21AIE201 - Introduction to Robotics

Assignment 1 PART 2

Group 14 Batch B

Kalyana Sundaram	CB.EN.U4AIE21120
Kaushik Jonnada	CB.EN.U4AIE21122
Praneetha K	CB.EN.U4AIE21147
Sarvesh Shashikumar	CB.EN.U4AIE21163
Subikksha	CB.EN.U4AIE21167

1. Find the rotation matrix representation of ZYX (RAR) from the given angles α, β and γ about Z, Y and X axis respectively? Also find the expressions for inverse case, i.e., angles given a rotation matrix.

Since the given rotation is RAR, we do the regular multiplication i.e without inverting the order of multiplication

Therefore the rotation matrix will be $R = R_Z * R_Y * R_X$

$$R_Z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R_Y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \qquad R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$$R = R_Z * R_Y * R_X$$

$$R(\alpha, \beta, \gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

we need to find the angles (α, β, ν) when rotation matrix is given

Let the rotation matrix be

$$R = \begin{bmatrix} 0.6123 & -0.0474 & 0.8623 \\ 0.6123 & 0.5561 & -0.4356 \\ -0.5 & 0.75 & 0.443 \end{bmatrix}$$

On comparing this with the general rotation matrix for ZYX(RAR), we get

$$\begin{bmatrix} 0.6123 & -0.0474 & 0.8623 \\ 0.6123 & 0.5561 & -0.4356 \\ -0.5 & 0.75 & 0.443 \end{bmatrix} = \begin{bmatrix} cos\alpha cos\beta & cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma & cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma \\ sin\alpha cos\beta & sin\alpha sin\beta sin\gamma + cos\alpha cos\gamma & sin\alpha sin\beta cos\gamma - cos\alpha sin\gamma \\ -sin\beta & cos\beta sin\gamma & cos\beta cos\gamma \end{bmatrix}$$

$$-sin\beta = 0.5$$

$$\boldsymbol{\beta = \sin^{-1}(0.5) = 30°}$$

$$sin\gamma cos\beta = 0.75$$

$$\frac{\sqrt{3}sin\gamma}{2} = 0.75$$

$$sin\gamma = 0.75(\frac{2}{\sqrt{3}}) = 0.866$$

$$\boldsymbol{\gamma = sin^{-1}(0.866) = 60°}$$

$$cos\alpha cos\beta = 0.6123$$

$$cos\alpha = 0.6123(\frac{2}{\sqrt{3}})$$

$$\boldsymbol{\alpha = cos^{-1}(0.7070) = 45°}$$

Code:

```
import numpy as np
import spatialmath.base.symbolic as sym
from numpy import linalg as ln
from spatialmath import *
from spatialmath.base import *
from sympy import *
```

```
T=Symbol('θ') # taking symbol theta and assigning to variable T
r=Matrix(rot2(T)).eigenvals() # eigenvalues for the rotation matrix with angle theta rotation assigned in variable
print(tuple(r)) #print r in tuple

r=rot2(45,'deg') # assigned theta value as 45degrees
e,v returns the eigenvalues
e,v= np.linalg.eig(r) # eigenvalues for the rotation matrix with angle theta rotation assigned in variable e
print('θ = 45 degrees')#print theta =45 degrees
print(tuple(e))#print eigen values in tuple
```

[29]

```
(-sqrt((cos(θ) - 1)*(cos(θ) + 1)) + cos(θ), sqrt((cos(θ) - 1)*(cos(θ) + 1)) + cos(θ))
θ = 45 degrees
((0.7071067811865476+0.7071067811865475j), (0.7071067811865476-0.7071067811865475j))
```

```
T=Symbol('θ')# taking symbol theta and assigning to variable T
r=(Matrix(rotx(T)).eigenvals())# eigenvalues for the 3Drotation matrix about x
print(tuple(r))#print r in tuple

R=rotx(30,'deg') # assigned theta value as 30degrees
e,v= np.linalg.eig(R) # this function return values of eigenvalues
print('θ = 30 degrees')#print theta =45 degrees
print(list(e))#print eigen values in tuple
```

```
(1, -sqrt((cos(θ) - 1)*(cos(θ) + 1)) + cos(θ), sqrt((cos(θ) - 1)*(cos(θ) + 1)) + cos(θ))
θ = 30 degrees
[(0.8660254037844387+0.4999999999999999j), (0.8660254037844387-0.4999999999999999j), (1+0j)]
```

2. Find the rotation matrix at $\beta = -90$ degrees for XYZ (FAR).

Since the given rotation is FAR, we do the pre-multiplication i.e inverting the order of multiplication

Therefore the rotation matrix will be $R = R_Z * R_Y * R_X$

$$R_Z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_Y = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

$R = R_Z * R_Y * R_X$

$$\mathrm{R}(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}$$

Substituting β=-90, we get

$$\mathrm{R}(\alpha,\beta,\gamma) = \begin{bmatrix} \cos\alpha\cos(-90) & \cos\alpha\sin(-90)\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin(-90)\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos(-90) & \sin\alpha\sin(-90)\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin(-90)\cos\gamma - \cos\alpha\sin\gamma \\ -\sin(-90) & \cos(-90)\sin\gamma & \cos(-90)\cos\gamma \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -s_\gamma c_\alpha - s_\alpha c_\gamma & -c_\alpha c_\gamma + s_\alpha s_\gamma \\ 0 & -s_\alpha s_\gamma + c_\alpha c_\gamma & -s_\alpha c_\gamma - c_\alpha s_\gamma \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\sin(\alpha+\gamma) & -\cos(\alpha+\gamma) \\ 0 & \cos(\alpha+\gamma) & -\sin(\alpha+\gamma) \\ 1 & 0 & 0 \end{bmatrix}$$

```python
a = sym.symbol('α')# assigning aplha symbol to variable a

b = sym.symbol('β') #assigning betha symbol to variable b
c = sym.symbol('γ')#assigning gamma symbol to variable c
rpy_matrix = np.matrix(rpy2r(a,b,c)) #2D array of row-pitch-yaw matrix with apl
print(rpy_matrix) #print 2D array of matrix rpy_matri
```

```
[[cos(β)*cos(γ) sin(β)*sin(α)*cos(γ) - sin(γ)*cos(α)
  sin(β)*cos(α)*cos(γ) + sin(α)*sin(γ)]
 [sin(γ)*cos(β) sin(β)*sin(α)*sin(γ) + cos(α)*cos(γ)
  sin(β)*sin(γ)*cos(α) - sin(α)*cos(γ)]
 [-sin(β) sin(α)*cos(β) cos(β)*cos(α)]]
```

```python
a = sym.symbol('α')# assigning aplha symbol to variable a

b = sym.symbol('β') # assigning betha symbol to variable b
c = sym.symbol('γ')# assigning gamma symbol to variable c

rpy_matrix = np.matrix(rpy2r(a,-90,c,unit = 'deg'))#2D array
print(rpy_matrix)#print 2D array of matrix rpy_matrix
```

```
[[6.12323399573677e-17*cos(0.0174532925199433*γ)
  -1.0*sin(0.0174532925199433*α)*cos(0.0174532925199433*γ) - 1.0*sin(0.0174532925199433*γ)*cos(0.0174532925199433*α)
  1.0*sin(0.0174532925199433*α)*sin(0.0174532925199433*γ) - 1.0*cos(0.0174532925199433*α)*cos(0.0174532925199433*γ)]
 [6.12323399573677e-17*sin(0.0174532925199433*γ)
  -1.0*sin(0.0174532925199433*α)*sin(0.0174532925199433*γ) + 1.0*cos(0.0174532925199433*α)*cos(0.0174532925199433*γ)
  -1.0*sin(0.0174532925199433*α)*cos(0.0174532925199433*γ) - 1.0*sin(0.0174532925199433*γ)*cos(0.0174532925199433*α)]
 [1.0 6.12323399573677e-17*sin(0.0174532925199433*α)
  6.12323399573677e-17*cos(0.0174532925199433*α)]]
```

3. Find the singularities (both angles and Rotation matrix at the singularities) for ZYX (RAR) and ZYZ (RAR)? Compare the solutions with XYZ (FAR) taught in the class. Find the solution for all 3 sequences at their singularities if alpha is zero degrees.

ZYZ (Rotated angle rotation)

$$R(\alpha,\beta,\alpha) = R_z(\alpha) \times R_y(\beta) \times R_z(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- *In Euler angles, when the middle orientation is 0, we have a gimble lock or approaches singularity. For ZYZ Euler angle , when the orientation around Y - axis is 0, the first rotation around Z and the last rotation and Z collides.*

$$R(\,at\ y=0) \Rightarrow \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(0) & 0 & \sin(0) \\ 0 & 1 & 0 \\ -\sin(0) & 0 & \cos(0) \end{bmatrix} \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{cos^2\alpha - sin^2\alpha} & \boldsymbol{-2sin\alpha cos\alpha} & \boldsymbol{0} \\ \boldsymbol{2cos\alpha sin\alpha} & \boldsymbol{cos^2\alpha - sin^2\alpha} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}$$

Rotation matrix at singularities

$$(\alpha = 0) => \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ZYX (Rotated angle rotation)

$$R(\alpha,\beta,\nu) = R_z(\alpha) \times R_y(\beta) \times R_x(\nu) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

- *In Cardan angles , singularity happens when, in the global frame, the second rotation makes the last rotational axis aligns with the first rotational axis, this happens for second rotation = 90*

$$R = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(90) & 0 & \sin(90) \\ 0 & 1 & 0 \\ -\sin(90) & 0 & \cos(90) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & -sin\alpha & cos\alpha \\ 0 & cos\alpha & sin\alpha \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & -sin\alpha cos\gamma + cos\alpha sin\gamma & sin\alpha sin\gamma + cos\alpha cos\gamma \\ 0 & cos\alpha cos\gamma + sin\alpha sin\gamma & -cos\alpha sin\gamma + sin\alpha cos\gamma \\ -1 & 0 & 0 \end{bmatrix}$$ *Rotation matrix at singularities*

$$\mathbf{R} = \begin{bmatrix} \mathbf{0} & \mathbf{-\sin(\alpha - \gamma)} & \mathbf{\cos(\alpha - \gamma)} \\ \mathbf{0} & \mathbf{\cos(\alpha - \gamma)} & \mathbf{\sin(\alpha - \gamma)} \\ \mathbf{-1} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$[\alpha = 0] \Rightarrow \begin{bmatrix} 0 & sin\gamma & cos\gamma \\ 0 & cos\gamma & -sin\gamma \\ -1 & 0 & 0 \end{bmatrix}$$

XYZ (Fixed angle rotation)

$$R(\alpha,\beta,\nu) = R_z(\alpha) \times R_y(\beta) \times R_x(\nu) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

In Cardan angles , singularity happens when, in the global frame, the second rotation makes the last rotational axis aligns with the first rotational axis, this happens for second rotation = 90

$$R = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(90) & 0 & \sin(90) \\ 0 & 1 & 0 \\ -\sin(90) & 0 & \cos(90) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$\mathrm{R} = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & -sin\alpha & cos\alpha \\ 0 & cos\alpha & sin\alpha \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix}$$

$$R = \begin{bmatrix} 0 & -sin\alpha cos\gamma + cos\alpha sin\gamma & sin\alpha sin\gamma + cos\alpha cos\gamma \\ 0 & cos\alpha cos\gamma + sin\alpha sin\gamma & -cos\alpha sin\gamma + sin\alpha cos\gamma \\ -1 & 0 & 0 \end{bmatrix}$$

Rotation matrix at singularities

$$R = \begin{bmatrix} 0 & -\sin(\alpha - \gamma) & \cos(\alpha - \gamma) \\ 0 & \cos(\alpha - \gamma) & \sin(\alpha - \gamma) \\ -1 & 0 & 0 \end{bmatrix}$$

$$\{\alpha = 0\} \Rightarrow \begin{bmatrix} 0 & sin\gamma & cos\gamma \\ 0 & cos\gamma & -sin\gamma \\ -1 & 0 & 0 \end{bmatrix}$$

```python
a = sym.symbol('α')

b = sym.symbol('β')
c = sym.symbol('γ')

rpy_zyx = SO3.RPY(a,np.pi/2,c)
print(rpy_zyx)
print()
rpy_zyz = SO3.Rz(a)*SO3.Ry(np.pi/2)*SO3.Rz(c)
print(rpy_zyz)
```

```
6.12323399573677e-17*cos(γ) 1.0*sin(α)*cos(γ) - 1.0*sin(γ)*cos(α) 1.0*sin(α)*sin(γ) + 1.0*cos(α)*cos(γ)
6.12323399573677e-17*sin(γ) 1.0*sin(α)*sin(γ) + 1.0*cos(α)*cos(γ) -1.0*sin(α)*cos(γ) + 1.0*sin(γ)*cos(α)
-1        6.12323399573677e-17*sin(α) 6.12323399573677e-17*cos(α)


-1.0*sin(α)*sin(γ) + 6.12323399573677e-17*cos(α)*cos(γ) -1.0*sin(α)*cos(γ) - 6.12323399573677e-17*sin(γ)*cos(α) 1.0*cos(α)
6.12323399573677e-17*sin(α)*cos(γ) + 1.0*sin(γ)*cos(α) -6.12323399573677e-17*sin(α)*sin(γ) + 1.0*cos(α)*cos(γ) 1.0*sin(α)
-1.0*cos(γ)  1.0*sin(γ)    0
```

4. What is a gimbal lock and how is it associated with the singularity of 3 angle representations?

A gimbal is a ring that is suspended so it can rotate about an axis. Gimbals are typically nested one within another to accommodate rotation about multiple axes.

GIMBAL LOCK : Gimbal lock is the loss of one degree of freedom in a three-dimensional, three-gimbal mechanism that occurs when the axes of two of the three gimbals are driven into a parallel configuration, "locking" the system into rotation in a degenerate two-dimensional space.

The cause of gimbal lock is representing an orientation as three axial rotations with Euler angles. A potential solution therefore is to represent the orientation in some other way. This could be as a rotation matrix, a quaternion

5. Find the equality between the quaternions (a, -b, -c, -d) and (-a,b,c,d).

If two quaternions are q1 and q2 respectively, they will represent the same rotation. This is only when either of the 2 following conditions are satisfied.

-> q1 is component wise approximately equal to q2
-> q1 is component wise approximately equal to -q2

Inverse of (a, -b, -c, -d) and (-a, b, c, d) we have (a, b, c, d) and (-a, -b,-c, -d)

Hence we show that:

(a, b, c, d) = (-a, -b, -c, -d)

This proves that (-a, b, c, d) = (a, -b, -c, -d)

$$Q = [\cos(\theta/2), V1\sin(\theta/2), V2\sin(\theta/2), V3\sin(\theta/2)]$$

A b c d

$$Q = -q$$

$$(a, b, c, d) = (-a, -b, -c, -d)$$

Replace θ by $\theta+2\pi$

$$Q = [\cos(\theta/2+\pi), V1\sin(\theta/2+\pi), V2\sin(\theta/2+\pi), V3\sin(\theta/2+\pi)]$$
$$= [-\cos(\theta/2), -V1\sin(\theta/2), -V2\sin(\theta/2), -V3\sin(\theta/2)]$$
$$= (-a,-b,-c,-d)$$
$$= -q$$

```python
from spatialmath import *
from spatialmath.base import *
import spatialmath.base.symbolic as sym
import numpy as np
from matplotlib import pyplot as plt



theta = sym.symbol('theta')
v1 = sym.symbol('v1')
v2 = sym.symbol('v2')
v3 = sym.symbol('v3')
a = sym.cos(theta/2)
b = v1*sym.sin(theta/2)
c = v2*sym.sin(theta/2)
d = v3*sym.sin(theta/2)
q1 = UnitQuaternion([np.cos(-45*np.pi/180), 0.256*np.sin(-60*np.pi/180), 0.343*np.sin(-30*np.pi/180), 0.904*np.sin(-30*np.pi/180)])
print("Quanternions of q1 ")
print(q1)
q2 = UnitQuaternion([np.cos(150*np.pi/180), 0.256*np.sin(150*np.pi/180), 0.343*np.sin(150*np.pi/180), 0.904*np.sin(150*np.pi/180)])
print("Quanternions of q2 ")
print(q2)


if q1==q1:
    print("q1 and q2 are equal and obeys equality")
```

```
Quanternions of q1
 0.7992 << -0.2506, -0.1938, -0.5109 >>
Quanternions of q2
 0.8660 << -0.1280, -0.1715, -0.4520 >>
q1 and q2 are equal and obeys equality
```

6. What is Hamiltonian product? Show that multiplying 2 quaternions will result in the Hamiltonian product. Also show that rotation matrix multiplication is equivalent to quaternion multiplication. (Hint: take two sample rotation matrix and multiply to get a product matrix and use it on a sample vector. Convert the rotation matrix or corresponding axis angle representation to quaternions and do the multiplication.)

After explaining what quaternions are, I'll clarify what the equation above means. Complex numbers are created by adding a special symbol called I to real numbers while keeping in mind that i2 = -1. Similar to trigonometric functions, quaternions are created by adding to the real integers I j, and k with the condition that

i2 = j2 = k2 = ijk = -1

A formal sum of a real number and real multiples of the letters I j, and k is known as a quaternion. Consider the formula q = w + xi + yj + zk.

In this illustration, the real and imaginary components of a complex number are represented by w and xi + yj + zk, respectively. The product of the quaternions q = xi + yj + zk and q' = x'i + y'j + z'k is the quaternion product of the two vectors (x, y, z) and (x', y', z'). The formula for the quaternion product qq' is (xx' + yy' + zz') + (yz' - zy')i + (zx' -xz')j + (xy' - yx')k.

The real part is the negative of the dot product of (x , y , z) and (x’ , y’ , z’) and the vector product of
(x , y , z) and (x’ , y’ , z’).

```
v = [5,7,9]
ham_prod = SO3.Rx(30,unit='deg')*v
print("The Hamiltonian product is:")
print(ham_prod)
q_mul = UnitQuaternion.Rx(30,"deg")*v
print("The quaternion multiplication is:")
print(q_mul)
```

```
The Hamiltonian product is:
[[ 5.        ]
 [ 1.56217783]
 [11.29422863]]
The quaternion multiplication is:
[ 5.          1.56217783 11.29422863]
```

Part B

1. A Roll-Pitch-Yaw rotation matrix is given below. Can you find the missing elements in the rotation matrix R and the Roll-Pitch-Yaw angles.

$$R = \begin{bmatrix} 0.4 & ? & ? \\ 0.138 & ? & ? \\ -0.906 & 0.224 & 0.358 \end{bmatrix}$$

Comparing the given rotation matrix with XYZ(FAR) matrix

$$\begin{bmatrix} 0.4 & ? & ? \\ 0.138 & ? & ? \\ -0.906 & 0.224 & 0.358 \end{bmatrix} = \begin{bmatrix} cos\alpha cos\beta & cos\alpha sin\beta siny - sin\alpha cosy & cos\alpha sin\beta cosy + sin\alpha siny \\ sin\alpha cos\beta & sin\alpha sin\beta siny + cos\alpha cosy & sin\alpha sin\beta cosy - cos\alpha siny \\ -sin\beta & cos\beta siny & cos\beta cosy \end{bmatrix}$$

$$-sin\beta = -0.906$$

$$\boldsymbol{\beta = \sin^{-1}(0.906) = 64.95°}$$

$$cos\alpha cos\beta = 0.4$$
$$cos\alpha(0.423) = 0.4$$
$$cos\alpha = 0.945$$

$$\boldsymbol{\alpha = cos^{-1}(0.945) = 19.09°}$$

$$cos\beta cos\gamma = 0.358$$
$$(0.423)cos\gamma = 0.358$$
$$cos\gamma = 0.358/0.423$$
$$cos\gamma = 0.846$$

$$\boldsymbol{\gamma = \cos^{-1}(0.846) = 32.24°}$$

So, from the β, $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ angles we ca get the following angles::

$$sin\beta = 0.906$$
$$cos\beta = 0.423$$
$$cos\alpha = 0.945$$
$$sin\alpha = 0.327$$
$$cos\gamma = 0.846$$
$$sin\gamma = 0.533$$

- $cos\alpha sin\beta sin\gamma - sin\alpha cos\gamma \ \ (r12)$

$= (0.945)(0.906)(0.533) - (0.327)(0.846)$
$= 0.456 - 0.276$
$= 0.18$

- $sin\alpha sin\beta sin\gamma + cos\alpha cos\gamma \quad (r22)$

$= (0.327)(0.906)(0.533) + (0.945)(0.846)$
$= 0.957$

- $cos\alpha sin\beta cos\gamma + sin\alpha sin\gamma \ \ (r13)$

$= (0.945)(0.906)(0.846) + (0.327)(0.533)$
$= 0.898$

- $sin\alpha sin\beta cos\gamma - cos\alpha sin\gamma \quad (r23)$

$= (0.327)(0.906)(0.846) + (0.945)(0.533)$
$= -0.2545$

$$R = \begin{bmatrix} 0.4 & 0.181 & 0.898 \\ 0.138 & 0.957 & -0.254 \\ -0.906 & 0.224 & 0.358 \end{bmatrix}$$

```python
x = sym.symbol('x')

rpy_matrix = np.matrix([[0.4,x,x],[0.138,x,x],[-0.906,0.224,0.358]])
print(rpy_matrix)
print()
tranform_matrix_2_rpy_angles = tr2rpy(rpy_matrix)
print(tranform_matrix_2_rpy_angles)
print()
new_rpy_matrix = rpy2r(tranform_matrix_2_rpy_angles)
print(new_rpy_matrix)
```

```
[[0.4 x x]
 [0.138 x x]
 [-0.906 0.224 0.358]]

[0.55910132 1.13386475 0.33221355]

[[ 0.400024    0.17783971  0.899085  ]
 [ 0.13800828  0.95811947 -0.25091989]
 [-0.90605436  0.22445515  0.35872743]]
```

2. An ZYZ (RAR) sequence produces the final matrix as given below. Find the missing elements and angles if angle α is 0°.

$$R = \begin{bmatrix} -0.82 & ? & 0 \\ ? & ? & ? \\ ? & ? & -1 \end{bmatrix}$$

Now, Comparing the given rotation matrix with ZYZ(RAR) matrix

$$\begin{bmatrix} -0.82 & ? & 0 \\ ? & ? & ? \\ ? & ? & -1 \end{bmatrix} = \begin{bmatrix} C_\alpha C_\beta C_\gamma - S_\alpha S_\gamma & -C_\alpha C_\beta S_\gamma - S_\alpha C_\gamma & C_\alpha S_\beta \\ S_\alpha C_\beta C_\gamma + C_\alpha S_\gamma & -S_\alpha C_\beta S_\gamma + C_\alpha C_\gamma & S_\alpha S_\beta \\ -S_\beta C_\gamma & S_\beta S_\gamma & C_\beta \end{bmatrix}$$

$$\boldsymbol{\alpha = 0}$$

$$cos\beta = -1$$

$$\boldsymbol{\beta = \cos^{-1}(-1) = 180^{\circ}}$$

$$cos\alpha cos\beta cos\gamma - sin\alpha sin\gamma = -0.82$$
$$(-1)cos\gamma - 0 = -0.82$$

$$\boldsymbol{\gamma = \cos^{-1}(0.82) = 34.91^{\circ}}$$

So, from the β, $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ angles we ca get the following angles::

$$sin\beta = 0$$
$$cos\beta = -1$$
$$cos\alpha = 1$$
$$sin\alpha = 0$$
$$cos\gamma = 0.82$$
$$sin\gamma = 0.57$$

$(r32) = sin\beta sin\gamma \Rightarrow 0$ $(r23) = sin\alpha sin\beta \Rightarrow 0$

$(r13) = cos\alpha si\beta \Rightarrow 0$ $(r31) = -sin\beta cos\gamma \Rightarrow 0$

.

$$-sin\alpha cos\beta sin\gamma + cos\alpha cos\gamma \Rightarrow 0.82 \qquad (r22)$$

$$sin\alpha cos\beta cos\gamma + cos\alpha sin\gamma \Rightarrow 0 + (1)(0.57) = 0.57 \qquad (r21)$$

$$-cos\alpha cos\beta sin\gamma - sin\alpha cos\gamma \Rightarrow (-1)(-1)(0.57) = 0.57 \qquad (r12)$$

$$\boldsymbol{R} = \begin{bmatrix} \mathbf{-0.82} & \mathbf{0.57} & \mathbf{0} \\ \mathbf{0.57} & \mathbf{0.82} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{-1} \end{bmatrix}$$

```python
y = sym.symbol('y')

b = sym.symbol('β')
c = sym.symbol('γ')

R = np.matrix([[-0.82,y,0],[y,y,y],[y,y,-1]])
print(R)
print()
R1 = SO3.Rz(0,unit = 'deg')*SO3.Ry(b)*SO3.Rz(c)
print(R1)
rpy2r(0,3.14,0.609)
```

```
[[-0.82 y 0]
 [y y y]
 [y y -1]]

  1.0*cos(β)*cos(γ) -1.0*sin(γ)*cos(β) 1.0*sin(β)
  1.0*sin(γ)   1.0*cos(γ)   0
  -1.0*sin(β)*cos(γ) 1.0*sin(β)*sin(γ) 1.0*cos(β)


array([[-8.20219435e-01, -5.72047526e-01,  1.30632653e-03],
       [-5.72046800e-01,  8.20220475e-01,  9.11073160e-04],
       [-1.59265292e-03,  0.00000000e+00, -9.99998732e-01]])
```

3. A rotation matrix is given by:

$$R = \begin{bmatrix} 0.3333 & 0.9107 & 0.2440 \\ -0.2440 & 0.3333 & 0.9107 \\ -0.9107 & 0.2440 & 0.3333 \end{bmatrix}$$

Can you determine the equivalent angle-axis representation that will result in this rotation matrix? Is your choice unique? If yes (or no) explain the reason. If you feel that the choice is non-unique, provide the alternate angle-axis representation and subsequently create quaternions using both the axis angle representations. Also verify whether the quaternions are unique or not.

```
import math
R = np.matrix([[0.3333,0.9107,0.2440],
               [-0.2440,0.3333,0.9107],
               [-0.9107,0.2440,0.3333]])

[x,e] = np.linalg.eig(R)
print(x)
print()
print(e)
print()
axis_angle = math.acos((np.trace(R) - 1)/2)
print(axis_angle)
print()
q = UnitQuaternion(rpy2r(axis_angle,axis_angle,axis_angle))
print(q)
```

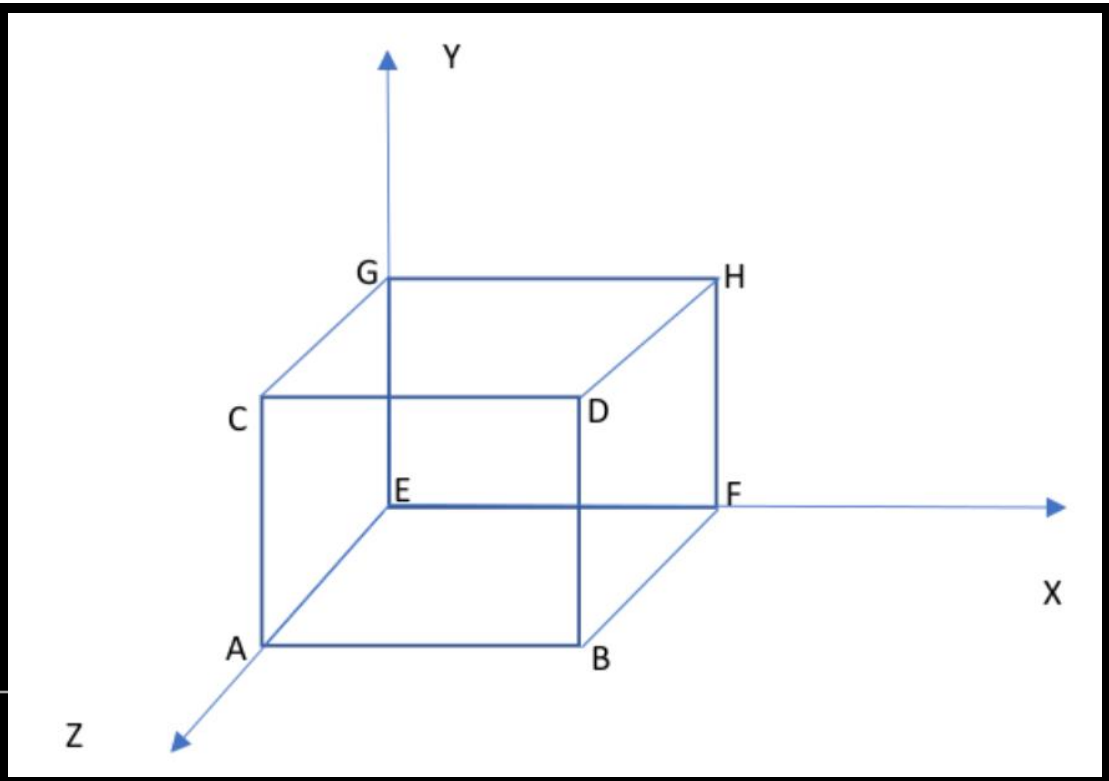
```
[ 0.75125533+0.86386965j  0.75125533-0.86386965j -0.50261066+0.j        ]

[[ 0.63276418+0.j          0.63276418-0.j          0.45217565+0.j        ]
 [ 0.32259785+0.46149244j  0.32259785-0.46149244j -0.59344164+0.j        ]
 [-0.12017499+0.51780577j -0.12017499-0.51780577j  0.66585599+0.j        ]]

1.5708463267949175

 0.7071 << -0.0000,  0.7071, -0.0000 >>
```

4. A unit cube (all sides are of unit length) is shown below. Determine the rotation matrix corresponding to rotation of the cube about its solid diagonal AH by 90° in the counterclockwise direction. Determine the global coordinates of the vertex F after the rotation. Can we arrive at the same orientation (orientation obtained by rotating the cube about the solid diagonal AH by 90°in the counterclockwise direction) by a combination of elementary rotations (Rotation about the coordinate axes)? If yes, mention the elementary rotations to be performed and their order.



In this question, rotation happens about the diagonal AH. The rotation is about 90 degrees in counter clockwise direction.
To find the rotation matrix we can make a quaternion representing the rotation matrix, where the V vector is the AH vector and the angle is 90 degrees.

We have the position of F with respect to local frame, I.e., F = [1,0,0]

So we can multiply the quaternion with quaternion and find its position with respect to the global frame.

Thank you!!!!