



## LINEAR ALGEBRA(23MAT117) LAB MANUAL

AMRITA SCHOOL OF COMPUTING  
AMRITA VISWAVIDYAPEETAM,AMARAVATI  
CAMPUS

NAME:M.SAI MOKSHITHA  
SECTION:CSE-B  
ROLL NO:AV.SC.U4CSE24151  
SEMESTER:2nd Semester

## MATRIX OPERATORS:

### 1)

The screenshot shows the MATLAB R2024b interface with the 'Editor' tab selected. The code in the editor window is as follows:

```
x = [1,7]
a = [1;3]
a = [2 3 4;5 6 7]
z = [2:10]
z1 = [2:2:10]
z2 = [2:2:13]
```

The status bar at the bottom right indicates the date as 16-05-2025 and the time as 21:59.

## OUTPUT:

The screenshot shows the MATLAB R2024b interface with the 'Command Window' tab selected. The output of the previous code is displayed:

```
X =
    1     7

a =
    1
    3

a =
    2     3     4
    5     6     7

z =
    2     3     4     5     6     7     8     9     10

z1 =
    2     4     6     8     10

z2 =
    2     4     6     8     10     12
```

The status bar at the bottom right indicates the date as 16-05-2025 and the time as 22:07.

### 2)

The screenshot shows the MATLAB R2024b interface with the 'Editor' tab selected. The code in the editor window is as follows:

```
sem2_3.m * +
zeros(2)
zeros(2,3)
ones(2,3)
ones(4,3)
eye(3,3)
```

The status bar at the bottom right indicates the date as 16-05-2025 and the time as 22:10.

## OUTPUT:

The screenshot shows the MATLAB R2024b interface with the command window open. The workspace pane on the right is visible. The command window displays the following matrix operations:

```
ans =
0 0
0 0

ans =
0 0 0
0 0 0

ans =
1 1 1
1 1 1
1 1 1
1 1 1

ans =
1 1 1
1 1 1
1 1 1
1 1 1

ans =
1 0 0
fx 0 1 0
```

3)

The screenshot shows the MATLAB R2024b interface with the code editor open. The workspace pane on the right is visible. The code editor displays the following script file content:

```
a = [10, 20, 30;
40, 50, 60;
70, 80, 90];
a(1,2)
a(end,2)
a(:,end)
```

## Output:

The screenshot shows the MATLAB R2024b interface with the command window open. The workspace pane on the right is visible. The command window displays the following matrix operations:

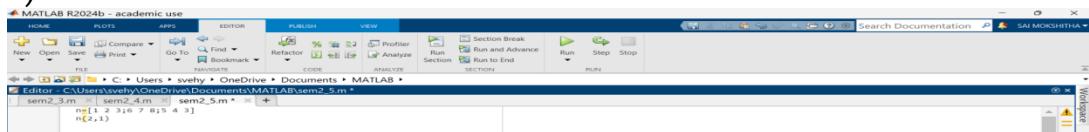
```
ans =
20

ans =
80

ans =
30
```



4)

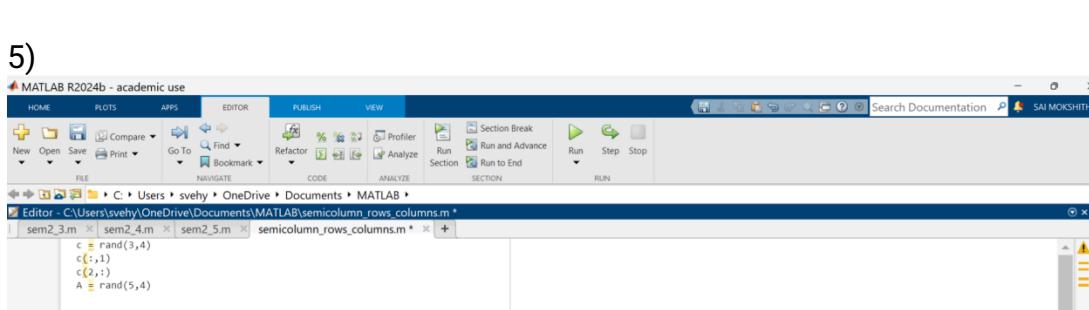


Output:

```
n =
1 2 3
6 7 8
5 4 3

ans =
6
```

5)



OUTPUT:

```
c =
0.5470 0.1890 0.3685 0.0811
0.2963 0.6868 0.6256 0.9294
0.7447 0.1835 0.7802 0.7757

ans =
0.5470
0.2963
0.7447

ans =
0.2963 0.6868 0.6256 0.9294

A =
0.4868 0.5108 0.8116 0.5502
0.4359 0.8176 0.5328 0.6225
0.4468 0.7948 0.3507 0.5870
0.3063 0.6443 0.9390 0.2077
0.5085 0.3786 0.8759 0.3012
```

6)

MATLAB R2024b - academic use

```

A=rand(3,3)
A(1,:)=1
A=rand(2,2)
A(:,1)=2

```

30°C Mostly cloudy 22:29 16-05-2025

**OUTPUT:**

MATLAB R2024b - academic use

```

A =
    0.8947  0.5767  0.4899  0.4711
    0.0699  0.1829  0.1679  0.0596
    0.6177  0.2399  0.9787  0.6828
    0.8594  0.8865  0.7127  0.6424
    0.8855  0.8287  0.3985  0.6714

ans =
    2x3
    0.0647  0.5767  0.4899  0.4711
    0.0699  0.1829  0.1679  0.0596
    0.6177  0.2399  0.9787  0.6828

A =
    2x2
    0.5216  0.8181
    0.0967  0.8175

ans =
    2x2
    0.5216  0.8181
    0.0967  0.8175

A =
    2x2
    0.5216  0.8181
    0.0967  0.8175

A =
    3x4
    0.5216  0.8181  0  0
    0.0967  0.8175  0  0
    0  0  0  0

```

0°C mostly cloudy 22:31 16-05-2025

7)

MATLAB R2024b - academic use

```

B=rand(5,5)
B(1,1:5)=1
B(2:6)
B+2

```

30°C Mostly cloudy 22:38 16-05-2025

**OUTPUT:**

MATLAB R2024b - academic use

```

B =
    0.6569  0.9841  0.4897  0.7379
    0.6288  0.1672  0.3395  0.2691
    0.2920  0.1142  0.9516  0.2678
    0.6317  0.3724  0.9283  0.5479
    0.0155  0.1981  0.0527  0.9427  0

B =
    5x5
    1.0000  1.0000  1.0000  1.0000  1.0000
    0.0000  0.1072  0.2399  0.2691  0
    0.2920  0.1062  0.9516  0.4228  0
    0.4317  0.3724  0.9283  0.5479  0
    0.0155  0.1981  0.0527  0.9427  0

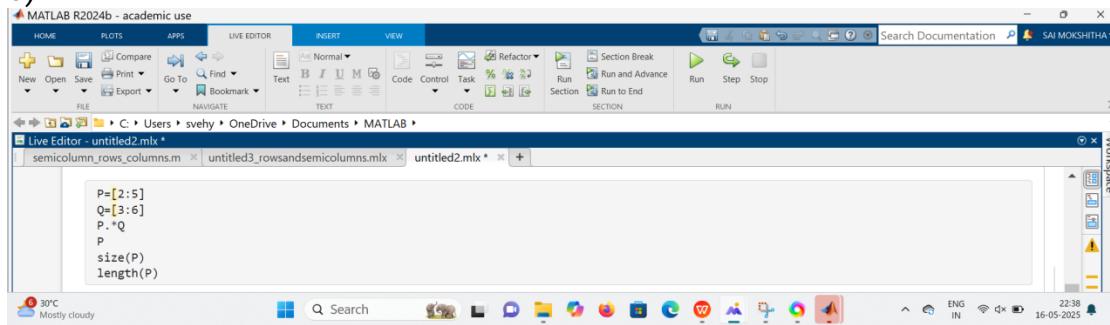
B =
    1x5
    2 3 4 5 6

ans =
    1x5
    4 5 6 7 8

```

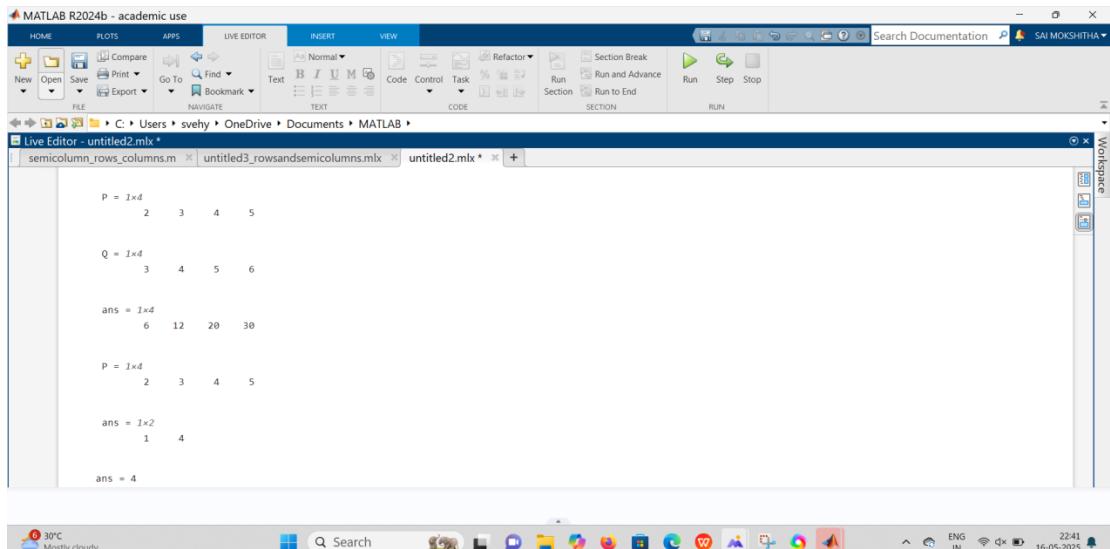
30°C Mostly cloudy 22:37 16-05-2025

8)



The screenshot shows the MATLAB R2024b interface with the LIVE EDITOR tab selected. In the code editor, the following command is entered:

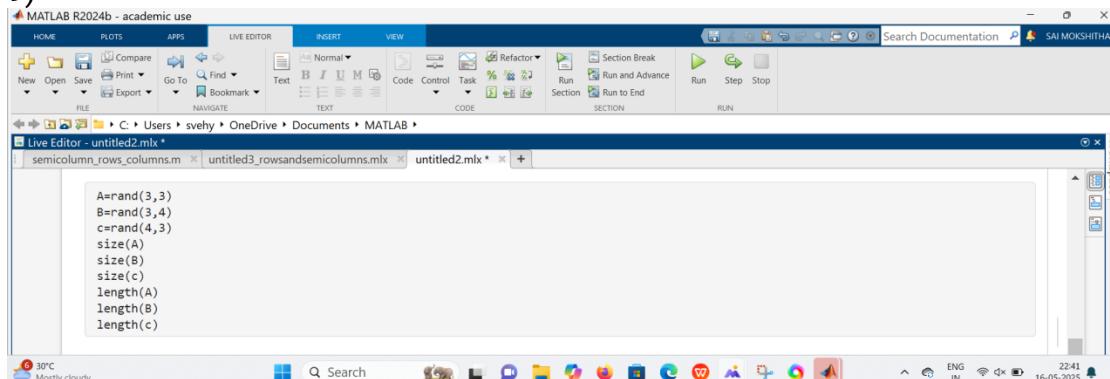
```
P=[2:5]
Q=[3:6]
P.*Q
P
size(P)
length(P)
```

**OUTPUT:**


The output window displays the results of the executed commands:

```
P = 1x5
2     3     4     5
Q = 1x6
3     4     5     6
ans = 1x4
6     12    20    30
P = 1x4
2     3     4     5
ans = 1x2
1     4
ans = 4
```

9)



The screenshot shows the MATLAB R2024b interface with the LIVE EDITOR tab selected. In the code editor, the following command is entered:

```
A=rand(3,3)
B=rand(3,4)
C=rand(4,3)
size(A)
size(B)
size(C)
length(A)
length(B)
length(C)
```

**OUTPUT:**

Live Editor - untitled2.mlx

```

semicolumn_rows_columns.m | untitled3_rowsandsemicolumns.mlx | untitled2.mlx * + 
ans = 3x2
    0.2984  0.8244
    0.6571  0.9827
    0.2653  0.7382
ans = 3x2
    0.2083  0.2687
    0.8797  0.5944
    0.8178  0.9225
ans = 3x2
    0.4213  0.1788
    0.3127  0.4229
    0.1615  0.8942
ans = 3x2
    0.5389  0.4349
    0.6544  0.8336
    0.8280  0.3196
ans = 3
ans = 4
ans = 4

```

## SYSTEM OF LINEAR EQUATIONS:- 10)

Live Editor - C:\Users\svemy\OneDrive\Documents\MATLAB\untitled5\_systemoflineareq.mlx

```

semicolumn_rows_columns.m | untitled3_rowsandsemicolumns.mlx | untitled2.mlx * + |
syms x y z
eqn1=2*x+y+z=2;
eqn2=x-y-z=3;
eqn3=x+2*y+3*z=-10;
[A,b]=equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z]);
sol=linsolve(A,b)

syms x y z
eqn1=x+y+z=3;
eqn2=2*x+2*y+2*z=6;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y,z]);
sol=linsolve(A,b)

syms x y
eqn1=x+y=3;
eqn2=x+y=5;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y]);
sol=linsolve(A,b)

```

## OUTPUT:

Live Editor - C:\Users\svemy\OneDrive\Documents\MATLAB\untitled5\_systemoflineareq.mlx

```

semicolumn_rows_columns.m | untitled3_rowsandsemicolumns.mlx | untitled2.mlx * + | untitled5_systemoflineareq.mlx
sol =
(3)
(1)
(-5)

Warning: symbolic:mldivide:RankDeficientSystem#Solution is not unique because the system is rank-deficient.#
sol =
(3)
(0)
(0)

Warning: symbolic:mldivide:InconsistentSystem#Solution does not exist because the system is inconsistent.#
sol =
(oo)
(oo)

```

```

syms x y z
eqn1=x+2*y+z==2;
eqn2=x+y+2*z==3;
eqn3=-x+2*y+3*z==10;
[A,b]=equationsToMatrix([eqn1,eqn2,eqn3],[x,y]);
sol=A\b

```

```

syms x y z
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y]);
sol=A\b

```

```

syms x y
eqn1=x+y==3;
eqn2=x+y==5;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y]);
sol=A\b

```

**OUTPUT:**

```

Warning: symbolic:mldivide:InconsistentSystem||Solution does not exist because the system is inconsistent.#
sol =

$$\begin{pmatrix} \infty \\ \infty \end{pmatrix}$$


Warning: symbolic:mldivide:RankDeficientSystem||Solution is not unique because the system is rank-deficient.#
sol =

$$\begin{pmatrix} 3-z \\ 0 \end{pmatrix}$$


Warning: symbolic:mldivide:InconsistentSystem||Solution does not exist because the system is inconsistent.#
sol =

$$\begin{pmatrix} \infty \\ \infty \end{pmatrix}$$


```

**HOMEWORK PROBLEMS:**

12)

Screenshot of MATLAB R2024b showing the Live Editor window. The code defines three systems of linear equations (eqn1, eqn2, eqn3) and converts them into matrices (A, b) using equationsToMatrix. It then solves the system using linsolve.

```
syms x y z
eqn1=2*x+y+z==2;
eqn2=-x+y-z==3;
eqn3=x+2*y+3*z==-10;
[A,b]=equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z]);
sol=linsolve(A,b)

syms x y z
eqn1=2*x+y+z==2;
eqn2=-x+y-z==3;
eqn3=x+2*y+3*z==-10;
[A,b]=equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z]);
sol=A\b

syms x y z
eqn1=2*x+y+z==2;
eqn2=-x+y-z==3;
eqn3=x+2*y+3*z==-10;
b=equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z]);
R=rref(b)
```

OUTPUT:

Screenshot of MATLAB R2024b showing the output of the solved system of linear equations. The solution is displayed as a column vector sol = [3; 1; -5]. A warning message indicates that the system is inconsistent.

```
sol =
( 3 )
( 1 )
( -5 )

Warning: symbolic:mldivide:InconsistentSystem||Solution does not exist because the system is inconsistent.
sol =
( ∞ )
( ∞ )

R =
( 1 0 0 )
( 0 1 0 )
( 0 0 1 )
```

13)

```

syms x y z
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y,z]);
sol=linsolve(A,b)

syms x y z
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y]);
sol=A\b

syms x y z
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
b=equationsToMatrix([eqn1,eqn2],[x,y,z]);
R=rref(b)

```

## OUTPUT:

```

Warning: symbolic:mldivide:RankDeficientSystem@Solution is not unique because the system is rank-deficient.#
sol =
(3
 0
 0)

Warning: symbolic:mldivide:RankDeficientSystem@Solution is not unique because the system is rank-deficient.#
sol =
(3 - z
 0
 0)

R =
( 1   1   1
  0   0   0)

```

14)

```

syms x y
eqn1=x+y==3;
eqn2=x+y==5;
[A,b]=equationsToMatrix([eqn1,eqn2],[x,y]);
sol=linsolve(A,b)

syms x y
eqn1=x+y==3;
eqn2=x+y==5;
[A,b]=equationsToMatrix([eqn1,eqn2],[x]);
sol=A\b

syms x y
eqn1=x+y==3;
eqn2=x+y==5;
b=equationsToMatrix([eqn1,eqn2],[x,y]);
R=rref(b)

```

Output:

15)

Warning: symbolic:mldivide:InconsistentSystem#Solution does not exist because the system is inconsistent.

```
sol =  
( $\infty$ )  
 $\infty$   
Warning: symbolic:mldivide:InconsistentSystem#Solution does not exist because the system is inconsistent.
```

sol =  $\infty$

R =  
$$\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$$

```
syms x y  
eqn1 = x + y == 5;  
eqn2 = x - y == 1;  
[A, b] = equationsToMatrix([eqn1, eqn2], [x, y]);  
sol1 = linsolve(A, b)  
  
syms x y  
eqn1 = x + y == 5;  
eqn2 = x - y == 1;  
[A, b] = equationsToMatrix([eqn1, eqn2], [x, y]);  
sol2 = A \ b  
  
syms x y  
eqn1 = x + y == 5;  
eqn2 = x - y == 1;  
b = equationsToMatrix([eqn1, eqn2], [x, y]);  
R = rref([b])
```

Output:

15)

MATLAB R2024b - academic use

LIVE EDITOR

```

sol1 =
(3)
(2)

sol2 =
(3)
(2)

R =
(1 0)
(0 1)

```

Command Window

Trending videos

Zoom: 100% | UTF-8 | LF | script | Ln 1 | Col 1

ENG IN 19:59 22-05-2025

16)

MATLAB R2024b - academic use

LIVE EDITOR

```

syms x a b
eqn1=a*x+b==0;
eqn2=b*x+a==0;
[A, B]=equationsToMatrix([eqn1, eqn2],[x]);
sol=linsolve(A, B)

syms x a b
eqn1=a*x+b==0;
eqn2=b*x+a==0;
sol=solve([eqn1, eqn2],x)

syms x a b
eqn1=a*x+b==0;
eqn2=b*x+a==0;
b=equationsToMatrix([eqn1, eqn2],[x]);
R=rref(b)

```

Output:

MATLAB R2024b - academic use

LIVE EDITOR

```

Warning: symbolic:mldivide:InconsistentSystem#Solution does not exist because the
system is inconsistent.#
sol = []

sol =
Empty sym: 0-by-1

R =
(1)

```

Solving three equations by three variables:

17)

**Output:**

```

syms x y z
eqn1=2*x+y+z==2;
eqn2=x-y-z==3;
eqn3=x+2*y+3*z==10;
sol=solve([eqn1,eqn2,eqn3],[x,y,z])

syms xy
eqn1=x+y==5;
eqn2==x-y==1;
sol=solve([eqn1,eqn2],[x,y])

syms xyz
eqn1=2*x+y+z==2;
eqn2=x+y-z==3;
eqn3=x+2*y+3*z==10;
sol=solve([eqn1,eqn2,eqn3],[x,y,z])

```

**Output:**

```

sol = struct with fields:
    x: 3
    y: 1
    z: -5

sol = struct with fields:
    x: 1 - z/2
    y: z/2 + 4

sol = struct with fields:
    x: [0x1 sym]
    y: [0x1 sym]

```

**18)**

**Output:**

```

syms a b
eqn=a*x+b==0;
sol=solve(eqn,x)

syms xyz
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
sol=solve([eqn1,eqn2],[x,y,z],'Returnconditions',true)

syms xyz
eqn1=x+y+z==3;
eqn2=2*x+2*y+2*z==6;
sol=solve([eqn1,eqn2],[x,y,z])

```

**Output:**

18)

MATLAB R2024b - academic use

Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\LINEARALGEBRA3.mlx \*

```

sol =

$$\frac{b}{a}$$


sol = struct with fields:
x: 3 - z2 - z1
y: z2
z: z1
parameters: [z1 z2]
conditions: symtrue

sol = struct with fields:
x: 3
y: 0
z: 0

```

30°C Mostly cloudy

ENG IN 23:00 16-05-2025

19)

MATLAB R2024b - academic use

Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\LINEARALGEBRA3.mlx \*

```

syms x y
eqn1=x+y==3;
eqn2=x+y==5;
sol=solve([eqn1,eqn2],[x,y])

syms x y z
eqn1=2*x+y+z==2;
eqn2=-x-y-z==3;
eqn3=x+2*y+3*z==10;
b=equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z]);
R=rref(b)

```

30°C Mostly cloudy

ENG IN 23:03 16-05-2025

Output:

MATLAB R2024b - academic use

Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\LINEARALGEBRA3.mlx \*

```

sol = struct with fields:
x: [0x1 sym]
y: [0x1 sym]

ans =
Empty sym: 0-by-1

sol = struct with fields:
x: [0x1 sym]
y: [0x1 sym]
parameters: [1x0 sym]
conditions: [0x1 sym]

R =

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$


```

30°C Mostly cloudy

ENG IN 23:05 16-05-2025

Graphs:  
20)

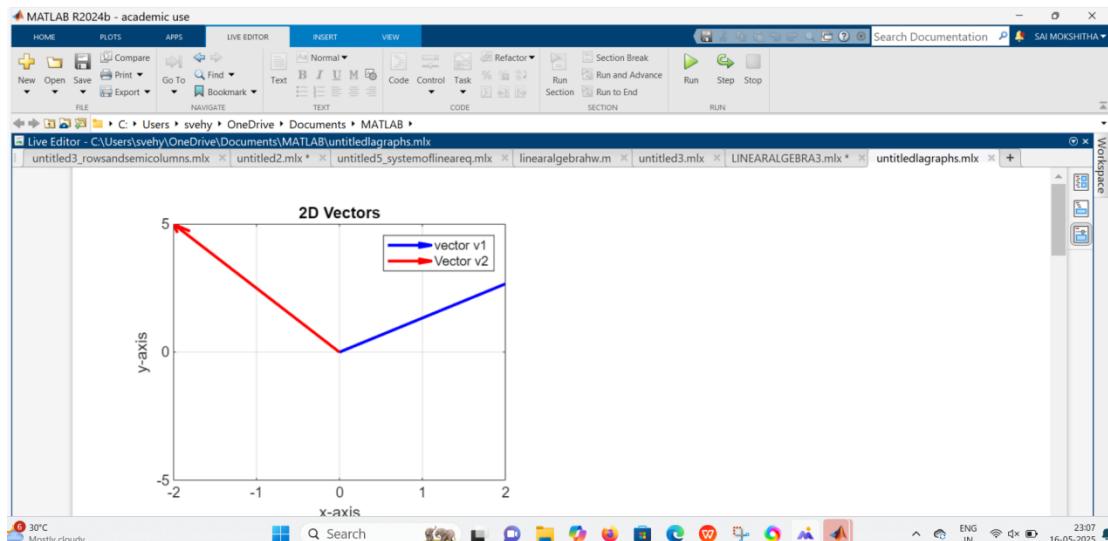
**OUTPUT:**

```

origin=[0,0];
v1=[3,4];
v2=[-2,5];
quiver(origin(1),origin(2),v1(1),v1(2),0,'b','Linewidth',2);
hold on;
quiver(origin(1),origin(2),v2(1),v2(2),0,'r','Linewidth',2);
 xlim([-2, 2]);
 ylim([-5, 5]);
 grid on;
 xlabel('x-axis');
 ylabel('y-axis');
 title('2D Vectors');
 legend('vector v1','Vector v2');
 hold off;

```

## OUTPUT:



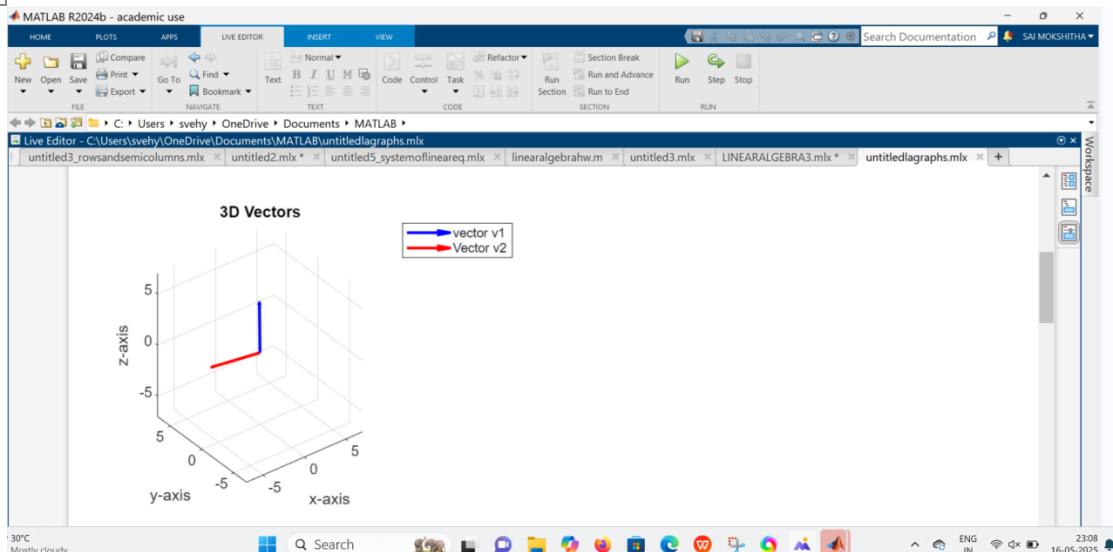
21)

```

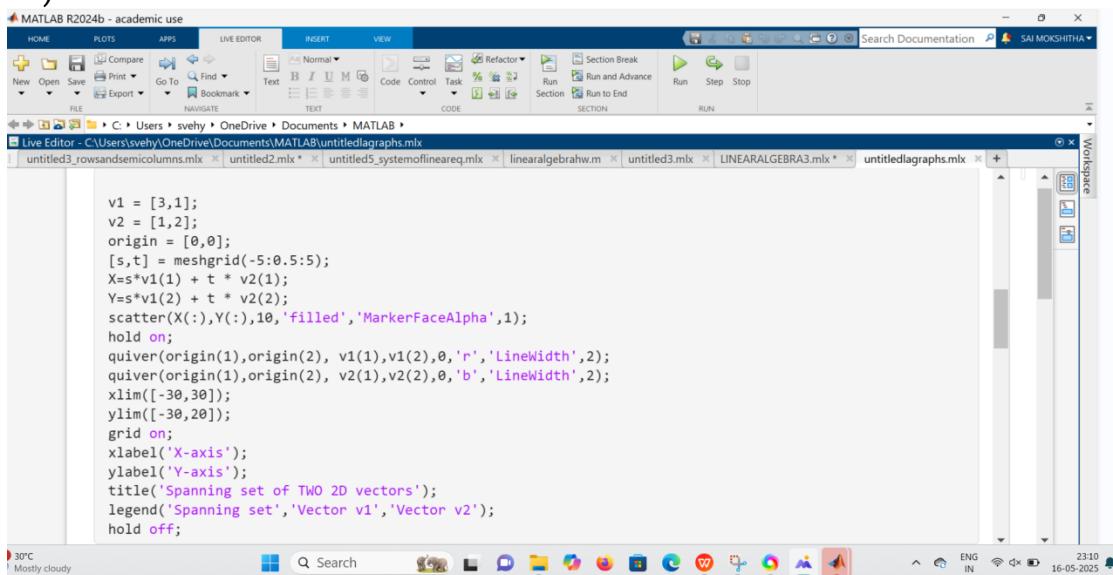
origin=[0,0,0];
v1=[3,4,2];
v2=[-2,5,-3];
quiver3(origin(1),origin(2),origin(3),v1(1),v1(2),v1(3),0,'b','Linewidth',2);
hold on;
quiver3(origin(1),origin(2),origin(3),v2(1),v2(2),v2(3),0,'r','Linewidth',2);
 xlim([-7, 7]);
 ylim([-7, 7]);
 zlim([-7, 7]);
 grid on;
 xlabel('x-axis');
 ylabel('y-axis');
 zlabel('z-axis');
 title('3D Vectors');
 legend('vector v1','Vector v2');
 hold off;

```

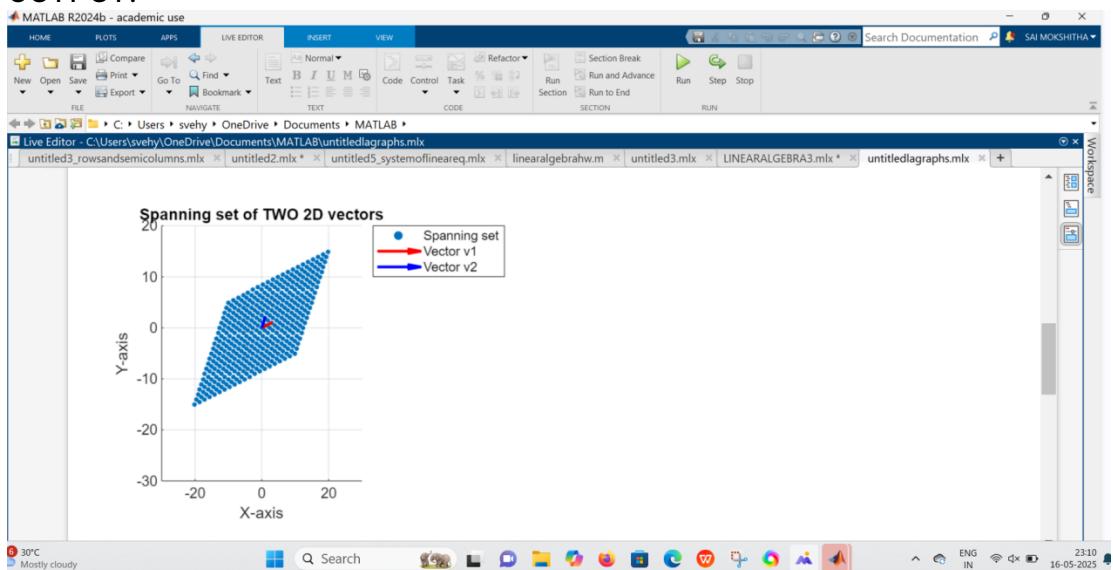
## OUTPUT:



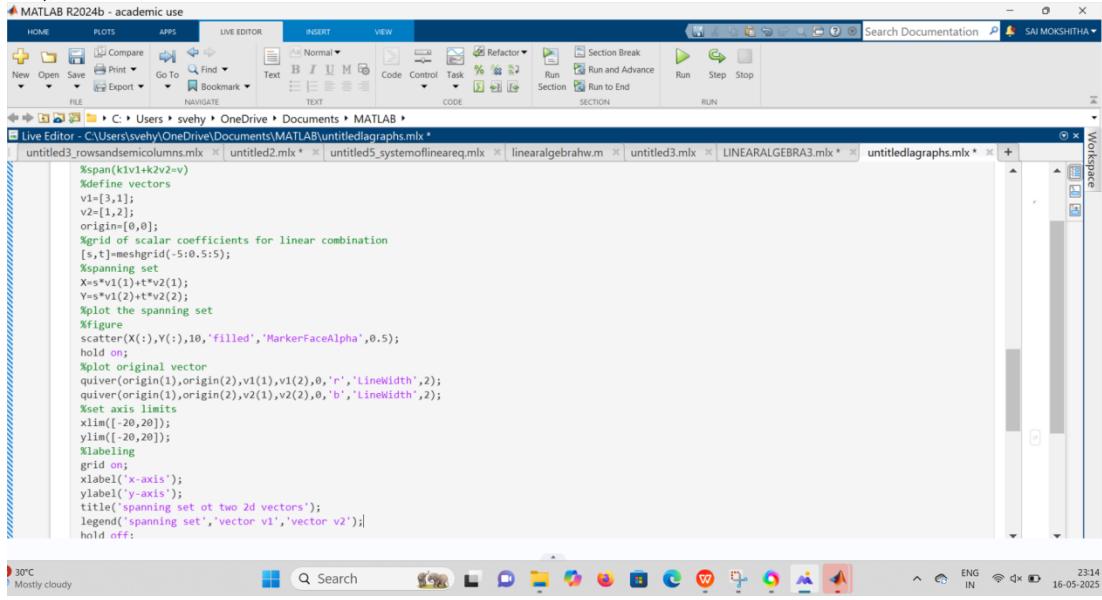
22)



## OUTPUT:



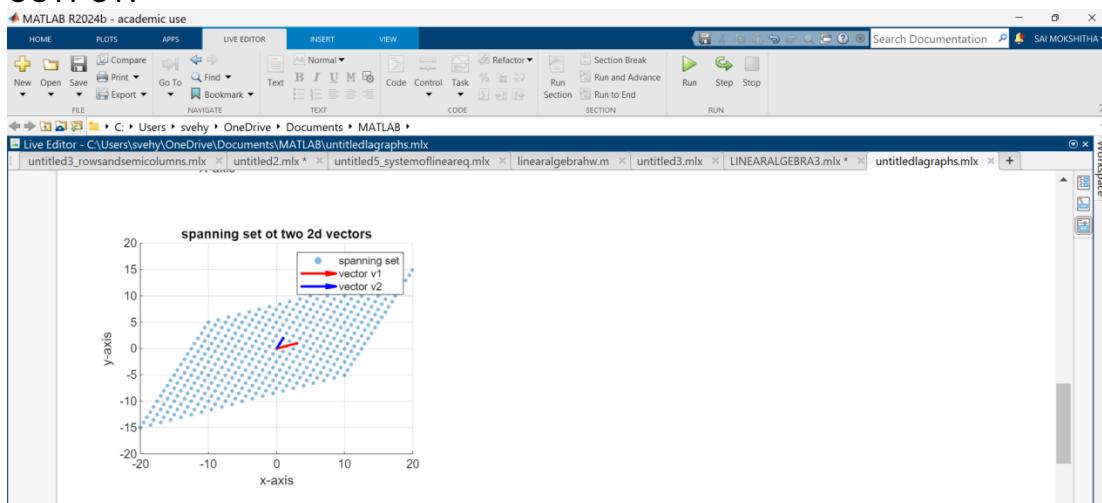
23)



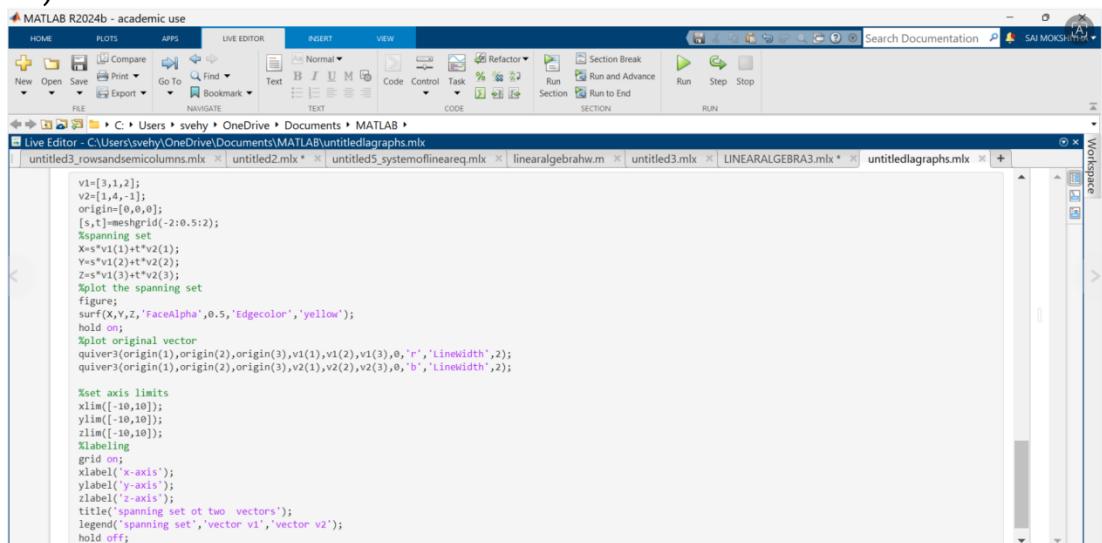
```
%span(v1+k2v2=v)
%define vectors
v1=[3,1];
v2=[1,2];
origin=[0,0];
%grid of scalar coefficients for linear combination
[s,t]=meshgrid(-5:0.5:5);
%spanning set
X=s*v1(1)+t*v2(1);
Y=s*v1(2)+t*v2(2);
%plot the spanning set
%figure
scatter(X(:,1),Y(:,1),10,'filled','MarkerFaceAlpha',0.5);
hold on;
%plot original vector
quiver(origin(1),origin(2),v1(1),v1(2),0,'r','LineWidth',2);
quiver(origin(1),origin(2),v2(1),v2(2),0,'b','LineWidth',2);
%set axis limits
xlin([-20,20]);
ylin([-20,20]);
%labeling
grid on;
xlabel('x-axis');
ylabel('y-axis');
title('spanning set of two 2d vectors');
legend('spanning set','vector v1','vector v2');
hold off;
```

30°C  
Mostly cloudy

**OUTPUT:**



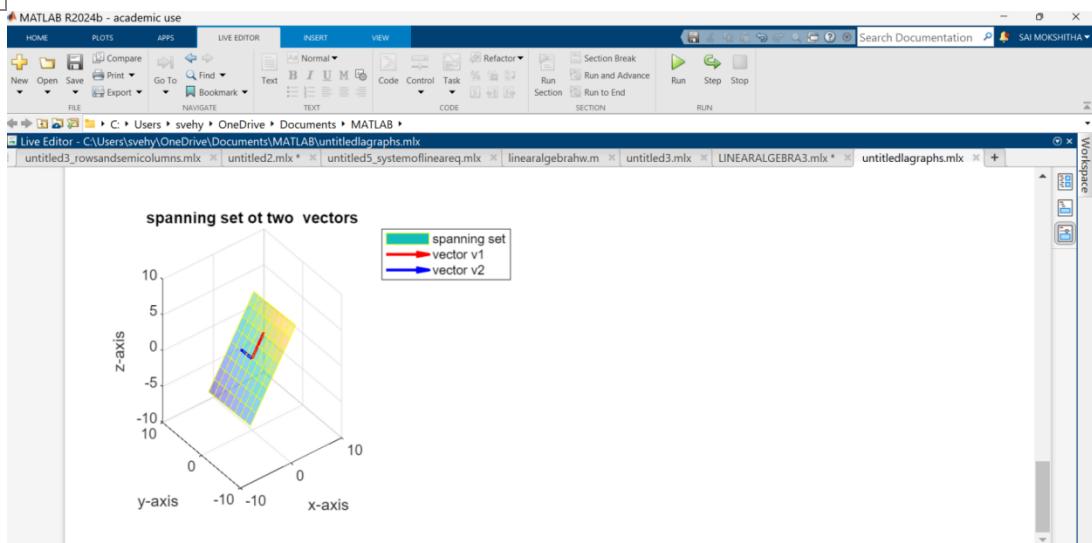
24)



```
v1=[3,1,2];
v2=[1,2,1];
origin=[0,0,0];
[s,t]=meshgrid(-2:0.5:2);
%spanning set
X=s*v1(1)+t*v2(1);
Y=s*v1(2)+t*v2(2);
Z=s*v1(3)+t*v2(3);
%plot the spanning set
figure;
surf(X,Y,Z,'FaceAlpha',0.5,'EdgeColor','yellow');
hold on;
%plot original vector
quiver3(origin(1),origin(2),origin(3),v1(1),v1(2),v1(3),0,'r','LineWidth',2);
quiver3(origin(1),origin(2),origin(3),v2(1),v2(2),v2(3),0,'b','LineWidth',2);
%set axis limits
xlin([-10,10]);
ylin([-10,10]);
zlin([-10,10]);
%labeling
grid on;
xlabel('x-axis');
ylabel('y-axis');
zlabel('z-axis');
title('spanning set of two vectors');
legend('spanning set','vector v1','vector v2');
hold off;
```

30°C  
Mostly cloudy

**OUTPUT:**



30°C  
Mostly cloudy

## Gram-Schmidt Problems:

### 25) Orthogonal check

```

clc;
clear all;
close all;
v1 = [1; 1; 0];
v2 = [0; 1; 1];
v3 = [1; 0; 1];
A = [v1 v2 v3];
fprintf('Original basis vectors:\n');
disp(A);
[m,n] = size(A);
Q = zeros(m,n);
for k = 1:n
    v = A(:,k);

    for j = 1:k-1
        q = Q(:,j);
        v = v - (q'*v)/(q*q)*q;
    end
    Q(:,k) = v;
end
R=Q'*A;
fprintf('\n Orthogonal basis vectors:\n');
disp(Q);
fprintf('\n Orthogonality check:\n');
disp(Q'*Q);
fprintf('\n upper traingular matrix R:\n');
disp(R);

```

30°C  
Mostly cloudy

**Output:**

Original basis vectors:

1	0	1
1	1	0
0	1	1

Orthogonal basis vectors:

1.0000	-0.5000	0.6667
1.0000	0.5000	-0.6667
0	1.0000	0.6667

Orthogonality check:

2.0000	0	0
0	1.5000	0.0000
0	0.0000	1.3333

upper traingular matrix R:

2.0000	1.0000	1.0000
0	1.5000	0.5000
0	0.0000	1.3333

30°C Mostly cloudy 23:20 16-05-2025

## 26) Orthonormal check

```

clc;
clear all;
close all;
v1 = [1; 1; 0];
v2 = [0; 1; 1];
v3 = [1; 0; 1];
A = [v1 v2 v3];
fprintf('Original basis vectors:\n');
disp(A);
[m,n] = size(A);
Q = zeros(m,n);
for k = 1:n
    v = A(:,k);
    for j = 1:k-1
        q = Q(:,j);
        v = v - (q'*v)/(q*q)*q;
    end
    Q(:,k) = v;
end
fprintf('\n Orthogonal basis vectors:\n');
disp(Q);
fprintf('\n Orthogonality check:\n');
orth_check = Q'* Q;
disp(orth_check);
Q_normalized = Q./vecnorm(Q);
fprintf('\n Orthonormal basis vectors:\n');
disp(Q_normalized);

```

30°C Mostly cloudy 23:19 16-05-2025

## OUTPUT:

```

Original basis vectors:
1 0 1
1 1 0
0 1 1

Orthogonal basis vectors:
1.0000 -0.5000 0.6667
1.0000 0.5000 -0.6667
0 1.0000 0.6667

Orthogonality check:
2.0000 0 0
0 1.5000 0.0000
0 0.0000 1.3333

Orthonormal basis vectors:
0.7071 -0.4882 0.5774
0.7071 0.4882 -0.5774
0 0.8165 0.5774

Orthonormality check:
1.0000 0 0
0 1.0000 0.0000
0 0.0000 1.0000

```

30°C Mostly cloudy 23:23 16-05-2025

## Transformation

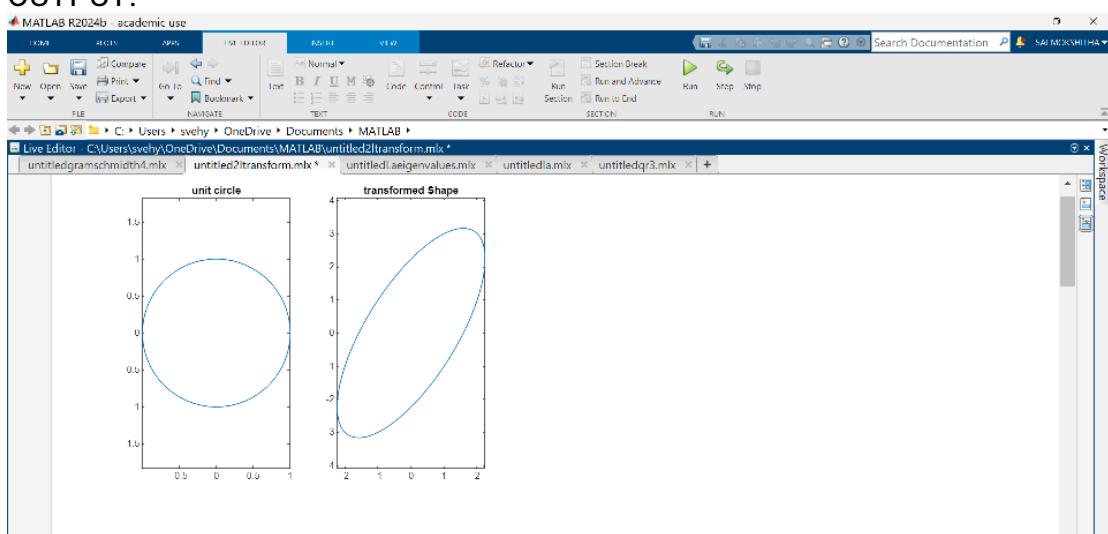
27)

MATLAB R2024b - academic use

Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\untitled2\transform.mlx

```
A=[1 2;3 1];
theta=linspace(0,2*pi,100);
x=cos(theta);
y=sin(theta);
circle=[x;y];
transformed=A*circle;
figure;
subplot(1,2,1);
plot(circle(:,1),circle(:,2));
title('unit circle');
axis equal;
subplot(1,2,2);
plot(transformed(:,1),transformed(:,2));
title('transformed Shape');
axis equal;
```

OUTPUT:



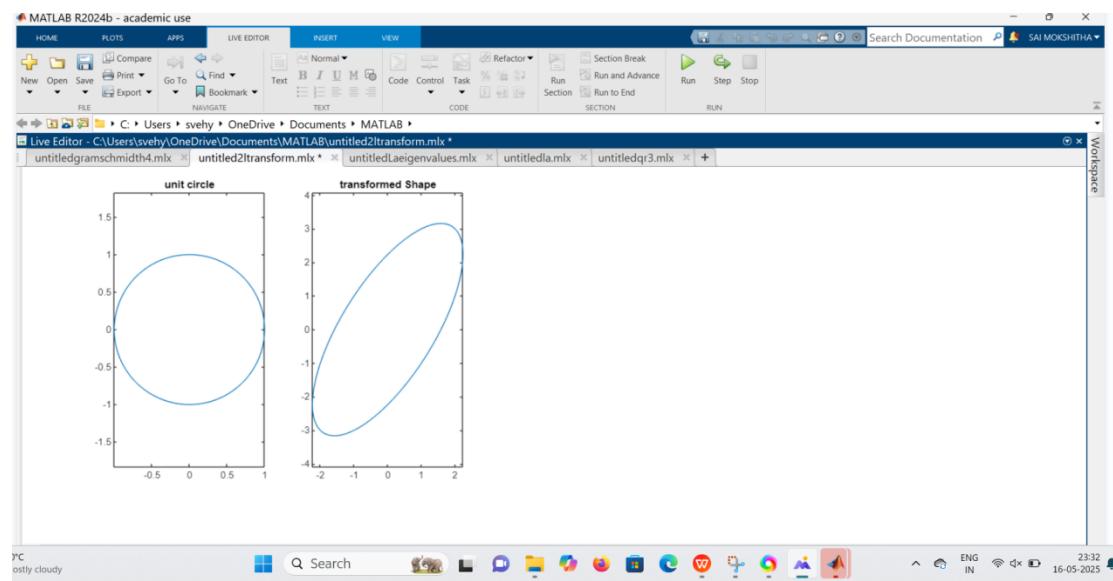
28)

MATLAB R2024b - academic use

Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\untitled2\transform.mlx

```
A=[2 0;0 2];
theta=linspace(0,2*pi,100);
x=cos(theta);
y=sin(theta);
circle=[x;y];
transformed=A*circle;
figure;
subplot(1,2,1);
plot(circle(:,1),circle(:,2));
title('unit circle');
axis equal;
subplot(1,2,2);
plot(transformed(:,1),transformed(:,2));
title('transformed Shape');
axis equal;
```

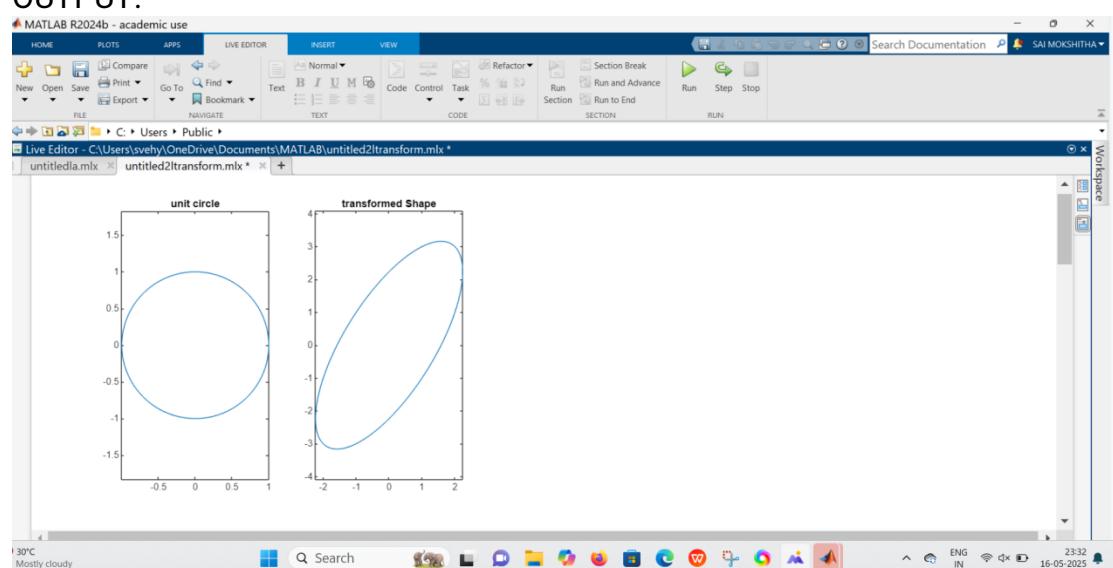
OUTPUT:



29)

```
A=[2 0;0 3];
theta=linspace(0,2*pi,100);
x=cos(theta);
y=sin(theta);
circle=[x;y];
transformed=A*circle;
figure;
subplot(1,2,1);
plot(circle(:,1),circle(:,2));
title('unit circle');
axis equal;
subplot(1,2,2);
plot(transformed(:,1),transformed(:,2));
title('transformed Shape');
axis equal;
```

## OUTPUT:



30)

MATLAB R2024b - academic use

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

New Open Save Print Export Go To Find Bookmark NAVIGATE

Text Normal Refactor Code Control Task Run Run and Advance Run Section Run End Run Step Stop SECTION RUN

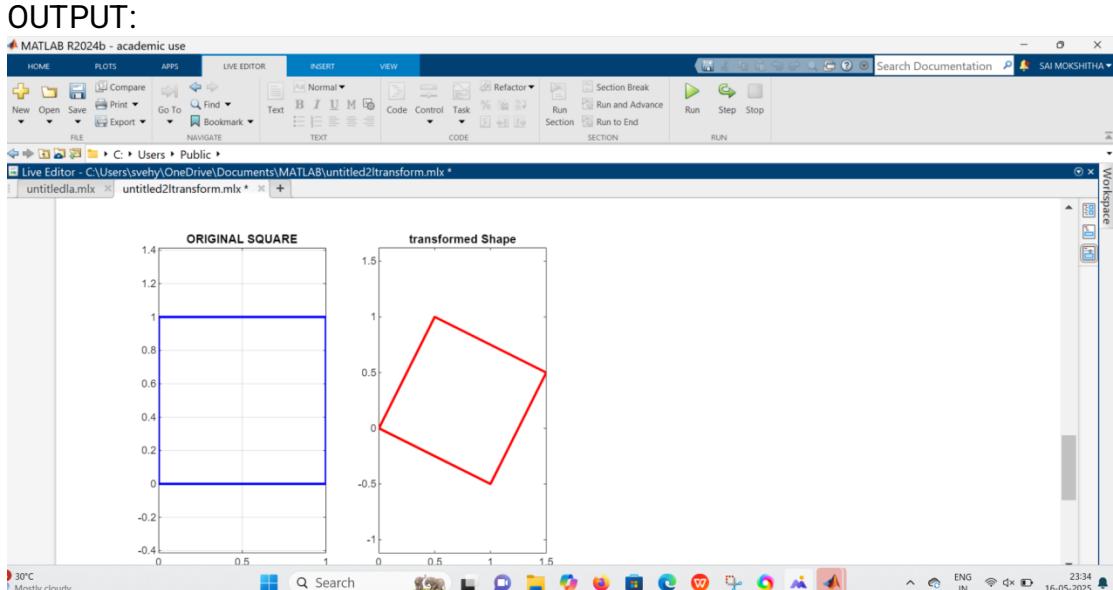
Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\untitled2\transform mlx \*

untitleddla mlx \* untitled2\transform mlx \*

```
A=[1 0.5;-0.5 1];
square=[0 1 1 0;0 0 1 1 0];
transformed=A*square;
figure;
subplot(1,2,1);
plot(square(1,:),square(2,:),'b-','LineWidth',2);
title('ORIGINAL SQUARE');
axis equal;
grid on;

subplot(1,2,2);
plot(transformed(1,:),transformed(2,:),'r-','LineWidth',2);
title('transformed Shape');
axis equal;
```

OUTPUT:



31)

MATLAB R2024b - academic use

HOME PLOTS APPS LIVE EDITOR INSERT VIEW

New Open Save Print Export Go To Find Bookmark NAVIGATE

Text Normal Refactor Code Control Task Run Run and Advance Run Section Run End Run Step Stop SECTION RUN

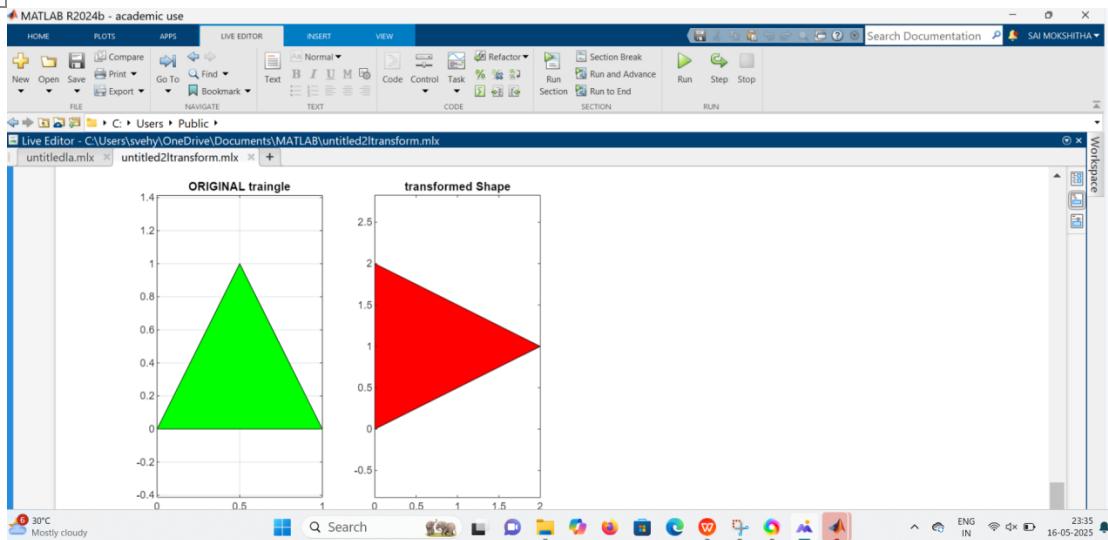
Live Editor - C:\Users\svehy\OneDrive\Documents\MATLAB\untitled2\transform mlx \*

untitleddla mlx \* untitled2\transform mlx \*

```
A=[0 2;2 0];
triangle=[0 1 0.5 0;0 0 1 0];
transformed=A*triangle;
figure;
subplot(1,2,1);
fill(triangle(1,:),triangle(2,:),'g-');
title('ORIGINAL traingle');
axis equal;
grid on;

subplot(1,2,2);
fill(transformed(1,:),transformed(2,:),'r-');
title('transformed Shape');
axis equal;
```

OUTPUT:



## EIGEN VALUES:

32a)

```

cic;
clear;
A=input('Enter your matrix:');
[m,n]=size(A);
if m==n
    error('the matrix should be square for diagonalization');
else
    [P,D]=eig(A);
    eigenvalues=diag(D);
    disp('Eigenvalues:');
    disp(eigenvalues);
    disp('Eigenvectors(column of P):');
    disp(P);
    if rank(P)==m
        disp('Matrix is diagonalizable');
        disp(P);
        disp('Inverse of P:');
        invP=inv(P);
        disp(invP);
        diagonal_matrix=invP*A*p;
        disp('Diagonal matrix(inv(P)*A*P):');
        disp(diagonal_matrix);
    else
        disp('Matrix is not diagonalizable-not enough linearly independent terms');
    end
end

```

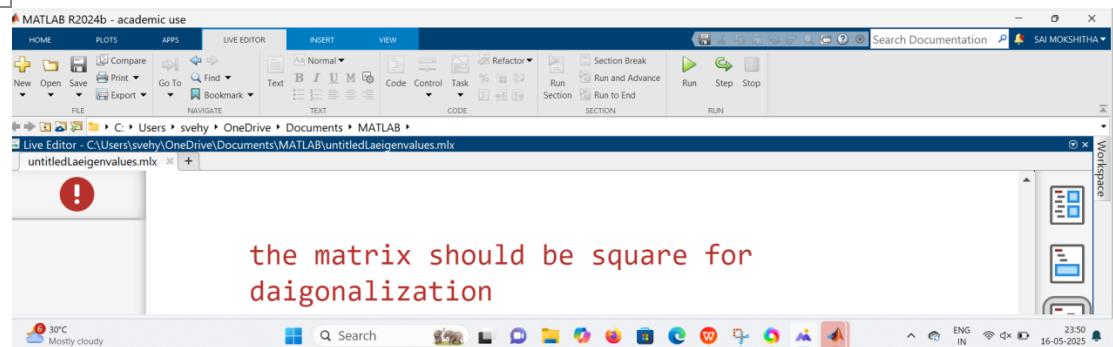
## OUTPUT:

IF I entered a non square matrix

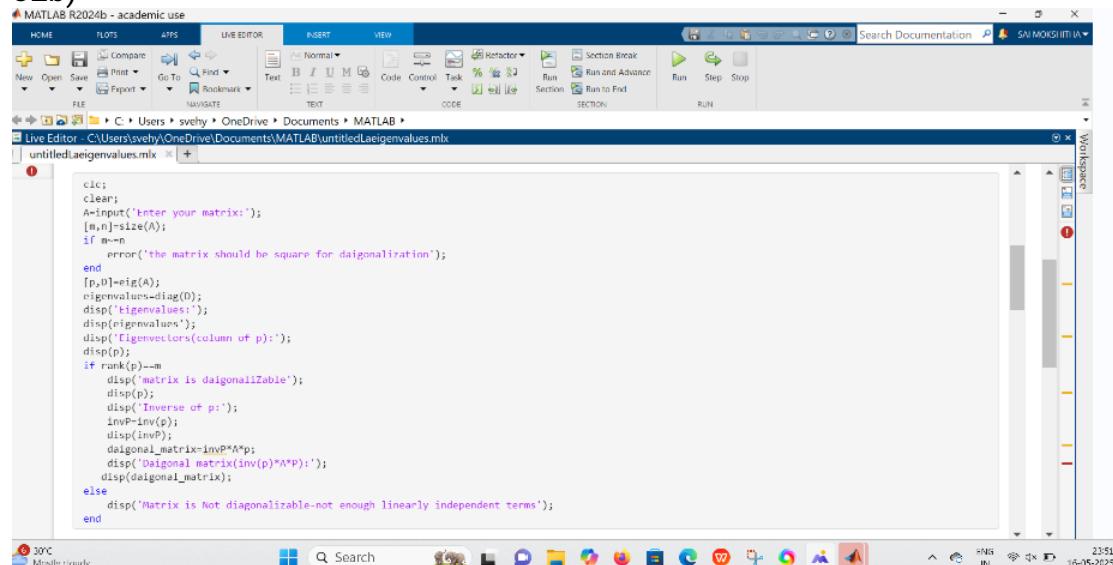
```

Enter your matrix:[1 2 3; 4 5 6]
f2 >>

```

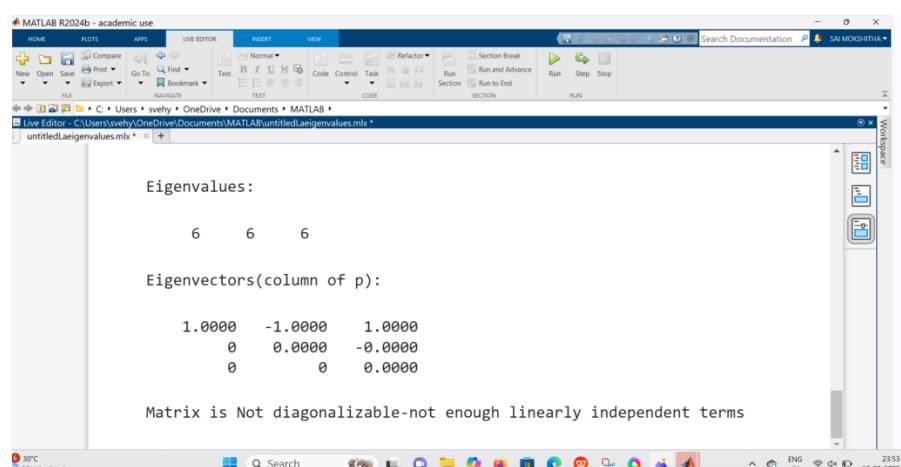
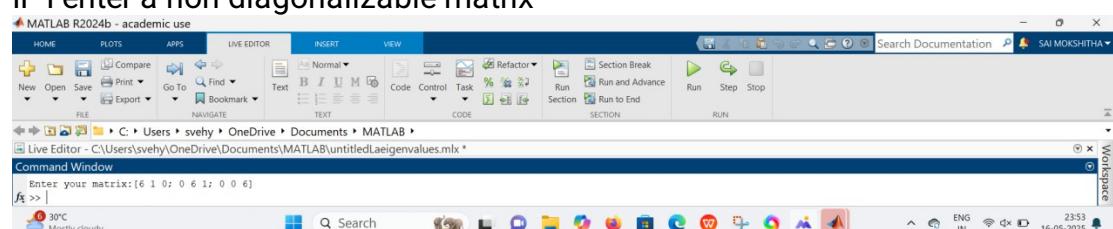


32b)



OUTPUT:

IF I enter a non diagonalizable matrix



32c)

Screenshot of MATLAB R2024b showing the code for matrix diagonalization. The code checks if the input matrix is square and diagonalizable, then finds eigenvalues and eigenvectors, and prints the results.

```
cic;
clear;
A=input('Enter your matrix:');
[n,n]=size(A);
if n~=n
    error('the matrix should be square for daigonalization');
end
[D,P]=eig(A);
eigenvalues=diag(D);
disp('Eigenvalues:');
disp(eigenvalues);
disp('Eigenvectors:');
disp('Eigenvectors(column of p):');
disp(p);
if rank(p)==n
    disp('matrix is daigonizable');
    disp(p);
    disp('Inverse of p:');
    invP=inv(p);
    disp(invP);
    diagonal_matrix=invP*A*p;
    disp('Diagonal matrix(inv(p)*A*p)');
    disp(diagonal_matrix);
else
    disp('Matrix is not diagonalizable-not enough linearly independent terms');
end
```

OUTPUT:

IF I enter a matrix that it it is a square and diagonalisable:

Screenshot of MATLAB Command Window showing the output for a diagonalizable matrix. The user enters a 2x2 identity matrix, and the program outputs eigenvalues, eigenvectors, and the diagonalized matrix.

```
f>>
```

Screenshot of MATLAB Live Editor showing the output for a diagonalizable matrix. The user enters a 2x2 identity matrix, and the program outputs eigenvalues, eigenvectors, and the diagonalized matrix.

```
Eigenvalues:
2 2
Eigenvectors(column of p):
1 0
0 1
matrix is daigonizable
1 0
0 1
Inverse of p:
1 0
0 1
Diagonal matrix(inv(p)*A*p):
2 0
0 2
```

32d)

```

clc;
clear;
A=input('Enter your matrix:');
[n,m]=size(A);
if m~n
    error('the matrix should be square for daigonalization');
end
[p,D]=eig(A);
eigenvalues=diag(D);
disp('Eigenvalues:');
disp(eigenvalues);
disp('Eigenvectors(column of p):');
disp(p);
if rank(p)~=n
    disp('matrix is diagonalizable');
    disp(p);
    disp('Inverse of p:');
    disp(inv(p));
    disp(invP);
    diagonal_matrix=invP*A*p;
    disp('Diagonal matrix(invP)*A*p:');
    disp(diagonal_matrix);
else
    disp('Matrix is not diagonalizable-not enough linearly independent terms');
end

```

## OUTPUT:

```

Enter your matrix:[2 1 0; 1 2 0; 0 0 3]
f8 >>


```

```

Eigenvalues:
1 3 3

Eigenvectors(column of p):

-0.7071 0.7071 0
0.7071 0.7071 0
0 0 1.0000

matrix is diagonalizable
-0.7071 0.7071 0
0.7071 0.7071 0
0 0 1.0000
Inverse of p:
-0.7071 0.7071 0
0.7071 0.7071 0
0 0 1.0000
Diagonal matrix(inv(p)*A*p):
1 0 0
0 3 0
0 0 3

```

33a) EXERCISE QUESTION: WRITE A MATLAB CODE TO ORTOGONALLY DIAGONALISE A GIVEN SYMMETRIC MATRIX .ASK FOR INPUT CHECK

WHETERER THE MATRIX IS SQUARE OR NOT.  
IF I ENTER A SYMMETRIC MATRIX

```

clc;
clear;

A = input('Enter a symmetric matrix: ');
[m, n] = size(A);

if m ~= n
    error('Matrix must be square for diagonalization');
end

if ~isequal(A, A')
    error('Matrix must be symmetric for orthogonal diagonalization');
end

[V, D] = eig(A);

disp('Orthogonal matrix P (eigenvectors):');
disp(P);

disp('Diagonal matrix D (eigenvalues):');
disp(D);

disp(['Check: P'' * A * P = D']);
disp(P'*A*P);

disp(['Check: P * D * P'' = A']);

```

Output:

```

Command Window
Enter a symmetric matrix: [ 1 2 3; 2 4 5; 3 5 6]
>>

```

```

MATLAB R2024b - academic use
Live Editor - untitled.mlx
untitledaeigenvalues.mlx  untitled.mlx * + | Orthogonal matrix P (eigenvectors):
0.7370   0.5910   0.3288
0.3288   -0.7370   0.5910
-0.5910   0.3288   0.7370

Diagonal matrix D (eigenvalues):
-0.5157      0      0
0      0.1709      0
0      0      11.3448

Check: P' * A * P = D
-0.5157   0.0000   0.0000
0.0000   0.1709  -0.0000
0      -0.0000  11.3448

Check: P * D * P' = A
1.0000   2.0000   3.0000
2.0000   4.0000   5.0000
3.0000   5.0000   6.0000

```

The screenshot shows the MATLAB R2024b interface with the 'Live Editor' tab selected. The code in the editor is as follows:

```
clic;
clear;

A = input('Enter a symmetric matrix: ');
[n, n] = size(A);

if n ~= n
    error('Matrix must be square for diagonalization');
end

if ~isequal(A, A')
    error('Matrix must be symmetric for orthogonal diagonalization');
end

[P, D] = eig(A);

disp('Orthogonal matrix P (eigenvectors):');
disp(P);

disp('Diagonal matrix D (eigenvalues):');
disp(D);

disp('Check: P' * A * P = D');
disp(P' * A * P);

disp('Check: P * D * P' = A');
```

The workspace on the right side shows variables: 'A' (symmetric matrix), 'D' (diagonal matrix), 'P' (orthogonal matrix), and 'check' (check variable).

## OUTPUT: IF I ENTER A NON SYMMETRIC MATRIX

The screenshot shows the MATLAB R2024b interface with the 'Command Window' tab selected. The user has entered:

```
>> Enter a symmetric matrix: [1 2 3 ;4 5 6;7 8 9]
```

The command window displays the error message:

```
Matrix must be symmetric for orthogonal diagonalization
```

The screenshot shows the MATLAB R2024b interface with the 'Live Editor' tab selected. The code in the editor is identical to the one in the first screenshot. The workspace on the right side shows variables: 'A' (symmetric matrix), 'D' (diagonal matrix), 'P' (orthogonal matrix), and 'check' (check variable). The error message 'Matrix must be symmetric for orthogonal diagonalization' is displayed in red text at the bottom of the editor.