Nome: Fabio Grassiotto
RA: 890441
Disciplina: IA368N, 2º S 2018

Atividade 1 – Cinemática de uma perna de robô

Objetivo

O objetivo desta atividade é analisar a cinemática direta e inversa de uma perna de robô com um ponto de apoio pontual.

A perna de robô que será utilizada na atividade está ilustrada abaixo na figura 1.

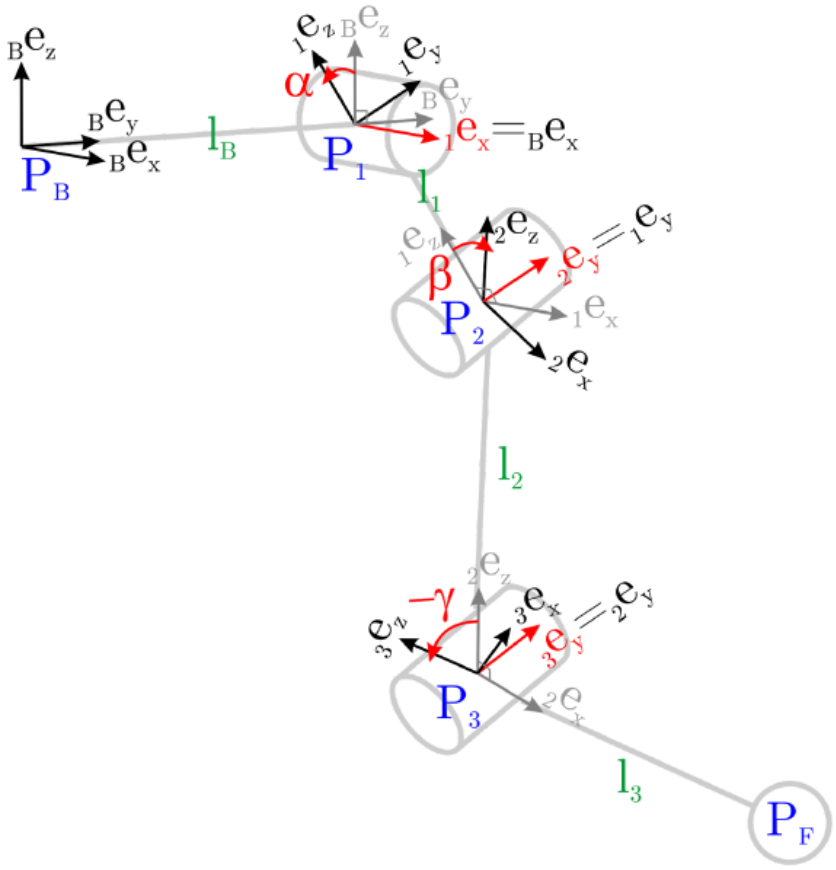

Figura 1 - Perna de Robô

Exercício 1

Dada a descrição cinemática de uma perna de robô, determinar as três matrizes de rotação relativas baseadas nas coordenadas generalizadas alpha, beta e gamma.

Solução:

Editando o arquivo ex01.m, foram executados os cálculos das matrizes de rotação nos eixos x e y conforme solicitado:

```
R_B1 = [ 1 0 0; 0 cos(alpha) -sin(alpha); 0 sin(alpha) cos(alpha)]; % x axis
R_12 = [ cos(beta) 0 sin(beta); 0 1 0; -sin(beta) 0 cos(beta)]; % y axis
R_23 = [ cos(gamma) 0 sin(gamma); 0 1 0; sin(gamma) 0 cos(gamma)]; % y axis again
```

Exercício 2

Utilizando o resultado do exercício anterior, determinar os vetores de posição relativa, assim como as matrizes de transformação homogêneas.

Solução:

Editando o arquivo ex02.m, foram adicionados os vetores relativos de posição, assim como as matrizes de transformação homogêneas.

```
% write down the 3x1 relative position vectors for link length l_i=1
r_B1_inB = [0 1 0];  % y offset
r_12_in1 = [0 0 -1]; % z offset
r_23_in2 = [0 0 -1]; % z offset
r_3F_in3 = [0 0 -1]; % z offset

% write down the homogeneous transformation matrices
H_B1 = [ 1 0 0 0; 0 cos(alpha) -sin(alpha), 1; 0 sin(alpha) cos(alpha) 0; 0 0 0 1];
H_12 = [ cos(beta) 0 sin(beta) 0; 0 1 0 0; -sin(beta) 0 cos(beta) -1; 0 0 0 1];
H_23 = [ cos(gamma) 0 sin(gamma) 0; 0 1 0 0; -sin(gamma) 0 cos(gamma) -1; 0 0 0 1];
```

Exercício 3

Neste exercício se procura determinar a matrix Jacobiana do ponto de apoio assim como a velocidade generalizada para o movimento cartesiano, utilizando como entrada o vetor de posição relativa do ponto de apoio.

Solução:

Editando o arquivo ex03.m, foram executadas as seguintes operações:

- Seleção dos componentes vetoriais do ponto de apoio em relação ao eixo inercial.
- Diferenciação parcial destes componentes vetoriais em relação a cada uma das coordenadas generalizadas alpha, beta e gamma, obtendo o Jacobiano.
- Determinação do valor numérico do Jacobiano para um ângulo inicial qi.

```
% determine the foot point Jacobian J_BF_inB=d(r_BF_inB)/dq
f = r_BF_inB;
fx = f(1);
fy = f(2);
fz = f(3);
J_BF_inB = [ diff(fx, alpha) diff(fx, beta) diff(fx, gamma); ...
             diff(fy, alpha) diff(fy, beta) diff(fy, gamma); ...
             diff(fz, alpha) diff(fz, beta) diff(fz, gamma) ];


% what generalized velocity dq do you have to apply in a configuration q = [0;60Â°;-120Â°]
% to lift the foot in vertical direction with v = [0;0;-1m/s];
v = [0; 0; -1];
qi = [0; 60*(pi/180); -120*(pi/180)];
```

```
15.
16. % Determine the numerical value of the foot point jacobian for initial joint angles qi
17. JBF = double(eval(subs(J_BF_inB, q, qi)));
18.
19. % Determine the numerical value for dq
20. dq = inv(JBF)*v;
```

Exercício 4

Neste exercício será utilizada a integração do ambiente de simulação V-Rep com o software Matlab para encontrar os ângulos alpha, beta e gamma que permitem o ponto de apoio tocar o solo. Será utilizada a implementação do método numérico de Newton para aproximar uma configuração do objetivo qgoal.

Solução:

Editando o arquivo ex04.m, foram executadas as seguintes operações:

- Criação de uma estrutura de repetição no Matlab com critério de parada.
- Cálculo do erro relativo entre o vetor de objetivo e a posição atual do vetor de acordo com as coordenadas generalizadas.
- Obtenção do Jacobiano utilizando o algoritmo de inversão de matrizes Moore-Penrose.
- Atualização da variável de error relativo até que o erro se torne pequeno o suficiente.

```
1.  % determine the foot point Jacobian J_BF_inB=d(r_BF_inB)/dq
2.  f = r_BF_inB;
3.  fx = f(1);
4.  fy = f(2);
5.  fz = f(3);
6.  while(1) % Repeat loop while error is not small enough (1e-5)
7.    error = rGoal - r_BF_inB(q0(1), q0(2), q0(3));
8.    % Get the vector magnitude of the error (norm)
9.    n = norm(error);
10.   disp(['Error norm = ', num2str(n)]);
11.   if (n < 1e-5)
12.     break;
13.   end
14.   % Get Moore-Penrose Inverse for the Jacobian J_BF_inB at the current q
15.   jInv = pinv(J_BF_inB(q0(1),q0(2),q0(3)));
16.   qGoal = q0 + jInv*error;
17.   q0 = qGoal;
18. end
```

Nota-se que o algoritmo foi capaz de convergir para um erro mínimo rapidamente, com apenas 5 interações.

```
1.  Inverse Kinematics Algorithm
2.  Error norm = 0.90879
3.  Error norm = 0.47752
4.  Error norm = 0.072341
5.  Error norm = 0.0025263
6.  Error norm = 3.2573e-06
7.  qGoal end = -0.24498 -1.1864 2.0004
```

Exercício 5

Novamente neste exercício será utilizado o ambiente de simulação V-Rep com o software Matlab. Neste problema, utilizaremos cinemática inversa diferencial para controlar a trajetória do ponto de apoio.

Solução:

Editando o arquivo ex05.m, foram executadas as seguintes operações:

1. Criação de um controlador proporcional para determinar a velocidade do ponto de apoio.
   O código do script Matlab foi alterado adicionando os passos:
   - Determinação do erro de posição entre o objetivo e a posição atual do ponto de apoio da perna.
   - Determinação interativa do valor do ganho k para minimizar a distância do caminho proposto.
   - Cálculo da velocidade do ponto de apoio.

```
% controller:
% step 1: create a simple p controller to determine the desired foot
% point velocity

% Error = Goal Position in the circle - Current position of the leg.
err = rGoal(t) - rArr(:,i);
% Gain for proportional controller, applied to the error.
% k value was adjusted with comparison to the path for the foot
% circle.
k = 16.0;
% Calculate foot point velocity
v = drGoal(t) + k*err;
```

2. Cálculo das velocidades generalizadas

   Foi alterado o código Matlab para se obter o Jacobiano utilizando o algoritmo de inversão de matrizes Moore-Penrose, obtendo as velocidades generalizadas.

```
% step 2: perform inverse differential kinematics to calculate the
% generalized velocities

% Get Moore-Penrose Inverse for the Jacobian J_BF_inB at the current q
jInv = pinv(J_BF_inB(q(1),q(2),q(3)));
dq = jInv * v;
```

Pode-se verificar o resultado da trajetória obtida no ambiente de simulação, importando os valores do gráfico no vrep e comparando com a trajetória planejada. Para tanto, após importar os dados de trajetória, um script para plotar os gráficos no Matlab foi implementado:

```
% Prepare Graph
title('Comparação de Trajetórias');
xlabel('X');
ylabel('Y');

% Center and Radius for the circle.
x = 0;
y = -2;
r = 0.5;

hold on
% Robot Circle
xRobot = export(:,2);
yRobot = export(:,4);
plot(xRobot, yRobot);

% Base Circle with same center and radius
th = 0:pi/50:2*pi;
xunit = r * cos(th) + x;
yunit = r * sin(th) + y;
plot(xunit, yunit);
hold off
```

O gráfico resultante comparando as duas trajetórias pode ser visualizado na Figura 2 abaixo.

Pode-se verificar que a trajetória obtida utilizando o controlador proporcional é bastante próxima à linha delimitada pelo círculo do objetivo.
Nota-se que o erro aumenta nas laterais da imagem (valores máximos e mínimos da variável X) e diminui na base e topo da imagem (valores máximos e mínimos da variável Y).
Também é possível notar que o traçado no término do trajeto da perna robótica apresenta um erro maior do que o esperado.

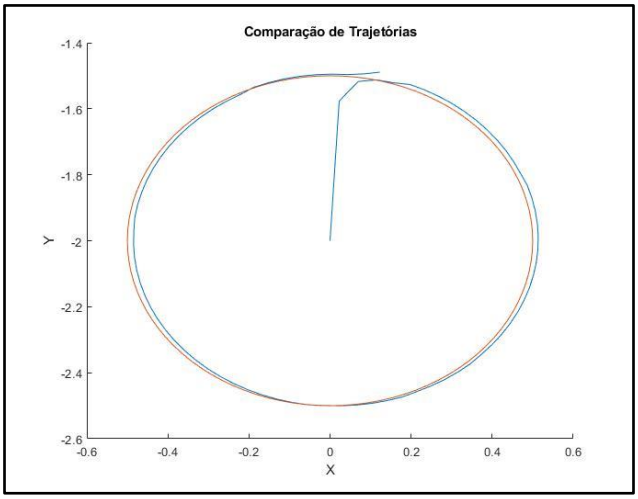

Figure 2 - Comparação de Trajetórias

Conclusão

Os exercícios propostos permitiram exercitar os passos necessários para a análise das cinemáticas direta e inversa para uma perna robótica. Através do cálculo de matrizes e Jacobianos no Matlab, foi possível integrar com o ambiente de simulação V-Rep e visualizar a manipulação da perna.