1.Gaussian Elimination

X+2y-z=3,2x+y-2z=3,-3x+y+z=-6

```
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

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)
```

3.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)
```

4.FOUR FUNDAMENTAL SUBSPACES

```
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')
```

5.Gram Schmidt orthogonalization

```
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
```

6.Projection by least squares

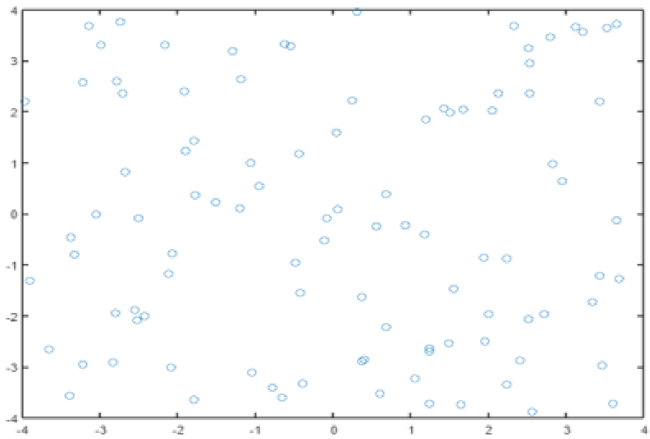
a) A=[1,0;0,1;1,1]
b=[1;3;4]
x = lsqr(A,b)

b) Find the point on the plane x+y-z=0 that is closest to (2,1,0)
```
syms c
P=[2,1,0]+c*[1,1,-1]
s=1*(c+2)+1*(c+1)-1(-c)==0
s1=solve(s,c)
p=[2,1,0]+s1*[1,1,-1]
```

c) Find the point on the plane 3x+4y+z=1 that is closest to (1,0,1)
```
syms c
P=[1,0,1]+c*[3,4,1]
s=3*(1+3*c)+4*(4*c)+(1+c)==1
s1=solve(s,c)
p=[1,0,1]+s1*[3,4,1]
```

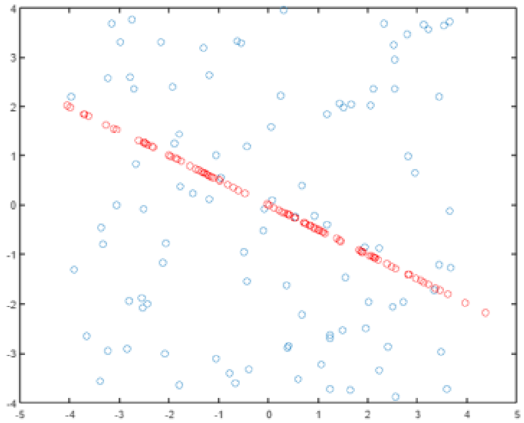
d) Let onto and find P, the matrix that will project any matrix onto the vector v. Use the result to find projv u. Code:
```
u=[1;7]
v=[-4;2]
P=(v*transpose(v))/(transpose(v)*v)
P*u
```

e) Projecting a lot of vectors on a single vector:
Code:
```
u=8*rand(2,100)-4;
x=u(1,: )
y=u(2,: )
plot(x,y,'o')
```
In the below figure I have generated a 100 random vectors.
In this figure each circle represents the tip of a vector whose tail begins at the origin.



Next , I will take the projection matrix P to project each of the 100 2 by 1 vectors in matrix U onto the vector v, Projection matrices and least squares: Code:
```
P=[0.8,-0.4;-0.4,0.2]
Pu=P*u;
```

```
x=Pu(1,:)
y=Pu(2,:)
hold on
plot(x,y,'o')
```



Here each vector in the matrix u is projected onto a line in the direction of the vector v=[-1;2]

7.QR FACTORIZATION

a) Find QR factorization of the matrix

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

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

b)QR Factorization of Pascal Matrix

```
A = sym(pascal(3))

[Q,R] = qr(A)

isAlways(A == Q*R)
```

c) QR Decomposition to Solve Matrix Equation of the form Ax=b

```
A = sym(invhilb(5))

b = sym([1:5])

[C,R] = qr(A,b);

X = R\C

isAlways(A*X == b)
```

8.Eigenvalues and Eigenvectors

a).computing detrminant and trace

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

e=eig(A)
```

```
det(A)

prod(eig(A))

det(A)=prod(eig(A))

 sum(eig(A))

trace(A)

[V,D]=eig(A)

A*V-V*D
```

b).Computing Eigenvalues and Eigenvectors

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

e=eig(A)

[V,D]=eig(A)


A*V(:,1)          % This command will gives you First eigenvectore

D(1,1)*V(:,1)    % First eigenvalue
```

If you replace 2 or 3 in place of 1, that will create second or third eigenvalue and eigenvectors respectively.

c)For similar Matrices

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

B=[1,-1,1;1,0,0;-1,1,-1]

e=eig(A,B)

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

[V,D,W]=eig(A,B)
```

e = eig(A,B) returns a column vector containing the generalized eigenvalues of square matrices A and B.

[V,D] = eig(A,B) returns diagonal matrix D of generalized eigenvalues and full matrix V whose columns are the corresponding right eigenvectors,

so that A*V = B*V*D.

[V,D,W] = eig(A,B) also returns full matrix W whose columns are the

corresponding left eigenvectors, so that W'*A = D*W'*B.

The generalized eigenvalue problem is to determine the solution to the equation $Av = \lambda Bv$, where A and B are n-by-n matrices, v is a column vector of length n, and λ is a scalar.

The values of λ that satisfy the equation are the generalized eigenvalues.

The corresponding values of v are the generalized right eigenvectors.

The left eigenvectors, w, satisfy the equation $w'A = \lambda w'B$.