Laboratorio #4 robótica 2020-2

Table of Contents

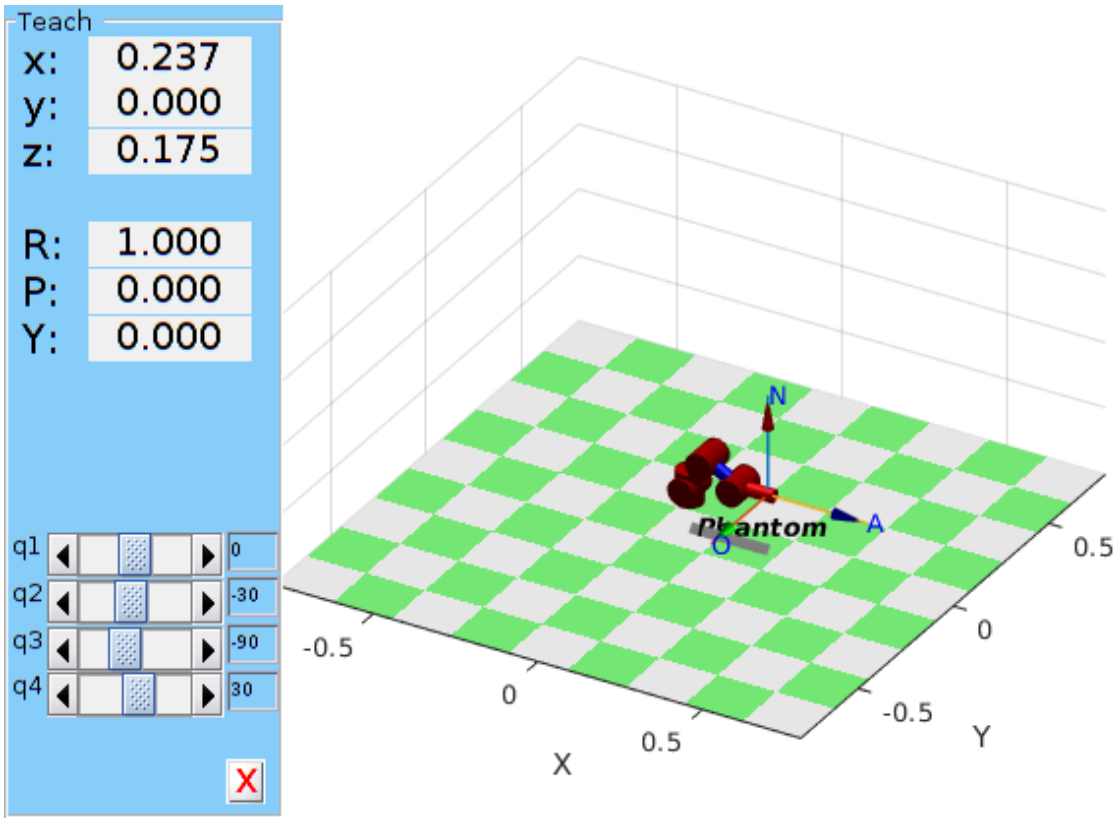
Leonardo Fabio Mercado Benítez

C.C: 1.016.050.737

Código: 25481090

Modelo del robot phanton X pincher:

```
clc;
clear;
close all;

syms Q1 Q2 Q3 Q4

l1 = 0.135875;
l2 = 0.107;
l3 = 0.107;
l4 = 0.091;


L(1) = Link('revolute','alpha', 0,    'a',0,   'd',l1,  'offset', 0,
  'modified', 'qlim',[-2*pi 2*pi]);
L(2) = Link('revolute','alpha', pi/2, 'a',0,   'd',0,   'offset',
 pi/2, 'modified', 'qlim',[-2*pi 2*pi]);
L(3) = Link('revolute','alpha', 0,    'a',l2,  'd',0,   'offset',
 0, 'modified', 'qlim',[-2*pi 2*pi]);
L(4) = Link('revolute','alpha', 0,    'a',l3,  'd',0,   'offset', 0,
  'modified', 'qlim',[-2*pi 2*pi]);


robot = SerialLink(L,'name','Phantom_x');
robot.tool = [0 0 1 l4;
              1 0 0 0;
              0 1 0 0;
              0 0 0 1];
maximo = [-0.800 0.800 -0.800 0.800 0 0.800];
pose_1 = [0 pi/4 -pi/2 -pi/4];
pose_2 = [0 -pi/6 -pi/2 pi/6];
robot.plot(pose_2,'workspace', maximo,'noa','view',[30 30]);
robot.teach;
```

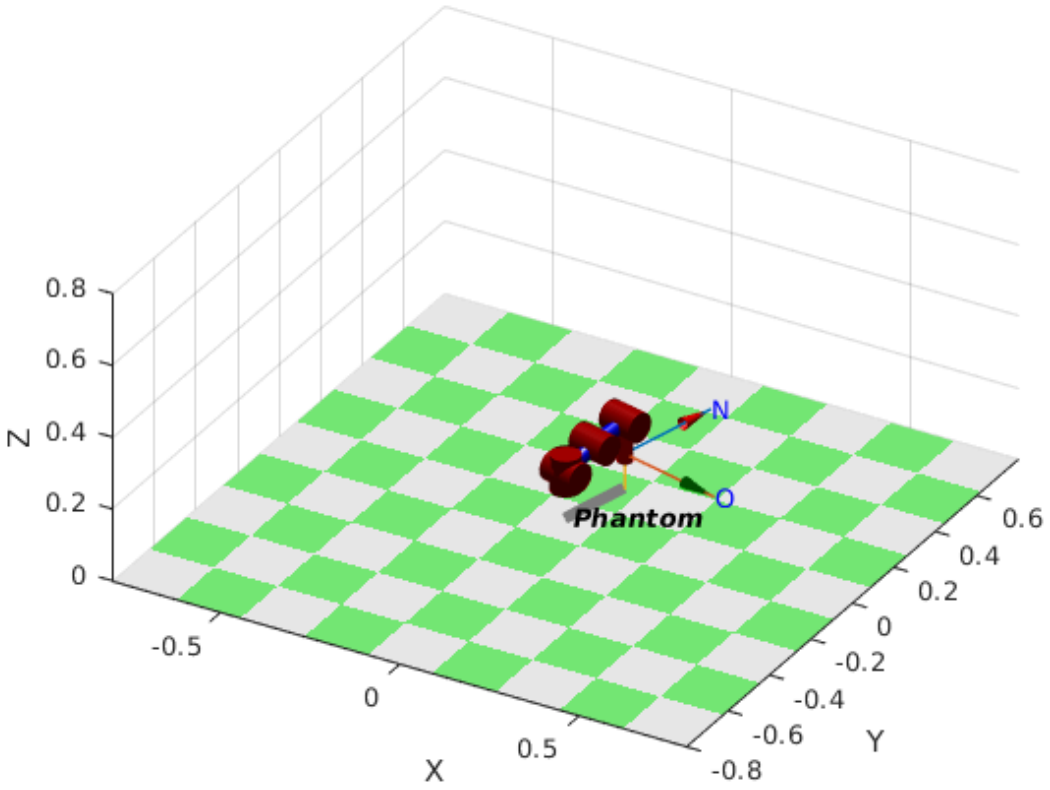


Modelo cinematico inverso del robot phantom X

```
close;
x = 0.05;
y = 0.2;
z = 0.1;
phi = deg2rad(-90);
l1 = 0.135875;
l2 = 0.107;
l3 = 0.107;
l4 = 0.091;
elbow = 0;

q = zeros(1,4);
q(1) = atan2(y,x);
x_0 = sqrt(x.^2 + y.^2) - l4 * cos(phi);
z_0 = (z-l1) - l4 * sin(phi);

num = x_0.^2 + z_0.^2 - l2.^2 - l3.^2;
den = 2*l2*l3;
D = num./den;
flag = (D<=1);

if flag
```

```
    q(3) = atan2(-sqrt(1-D.^2),D);
    if elbow
        q(3) = atan2(sqrt(1-D.^2),D);
    end

    q(2) = -pi/2 + (atan2(z_0,x_0) - atan2(l3*sin(q(3)),
 l2+l3*cos(q(3))));
    q(4) = phi - pi/2 - q(2) - q(3);


    disp('La solución hallada es: ');
    disp(q)

    tg = jtraj(pose_1,q,50);
    robot.plot(tg,'workspace', maximo,'noa','view',[30 30]);


else
    warning('No se hallo una solución real');
    q = NaN(1,4);
end
```

```
La solución hallada es:
    1.3258   -1.2345   -0.1500   -1.7571
```



Espacio Diestro de un manipulador:

```
% Es el espacio que puede alcanzar el efector final del robot con
 todas las
% orientaciones.
```

Métodos disponibles en el toolbox para determinar la cinemática inversa de un manipulador:

El toolbox posee los siguientes métodos para hallar la cinemática inversa de un robot:

1) ikine6s: Halla la cinemática inversa para un robot de 6 ejes con muñeca esferica.

2) ikine: Halla la cinemática inversa usando metodos numericos iterativos.

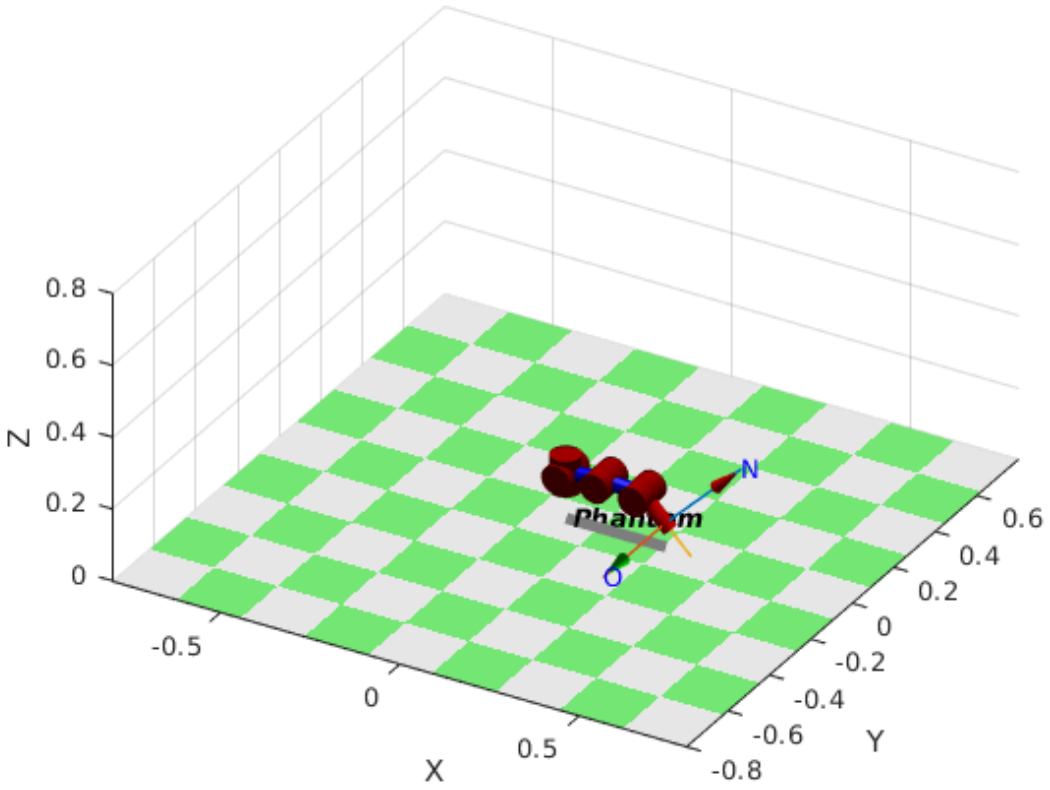
3) ikunc: Halla la cinemática inversa usando optimización.

4) ikcon: Halla la cinemática inversa usando optimización con limite de articulación.

5) ikine_sym: Halla la cinemática inversa analitica de forma símbolica.

Dada una matriz MTH para el efector final llevar el robot a esa posición:

```
close;
matriz_objetivo = transl(0.278,0.0,0.072)*rpy2tr(180,45,0,'deg');
configuracion_matriz_objetivo = robot.ikunc(matriz_objetivo, pose_1);
tg = jtraj(pose_1,configuracion_matriz_objetivo,50);
robot.plot(tg,'workspace', maximo,'noa','view',[30 30]);
```



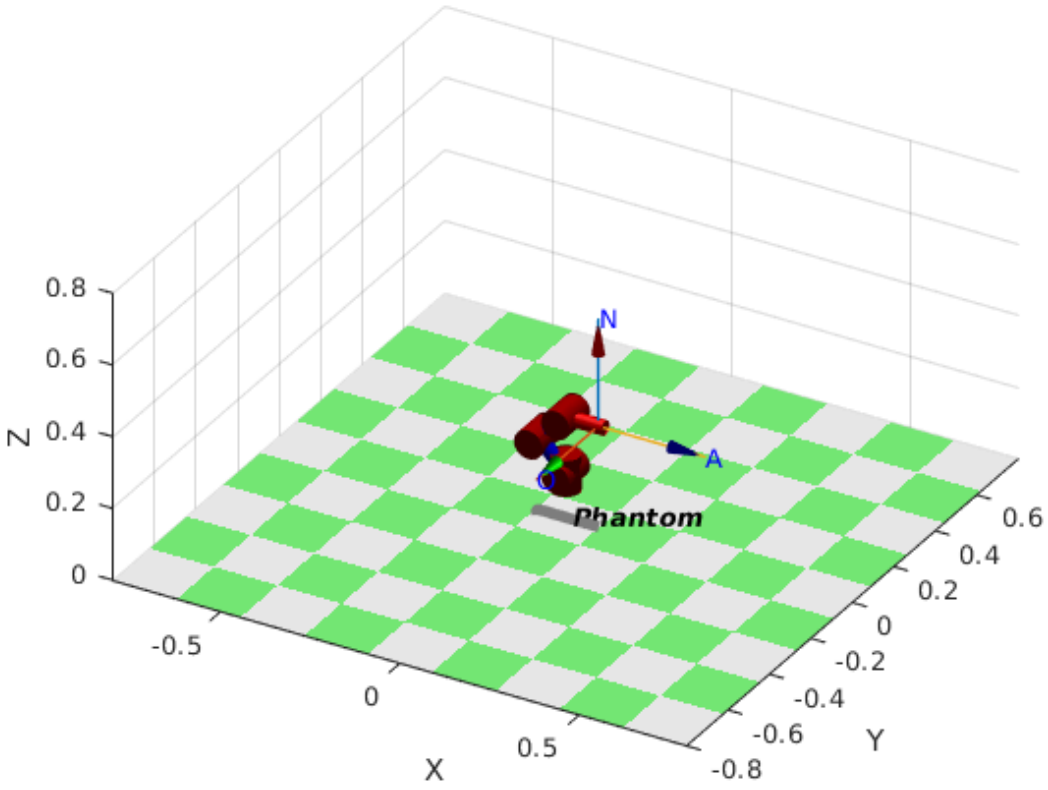
Puntos de pick and place:

```
close;
%--------------------------------------------------------------------------
% Puntos y orientaciones:
punto_orientado_1 = [0.2 0.0 0.10 -90];
punto_orientado_2 = [0.2 0.0 0.0 -90];
punto_orientado_3 = [0.2 0.0 0.10 -90];
punto_orientado_4 = [0.1 0.0 0.340 0.0];
punto_orientado_5 = [-0.2 0.0 0.10 -90];
punto_orientado_6 = [-0.2 0.0 0.0 -90];
punto_orientado_7 = [-0.2 0.0 0.10 -90];


%--------------------------------------------------------------------------
q_1 = solucion(punto_orientado_1);
tg_1 = jtraj(pose_1,q_1,50);
robot.plot(tg_1,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 1');


q_2 = solucion(punto_orientado_2);
tg_2 = jtraj(tg_1(end,:),q_2,50);
robot.plot(tg_2,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 2');
```

```
q_3 = solucion(punto_orientado_3);
tg_3 = jtraj(tg_2(end,:),q_3,50);
robot.plot(tg_3,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 3');

q_4 = solucion(punto_orientado_4);
tg_4 = jtraj(tg_3(end,:),q_4,50);
robot.plot(tg_4,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 4');

q_5 = solucion(punto_orientado_5);
tg_5 = jtraj(tg_4(end,:),q_5,50);
robot.plot(tg_5,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 5');

q_6 = solucion(punto_orientado_6);
tg_6 = jtraj(tg_5(end,:),q_6,50);
robot.plot(tg_6,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 6');

q_7 = solucion(punto_orientado_7);
tg_7 = jtraj(tg_6(end,:),q_7,50);
robot.plot(tg_7,'workspace', maximo,'noa','view',[30 30]);
disp('Fin punto 7');

tg_8 = jtraj(tg_7(end,:),pose_1,50);
robot.plot(tg_8,'workspace', maximo,'noa','view',[30 30]);

function q = solucion(data)

x = data(1);
y = data(2);
z = data(3);
phi = deg2rad(data(4));
l1 = 0.135875;
l2 = 0.107;
l3 = 0.107;
l4 = 0.091;
elbow = 0;

q = zeros(1,4);
q(1) = atan2(y,x);
x_0 = sqrt(x.^2 + y.^2) - l4 * cos(phi);
z_0 = (z-l1) - l4 * sin(phi);

num = x_0.^2 + z_0.^2 - l2.^2 - l3.^2;
den = 2*l2*l3;
D = num./den;
flag = (D<=1);

if flag
    q(3) = atan2(-sqrt(1-D.^2),D);
    if elbow
        q(3) = atan2(sqrt(1-D.^2),D);
```

```
    end

    q(2) = -pi/2 + (atan2(z_0,x_0) - atan2(l3*sin(q(3)),
 l2+l3*cos(q(3))));
    q(4) = phi - pi/2 - q(2) - q(3);


else
    warning('No se hallo una solución real');
    q = NaN(1,4);
end



end
```

```
Fin punto 1
Fin punto 2
Fin punto 3
Fin punto 4
Fin punto 5
Fin punto 6
Fin punto 7
```



Published with MATLAB® R2020a