Robótica: Atividade 02 - Prática

Aluno: Aldemir Melo Rocha Filho **Matricula:**17212086
Aluno: Sandoval da Silva Almeida Junior **Matricula:**18210505
Aluno: Tayco Murilo Santos Rodrigues **Matricula:**17211250

Descrição da atividade:

Fazer cinemática direta do SCARA, considerando *offset* $= 0,1m$; $l_1 = 0,475m$, $l_2 = 0,4m$ e $0m \leq d_4 \leq 0,1m$.

1. Esquemático - SCARA:

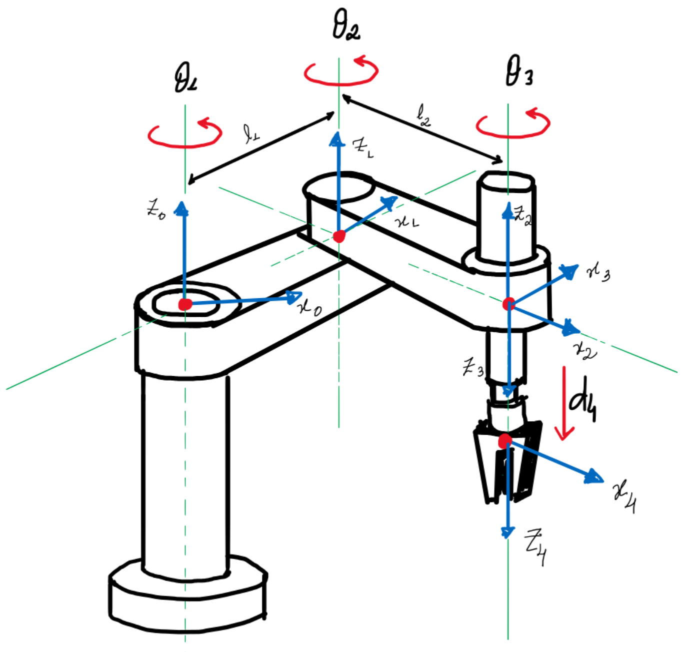

Figura 1: Esquemático - SCARA.

2. Tabela de Parâmetros DH:

Baseado no esquemático presente na Figura 1, podemos montar a seguinte tabela de parâmetros DH:

i	θ	d_i	a_i	α
1	θ_1	0	l_1	0
2	θ_2	0	l_2	0
3	θ_3	0	0	π
4	0	d_4	0	0

Tabela 1: Parâmetros DH para a Figura 1

3. Cálculo das matrizes de transformação até o efetuador:

Partindo da Tabela 1 podemos montar as seguintes matrizes de transformações até o efetuador:

$$^{1}_{0}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & l_1 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{2}_{1}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & l_2 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{3}_{2}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\theta_3 & s\theta_3 & 0 & 0 \\ s\theta_3 & -c\theta_3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{4}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dessa forma podemos obter a matriz de transformação que leva de 0 até 4 usando a seguinte operação:

$$^{4}_{0}T =^{1}_{0} T \cdot^{2}_{1} T \cdot^{3}_{2} T \cdot^{4}_{3} T \tag{1}$$

Logo, temos de (1):

$$^{4}_{0}T = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta 3) & \sin(\theta_1+\theta_2+\theta_3) & 0 & l_1\cos(\theta_1)+l_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2+\theta_3) & -\cos(\theta_1+\theta_2+\theta 3) & 0 & l_1\sin(\theta_1)+l_2\sin(\theta_1+\theta_2) \\ 0 & 0 & -1 & -d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Código da função de cinemática direta fkine:

Partindo do resultado obtido em **3.**, podemos definir a função `fkine( )` em `Python` da seguinte forma:

Primeiro, definimos as importações necessárias:

```
import numpy as np
import math as m
import roboticstoolbox as rtb
```

Por fim, definimos a função `fkine( )`:

```python
def fkine(theta_1, theta_2, theta_3, d_4):
  return np.array([
  [m.cos(theta_3+theta_2+theta_1),m.sin(theta_3+theta_2+theta_1),0,l1*m.cos(theta_1)+l2*m.cos(theta_2+theta_1)],
  [m.sin(theta_3+theta_2+theta_1), -m.cos(theta_3+theta_2+theta_1), 0, l1*m.sin(theta_1)+l2*m.sin(theta_2+theta_1)],
  [0, 0, -1, -d_4],
  [0, 0, 0, 1]])
```

5. Print do objeto SerialLink gerado na `robotics toolbox`:

Primeiro montamos o objeto "SCARA":

```python
1     t01 = rtb.robot.DHLink(a = l1, offset=0)
2     t12 = rtb.robot.DHLink(a = l2, offset=0)
3     t23 = rtb.robot.DHLink(alpha = m.pi, offset=0)
4     t34 = rtb.robot.DHLink(sigma = 1, qlim=[0, 0.1])
5     scara = rtb.robot.DHRobot([t01, t12, t23, t34], name = 'SCARA')
6     print(scara)
```

A saída da `linha 6` do código acima pode ser observada a seguir.

θ_j	d_j	a_j	α_j	q^-	q^+
q1	0	0.475	0.0°	-180.0°	180.0°
q2	0	0.4	0.0°	-180.0°	180.0°
q3	0	0	180.0°	-180.0°	180.0°
0.0°	q4	0	0.0°	0.0	0.1

Figura 2: Objeto SerialLink gerado na robotics toolbox.

6. Comparação do resultado da função fkine implementada e a função fkine dentro do objeto SerialLink:

6.1. Para os valores $\theta_1 = 0$, $\theta_2 = 0$, $\theta_3 = 0$ e $d_4 = 0$

```
[ 1.     0.     0.     0.875]
[ 0.    -1.     0.     0.   ]
[ 0.     0.    -1.     0.   ]
[ 0.     0.     0.     1.   ]
```

Figura 3: Saída da função `fkine( )` implementada.

```
1        0        0        0.875
0       -1        0        0
0        0       -1        0
0        0        0        1
```

Figura 4: Saída da função `fkine( )`
dentro do objeto SerialLink.

6.2. Para os valores $\theta_1 = \frac{\pi}{2}$, $\theta_2 = -\frac{\pi}{2}$, $\theta_3 = 0$ e $d_4 = 0$

```
[-1.0000000e+00  1.2246468e-16  0.0000000e+00 -4.0000000e-01]
[ 1.2246468e-16  1.0000000e+00  0.0000000e+00  4.7500000e-01]
[ 0.0000000e+00  0.0000000e+00 -1.0000000e+00  0.0000000e+00]
[ 0.0000000e+00  0.0000000e+00  0.0000000e+00  1.0000000e+00]
```

Figura 5: Saída da função `fkine( )`
implementada.

```
-1        0        0       -0.4
 0        1        0        0.475
 0        0       -1        0
 0        0        0        1
```

Figura 6: Saída da função `fkine( )`
dentro do objeto SerialLink.

6.3. Para os valores $\theta_1 = \frac{\pi}{2}$, $\theta_2 = -\frac{\pi}{2}$, $\theta_3 = 0$ e $d_4 = 0.05$

```
[ 1.     0.     0.     0.4  ]
[ 0.    -1.     0.     0.475]
[ 0.     0.    -1.    -0.05 ]
[ 0.     0.     0.     1.   ]
```

Figura 7: Saída da função `fkine( )`
implementada.

```
1        0        0        0.4
0       -1        0        0.475
0        0       -1       -0.05
0        0        0        1
```

Figura 8: Saída da função `fkine( )`
dentro do objeto SerialLink.

7. Implementação da função da cinemática inversa para o SCARA, considerando como entrada (x, y, z, ϕ) e como saída $(\theta_1, \theta_2, \theta_3, d_4)$:

A seguinte associação foi realizada para a implementação do código:

$$
{}^4_0T = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & x \\ \sin\psi & \cos\psi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

A implementação da cinemática inversa pode ser visualizada abaixo:

```python
def invFkine(Matrix, l1, l2, theta1_ref, theta2_ref, theta3_ref):
    # x, y, z = Matrix[0:3, -1]
    print(Matrix)
    x, y = Matrix[0:2, -1]
    z = 0
    phi = np.arctan2(Matrix[1, 0], Matrix[0, 0])
    if z != 0:
        raise Exception()
    c2 = (x ** 2 + y ** 2 - l1 ** 2 - l2 ** 2) / (2 * l1 * l2)
    if c2 > 1 or c2 < -1:
        raise Exception()
    s2_1 = np.sqrt(1 - (c2 ** 2))
    s2_2 = -np.sqrt(1 - (c2 ** 2))
    theta2_1 = np.arctan2(s2_1, c2)
    theta2_2 = np.arctan2(s2_2, c2)

    k1 = l2 * c2 + l1
    k2_1 = l2 * s2_1
    k2_2 = l2 * s2_2
    theta1_1 = np.arctan2(y, x) - np.arctan2(k2_1, k1)
    theta1_2 = np.arctan2(y, x) - np.arctan2(k2_2, k1)

    theta3_1 = phi - theta1_1 - theta2_1
    theta3_2 = phi - theta1_2 - theta2_2

    d4 = -Matrix[2, -1]

    opt1 = (theta1_ref - theta1_1)**2 + (theta2_ref - theta2_1)**2 + (theta3_ref - theta3_1)**2
    opt2 = (theta1_ref - theta1_2)**2 + (theta2_ref - theta2_2)**2 + (theta3_ref - theta3_2)**2

    if opt1 <= opt2:
        return theta1_1, theta2_1, theta3_1, d4
    return theta1_2, theta2_2, theta3_2, d4
```

Referências:

[1] **CoppeliaSim User Manual.** Disponível em: <https://www.coppeliarobotics.com/helpFiles/>.

[2] CRAIG, J. J. **Introduction to robotics : mechanics and control.** Upper Saddle River, Nj: Pearson, 2018.