MEAM 6200

Project 0

Q1 (a) $T_1 = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 & 3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

(b) $T_2 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

(c) $T_3 = T_1 T_2$

$$= \begin{bmatrix} -1/\sqrt{2} & 0 & 1/\sqrt{2} & \sqrt{2} \\ 1/\sqrt{2} & 0 & 1/\sqrt{2} & 3+\sqrt{2} \\ & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q2

$$R = \begin{bmatrix} a_{11} & 0.892 & 0.423 \\ a_{21} & a_{22} & a_{23} \\ -0.186 & a_{32} & a_{33} \end{bmatrix}$$

$\Rightarrow$ using the fact that $R^T R = R R^T = I$
$\therefore$ rows & columns of R are orthogonal + unit norms

* Using row 1 norm $= 1$

$$\Rightarrow a_{11}^2 + 0.892^2 + 0.423^2 = 1$$
$$\Rightarrow a_{11} = \pm 0.159$$

* 11^{ly} Using column 1 norm $= 1$

$$\Rightarrow a_{11}^2 + a_{21}^2 + (-0.186)^2 = 1$$
$$\Rightarrow a_{21} = \pm 0.97$$

* Now, for each of these (a_{11}, a_{21}) combinatn; I calculate (a_{22}, a_{32}) pair using the following facts

$\rightarrow$ dot product of column 1 & 2 $= 0$

$\rightarrow$ norm of col 2 $= 1$

$$\therefore \quad a_{22}^2 + a_{32}^2 = 1 - (0.892)^2$$
$$\& \quad a_{22} * a_{21} + a_{32} * (-0.186) = -0.892 * a_{11}$$

$\Rightarrow$ this is equivalent to solving

$$\left.\begin{aligned} x^2 + y^2 &= c \\ ax + by &= d \end{aligned}\right] - \text{can result in two possible } (x, y) \text{ values}$$

11^{ly} I do for (a_{23}, a_{33}) pair.

→ In total ; this results in 32 possible rotatⁿ matrices — but not all are valid

→ Finally ; I eliminate the wrong Rotⁿ matrics ~~possible~~ using following criterias

$$\rightarrow RR^T = R^TR = I$$

$$\rightarrow \det(R) = 1$$

⇒ leading to following 4 valid ~~possibilities~~ solutⁿs

$$\rightarrow \begin{bmatrix} 0.159, & 0.892, & 0.423 \\ 0.97, & -0.061, & -0.237 \\ -0.186, & 0.448, & -0.875 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 0.159, & 0.892, & 0.423 \\ -0.97, & 0.222 & -0.103 \\ -0.186, & -0.394, & 0.9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} -0.159, & 0.892, & 0.423 \\ 0.97, & 0.222, & -0.103 \\ -0.186, & 0.394, & -0.9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} -0.159, & 0.892, & 0.423 \\ -0.97, & -0.061, & -0.237 \\ -0.186, & -0.448, & 0.875 \end{bmatrix}$$

* $\boxed{\text{Attached code for solving these}}$

Q3 (a) No; the axis-angle representatn is not unique.
Take the following cases

(i) R = Identity
in this case $\theta = 0$;
but the axis can be anything
(any unit ~~norm~~ vector)

(ii) even for non-identity rotatns
$(u, \theta) \equiv (-u, -\theta)$

reverse the axis & negating the angle result in same rotatn

(b) $$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\cos(\pi/6) & \sin(\pi/6) \\ 0 & \sin(\pi/6) & \cos(\pi/6) \end{bmatrix}$$

for finding (u, θ)

$\rightarrow$ $\cos\theta = \dfrac{tr(R) - 1}{2} = \dfrac{-\cancel{2}}{\cancel{2}} = -1$

$\Rightarrow$ $\boxed{\theta = \pi}$

$\rightarrow$ for 'u' $\rightarrow$ $Ru = u$
$\Rightarrow$ 'u' is the eigen-vector corresp. to eigenvalue $= 1$

$\Rightarrow$ $\boxed{u = [0, 0.259, 0.966]}$ - found using numpy

→ Calculating 'u' manually.

$$Ru = u$$

$$\Rightarrow \begin{bmatrix} -1 & 0 & 0 \\ 0 & -\cos(\pi/6) & \sin(\pi/6) \\ 0 & \sin(\pi/6) & \cos(\pi/6) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} -2 & 0 & 0 \\ 0 & -\cos\pi/6 - 1 & \sin(\pi/6) \\ 0 & \sin(\pi/6) & \cos(\pi/6) - 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\Rightarrow \quad u_1 = 0$$

$$\left(-1 - \frac{\sqrt{3}}{2}\right) u_2 + \frac{u_3}{2} = 0$$

$$\Rightarrow \quad u_3 = (2+\sqrt{3})\, u_2$$

also $u_1^2 + u_2^2 + u_3^2 = 1$

$$\Rightarrow \quad u = \begin{bmatrix} 0 \\ 1/2\sqrt{2+\sqrt{3}} \\ (2+\sqrt{3})/2\sqrt{2+\sqrt{3}} \end{bmatrix}$$

⇒ Axis-angle representatⁿ

$\Rightarrow u = [0, 0.259, 0.966]$

$\theta = \pi$

⇒ Quaternion representatⁿ

$$\Rightarrow q = \cos\frac{\theta}{2} + u \sin\frac{\theta}{2}$$

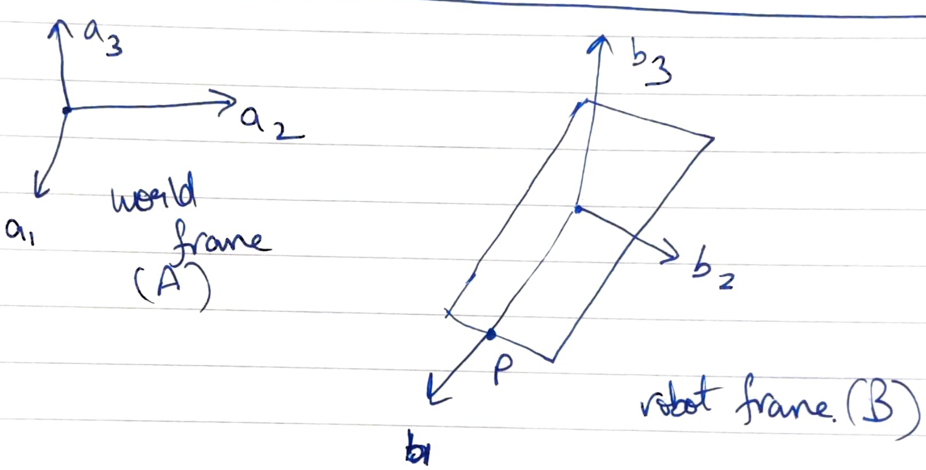
~~⇒ [0, 0.259,~~

$$q = [0, 0, 0.259, 0.966]$$

scalar part $\Rightarrow \boxed{0 + 0\hat{i} + 0.259\hat{j} + 0.966\hat{k}}$

---

(4)



$$d(t) = \begin{bmatrix} \cos(0.1t) \\ \sin(0.12t) \\ \sin(0.08t) \end{bmatrix} \quad ; \quad R(t) = \begin{bmatrix} \cos(t) & -\sin(t) & 0 \\ \sin(t) & \cos(t) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(a) Linear velocity $v(t) = \dot{d}(t)$

$$= \frac{d}{dt}\begin{bmatrix} \cos(0.1t) \\ \sin(0.12t) \\ \sin(0.08t) \end{bmatrix} = \begin{bmatrix} -0.1\sin(0.1t) \\ 0.12\cos(0.12t) \\ 0.08\cos(0.08t) \end{bmatrix}$$

(b) $${}^{A}P(t) = \underbrace{{}^{A}H_{B}}_{\text{Homogenous matrix from } B \text{ to } A} P^{B}$$

$$= R(t)\, P^{B} + d(t)$$

$$= c\begin{bmatrix} \cos(t) \\ \sin(t) \\ 0 \end{bmatrix} + \begin{bmatrix} \cos(0.1t) \\ \sin(0.12t) \\ \sin(0.08t) \end{bmatrix}$$

$${}^{A}P(t) = \begin{bmatrix} c.\cos(t) + \cos(0.1t) \\ c.\sin(t) + \sin(0.12t) \\ \sin(0.08t) \end{bmatrix}$$

(d) Let the frame of reference of robot be B_1 at time $=1$ & B_5 at time $=5$

$\Rightarrow$ Using the fact that gravity vector is same in world co-ordinates at all times

$$\therefore\ {}^{A}H_{B_1}\, {}^{B_1}g = \underbrace{{}^{A}H_{B_5}}_{\text{Homogeneous matrix to move from coordinates in } B_5 \text{ to coordinates in world frame } A} \underbrace{{}^{B_5}g}_{\text{gravity vector in Body frame at } t=5}$$

but given that ~~g/vect~~ gravity is a fixed vector ; hence it won't change with translatn

$$\Rightarrow \; {}^{A}R_{B_1}\,{}^{B_1}g = {}^{A}R_{B_5}\,{}^{B_5}g$$

$$\cancel{\Rightarrow \; {}^{B_1}g = \left({}^{A}R_{B_1}\right)^T {}^{A}R_{B_5}\,{}^{B_5}g}$$

$$\Rightarrow \; {}^{B_5}g = \left({}^{A}R_{B_5}\right)^T {}^{A}R_{B_1}\,{}^{B_1}g$$

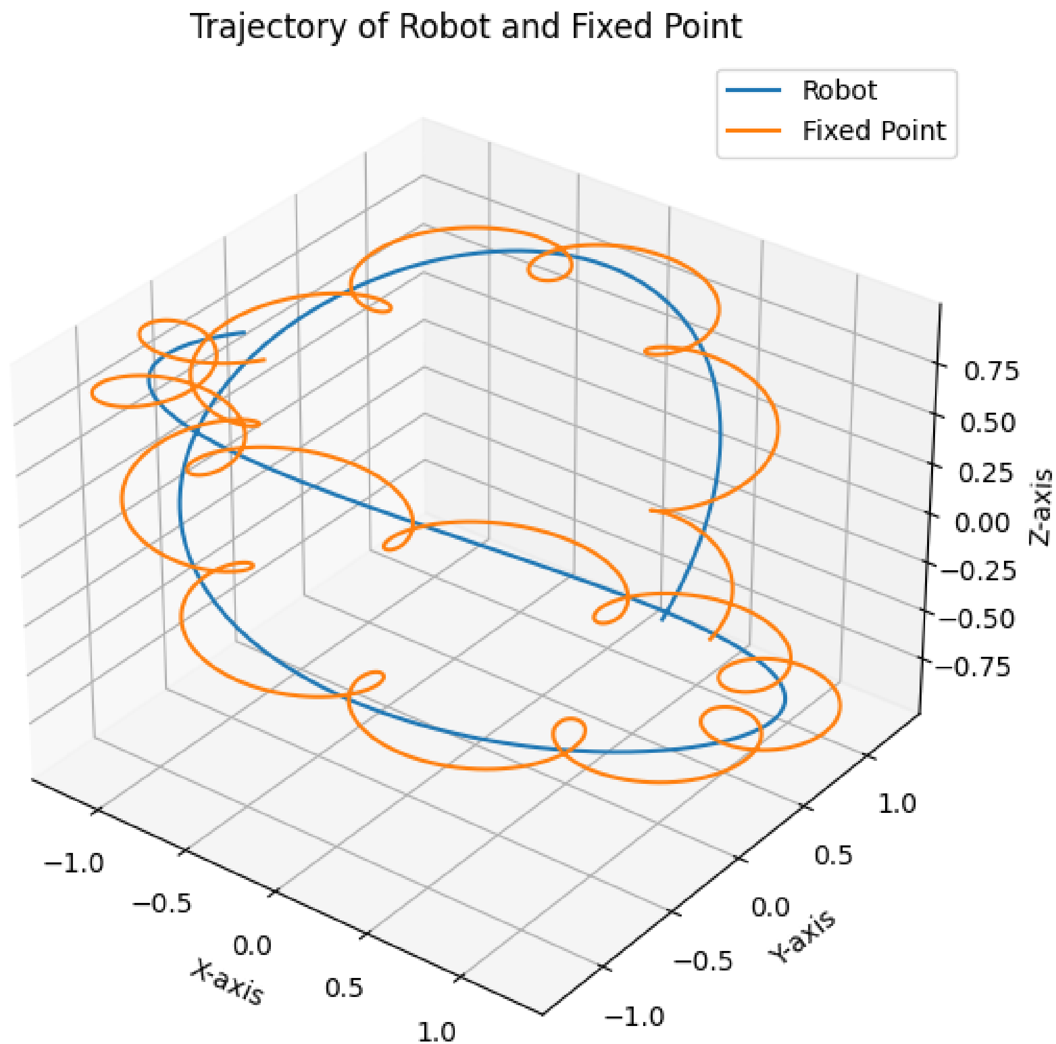
$$= \begin{bmatrix} \cos(5) & \sin(5) & 0 \\ -\sin(5) & \cos(5) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(1) & -\sin(1) & 0 \\ \sin(1) & \cos(1) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1.1 \\ 0.9 \\ -9.7 \end{bmatrix}$$

$$= \begin{bmatrix} -1.40 \\ 0.24 \\ -9.7 \end{bmatrix}$$

```python
import numpy as np

DEBUG = False

def debug_print(*args):
    if DEBUG:
        print(*args)


def printRotationMatrix(R):
    print("[")
    for row in R:
        print("    ", np.round(row, 3))
    print("]")


def checkValidRotMatrix(R):
    # Check if the given matrix is a rotation matrix
    # Check if the determinant is 1
    det = np.linalg.det(R)
    if np.abs(det - 1) > 1e-6:
        debug_print("Determinant is not 1: ", det)
        return False

    # Check if the inverse is equal to the transpose
    R_inv = np.linalg.inv(R)
    R_T = np.transpose(R)
    if not np.allclose(R_inv, R_T):
        debug_print("Inverse is not equal to transpose")
        return False

    return True


"""
Given
    -> x^2 + y^2 = c,
    -> ax + by = d
Find x, y
"""
def solve_quadratic_and_linear(a, b, c, d):
    # x = (d - by) / a
    # -> (d - by)^2 / a^2 + y^2 = c
    # -> (d^2 - 2bdy + b^2y^2) / a^2 + y^2 = c
    # -> d^2 - 2bdy + b^2y^2 + a^2y^2 = c*a^2
    # -> (b^2 + a^2)y^2 - 2bdy + d^2 - c*a^2 = 0
    # -> y = (bd +- sqrt(b^2d^2 - (b^2 + a^2)(d^2 - c*a^2))) / (b^2 + a^2)
    y = (
```

```
        (b*d + np.sqrt(b**2 * d**2 - (b**2 + a**2)*(d**2 - c*a**2))) / (b**2 +
a**2),
        (b*d - np.sqrt(b**2 * d**2 - (b**2 + a**2)*(d**2 - c*a**2))) / (b**2 +
a**2)
    )

    x = (
        (d - b*y[0]) / a,
        (d - b*y[1]) / a
    )

    return x, y


def RotationMatrixComplete(a12 = 0.892, a13 = 0.423, a31 = -0.186):
    # Given three entries of the rotation matrix
    # Find possible rotation matrices
    # R = [
    #     [a11,    0.892, 0.423],
    #     [a21,    a22  , a23  ],
    #     [-0.186, a32  , a33  ]
    # ]

    # 1. First find possible values of a11, using row 1 being
    #    unit norm vector.
    a11 = (np.sqrt(1 - a12**2 - a13**2), -np.sqrt(1 - a12**2 - a13**2))

    # 2. Next find possible values of a21, using column 1 being
    #    unit norm vector.
    a21 = (np.sqrt(1 - a31**2 - a11[0]**2), -np.sqrt(1 - a31**2 - a11[0]**2))

    a11_21_combined = [(a11[0], a21[0]), (a11[0], a21[1]), (a11[1], a21[0]),
(a11[1], a21[1])]

    # 3. Use norm of row 2 = 1 to create one equation
    # -> a22^2 + a23^2 = 1 - a21^2
    # Use orthogonality of row 2 and row 1 to create another equation
    # -> a22*a12 + a23*a13 = -a21*a11
    # We can solve these two equations to find possible values of a22, a23
    a11_21_22_23_combined = []
    for a11_21 in a11_21_combined:
        a22, a23 = solve_quadratic_and_linear(a12, a13, 1 - a11_21[1]**2, -
a11_21[1]*a11_21[0])
        a11_21_22_23_combined.append((a11_21[0], a11_21[1], a22[0], a23[0]))
        a11_21_22_23_combined.append((a11_21[0], a11_21[1], a22[1], a23[1]))

    # 4. Use norm of col2 and col3 = 1 to find a32, a33
    # -> a12^2 + a22^2 + a32^2 = 1
    all_combined = []
    for a11_21_22_23 in a11_21_22_23_combined:
        a32 = np.sqrt(1 - a12**2 - a11_21_22_23[2]**2)
```

```python
        a33 = np.sqrt(1 - a13**2 - a11_21_22_23[3]**2)
        all_combined.append((a11_21_22_23[0], a11_21_22_23[1], a11_21_22_23[2], a11_21_22_23[3], a32, a33))
        all_combined.append((a11_21_22_23[0], a11_21_22_23[1], a11_21_22_23[2], a11_21_22_23[3], -a32, a33))
        all_combined.append((a11_21_22_23[0], a11_21_22_23[1], a11_21_22_23[2], a11_21_22_23[3], a32, -a33))
        all_combined.append((a11_21_22_23[0], a11_21_22_23[1], a11_21_22_23[2], a11_21_22_23[3], -a32, -a33))

    # Compute all possible rotation matrices, print only those which are valid
    print("Possible Rotation Matrices: ")
    for a11_21_22_23_32_33 in all_combined:
        a11, a21, a22, a23, a32, a33 = a11_21_22_23_32_33
        R= np.array([
            [a11, a12, a13],
            [a21, a22, a23],
            [a31, a32, a33]
        ])
        if checkValidRotMatrix(R):
            printRotationMatrix(R)
            print("")


if __name__ == "__main__":
    RotationMatrixComplete()
```



Trajectory of Robot and Fixed Point
Robot
Fixed Point
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
Z-axis
−1.0
−0.5
0.0
0.5
1.0
X-axis
1.0
0.5
0.0
−0.5
−1.0
Y-axis