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.

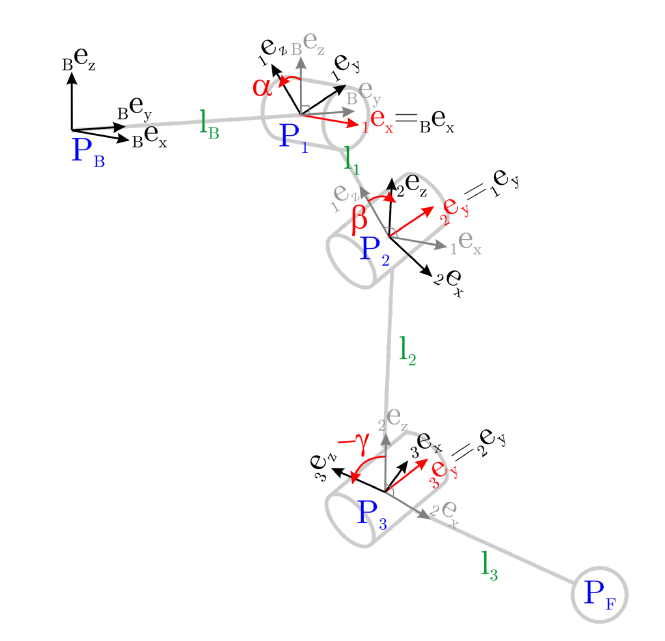


Figura - Perna de Robô

**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:

1. R\_B1 = [ 1 0 0; 0 cos(alpha) -sin(alpha); 0 sin(alpha) cos(alpha)]; % x axis
2. R\_12 = [ cos(beta) 0 sin(beta); 0 1 0; -sin(beta) 0 cos(beta)]; % y axis
3. 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.

1. % write down the 3x1 relative position vectors **for** link length l\_i=1
2. r\_B1\_inB = [0 1 0];  % y offset
3. r\_12\_in1 = [0 0 -1]; % z offset
4. r\_23\_in2 = [0 0 -1]; % z offset
5. r\_3F\_in3 = [0 0 -1]; % z offset
7. % write down the homogeneous transformation matrices
8. H\_B1 = [ 1 0 0 0; 0 cos(alpha) -sin(alpha), 1; 0 sin(alpha) cos(alpha) 0; 0 0 0 1];
9. H\_12 = [ cos(beta) 0 sin(beta) 0; 0 1 0 0; -sin(beta) 0 cos(beta) -1; 0 0 0 1];
10. 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.

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. J\_BF\_inB = [ diff(fx, alpha) diff(fx, beta) diff(fx, gamma); ...
7. diff(fy, alpha) diff(fy, beta) diff(fy, gamma); ...
8. diff(fz, alpha) diff(fz, beta) diff(fz, gamma) ];

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

1. % controller:
2. % step 1: create a simple p controller to determine the desired foot
3. % point velocity
5. % Error = Goal Position in the circle - Current position of the leg.
6. err = rGoal(t) - rArr(:,i);
7. % Gain **for** proportional controller, applied to the error.
8. % k value was adjusted with comparison to the path **for** the foot
9. % circle.
10. k = 16.0;
11. % Calculate foot point velocity
12. v = drGoal(t) + k\*err;
13. 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.

1. % step 2: perform inverse differential kinematics to calculate the
2. % generalized velocities
4. % Get Moore-Penrose Inverse **for** the Jacobian J\_BF\_inB at the current q
5. jInv = pinv(J\_BF\_inB(q(1),q(2),q(3)));
6. 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:

1. % Prepare Graph
2. title('Comparação de Trajetórias');
3. xlabel('X');
4. ylabel('Y');
6. % Center and Radius **for** the circle.
7. x = 0;
8. y = -2;
9. r = 0.5;
11. hold on
12. % Robot Circle
13. xRobot = export(:,2);
14. yRobot = export(:,4);
15. plot(xRobot, yRobot);
17. % Base Circle with same center and radius
18. th = 0:pi/50:2\*pi;
19. xunit = r \* cos(th) + x;
20. yunit = r \* sin(th) + y;
21. plot(xunit, yunit);
22. hold off

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

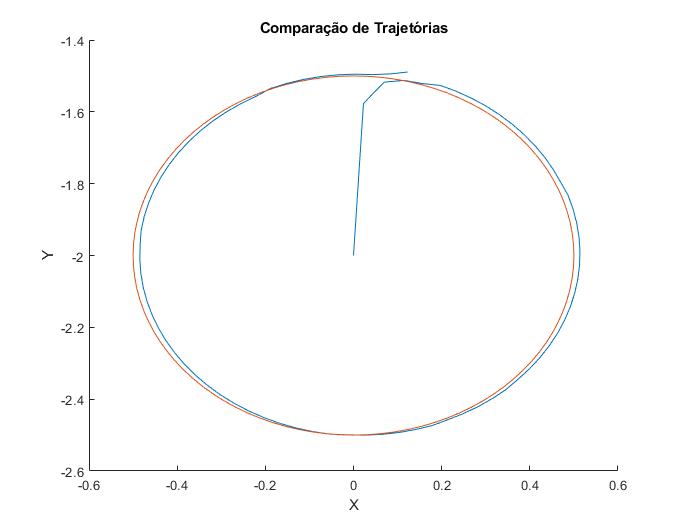
Pode-se verificar que a trajetória obtida utilizando o controlador proporcional é bastante próxima à linha delimitada pelo círculo do objetivo.

Figure - Comparação de Trajetórias

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.

**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.