m = -0.7500
A = 3×4
    4.0000    5.0000    8.0000    2.0000
         0    3.7500   -4.0000    0.5000
         0    8.7500   12.0000   -0.5000

m = 2.3333
A = 3×4
    4.0000    5.0000    8.0000    2.0000
         0    3.7500   -4.0000    0.5000
         0         0   21.3333   -1.6667

x = 3×1
         0
         0
   -0.0781

summ = 0
summ = 0.3125
```

```
x = 3×1
            0
       0.0500
      -0.0781

summ = 0
summ = 0.2500
x = 3×1
       0.4375
       0.0500
      -0.0781

summ = -0.3750
x = 3×1
       0.5938
       0.0500
      -0.0781
```

Gauss Jordan

Program Number	2
Program Qn	Find the inverse of A using Gauss-Jordon Method where A = 1 1 1 4 3 -1 3 5 3
Source 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(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 Screenshot

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

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
    1.0000    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 = 3×6
    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 = 3×6
    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 = 3×6
    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 = 3×6
      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 = 3×3
      1.4000    0.2000   -0.4000
     -1.5000         0    0.5000
      1.1000   -0.2000   -0.1000
```

Program Number	2 a

Source Code

```
A=[2,4,5;5,1,1;7,8,9]
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 Screenshot

```
A = 3×3
     2     4     5
     5     1     1
     7     8     9

Aug = 3×6
     2     4     5     1     0     0
     5     1     1     0     1     0
     7     8     9     0     0     1

Aug = 3×6
    2.0000    4.0000    5.0000    1.0000         0         0
         0   -9.0000  -11.5000   -2.5000    1.0000         0
    7.0000    8.0000    9.0000         0         0    1.0000

Aug = 3×6
    2.0000    4.0000    5.0000    1.0000         0         0
         0   -9.0000  -11.5000   -2.5000    1.0000         0
         0   -6.0000   -8.5000   -3.5000         0    1.0000

Aug = 3×6
    2.0000    4.0000    5.0000    1.0000         0         0
         0   -9.0000  -11.5000   -2.5000    1.0000         0
         0         0   -0.8333   -1.8333   -0.6667    1.0000

Aug = 3×6
    2.0000    4.0000         0  -10.0000   -4.0000    6.0000
         0   -9.0000         0   22.8000   10.2000  -13.8000
         0         0   -0.8333   -1.8333   -0.6667    1.0000

Aug = 3×6
    2.0000         0         0    0.1333    0.5333   -0.1333
         0   -9.0000         0   22.8000   10.2000  -13.8000
         0         0   -0.8333   -1.8333   -0.6667    1.0000

Aug = 3×6
    1.0000         0         0    0.0667    0.2667   -0.0667
         0   -9.0000         0   22.8000   10.2000  -13.8000
         0         0   -0.8333   -1.8333   -0.6667    1.0000

Aug = 3×6
    1.0000         0         0    0.0667    0.2667   -0.0667
         0    1.0000         0   -2.5333   -1.1333    1.5333
         0         0   -0.8333   -1.8333   -0.6667    1.0000
```

```
Aug = 3×6
      1.0000         0         0    0.0667    0.2667   -0.0667
           0    1.0000         0   -2.5333   -1.1333    1.5333
           0         0    1.0000    2.2000    0.8000   -1.2000

B = 3×3
      0.0667    0.2667   -0.0667
     -2.5333   -1.1333    1.5333
      2.2000    0.8000   -1.2000
```

Program Number	2 b

Source Code

```
A=[2,5,7;-2,8,2;3,5,6]
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 Screenshot

```
A = 3×3
     2     5     7
    -2     8     2
     3     5     6

Aug = 3×6
     2     5     7     1     0     0
    -2     8     2     0     1     0
     3     5     6     0     0     1

Aug = 3×6
     2     5     7     1     0     0
     0    13     9     1     1     0
     3     5     6     0     0     1

Aug = 3×6
    2.0000    5.0000    7.0000    1.0000         0         0
         0   13.0000    9.0000    1.0000    1.0000         0
         0   -2.5000   -4.5000   -1.5000         0    1.0000

Aug = 3×6
    2.0000    5.0000    7.0000    1.0000         0         0
         0   13.0000    9.0000    1.0000    1.0000         0
         0         0   -2.7692   -1.3077    0.1923    1.0000

Aug = 3×6
    2.0000    5.0000    0.0000   -2.3056    0.4861    2.5278
         0   13.0000         0   -3.2500    1.6250    3.2500
         0         0   -2.7692   -1.3077    0.1923    1.0000

Aug = 3×6
    2.0000         0    0.0000   -1.0556   -0.1389    1.2778
         0   13.0000         0   -3.2500    1.6250    3.2500
         0         0   -2.7692   -1.3077    0.1923    1.0000

Aug = 3×6
    1.0000         0    0.0000   -0.5278   -0.0694    0.6389
         0   13.0000         0   -3.2500    1.6250    3.2500
         0         0   -2.7692   -1.3077    0.1923    1.0000

Aug = 3×6
    1.0000         0    0.0000   -0.5278   -0.0694    0.6389
         0    1.0000         0   -0.2500    0.1250    0.2500
         0         0   -2.7692   -1.3077    0.1923    1.0000
```

```
Aug = 3×6
      1.0000         0    0.0000   -0.5278   -0.0694    0.6389
           0    1.0000         0   -0.2500    0.1250    0.2500
           0         0    1.0000    0.4722   -0.0694   -0.3611

B = 3×3
     -0.5278   -0.0694    0.6389
     -0.2500    0.1250    0.2500
      0.4722   -0.0694   -0.3611
```

LU Decomposition

Program Number	3
Program Qn	LU decomposition of A were A = 1 1 -1 3 5 6 7 8 9

Source Code

```
%LU Decomposition
Ab = [1 1 -1;3 5 6;7 8 9];
%% 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 Screenshot

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

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

Program Number	3 a

Source Code

```
%LU Decomposition
Ab = [3 5 -1;3 8 6;9 8 1];
%% 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 Screenshot

```
L = 3×3
      1.0000         0         0
      1.0000    1.0000         0
      3.0000   -2.3333    1.0000

U = 3×3
      3.0000    5.0000   -1.0000
           0    3.0000    7.0000
           0         0   20.3333
```

Program Number	3 b

Source Code

```
%LU Decomposition
Ab = [3 1 -1;6 4 6;-9 3 7];
%% 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 Screenshot

```
L = 3×3
     1     0     0
     2     1     0
    -3     3     1

U = 3×3
     3     1    -1
     0     2     8
     0     0   -20
```

Gram Schmidt Orthogonalization

Program Number	4
Program Qn	Apply the Gram-Schmidt process to the vectors (1,1,2), (0,0,1) and (1,0,0) to produce a set of Orthonormal vectors.

Source 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: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 Screenshot

```
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    0.7071         0
       0         0         0
       0         0         0

v = 3×1
  0.5000
       0
 -0.5000
```

```
R = 3×3
      1.4142    0.7071         0
           0    0.7071         0
           0         0         0

Q = 3×3
      0.7071    0.7071         0
           0         0         0
      0.7071   -0.7071         0

v = 3×1
       2
       1
       0

R = 3×3
      1.4142    0.7071    1.4142
           0    0.7071         0
           0         0         0

v = 3×1
      1.0000
      1.0000
     -1.0000

R = 3×3
      1.4142    0.7071    1.4142
           0    0.7071    1.4142
           0         0         0

v = 3×1
     -0.0000
      1.0000
      0.0000

R = 3×3
      1.4142    0.7071    1.4142
           0    0.7071    1.4142
           0         0    1.0000

Q = 3×3
      0.7071    0.7071   -0.0000
           0         0    1.0000
      0.7071   -0.7071    0.0000
```

Program Number	4 a
Source Code	

```
A=[1,0,1;0,0,2;3,1,0]
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 Screenshot

```
Q = 3×3
      0.3162         0         0
           0         0         0
      0.9487         0         0

v = 3×1
       0
       0
       1

R = 3×3
      3.1623    0.9487         0
           0         0         0
           0         0         0

v = 3×1
     -0.3000
           0
      0.1000

R = 3×3
      3.1623    0.9487         0
           0    0.3162         0
           0         0         0

Q = 3×3
      0.3162   -0.9487         0
           0         0         0
      0.9487    0.3162         0

v = 3×1
       1
       2
       0

R = 3×3
      3.1623    0.9487    0.3162
           0    0.3162         0
           0         0         0

v = 3×1
      0.9000
      2.0000
     -0.3000

R = 3×3
      3.1623    0.9487    0.3162
```

```
R = 3×3
    3.1623    0.9487    0.3162
         0    0.3162   -0.9487
         0         0         0

v = 3×1
    0.0000
    2.0000
    0.0000

R = 3×3
    3.1623    0.9487    0.3162
         0    0.3162   -0.9487
         0         0    2.0000

Q = 3×3
    0.3162   -0.9487    0.0000
         0         0    1.0000
    0.9487    0.3162    0.0000
```

Program Number	4 b

Source Code

```
A=[3,0,2;0,0,2;0,4,0]
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 Screenshot

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

R = 3×3
   3.1623         0         0
        0         0         0
        0         0         0

Q = 3×3
   0.3162         0         0
        0         0         0
   0.9487         0         0

v = 3×1
     0
     0
     1

R = 3×3
   3.1623    0.9487         0
        0         0         0
        0         0         0

v = 3×1
```

```
Q = 3×3
      0.3162   -0.9487         0
           0         0         0
      0.9487    0.3162         0

v = 3×1
       1
       2
       0

R = 3×3
      3.1623    0.9487    0.3162
           0    0.3162         0
           0         0         0

v = 3×1
      0.9000
      2.0000
     -0.3000

R = 3×3
      3.1623    0.9487    0.3162
           0    0.3162   -0.9487
           0         0         0

v = 3×1
      0.0000
      2.0000
      0.0000

R = 3×3
      3.1623    0.9487    0.3162
           0    0.3162   -0.9487
           0         0    2.0000

Q = 3×3
      0.3162   -0.9487    0.0000
           0         0    1.0000
      0.9487    0.3162    0.0000
```

In Built

Eigen value Eigen vector

Program Number	5 a
Program Qn	Find the eigenvalues and the corresponding eigenvectors of the matrix A= [1,1,3;1,5,1;3,1,1]
Source Code	

```
A=[1,1,3;1,5,1;3,1,1]
e=eig(A)
det(A)
prod(eig(A))
sum(eig(A))
trace(A)
[V,D]=eig(A)
```

Output Screenshot

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

e = 3×1
    -2.0000
     3.0000
     6.0000

ans = -36
ans = -36.0000
ans = 7.0000
ans = 7
V = 3×3
    -0.7071    0.5774    0.4082
          0   -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
```

QR Factorisation

Program Number	5 b
Program Qn	Find the QR factorisation of the matrix A where A= 1 1 0 1 0 1 0 1 1

Source Code

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

Output Screenshot

```
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
```

Projection of least squares

Program Number	5 c
Program Qn	Projection of least squares

Source Code

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

Output Screenshot

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

b = 3×1
     1
     3
     4

eration 2 to a solution with relative residual 4.3e-17.

x = 2×1
     1
     3
```

Four fundamental subspaces

Program Number	5 d
Program Qn	Four fundamental subspaces

Source Code

```
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 Screenshot

```
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
```