[Robotics]

(Project_1)

Mostafa Hisham
Sec :2
BN:27

Sub.To:
DR/ Muhammed Rushdy
E/ Eslam Mahmoud

Project 01 - Representing Position and Orientation

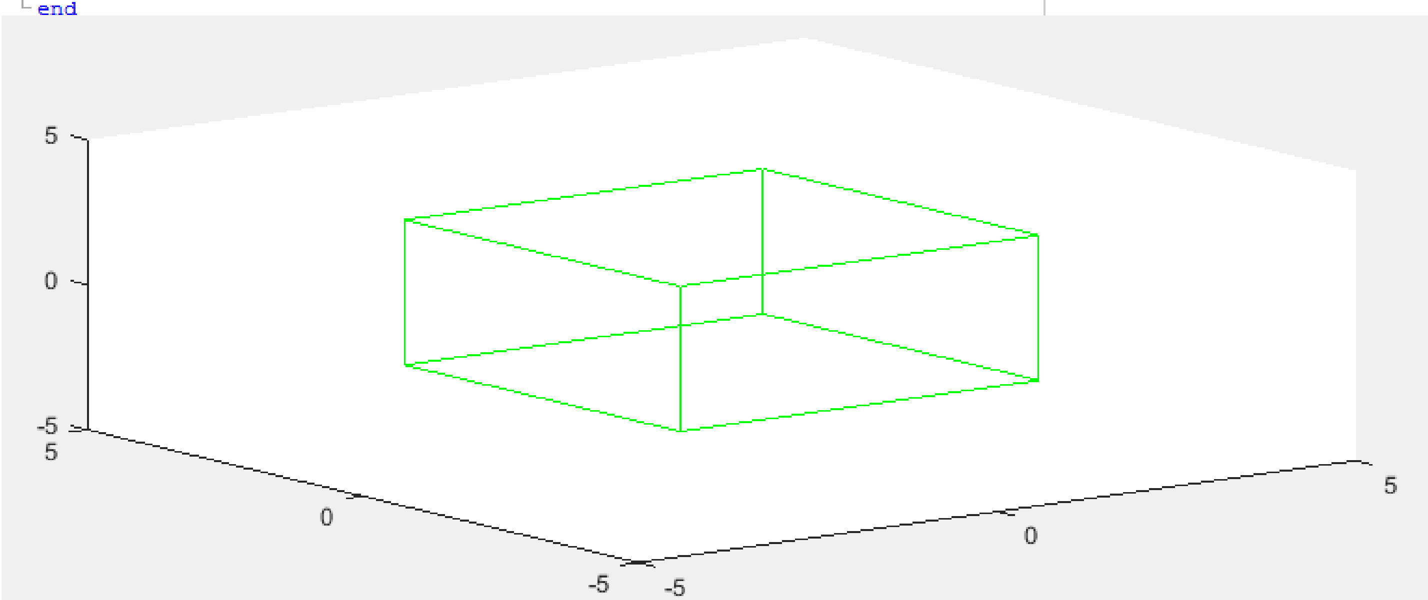
A)From [Corke 2011], solve the following exercises:

4. Animate a tumbling cube
   a) Write a function to plot the edges of a cube centred at the origin.

or - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\Corke_2_4_Draw_Cube.m*

rke2_4_FUNC.m | corke.m | Corke_2_4_Draw_Cube.m*

```
function Corke_2_4_Draw_Cube(length, width, height )
R = eul2r(pi/2, pi/2, pi/2);
p = [[width/2; length/2; height/2],[width/2; -length/2; height/2],[width/2; length/2; -height/2],...
    [width/2; -length/2; -height/2],[-width/2; length/2; height/2],[-width/2; -length/2; height/2],...
    [-width/2; length/2; -height/2],[-width/2; -length/2; -height/2]];
p = R*p;
x = [[p(1,1); p(1,2)], [p(1,1); p(1,3)], [p(1,1); p(1,5)], ...
        [p(1,6); p(1,5)], [p(1,6); p(1,2)], [p(1,6); p(1,8)], ...
        [p(1,7); p(1,5)], [p(1,7); p(1,3)], [p(1,7); p(1,8)], ...
        [p(1,4); p(1,2)], [p(1,4); p(1,3)], [p(1,4); p(1,8)]];

y = [[p(2,1); p(2,2)], [p(2,1); p(2,3)], [p(2,1); p(2,5)], ...
        [p(2,6); p(2,5)], [p(2,6); p(2,2)], [p(2,6); p(2,8)], ...
        [p(2,7); p(2,5)], [p(2,7); p(2,3)], [p(2,7); p(2,8)], ...
        [p(2,4); p(2,2)], [p(2,4); p(2,3)], [p(2,4); p(2,8)]];

z = [[p(3,1); p(3,2)], [p(3,1); p(3,3)], [p(3,1); p(3,5)], ...
        [p(3,6); p(3,5)], [p(3,6); p(3,2)], [p(3,6); p(3,8)], ...
        [p(3,7); p(3,5)], [p(3,7); p(3,3)], [p(3,7); p(3,8)], ...
        [p(3,4); p(3,2)], [p(3,4); p(3,3)], [p(3,4); p(3,8)]];

plot3(x, y, z, 'g');
maxDim = max([length, width, height]);
limit = maxDim;
xlim([-limit, limit]); ylim([-limit, limit]); zlim([-limit, limit]);
end
```



b) Modify the function to accept an argument which is a homogeneous transformation which is applied to the cube vertices before plotting.

a)

tor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\Corke_2_4_Draw_Cube.m*

rke2_4_FUNC.m | corke.m | Corke_2_4_Draw_Cube.m*

```
function Corke_2_4_Draw_Cube(length, width, height )
R = eul2r(pi/2, pi/2, pi/2);
```

The Modification

tor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\Corke_2_4_Draw_Cube.m

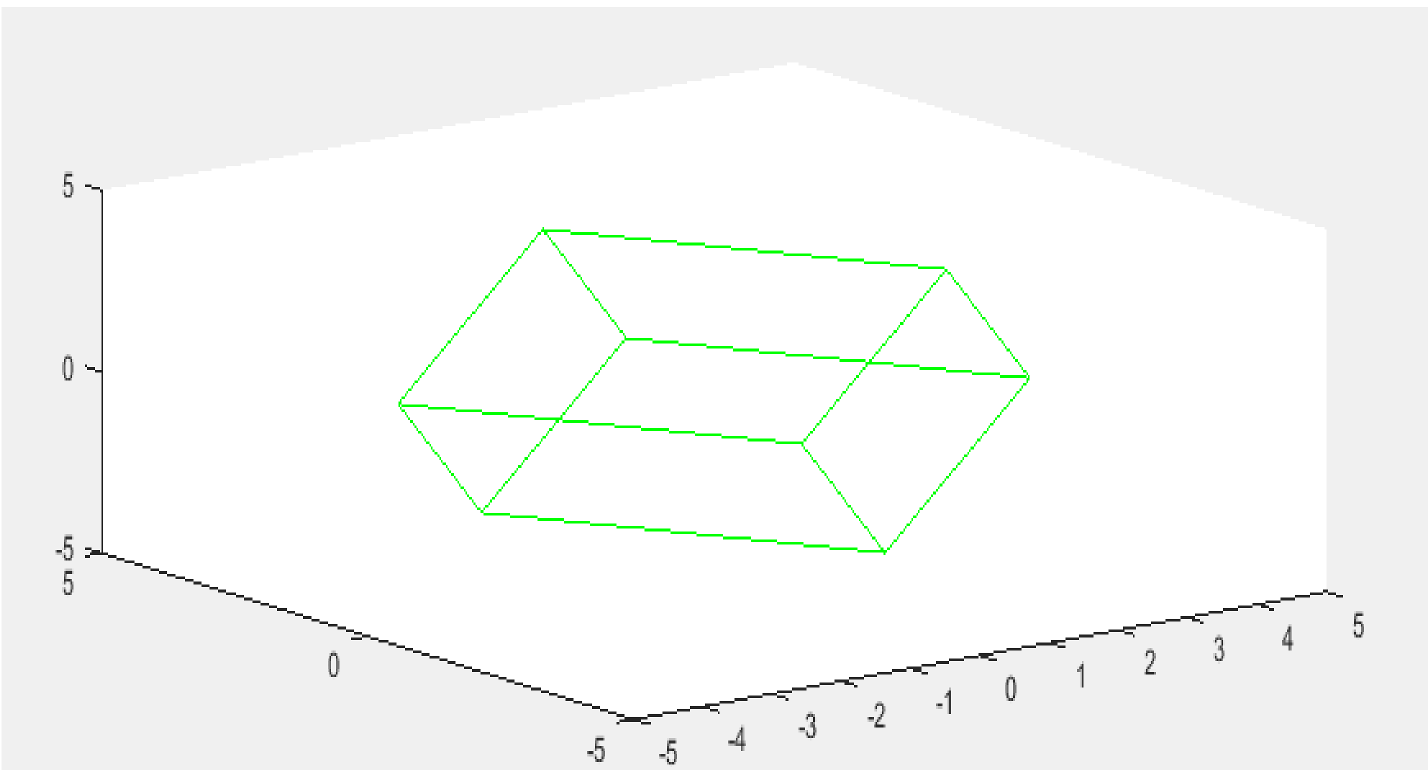
rke2_4_FUNC.m | corke.m | Corke_2_4_Draw_Cube.m

```
function Corke_2_4_Draw_Cube(length, width, height , Raw , Pitch , yaw)
Raw = Raw*pi/180; Pitch = Pitch*pi/180; yaw = yaw*pi/180;
R = eul2r(Raw, Pitch, yaw);
```

itor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\corke.m*

orke2_4_FUNC.m | corke.m* | Corke_2_4_Draw_Cube.m

```
Corke_2_4_Draw_Cube(5 , 5 , 5 , 30 , 60 ,0)
```

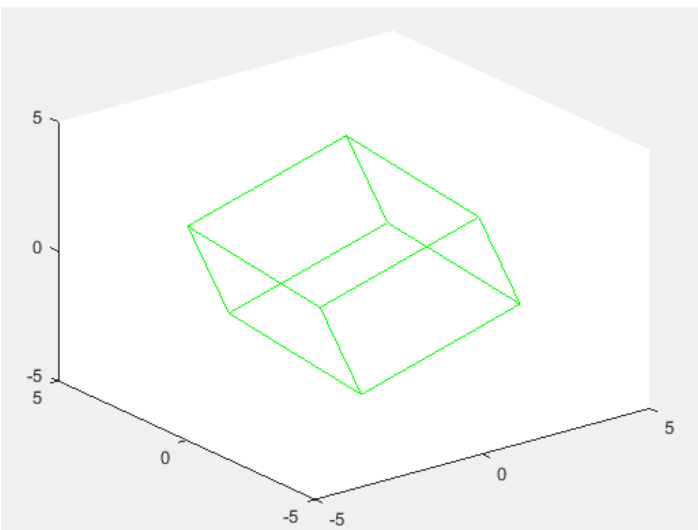


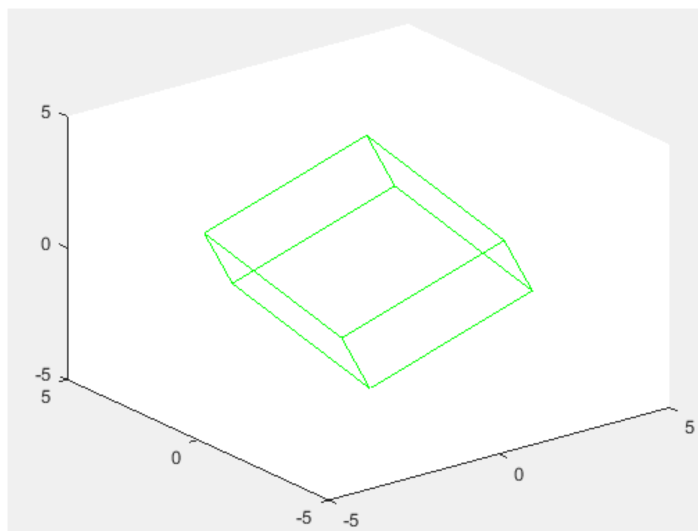
c) Animate rotation about the *x*-axis.
d) Animate rotation about all axes.

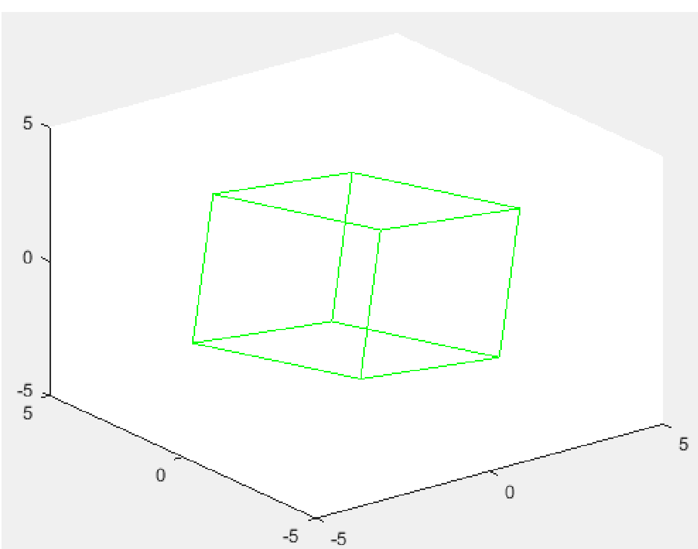
or - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\Corke2_4_Animate.m

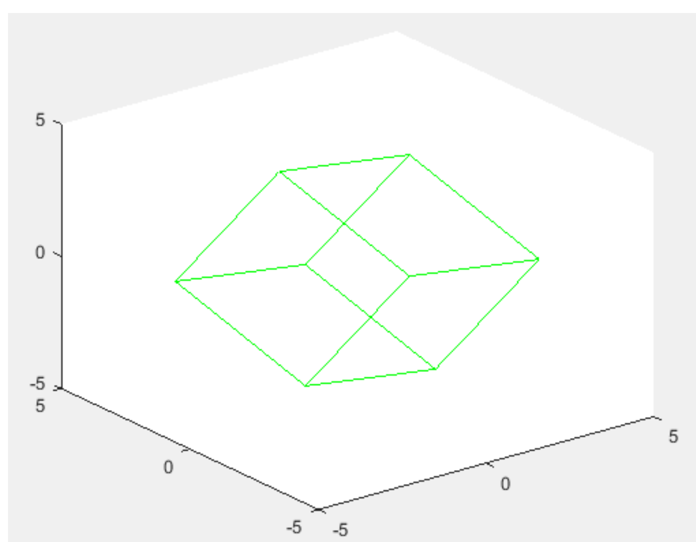
rke2_4_Animate.m | corke.m | Corke_2_4_Draw_Cube.m | +

```
function Corke2_4_Animate(length, width, height)

r = 0; p = 0; y = 0;

maxDim = max([length, width, height]);

while true
     r = r+1;p = p+1;y = y+1;
     Corke_2_4_Draw_Cube(length, width, height, r, p, y, maxDim);
     drawnow;
end

end
```





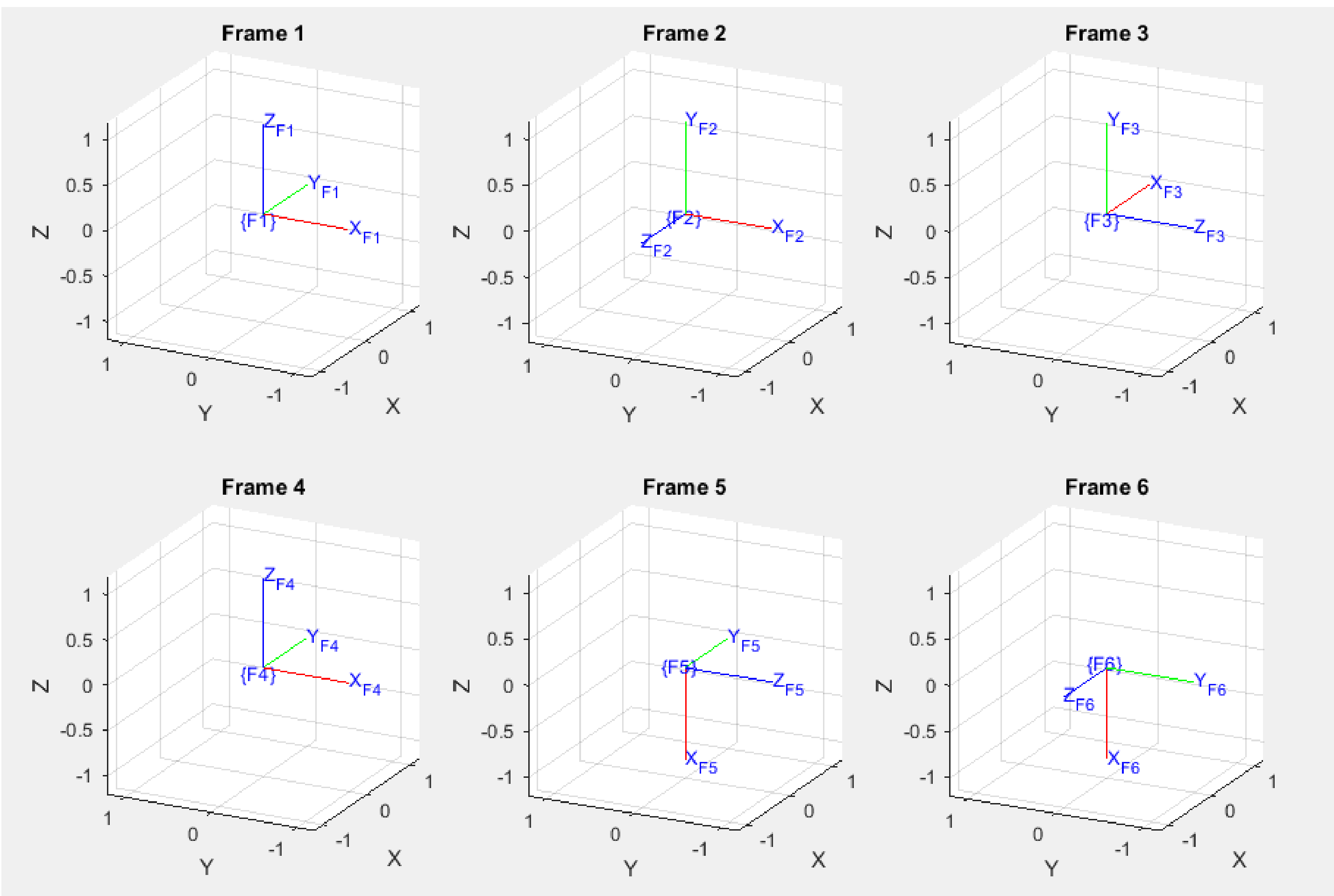




6. Generate the sequence of plots shown in Fig. 2.11.

2_14_QUAT.m | NIKU_2_14.m | Corke2_4_FUNC.m | corke.m | Corke2_6.m* | cariag2_13.m | NIKU_2_27.m

```
fig= [1 0 0 0 ; 0 1 0 0 ; 0 0 1 0 ; 0 0 0 1];
rot_z_n= [0 1 0 0 ; -1 0 0 0 ; 0 0 1 0 ; 0 0 0 1 ];
rot_x = [1 0 0 0 ;0 0 -1 0 ;0 1 0 0 ; 0 0 0 1];
rot_y = [0 0 1 0 ; 0 1 0 0 ;-1 0 0 0 ; 0 0 0 1];
fig1 = rot_z_n * fig; fig2 = fig1 * rot_x; fig3 = fig2 * rot_y ;
fig5 = fig1 * rot_y; fig6 = fig5 * rot_x;
subplot(2,3,1); trplot(fig1,'frame','F1','rgb');title('Frame 1')
subplot(2,3,2); trplot(fig2,'frame','F2','rgb');title('Frame 2')
subplot(2,3,3); trplot(fig3,'frame','F3','rgb');title('Frame 3')
subplot(2,3,4); trplot(fig1,'frame','F4','rgb');title('Frame 4')
subplot(2,3,5); trplot(fig5,'frame','F5','rgb');title('Frame 5')
subplot(2,3,6); trplot(fig6,'frame','F6','rgb');title('Frame 6')
```



B) From [Niku 2010], solve the following problems:

2.5. The following frame B was moved a distance of $d = (5, 2, 6)^T$. Find the new location of the frame relative to the reference frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & -1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

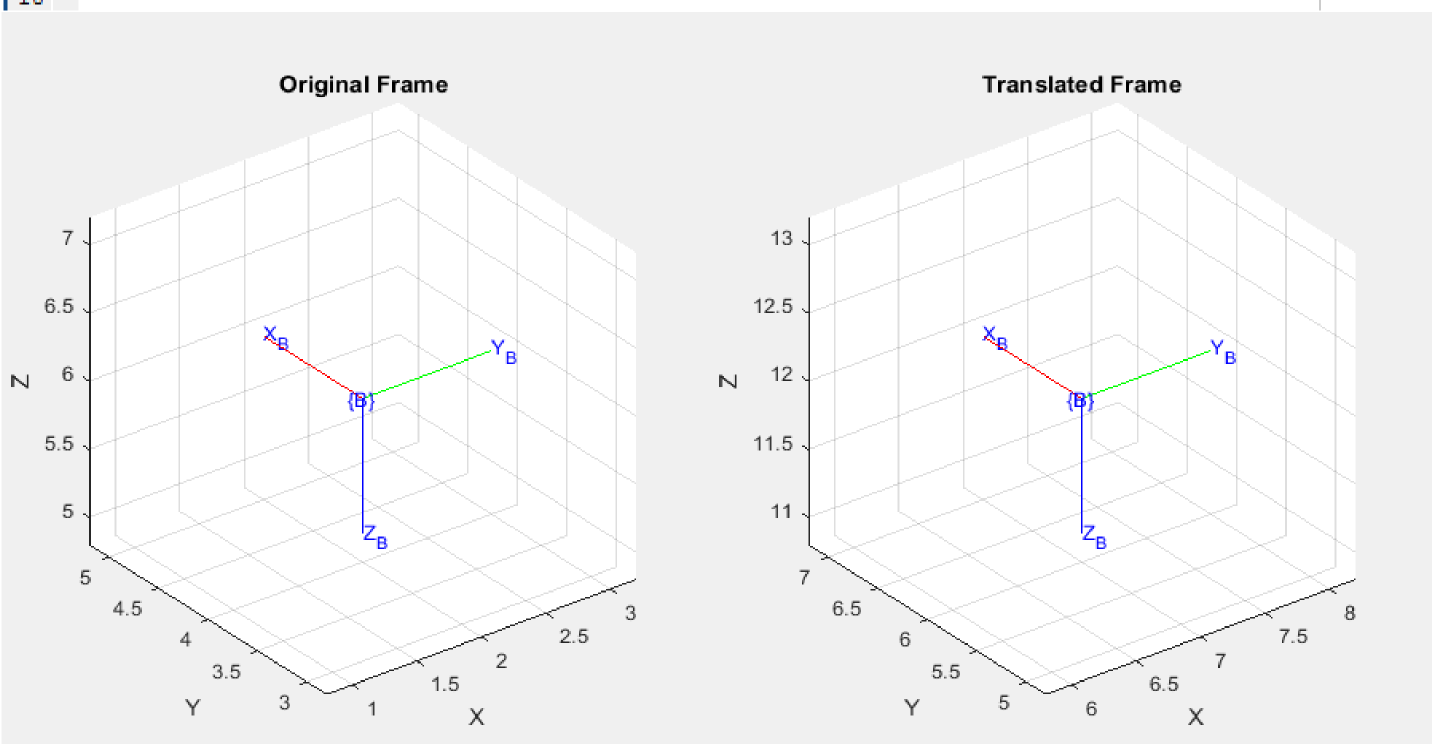
<u>ANS</u>

$$\mathrm{B1} = \mathrm{tran}(5,2,6) * B \; = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 2 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & -1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{7} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{6} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{12} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU2_5.m

NIKU2_5.m

```
B = [0 1 0 2 ; 1 0 0 4 ; 0 0 -1 6 ; 0 0 0 1];
Trans = [1 0 0 5 ; 0 1 0 2 ; 0 0 1 6 ; 0 0 0 1];
B_Translated = Trans * B ;
subplot(1,2,1);
trplot(B,'frame','B','rgb');
title('Original Frame')
subplot(1,2,2);
trplot(B_Translated,'frame','B','rgb');
title('Translated Frame')
```



2.6. For frame F, find the values of the missing elements and complete the matrix representation of the frame.

$$F = \begin{bmatrix} ? & 0 & -1 & 5 \\ ? & 0 & 0 & 3 \\ ? & -1 & 0 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

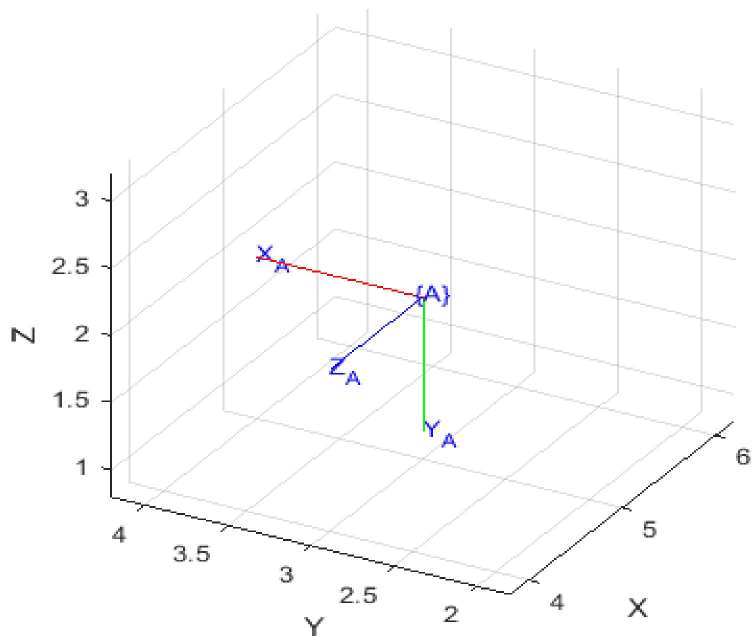
ANS

- Using cross product to get n → O * A = N

$$\begin{bmatrix} i & j & k \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix} = i \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} - j \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} + k \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}$$

Therefore, $\mathrm{n} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

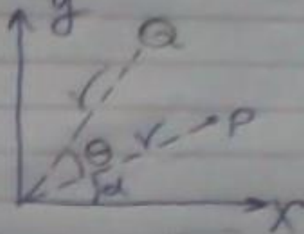
$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & -\mathbf{1} & \mathbf{5} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{2} \\ \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{6} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$



2.9. Derive the matrix that represents a pure rotation about the z-axis of the reference frame.

2.9

⇒ rotation about z axis means z is Fixed and rotate the other 2 axes



$$\therefore P = \begin{pmatrix} r\cos\alpha \\ r\sin\alpha \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$r' = \begin{pmatrix} r\cos(\alpha+\theta) \\ r\sin(\alpha+\theta) \end{pmatrix} = \begin{pmatrix} r\cos\alpha\cos\theta - r\sin\alpha\sin\theta \\ r\sin\theta\cos\alpha + r\sin\alpha\cos\theta \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

$$\therefore \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{pmatrix}$$

$$\therefore \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$\therefore$ in 3d $\because$ Z is constant $(3,3) = 1$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

2.11. Find the coordinates of point $P(2,3,4)^T$ relative to the reference frame after a rotation of 45° about the x-axis.

ANS

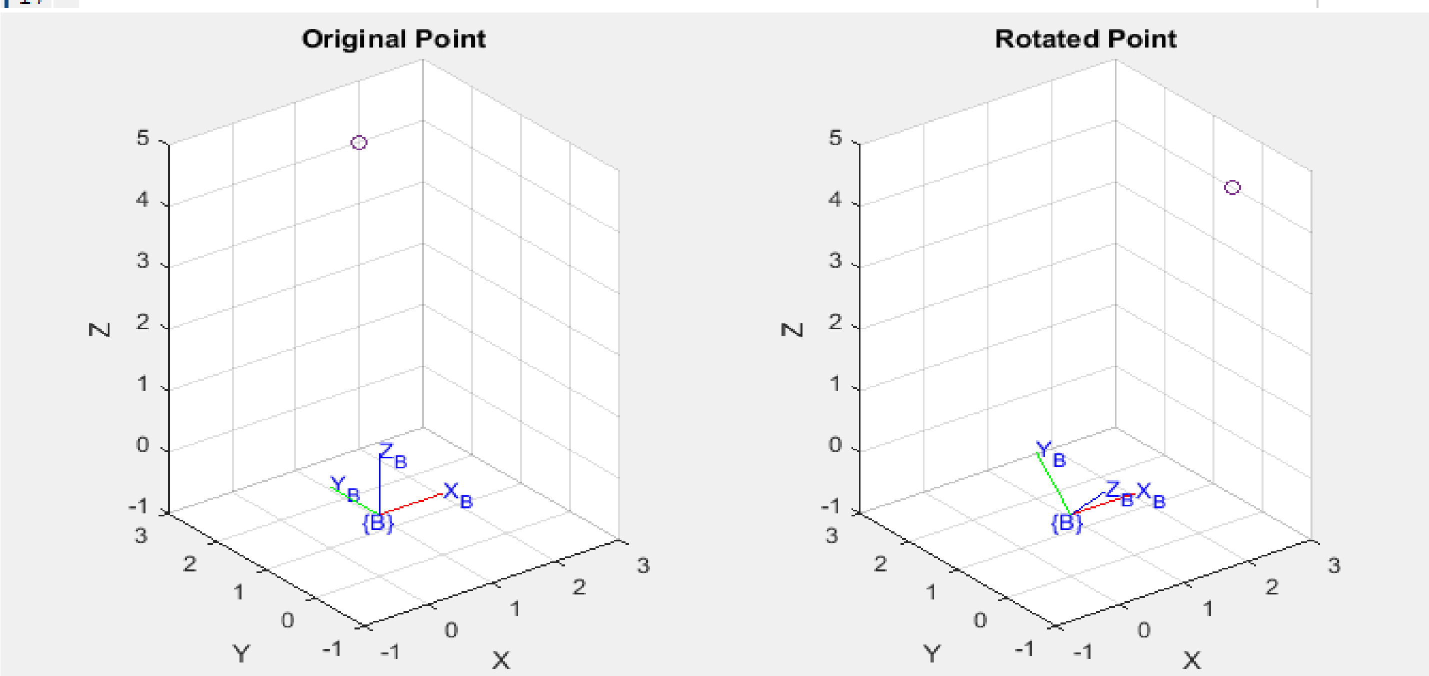
$$P1 = \text{rot}\,(X = 45) * P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C(45) & -S(45) \\ 0 & S(45) & C(45) \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.707 & -0.707 \\ 0 & 0.707 & 0.707 \end{bmatrix} * \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} = \begin{bmatrix} \mathbf{2} \\ \mathbf{-0.707} \\ \mathbf{4.95} \end{bmatrix}$$

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_11.m

NIKU2_5.m | NIKU_2_11.m

```
Orignal_frame = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
X_ROT = [1 0 0 0;0 0.707 -0.707 0;0 0.707 0.707 0 ;0 0 0 1];
Point = [2 ; 3 ; 4 ;1];
Rotated_point= X_ROT * Point;
Rotated_Frame=X_ROT *Orignal_frame;
subplot(1,2,1);
x=[-1 3 -1 3 -1 5];
trplot(Orignal_frame,'frame','B','rgb' ,'axis', x );
hold on
plot3( Point(1),Point(2),Point(3), 'o');
title('Original Point')
subplot(1,2,2);
trplot(Rotated_Frame,'frame','B','rgb' ,'axis', x );
hold on
plot3( Rotated_point(1),Rotated_point(2),Rotated_point(3), 'o');
title('Rotated Point')
```



2.14. A point P in space is defined as ${}^{B}P = (5,3,4)^T$ relative to frame B which is attached to the origin of the reference frame A and is parallel to it. Apply the following transformations to frame B and find ${}^{A}P$. Using the 3-D grid, plot the transformations and the result and verify it. Also verify graphically that you would not get the same results if you apply the transformations relative to the current frame:

- Rotate 90° about x-axis; then
- Translate 3 units about y-axis, 6 units about z-axis, and 5 units about x-axis; then,
- Rotate 90° about z-axis.

<u>ANS</u>

A_B = ROT (Z= 90) Trans (X=5, Y =3, Z=6) * ROT (X=90) * B

<u>First Step (Rot (x = 90))</u>

<u>For the Frame:</u>

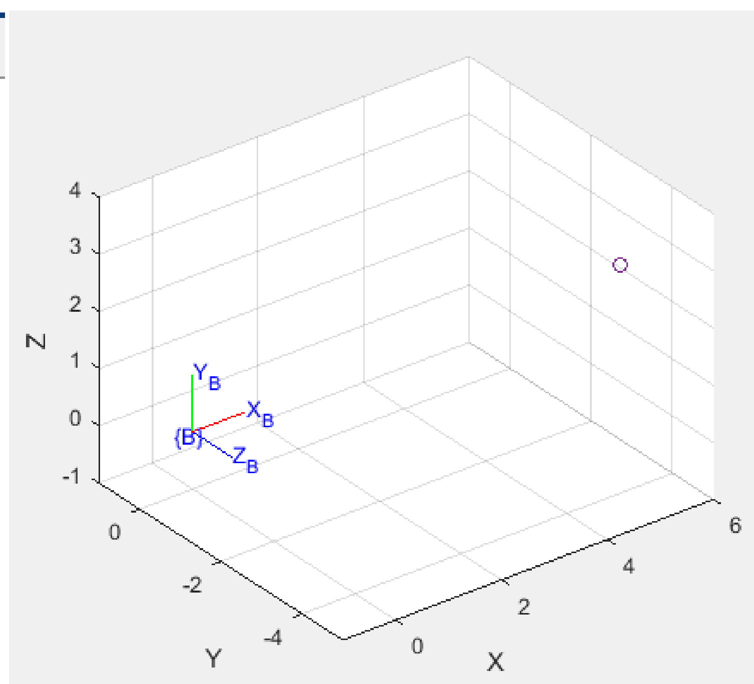
$$\mathrm{B1} = \mathrm{rot}(\mathrm{x}=90) * B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

<u>For the point:</u>

$$\mathrm{P1} = \mathrm{rot}(\mathrm{x}=90) * P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 \\ 3 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{5} \\ \mathbf{-4} \\ \mathbf{3} \\ \mathbf{1} \end{bmatrix}$$

2_14.m*

```
X_ROT= [1 0 0 0;0 0 -1 0;0 1 0 0;0 0 0 1];
B = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
P = [5 ; 3 ; 4 ; 1];
B1 = X_ROT* B;
P1 = X_ROT* P;
x=[-1 6 -5 1 -1 4];
trplot(B1,'frame','B','rgb' ,'axis', x );
hold on
plot3( P1(1),P1(2),P1(3), 'o');
```



Second Step (Trans (X=5, Y =3, Z=6))

For the frame:

$$B2 = \text{trans}\ (5,3,6) * B1 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{5} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{3} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{6} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

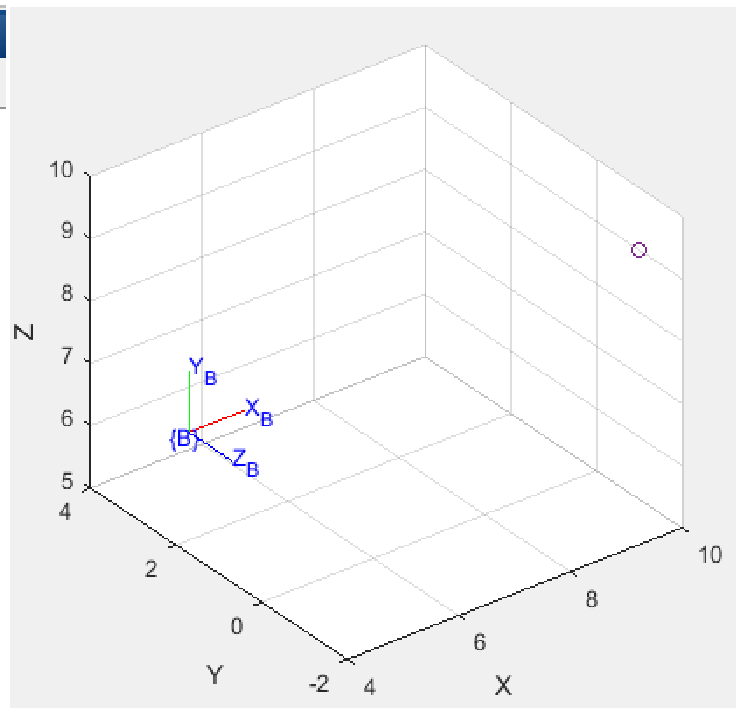
For the point:

$$P2 = \text{trans}\ (5,3,6) * P1 = \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 5 \\ -4 \\ 3 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{10} \\ \mathbf{-1} \\ \mathbf{9} \\ \mathbf{1} \end{bmatrix}$$

D:\4th yaer\2nd term\robotics\matlab\rvctools\NIKU_2_14.m

2_14.m

```
X_ROT= [1 0 0 0;0 0 -1 0;0 1 0 0;0 0 0 1];
trans = [1 0 0 5;0 1 0 3;0 0 1 6;0 0 0 1];
B = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
P = [5 ; 3 ; 4 ; 1];
B1 = X_ROT* B;
P1 = X_ROT* P;
B2 = trans*B1;
P2 = trans*P1;
x=[4 10 -2 4 5 10];
trplot(B2,'frame','B','rgb' ,'axis', x );
hold on
plot3( P2(1),P2(2),P2(3), 'o');
```



Third Step (ROT (Z= 90))

For the frame:

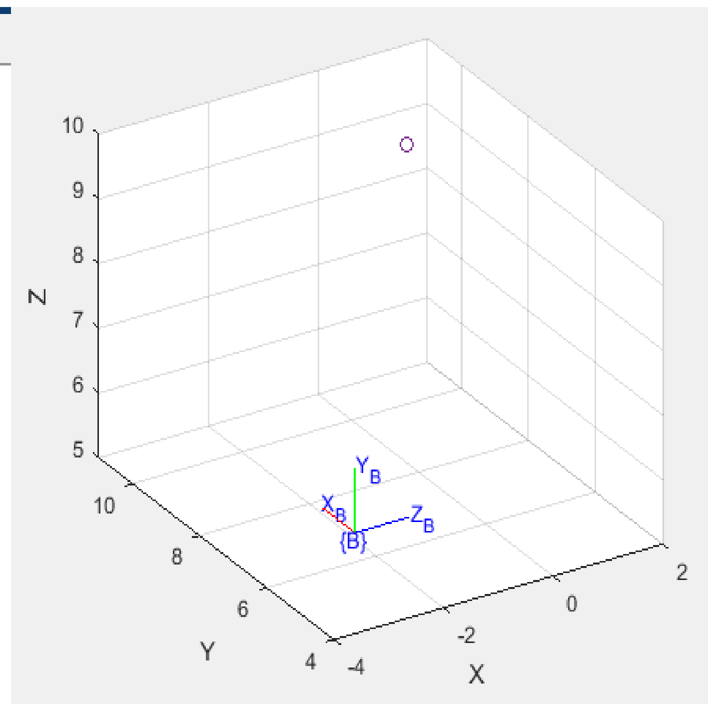
$$B3 = \mathrm{rot}(Z = 90) * B2 \ = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 0 & -1 & 3 \\ 0 & 1 & 0 & 6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{-3} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{5} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{6} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

For the point:

$$P3 = \mathrm{rot}(Z = 90) * P2 \ = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 10 \\ -1 \\ 9 \\ 1 \end{bmatrix} \ = \begin{bmatrix} \mathbf{1} \\ \mathbf{10} \\ \mathbf{9} \\ \mathbf{1} \end{bmatrix}$$

_2_14.m

```
X_ROT = [1 0 0 0;0 0 -1 0;0 1 0 0;0 0 0 1];
trans = [1 0 0 5;0 1 0 3;0 0 1 6;0 0 0 1];
Z_ROT = [0 -1 0 0;1 0 0 0;0 0 1 0;0 0 0 1];
B = [1 0 0 0;0 1 0 0;0 0 1 0;0 0 0 1];
P = [5 ; 3 ; 4 ; 1];
B1 = X_ROT* B;
P1 = X_ROT*P;
B2 = trans*B1;
P2 = trans*P1;
B3 = Z_ROT*B2;
P3 = Z_ROT*P2;
x=[4 10 -2 4 5 10];
trplot(B3,'frame','B','rgb' ,'axis', x );
hold on
plot3( P3(1),P3(2),P3(3), 'o');
```



2.17. The frame B of Problem 2.16 is rotated 90° about the a-axis, 90° about the y-axis, then translated 2 and 4 units relative to the x- and y-axes respectively, then rotated another 90° about the n-axis. Find the new location and orientation of the frame.

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ANS

We have 4 transformations

- 2 for the rotated frame which are post calculated.
- 2 for the reference frame.

Therefore, the seq is

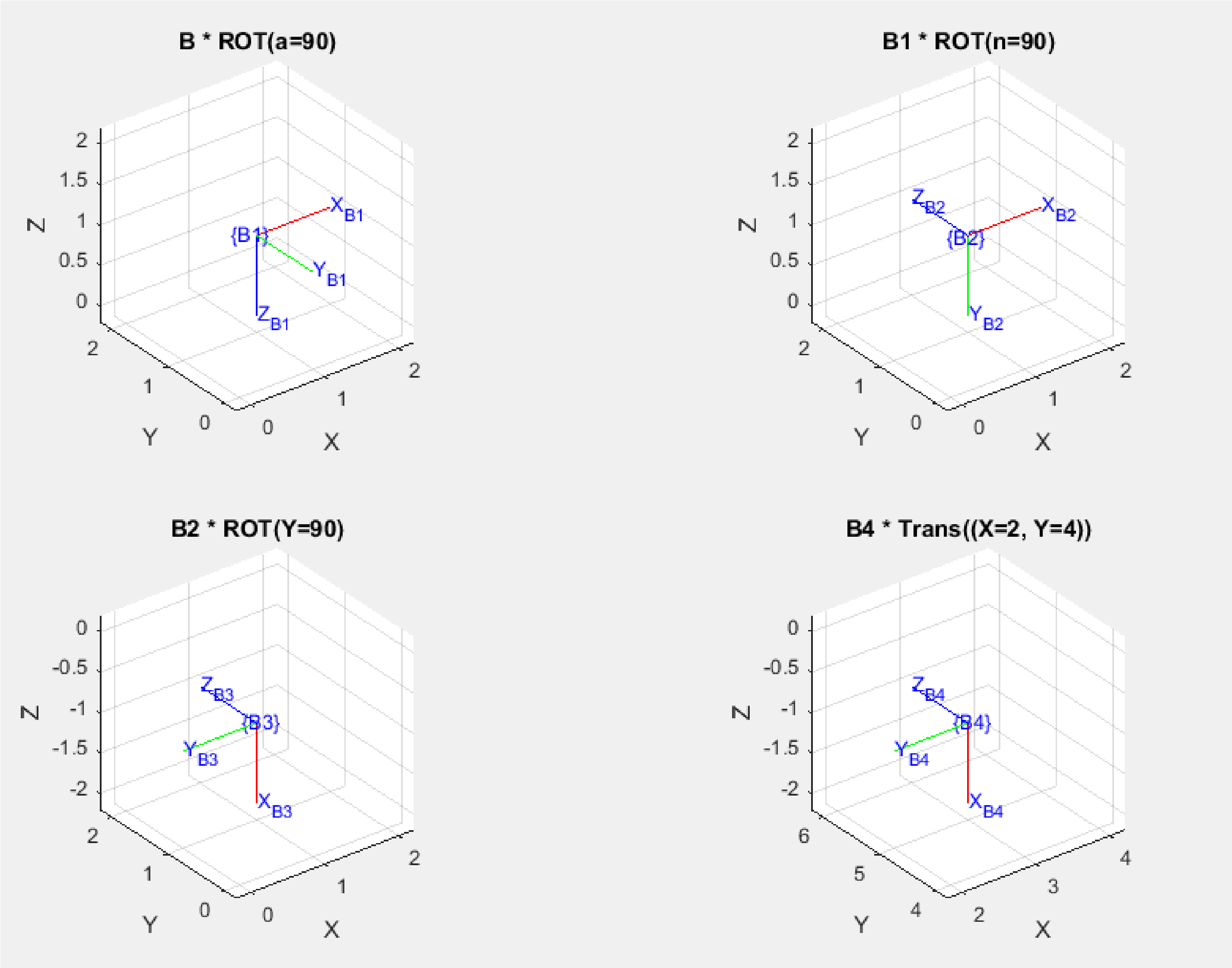
B_new = Trans (x=2, y =4) * ROT (y=90) * B * ROT (a= 90) * ROT (n=90)

$$B_{new} = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$* \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 3 \\ 0 & 0 & 1 & 5 \\ -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_17.m*

NIKU_2_17.m*

```
ROT_n = [1 0 0 0;0 0 -1 0;0 1 0 0;0 0 0 1];
ROT_a = [0 -1 0 0;1 0 0 0;0 0 1 0;0 0 0 1];
ROT_y= [0 0 1 0;0 1 0 0;-1 0 0 0;0 0 0 1];
Trans = [1 0 0 2 ; 0 1 0 4 ;0 0 1 0 ; 0 0 0 1];
B = [0 1 0 1;1 0 0 1;0 0 -1 1;0 0 0 1];
B1 = B * ROT_a;
B2 = B1* ROT_n;
B3 = ROT_y * B2;
B4 = Trans * B3;
subplot(2,2,1);
trplot(B1,'frame','B1','rgb');
title('B * ROT(a=90)')
subplot(2,2,2);
trplot(B2,'frame','B2','rgb');
title('B1 * ROT(n=90)')
subplot(2,2,3);
trplot(B3,'frame','B3','rgb');
title('B2 * ROT(Y=90)')
subplot(2,2,4);
trplot(B4,'frame','B4','rgb');
title('B4 * Trans((X=2, Y=4))')
```



B * ROT(a=90)
B1 * ROT(n=90)
B2 * ROT(Y=90)
B4 * Trans((X=2, Y=4))
{B1}
X B1
Y B1
Z B1
{B2}
X B2
Y B2
Z B2
{B3}
X B3
Y B3
Z B3
{B4}
X B4
Y B4
Z B4
X
Y
Z

2.20. Calculate the inverse of the matrix B of Problem 2.17.

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\mathbf{p}\cdot\mathbf{n} \\ o_x & o_y & o_z & -\mathbf{p}\cdot\mathbf{o} \\ a_x & a_y & a_z & -\mathbf{p}\cdot\mathbf{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{B} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} and \;\; B^{-1} = \begin{bmatrix} 0 & 1 & 0 & -((1*0)+(1*1)+(1*0)) \\ 1 & 0 & 1 & -((1*1)+(1*0)+(1*0)) \\ 0 & 0 & -1 & -((1*0)+(1*0)+(1*-1)) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

therefore, $\boldsymbol{B^{-1}} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$

2.24. Suppose that a robot is made of a Cartesian and Euler combination of joints. Find the necessary Euler angles to achieve the following:

$$T = \begin{bmatrix} 0.527 & -0.574 & 0.628 & 4 \\ 0.369 & 0.819 & 0.439 & 6 \\ -0.766 & 0 & 0.643 & 9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} n_x C\phi + n_y S\phi & o_x C\phi + o_y S\phi & a_x C\phi + a_y S\phi & 0 \\ -n_x S\phi + n_y C\phi & -o_x S\phi + o_y C\phi & -a_x S\phi + a_y C\phi & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\theta C\psi & -C\theta S\psi & S\theta & 0 \\ S\psi & C\psi & 0 & 0 \\ -S\theta C\psi & S\theta S\psi & C\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

SIN 180- atan(ang)	ALL = atan(ang)
TAN 180+atan(ang)	COS 360+atan(ang)

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_24.m*

NIKU_2_24.m*

```
T = [0.527 -0.574 0.628 4 ; 0.369 0.819 0.439 6; -0.766 0 0.643 9 ; 0 0 0 1];
EUL_ANG = tr2eul (T);
EUL_ANG = EUL_ANG*180/pi;
```

Command Window

```
EUL_ANG =

   34.9553   49.9974    0.0283
```

By equating [(3,1)]

$-a_x S\phi + a_y C\phi = 0 \Rightarrow \tan\phi = \frac{S\phi}{C\phi} = \frac{a_y}{a_x}$

∴ $\tan\phi = \frac{0.439}{0.628} \Rightarrow \phi = 34.95 \simeq 35°$

∵ tan +ve ⇒ $\boxed{1}$ & $\boxed{3}$ $\phi = (35°, 215°)$

By equating [(1,2)], [(2,2)] any of the 2Ø can be used

∴ $S\psi = -n_x S\phi + n_y C\phi \Rightarrow -0.527 S(35) + 0.369 C(35)$

∴ $C\psi = -a_x S\phi + a_y C\phi \Rightarrow -0.628 S(35) + 0.439 C(35)$

∴ $\tan\psi = 0.012 \Rightarrow \phi = .68° \simeq 1°$

∵ tan +ve → $\boxed{1}$ & $\boxed{3}$ ∴ $\psi = (1°, 181°)$

By equating [(1,3)] [(3,3)]

∴ $-S\theta C\psi = n_z \Rightarrow S\theta = \frac{-n_z}{C\psi} = \frac{0.766}{C(1)} = 0.766$

∴ $C\theta = a_z = 0.643$

∴ $\tan\theta = 1.19 \Rightarrow \theta = 49.9° = 50°$ ∴ $\theta = (50, -50)$

Φ = [35 , 215] Θ = [50 , -50] **ψ = [1 , 181]**

2.27. A frame ^{U}F was moved along its own n-axis a distance of 5 units, then rotated about its o-axis an angle of 60°, followed by a rotation of 60° about the z-axis, then translated about its a-axis for 3 units, and finally rotated 45° about the x-axis.

- Calculate the total transformation performed.
- What angles and movements would we have to make if we were to create the same location and orientation using Cartesian and RPY configurations?

The sequence is

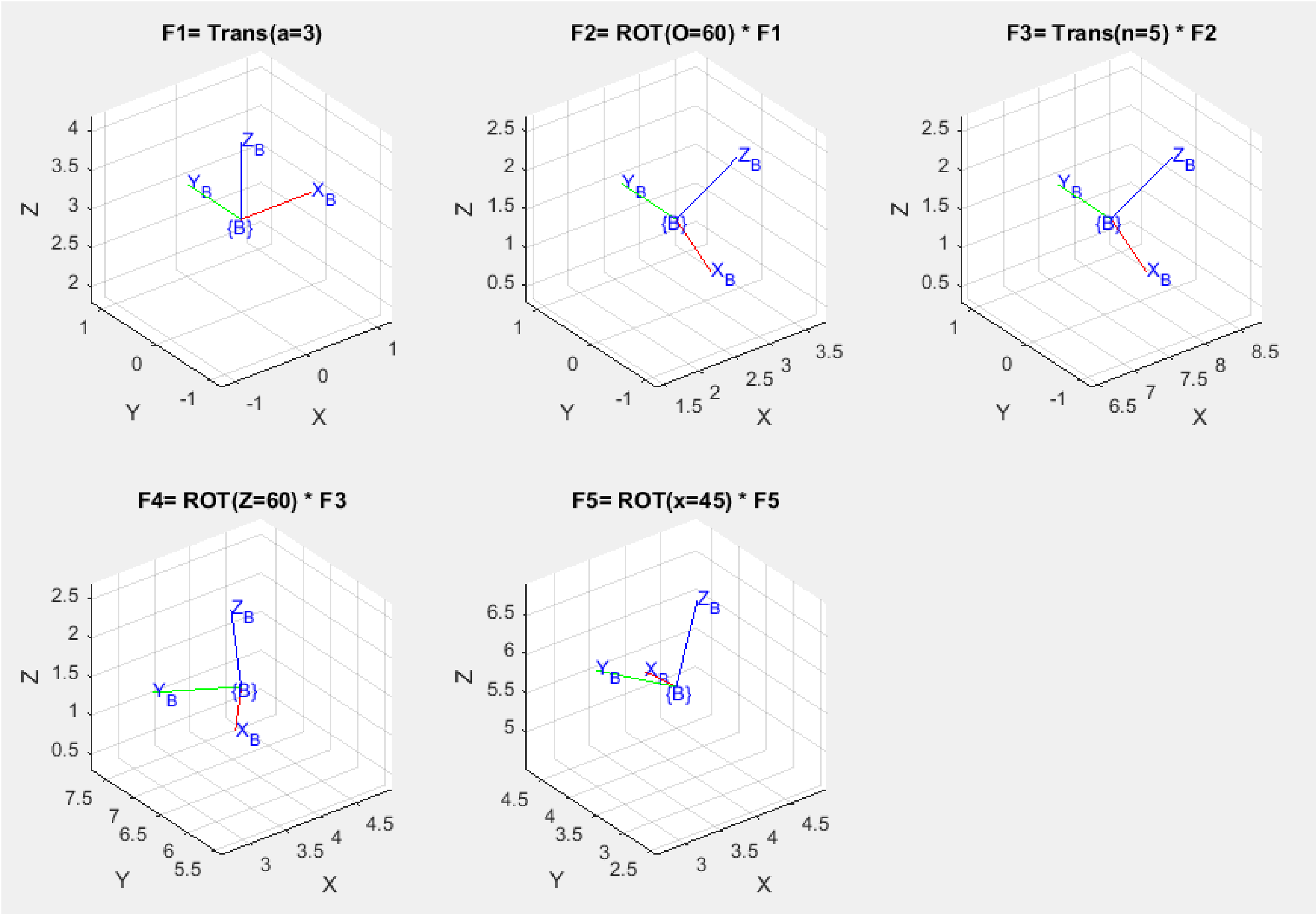
F (new)=ROT(X=45)*ROT(Z=60)*Trans(n=5, O=0, a=0)*ROT(O= 60)*Trans(n=0, O=0, a=3)

$$T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & C(45) & -S(45) & 1 \\ 0 & S(45) & C(45) & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C(60) & -S(60) & 0 & 0 \\ S(60) & C(60) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} C(60) & 0 & S(60) & 0 \\ 0 & 1 & 0 & 0 \\ -S(60) & 0 & C(60) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0.25} & \mathbf{-0.866} & \mathbf{0.433} & \mathbf{3.799} \\ \mathbf{0.918} & \mathbf{0.354} & \mathbf{0.177} & \mathbf{3.592} \\ \mathbf{-0.306} & \mathbf{0.354} & \mathbf{0.884} & \mathbf{5.713} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

2_17.m | NIKU_2_27.m | +

```
x_ROT = [1 0 0 0;0 .707 -.707 0;0 .707 .707 0;0 0 0 1];
z_ROT = [.5 -.866 0 0;.866 .5 0 0;0 0 1 0;0 0 0 1];
n_Trans = [1 0 0 5 ; 0 1 0 0 ;0 0 1 0 ; 0 0 0 1];
o_ROT = [.5 0 .866 0;0 1 0 0;-.866 0 .5 0;0 0 0 1];
a_Trans = [1 0 0 0 ; 0 1 0 0 ;0 0 1 3 ; 0 0 0 1];
F1=a_Trans;
F2=o_ROT * F1;
F3=n_Trans * F2;
F4=z_ROT * F3;
F5=x_ROT * F4;
subplot(2,3,1);
trplot(F1,'frame','B','rgb');
title('F1= Trans(a=3)')
subplot(2,3,2);
trplot(F2,'frame','B','rgb');
title('F2= ROT(O=60) * F1')
subplot(2,3,3);
trplot(F3,'frame','B','rgb');
title('F3= Trans(n=5) * F2')
subplot(2,3,4);
trplot(F4,'frame','B','rgb');
title('F4= ROT(Z=60) * F3')
subplot(2,3,5);
trplot(F5,'frame','B','rgb');
title('F5= ROT(x=45) * F5')
```



B) For **Cartesian** the angles and movements as the F5, point (3.799, 3.592, 5.713)

C) For RPY

$$\begin{bmatrix} n_x C\phi_a + n_y S\phi_a & o_x C\phi_a + o_y S\phi_a & a_x C\phi_a + a_y S\phi_a & 0 \\ n_y C\phi_a - n_x S\phi_a & o_y C\phi_a - o_x S\phi_a & a_y C\phi_a - a_x S\phi_a & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 0.25 & -0.866 & 0.433 & 3.799 \\ 0.918 & 0.354 & 0.177 & 3.592 \\ -0.306 & 0.354 & 0.884 & 5.713 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} C\phi_o & S\phi_o S\phi_n & S\phi_o C\phi_n & 0 \\ 0 & C\phi_n & -S\phi_n & 0 \\ -S\phi_o & C\phi_o S\phi_n & C\phi_o C\phi_n & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_27.m

NIKU_2_24.m NIKU_2_27.m

```
RPY_ANG = tr2rpy (F5);
RPY_ANG = RPY_ANG*180/pi;
```

By equating [1,2]

$n_y C\phi_a - n_x S\phi_a = 0 \Rightarrow \tan\phi_a = \frac{S\phi_a}{C\phi_a} = \frac{n_y}{n_x}$

$\therefore \tan\phi_a = \frac{0.918}{0.25} \Rightarrow \phi_a = 74.76^\circ$

$\therefore$ Tan +ve $\rightarrow$ [1] & [3] $\phi_a = [74.8, 254.8^\circ]$

By equating [[2,2]] [[3,2]]

$\therefore C\phi_n = o_y C\phi_a - o_x S\phi_a \Rightarrow C\phi_n = 0.93$

$\therefore -S\phi_n = a_y C\phi_a - a_x S\phi_a \Rightarrow S\phi_n = +0.371$

$\cancel{\therefore \tan\phi_n = \frac{0.93}{0.364}}$ $\tan\phi_n = \frac{0.371}{0.93}$ $\therefore \phi_n = 21.7^\circ$

$\therefore$ Tan +ve $\rightarrow$ [1], [3] $\phi_n = [21.7, 201.7]$

By equating [(1,1)] [(1,3)]

$C\phi_o = n_x C\phi_a + n_y S\phi_a \Rightarrow C\phi_o = 0.951$

$-S\phi_o = n_z \rightarrow S\phi_o = 0.306$

$\therefore \tan\phi_o = \frac{0.306}{0.951} = 17.83$

$\pm$

$\therefore \phi_o = \phi_o = [17.83, -17.83]$

Φa= [74.8 , 254.8] Φn= [21.7 , 201.7] Φo= [17.83 , -17.83]

3) From [Craig 2005], solve the following exercises:

2.4 [16] A frame $\{B\}$ is located initially coincident with a frame $\{A\}$. We rotate $\{B\}$ about $\hat{Z}_B$ by 30 degrees, and then we rotate the resulting frame about $\hat{X}_B$ by 45 degrees. Give the rotation matrix that will change the description of vectors from ${}^{B}P$ to ${}^{A}P$.

ANS

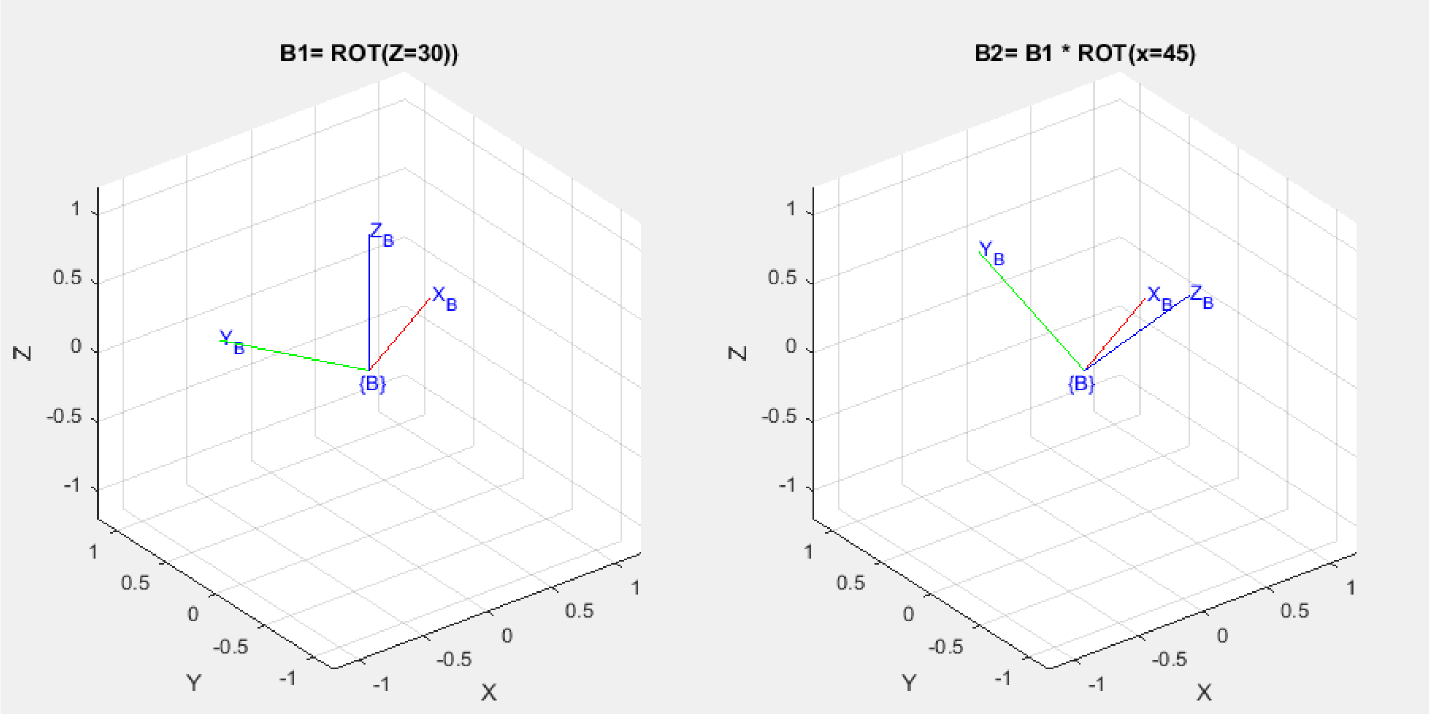
Note : We rotate around the object B frame

$$F = \begin{bmatrix} C(30) & -S(30) & 0 & 0 \\ S(30) & C(30) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & C(45) & -S(45) & 1 \\ 0 & S(45) & C(45) & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.8 & -0.354 & 0.354 & 0 \\ 0.5 & 0.61 & -0.61 & 0 \\ 0 & 0.71 & 0.71 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\carrige_2_5.m

carrige_2_5.m

```
x_ROT = [1 0 0 0;0 .707 -.707 0;0 .707 .707 0;0 0 0 1];
z_ROT = [.866 -.5 0 0;0.5 .866 0 0;0 0 1 0;0 0 0 1];
B1 = z_ROT;
B2 = B1 * x_ROT;
subplot(1,2,1);
trplot(B1,'frame','B','rgb');
title('B1= ROT(Z=30))')
subplot(1,2,2);
trplot(B2,'frame','B','rgb');
title('B2= B1 * ROT(x=45)')
```



2.5 [13] ${}^A_B R$ is a 3×3 matrix with eigenvalues $1, e^{+ai}$, and e^{-ai}, where $i = \sqrt{-1}$. What is the physical meaning of the eigenvector of ${}^A_B R$ associated with the eigenvalue 1?

ANS

As in Angle axis representation we have only one axis of rotation and one angle.

If we have Rotation Matrix **R** and we want to get the vector of the Angle axis representation **V,**

From the definition of eigenvalues and eigenvectors **R*V** = λ***V**

where v is the eigenvector corresponding to λ. For the case λ =1 then

$$\mathbf{R*V} = \mathbf{V}$$

So, when the eigenvalue is =1 the corresponding eigenvector is the Angle axis representation vector, the rotation vector.

2.8 [29] Write a subroutine that changes representation of orientation from rotation-matrix form to equivalent angle–axis form. A Pascal-style procedure declaration would begin

```
Procedure RMTOAA (VAR R:mat33; VAR K:vec3; VAR theta: real);
```

Write another subroutine that changes from equivalent angle–axis representation to rotation-matrix representation:

```
Procedure AATORM(VAR K:vec3; VAR theta: real: VAR R:mat33);
```

Write the routines in C if you prefer.
Run these procedures on several cases of test data back-to-back and verify that you get back what you put in. Include some of the difficult cases!

2.9 [27] Do Exercise 2.8 for roll, pitch, yaw angles about fixed axes.

ANS

The script tested by the example (2.20) in **[Niku 2010]**

Example 2.20

The desired final position and orientation of the hand of a Cartesian-RPY robot is given below. Find the necessary RPY angles and displacements.

$$^{R}T_{P} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.354 & -0.674 & 0.649 & 4.33 \\ 0.505 & 0.722 & 0.475 & 2.50 \\ -0.788 & 0.160 & 0.595 & 8 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Solution: From the above equations, we find two sets of answers:

$$\phi_a = ATAN2(n_y, n_x) = ATAN2(0.505, 0.354) = 55^\circ \text{ or } 235^\circ$$
$$\phi_o = ATAN2(-n_z, (n_x C\phi_a + n_y S\phi_a)) = ATAN2(0.788, 0.616) = 52^\circ \text{ or } 128^\circ$$
$$\phi_n = ATAN2((-a_y C\phi_a + a_x S\phi_a), (o_y C\phi_a - o_x S\phi_a))$$
$$= ATAN2(0.259, 0.966) = 15^\circ \text{ or } 195^\circ$$

Python Script to convert Rotational to RPY

```python
import numpy as np
import math
R = [0.354 , -0.674 , 0.649 , 4.33 , 0.505 ,0.722, 0.475, 2.50,-0.788 , 0.160 , 0.595 , 8, 0 ,0 ,0 ,1]
R = np.reshape(R, (4, 4))
def RPY_FROM_R (R):
    array = np.zeros (6)
    yaw1 = math.atan2 (R[1,0] ,R[0,0] )
    pitch1= math.atan2 (-R[2,0] ,(R[0,0]*math.cos(yaw1)+R[1,0]*math.sin(yaw1) ) )
    roll1 = math.atan2 (((-R[1,2]*math.cos(yaw1))+(R[0,2]*math.sin(yaw1)))
            ,((R[1,1]*math.cos(yaw1))-(R[0,1]*math.sin(yaw1))))

    yaw2 = math.atan2 (-R[1,0] ,-R[0,0] )
    pitch2= math.atan2 (-R[2,0] ,-(R[0,0]*math.cos(yaw1)+R[1,0]*math.sin(yaw1) ) )
    roll2 = math.atan2 (-((-R[1,2]*math.cos(yaw1))+(R[0,2]*math.sin(yaw1)))
            ,-((R[1,1]*math.cos(yaw1))-(R[0,1]*math.sin(yaw1))))

    yaw1= int (np.ceil(math.degrees(yaw1)))
    array[0]= yaw1
    pitch1= int (np.ceil(math.degrees(pitch1)))
    array[1]= pitch1
    roll1= int (np.ceil(math.degrees(roll1)))
    array[2]= roll1
    yaw2= int (np.ceil(math.degrees(yaw2)))
    array[3]= yaw2
    pitch2= int (np.ceil(math.degrees(pitch2)))
    array[4]= pitch2
    roll2= int (np.ceil(math.degrees(roll2)))
    array[5]= roll2

    for i in range (len (array)):
        if (array[i]<0):
            array[i] = array[i]+360
    print ("Pitch angles is (" + str (array [1]) + " ," +str (array [4] ) + ")")
    print ("roll angles is (" + str (array [2]) + " ," +str (array [5] ) + ")")
    print ("yaw angles is (" + str (array [0]) + " ," +str (array [3] ) + ")")
RPY_FROM_R (R)
```

OUTPUT

```
In [52]: runfile('C:/Users/Mostafa Hisham/carig2_9.py', wdir='C:/Users/Mostafa
Hisham')
Input Rotationl matrix
[[ 0.354 -0.674  0.649  4.33 ]
 [ 0.505  0.722  0.475  2.5  ]
 [-0.788  0.16   0.595  8.   ]
 [ 0.     0.     0.     1.   ]]
Pitch angles is (52.0 ,129.0)
roll angles is (15.0 ,195.0)
yaw angles is (55.0 ,235.0)
```

Script to convert from RPY to Rotational

```
def Rotational_from_RPY (roll_angle , Pitch_angle  ,   yaw_angle):
    roll_angle= math.radians(roll_angle)
    Pitch_angle= math.radians(Pitch_angle)
    yaw_angle= math.radians(yaw_angle)

    Rot_Z = [math.cos(roll_angle) , -math.sin(roll_angle) , 0 , 0 ,
             math.sin(roll_angle), math.cos(roll_angle),0,0, 0, 0 , 1 , 0, 0 ,0 ,0 ,1]
    Rot_Z= np.reshape(Rot_Z, (4, 4))
    Rot_Y = [math.cos(Pitch_angle) , 0, math.sin(Pitch_angle)  , 0 , 0 , 1, 0, 0
             , -math.sin(Pitch_angle) ,0 , math.cos(Pitch_angle),0 , 0 ,0 ,0 ,1]
    Rot_Y= np.reshape(Rot_Y, (4, 4))
    Rot_X = [1 , 0, 0, 0, 0 , math.cos(yaw_angle) ,- math.sin(yaw_angle), 0 , 0
             , math.sin(yaw_angle) , math.cos(yaw_angle),0 , 0 ,0 ,0 ,1]
    Rot_X= np.reshape(Rot_X, (4, 4))
    R1 = np.dot(Rot_Y,Rot_X)
    Rotational = np.dot(Rot_Z,R1)
    print ("________________________________")
    print ("Rotational array from RPY")
    print (Rotational)


Rotational_from_RPY(array1[0] ,array1[1]  ,array1[2] )
```

OUTPUT

```
In [64]: runfile('C:/Users/Mostafa Hisham/carig2_9.py', wdir='C:/Users/Mostafa Hisham')
Input Rotationl matrix
[[ 0.354 -0.674  0.649  4.33 ]
 [ 0.505  0.722  0.475  2.5  ]
 [-0.788  0.16   0.595  8.   ]
 [ 0.     0.     0.     1.   ]]
roll angles is (55.0 ,235.0)
pitch angles is (52.0 ,129.0)
yaw angles is (15.0 ,195.0)
________________________________
Rotational array from RPY
[[ 0.35312892 -0.67425794  0.64859555  0.        ]
 [ 0.50432036  0.72110015  0.47505321  0.        ]
 [-0.78801075  0.15934492  0.59468332  0.        ]
 [ 0.          0.          0.          1.        ]]

In [65]:
```

I used the output of the first function as input to the second function and the result was the input for the first function except the point because it wasn't required to put it in the array.

2.13 [21] The following frame definitions are given as known:

$$
{}^{U}_{A}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & 11.0 \\ 0.500 & 0.866 & 0.000 & -1.0 \\ 0.000 & 0.000 & 1.000 & 8.0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{B}_{A}T = \begin{bmatrix} 1.000 & 0.000 & 0.000 & 0.0 \\ 0.000 & 0.866 & -0.500 & 10.0 \\ 0.000 & 0.500 & 0.866 & -20.0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}^{C}_{U}T = \begin{bmatrix} 0.866 & -0.500 & 0.000 & -3.0 \\ 0.433 & 0.750 & -0.500 & -3.0 \\ 0.250 & 0.433 & 0.866 & 3.0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.
$$

Draw a frame diagram (like that of Fig. 2.15) to show their arrangement qualitatively, and solve for ${}^{B}_{C}T$.

$$
\mathbf{BTC} = \mathbf{BTA} * \boldsymbol{UTA^{-1}} * \boldsymbol{CTU^{-1}}
$$

- D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\cariag2_13.m

g2_13.m +

```
UTA=[0.866 -0.5 0 11; 0.5 0.866 0 -1; 0 0 1 8.0 ;0 0 0 1];
BTA=[1 0 0 0 ; 0 0.866 -0.5 10 ; 0 5 0.866 -20 ; 0 0 0 1];
CTU=[0.866 -0.5 0 -3 ; 0.433 0.75 -0.5 -3 ; 0.25 0.433 0.866 3
    ; 0 0 0 1];
BTC= BTA * inv(UTA)* inv(CTU);
x=[-5 15 -5 12 -21 10];
```

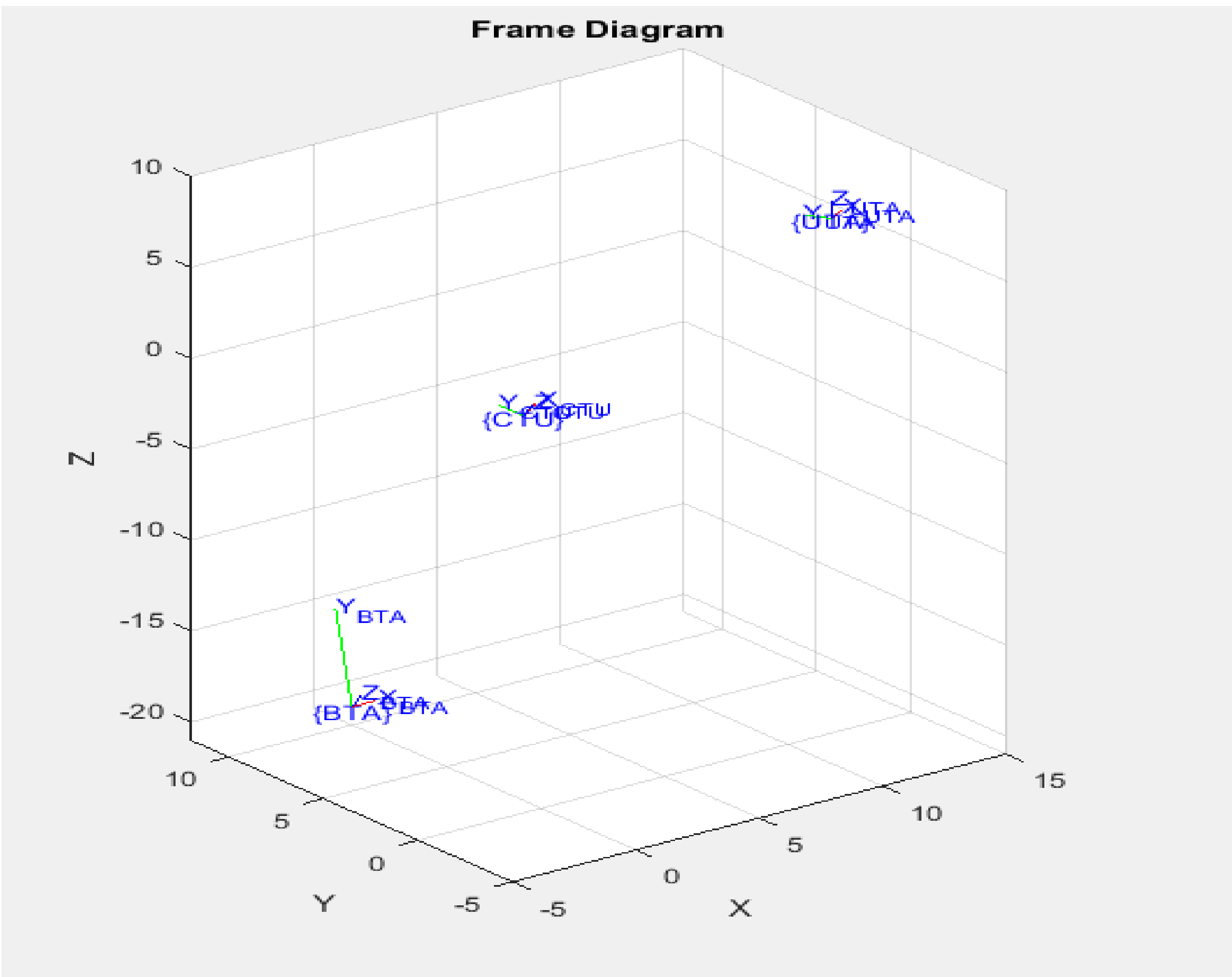
$$
\mathbf{BTC} = \begin{bmatrix} 0.5 & 0.75 & 0.43 & -6.6 \\ -0.75 & 0.63 & -0.22 & 19.75 \\ -4.33 & 1.73 & 2 & -8.99 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Frame diagram Script

D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\cariag2_13.m

g2_13.m

```
UTA=[0.866 -0.5 0 11; 0.5 0.866 0 -1; 0 0 1 8.0 ;0 0 0 1];
BTA=[1 0 0 0 ; 0 0.866 -0.5 10 ; 0 5 0.866 -20 ; 0 0 0 1];
CTU=[0.866 -0.5 0 -3 ; 0.433 0.75 -0.5 -3 ; 0.25 0.433 0.866 3 ; 0 0 0 1];
x=[-5 15 -5 12 -21 10];
title('F1= Trans(a=3)')
trplot(UTA,'frame','UTA','rgb' ,'axis', x );
hold on
trplot(BTA,'frame','BTA','rgb' ,'axis', x );
trplot(CTU,'frame','CTU','rgb' ,'axis', x );
title('Frame Diagram')
```



2.16 [22] A vector must be mapped through three rotation matrices:

$$^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P.$$

One choice is to first multiply the three rotation matrices together, to form ${}^{A}_{D}R$ in the expression

$$^{A}P = {}^{A}_{D}R\,{}^{D}P.$$

Another choice is to transform the vector through the matrices one at a time—that is,

$$^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}_{D}R\,{}^{D}P,$$

$$^{A}P = {}^{A}_{B}R\,{}^{B}_{C}R\,{}^{C}P,$$

$$^{A}P = {}^{A}_{B}R\,{}^{B}P,$$

$$^{A}P = {}^{A}P.$$

If ${}^{D}P$ is changing at 100 Hz, we would have to recalculate ${}^{A}P$ at the same rate. However, the three rotation matrices are also changing, as reported by a vision system that gives us new values for ${}^{A}_{B}R$, ${}^{B}_{C}R$, and ${}^{C}_{D}R$ at 30 Hz. What is the best way to organize the computation to minimize the calculation effort (multiplications and additions)?

ANS

First Method	Second Method
The three rotations will mutiply with each other and then multiply with the point	the result of each muliplication is a point and will multipy with the rotation
• Two rotaion matrices multiplication will result in 27 product and 18 addtion, and will happen 2 times. 27 * 2 * 30 = 1620 product 18 * 2 * 30 = 1080 addtion • The result rotation matrix will multpy by the point , 9 product and 6 addition 9 * 100 = 900 6 * 100 = 600 • The total 1620 + 900 = 2520 product 1080 * 600 = 1680 addition	• The rotation matrix will multpy by the point , 9 product and 6 addition ,and will happed 3 times • The total is 3*7= 27 product 3*6= 18 addition • The total 27 * 100 = 2700 product 18 * 100 = 1800 addition
Therefore, the first method is more effictive	

2.20 [20] Imagine rotating a vector Q about a vector $\hat{K}$ by an amount θ to form a new vector, Q'—that is,

$$Q' = R_K(\theta)Q.$$

Use (2.80) to derive **Rodriques's formula**,

$$Q' = Q\cos\theta + \sin\theta(\hat{K} \times Q) + (1 - \cos\theta)(\hat{K} \cdot \hat{Q})\hat{K}.$$

$$R_K(\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix}, \qquad (2.80)$$

where $c\theta = \cos\theta$, $s\theta = \sin\theta$, $v\theta = 1 - \cos\theta$, and ${}^A\hat{K} = [k_x k_y k_z]^T$. The sign of θ is determined by the right-hand rule, with the thumb pointing along the positive sense of ${}^A\hat{K}$.

ANS

Rodrigues's formula = $Q^{\sim} = Q\ C(\Theta) + S(\Theta)\ (K\ x\ Q) + \ v\Theta(K.Q)\ K$

- $(K\ x\ Q) = \begin{bmatrix} i & j & k \\ Kx & Ky & Kz \\ Q1 & Q2 & Q3 \end{bmatrix} = i\begin{bmatrix} Ky & Kz \\ Q2 & Q3 \end{bmatrix} - j\begin{bmatrix} Kx & Kz \\ Q1 & Q3 \end{bmatrix} + k\begin{bmatrix} Kx & Ky \\ Q1 & Q2 \end{bmatrix}$

$$= i\ (Ky.Q3 - Kz.Q2) - j\ (Kx.Q3 - Kz.Q1) + (Kx.Q2 - \ Ky.Q1)$$

Assume: $Q = [1 \quad 1 \quad 1]$

Therefore, $(K\ x\ Q) = i\ (Ky - Kz) - j\ (Kx - Kz) + k\ (Kx - \ Ky)$ $(K\ x\ Q) = \begin{bmatrix} Ky - Kz \\ Kx - Kz \\ Kx - Ky \end{bmatrix}$

- $(K.Q) = [Kx \quad Ky \quad Kz]\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = (Kx + Ky + Kz)$
- $v\Theta(K.Q)K = \ v\Theta.(Kx + Ky + Kz).Kx \ = \begin{bmatrix} Kx.v\Theta.(Kx + Ky + Kz) \\ Ky.v\Theta.(Kx + Ky + Kz) \\ Kz.v\Theta.(Kx + Ky + Kz) \end{bmatrix}$

$therfore\ @\ Q = [1\ 1\ 1], R = \begin{bmatrix} C(\Theta) \\ C(\Theta) \\ C(\Theta) \end{bmatrix} + \begin{bmatrix} S(\Theta)(Ky - Kz) \\ S(\Theta)(Kx - Kz) \\ S(\Theta)(Kx - Ky) \end{bmatrix} +$

$\begin{bmatrix} Kx.v\Theta.(Kx + Ky + Kz) \\ Ky.v\Theta.(Kx + Ky + Kz) \\ Kz.v\Theta.(Kx + Ky + Kz) \end{bmatrix} = \begin{bmatrix} C(\Theta) + S(\Theta)(Ky - Kz) \ + Kx.v\Theta.(Kx + Ky + Kz) \\ C(\Theta) + S(\Theta)(Kx - Kz) + Ky.v\Theta.(Kx + Ky + Kz) \\ C(\Theta) + S(\Theta)(Kx - Ky) + Kz.v\Theta.(Kx + Ky + Kz) \end{bmatrix}$ → (1)

Multiply Rk (Θ) * Q

$$R_K(\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} Kx * Kx * v\Theta + C(\Theta) + Kx * Kyv\Theta - KzS(\Theta) + Kx * Kz * v\Theta + KyS(\Theta) \\ Kx * Ky * v\Theta + KzS(\Theta) + Ky * Ky * v\Theta + C(\Theta) + Ky * Kzv\Theta - KxS(\Theta) \\ Kx * Kz * v\Theta - KyS(\Theta) + Ky * Kzv\Theta + KxS(\Theta) + Kz * Kz * v\Theta + C(\Theta) \end{bmatrix}$$

If we rearrange this matrix we will get

$$\begin{bmatrix} C(\Theta) + Ky - Kz \ + Kx.v\Theta.(Kx + Ky + Kz) \\ C(\Theta) + Kx - Kz + Ky.v\Theta.(Kx + Ky + Kz) \\ C(\Theta) + Kx - Ky + Kz.v\Theta.(Kx + Ky + Kz) \end{bmatrix} \text{-> (2)}$$

<u>From (1) And (2)</u>

<u>The Formula is Proved</u>

2.21 [15] For rotations sufficiently small that the approximations $\sin\theta = \theta$, $\cos\theta = 1$, and $\theta^2 = 0$ hold, derive the rotation-matrix equivalent to a rotation of θ about a

$$R_K(\theta) = \begin{bmatrix} k_x k_x v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y k_y v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z k_z v\theta + c\theta \end{bmatrix}, \qquad (2.80)$$

where $c\theta = \cos\theta$, $s\theta = \sin\theta$, $v\theta = 1 - \cos\theta$, and ${}^A\hat{K} = [k_x k_y k_z]^T$. The sign of θ is determined by the right-hand rule, with the thumb pointing along the positive sense of ${}^A\hat{K}$.

<u>ANS</u>

- We have C(Θ) =1, therefore $v\,\Theta = 0$

$$\boldsymbol{Rk}\ (\Theta) = \begin{bmatrix} 1 & -\,Kz.\Theta & Ky.\Theta \\ Kz.\Theta & 1 & -\,Kx.\Theta \\ Ky.\Theta & Kx.\Theta & 1 \end{bmatrix}$$

2.22 [20] Using the result from Exercise 2.21, show that two infinitesimal rotations commute (i.e., the order in which the rotations are performed is not important).

- Assume that R1 (Φ) where Φ >>>>1, and R2 (Θ), Θ >>>>1

$$\boldsymbol{R1}(\Phi) * \boldsymbol{R2}(\Theta) = \begin{bmatrix} 1 & -\,Kz.\Phi & Ky.\Phi \\ Kz.\Phi & 1 & -\,Kx.\Phi \\ Ky.\Phi & Kx.\Phi & 1 \end{bmatrix} * \begin{bmatrix} 1 & -\,Kz.\Theta & Ky.\Theta \\ Kz.\Theta & 1 & -\,Kx.\Theta \\ Ky.\Theta & Kx.\Theta & 1 \end{bmatrix}$$

$$\boldsymbol{R1}(\Phi) * \boldsymbol{R2}(\Theta) \\ = \begin{bmatrix} 1 + (-Kz.\Phi * Kz.\Theta) + (Ky.\Phi * Ky.\Theta) & -\,Kz.\Theta - Kz.\Phi + (\,Ky.\Phi * Kx.\Theta) & Ky.\Theta + (-Kz.\Phi * Kx.\Theta) + Ky.\Phi \\ Kz.\Phi + \ Kz.\Theta + (-Kx.\Phi * Ky.\Theta) & (-Kz.\Phi * Kz.\Theta\,) + 1 + (-Kx.\Phi * Kx.\Theta) & (Kz.\Phi * Ky.\Theta) - Kx.\Phi - Kx.\Phi \\ Ky.\Phi + (Kx.\Phi * Kz.\Theta) + (Ky.\Theta) & (Ky.\Phi * -Kz.\Theta) + Kx.\Phi + Kx.\Theta & (Ky.\Phi * Ky.\Theta) + (Kx.\Phi * -Kx.\Theta) + 1 \end{bmatrix}$$

where Φ >>>>1, and R2 (Θ), Θ >>>>1

therefore, Φ * Θ = 0

$$\boldsymbol{R1}(\Phi) * \boldsymbol{R2}(\Theta) = \begin{bmatrix} 1 & -\,Kz.\Theta - Kz.\Phi & Ky.\Theta + Ky.\Phi \\ Kz.\Phi + \ Kz.\Theta & 1 & -\,Kx.\Phi - Kx.\Phi \\ Ky.\Phi(Ky.\Theta) & Kx.\Phi + Kx.\Theta & \end{bmatrix}$$

We have $\boldsymbol{R1}(\Phi) * \boldsymbol{R2}(\Theta)$ symmetric matrix

So $\boldsymbol{R1}(\Phi) * \boldsymbol{R2}(\Theta) = \ \boldsymbol{R2}(\Theta) * \ \boldsymbol{R1}(\Phi)$

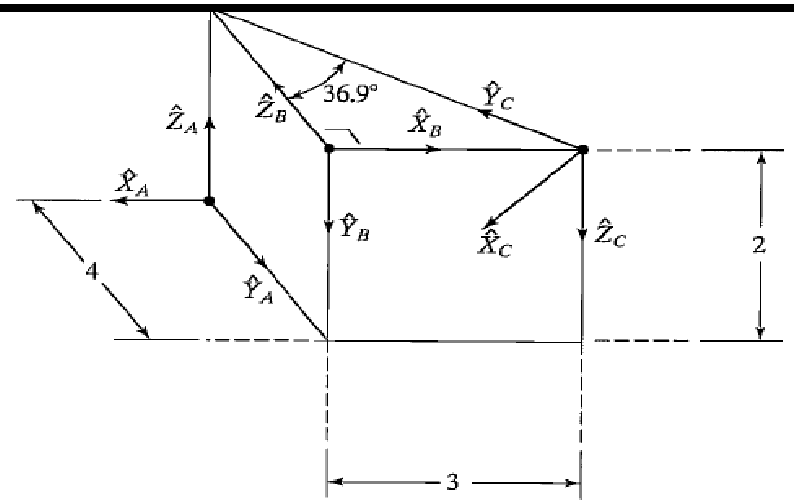


FIGURE 2.26: Frames at the corners of a wedge.

2.32 [15] Referring to Fig. 2.26, give the value of ${}^{A}_{C}T$.
2.33 [15] Referring to Fig. 2.26, give the value of ${}^{B}_{C}T$.
2.34 [15] Referring to Fig. 2.26, give the value of ${}^{C}_{A}T$.

ANS

((((((Assume A is the Reference Frame))))))

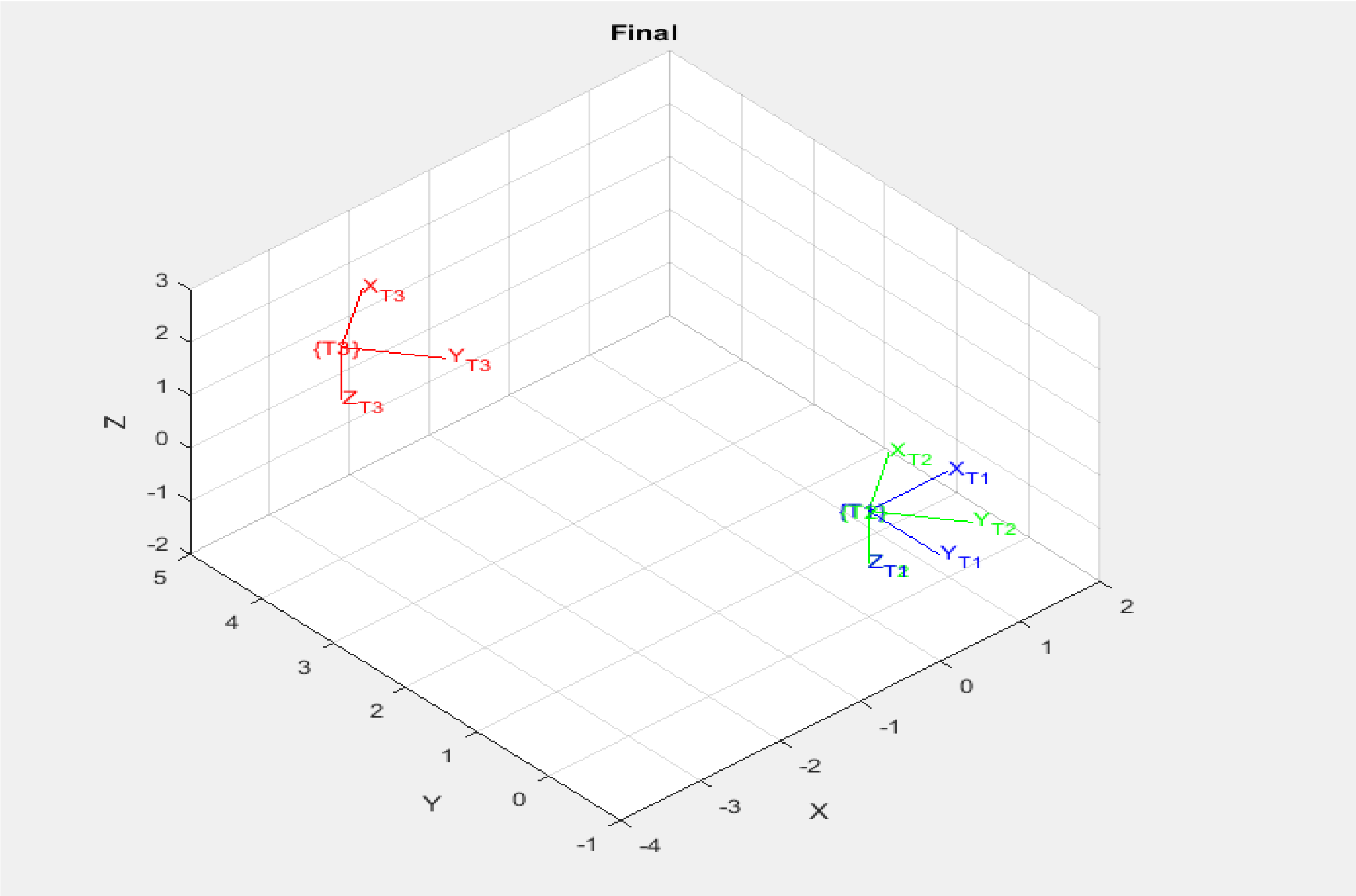
The sequence is: Trans (x=-3, y =4, z=2) * ROT (z= 36.9) * ROT (x=180) = T(new)

$$T = \begin{bmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0.79 & -0.6 & 0 & 0 \\ 0.6 & 0.799 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.79 & 0.6 & 0 & -3 \\ 0.6 & -0.799 & 0 & 4 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\carig_2_32.m

2_13.m | Project1.m | NIKU_2_11.m | carig_2_32.m | NIKU_2_27.m | +

```
ROTX_180 = [1 0 0 0 ; 0 -1 0 0 ; 0 0 -1 0 ; 0 0 0 1];
ROTZ_37 = [0.799 -0.6 0 0 ; 0.6 0.799 0 0 ; 0 0 1 0 ; 0 0 0 1];
trans = [1 0 0 -3 ; 0 1 0 4 ; 0 0 1 2 ; 0 0 0 1 ];
F1 = ROTX_180;
F2 = ROTZ_37 * F1;
Final = trans * F2;
x=[-4 2 -1 5 -2 3];
%subplot(1,3,1);
trplot(F1,'frame','T1','color', 'b','axis', x);
title('F1 = ROTX180')
hold on
%subplot(1,3,2);
trplot(F2,'frame','T2','color', 'g','axis', x);
title('F2 = ROTZ37 * F1')
%subplot(1,3,3);
trplot(Final,'frame','T3','color', 'r','axis', x);
title('Final')
```



2.43

CTA = INV (ATC) , and we calculate ATC in 2.32.

$$T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad T^{-1} = \begin{bmatrix} n_x & n_y & n_z & -\mathbf{p}\cdot\mathbf{n} \\ o_x & o_y & o_z & -\mathbf{p}\cdot\mathbf{o} \\ a_x & a_y & a_z & -\mathbf{p}\cdot\mathbf{a} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$therfore\ \ \mathbf{ATC}^{\,-1} = \begin{bmatrix} 0.79 & 0.60 & 0 & -((-3*0.79)+(4*0.6)+(2*0)) \\ 0.6 & -0.799 & 0 & -((-3*0.6)+(4*-0.799)+(2*0)) \\ 0 & 0 & -1 & -((-3*0)+(4*0)+(2*-1)) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{ATC}^{\,-1} = \ \mathbf{CTA} = \begin{bmatrix} 0.79 & 0.60 & 0 & -0.03 \\ 0.6 & -0.799 & 0 & 4.996 \\ 0 & 0 & -1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

MATLAB EXERCISE 2B

a) Write a MATLAB program to calculate the homogeneous transformation matrix ${}^{A}_{B}T$ when the user enters $Z\!-\!Y\!-\!X$ Euler angles $\alpha-\beta-\gamma$ and the position vector ${}^{A}P_B$. Test for two examples:

i) $\alpha = 10^{\circ}$, $\beta = 20^{\circ}$, $\gamma = 30^{\circ}$, and ${}^{A}P_B = \{1\ \ 2\ \ 3\ \}^T$.

ii) For $\beta = 20^{\circ}$ $(\alpha = \gamma = 0^{\circ})$, ${}^{A}P_B = \{3\ \ 0\ \ 1\ \}^T$.

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\craig_2B_1_rpy2tr.m

craige_2_B.m | craig_2B_1_rpy2tr.m | +

```matlab

function ROT = craig_2B_1_rpy2tr(alpha ,beta, gama , p )
ROT_Z = [cos(alpha*pi/180) -sin(alpha*pi/180) 0 0 ;
    sin(alpha*pi/180) cos(alpha*pi/180) 0 0 ; 0 0 1 0 ; 0 0 0 1];

ROT_Y = [cos(beta*pi/180) 0 sin(beta*pi/180) 0 ; 0 1 0 0 ;
    -sin(beta*pi/180) 0 cos(beta*pi/180) 0 ; 0 0 0 1 ];

ROT_X = [1 0 0 0 ; 0 cos(gama*pi/180) -sin(gama*pi/180) 0 ;
    0 sin(gama*pi/180) cos(gama*pi/180) 0; 0 0 0 1];

trans = [1 0 0 p(1); 0 1 0 p(2) ; 0 0 1 p(3) ; 0 0 0 1];
ROT =trans * ROT_Z * ROT_Y * ROT_X;
end

```

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\craige_2_B.m*

craige_2_B.m* | craig_2B_1_rpy2tr.m | +

```matlab
alpha1 = 10 ;beta1 = 20 ;gama1 = 30 ; Bp1 = [1 ; 2 ; 3 ; 1];
alpha2 = 0 ;beta2 = 20 ;gama2 = 0 ; Bp2 = [3 ; 0 ; 1 ; 1];
%a)calculate the homogeneous transformation matrix
Homo_trans_1 = craig_2B_1_rpy2tr(alpha1 , beta1 ,gama1, Bp1);
Homo_trans_2 = craig_2B_1_rpy2tr(alpha2 , beta2 ,gama2, Bp2);

```

Command Window

```
>> craige_2_B

Homo_trans_1 =

    0.9254    0.0180    0.3785    1.0000
    0.1632    0.8826   -0.4410    2.0000
   -0.3420    0.4698    0.8138    3.0000
         0         0         0    1.0000


Homo_trans_2 =

    0.9397         0    0.3420    3.0000
         0    1.0000         0         0
   -0.3420         0    0.9397    1.0000
         0         0         0    1.0000
```

b) For $\beta = 20^\circ$ $(\alpha = \gamma = 0^\circ)$, ${}^{A}P_{B} = \{3 \;\; 0 \;\; 1\}^{T}$, and ${}^{B}P = \{1 \;\; 0 \;\; 1\}^{T}$, use MATLAB to calculate ${}^{A}P$; demonstrate with a sketch that your results are correct. Also, using the same numbers, demonstrate all three interpretations of the homogeneous transformation matrix—the (b) assignment is the second interpretation, transform mapping.

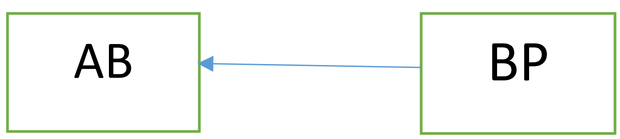
Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\craige_2_B.m

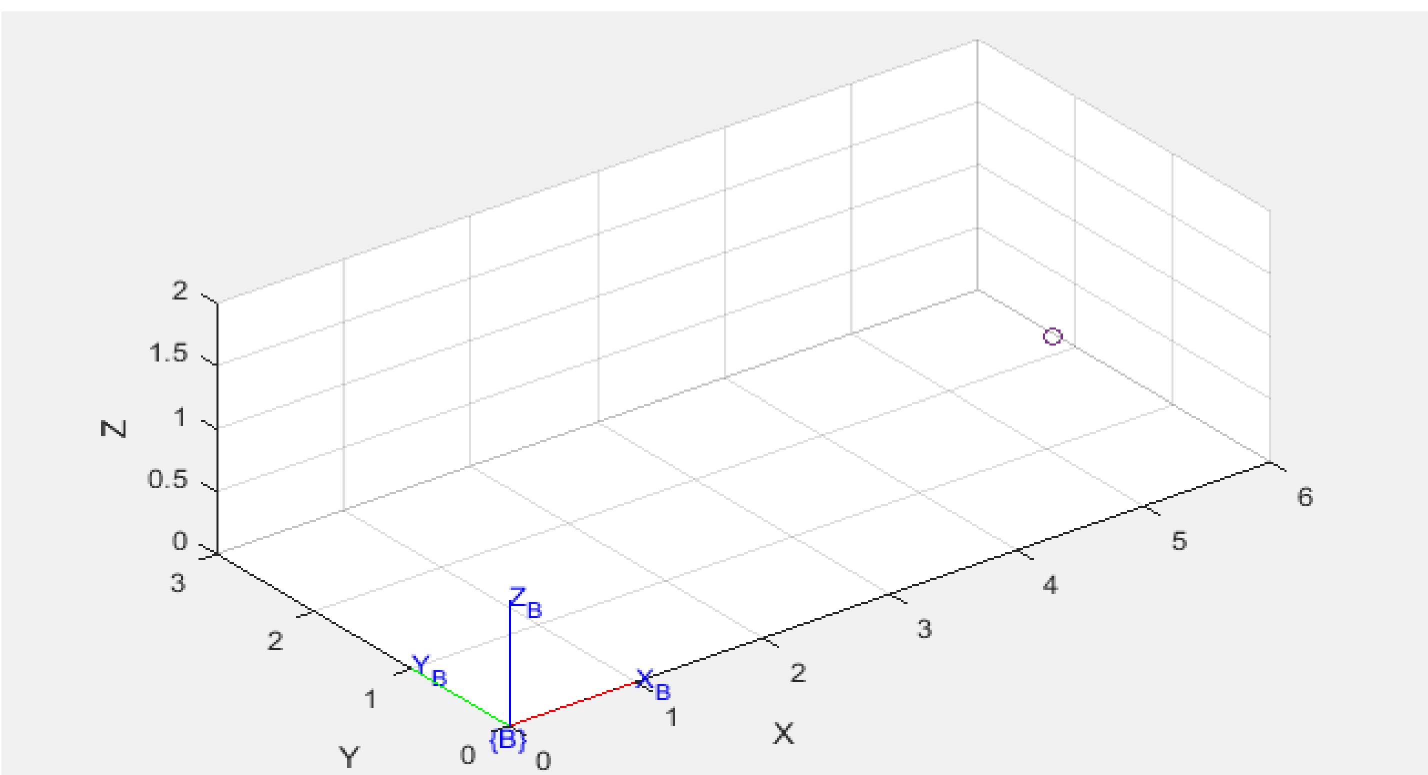
craige_2_B.m | craig_2B_1_rpy2tr.m

```
alpha1 = 10 ;beta1 = 20 ;gama1 = 30 ; Bp1 = [1 ; 2 ; 3 ; 1];
alpha2 = 0 ;beta2 = 20 ;gama2 = 0 ; Bp2 = [3 ; 0 ; 1 ; 1];
%a)calculate the homogeneous transformation matrix
Homo_trans_1 = craig_2B_1_rpy2tr(alpha1 , beta1 ,gama1, Bp1);
Homo_trans_2 = craig_2B_1_rpy2tr(alpha2 , beta2 ,gama2, Bp2);
%b)For second input calculate AP
% Ap = AB * Bp
frame= [1 0 0 0; 0 1 0 0 ; 0 0 1 0 ; 0 0 0 1];
Bp = [1 ; 0 ; 1 ; 1];
Ap = Homo_trans_2 * Bp;
x=[0 6 0 3 0 2];
trplot(frame,'frame','B','rgb' ,'axis', x );
hold on
plot3( Ap(1),Ap(2),Ap(3), 'o');
```

Command Window

```
>> craige_2_B

Ap =

    4.2817
         0
    1.5977
    1.0000

>>
```





c) Write a MATLAB program to calculate the inverse homogeneous transformation matrix ${}^{A}_{B}T^{-1} = {}^{B}_{A}T$, using the symbolic formula. Compare your result with a numerical MATLAB function (e.g., *inv*). Demonstrate that both methods yield correct results (i.e., ${}^{A}_{B}T\ {}^{A}_{B}T^{-1} = {}^{A}_{B}T^{-1}\ {}^{A}_{B}T = I_4$). Demonstrate this for examples (i) and (ii) from (a) above.

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\craig_2B_C_Inverse.m

craige_2_B.m | craig_2B_1_rpy2tr.m | craig_2B_C_Inverse.m

```matlab
function Inverse = craig_2B_C_Inverse(Homo_R )
Inverse (1:3 , 1:3) = transpose (Homo_R(1:3 , 1:3) );
Inverse(4 ,1:3 )=0;
Inverse (4 ,4) = 1;
for i = 1:3
    Inverse(i,4) = -dot (Homo_R(1:3 , 4) ,Homo_R(1:3 , i));
end

end
```

D:\matlab setup\bin\carig2B.m

2B.m | craig_2B_C.m | Carige_2B_1.m

```matlab
alpha = 0; beta = 20; gama = 0; Bp = [1 ; 0 ; 1 ; 1];
ApB_rot = Carige_2B_1(alpha ,beta, gama , ApB_point);
Calculated_Inverse = craig_2B_C(ApB_rot );
INV = inv (ApB_rot);
Test1 = Calculated_Inverse * ApB_rot;
test2 =  ApB_rot * INV;
```

```
Calculated_Inverse =

    0.9397         0   -0.3420   -2.4771
         0    1.0000         0         0
    0.3420         0    0.9397   -1.9658
         0         0         0    1.0000

INV =

    0.9397         0   -0.3420   -2.4771
         0    1.0000         0         0
    0.3420         0    0.9397   -1.9658
         0         0         0    1.0000

Test1 =

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

test2 =

    1.0000         0    0.0000         0
         0    1.0000         0         0
         0         0    1.0000    0.0000
         0         0         0    1.0000
```

D:\matlab setup\bin\carig2B.m

2B.m | craig_2B_C.m | Carige_2B_1.m

```matlab
alpha = 10; beta = 20; gama = 30; Bp = [1 ; 2 ; 3 ; 1];
ApB_rot = Carige_2B_1(alpha ,beta, gama , ApB_point);
Calculated_Inverse = craig_2B_C(ApB_rot );
INV = inv (ApB_rot);
Test1 = Calculated_Inverse * ApB_rot;
test2 =  ApB_rot * INV;
```

```
Calculated_Inverse =

    0.9254    0.1632   -0.3420   -2.4342
    0.0180    0.8826    0.4698   -0.5239
    0.3785   -0.4410    0.8138   -1.9494
         0         0         0    1.0000

INV =

    0.9254    0.1632   -0.3420   -2.4342
    0.0180    0.8826    0.4698   -0.5239
    0.3785   -0.4410    0.8138   -1.9494
         0         0         0    1.0000

Test1 =

    1.0000   -0.0000         0         0
   -0.0000    1.0000    0.0000         0
         0    0.0000    1.0000         0
         0         0         0    1.0000

test2 =

    1.0000         0    0.0000         0
   -0.0000    1.0000         0         0
    0.0000    0.0000    1.0000         0
         0         0         0    1.0000
```

d) Define ${}^{A}_{B}T$ to be the result from (a)(i) and ${}^{B}_{C}T$ to be the result from (a)(ii).

i) Calculate ${}^{A}_{C}T$, and show the relationship via a transform graph. Do the same for ${}^{C}_{A}T$.

ii) Given ${}^{A}_{C}T$ and ${}^{B}_{C}T$ from (d)(i)—assume you don't know ${}^{A}_{B}T$, calculate it, and compare your result with the answer you know.

iii) Given ${}^{A}_{C}T$ and ${}^{A}_{B}T$ from (d)(i)—assume you don't know ${}^{B}_{C}T$, calculate it, and compare your result with the answer you know.

(i) <u>ATC = ATB * BTC and CTA = inv (ATC)</u>

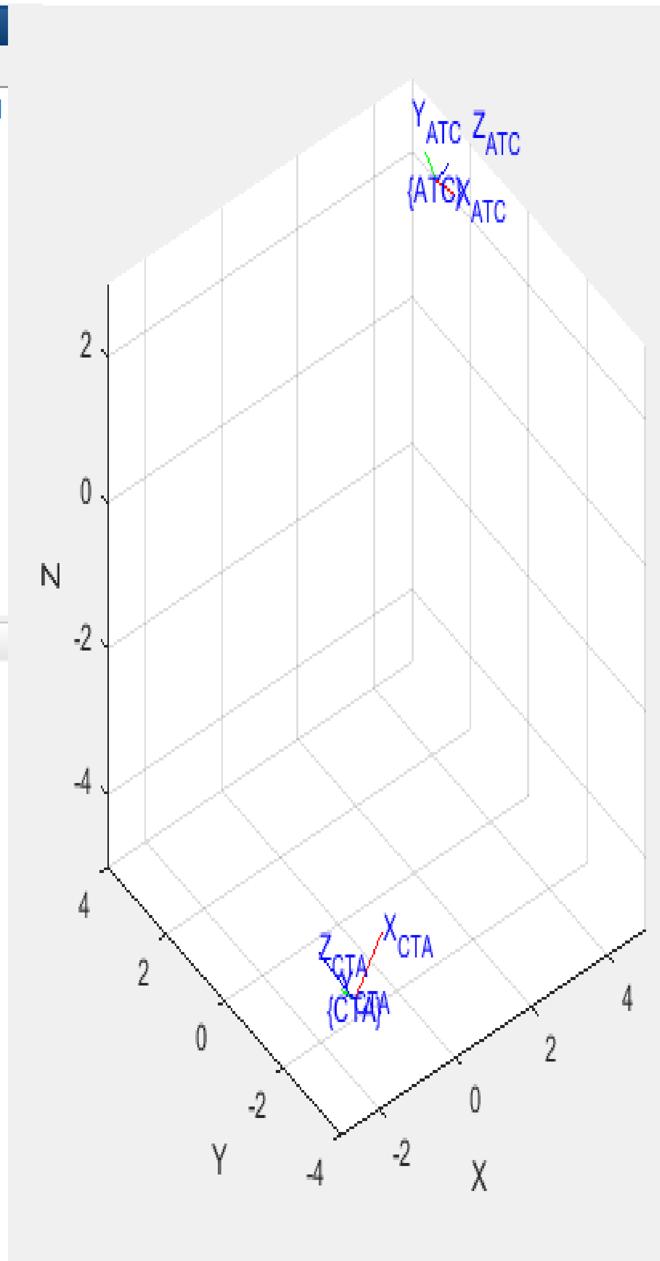
r - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\carig2B_Di.m

g2B_Di.m | Carige_2B_1.m | +

```
alpha1 = 10; beta1 = 20; gama1 = 30; P1 = [1 ; 2 ; 3 ; 1]
ATB = Carige_2B_1(alpha1 ,beta1, gama1 , P1);
alpha2 = 0; beta2 = 20; gama2 = 0; P2 = [3 ; 0 ; 1 ; 1];
BTC = Carige_2B_1(alpha2 ,beta2, gama2 , P2);
ATC = ATB * BTC
% CTA = inverse (ATC)
CTA = craig_2B_C(ATC )
x=[-3 5 -4 4 -5 3];
trplot(ATC,'frame','ATC','rgb' ,'axis', x );
hold on
trplot(CTA,'frame','CTA','rgb' ,'axis', x );
```

Command Window

```
ATC =

    0.7401    0.0180    0.6722    4.1548
    0.3042    0.8826   -0.3586    2.0486
   -0.5997    0.4698    0.6477    2.7877
         0         0         0    1.0000


CTA =

    0.7401    0.3042   -0.5997   -2.0263
    0.0180    0.8826    0.4698   -3.1927
    0.6722   -0.3586    0.6477   -3.8641
         0         0         0    1.0000

>>
```



(ii) ATB = ATC * inv (BTC)

Note : ATC and BTC are stored variables from D(i)

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\craig2B_DII.m*

carig2B_Di.m | Carige_2B_1.m | Craig_2B_DIII.m | craig2B_DII.m*

```matlab
%(ii)   ATB_new = ATC * inv (BTC)
ATB_new = ATC * craig_2B_C(BTC);
% COmpare to Known ATB
alpha1 = 10; beta1 = 20; gama1 = 30; P1 = [1 ; 2 ; 3 ; 1];
ATB = Carige_2B_1(alpha1 ,beta1, gama1 , P1);
```

Command Window

```
ATB_new =

    0.9254    0.0180    0.3785    1.0000
    0.1632    0.8826   -0.4410    2.0000
   -0.3420    0.4698    0.8138    3.0000
         0         0         0    1.0000


ATB =

    0.9254    0.0180    0.3785    1.0000
    0.1632    0.8826   -0.4410    2.0000
   -0.3420    0.4698    0.8138    3.0000
         0         0         0    1.0000
```

The results are equal.

(iii) BTC = inv (ATB) * ATC

Note : ATC and BTC are stored variables from D(i)

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\Craig_2B_DIII.m

carig2B_Di.m | Craig_2B_DIII.m | craig2B_DII.m

```matlab
%(iii)  BTC_new = inv (ATB) * ATC
BTC_new =  craig_2B_C(ATB) * ATC ;
%Compare to BTC
alpha2 = 0; beta2 = 20; gama2 = 0; P2 = [3 ; 0 ; 1 ; 1];
BTC = Carige_2B_1(alpha2 ,beta2, gama2 , P2);
```

Command Window

```
>> carig2B_Di
>> Craig_2B_DIII

BTC_new =

    0.9397   -0.0000    0.3420    3.0000
   -0.0000    1.0000    0.0000         0
   -0.3420    0.0000    0.9397    1.0000
         0         0         0    1.0000


BTC =

    0.9397         0    0.3420    3.0000
         0    1.0000         0         0
   -0.3420         0    0.9397    1.0000
         0         0         0    1.0000
```

The results are equal.

e) Check all results by means of the Corke MATLAB Robotics Toolbox. Try functions *rpy2tr*() and *transl*().

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\criag_2B_E.m

criag_2B_E.m

```
alpha1 = 10; beta1 = 20; gama1 = 30; P1 = [1 ; 2 ; 3 ; 1];
ATB = Carige_2B_1(alpha1 ,beta1, gama1 , P1);
ATB_Corke_toll= rpy2tr (30 *pi/180 , 20*pi/180 , 10*pi/180);
alpha2 = 0; beta2 = 20; gama2 = 0; P2 = [3 ; 0 ; 1 ; 1];
BTC = Carige_2B_1(alpha2 ,beta2, gama2 , P2);
BTC_Corke_toll= rpy2tr (0*pi/180 , 20*pi/180 , 0*pi/180 ) ;
```

```
>> criag_2B_E

ATB =

    0.9254    0.0180    0.3785    1.0000
    0.1632    0.8826   -0.4410    2.0000
   -0.3420    0.4698    0.8138    3.0000
         0         0         0    1.0000


ATB_Corke_toll =

    0.9254   -0.1632    0.3420         0
    0.3188    0.8232   -0.4698         0
   -0.2049    0.5438    0.8138         0
         0         0         0    1.0000


BTC =

    0.9397         0    0.3420    3.0000
         0    1.0000         0         0
   -0.3420         0    0.9397    1.0000
         0         0         0    1.0000
```

The results are equal.

```
BTC_Corke_toll =

    0.9397         0    0.3420         0
         0    1.0000         0         0
   -0.3420         0    0.9397         0
         0         0         0    1.0000

>>
```

From [Niku 2010], resolve the following problems using quaternions and visualize your steps using the Robotics Toolbox of [Corke 2011]:

2.11. Find the coordinates of point $P(2,3,4)^T$ relative to the reference frame after a rotation of $45°$ about the x-axis.

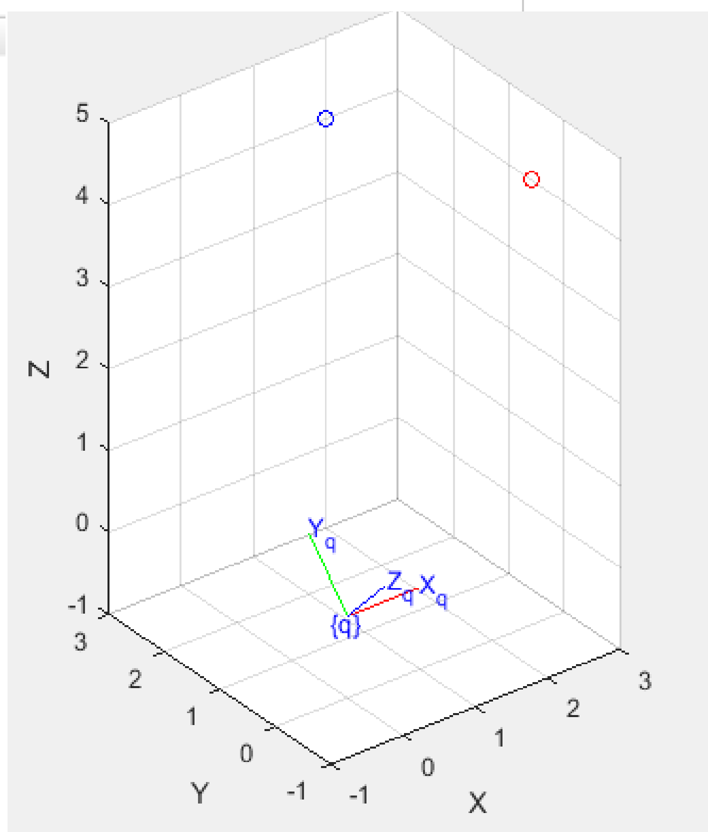
Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_11_QUAT.m

NIKU_2_11_QUAT.m | NIKU_2_11.m

```
P = [2 ; 3 ; 4 ];
theta = 45/2;
q = Quaternion([cos(theta*pi/180 ) sin(theta*pi/180) 0 0]);
P_Rotated= q * P;
x=[-1 3 -1 3 -1 5];
q.plot('frame', 'F' , 'rgb' ,'axis', x)
hold on
%blue point is the original point
plot3( P(1),P(2),P(3), 'o' , 'color', 'b')
%red point is the rotated point
plot3( P_Rotated(1),P_Rotated(2),P_Rotated(3), 'o' ,'color', 'r')
```

Command Window

```
>> NIKU_2_11_QUAT

q =

0.92388 < 0.38268, 0, 0 >


P_Rotated =

    2.0000
   -0.7071
    4.9497
```



2.14. A point P in space is defined as ${}^{B}P = (5, 3, 4)^T$ relative to frame B which is attached to the origin of the reference frame A and is parallel to it. Apply the following transformations to frame B and find ${}^{A}P$. Using the 3-D grid, plot the transformations and the result and verify it. Also verify graphically that you would not get the same results if you apply the transformations relative to the current frame:

- Rotate 90° about x-axis; then
- Translate 3 units about y-axis, 6 units about z-axis, and 5 units about x-axis; then,
- Rotate 90° about z-axis.

Editor - D:\4th yaer\2nd term\robotics\matlab\robot-9.10\rvctools\NIKU_2_14_QUAT.m

NIKU_2_14_QUAT.m

```
P = [5 ; 3 ; 4];
theta = 90/2 ;
q_x = Quaternion([cos(theta*pi/180 ) sin(theta*pi/180) 0 0]);
Trans_point = [5 ; 3 ; 6];
q_y = Quaternion([cos(theta*pi/180 ) 0 0 sin(theta*pi/180)]);
p1= q_x * P;
p2= p1 + Trans_point;
F2= q_x + Quaternion([0 5 3 6]);
p3= q_y * p2;
F3= q_y * F2;
x=[-1 6 -5 1 -1 4];
subplot(1,3,1); q_x.plot('frame','F1','rgb' ,'axis', x );
hold on
plot3( p1(1),p1(2),p1(3), 'o');title('X_ROT')

x1=[4 10 -2 4 5 10];
subplot(1,3,2); F2.plot('frame','F2','rgb' ,'axis', x1 );
hold on
plot3( p2(1),p2(2),p2(3), 'o');title('Trans')

x2=[-1 10 -2 11 2 10];
subplot(1,3,3); F3.plot('frame','F3','rgb' ,'axis', x2 );
hold on
plot3( p3(1),p3(2),p3(3), 'o');title('Z_ROT')
```

```
p3 =

    1.0000
   10.0000
    9.0000
```