**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 :** KEERTHI

**SRN :** PES1UG20ME040

**Branch : Mechanical Engineering**

**Semester & Section : Semester IV Section A**

**FOR OFFICE USE ONLY:**

**Marks Allotted :/ 05**

Name of the Course Instructor : **Prof. Jyothsna K M**

Signature of the Course Instructor : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Eigenvalues and Eigenvectors**

1. Find the eigenvalues and the corresponding eigenvectors of the matrix

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

CODE:

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

e=eig(A)

det(A)

prod(eig(A))

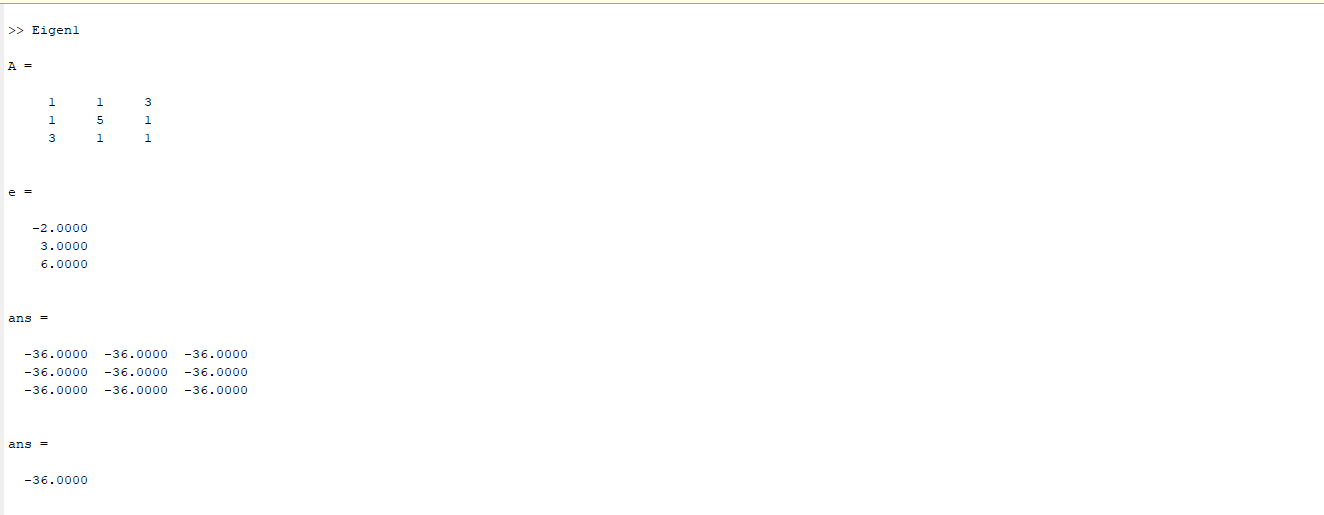
det(A)=prod(eig(A))

sum(eig(A))

trace(A)

[V,D]=eig(A)

OUTPUT:



2. Find the eigenvalue and eigenvectors of the matrix

A=[2,2,1;1,3,1;1,2,2] and B=[1,-1,1;1,0,0;-1,1,-1]

CODE:

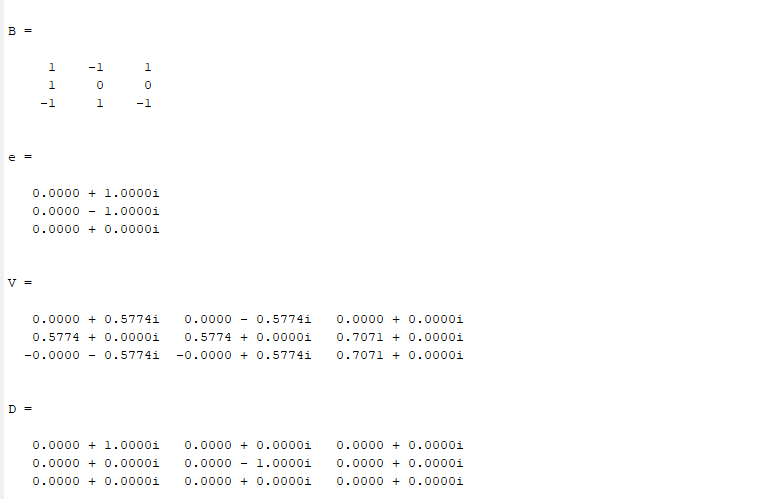
A=[2,2,1;1,3,1;1,2,2]

e=eig(A)

[V,D]=eig(A)

OUTPUT:





3. Find the Eigenvalue and the corresponding Eigenvectors of A=[1,-1,1;1,0,0;-1,1,-1].

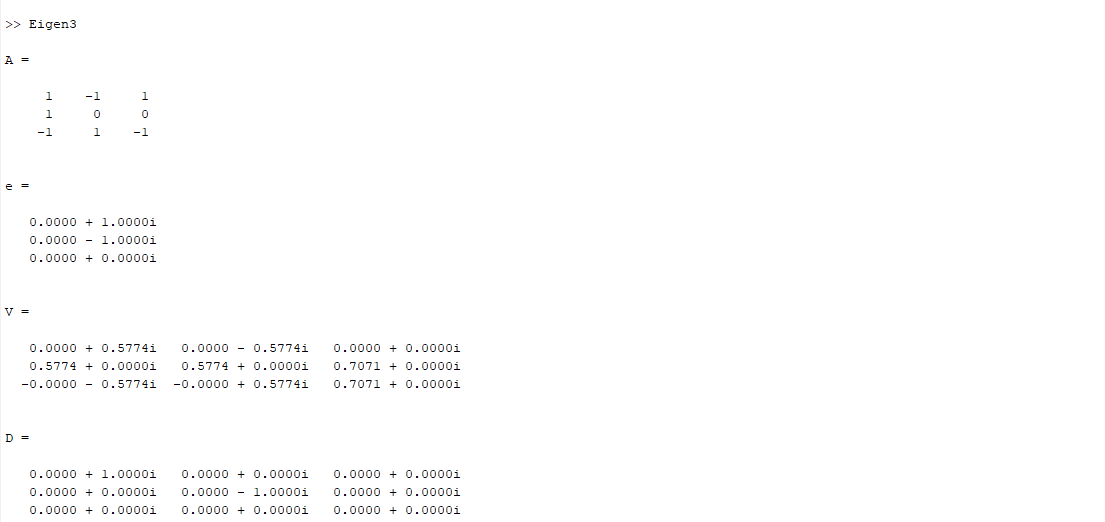
CODE:

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

e=eig(A)

[V,D]=eig(A)

OUTPUT:



4 . Find the Eigenvalue and the corresponding Eigenvectors of A=[1,3,1;4,1,3;2,1,3]

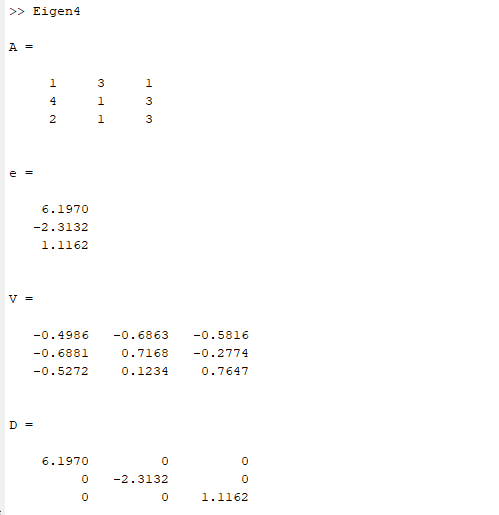
CODE:

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

e=eig(A)

[V,D]=eig(A)

OUTPUT:



5 . Find the Eigenvalue and the corresponding Eigenvectors of A=[2,3,4;5,3,2;1,2,2]

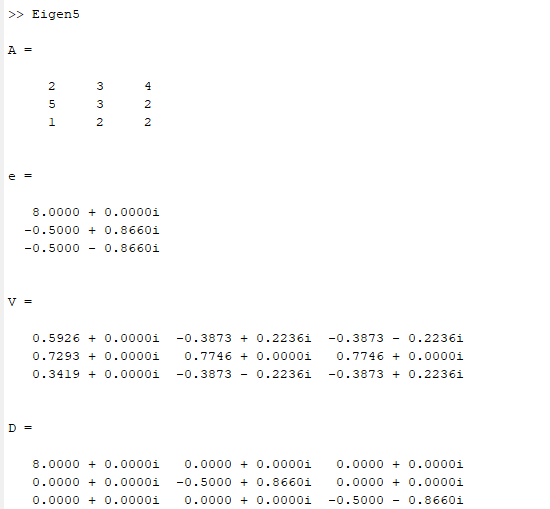
CODE:

A=[2,3,4;5,3,2;1,2,2]

e=eig(A)

[V,D]=eig(A)

OUTPUT:



**Gauss Jordan Method to find Inverse**

1. A=[1,1,1;4,3,-1;3,5,3]

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

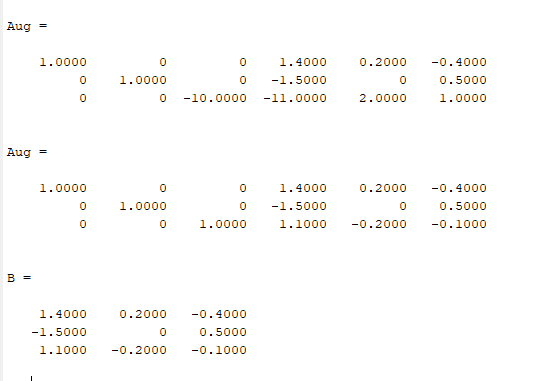
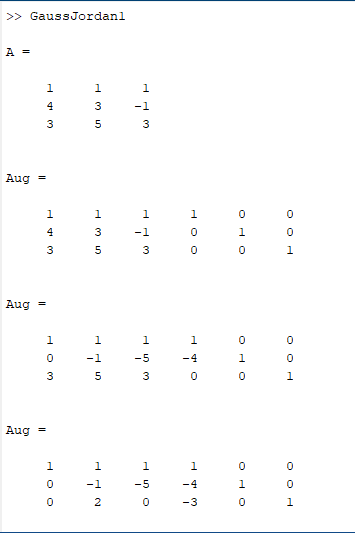
for j = 1:n

Aug(j,:) = Aug(j,:)/Aug(j,j)

end

B = Aug(:,n+1:2\*n)

OUTPUT:



1. A=[1,2,3;1,2,4;1,1,5]

CODE:

A=[1,2,3;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(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j)/Aug(j,j)\*Aug(j,:)

end

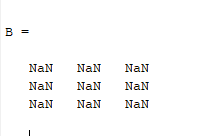
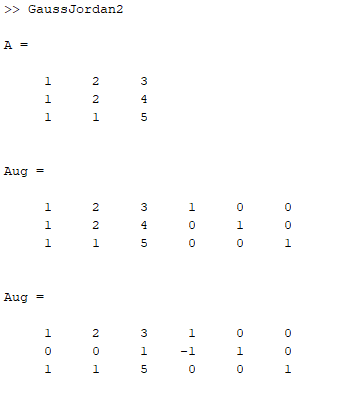
for j = 1:n

Aug(j,:) = Aug(j,:)/Aug(j,j)

end

B = Aug(:,n+1:2\*n)

OUTPUT:



1. A=[-1,2,6;-1,-2,4;-1,1,5]

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(1:j-1,:) = Aug(1:j-1,:) - Aug(1:j-1,j)/Aug(j,j)\*Aug(j,:)

end

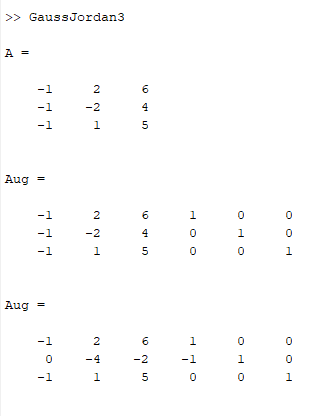
for j = 1:n

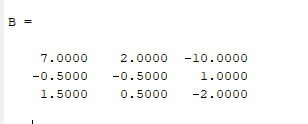
Aug(j,:) = Aug(j,:)/Aug(j,j)

end

B = Aug(:,n+1:2\*n)

OUTPUT:





**LU Decomposition Method**

1. Ab = [1,1,-3;3,5,6;7,8,9];

CODE:

Ab = [1,1,-3;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;

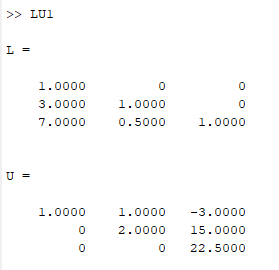
alpha = Ab(i,2)/Ab(2,2);

L(i,2) = alpha

Ab(i,:) = Ab(i,:) - alpha\*Ab(2,:);

U = Ab(1:n,1:n)

OUTPUT:



1. Ab = [1,1,3;1,2,4;1,1,5];

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;

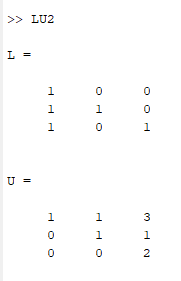
alpha = Ab(i,2)/Ab(2,2);

L(i,2) = alpha

Ab(i,:) = Ab(i,:) - alpha\*Ab(2,:);

U = Ab(1:n,1:n)

OUTPUT:



1. Ab = [-1,4,6;0,-2,4;0,0,5];

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;

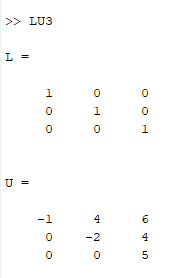
alpha = Ab(i,2)/Ab(2,2);

L(i,2) = alpha

Ab(i,:) = Ab(i,:) - alpha\*Ab(2,:);

U = Ab(1:n,1:n)

OUTPUT:



**Gram - Schmidt Algorithm**

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

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)

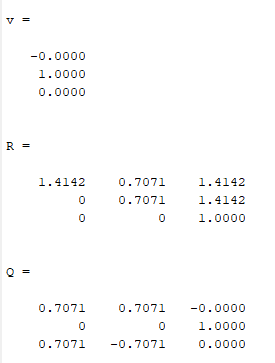
end

R(j,j)=norm(v)

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

end

OUTPUT:



1. Apply the Gram-Schmidt process to the vectors a=(0,1,1,1), b=(1,1,-1,0) and c=(1,0,2,-1).

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:j-1

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

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

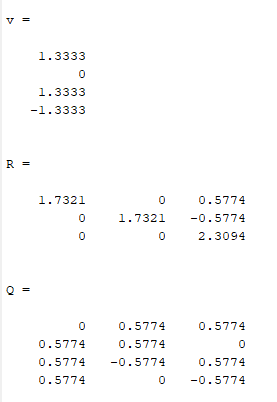
end

R(j,j)=norm(v)

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

end

OUTPUT:



**Fundamental Subspaces**

1. Bases of four fundamental vector spaces of matrix A.

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

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')

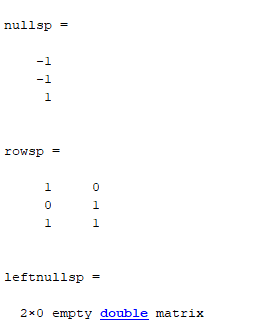
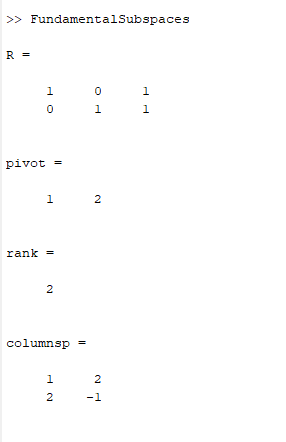
% basis of the row space of A

rowsp = R(1:rank,:)'

% basis of the left nullspace of A

leftnullsp = null(A','r')

OUTPUT:



**Projections by Least Square**

1. Find the projection for the matrix: A = [1,0;0,1;1,1] ; x = [u,v] and b = [1;3;4]

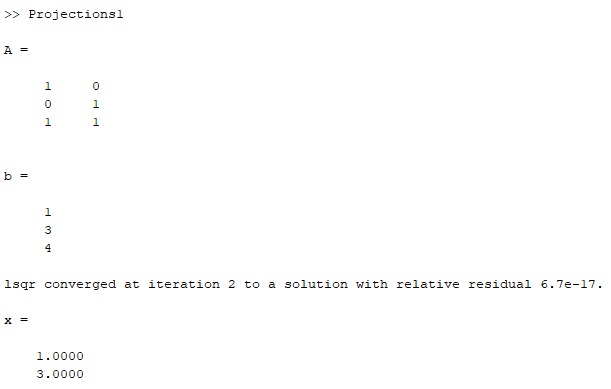
CODE:

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

b=[1;3;4]

x = lsqr(A,b)

OUTPUT:



1. Find the projection for the matrix: A = [1,0;0,2;3,1] ; x = [u,v] and b = [1;0;4]

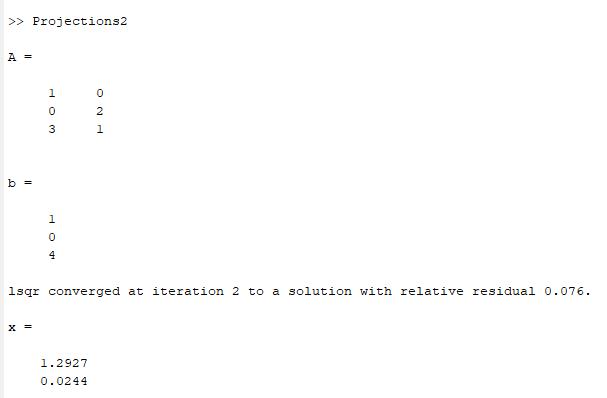
CODE:

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

b=[1;0;4]

x = lsqr(A,b)

OUTPUT:



1. Find the point on a plane x + y - z =0 that is closest to (2,1,0).

CODE:

syms c

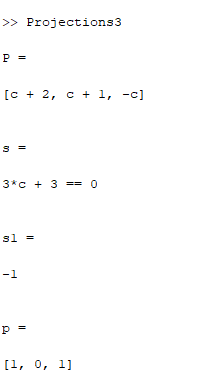
P=[2,1,0]+c\*[1,1,-1]

s=1\*(c+2)+1\*(c+1)-1\*(-c)==0

s1=solve(s,c)

p=[2,1,0]+s1\*[1,1,-1]

OUTPUT:



1. Find the point on a plane 3x +4 y + z = 1 that is closest to (1,0,1).

CODE:

syms c

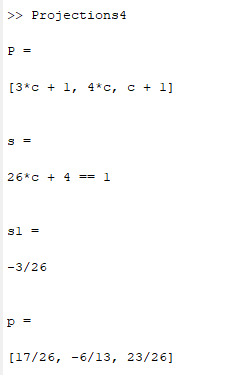
P=[1,0,1]+c\*[3,4,1]

s=3\*(1+3\*c)+4\*(4\*c)+(1+c)==1

s1=solve(s,c)

p=[1,0,1]+s1\*[3,4,1]

OUTPUT:



1. Let u = [1;7] onto v = [-4;2] and find P, the matrix that will project any matrix onto vector v. Use the result to find projection u.

CODE:

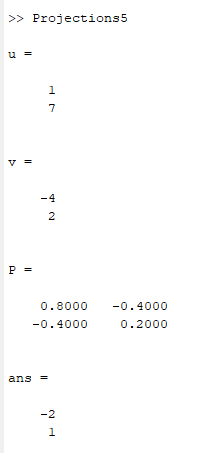
u=[1;7]

v=[-4;2]

P=(v\*transpose(v))/(transpose(v)\*v)

P\*u

OUTPUT:



1. Projecting a lot of vector on a single vector:

CODE:

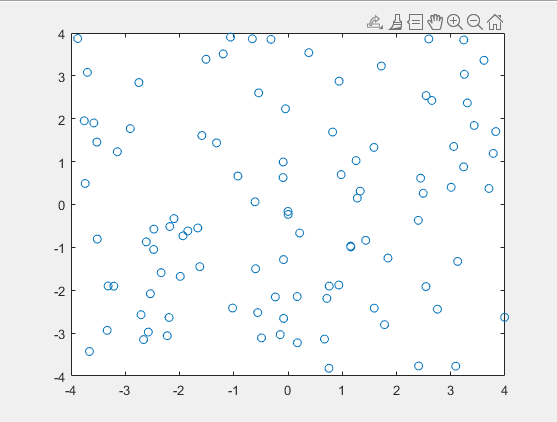
u=8\*rand(2,100)-4;

x=u(1,: )

y=u(2,: )

plot(x,y,'o')

OUTPUT:



1. Take the projection matrix P to project each of the 100 2 by 1 vectors in matrix U onto the vector v.

CODE:

P=[0.8,-0.4;-0.4,0.2]

Pu=P\*u;

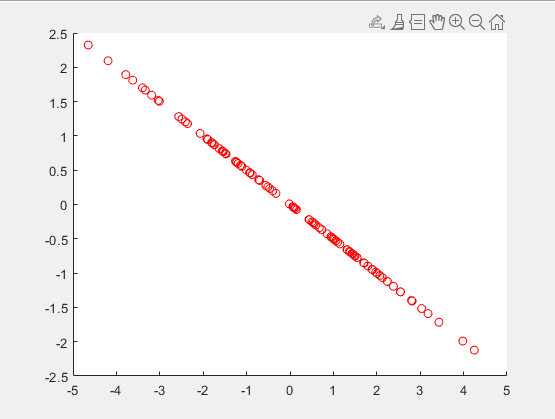
x=Pu(1,:)

y=Pu(2,:)

hold on

plot(x,y,'ro‘)

OUTPUT:



1. Find the least square fit for this system A = [1,2,1;3,2,-2;1,1,7] v = [x;y;z] b = [3;5;21.09]

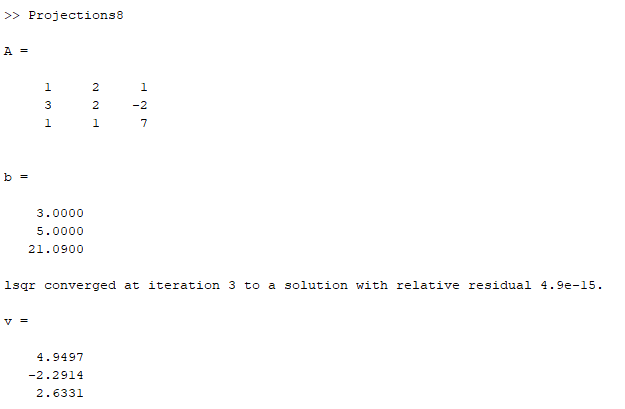
CODE:

A = [1,2,1;3,2,-2;1,1,7]

b = [3;5;21.09]

v = lsqr(A,b)

OUTPUT:



1. Find the least square fit for this system A = [1,2;3,2;1,1] v = [x,y] b = [3;5;2.09]

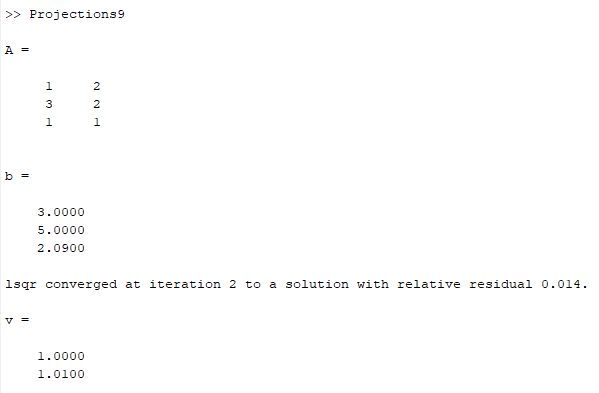
CODE:

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

b = [3;5;2.09]

v = lsqr(A,b)

OUTPUT:



1. Find the point on a plane 13x + 4y +z = 1 that is closest to (1,-1,1)

CODE:

syms c

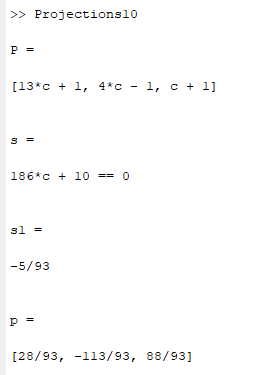
P=[1,-1,1]+c\*[13,4,1]

s=13\*(1+13\*c)+4\*(-1+4\*c)+1(1+c)==0

s1=solve(s,c)

p=[1,-1,1]+s1\*[13,4,1]

OUTPUT:



**QR Factorization**

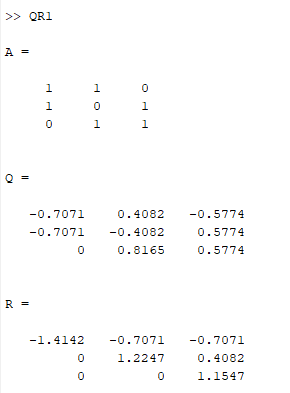
1. Find QR factorization of the matrix A = [1,1,0;1,0,1;0,1,1]

CODE:

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

[Q,R]=qr(A)

OUTPUT:



1. QR Factorization of Pascal Matrix

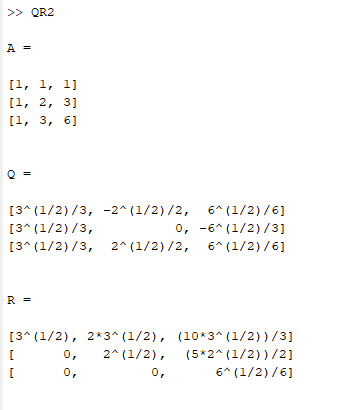
CODE:

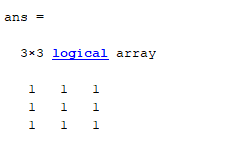
A = sym(pascal(3))

[Q,R] = qr(A)

isAlways(A == Q\*R)

OUTPUT:





1. QR Decomposition to Solve Matrix Equation of the form Ax=b

CODE:

A = sym(invhilb(5))

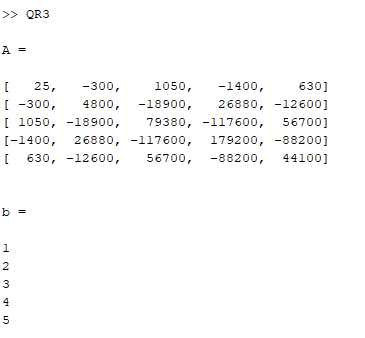
b = sym([1:5]‘)

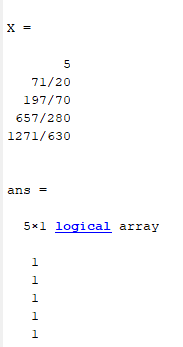
[C,R] = qr(A,b);

X = R\C

isAlways(A\*X == b)

OUTPUT:





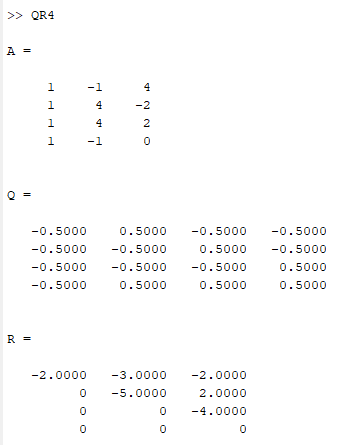
1. Find QR Decomposition of A = [1,-1,4;1,4,-2;1,4,2;1,-1,0]

CODE:

A = [1,-1,4;1,4,-2;1,4,2;1,-1,0]

[Q,R]=qr(A)

OUTPUT:



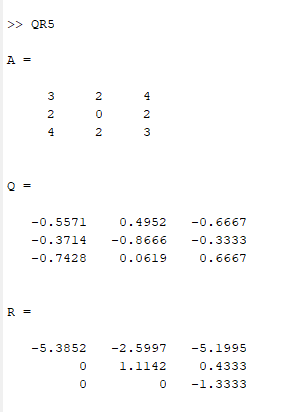
1. Find QR Decomposition of A = [3,2,4;2,0,2;4,2,3]

CODE:

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

[Q,R]=qr(A)

OUTPUT:



1. Find QR Decomposition of A = [1,2,4;3,8,9;5,7,3]

CODE:

A = [1,2,4;3,8,9;5,7,3]

[Q,R]=qr(A)

OUTPUT:

