> Note : No semicolon means print

1 - Eigenvalues and Eigenvectors (Inbuilt)

Preface :

The `eig` function in matlab can be used to get Eigenvalues and Eigenvectors.

Questions :

Find the Eigenvalues and Eigenvectors for the following matrices :

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

Code :

```
A=[3,1;0,3];
eig(A)
% for verifying results : trace(A) = sum(eig(A)) and prod(eig(A)) = det(A)
trace(A)
sum(eig(A))
%
prod(eig(A))
det(A)
% eig return eigen vectors as well
[V,D]=eig(A);
V
```

Screenshots :

```
>> eigen1

ans =

     3
     3


ans =

     6


ans =

     6


ans =

     9


ans =

     9


V =

    1.0000   -1.0000
         0    0.0000
```

2. A=[1,2,3;4,5,6;0,0,0]

Code :

```
A=[1,2,3;4,5,6;0,0,0];
eig(A)
% for verifying results : trace(A) = sum(eig(A)) and prod(eig(A)) = det(A)
trace(A)
sum(eig(A))
%
prod(eig(A))
det(A)
% eig return eigen vectors as well
[V,D]=eig(A);
V
```

Screenshots :

```
>> eigen2

ans =

   -0.4641
    6.4641
         0


ans =

     6


ans =

     6


ans =

     0


ans =

     0


V =

   -0.8069   -0.3437    0.4082
    0.5907   -0.9391   -0.8165
         0         0    0.4082
```

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

Code :

```
A=[1,1,3;1,5,1;3,1,1];
eig(A)
% for verifying results : trace(A) = sum(eig(A)) and prod(eig(A)) = det(A)
trace(A)
sum(eig(A))
%
prod(eig(A))
det(A)
% eig return eigen vectors as well
[V,D]=eig(A);
V
```

Screenshots :

```
>> eigen3

ans =

   -2.0000
    3.0000
    6.0000


ans =

     7


ans =

    7.0000


ans =

  -36.0000


ans =

   -36


V =

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

2 - Projection matrices and least squared method (Inbuilt)

Preface :

- projection can be found using `lsqr` function in matlab
- Projection matrices can be found using : a*transpose(a)/transpose(a)*a
- `lsqr` is also used for finding least square

Questions :

1. Find projection of b on A given, A=[1,0;0,1;1,1], b=[1,3,4]

Code :

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

Screenshots :

```
>> proj1
lsqr converged at iteration 2

ans =

     1
     3
```

2. find the projection a given,A=[1,0,0;0,1,0;0,0,1]

code :

```
A=[1,0,0;0,1,0;0,0,1];
(A*transpose(A))/(transpose(A)*A)
```

screenshots :

```
>> proj2

ans =

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

3. find the least square fit for the following equations : X+2Y=3 3X+2Y=5 X+Y=2.09

code :

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

screenshots :

```
>> proj3
lsqr converged at iteration 2

ans =

    1.0000
    1.0100
```

3 - QR Factorisation (Inbuilt)

Preface :

- Any matrix can be factorised into Q and R using `qr` function in matlab

Questions :

Find the QR factors for the following :

1. 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);
Q
R
```

Screenshots :

```
>> qr1

Q =

   -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 =

   -2.0000   -3.0000   -2.0000
         0   -5.0000    2.0000
         0         0   -4.0000
         0         0         0
```

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

code :

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

screenshots :

```
>> qr2

A =

     3     2     4
     2     0     2
     4     2     3


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
```

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

code :

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

screenshots :

```
>> qr3

A =

     1     0
     0     1


Q =

     1     0
     0     1


R =

     1     0
     0     1
```

4 - QR Factorisation Using Gram-Schmidt process (Not Inbuilt)

Preface :

- here we factorise the matrices without using the qr command

Questions :

Find the QR factors for the following :

1. 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=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
v
```

```
R
Q
```

Screenshots :

```
>> gramqr1

A =

     1    -1     4
     1     4    -2
     1     4     2
     1    -1     0


v =

     2
    -2
     2
    -2


R =

     2     3     2
     0     5    -2
     0     0     4


Q =

    0.5000   -0.5000    0.5000
    0.5000    0.5000   -0.5000
    0.5000    0.5000    0.5000
    0.5000   -0.5000   -0.5000
```

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

code :

```
A=[3,2,4;2,0,2;4,2,3]
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
v
R
Q
```

screenshots :

```
>> gramqr2

A =

     3     2     4
     2     0     2
     4     2     3


v =

    0.8889
    0.4444
   -0.8889


R =

    5.3852    2.5997    5.1995
         0    1.1142    0.4333
         0         0    1.3333


Q =

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

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

code :

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

screenshots :

```
>> gramqr3

A =

     1     0
     0     1


v =

     0
     1


R =

     1     0
     0     1


Q =

     1     0
     0     1
```

5 - Gauss Jordan method of finding inverses (Not inbuilt)

Preface :

- We can find inverses of matrices using Gauss Jordan method

Problems :

Find inverses of the following matrices :

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

Code :

```
A=[1,-1,4;1,4,-2;1,4,2]
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)
```

Screenshots :

```
>> gaussinv1

A =

     1    -1     4
     1     4    -2
     1     4     2


Aug =

     1    -1     4     1     0     0
     1     4    -2     0     1     0
     1     4     2     0     0     1


B =

    0.8000    0.9000   -0.7000
   -0.2000   -0.1000    0.3000
         0   -0.2500    0.2500
```

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

code :

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

screenshots :

```
>> gaussinv2

A =

     3     2     4
     2     0     2
     4     2     3


Aug =

     3     2     4     1     0     0
     2     0     2     0     1     0
     4     2     3     0     0     1


B =

   -0.5000    0.2500    0.5000
    0.2500   -0.8750    0.2500
    0.5000    0.2500   -0.5000
```

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

code :

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

screenshots :

```
>> gaussinv3

A =

     1     0     0
     0     1     0
     0     0     1


Aug =

     1     0     0     1     0     0
     0     1     0     0     1     0
     0     0     1     0     0     1


B =

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

6 - Gauss Elimination (Not inbuilt)

Preface :

- We can get reduced forms using Gauss elimination methods

Problems :

Find reduced form of the following matrices, and solve them:

1. X+2Y+Z=3,2X+Y-2Z=3,-3X+Y+Z=-6

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);%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
x
```

Screenshots :

```
>> gausselim1

A =

     1     2    -1     3
     0    -3     0    -3
    -3     1     1    -6


A =

     1     2    -1     3
     0    -3     0    -3
     0     7    -2     3


A =

     1     2    -1     3
     0    -3     0    -3
     0     0    -2    -4


x =

     3
     1
     2
```

2. C=[1,1,1;2,-6,-1;3,4,2], b=[11;0;0]

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);%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
x
```

screenshots :

```
>> gausselim2

A =

     1     1     1    11
     0    -8    -3   -22
     3     4     2     0


A =

     1     1     1    11
     0    -8    -3   -22
     0     1    -1   -33


A =

    1.0000    1.0000    1.0000   11.0000
         0   -8.0000   -3.0000  -22.0000
         0         0   -1.3750  -35.7500


x =

    -8
    -7
    26
```

3. C=[2,1,-1;2,5,7;1,1,1], b=[0;52;9]

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);%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
x
```

screenshots :

```
>> gausselim3

A =

     2     1    -1     0
     0     4     8    52
     1     1     1     9


A =

    2.0000    1.0000   -1.0000         0
         0    4.0000    8.0000   52.0000
         0    0.5000    1.5000    9.0000


A =

    2.0000    1.0000   -1.0000         0
         0    4.0000    8.0000   52.0000
         0         0    0.5000    2.5000


x =

     1
     3
     5
```

7 - LU Decomposistion

Preface :

- Here we decompose a given matrix to its L (Lower triangular) and U (Upper triangular) components

Problems :

Decompose the following matrices to L and U components

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

Code :

```
Ab = [1,-1,4;1,4,-2;1,4,2]
n = length(Ab);
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)
```

Screenshots :

```
>> LUdecomp1

Ab =

     1    -1     4
     1     4    -2
     1     4     2


L =

     1     0     0     0
     1     1     0     0
     1     1     1     0
     0     0     0     1


U =

     1    -1     4
     0     5    -6
     0     0     4
```

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

code :

```
Ab = [3,2,4;2,0,2;4,2,3]
n = length(Ab);
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)
```

screenshots :

```
>> LUdecomp2

Ab =

     3     2     4
     2     0     2
     4     2     3


L =

    1.0000         0         0         0
    0.6667    1.0000         0         0
    1.3333    0.5000    1.0000         0
         0         0         0    1.0000


U =

    3.0000    2.0000    4.0000
         0   -1.3333   -0.6667
         0         0   -2.0000
```

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

code :

```
Ab = [1,0,0;0,1,0;0,0,1]
n = length(Ab);
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)
```

screenshots :

```
>> LUdecomp3

Ab =

     1     0     0
     0     1     0
     0     0     1


L =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1


U =

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

8 - 4 Fundamental Subspaces (Inbuilt commands)

Preface :

- Using inbuilt matlab functions find the 4 fundamental space of a matrix.

Problems :

Find the 4 inbuilt fundamental subspaces.

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

Code :

```
 A=[1,-1,4;1,4,-2;1,4,2];
% 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')
```

Screenshots :

```
>> fundspace1

columnsp =

     1    -1     4
     1     4    -2
     1     4     2


nullsp =

  3×0 empty double matrix


rowsp =

     1     0     0
     0     1     0
     0     0     1


leftnullsp =

  3×0 empty double matrix
```

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

code :

```
A=[3,2,4;2,0,2;4,2,3];
% 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')
```

screenshots :

```
>> fundspace2

columnsp =

     3     2     4
     2     0     2
     4     2     3


nullsp =

  3×0 empty double matrix


rowsp =

     1     0     0
     0     1     0
     0     0     1


leftnullsp =

  3×0 empty double matrix
```

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

code :

```
A=[1,0,0;0,1,0;0,0,1];
% 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')
```

screenshots :

```
>> fundspace3

columnsp =

     1     0     0
     0     1     0
     0     0     1


nullsp =

  3×0 empty double matrix


rowsp =

     1     0     0
     0     1     0
     0     0     1


leftnullsp =

  3×0 empty double matrix
```