



Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

IA903 – Introdução à Robótica Móvel

18 de setembro de 2024

Docente: Eric Rohmer

Discente:

– Gabriel Toffanetto França da Rocha – 289320

Exercise 1: Kinematics of a single robot leg

Sumário

1	2
2	3
3	4
4	5
5	6
Anexos	9

1

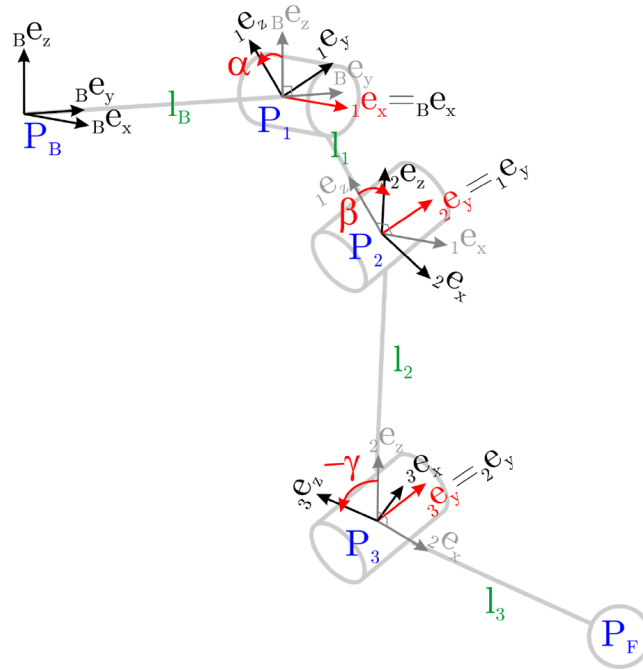


Figura 1: Sistemas de coordenadas da perna robótica.

Uma vez dados os sistemas de coordenadas da perna robótica, exibidos na Figura 1, é possível observar que o *frame* $\{P_1\}$ apresenta rotação em torno do eixo x , o *frame* $\{P_2\}$ em torno do eixo y , assim como o *frame* $\{P_3\}$. Considerando o vetor de coordenadas generalizadas, $\mathbf{q} = [\alpha \ \beta \ \gamma]^T$, sendo α , β e γ respectivamente os ângulos de rotação dos sistemas de coordenadas $\{P_1\}$, $\{P_2\}$ e $\{P_3\}$.

Com isso, têm-se que a rotação entre os sistemas de coordenadas da base e da junta 1 é dado pela matriz (1), a rotação entre o *frame* $\{P_1\}$ e o *frame* $\{P_2\}$ é dada por (2) e a rotação entre o sistema de coordenadas da junta 2 e o sistema da junta 3 é dado pela matriz (3).

$$\mathbf{R}_{B1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix} \quad (1)$$

$$\mathbf{R}_{12} = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \quad (2)$$

$$\mathbf{R}_{23} = \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \quad (3)$$

2

Os sistemas de coordenadas $\{P_B\}$, $\{P_1\}$, $\{P_2\}$, $\{P_3\}$ e $\{P_F\}$ da perna robótica estão posicionados entre eles em distâncias l_B , l_1 , l_2 , l_3 e l_F , respectivamente. Considerando todas essas distâncias iguais e unitárias, obtêm-se os vetores de translação entre os referenciais mostrados em (4), (5), (6) e (7).

$${}_B\mathbf{r}_{B1} = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T; \quad (4)$$

$${}_1\mathbf{r}_{12} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T; \quad (5)$$

$${}_2\mathbf{r}_{23} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T; \quad (6)$$

$${}_3\mathbf{r}_{3F} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T; \quad (7)$$

Permitindo a transformação entre os sistemas de coordenada, incluindo rotação e translação em uma só operação, a utilização de transformações homogêneas se faz interessante, sendo a mesma enunciada em (8).

$$\mathbf{H} = \begin{bmatrix} \mathbf{R}_{[3 \times 3]} & \mathbf{r}_{[3 \times 1]} \\ 0 & 1 \end{bmatrix}_{[4 \times 4]} \quad (8)$$

Dessa forma, a transformação homogênea entre o *frame* $\{P_B\}$ e $\{P_1\}$ é dada por (9), entre $\{P_1\}$ e $\{P_2\}$ é dada por (10), entre $\{P_2\}$ e $\{P_3\}$ é dada por (11) e entre $\{P_3\}$ e $\{P_F\}$ é dada por (12),

$${}_B\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 1 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (9)$$

$${}_1\mathbf{H}_2 = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$${}_2\mathbf{H}_3 = \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

$${}_3\mathbf{H}_F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

Para se obter a transformação homogênea entre *frames* de um conjunto consecutivo de transformações de referencial, basta multiplicar as matrizes de transformação homogênea, como feito em (13), obtendo assim a transformação da base para a terceira junta da perna robótica.

$${}_B\mathbf{H}_3 = {}_B\mathbf{H}_{1\ 1} \mathbf{H}_{2\ 2} \mathbf{H}_3 = \begin{bmatrix} c(\beta + \gamma) & 0 & s(\beta + \gamma) & -s(\beta) \\ s(\beta + \gamma)s(\alpha) & c(\alpha) & -c(\beta + \gamma)s(\alpha) & s(\alpha) + c(\beta)s(\alpha) + 1 \\ -s(\beta + \gamma)c(\alpha) & s(\alpha) & c(\beta + \gamma)c(\alpha) & -c(\alpha)(c(\beta) + 1) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$\begin{aligned} {}_B\mathbf{H}_F &= {}_B\mathbf{H}_{1\ 1} \mathbf{H}_{2\ 2} \mathbf{H}_3 {}_3\mathbf{H}_F \\ &= \begin{bmatrix} c(\beta + \gamma) & 0 & s(\beta + \gamma) & -s(\beta + \gamma) - s(\beta) \\ s(\beta + \gamma)s(\alpha) & c(\alpha) & -c(\beta + \gamma)s(\alpha) & s(\alpha)[1 + c(\beta) + c(\beta)c(\gamma) - s(\beta)s(\gamma)] + 1 \\ -s(\beta + \gamma)c(\alpha) & s(\alpha) & c(\beta + \gamma)c(\alpha) & -c(\alpha)[c(\beta + \gamma) + c(\beta) + 1] \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (14)$$

Por meio da transformação de referencial entre o *frame* da base e da ferramenta, (14), é possível expressar o vetor da base para a ferramenta, pré-multiplicando o vetor da origem da base (${}_B\mathbf{r}_{BB} = [0\ 0\ 0]^T_B$) pela transformação homogênea em questão.

$${}_B\mathbf{r}_{BF} = {}_B\mathbf{H}_F {}_B\mathbf{r}_{BB} = \begin{bmatrix} -\sin(\beta + \gamma) - \sin(\beta) \\ \sin(\alpha)[\cos(\beta + \gamma) + \cos(\beta) + 1] + 1 \\ -\cos(\alpha)[\cos(\beta + \gamma) + \cos(\beta) + 1] \end{bmatrix} \quad (15)$$

3

$${}_B\mathbf{J}_{BF} = \frac{\partial {}_B\mathbf{r}_{BF}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial {}_B\mathbf{r}_{BF}}{\partial q_1} & \frac{\partial {}_B\mathbf{r}_{BF}}{\partial q_2} & \frac{\partial {}_B\mathbf{r}_{BF}}{\partial q_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial {}_B r_{xBF}}{\partial q_1} & \frac{\partial {}_B r_{xBF}}{\partial q_2} & \frac{\partial {}_B r_{xBF}}{\partial q_3} \\ \frac{\partial {}_B r_{yBF}}{\partial q_1} & \frac{\partial {}_B r_{yBF}}{\partial q_2} & \frac{\partial {}_B r_{yBF}}{\partial q_3} \\ \frac{\partial {}_B r_{zBF}}{\partial q_1} & \frac{\partial {}_B r_{zBF}}{\partial q_2} & \frac{\partial {}_B r_{zBF}}{\partial q_3} \end{bmatrix} \quad (16)$$

$${}_B\mathbf{J}_{BF} = \begin{bmatrix} 0 & -c(\beta + \gamma) - c(\beta) & -c(\beta + \gamma) \\ c(\alpha)[c(\beta + \gamma) + c(\beta) + 1] & s(\alpha)[-s(\beta + \gamma) - s(\beta)] & -s(\alpha)s(\beta + \gamma) \\ s(\alpha)[c(\beta + \gamma) + c(\beta) + 1] & c(\alpha)[s(\beta + \gamma) + s(\beta)] & c(\alpha)s(\beta + \gamma) \end{bmatrix} \quad (17)$$

Onde $s(\cdot) = \sin(\cdot)$ e $c(\cdot) = \cos(\cdot)$.

Considerando uma posição inicial para a perna robótica, (18), e uma velocidade para a ferramenta (19), obtém-se um jacobiano instantâneo igual a (20).

$$\mathbf{q}_i = \begin{bmatrix} 0^\circ & 60^\circ & -120^\circ \end{bmatrix}^T \quad (18)$$

$$\mathbf{v} = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T \text{ (m/s)} \quad (19)$$

$$\mathbf{J}_{BF} = \begin{bmatrix} 0 & -1 & -0,5 \\ 2 & 0 & 0 \\ 0 & 0 & 0,866 \end{bmatrix} \quad (20)$$

Com isso, uma vez que o jacobiano relaciona a velocidade no espaço cartesiano com a velocidade no espaço das juntas, $\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$, pode-se obter a velocidade instantânea das juntas da perna robótica, $d\mathbf{q}$, por meio de (21), para a velocidade desejada do efetuador e a posição inicial das juntas.

$$d\mathbf{q} = \mathbf{J}_{BF}^+ \mathbf{v} = \begin{bmatrix} 0 \\ -0,5774 \\ 1,1547 \end{bmatrix} \quad (21)$$

4

Para realizar o posicionamento do manipulador em um ponto desejado, deve-se fazer o uso da cinemática inversa, porém, sua obtenção de forma analítica é um procedimento por muitas vezes complexo e massante. Desas forma, pode-se fazer o uso de métodos numéricos, como o de Newton, para convergir para a posição desejada do efetuador em n passos, até que o erro ζ seja menor que o desejado.

$$\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}^+(\mathbf{r}^{goal} - \mathbf{r}^i) \quad (22)$$

Por meio do Algoritmo 1, é feita a resolução da cinemática inversa de forma numérica, atualizando a posição da perna robótica a cada interação para acompanhamento da convergência do algoritmo.

Algoritmo 1 Cinemática inversa numérica

- | | |
|---|---|
| 1: $\zeta \leftarrow 0.01$;
2: $\mathbf{q}^i \leftarrow \mathbf{q}^0$;
3: while $\ \mathbf{r}^{goal} - \mathbf{r}^i\ _2 \geq \zeta$ do
4: $\mathbf{q}^{i+1} \leftarrow \mathbf{q}^i + \text{pinv}(\mathbf{J}(\mathbf{q}^i)) * (\mathbf{r}^{goal} - \mathbf{r}^i)$;
UPDATEROBOTLEGPOSITION(\mathbf{q}^{i+1});
5: $\mathbf{q}^i \leftarrow \mathbf{q}^{i+1}$;
6: end while
7: $\mathbf{q}^{goal} \leftarrow \mathbf{q}^i$ | <div style="text-align: right;">▷ Stop error</div> <div style="text-align: right;">▷ While solution does not converge...</div> <div style="text-align: right;">▷ Generalized coordinates for the solution</div> |
|---|---|
-

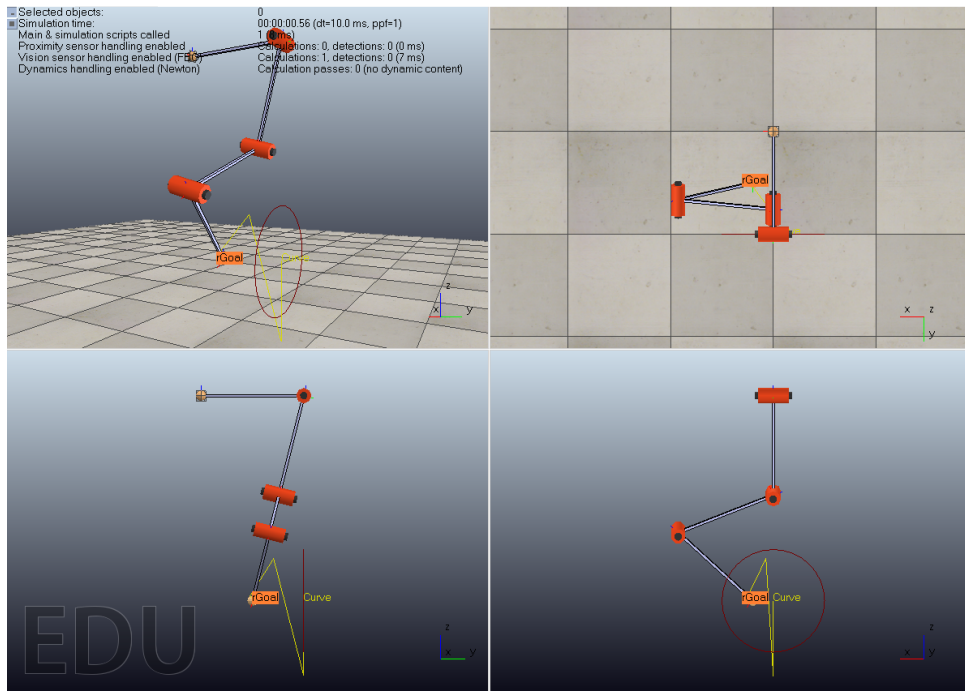


Figura 2: Posicionamento da perna robótica em uma posição desejada por meio da cinemática inversa numérica.

A convergência se deu em três passos, como mostrado no *log* abaixo.

i = 1 | $\mathbf{rGoal} = [0.200 \ 0.500 \ -2.000]$; $\mathbf{r} = [0.000 \ 1.000 \ -2.732]$; error = 0.478
i = 2 | $\mathbf{rGoal} = [0.200 \ 0.500 \ -2.000]$; $\mathbf{r} = [0.071 \ 0.706 \ -1.589]$; error = 0.072
i = 3 | $\mathbf{rGoal} = [0.200 \ 0.500 \ -2.000]$; $\mathbf{r} = [0.249 \ 0.476 \ -1.953]$; error = 0.003

5

Uma vez dada os *waypoints* da trajetória a ser seguida pela perna robótica, faz-se o uso novamente da relação (21) para obtenção da velocidade dos atuadores. Porém, dessa vez \mathbf{v} será dado com base no erro de posição entre o efetuador da perna robótica e o ponto da trajetória atual, multiplicado por um ganho proporcional, como em (23).

$$\mathbf{v} = K_p(\mathbf{r}^{goal} - \mathbf{r}^i) \quad (23)$$

Considerando um ganho proporcional igual $K_p = 10$, obteve-se um bom rastreo da trajetória, com um transitório consideravelmente curto, e boa convergência ao *setpoint*, como pode ser visto nas Figuras 3 e 4.

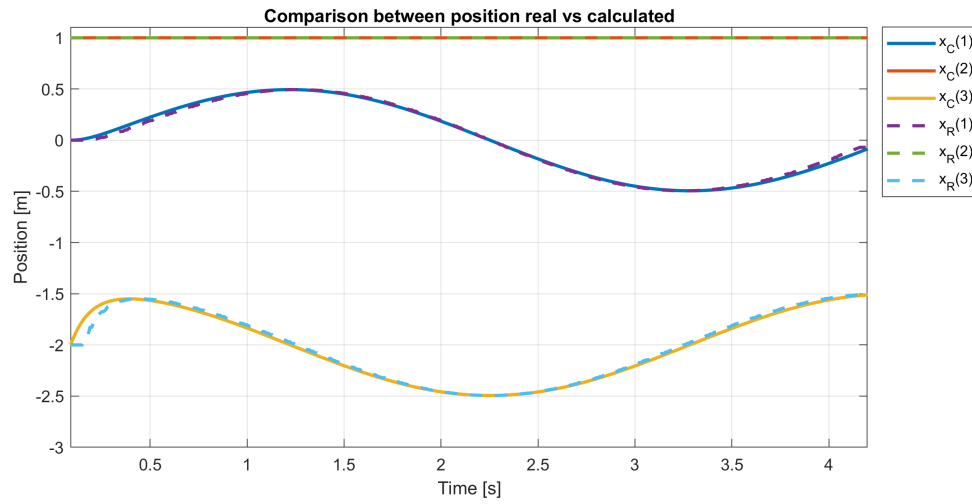


Figura 3: Rastreio da posição do efetuador da perna robótica em x , y e z .

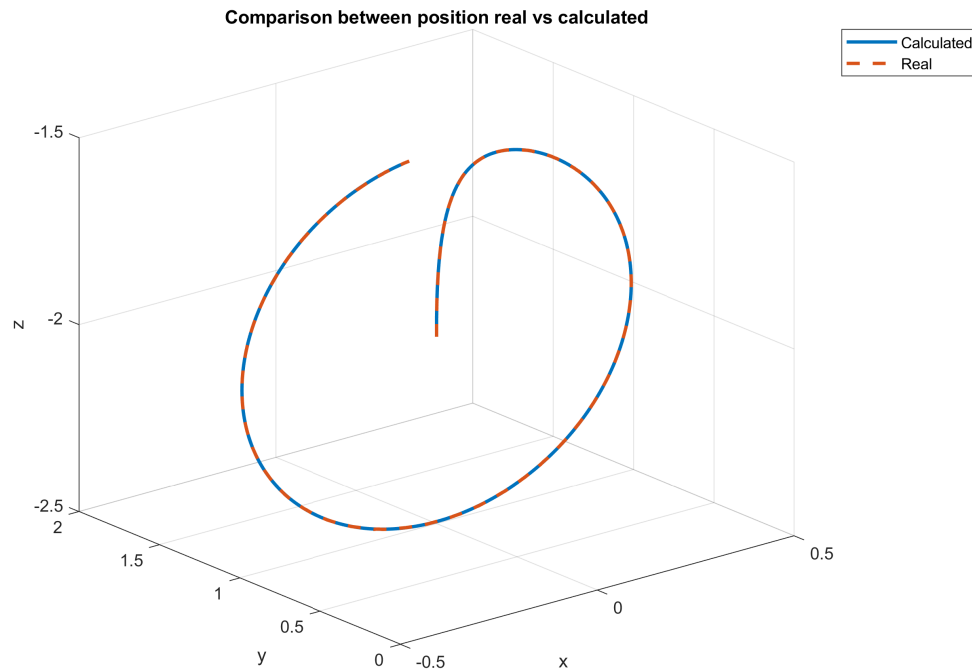


Figura 4: Rastreio da posição do efetuador da perna robótica no espaço.

Por fim, é possível observar na Figura 5 a trajetória realizada pela perna robótica partindo da posição inicial no centro do círculo.

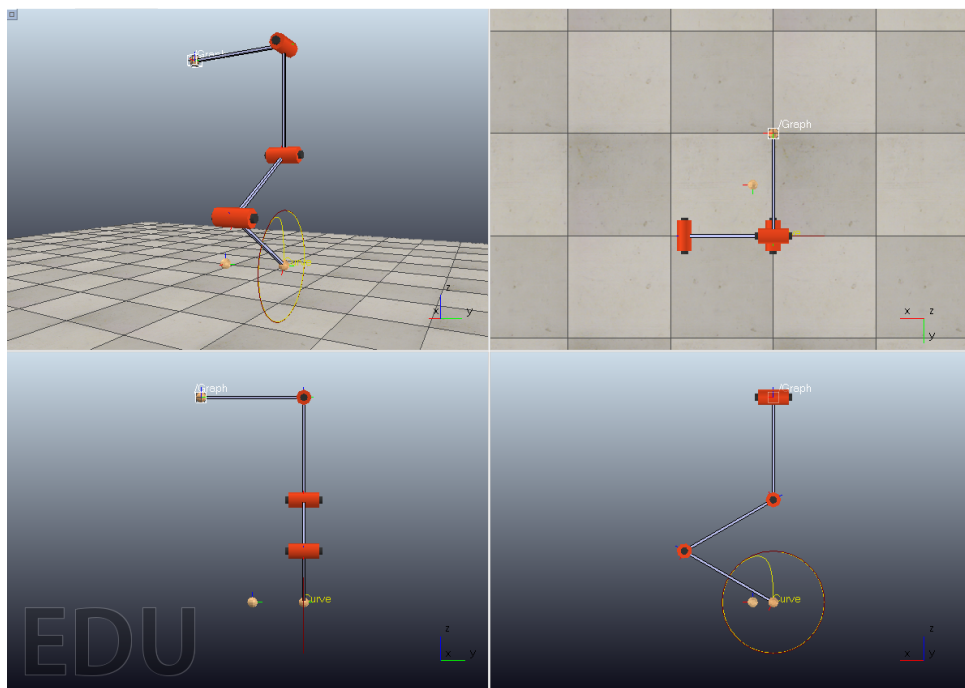


Figura 5: Trajetória realizada pela perna robótica.

Apêndices

Código fonte – Exercício 1

```
1 clc; clear all;
2
3 syms alpha beta gamma real
4
5 % write down the rotation matrices using the symbolic parameters alpha, beta
  , gamma
6
7 % Rotation around 1_e_x
8 R_B1 = [1 0          0; ...
9         0 cos(alpha) -sin(alpha); ...
10        0 sin(alpha) cos(alpha) ];
11
12 % Rotation around 2_e_y
13 R_12 = [cos(beta)  0 sin(beta); ...
14         0          1 0; ...
15        -sin(beta)  0 cos(beta)];
16
17 % Rotation around 3_e_y
18 R_23 = [cos(gamma)  0 sin(gamma); ...
19         0          1 0; ...
20        -sin(gamma)  0 cos(gamma)];
21
22 % answers
23 valid
```

Código fonte – Exercício 2

```
1 clc; clear all
2
3 syms alpha beta gamma
4
5 lb = 1; l1 = 1; l2 = 1; l3 = 1;
6
7 % rotational matrices calculated in previous problem set
8 R_B1 = [1,0,0;0,cos(alpha),-sin(alpha);0,sin(alpha),cos(alpha)];
9 R_12 = [cos(beta),0,sin(beta);0,1,0;-sin(beta),0,cos(beta)];
10 R_23 = [cos(gamma),0,sin(gamma);0,1,0;-sin(gamma),0,cos(gamma)];
11
12 % write down the 3x1 relative position vectors for link length l_i=1
13 r_B1_inB = [0; 1; 0];
14 r_12_in1 = [0; 0; -1];
15 r_23_in2 = [0; 0; -1];
16 r_3F_in3 = [0; 0; -1];
17
18 % write down the homogeneous transformation matrices
19 H_B1 = [R_B1 r_B1_inB ; [0 0 0 1]]
20 H_12 = [R_12 r_12_in1 ; [0 0 0 1]]
21 H_23 = [R_23 r_23_in2 ; [0 0 0 1]]
22 H_3F = [eye(3) r_3F_in3 ; [0 0 0 1]]
23
24 % create the cumulative transformation matrix
25 H_B3 = H_B1*H_12*H_23*H_3F
26
27 % find the foot point position vector
28 r_BF_inB = H_B3*[0; 0; 0; 1]
29 r_BF_inB = r_BF_inB(1:3)
30
31 valid
```

Código fonte – Exercício 3

```

1  clc; clear all;
2
3  syms alpha beta gamma real
4
5  q = [alpha;beta;gamma];
6
7  r_BF_inB = [...
8      - sin(beta + gamma) - sin(beta);...
9      sin(alpha)*(cos(beta + gamma) + cos(beta) + 1) + 1;...
10     -cos(alpha)*(cos(beta + gamma) + cos(beta) + 1)]
11
12
13 % determine the foot point Jacobian J_BF_inB=d(r_BF_inB)/dq
14 dr_dalpha = [0; ...
15              cos(alpha)*(cos(beta + gamma) + cos(beta) + 1); ...
16              sin(alpha)*(cos(beta + gamma) + cos(beta) + 1)];
17
18 dr_dbeta = [-cos(beta + gamma) + cos(beta); ...
19             sin(alpha)*(-sin(beta + gamma) -sin(beta)); ...
20             cos(alpha)*(+sin(beta + gamma) +sin(beta))];
21
22 dr_dgamma = [-cos(beta + gamma); ...
23              -sin(alpha)*sin(beta + gamma); ...
24              cos(alpha)*sin(beta + gamma)];
25
26 J_BF_inB = [dr_dalpha dr_dbeta dr_dgamma]
27
28
29 % what generalized velocity dq do you have to apply in a configuration q =
30 % [0;60;-120]
31 % to lift the foot in vertical direction with v = [0;0;-1m/s];
32 v = [0; 0; -1]
33 qi = [0; 60*(pi/180); -120*(pi/180)];
34
35 % Determine the numerical value of the foot point jacobian for initial joint
36 % angles qi
37
38 alpha = qi(1);
39 beta = qi(2);
40 gamma = qi(3);
41
42 dr_dalpha = [0; ...
43              cos(alpha)*(cos(beta + gamma) + cos(beta) + 1); ...
44              sin(alpha)*(cos(beta + gamma) + cos(beta) + 1)];
45
46 dr_dbeta = [-cos(beta + gamma) - cos(beta); ...
47             sin(alpha)*(-sin(beta + gamma) -sin(beta)); ...

```

```
46         cos(alpha)*(sin(beta + gamma) + sin(beta))]);
47
48 dr_dgamma = [-cos(beta + gamma); ...
49             -sin(alpha)*sin(beta + gamma); ...
50             cos(alpha)*sin(beta + gamma)];
51
52 JBF = [dr_dalpha dr_dbeta dr_dgamma]
53
54 % Determine the numerical value for dq
55 dq = inv(JBF)*v
56
57 valid
```

Código fonte – Exercício 4

```

1 % Make sure to have the simulation scene mooc_exercise.ttt running in V-REP!
2
3 % simulation setup, will add the matlab paths
4 connection = simulation_setup();
5
6 % the robot we want to interact with
7 robotNb = 0;
8
9 % open the connection
10 connection = simulation_openConnection(connection, robotNb);
11
12 % start simulation if not already started
13 simulation_start(connection);
14
15 vrep=connection.vrep;
16 % initialize connection
17 [err dt]=vrep.simxGetFloatingParameter(connection.clientID,vrep.
    sim_floatparam_simulation_time_step,vrep.simx_opmode_oneshot_wait);
18
19 % now enable stepped simulation mode:
20 simulation_setStepped(connection,true);
21
22 % given are the functions
23 %   r_BF_inB(alpha,beta,gamma) and
24 %   J_BF_inB(alpha,beta,gamma)
25 % for the foot position respectively Jacobian
26
27 r_BF_inB = @(alpha,beta,gamma)[...
28     -sin(beta + gamma) - sin(beta);...
29     sin(alpha)*(cos(beta + gamma) + cos(beta) + 1) + 1;...
30     -cos(alpha)*(cos(beta + gamma) + cos(beta) + 1)];
31
32 J_BF_inB = @(alpha,beta,gamma)[...
33     0, -cos(beta + gamma) - cos(beta), -cos(beta + gamma);...
34     cos(alpha)*(cos(beta + gamma) + cos(beta) + 1), -sin(alpha)*(sin(beta + gamma) + sin(beta)), -sin(beta + gamma)*sin(alpha);...
35     sin(alpha)*(cos(beta + gamma) + cos(beta) + 1), cos(alpha)*(sin(beta + gamma) + sin(beta)), sin(beta + gamma)*cos(alpha)];
36
37 % write an algorithm for the inverse kinematics problem to
38 % find the generalized coordinates q that gives the endeffector position
    rGoal =
39 % [0.2,0.5,-2]' and store it in qGoal
40 q0 = pi/180*([0,-30,60])';
41 updatePos(vrep,connection.clientID,q0)
42 pause(0.5)

```

```

43
44 rGoal = [0.2,0.5,-2]';
45
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47 %%% enter here your algorithm
48 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
49
50 k = 0;
51
52 zeta = 0.01; % Convergence error
53
54 while(norm(rGoal - r_BF_inB(q0(1),q0(2),q0(3))) > zeta)
55
56     r = r_BF_inB(q0(1),q0(2),q0(3));
57
58     qGoal = q0 + inv(J_BF_inB(q0(1),q0(2),q0(3)))*(rGoal - r);
59
60     updatePos(vrep,connection.clientID,qGoal)
61     pause(0.2)
62     q0 = qGoal;
63     k = k+1;
64
65     fprintf('i = %d | rGoal = [%0.3f %0.3f %0.3f]; r = [%0.3f %0.3f %0.3f];
66     error = %0.3f \n', ...
67         k, rGoal(1), rGoal(2), rGoal(3), r(1), r(2), r(3), norm(rGoal -
68         r_BF_inB(q0(1),q0(2),q0(3))))
69
70 end
71
72 % now disable stepped simulation mode:
73 simulation_setStepped(connection,false);
74
75
76 pause(5)
77
78 % stop the simulation
79 simulation_stop(connection);
80
81 % close the connection
82 simulation_closeConnection(connection);

```

Código fonte – Exercício 5

```

1 dqMatrix = [];
2
3 % Make sure to have the simulation scene mooc_exercise.ttt running in V-REP!
4
5 % simulation setup, will add the matlab paths
6 connection = simulation_setup();
7
8 % the robot we want to interact with
9 robotNb = 0;
10
11 % open the connection
12 connection = simulation_openConnection(connection, robotNb);
13
14 % start simulation if not already started
15 simulation_start(connection);
16
17 vrep=connection.vrep;
18
19 % initialize connection
20 % do a function simulation_getDt to do the following
21 dt= simulation_getDt(connection)
22
23 % now enable stepped simulation mode:
24 simulation_setStepped(connection,true);
25
26 % given are the functions
27 %   r_BF_inB(alpha,beta,gamma) and
28 %   J_BF_inB(alpha,beta,gamma)
29 % for the foot position respectively Jacobian
30
31 r_BF_inB = @(alpha,beta,gamma)[...
32     - sin(beta + gamma) - sin(beta);...
33     sin(alpha)*(cos(beta + gamma) + cos(beta) + 1) + 1;...
34     -cos(alpha)*(cos(beta + gamma) + cos(beta) + 1)];
35
36 J_BF_inB = @(alpha,beta,gamma)[...
37     0, - cos(beta + gamma) - cos(beta), -cos(beta
38 + gamma);...
39     cos(alpha)*(cos(beta + gamma) + cos(beta) + 1), -sin(alpha)*(sin(
40 beta + gamma) + sin(beta)), -sin(beta + gamma)*sin(alpha);...
41     sin(alpha)*(cos(beta + gamma) + cos(beta) + 1), cos(alpha)*(sin(
42 beta + gamma) + sin(beta)), sin(beta + gamma)*cos(alpha)];
43
44 % write an algorithm for the inverse differential kinematics problem to
45 % find the generalized velocities dq to follow a circle in the body xz
46 % plane
47 % around the start point rCenter with a radius of r=0.5 and a

```

```

44 % frequency of 1Hz. The start configuration is q = pi/180*([0,-60,120])'
45 q0 = pi/180*([0,-60,120])';
46 %q0 = pi/180*([0,-80,140])';
47
48 updatePos(vrep,connection.clientID,q0)
49 % pause(1.0)
50
51 dq0 = zeros(3,1);
52 rCenter = r_BF_inB(q0(1),q0(2),q0(3));
53 radius = 0.5;
54 f = 0.25;
55 rGoal = @(t) rCenter + radius*[sin(2*pi*f*t),0,cos(2*pi*f*t)]';
56 drGoal = @(t) 2*pi*f*radius*[cos(2*pi*f*t),0,-sin(2*pi*f*t)]';
57
58 % define here the time resolution
59 deltaT = dt;%0.01;
60 timeArr = 0:deltaT:1/f;
61
62 % q, r, and rGoal are stored for every point in time in the following
63 arrays
64 qArr = zeros(3,length(timeArr));
65 rArr = zeros(3,length(timeArr));
66 rGoalArr = zeros(3,length(timeArr));
67
68 q = q0;
69 dq = dq0;
70
71 kp = 10;
72
73 for i=1:length(timeArr)
74     t = timeArr(i);
75     % data logging, don't change this!
76     q = q+deltaT*dq;
77     qArr(:,i) = q;
78     rArr(:,i) = r_BF_inB(q(1),q(2),q(3));
79     rGoalArr(:,i) = rGoal(t);
80
81     % controller:
82     % step 1: create a simple p controller to determine the desired foot
83     % point velocity
84     v = kp*(rGoalArr(:,i) - rArr(:,i));
85     % step 2: perform inverse differential kinematics to calculate the
86     % gneralized velocities
87     dq = inv(J_BF_inB(q(1),q(2),q(3)))*v;
88
89     updateVels(vrep,connection.clientID,dq,q)
90
91 end

```



```
92
93 % now disable stepped simulation mode:
94 simulation_setStepped(connection,false);
95
96 % stop the simulation
97 simulation_stop(connection);
98
99 % close the connection
100 simulation_closeConnection(connection);
101
102
103
104 %% Set up the Import Options and import the data
105 opts = delimitedTextImportOptions("NumVariables", 4);
106
107 % Specify range and delimiter
108 opts.DataLines = [3, Inf];
109 opts.Delimiter = ",";
110
111 % Specify column names and types
112 opts.VariableNames = ["Times", "DataMeters", "Data0Meters", "Data1Meters"];
113 opts.VariableTypes = ["double", "double", "double", "double"];
114
115 % Specify file level properties
116 opts.ExtraColumnsRule = "ignore";
117 opts.EmptyLineRule = "read";
118
119 % Import the data
120 coppeliagraph = readtable("./coppelia_graph.csv", opts);
121
122 %% Convert to output type
123 coppeliagraph = table2array(coppeliagraph);
124
125 %% Clear temporary variables
126 clear opts
127
128 %% Plot
129
130 close all
131
132 n = linspace(coppeliagraph(1244,1),coppeliagraph(end-1,1),401);
133
134 figure(1)
135 plot(n,rArr(1,:), 'Linewidth',2)
136 hold on
137 plot(n,rArr(2,:), 'Linewidth',2)
138 plot(n,rArr(3,:), 'Linewidth',2)
139 plot(coppeliagraph(1244:end-1,1),coppeliagraph(1244:end-1,2), '--', 'Linewidth',2)
```

```
140 plot(coppeliagraph(1244:end-1,1),coppeliagraph(1244:end-1,3),'--','Linewidth',2)
141 plot(coppeliagraph(1244:end-1,1),coppeliagraph(1244:end-1,4),'--','Linewidth',2)
142 axis([0.09 4.2 -3 1.1])
143 xlabel('Time [s]')
144 ylabel('Position [m]')
145 title('Comparison between position real vs calculated')
146 grid on
147 legend('x_C(1)','y_C(2)','z_C(3)','x_R(1)','y_R(2)','z_R(3)','Location','northeastoutside')
148 hold off
149
150 n = length(coppeliagraph(1244:end-1,3));
151 m = length(rArr(1,:));
152
153 x = interp1(linspace(1,n,m)',rArr(1,:),(1:n)');
154 y = interp1(linspace(1,n,m)',rArr(2,:),(1:n)');
155 z = interp1(linspace(1,n,m)',rArr(3,:),(1:n)');
156
157 figure(1)
158 plot3(x,y,z,coppeliagraph(1244:end-1,2),coppeliagraph(1244:end-1,3),
        coppeliagraph(1244:end-1,4),'--','Linewidth',2)
159 hold on
160 xlabel('x')
161 ylabel('y')
162 zlabel('z')
163 title('Comparison between position real vs calculated')
164 grid on
165 legend('Calculated','Real')
166 hold off
167
168 % exportgraphics(gca,'effector_position.png','Resolution',300)
```