22AIE214 – INTRODUCTION TO ROBOTICS

LABSHEET 4

Name : Anuvind M P

Roll no : AM.EN.U4AIE22010

Q1) Justify the statement that "*Rotations are non-commutative in 3D*".

Instruction: Show the Matlab operations with example. You can put snapshots in your answer sheet and give relevant explanation.

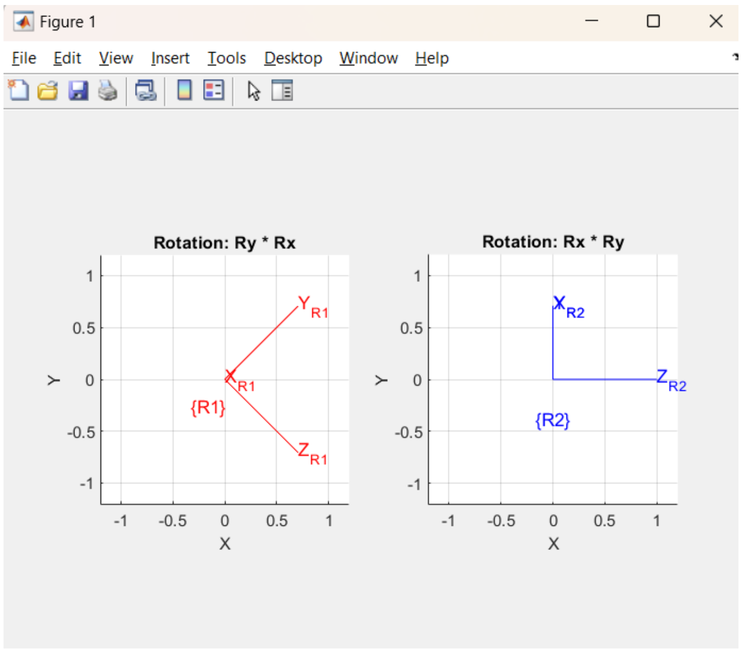
Refer Text book Robotics Vison and Control by Peter Corke page 29-35

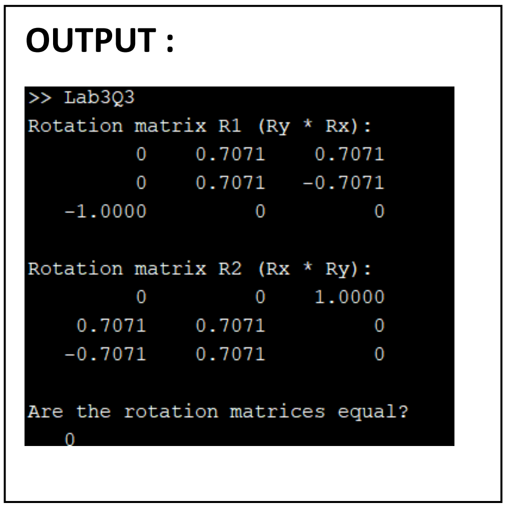
CODE :

```
Rx = rotx(45, 'deg');
Ry = roty(90, 'deg');
R1 = Ry * Rx;
R2 = Rx * Ry;
disp('Rotation matrix R1 (Ry * Rx):');
disp(R1);

disp('Rotation matrix R2 (Rx * Ry):');
disp(R2);

isEqual = isequal(round(R1, 10), round(R2, 10));
disp('Are the rotation matrices equal?');
disp(isEqual);

figure;
subplot(1,2,1);
trplot(R1, 'frame', 'R1', 'color', 'r');
title('Rotation: Ry * Rx');

subplot(1,2,2);
trplot(R2, 'frame', 'R2', 'color', 'b');
title('Rotation: Rx * Ry');
end
```

Justification :

Rotations in 3D are non-commutative, meaning the order in which rotations are applied matters. This can be mathematically demonstrated using rotation matrices. For instance, rotating first about the x-axis and then the y-axis yields a different result than rotating first about the y-axis and then the x-axis. Physically, imagine rotating a book 90 degrees around the x-axis and then 90 degrees around the y-axis, and compare this to doing the rotations in the reverse order—the book ends up in different orientations. Thus, both mathematical representation and physical experiments show that 3D rotations do not commute.





OUTPUT :

```
>> Lab3Q3
Rotation matrix R1 (Ry * Rx):
         0    0.7071    0.7071
         0    0.7071   -0.7071
   -1.0000         0         0

Rotation matrix R2 (Rx * Ry):
         0         0    1.0000
    0.7071    0.7071         0
   -0.7071    0.7071         0

Are the rotation matrices equal?
   0
```

Q2) Describe twist in 2D.

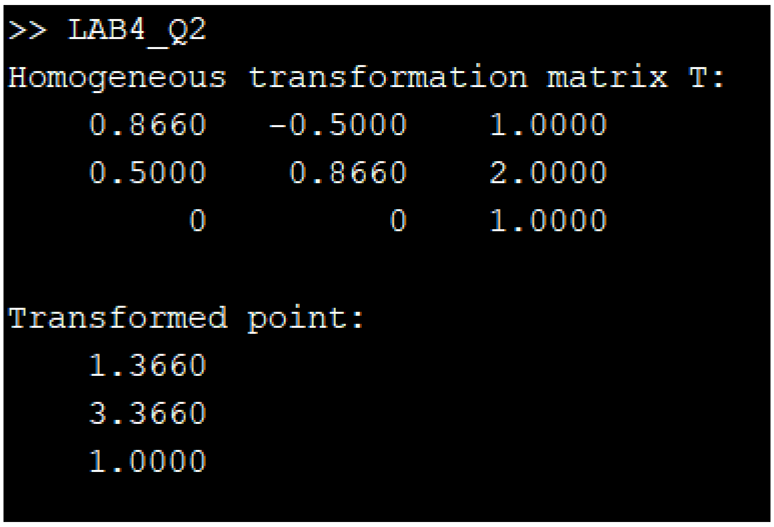
Use matlab functions to show how to regenerate homogeneous transformations

Refer Text book Robotics Vison and Control by Peter Corke page 29-35

CODE :

```
vx = 1; % Translational velocity in x
vy = 2; % Translational velocity in y
omega = pi/6;
dt = 1;

theta = omega * dt;
R = [cos(theta), -sin(theta); sin(theta), cos(theta)];

t = [vx * dt; vy * dt];

T = [R, t; 0, 0, 1];
% Display the result
disp('Homogeneous transformation matrix T:');
disp(T);
% a point in homogeneous coordinates
point = [1; 1; 1];
transformed_point = T * point;
disp('Transformed point:');
disp(transformed_point);

end
```

OUTPUT :



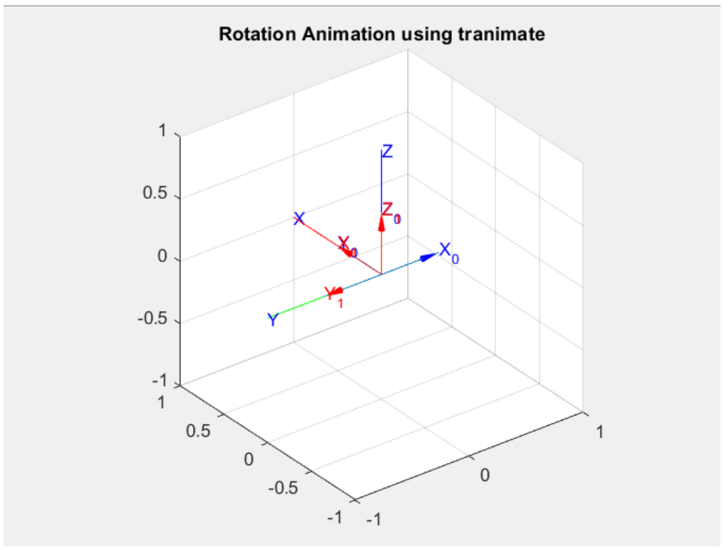
```
>> LAB4_Q2
Homogeneous transformation matrix T:
    0.8660   -0.5000    1.0000
    0.5000    0.8660    2.0000
         0         0    1.0000

Transformed point:
    1.3660
    3.3660
    1.0000
```

Q3) Demonstrate the Toolbox function tranimate which animates a rotation using Matlab.

Refer Text book Robotics Vison and Control by Peter Corke page 29-35

CODE :

```
T0 = eye(4);
theta = pi/2;
R = trotz(theta);
T1 = R;
figure;
axis([-1 1 -1 1 -1 1]);
view(3);
grid on;
hold on;

trplot(T0, 'frame', '0', 'color', 'b', 'arrow', 'length', 0.5);
title('Rotation Animation using tranimate');
tranimate(T0, T1, 'frames', 50, 'rgb');
trplot(T1, 'frame', '1', 'color', 'r', 'arrow', 'length', 0.5);
hold off;
end
```


Rotation Animation using tranimate
Z
Z0
X
X0
X0
Y
Y1