ME 639

INTRODUCTION TO ROBOTICS

Assignment 2

Submitted by:

Rhitosparsha Baishya
23310039
PhD (Mechanical Engineering)

CONTENTS

Question 1

Q1

To prove $\rightarrow R\,S(a)R^T = S(Ra)$

Let $a = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix}$, $R = R_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$

$$S(a) = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix} \; ; \; R^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

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

$$S(Ra) = \begin{bmatrix} 0 & -\beta\sin\theta - \gamma\cos\theta & \beta\cos\theta - \gamma\sin\theta \\ \beta\sin\theta + \gamma\cos\theta & 0 & -\alpha \\ -\beta\cos\theta + \gamma\sin\theta & \alpha & 0 \end{bmatrix}$$

Now,

$$R\,S(a)\,R^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

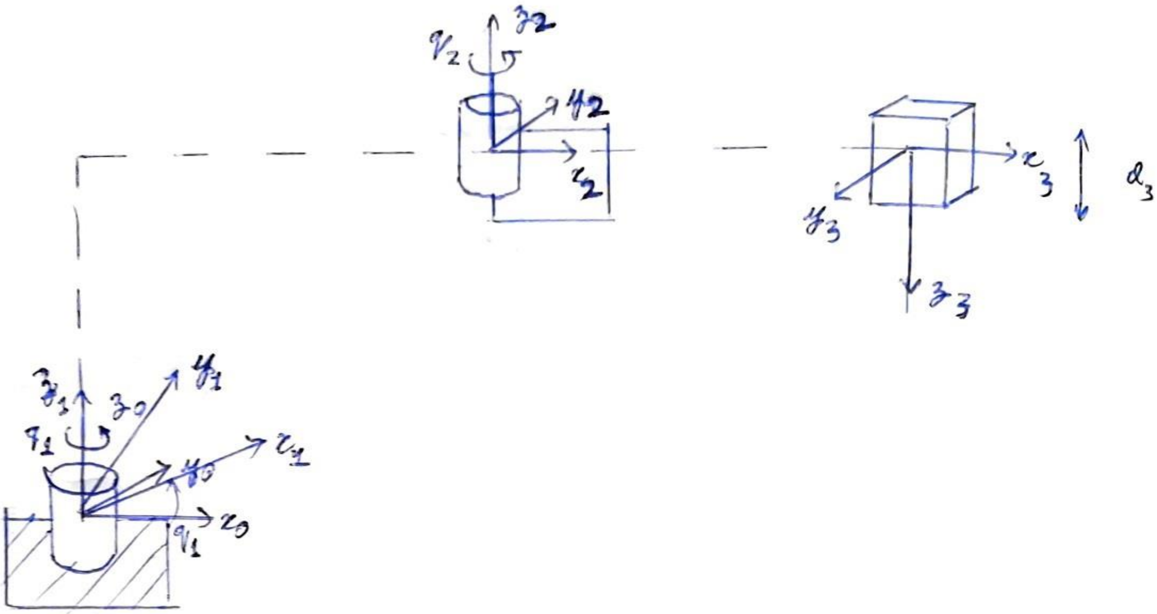
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 0 & -\beta\sin\theta - \gamma\cos\theta & \beta\cos\theta - \gamma\sin\theta \\ \gamma & \alpha\sin\theta & -\alpha\cos\theta \\ -\beta & \alpha\cos\theta & \alpha\sin\theta \end{bmatrix}$$

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

$= S(Ra)$ // $\leftarrow$ Hence proved.

Question 2

Q2



Set $P_0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Here, $P_3 = \begin{bmatrix} 0 \\ 0 \\ -l_3 \end{bmatrix}$, $R_2^3 = R_{z,0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$d_2^3 = \begin{bmatrix} 0 \\ l_2 \\ -d_3 \end{bmatrix}$, $R_1^2 = R_{z,q_2} = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 \\ \sin q_2 & \cos q_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$d_1^2 = \begin{bmatrix} 0 \\ l_1 \\ 0 \end{bmatrix}$, $R_0^1 = R_{z,q_1} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 \\ \sin q_1 & \cos q_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$d_0^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Now,

$$H_0^1 = \begin{bmatrix} R_0^1 & d_0^1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_1^2 = \begin{bmatrix} R_1^2 & d_1^2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & 0 \\ \sin q_2 & \cos q_2 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^3 = \begin{bmatrix} R_2^3 & d_2^3 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & -l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore \quad P_0 = H_0^1 \, H_1^2 \, H_2^3 \, P_3$$

$$\Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & 0 \\ \sin q_2 & \cos q_2 & 0 & l_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & l_2 \\ 0 & 0 & 1 & -l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -l_3 \\ 1 \end{bmatrix}$$

Question 3

Source Code: (Link to GitHub)

```
import numpy as np

"""
Program to calculate the end effector position vector for an RRP SCARA robot.

Parameters:
q1 = Angle of Joint 1
q2 = Angle of Joint 2
d3 = Extension length of the robot arm
l1 = Length of Link 1
l2 = Length of Link 2
"""

def scara_fwd_kin(q1, q2, l1, l2, d3):
    x = l1 * np.cos(q1) + l2 * np.cos(q1 + q2)
    y = l1 * np.sin(q1) + l2 * np.sin(q1 + q2)
    z = d3
    return x, y, z

l1 = float(input("Enter length of Link 1 (m): "))
l2 = float(input("Enter length of Link 2 (m): "))
q1 = np.radians(float(input("Enter angle of Joint 1 (deg): ")))
q2 = np.radians(float(input("Enter angle of Joint 2 (deg): ")))
d3 = float(input("Enter extension length (m): "))

x, y, z = scara_fwd_kin(q1, q2, l1, l2, d3)

print("\nEnd effector position :")
print(str(x) + " \033[1mi\033[0m + " + str(y) + " \033[1mj\033[0m + " + str(z)
+ " \033[1mk\033[0m ")

input()
```

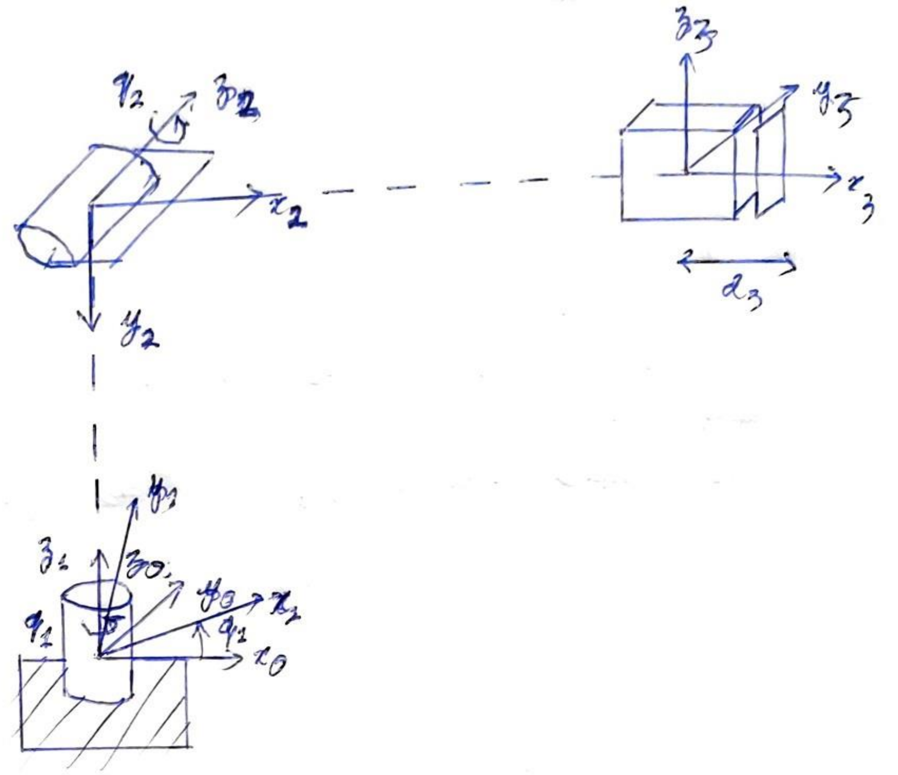
Output:

```
Enter length of Link 1 (m): 2
Enter length of Link 2 (m): 2
Enter angle of Joint 1 (deg): 30
Enter angle of Joint 2 (deg): 45
Enter extension length (m): 1

End effector position vector:
2.249688897773919 i + 2.9318516525781364 j + 1.0 k
```

Question 4

Q4



Let $P_0 = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

Here, $P_3 = \begin{bmatrix} l_3 \\ 0 \\ 0 \end{bmatrix}$, $R_2^3 = R_{z,0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$, $d_2^3 = \begin{bmatrix} l_2 + d_3 \\ 0 \\ 0 \end{bmatrix}$

$$R_1^2 = R_{x,\pi/2} \; R_{z,q_2}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \pi/2 & -\sin \pi/2 \\ 0 & \sin \pi/2 & \cos \pi/2 \end{bmatrix} \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 \\ \sin q_2 & \cos q_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 \\ 0 & 0 & -1 \\ \sin q_2 & \cos q_2 & 0 \end{bmatrix}$$

$$d_1^2 = \begin{bmatrix} 0 \\ 0 \\ l_1 \end{bmatrix} \quad , \quad R_0^1 = R_{z,q_1} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 \\ \sin q_1 & \cos q_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad , \quad d_0^1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Now,

$$H_0^1 = \begin{bmatrix} R_0^1 & d_0^1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_1^2 = \begin{bmatrix} R_1^2 & d_1^2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin q_2 & \cos q_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_2^3 = \begin{bmatrix} R_2^3 & d_2^3 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & l_2 + l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\therefore P_0 = H_0^1 \, H_1^2 \, H_2^3 \, P_3$$

$$\Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & 0 \\ \sin q_1 & \cos q_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos q_2 & -\sin q_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin q_2 & \cos q_2 & 0 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & l_2+l_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} l_3 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Source Code: (Link to GitHub)

```
import numpy as np

"""
Program to calculate the end effector position vector for an RRP Stanford
robot.

Parameters:
q1 =  Angle of Joint 1
q2 =  Angle of Joint 2
l1 = Length of Link 1
l2 = Length of Link 2
l3 = Length of link 3
d3 = Linear displacement
"""

def stanford_fwd_kin(q1, q2, l1, l2, l3, d3):
    P3 = np.array([l3, 0, 0, 1])

    H_01 = np.array([
                        [np.cos(q1), np.sin(q1), 0, 0],
                        [-np.sin(q1), np.sin(q1), 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]
                    ])

    H_12 = np.array([
                        [np.cos(q2), -np.sin(q2), 0, 0],
                        [0, 0, -1, 0],
                        [np.sin(q2), np.cos(q2), 0, l1],
                        [0, 0, 0, 1]
                    ])

    H_23 = np.array([
                        [1, 0, 0, l2 + d3],
                        [0, 1, 0, 0],
                        [0, 0, 1, 0],
                        [0, 0, 0, 1]
                    ])

    P0 = np.dot(np.dot(np.dot(H_01, H_12), H_23), P3)

    return P0

l1 = float(input("Enter length of Link 1 (m): "))
l2 = float(input("Enter length of Link 2 (m): "))
```

```
l3 = float(input("Enter length of Link 3 (m): "))
q1 = np.radians(float(input("Enter angle of Joint 1 (deg): ")))
q2 = np.radians(float(input("Enter angle of Joint 2 (deg): ")))
d3 = float(input("Enter extension length (m): "))

pos = stanford_fwd_kin(q1, q2, l1, l2, l3, d3)

print("\nEnd effector position :")
print(str(pos[0]) + " \033[1mi\033[0m + " + str(pos[1]) + " \033[1mj\033[0m + " + str(pos[2]) + " \033[1mk\033[0m ")

input()
```

Output:

```
Enter length of Link 1 (m): 2
Enter length of Link 2 (m): 2
Enter length of Link 3 (m): 2
Enter angle of Joint 1 (deg): 30
Enter angle of Joint 2 (deg): 45
Enter extension length (m): 1

End effector position :
3.0618621784789726 i + -1.7677669529663687 j + 5.535533905932738 k
```

Question 5

Q5

Let $x_0 - y_0 - z_0$ be the base frame

$x_1 - y_1 - z_1$ be the drone frame

$x_2 - y_2 - z_2$ be drone frame after first rotation

$x_3 - y_3 - z_3$ be drone frame after second rotation

Initial pos. of drone,

$$T_{init} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

First rot^n of drone,

$$R_1 = R_{x,30^\circ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos 30^\circ & -\sin 30^\circ & 0 \\ 0 & \sin 30^\circ & \cos 30^\circ & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Second rot^n of drone,

$$R_2 = R_{z,60^\circ} = \begin{bmatrix} \cos 60^\circ & -\sin 60^\circ & 0 & 0 \\ \sin 60^\circ & \cos 60^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Pos. of obstacle relative to drone,

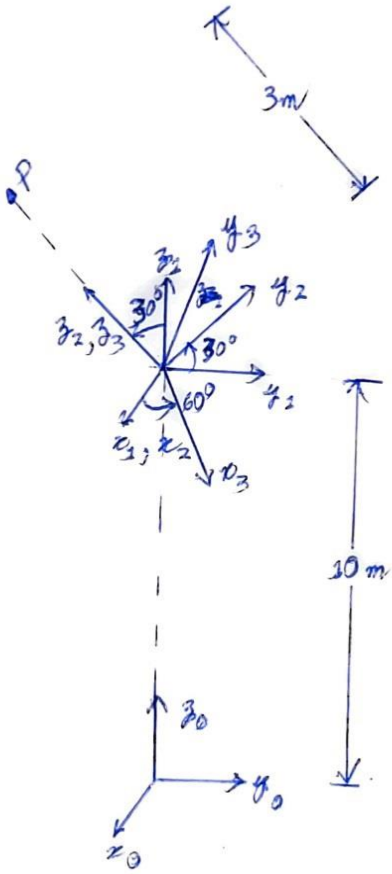
$$P_{drone} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

∴ Position of obstacle w. r. t. base frame,

$$P_{base} = T_{init}\ R_1\ R_2\ P_{drone}$$

$$\Rightarrow P_{base} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{3}/2 & -1/2 & 0 \\ 0 & 1/2 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & -\sqrt{3}/2 & 0 & 0 \\ \sqrt{3}/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 3 \\ 1 \end{bmatrix}$$

$$\therefore P_{base} = \begin{bmatrix} 0 \\ -1.5 \\ 12.598 \\ 1 \end{bmatrix} \quad \underline{\text{Ans}}$$



Question 6

Q6

1. Planetary Gearbox

Pros:

- Offers high torque in a compact size.
- They have low backlash and high efficiency.
- Can handle both low and high speed applications.

Cons:

- Complicated design and manufacturing.
- They are costly.
- They dissipate large amounts of heat, so may need cooling.

Usage:

- Robotic arms.
- Humanoid robots.

2. Spur Gearbox

Pros:

- Simple and cost-effective design.
- They have high efficiency.

Cons:

- Larger in size than other types of gearboxes with similar torque ratings.
- Cannot handle very high torques.

Usage:

- Conveyor belts.
- Simple robotic vehicles.

3. Cycloidal Gearbox

Pros:

- Can transmit high amounts of torque.
- Can handle shock loads.
- Relatively compact in size.

Cons:

- Lower efficiency than planetary and spur gearboxes.
- Balancing is difficult.

Usage:

- Industrial automation.
- Robotic manipulators.

4. Worm Gearbox

Pros:

- High reduction ratios.
- Self-locking property.
- Precise linear motion possible.

Cons:

- Lower efficiency due to sliding contact.
- Generate more heat.

Usage:

- Robot ~~usage~~ joints.
- Automatic doors.

Gearboxes In Drones

Gearboxes are frequently used in drones, Drones are usually fitted with brushless DC motors (BLDC), which have high speed but low torque. For the propellers to be able to lift the drone and control it, they should produce enough thrust. To achieve this, a gearbox can be used to increase output torque while decreasing rotational speed. Hence the motor can run at a speed where it has the highest efficiency, while also providing adequate thrust. Moreover, the gearbox allows silent operation and more efficiency.

Question 7

Q7 Jacobian of SCARA manipulator

From DH coordinate frame assignment, the A-matrices

$$A_1 = \begin{bmatrix} \cos q_1 & -\sin q_1 & 0 & l_1 \cos q_1 \\ \sin q_1 & \cos q_1 & 0 & l_1 \sin q_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos q_2 & \sin q_2 & 0 & l_2 \cos q_2 \\ \sin q_2 & -\cos q_2 & 0 & l_2 \sin q_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} \cos q_4 & -\sin q_4 & 0 & 0 \\ \sin q_4 & \cos q_4 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Here, joints 1, 2, and 4 are revolute and joint 3 is prismatic, and $o_4 - o_3$ is parallel to z_3.

$$\therefore \; z_3 \times (o_4 - o_3) = 0$$

The Jacobian is of the form:

$$J = \begin{bmatrix} z_0 \times (o_4 - o_0) & z_1 \times (o_4 - o_1) & z_2 & 0 \\ z_0 & z_1 & 0 & z_3 \end{bmatrix}$$

Here; $o_1 = \begin{bmatrix} l_1 \cos q_1 \\ l_1 \sin q_1 \\ 0 \end{bmatrix}$, $o_2 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ 0 \end{bmatrix}$

$$o_4 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ d_3 - d_4 \end{bmatrix}$$

Then, $z_0 = z_1 = \hat{k}$, $z_2 = z_3 = -\hat{k}$

$\therefore$ Jacobian of SCARA manipulator is:

$$J = \begin{bmatrix} -l_1 \sin q_1 - l_2 \sin(q_1 + q_2) & -l_2 \sin(q_1 + q_2) & 0 & 0 \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) & l_2 \cos(q_1 + q_2) & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 \end{bmatrix}$$

Question 8

Source Code: (Link to GitHub)

```
import numpy as np

"""
Program to calculate the manipulator Jacobian for an RRP SCARA robot.

Parameters:
q1 = Angle of Joint 1
q2 = Angle of Joint 2
d3 = Extension length of the robot arm
l1 = Length of Link 1
l2 = Length of Link 2
"""

def scara_jacobian(q1, q2, l1, l2, d3):
    jacobian = np.array([
                           [-l1 * np.sin(q1) - l2 * np.sin(q1 + q2), -l2 *
np.sin(q1 + q2), 0],
                           [l1 * np.cos(q1) + l2 * np.cos(q1 + q2), l2 *
np.cos(q1 + q2), 0],
                           [0, 0, -d3],
                           [0, 0, 0],
                           [0, 0, 0],
                           [1, 1, 1]
                        ])
    return jacobian

l1 = float(input("Enter length of Link 1 (m): "))
l2 = float(input("Enter length of Link 2 (m): "))
q1 = np.radians(float(input("Enter angle of Joint 1 (deg): ")))
q2 = np.radians(float(input("Enter angle of Joint 2 (deg): ")))
d3 = float(input("Enter extension length (m): "))

jacobian = scara_jacobian(q1, q2, l1, l2, d3)

print("\nJacobian matrix for SCARA robot:")
print(jacobian)

input()
```

Output:

```
Enter length of Link 1 (m): 2
Enter length of Link 2 (m): 2
Enter angle of Joint 1 (deg): 30
Enter angle of Joint 2 (deg): 45
Enter extension length (m): 1

Jacobian matrix for SCARA robot:
[[-2.93185165 -1.93185165  0.        ]
 [ 2.2496889   0.51763809  0.        ]
 [ 0.          0.         -1.        ]
 [ 0.          0.          0.        ]
 [ 0.          0.          0.        ]
 [ 1.          1.          1.        ]]
```

Question 9

Q9 Jacobian of RRR Planar manipulator

Here, the Jacobian is of the form

$$J = \begin{bmatrix} z_0 \times (o_3 - o_0) & z_1 \times (o_3 - o_1) & z_2 \times (o_3 - o_2) \\ z_0 & z_1 & z_2 \end{bmatrix}$$

Here, $o_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$, $o_1 = \begin{bmatrix} l_1 \cos q_1 \\ l_1 \sin q_1 \\ 0 \end{bmatrix}$

$$o_2 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) \\ l_1 \sin q_1 + l_2 \sin(q_1 + q_2) \\ 0 \end{bmatrix}$$

$$o_3 = \begin{bmatrix} l_1 \cos q_1 + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \\ l_1 \cos q_1 + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \\ 0 \end{bmatrix}$$

Unit vectors,

$$z_0 = z_1 = z_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

as they're all parallel to z_0 axis.

$\therefore$ Jacobian of RRR Planar manipulator is :

$$J = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} - l_3 s_{123} & -l_2 s_{12} - l_3 s_{123} & -l_3 s_{123} \\ l_1 c_1 + l_2 c_{12} + l_3 s_{123} & l_2 c_{12} + l_3 c_{123} & l_3 c_{123} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Question 10

Source Code: (Link to GitHub)

```
import numpy as np

"""
Program to calculate the manipulator Jacobian for an RRR Planar robot.

Parameters:
q1 = Angle of Joint 1
q2 = Angle of Joint 2
q3 = Angle of Joint 3
l1 = Length of Link 1
l2 = Length of Link 2
l3 = Length of Link 3
"""

def rrr_planar_jacobian(q1, q2, q3, l1, l2, l3):
    jacobian = np.array([
                            [-l1 * np.sin(q1) - l2 * np.sin(q1 + q2) - l3 *
np.sin(q1 + q2 + q3), -l2 * np.sin(q1 + q2) - l3 * np.sin(q1 + q2 + q3), -l3 *
np.sin(q1 + q2 + q3)],
                            [l1 * np.cos(q1) + l2 * np.cos(q1 + q2) + l3 *
np.cos(q1 + q2 + q3), l2 * np.cos(q1 + q2) + l3 * np.cos(q1 + q2 + q3), l3 *
np.cos(q1 + q2 + q3)],
                            [0, 0, 0],
                            [0, 0, 0],
                            [0, 0, 0],
                            [1, 1, 1]
                         ])
    return jacobian

l1 = float(input("Enter length of Link 1 (m): "))
l2 = float(input("Enter length of Link 2 (m): "))
l3 = float(input("Enter length of Link 3 (m): "))
q1 = np.radians(float(input("Enter angle of Joint 1 (deg): ")))
q2 = np.radians(float(input("Enter angle of Joint 2 (deg): ")))
q3 = np.radians(float(input("Enter angle of Joint 3 (deg): ")))


jacobian = rrr_planar_jacobian(q1, q2, q3, l1, l2, l3)

print("\nJacobian matrix for RRR Planar robot:")
print(jacobian)

input()
```

Output:

```
Enter length of Link 1 (m): 2
Enter length of Link 2 (m): 2
Enter length of Link 3 (m): 2
Enter angle of Joint 1 (deg): 30
Enter angle of Joint 2 (deg): 45
Enter angle of Joint 3 (deg): 60

Jacobian matrix for RRR Planar robot:
[[-4.34606521 -3.34606521 -1.41421356]
 [ 0.83547534 -0.89657547 -1.41421356]
 [ 0.          0.          0.        ]
 [ 0.          0.          0.        ]
 [ 0.          0.          0.        ]
 [ 1.          1.          1.        ]]
```

REFERENCES

1. Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control*. https://www.kramirez.net/Robotica/Tareas/Kinematics.pdf

2. Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo. *Robots: Modelling, Planning and Control*. http://people.disim.univaq.it/~costanzo.manes/EDU_stuff/Robotics_Modelling,%20Planning%20and%20Control_Sciavicco_extract.pdf

3. What is Planetary Gearbox and How Does it Works? : https://www.linquip.com/blog/what-is-planetary-gearbox/

4. The Pros and Cons of Spur Gears: https://fg-machine.com/blog/the-pros-and-cons-of-spur-gears/

5. Cycloidal drive: https://en.wikipedia.org/wiki/Cycloidal_drive

6. Cycloidal Gearbox: https://www.autoprotips.com/cycloidal-gearbox/

7. Advantages and disadvantages of a worm drive: https://clr.es/blog/en/advantages-and-disadvantages-of-a-worm-drive/

8. ChatGPT: https://chat.openai.com/