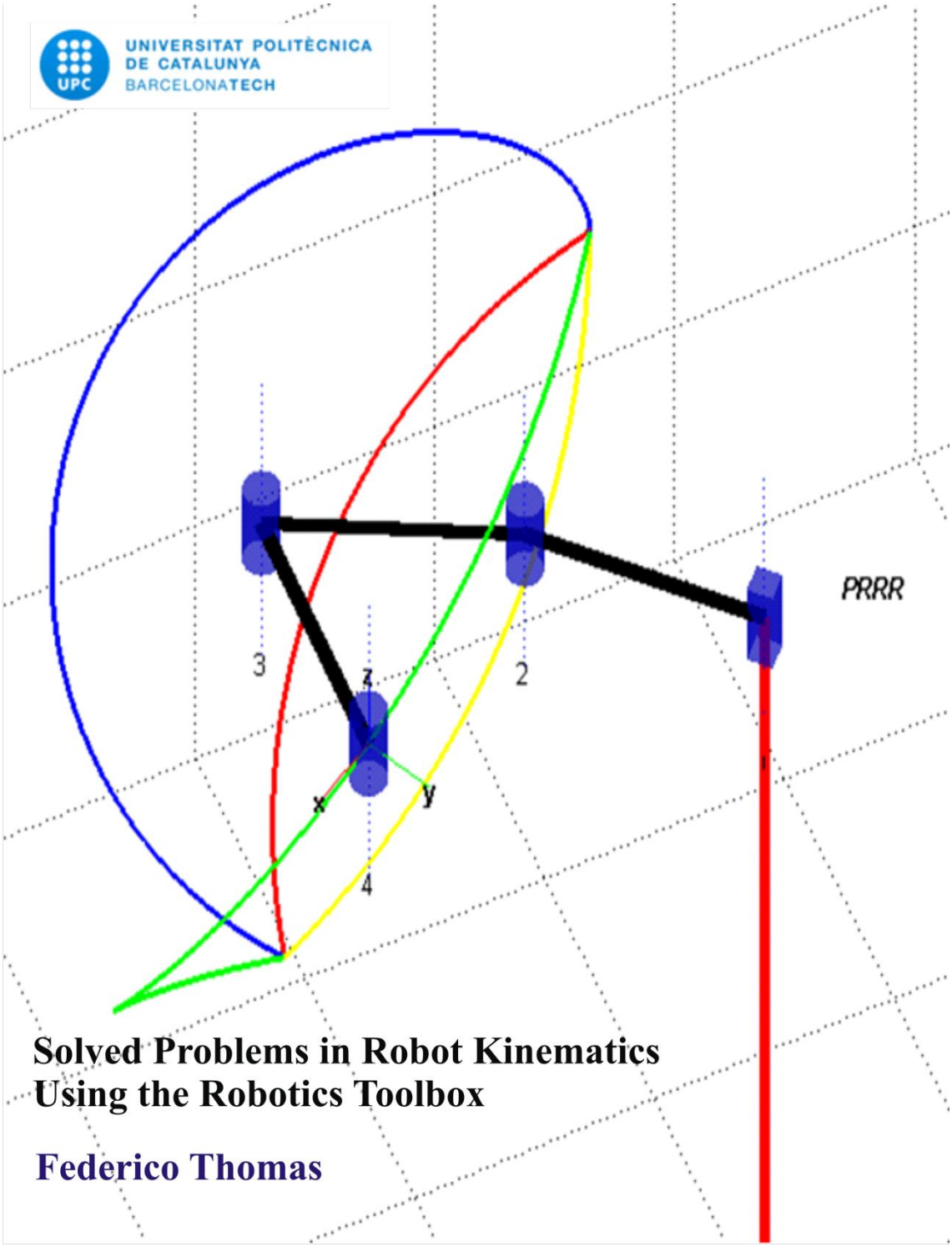
UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH



Solved Problems in Robot Kinematics Using the Robotics Toolbox

Federico Thomas

INTRODUCTION

This document is the result of the undergraduate course "Robotics" given at the Computer Science Faculty, Technical University of Catalonia, Spain. It is organized as a collection of 20 problems, each of them designed to exemplify a particular aspect of the kinematics of serial robots. Throughout this document, the Robotics Toolbox (V8) and MATLAB™ (R2011b) are used.

It is advisable to have a copy of the following book to fully understand the adopted solutions:

P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, Springer Tracts in Advanced Robotics, 2011.

To report any mistake or submit suggestions, please send me an email to the address given below.

Federico Thomas
fthomas@iri.upc.edu

Barcelona, April 2nd, 2012

PROBLEM 1

Consider the wedge-shaped object in the following drawing.

1) Obtain the transformation that should be applied to take it from the origin (left) to its final location (right). Represent this transformation as a translation followed by a rotation and vice versa. Verify that the solution is the same.

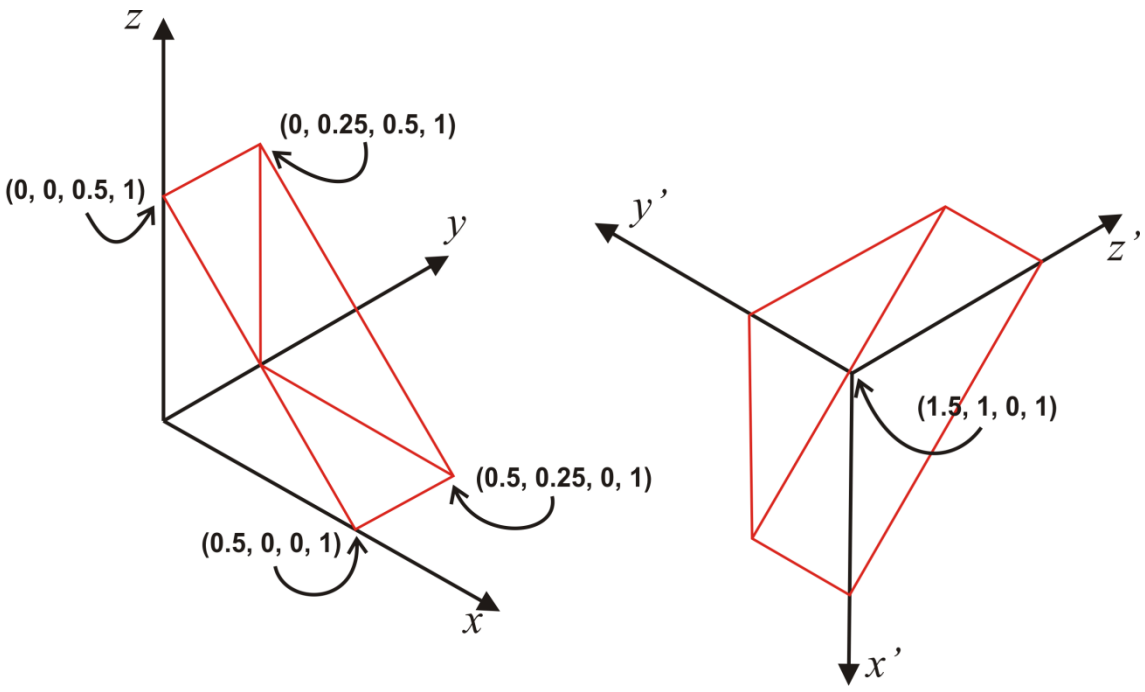
2) Compute the coordinates of the vertices of the translated and rotated wedge with respect to the original frame.

3) Decompose the rotation obtained in question 1) as a sequence of rotations about the axes X, Y, and Z.

4) The same as above but as a sequence of rotations about the axes Z, Y, and Z.

5) Obtain the equivalent axis of rotation, and the angle rotated about it, for the rotation obtained in question 1).

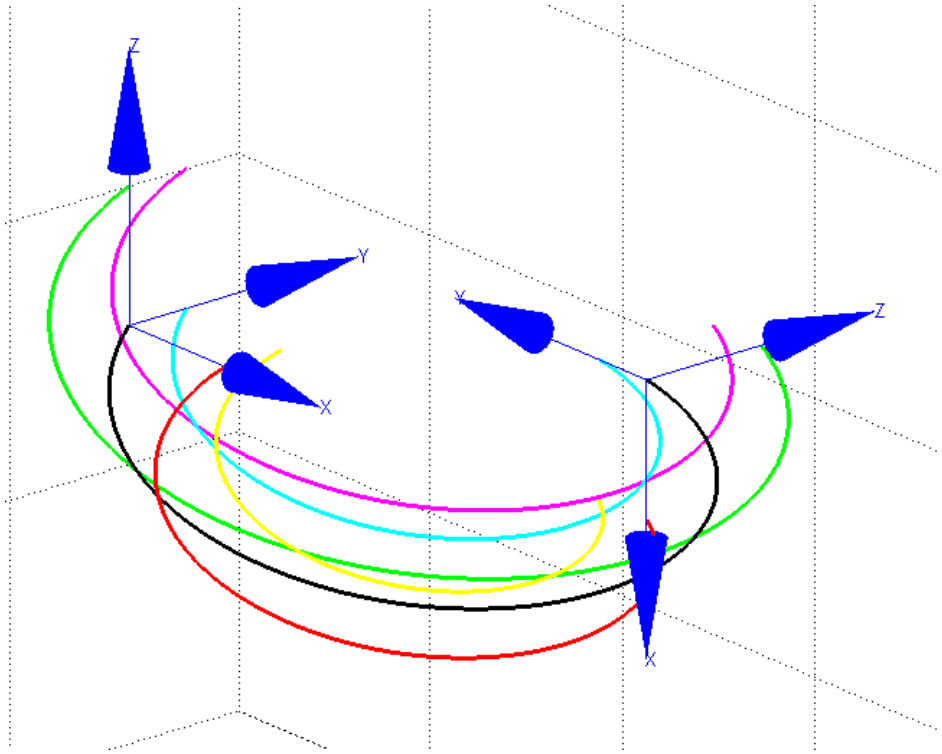
6) Obtain the screw axis for the transformation obtained in question 1) and generate the trajectory followed by the vertices of the wedge from their initial location to their final location by simultaneously interpolating the rotation about this axis and the translation along it.



SOLUTION:

```
%%% First we clean the workspace

clc;
clear;

%%% QUESTION 1:
%%%
%%% The rotation is given by the transformation

TR = [0 -1 0 0; 0 0 1 0; -1 0 0 0; 0 0 0 1];

%%% If we translate with respect to the original reference frame, the
%%% transformation is obtained by post-multiplying the rotation

T1 = transl(1.5, 1, 0)*TR;

%%% If we translate with respect to the transformed reference frame,
%%% the transformation is obtained by pre-multiplying the rotation

T2 = TR*transl(0, -1.5, 1);

%%% It can be checked that T1 and T2 are identical

%%% QUESTION 2:
%%%
%%% The coordinates of the six vertices of the wedge can be organized
%%% as the columns of a matrix as follows:

Vertices = [0   0.5 0   0    0.5   0;    ...
            0   0   0   0.25 0.25  0.25; ...
            0   0   0.5 0    0     0.5;  ...
            1   1   1   1    1     1];

%%% Then, the coordinates of the six vertices of the wedge after it
%%% is translated and rotated are:

NewVertices = T1*Vertices

%%% QUESTION 3:
%%%
%%% The decomposition of a rotation into a sequence of rotations
%%% about axes X, Y, and Z, corresponds to the computation of
%%% the roll, pitch, and yaw angles

q = tr2rpy(TR)

%%% Then, we can verify that TR is identical to

trotx(q(1))*troty(q(2))*trotz(q(3))

%%% QUESTION 4:
%%%
%%% The decomposition of a rotation into a sequence of rotations
%%% about axes X, Y, and Z, corresponds to the computation of
%%% the Euler angles
```

```
q = tr2eul(TR)

%%% Then, we can verify that TR is identical to

trotz(q(1))*troty(q(2))*trotz(q(3))

%%% QUESTION 5:

[ang vec] = tr2angvec(TR);

%%% QUESTION 6:
%%%
%%% We decompose the translation into a translation aligned with the
%%% equivalent axis of rotation and a translation perpendicular to it.

trans_align = (vec*transl(TR))*vec';
trans_ortho = transl(TR) - trans_align;

%%% We obtain a point on the screw axis. Since there are infinite
%%% solutions, we use the lsqnonneg function.

foc = lsqnonneg(eye(3)-t2r(TR), trans_ortho);

%%% We obtain a trajectory followed by the vertices by interpolating
%%% the angle and the translation.

n = 100;

for i=1:n+1
    TRINTP(:,:,i) = transl(foc)*angvec2tr((pi+ang)*(i-1)/n, vec)*...
        transl(-foc)*transl((i-1)/n*trans_align);
    new(i,:,:) = TRINTP(:,:,i)*Vertices(:,:);
end

%%% Finally, we represent these trajectories

hold on;
grid on;
view(45,45);
axis([-3 3 -3 3 -3 3]);
daspect([1 1 1]);
trplot(transl(0,0,0), 'arrow');
trplot(TR, 'arrow');

plot3(new(:,1,1), new(:,2,1), new(:,3,1), 'k', 'LineWidth', 2);
plot3(new(:,1,2), new(:,2,2), new(:,3,2), 'r', 'LineWidth', 2);
plot3(new(:,1,3), new(:,2,3), new(:,3,3), 'g', 'LineWidth', 2);
plot3(new(:,1,4), new(:,2,4), new(:,3,4), 'c', 'LineWidth', 2);
plot3(new(:,1,5), new(:,2,5), new(:,3,5), 'y', 'LineWidth', 2);
plot3(new(:,1,6), new(:,2,6), new(:,3,6), 'm', 'LineWidth', 2);
```

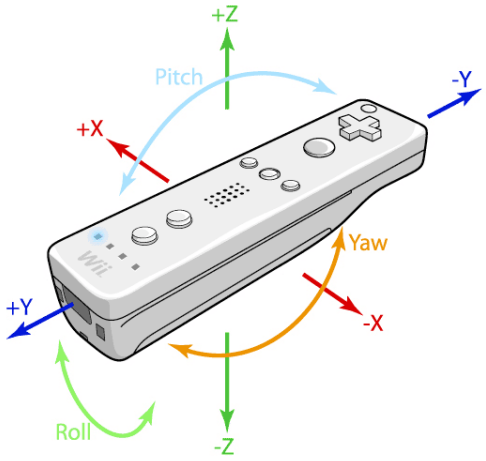


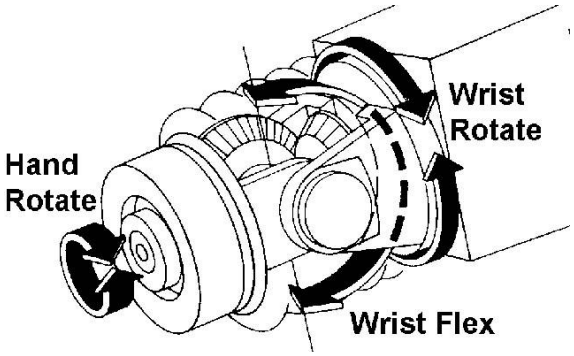
To know more:

K. Waldron and J. Schmiedeler, *Kinematics*, Chapter 1 in *Handbook of Robotics*, Springer, 2008.

PROBLEM 2

Let us suppose that we want to make a robot hand to follow the orientation of a WII™ remote controller. The problem is that the orientation of the remote controller is given in terms of roll-pitch-yaw (RPY) angles while the orientation of the robot's hand is set using ZYZ Euler angles.





1) Using the Robotics Toolbox, write a routine that returns the Euler angles corresponding to a set of RPY angles.

2) There are two solutions to determine Euler angles from a rotation matrix. The Robotics Toolbox makes the solution unique by constraining the second rotation to lie between 0° and 180°. Modify the routine written above to provide both solutions.

SOLUTION:

```
%%% QUESTION 1:

function euler = rpy2eul(rpy)
% RPY2EUL Converts a RPY angles to Euler angles form
%
%   [PHI THETA PSI] = RPY2EUL([ROLL PITCH YAW])
%
% Returns a vector of 3 angles corresponding to rotations about
% the Z, Y and Z axes, respectively, from roll/pitch/yaw angles.
%
% See also:  RPY2TR, TR2EUL

    euler =  tr2eul(rpy2tr(rpy));

end
```

```
%%% QUESTION 2:
```

The rotation matrix given by a set of Euler angles (phi, theta, psi) can be expressed as

R = rotz(phi)*roty(theta)*rotz(psi)

Now observe that

roty(theta) = rotz(pi)*roty(-theta)*rotz(pi)

Then,

R = rotz(phi)*rotz(pi)*roty(-theta)*rotz(pi)*rotz(psi)

In other words,

R = rotz(phi+pi)*roty(-theta)*rotz(psi+pi)

Thus, we conclude that, if (phi, theta, psi) is a set of Euler angles, (phi+pi, -theta, psi+pi) is another set of Euler angles that represent the same orientation. If theta is constrained to be in the interval [0, pi], then –theta is in the interval [-pi, 0].

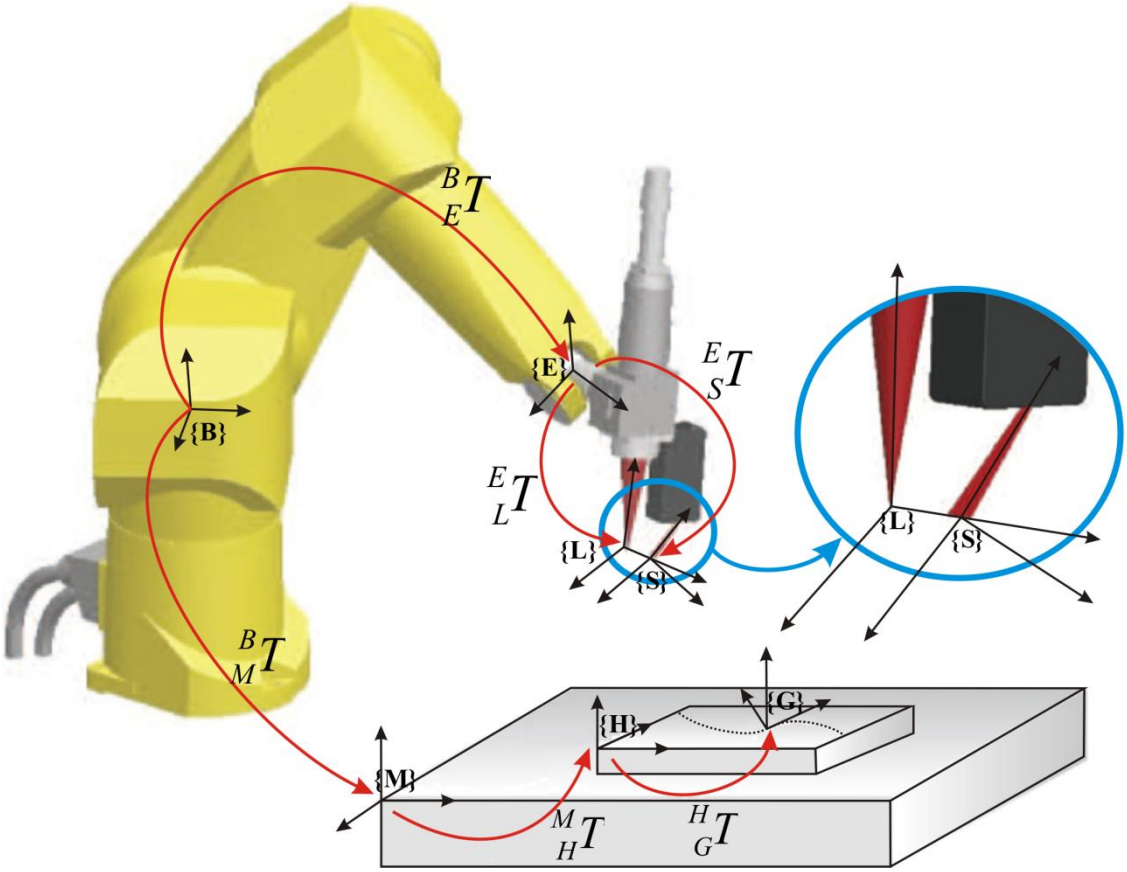
The above function can be modified to provide both solutions as follows:

```
function euler = rpy2eul(rpy, sigma)
% RPY2EUL Converts a RPY angles to Euler angles form
%
%   [PHI THETA PSI] = RPY2EUL([ROLL PITCH YAW])
%
% Returns a vector of 3 angles corresponding to rotations about
% the Z, Y and Z axes, respectively, from roll/pitch/yaw angles.
%
% If sigma=-1 theta is in the interval [-pi, 0]. Otherwise is in
% the interval [0, pi]
%
% See also:  RPY2TR, TR2EUL

euler =  tr2eul(rpy2tr(rpy));

if sigma == -1
        euler(1) = euler(1) + pi;
        euler(2) = - euler(2);
        euler(3) = euler(3) + pi;
end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 3

Laser welding requires high accuracies of the position of the laser focal point with respect to the seam trajectory. Therefore seam-tracking sensors, that measure the seam trajectory close to the laser focal spot, are required.

Illustrated below is a robotized laser welding workstation.



The following frames are defined:

{B}: Base frame. This frame is attached to the robot base.

{E}: End-effector frame. It is described with respect to the base frame by coordinate transformation ${}^{B}_{E}T$, which is a function of the joint angles of the robot arm.

{L}: Laser tool frame. The laser tool frame is located at the focal point of the laser beam. The z-axis of this frame coincides with the laser beam axis. Because the laser beam is axi-symmetric, the direction of the y-axis is arbitrary. Without loss of generality it is chosen in the direction of the sensor tool frame. This frame is described with respect to the base frame by coordinate transformation ${}^{E}_{L}T$.

{S}: Sensor tool frame. The seam tracking sensor is fixed to the welding head. The transformation ${}^{E}_{S}T$ describes the sensor tool frame with respect to the end-effector frame.

{M}: Station frame. The station frame is the base of the worktable where a product is attached to. It is possible that the product is clamped on a manipulator which moves the product with respect to the base frame. In that case the transformation ${}^{B}_{M}T$ describes the station frame with respect to the base frame and depends on the joint values of the manipulator.

{H}: Product frame. The product frame is located on a product. The transformation ${}^{M}_{H}T$ describes it with respect to the station frame. This frame is useful if a series of similar products are welded on different locations of a station.

{G}: Seam frame. Every discrete point on a seam can be described with a different coordinate frame. The transformation ${}^{H}_{G}T$ describes a seam frame with respect to the product frame.

Questions:

1) Obtain ${}^{L}_{S}T$ as a function of the other known transformations.
2) Obtain ${}^{B}_{E}T$ as a function of the other known transformations when **{L}** is made coincident with **{G}**.
3) The seam-tracking sensor can measure the location ${}^{S}_{G}T$ on the seam trajectory with respect to its coordinate frame S. Obtain this seam location with respect to generic frame **{F}** as a function of ${}^{B}_{F}T$, a known transformation from the robot base frame B to the generic frame F.
4) Let us suppose that the laser is changed by another model. Then, ${}^{E}_{L}T$ changes. Let ${}^{E}_{L}T'$ denote the new transformation. Obtain ${}^{B}_{E}T$ as a function of the other known transformations when **{L}** is made coincident with **{G}**.

SOLUTION

…

To know more:

M. de Graaf, R. Aarts, B. Jonker, and J. Meijer, “Real-time seam tracking for robotic laser welding using trajectory-based control,” *Control Engineering Practice*, Vol. 18, No. 8, pp. 944–953, 2010.

PROBLEM 4

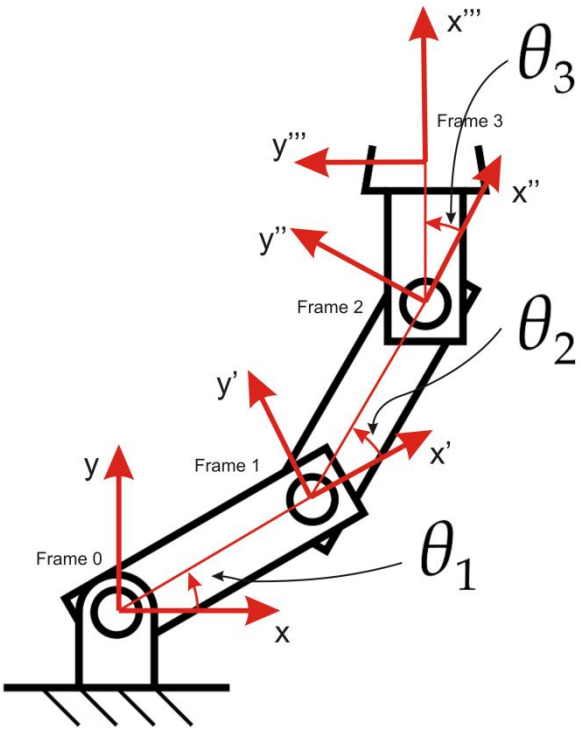
Consider the three-link planar robot shown in the following drawing. Reference frame 0 represents the world reference frame and reference frame i, for i=1,2,3, is a reference frame attached to link i. The lengths of the links are l_1, l_2, and l_3, respectively.

1) Obtain the reference frame transformations between neighboring links, that is, ${}^{0}_{1}T$, ${}^{1}_{2}T$, and ${}^{2}_{3}T$ as a function of θ_1, θ_2, and θ_3, respectively.

2) Obtain the transformation that links the reference frame between the world reference frame and that in the robot hand. Obtain its symbolic expression.

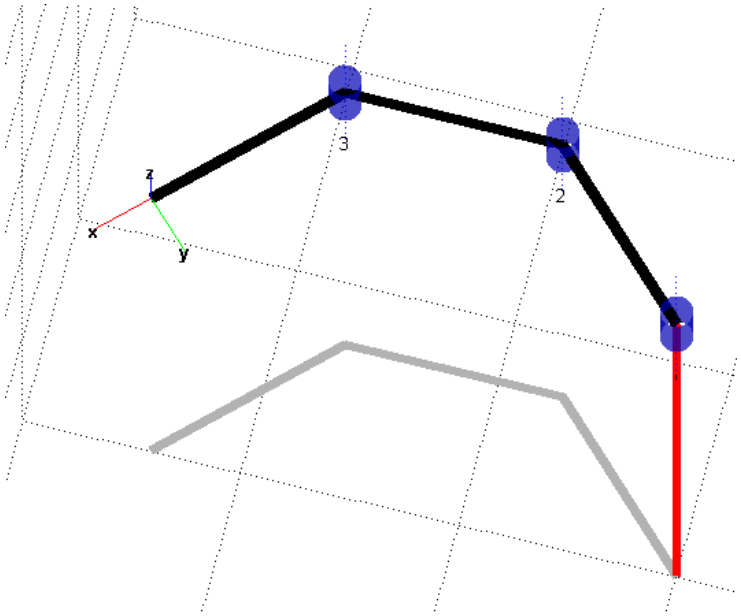
3) Try to obtain the set of values for θ_1, θ_2, and θ_3 that takes the robot hand to the location given by $\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ with respect to the world reference frame. To simplify the problem, assume that $l_1 = l_2 = l_3 = 1$.

4) Prove that, in general, there two solutions to the inverse kinematic problem for this robot and write a function in MATLAB that returns both.



SOLUTION

```
%%% First we clean the workspace

clc;
clear;

%%% QUESTION 1:

a1 = sym('a1');
l1 = sym('l1');
a2 = sym('a2');
l2 = sym('l2');
a3 = sym('a3');
l3 = sym('l3');

T01 = trotz(a1)*transl(l1, 0, 0);
T12 = trotz(a2)*transl(l2, 0, 0);
T23 = trotz(a3)*transl(l3, 0, 0);

%%% QUESTION 2:

T03 = T01*T12*T23;

%%% Alternatively, we could also solve the forward kinematics for
%%% this robot using its D-H parameters. Then, ssuming a_i=1

L(1) = Link([0 0 1 0]);
L(2) = Link([0 0 1 0]);
L(3) = Link([0 0 1 0]);

ThreeLink = SerialLink(L);
ThreeLink.name = 'Planar3R';

%%% To verify that we have correctly built the robot, we can plot it

ThreeLink.plot([pi/4 pi/4 pi/4]);
```



```
%%% Now, the forward kinematics can be solved using the
%%% fkine function. For example:

ThreeLink.fkine([pi/4 pi/4 pi/4]);

%%% QUESTION 3:
%%%
%%% The problem consist in solving the inverse kinematics of this
%%% robot for the following configuration:

TL = [-1 0 0 0 ; 0 -1 0 1; 0 0 1 0; 0 0 0 1];

%%% One possibility consists in using the numerical method
%%% implemented by ikine. The problem is to give a good guess so that
%%% the algorithm converges to a valid solution. After some trial and
%%% error a good starting point is theta_1=0, theta_2=pi, and
%%% theta_3=pi

Q0 = [0 -pi/2 pi/2];

QF = ThreeLink.ikine(TL, Q0, [1 1 0 0 0 1]);

ThreeLink.plot(QF);

%%% We can generate a trajectory between Q0 and QF

TRAJ = jtraj(Q0, QF, (0:.05:1));

ThreeLink.plot(TRAJ);

%%% QUESTION 4:
%%%
```

From basic trigonometry, the position and orientation of the end effector can be written in terms of the joint coordinates in the following way:

$$x = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3)$$
$$y = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3)$$
$$\varphi = \theta_1 + \theta_2 + \theta_3$$

To find the joint coordinates for a given set of end-effector coordinates (x, y, φ), one needs to solve the above nonlinear equations for $\theta_1, \theta_2,$ and θ_3.

Substituting the last of the three above equations into the other two we can eliminate θ_3. Then, we have two equations in θ_1 and θ_2.

$$x - l_3 \cos\varphi = l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2)$$
$$y - l_3 \sin\varphi = l_1 \sin\theta_1 + l_2 \sin(\theta_1 + \theta_2)$$

where the unknowns have been grouped on the right hand side. The left hand side depends only on the end-effector Cartesian coordinates and are therefore known. Now,

renaming the left hand sides, $x' = x - l_3 \cos\varphi$ and $y' = y - l_3 \sin\varphi$, regrouping terms, squaring both sides in each equation and adding them, we get a single nonlinear equation in θ_1 :

$$2l_1 x' \cos\theta_1 + 2l_1 y' \sin\theta_1 + (l_2^2 - l_1^2 - x'^2 - y'^2) = 0$$

There are two solutions for θ_1 in the above equation given by

$$\theta_1 = \arctan 2(y', x') \pm \arccos\left(\frac{l_1^2 + x'^2 + y'^2 - l_2^2}{2l_1\sqrt{x'^2 + y'^2}} \right)$$

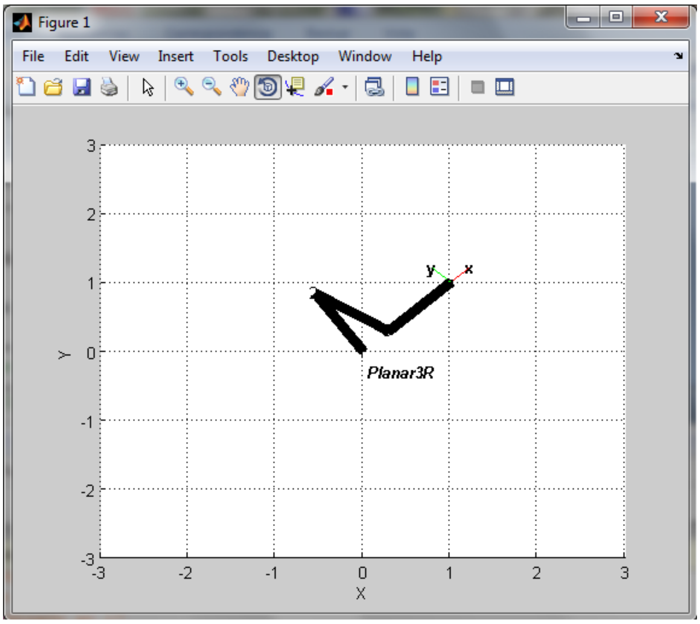
Substituting any of these solutions back gives us

$$\theta_2 = \arctan 2(y' - l_1 \sin\theta_1, x' - l_1 \cos\theta_1) - \theta_1$$
$$\theta_3 = \varphi - \theta_1 - \theta_2$$

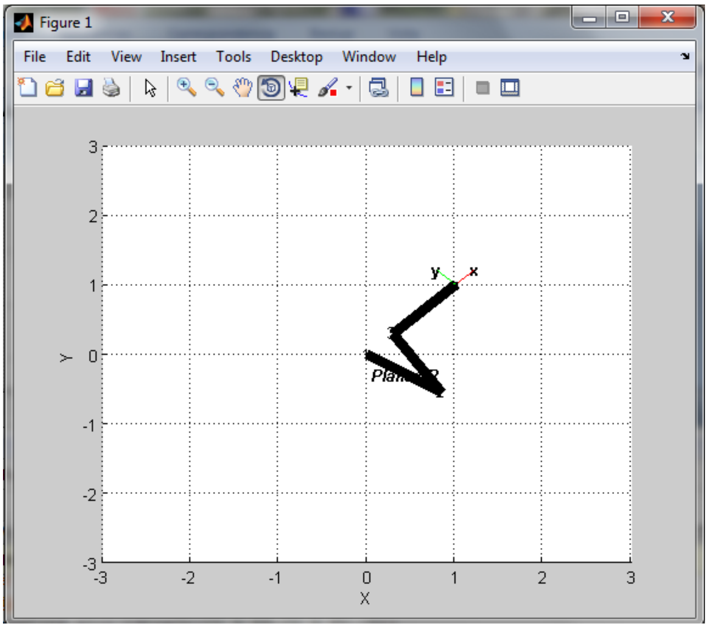
Thus, for each solution for θ_1, there is one (unique) solution for θ_2 and θ_3.

The above formulas can be implemented as a function in MATLAB as follows:

```
function theta = ikine3r(l, conf, sigma)
% IKINE3R solves the inverse kinematics for a 3R planar robot
%
%   [THETA1 THETA2 THETA3] = ikine3r([l1 l2 l3], [X Y PHI], SIGMA)
%
% Returns a vector of 3 angles corresponding to rotations about
% the three joints.
%
% sigma= +/-1 is a flag that gives us one of the two possible
% solutions
%

xx = conf(1)-l(3)*cos(conf(3));
yy = conf(2)-l(3)*sin(conf(3));

theta(1)= atan2(yy,xx)+...
    sigma*acos((l(1)*l(1)+xx*xx+yy*yy-l(2)*l(2))/...
                (2*l(1)*sqrt(xx*xx+yy*yy)));
theta(2)=atan2(yy-l(1)*sin(theta(1)),...
                xx-l(1)*cos(theta(1)))-theta(1);
theta(3)=conf(3)-theta(1)-theta(2);

end

%%% Now, we can plot the two possible solutions for a
%%% given pose of the end-effector.

view(0,90);
ThreeLink.plot(ikine3r([1 1 1],[1 1 pi/4],1));
```




```
view(0,90);
ThreeLink.plot(ikine3r([1 1 1],[1 1 pi/4],-1));
```



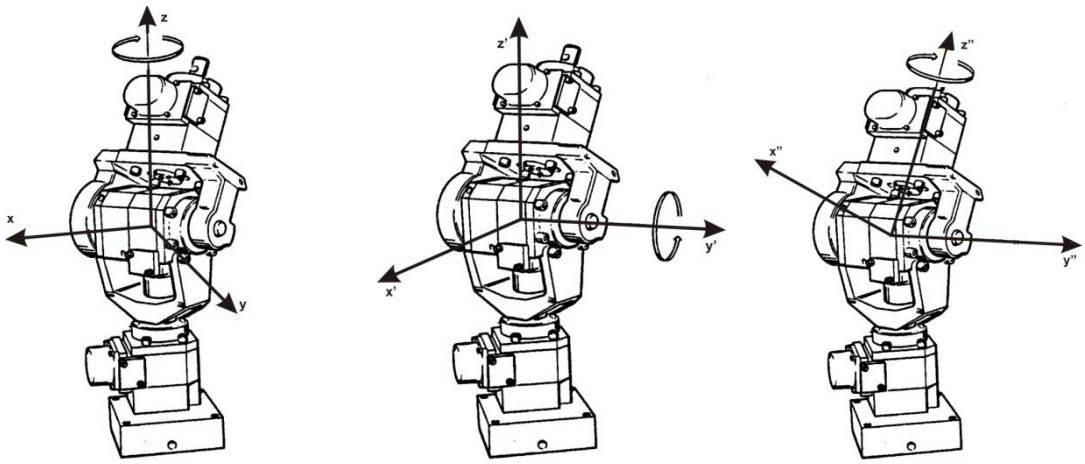
%%

To know more:

R. L. Williams and B. H. Shelley, “Inverse kinematics for planar parallel manipulators,” Proceedings of DETC ’97, 1997 ASME Design Technical Conferences, September 14-17, 1997, Sacramento, USA.

PROBLEM 5

Consider the hand of a decoupled robot. It consists of three consecutive revolute joints whose axes intersect in a point as shown below.



1) Obtain the DH parameters for this part of the robot and build it using the Robotics Toolbox.

2) Solve its forward kinematics for the case in which the three angles are $\pi/6$.

3) Solve its inverse kinematics for the pose given by the homogeneous transformation

$$\begin{pmatrix} 0.2549 & -0.9513 & -0.1736 & 0 \\ 0.9659 & 0.2588 & 0 & 0 \\ 0.0449 & -0.1677 & 0.9848 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Is the solution unique?

4) What happens when the angle for the second revolute joint is zero?

SOLUTION

The forward kinematic of this 3R robot can be expressed as

```
T03 = trotz(theta_1)*troty(theta_2)*trotz(theta_3)
```

Alternatively, we can express all rotations about the z-axis as follows

```
T03 = trotz(theta_1)*trotx(-pi/2)*
      trotz(theta_2)*trotx(pi/2)*
      trotz(theta_3)
```

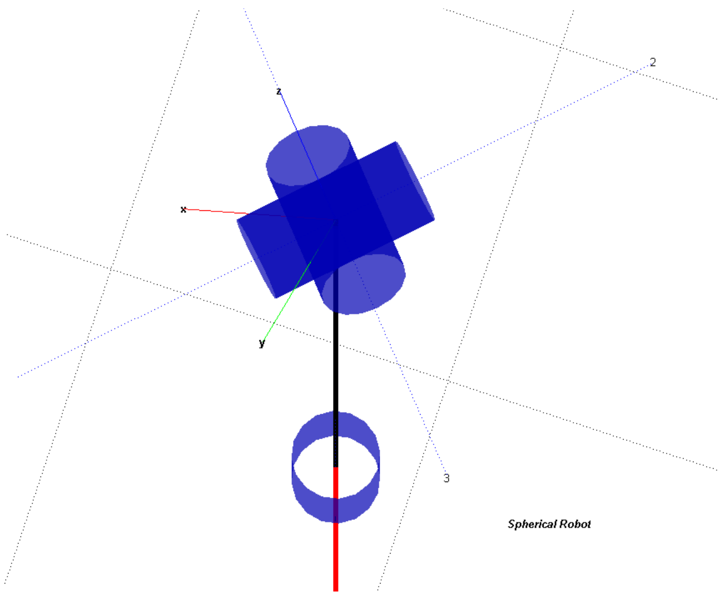
From which we can easily deduced that its DH parameters are:

Joint/Link i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	θ_1	0	0	$-\pi/2$
2	θ_2	0	0	$\pi/2$
3	θ_3	0	0	0

The implementation using the Robotics Toolbox follows:

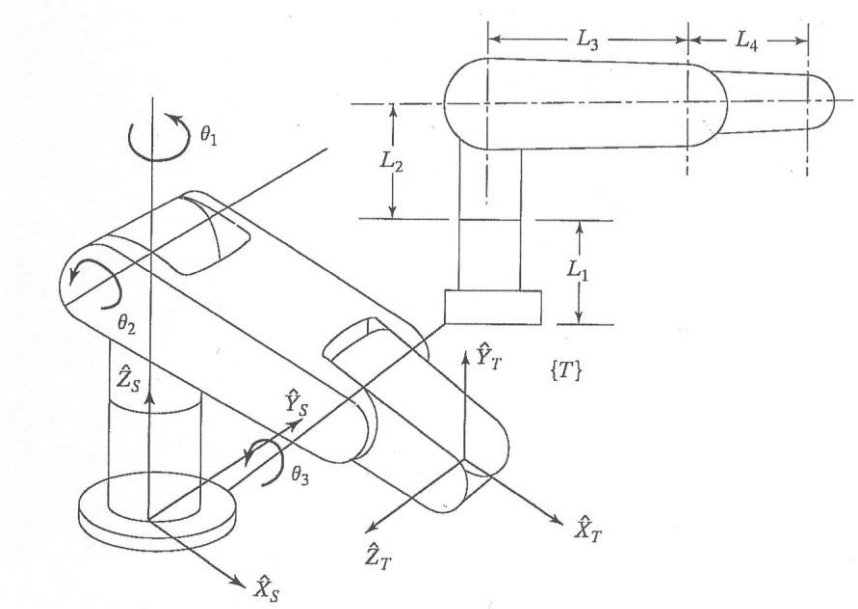
```
%%% First we clean the workspace

clc;
clear;

%%% QUESTION 1:
%%%
%%% We define the robot links from its DH parameters

L(1) = Link([0 0 0 -pi/2]);
L(2) = Link([0 0 0  pi/2]);
L(3) = Link([0 0 0     0]);

SphericalRobot = SerialLink(L);

%%% QUESTION 2:
%%%
%%% We solve the forward kinematics for the required joint angles

tr1 = SphericalRobot.fkine([pi/6 pi/6 pi/6]);

%%% It can be checked that the solution is the same if we
%%% directly compute

tr2 = trotz(pi/6)*troty(pi/6)*trotz(pi/6)

%%% QUESTION 3:
%%%
%%% The inverse kinematics for the given pose can be easily solved by
%%% realizing that these angles correspond to the Euler angles. Then,
```

```
T13= [ 0.2549 -0.9513 -0.1736 0;
       0.9659  0.2588  0      0;
       0.0449 -0.1677  0.9848 0;
       0       0       0      1];

q1 = tr2eul(T13);

%%%
%%% The set of Euler angles for a given orientation is not unique.
%%% There are always two solutions (see Problem 2). The other
%%% solution is:

q2 = [q1(1)+pi -q1(2) q1(3)+pi];

%%% QUESTION 4:
%%%
%%% When the second join angle is zero the first and the third
%%% revolute axes are aligned. Their associated rotation angles
%%% become dependent. The robot is in a singularity because
%%% its inverse kinematics is not defined.
%%%
%%% NOTE:
%%%
%%% If we try to plot this robot, we will get an error. This is due
%%% to the fact that it only involves rotations and the program
%%% is unable to assign ranges for the axes. To overcome this
%%% problem, we can introduce a translation in the first link.

L(1) = Link([0 1 0 -pi/2]);

SphericalRobot = SerialLink(L);
SphericalRobot.name ='Spherical Robot';

SphericalRobot.plot([pi/4 pi/4 pi/4 ]);
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 6
Given the following 3R robot



.

where L_1=4, L_2=3, L_3=2, and L_4=1.

1) Derive the DH parameters and the neighbouring homogeneous transformation matrices ${}^{i-1}_{i}T$, for i=1,2,3, as functions of the joint angles.
2) Implement the forward kinematics, that is, ${}^{0}_{3}T(\theta_1, \theta_2, \theta_3)$.

3) Calculate the result for the following joint angles: (0, 0,0), (0, π/2, 0), and (0, π/2, π/6).

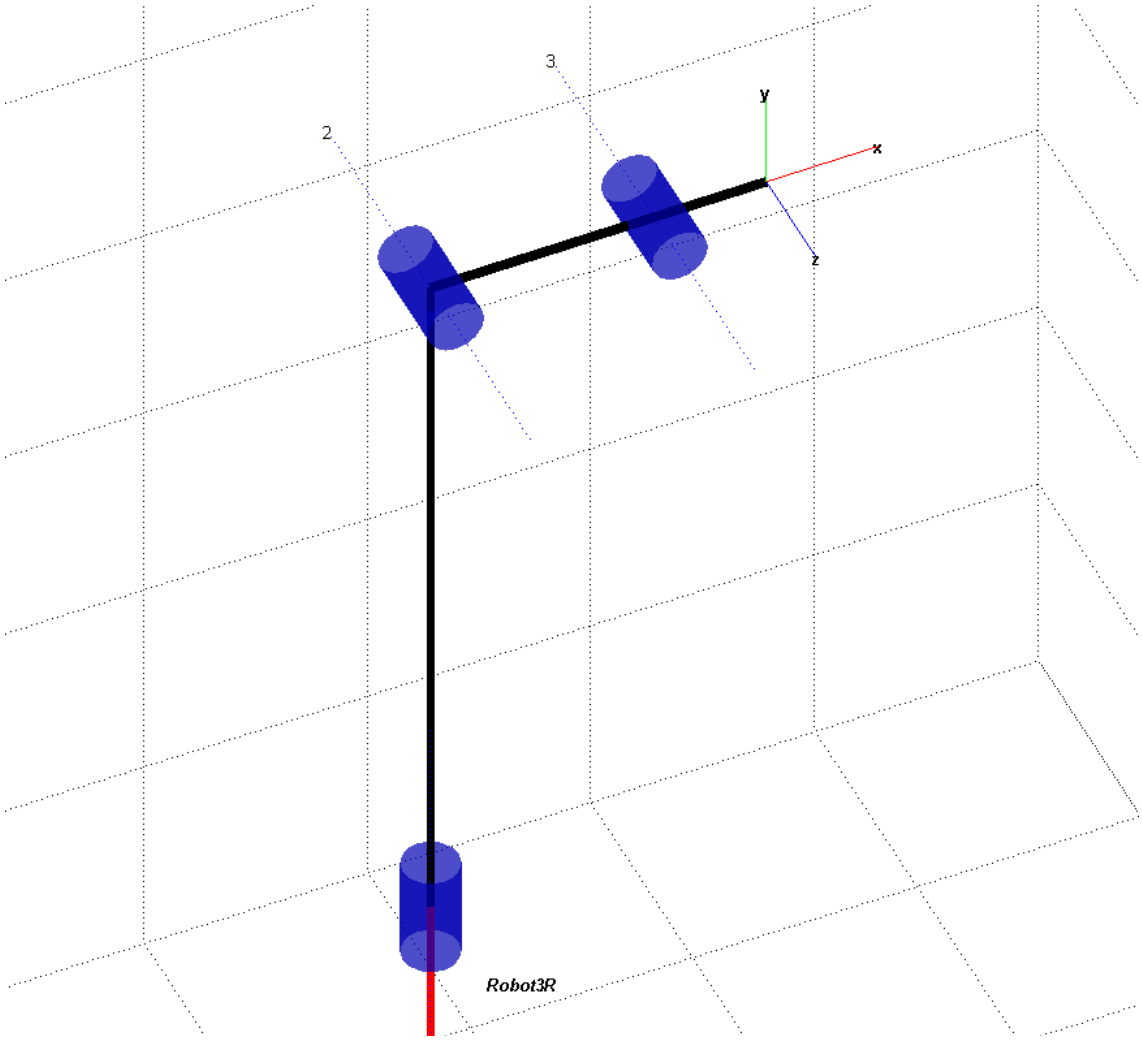
4) Build the robot using the obtained DH parameters and compare the results using the forward kinematics associated method with that obtained multipliying the neighboring matrices.

SOLUTION

According to the assigment of axes in the above drawing, the DH parameters are:

Joint/Link i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	θ_1	$L_1 + L_2$	0	$\pi / 2$
2	θ_2	0	L_3	0
3	θ_3	0	L_4	0

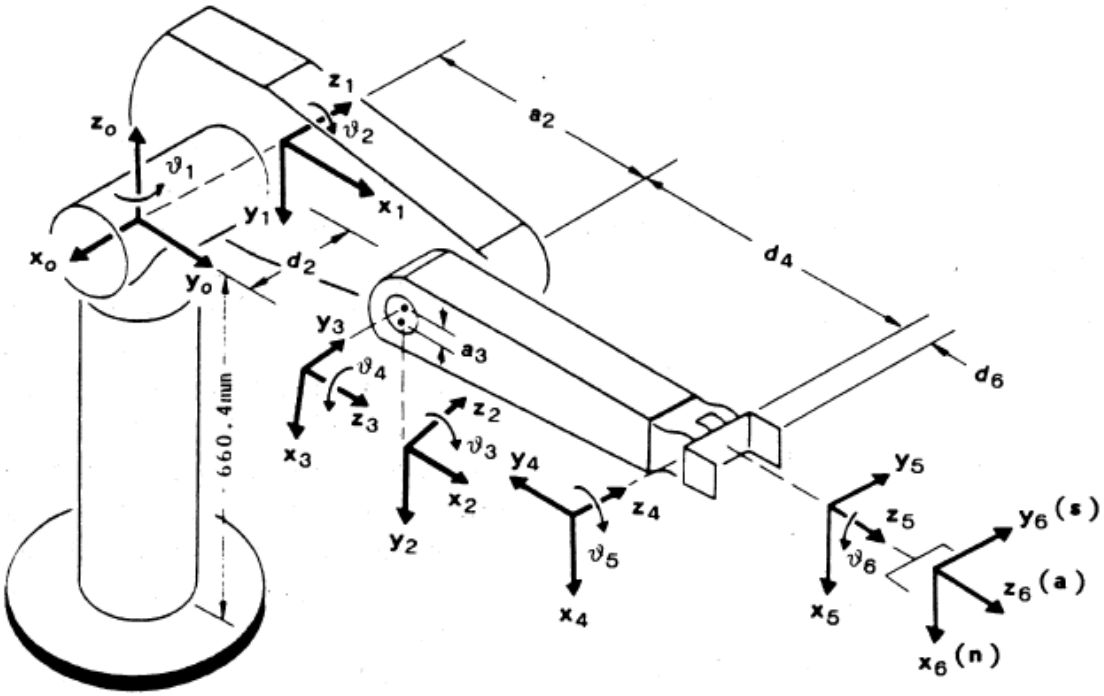
```
%%% First we clean the workspace

clear;
clc;

%%% We define the constants

L1 = 4;
L2 = 3;
L3 = 2;
L4 = 1;

%%% We introduce three symbols denoting the joint angles

a1 = sym('a1');
a2 = sym('a2');
a3 = sym('a3');

T01 = trotz(a1)*transl(0, 0, L1+L2)*trotx(pi/2);
T12 = trotz(a2)*transl(L3, 0, 0);
T23 = trotz(a3)*transl(L4, 0, 0);

T03 = T01*T12*T23;

T1 = subs(T03, {a1, a2, a3}, {0, 0, 0});
T2 = subs(T03, {a1, a2, a3}, {0, pi/2, 0});
T3 = subs(T03, {a1, a2, a3}, {0, pi/2, pi/6});

%%% We build the robot using DH parameters

L(1) = Link([0 L1+L2   0  pi/2]);
L(2) = Link([0     0  L3     0]);
L(3) = Link([0     0  L4     0]);

Robot3R = SerialLink(L);

%%% Now we plot the robot to check if it has been correctly built

Robot3R.name = 'Robot3R';
Robot3R.plot([0 0 0]);
```




```
TT1 = Robot3R.fkine([0 0 0]);
TT2 = Robot3R.fkine([0 pi/2 0]);
TT3 = Robot3R.fkine([0 pi/2 pi/6]);

%%% It can be checked that Ti is identical to TTi

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 7

Consider the PUMA 560 robot shown in the following drawing.



Joint/Link i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]	Range [degrees]
1	$\theta_1 + \pi/2$	0	0	$-\pi/2$	-160 to +160
2	θ_2	0.149	0.4318	0	-225 to +45
3	$\theta_3 + \pi/2$	0	0.0203	$\pi/2$	-45 to +225
4	θ_4	0.433	0	$-\pi/2$	-110 to +170
5	θ_5	0	0	$\pi/2$	-100 to +100
6	θ_6	0.0562	0	0	-266 to +266

1) Model it using the Robotics Toolbox and compare the result with the built-in PUMA 560 robot.

2) Find the joint angles that allow the robot's end-effector to attain the pose, given in homogeneous coordinates, by transl(0.1, 0.25, -0.5)*trotx(pi)*troty(pi) with respect to the world's reference frame (Hint: use the *ikine6s* function).

SOLUTION

To know more:

P.I. Corke and B. Armstrong-Helouvry, “A search for consensus among model parameters reported for the PUMA 560 robot,” *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1608-1613, 1994.

PROBLEM 8

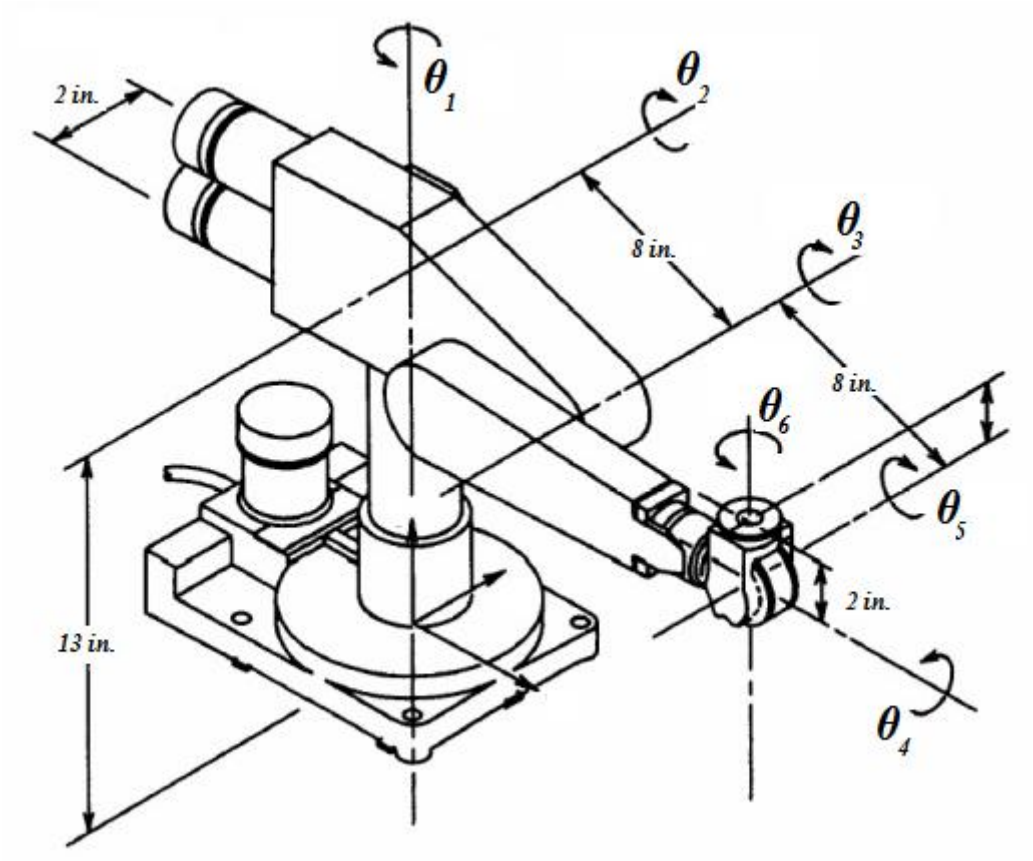
Consider the PUMA 260 manipulator shown in the following drawing.

1) Derive the complete set of forward kinematics equations for the center of the wrist by establishing the DH frames, finding and taking the product of the corresponding homogenous transformations.

2) Does it represent a decoupled robot? Why?

3) If d_3 is set to 0, how would you solve its inverse kinematics? Just sketch the idea.

4) Using the Robotics Toolbox compute the workspace of the end-effector – i.e. move each joint through its range of motion and plot the isometric view of the total workspace of the end-effector as well as the three Cartesian projections of this workspace onto the XY, YZ and XZ planes.



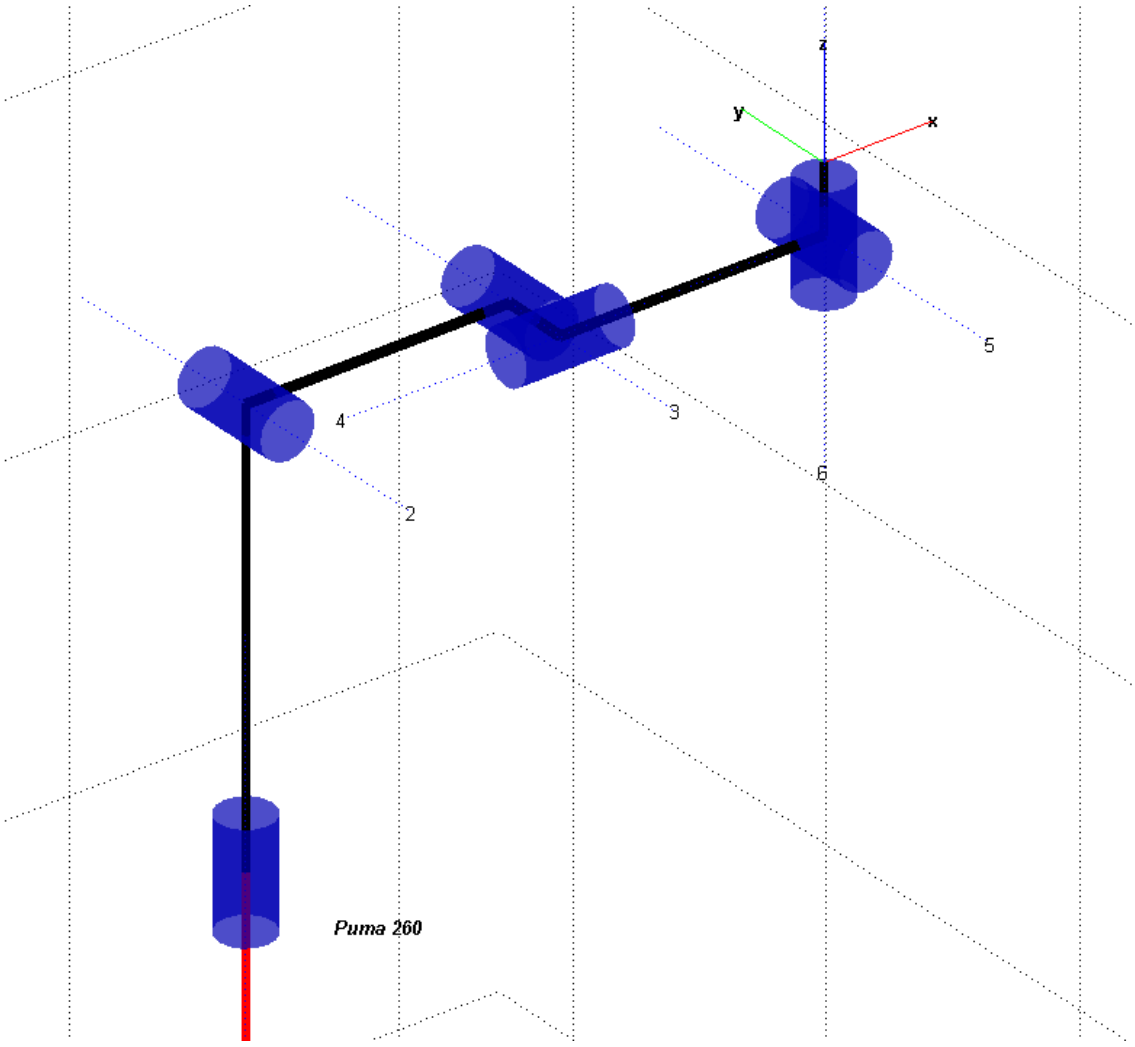
SOLUTION

```
%%% QUESTION 1:
%%%
%%% The sought DH parameters are:
```

Joint/Link i	θ_i [rad]	d_i [inch]	a_i [inch]	α_i [rad]
1	θ_1	13	0	$-\pi/2$
2	θ_2	0	8	0
3	$\theta_3+\pi/2$	-2	0	$\pi/2$
4	θ_4	8	0	$-\pi/2$
5	$\theta_5+\pi/2$	0	0	$\pi/2$
6	θ_6	2	0	0

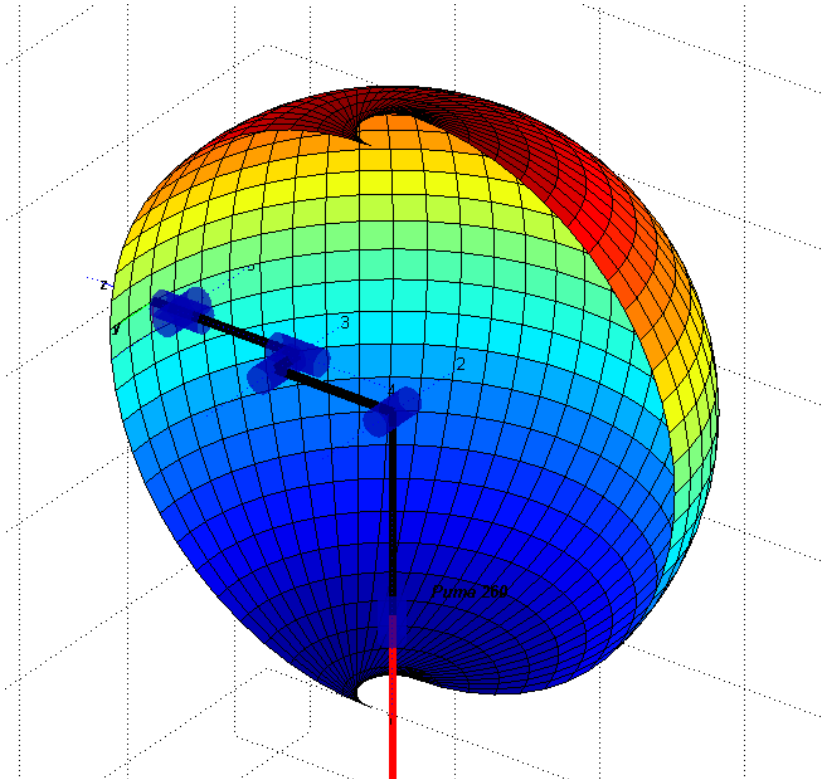
```
%%% Now, we create the model but first we clean the workspace

clc;
clear;

%%% The standard DH-parameters of a link represent the transformation
%%%
%%% rotz(theta_i)*trans(0,0,d_i)*trans(a_i,0,0)*rotx(alpha_i)

a1 = sym('a1');
a2 = sym('a2');
a3 = sym('a3');
a4 = sym('a4');
a5 = sym('a5');
a6 = sym('a6');

T01 = trotz(a1)*transl(0,0,13)*trotx(-pi/2);
T12 = trotz(a2)*transl(8,0,0);
T23 = trotz(a3+pi/2)*transl(0,0,-2)*trotx(pi/2);
T34 = trotz(a4)*transl(0,0,8)*trotx(-pi/2);
T45 = trotz(a5-pi/2)*trotx(pi/2);
T56 = trotz(a6);

%%% Forward kinematics

T06 = T01*T12*T23*T34*T45*T56

%%% Now, to substitute the symbols by numerical values, we can use the
%%% 'subs' function. For example, if we want to set all variables to
%%% zero, we can execute the following substitution:

subs(T06, {a1, a2, a3, a4, a5, a5}, {0, 0, 0, 0, 0, 0})

%%% Alternatively we can build the robot using the SerialLink function
```

```
L(1) = Link([0 13 0 -pi/2]);
L(2) = Link([0 0 8 0]);
L(3) = Link([0 -2 0 pi/2]);
L(4) = Link([0 8 0 -pi/2]);
L(5) = Link([0 0 0 pi/2]);
L(6) = Link([0 0 0 0]);

L(3).offset = pi/2;
L(5).offset = -pi/2;

Puma260 = SerialLink(L);
Puma260.name = 'Puma 260';
Puma260.plot([0 0 0 0 0 0]);
```




```
%%% We can obtain the general expression for the forward kinematics

Puma260.fkine([a1 a2 a3 a3 a4 a5]);

%%% QUESTION 2:
%%%
%%% The robot is decoupled because the last three revolute joint axes
%%% intersect in a point. They implement a spherical wrist.
```

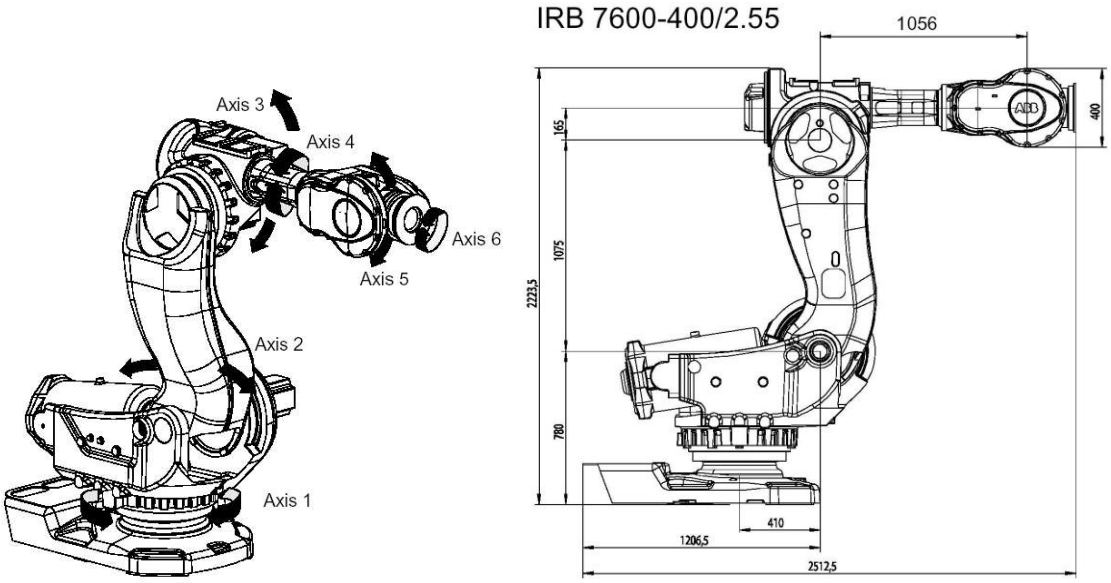
```
%%% QUESTION 3:
%%%
%%%

%%% QUESTION 4:
%%%
%%% To obtain the workspace, we can fix a3 and vary a1 and a2 to
%%% calculate the end-effector's position and plot it using the
%%% surf function.

Puma260.plot([0 0 0 0 pi/2 0]);

xlabel('X (in inch)');
ylabel('Y (in inch)');
zlabel('Z (in inch)');
title('Workspace of Puma 260');

hold on;

N=30;

for i= 1:N+1
    for j= 1:N+1
        TR = Puma260.fkine([pi*(i-1)/N pi*(j-1)/N+pi/2 0 0 pi/2 0]);
        SURF(i,j,:) = TR(:,4);
    end
end

surf(SURF(:,:,1), SURF(:,:,2), SURF(:,:,3));
```



%%
%%
%%

PROBLEM 9

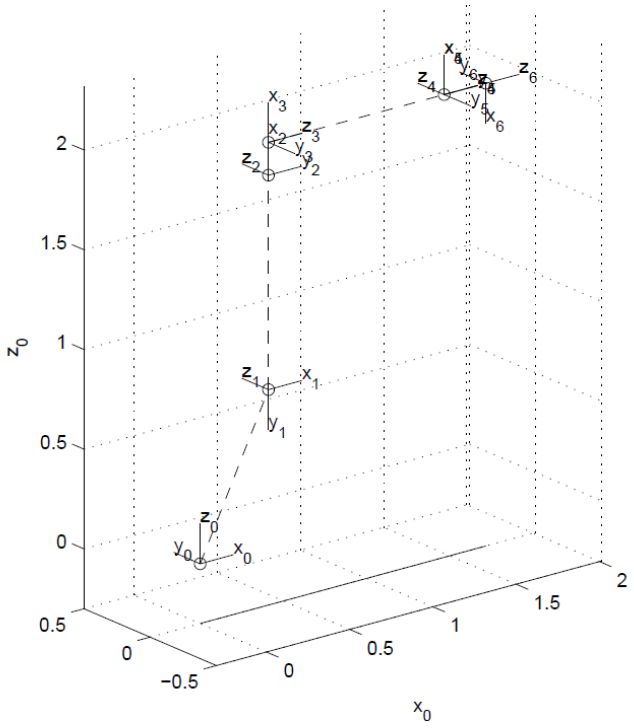
Consider the ABB industrial robot IRB 7600 shown below



Obtain its DH parameters, model its kinematics using the Robotics Toolbox, and solve its forward kinematics.

SOLUTION

First we have to assign a reference frame to each joint according to the DH convention.



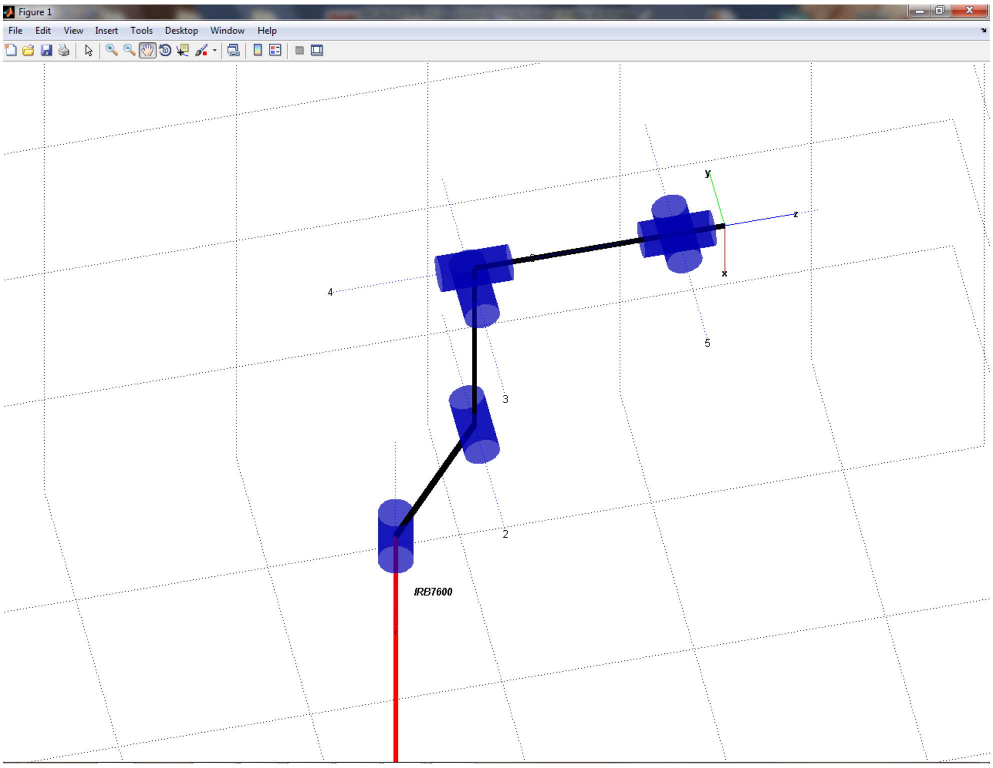
According to this frame assigment the standard DH parameters are the following:

Joint/Link i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	θ_1	0.78	0.41	$-\pi/2$
2	$\theta_2 - \pi/2$	0	1.075	0
3	θ_3	0	0.165	$-\pi/2$
4	θ_4	1.056	0	$\pi/2$
5	θ_5	0	0	$-\pi/2$
6	$\theta_6 + \pi$	0.25	0	0

Now, we can model this robot using the Robotics Toolbox.

```
%%% First we clean the workspace

clear;
clc;

%%% Create links

L(1) = Link([0  0.78   0.41    -pi/2]);
L(2) = Link([0  0      1.075       0]);
L(3) = Link([0  0      0.165   -pi/2]);
L(4) = Link([0  1.056  0        pi/2]);
L(5) = Link([0  0      0       -pi/2]);
L(6) = Link([0  0.25   0           0]);

L(2).offset = -pi/2;
L(6).offset = pi;

%%% Joint limits

L(1).qlim = pi/180*[-180 180];
L(2).qlim = pi/180*[-60 85];
L(3).qlim = pi/180*[-180 60];
L(4).qlim = pi/180*[-300 300];
L(5).qlim = pi/180*[-100 100];
L(6).qlim = pi/180*[-300 300];

%%% Create robot and plot it

IRB7600 = SerialLink(L);

IRB7600.name = 'IRB7600';
IRB7600.plot([0 0 0 0 0 0]);

%%% To solve the forward kinematics in the general case, we have to
%%% introduce the six joint angles as symbols
```

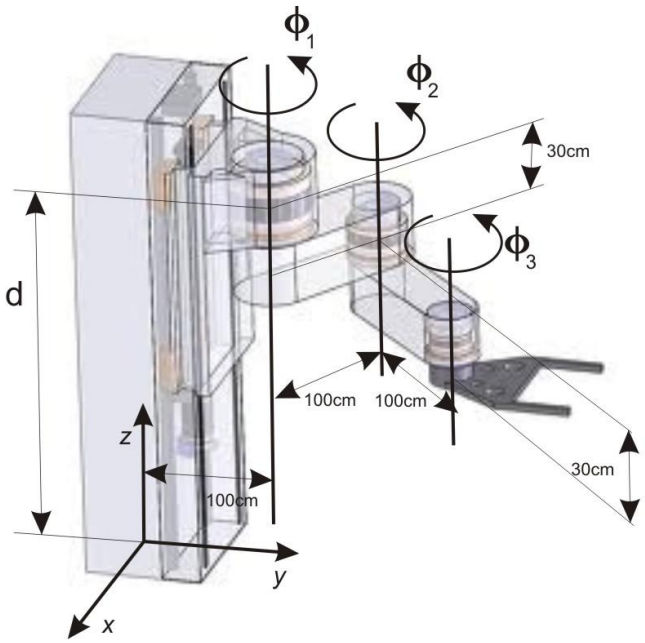
```
a1 = sym('a1');
a2 = sym('a2');
a3 = sym('a3');
a4 = sym('a4');
a5 = sym('a5');
a6 = sym('a6');

IRB7600.fkine([a1 a2 a3 a4 a5 a6])
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 10

Consider the following PRRR robot



1) Obtain its DH parameters and build it using the Robotics Toolbox.

2) Generate a joint-interpolated trajectory for its end-effector from the initial pose defined by transl(0,200,0) to the final pose defined by transl(50,50,200)*rotz(pi/2) and simulate it using the Robotics Toolbox (distances are given in centimetres). To this end, use the function implemented in Problem 4(4). How many such trajectories exist?

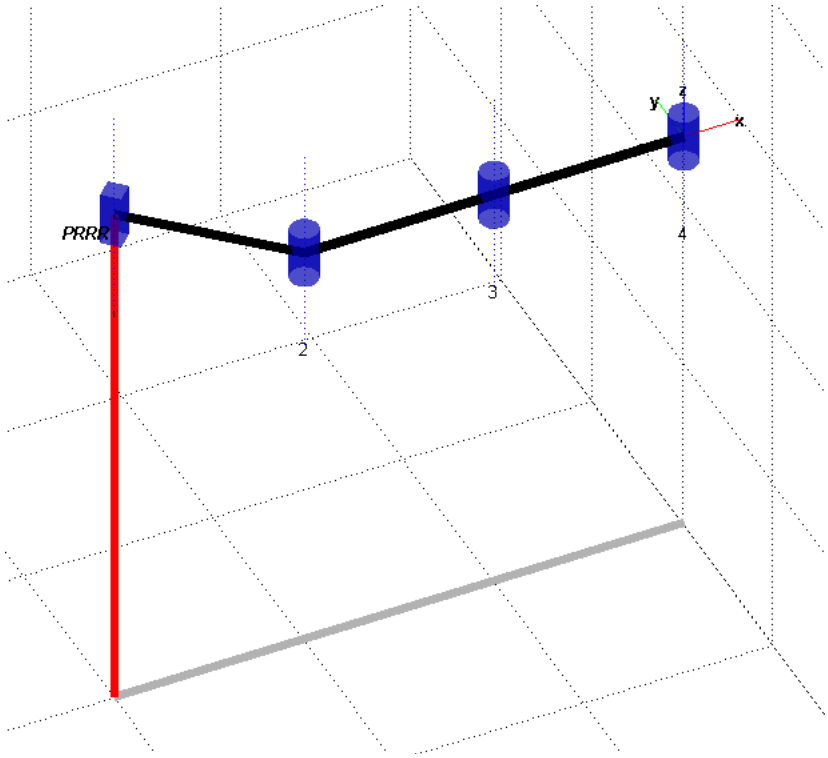
3) Generate a straight-line trajectory for the same initial and final poses given above and simulate it. How many such trajectories exist?

SOLUTION

When the consecutive axes are parallel, there are infinite solutions for the DH parameters. In these cases, it is advisable to chose them so that as many DH parameters as possible are zero. Thus, a valid set of these parameters following this criterium is:

Joint/Link i	θ_i [rad]	d_i [mm]	a_i [mm]	α_i [rad]
1	$\pi/2$	d-60	100	0
2	θ_2	0	100	0
3	θ_3	0	100	0
4	θ_4	0	0	0

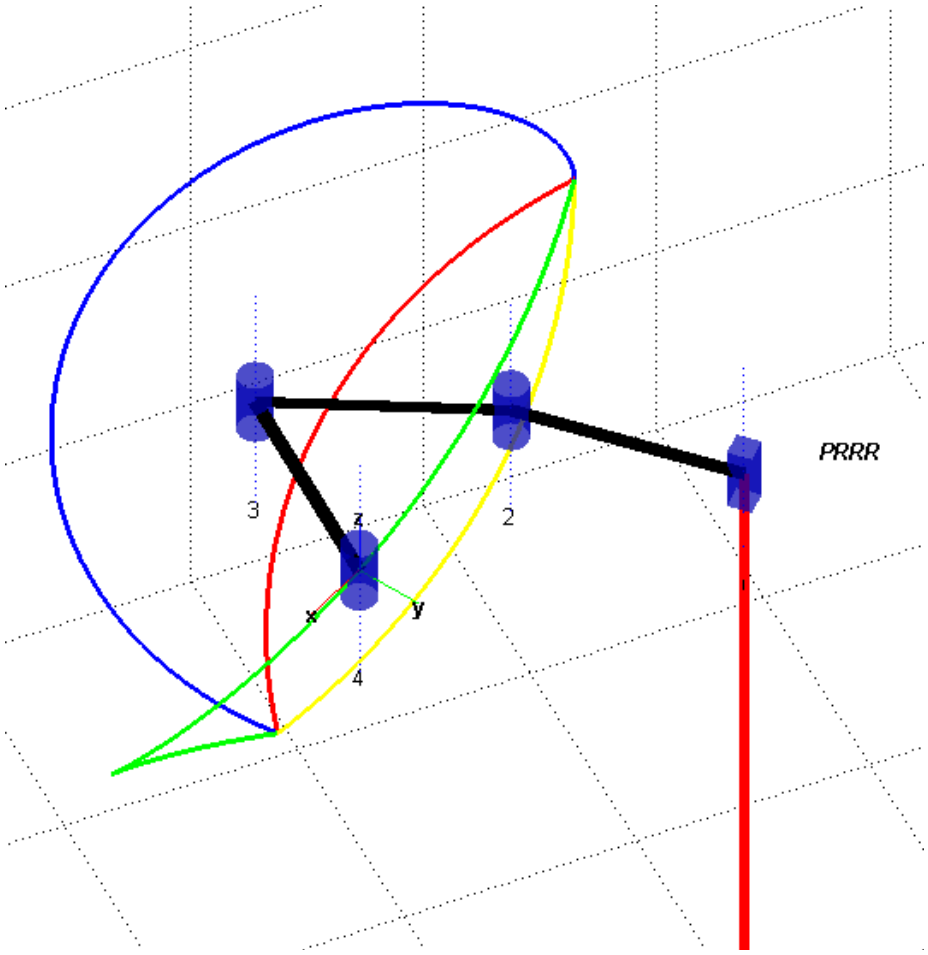
```
%%% First, we clean the workspace

clear
clc

%%% QUESTION 1:

%%% Link generation

L(1) = Link([pi/2,   0, 100, 0, 1]);
L(2) = Link([   0,   0, 100, 0]);
L(3) = Link([   0,   0, 100, 0]);
L(4) = Link([   0,   0,   0, 0]);

L(1).offset = -60;

%%% Robot generation

PRRR = SerialLink(L);
PRRR.name = 'PRRR';

%%% We plot the robot to verify that we have correctly generated it.

PRRR.plot([0 0 0 0])
```




```
%%% QUESTION 2:

%%% We define the initial and final hand poses

TINI = transl(0,200,0);
TEND = transl(50,50,200)*trotz(pi/2);

%%% Before generating the trajectory, we can try to obtain the joint
%%% angles in the initial configuration using the ikine function

q0 = PRRR.ikine(TINI, [0 0 0 0], [1 1 1 0 0 1]);

%%% It does not converge to a solution. We have to change the initial
%%% guess but, after some trial and error, we observe that it is
%%% difficult to obtain a good initial guess. Thus, to overcome this
%%% problem we can use the function implemented in Problem 4(4)

T01INI = trotz(pi/2)*transl(100,0,0);
T01END = trotz(pi/2)*transl(100,0,200);

T14INI = inv(T01INI)*TINI;
T14END = inv(T01END)*TEND;

samples=100;

%%% TRAJECTORY 1

qini1 = [T01INI(3,4) ikine3r([100 100 0],...
    [T14INI(1,4) T14INI(2,4) atan2(T14INI(1,2),T14INI(1,1))],-1)];
qend1 = [T01END(3,4) ikine3r([100 100 0],...
    [T14END(1,4) T14END(2,4) atan2(T14END(1,2),T14END(1,1))],-1)];
Q1 = jtraj(qini1, qend1, samples);
TRAJ1 = PRRR.fkine(Q1);
```

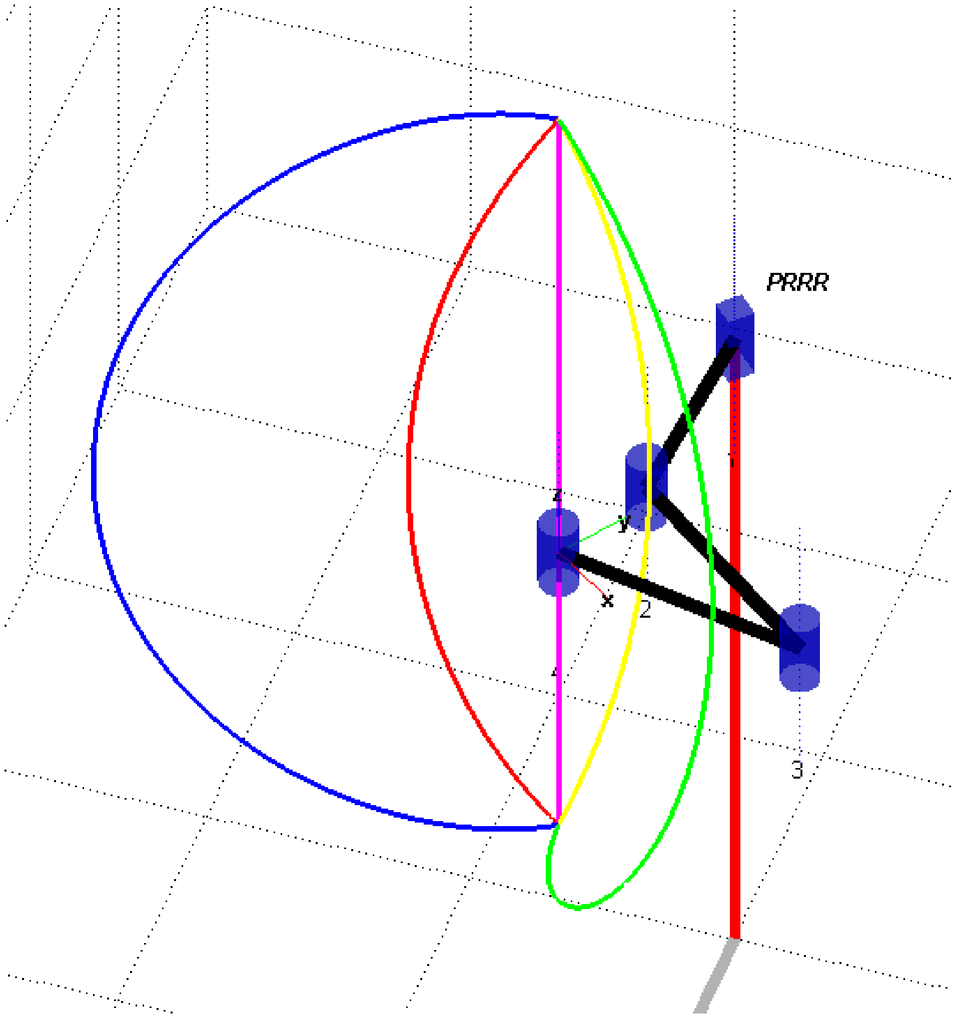
```
for i=1:samples
    xx1(i) = TRAJ1(1,4,i);
    yy1(i) = TRAJ1(2,4,i);
    zz1(i) = TRAJ1(3,4,i);
end

%%% TRAJECTORY 2

qini2 = [T01INI(3,4) ikine3r([100 100 0],...
    [T14INI(1,4) T14INI(2,4) atan2(T14INI(1,2),T14INI(1,1))],1)];
qend2 = [T01END(3,4) ikine3r([100 100 0],...
    [T14END(1,4) T14END(2,4) atan2(T14END(1,2),T14END(1,1))],1)];
Q2 = jtraj(qini2, qend2, samples);
TRAJ2 = PRRR.fkine(Q2);

for i=1:samples
    xx2(i) = TRAJ2(1,4,i);
    yy2(i) = TRAJ2(2,4,i);
    zz2(i) = TRAJ2(3,4,i);
end

%%% TRAJECTORY 3

qini3 = [T01INI(3,4) ikine3r([100 100 0],...
    [T14INI(1,4) T14INI(2,4) atan2(T14INI(1,2),T14INI(1,1))],1)];
qend3 = [T01END(3,4) ikine3r([100 100 0],...
    [T14END(1,4) T14END(2,4) atan2(T14END(1,2),T14END(1,1))],-1)];
Q3 = jtraj(qini3, qend3, samples);
TRAJ3 = PRRR.fkine(Q3);

for i=1:samples
    xx3(i) = TRAJ3(1,4,i);
    yy3(i) = TRAJ3(2,4,i);
    zz3(i) = TRAJ3(3,4,i);
end

%%% TRAJECTORY 4

qini4 = [T01INI(3,4) ikine3r([100 100 0],...
    [T14INI(1,4) T14INI(2,4) atan2(T14INI(1,2),T14INI(1,1))],-1)];
qend4 = [T01END(3,4) ikine3r([100 100 0],...
    [T14END(1,4) T14END(2,4) atan2(T14END(1,2),T14END(1,1))],1)];
Q4 = jtraj(qini4, qend4, samples);
TRAJ4 = PRRR.fkine(Q4);

for i=1:samples
    xx4(i) = TRAJ4(1,4,i);
    yy4(i) = TRAJ4(2,4,i);
    zz4(i) = TRAJ4(3,4,i);
end

hold on;

plot3(xx1, yy1, zz1, 'Color', [1 0 0], 'LineWidth',2);
plot3(xx2, yy2, zz2, 'Color', [0 1 0], 'LineWidth',2);
plot3(xx3, yy3, zz3, 'Color', [0 0 1], 'LineWidth',2);
plot3(xx4, yy4, zz4, 'Color', [1 1 0], 'LineWidth',2);
```

```
PRRR.plot(Q1);
PRRR.plot(Q2);
PRRR.plot(Q3);
PRRR.plot(Q4);
```




```
%%% QUESTION 3:

TC = ctraj(TINI, TEND, samples);

for i=1:samples
    T01(:,:,i) = trotz(pi/2)*transl(100,0,TC(3,4,i));
    T14(:,:,i) = inv(T01(:,:,i))*TC(:,:,i);
    QTRAJ1(i,:) = [T01(3,4,i) ...
        ikine3r([100 100 0], ...
        [T14(1,4,i) T14(2,4,i) atan2(T14(1,2,i),T14(1,1,i))],-1)];
    QTRAJ2(i,:) = [T01(3,4,i) ...
        ikine3r([100 100 0], ...
        [T14(1,4,i) T14(2,4,i) atan2(T14(1,2,i),T14(1,1,i))],1)];
end

TRAJ5 = PRRR.fkine(QTRAJ1);
TRAJ6 = PRRR.fkine(QTRAJ2);

for i=1:samples
    xx5(i) = TRAJ5(1,4,i);
    yy5(i) = TRAJ5(2,4,i);
    zz5(i) = TRAJ5(3,4,i);
```

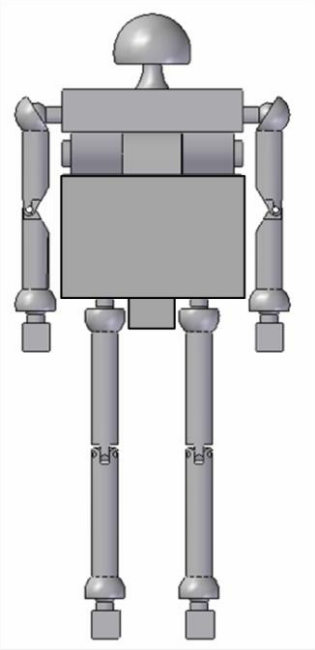
```
    xx6(i) = TRAJ6(1,4,i);
    yy6(i) = TRAJ6(2,4,i);
    zz6(i) = TRAJ6(3,4,i);
end

hold on;

plot3(xx5, yy5, zz5, 'Color', [0 1 1], 'LineWidth',2);
plot3(xx6, yy6, zz6, 'Color', [1 0 1], 'LineWidth',2);

PRRR.plot(QTRAJ1);
PRRR.plot(QTRAJ2);
```

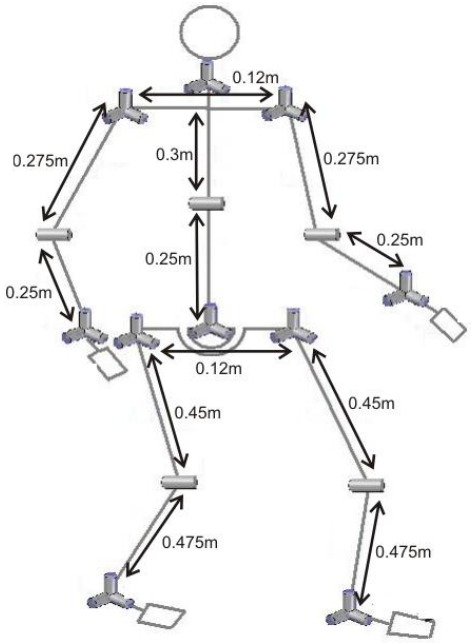



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

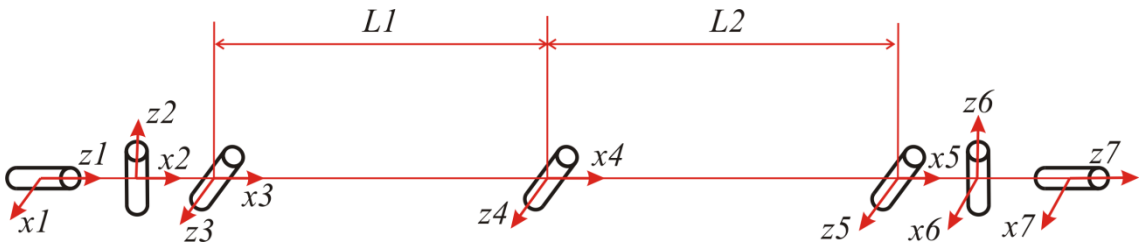
PROBLEM 11

A research group is enrolled in the implementation of a humanoid robot. The conceptual design appears below. It consists of 35 revolute joints but observe that 30 of them are organized in groups of three whose axes are intersecting in a single point to reproduce spherical motions (in the way as for the three last revolute joints of a decoupled robot), as shown below.





The four limbs of the robot have the same kinematic structure that can be schematically represented as follows:



1) Find the DH parameters for the limbs (*i.e.*, the DH---parameters for the kinematic chain in the above scheme).

2) Build the geometric model of one arm using the Robotics Toolbox and simulate an arbitrary joint-interpolated motion between two arbitrary configurations.

3) How many solutions to the inverse kinematics does the simulated kinematics chain have? Give a reasoned answer.

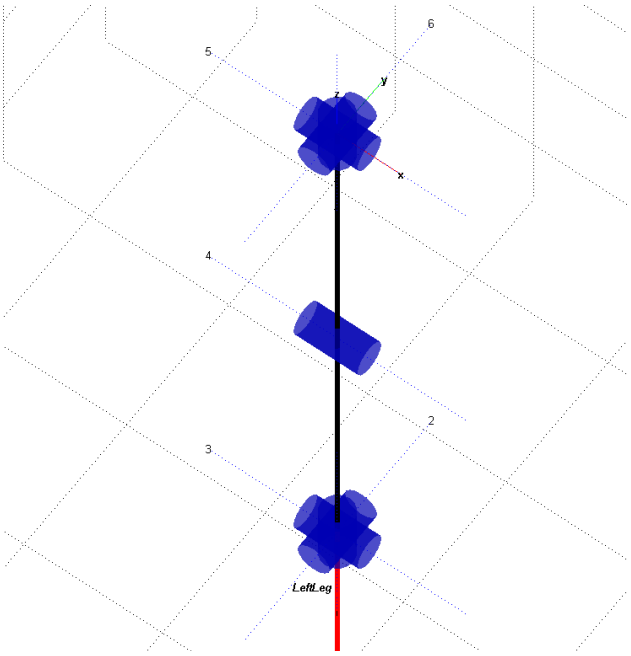
4) Assemble two legs and two arms so that the result corresponds to the designed robot in the case in which the robot's trunk joints are blocked.

SOLUTION

According to the frame assigment that appears in the above drawing, the standard DH parameters are the following:

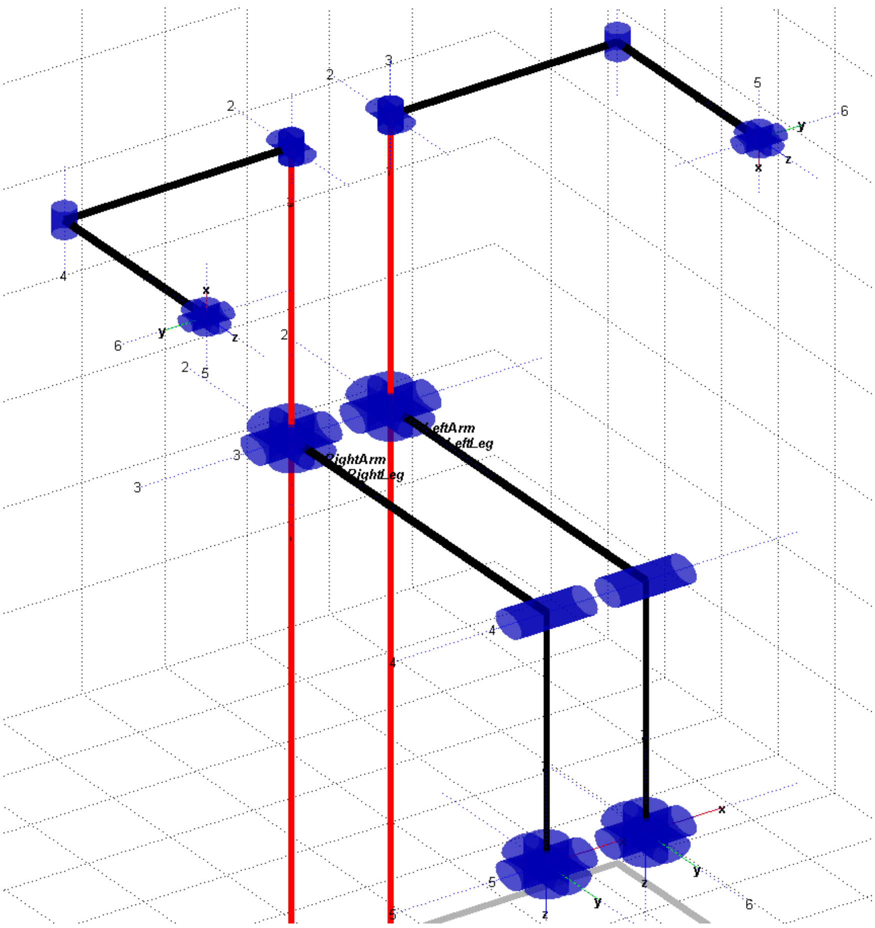
Joint/Link i	θ_i [rad]	d_i [mm]	a_i [mm]	α_i [rad]
1	θ_1	0	0	$\pi/2$
2	$\theta_2 + \pi/2$	0	0	$\pi/2$
3	θ_3	0	L1	0
4	θ_4	0	L2	0
5	θ_5	0	0	$-\pi/2$
6	$\theta_6 - \pi/2$	0	0	$-\pi/2$
7	θ_7	0	0	0

```
%%% First, we clean the workspace

clear;
clc;

L1 = 0.45;
L2 = 0.475;

L(1) = Link([0,  0,  0,  pi/2]);
L(2) = Link([0,  0,  0,  pi/2]);
L(3) = Link([0,  0, L1,     0]);
L(4) = Link([0,  0, L2,     0]);
L(5) = Link([0,  0,  0, -pi/2]);
L(6) = Link([0,  0,  0, -pi/2]);
L(7) = Link([0,  0,  0,     0]);

L(2).offset =  pi/2;
L(6).offset = -pi/2;

LeftLeg = SerialLink(L);
LeftLeg.name = 'LeftLeg';

RightLeg = SerialLink(L);
RightLeg.name = 'RightLeg';

%%% Observe that we have added an offset to theta_2 and theta_6
%%% so that, when we plot the kinematic chain for all actuated
%%% angles set to 0, the result corresponds to the drawing.
%%%
%%% We can plot the result to verify that we have correctly
%%% generated the model

LeftLeg.plot([0 0 0 0 0 0 0]);
```




```
%%% We can also generate the arms. We only need to change L1 and L2.

L1 = 0.275;
L2 = 0.25;

L(3) = Link([0,  0, L1,     0]);
L(4) = Link([0,  0, L2,     0]);

LeftArm = SerialLink(L);
LeftArm.name = 'LeftArm';

RightArm = SerialLink(L);
RightArm.name = 'RightArm';

%%% QUESTION 2:
%%%
%%% We choose the motion to occur over a period of 2 seconds in
%%% 5 ms time steps and an arbitrary initial and final configurations.

t = [0:0.005:2];
q_inicial= [0 0 0 0 0 0 0]';
q_final= [pi pi pi pi pi pi pi]';
Q = jtraj(q_inicial, q_final, t);

LeftArm.plot(Q);

%%% QUESTION 3:
%%%
%%% Six joints are theoretically sufficient to achieve any desired
%%% pose in Cartesian space. The analyzed kinematic chain has seven.
%%% Thus, it has infinite solutions to the inverse kinematic problem.
%%%
%%% QUESTION 4:

LeftLeg.base = transl([0.06 0 0]);
RightLeg.base = transl([-0.06 0 0]);

LeftArm.base = transl([0.06 0 0.55]);
RightArm.base = transl([-0.06 0 0.55]);
```

```
%%% We open a new window and plot the limbs. We assign angles to the
%%% actuators as if the robot were sitting but this is an arbitrary
%%% choice.

figure;
axis([-0.6 0.6 -0.6 0.6 -1 1]);
grid on;
hold on;

LeftLeg.plot([0 0 pi/2 pi/2 0 0 0])
RightLeg.plot([0 0 pi/2 pi/2 0 0 0])

LeftArm.plot([0 -pi/2 0 pi/2 0 0 0])
RightArm.plot([0 pi/2 0 pi/2 0 0 0])


title(‘Humanoid robot’)
ylabel('Y in meters')
xlabel('X in meters')
```



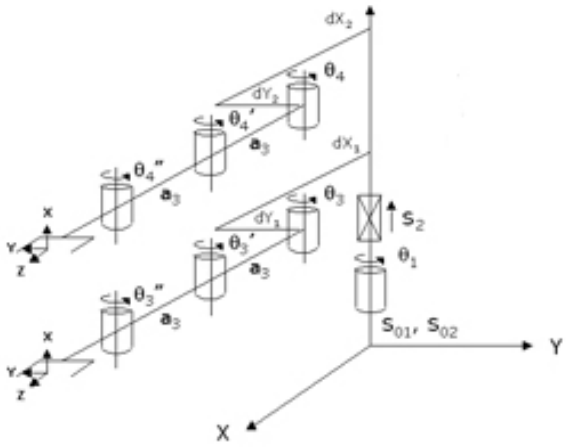
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To know more:

C. Venkatayogi, *Simulation of a Humanoid Robot*, Master Thesis, Russ College of Engineering and Technology of Ohio University, USA, 2007.

PROBLEM 12

The four-axis twin arm AR-W200CL-6-TRR-350 shown below consists of a center column that rotates about and moves up and down in Z. The column supports two arms that move in a linear radial direction while maintaining a fixed radial gripper orientation. This configuration is very popular in the semi-conductor market.



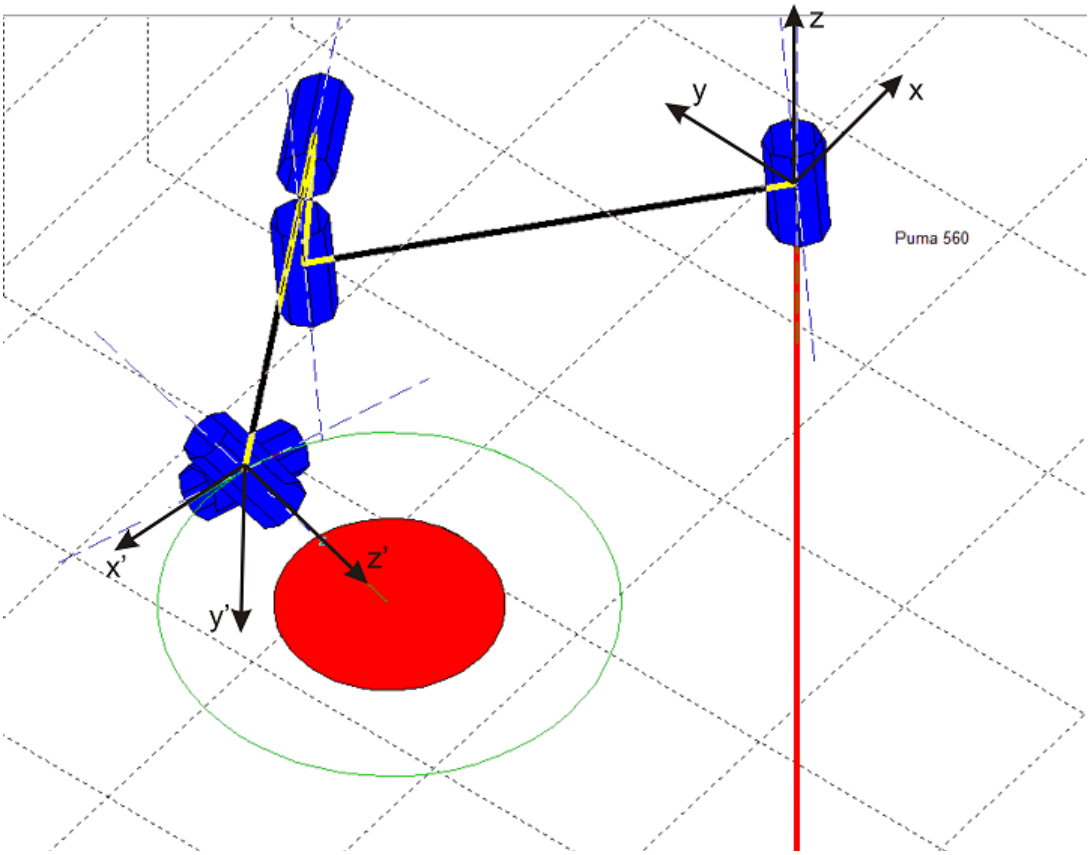
PROBLEM 13

A factory uses a laser to print different marks on cylinders. With the aim of increasing flexibility, this company decides to robotize this process using a PUMA 560 robot.

The maximum radius of the manufactured cylinders is 0.10 m and a feeder is used to place them parallel to the robot's xy-plane reference frame and centered at (-0.25, 0.25, -0.5) meters, as shown in the drawing below.

To correctly perform the marking task, the laser is used as the robot's end-effector in such a way that its z-axis is orthogonal to the cylindrical surface and its y-axis is pointing in the opposite direction of the z-axis of the robot's base reference frame (refer again to the drawing below).

To simulate the described task, using the Robotics Toolbox, generate a trajectory for the end-effector of a PUMA robot that executes a circular motion of radius 0.20 m around the manufactured cylinder.



SOLUTION

```
%%% First, we clean the workspace

clc;
clear;

%%% Initilize some variables: radius of the cylinder, number
%%% of samples for the trajectory, and location of the center
%%% of the cylinder.

radius=0.20;
n=200;
INI = transl(-0.25, 0.25, -0.5);

%%% Generation of the trajectory

for i=1:n
  TRAJ(:,:,i)= INI*trotx(-pi/2)*troty(2*pi*i/n)*transl(0, 0, -radius);
end

%%% Inicialization of the robot

mdl_puma560;

%%% Computation of the joint angles along the trajectory using the
%%% inverse kinematics

Q1= p560.ikine6s(TRAJ, 'run');

%%% Representation of the robot in the initial position

p560.plot(Q1(1,:));

%%% To visually verify the correctness of the trajectory, we can draw
%%% some elements of reference

hold on;

circle1 = circle([-0.25 0.25 -0.5], radius)
circle2 = circle([-0.25 0.25 -0.5], 0.5*radius)

plot3(circle1(1,:), circle1(2,:), circle1(3,:),...
 'g', 'LineWidth',1);

patch(circle2(1,:), circle2(2,:), circle2(3,:), 'r');

p560.plot(Q1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 14

Let us suppose that we want to generate a straight-line path between two poses for the end-effector of a PUMA 560 robot. These two poses are given in homogeneous coordinates, by:

transl(0.1, 0.25, -0.5)*trotx(pi)*troty(pi)

and

transl(0.25, 0.1, 0.5)*trotx(pi/2)*troty(pi/2)*trotz(pi/2)

Assuming that the path is approximated by 200 samples,

1) Obtain the path by interpolating the angle resulting from the angle-vector representation.

2) Obtain the path by interpolating Euler angles ZYZ.

3) Obtain the path using quaternions (the one implemented in the Robotics Toolbox).

4) Compare the results obtain in (1) and (3).

5) Simulate the motion of the end-effector. Repeat this for different solutions of the inverse kinematics.

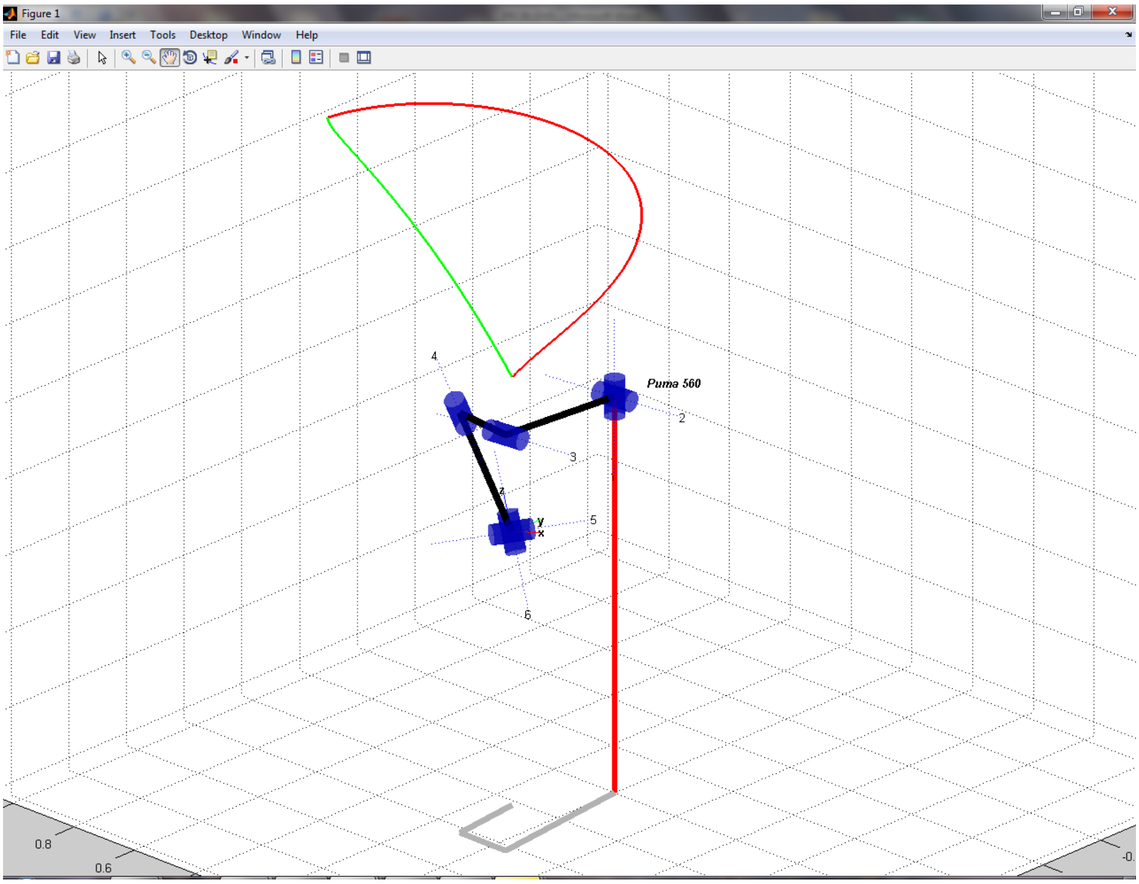
6) Let us suppose that the robot hand is holding a stick of length 0.6 meters aligned with the hand's z axis. Plot the trajectory followed by the extreme of the stick for the trajectory found in (1) and (3).

SOLUTION:

```
%%% First, we clean the workspace

clc;
clear;

%%% QUESTION 1:
%%%
%%% We define the initial and final poses and the transformation to be
%%% interpolated

TA = transl(0.1, 0.25, -0.5)*trotx(pi)*troty(pi);
TB = transl(0.25, 0.1, 0.5)*trotx(pi/2)*troty(pi/2)*trotz(pi/2);
TAB = inv(TA)*TB;

%%% Orientation to be interpolated

R = t2r(TAB);

%%% Angle-vector representation

[alpha vector] = tr2angvec(R);
```

```
%%% Interpolation based on the angle-vector representation

n=200;

for i=1:n+1
    TRAJ1(:,:,i) = TA*transl((i-1)/n*transl(TAB))*...
        angvec2tr(alpha*(i-1)/n, vector);
end

%%% QUESTION 2:
%%%
%%% Interpolation based on Euler angles

angles = tr2eul(R);

for i=1:n+1
    TRAJ2(:,:,i) = TA*transl((i-1)/n*transl(TAB))*...
        eul2tr(angles(1)*(i-1)/n, angles(2)*(i-1)/n, angles(3)*(i-
1)/n);
end

%%% QUESTION 3:
%%%
%%% Interpolation using quaternion spherical interpolation

TRAJ3 = ctraj(TA, TB, (0:1/n:1));

%%% QUESTION 4:
%%%
%%% It can be observed that TRAJ1 and TRAJ3 are identical

%%% QUESTION 5:
%%%
%%% We can create the robot and obtain the joint angles for the three
%%% trajectories

mdl_puma560;

Q1 = p560.ikine6s(TRAJ1, 'run');
Q2 = p560.ikine6s(TRAJ2, 'run');
Q3 = p560.ikine6s(TRAJ3, 'run');

p560.plot(Q1);
p560.plot(Q2);
p560.plot(Q3);


%%% QUESTION 6:

length = 0.6;

for i=1:n+1
    xx1(i) = TRAJ1(1,4,i)+ length*TRAJ1(1,3,i);
    yy1(i) = TRAJ1(2,4,i)+ length*TRAJ1(2,3,i);
    zz1(i) = TRAJ1(3,4,i)+ length*TRAJ1(3,3,i);
    xx2(i) = TRAJ2(1,4,i)+ length*TRAJ2(1,3,i);
    yy2(i) = TRAJ2(2,4,i)+ length*TRAJ2(2,3,i);
    zz2(i) = TRAJ2(3,4,i)+ length*TRAJ2(3,3,i);
end
```

```
figure;

p560.plot([0 0 0 0 0 0]);

hold on;

plot3(xx1, yy1, zz1, 'g', 'LineWidth',2);
plot3(xx2, yy2, zz2, 'r', 'LineWidth',2);


p560.plot(Q1);
p560.plot(Q2);
```




```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 16

Model a SCARA-like robot with a spherical wrist using the DH parameters given below.

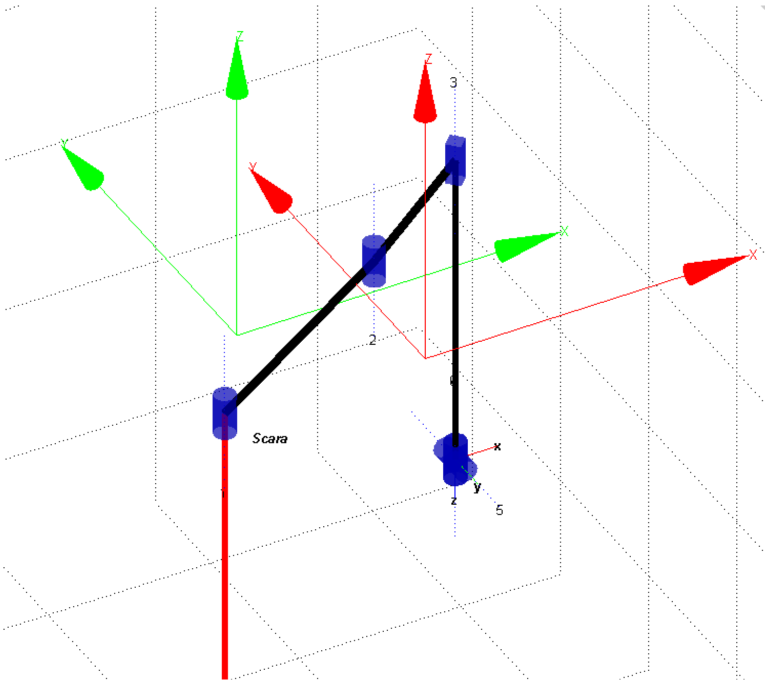
Joint/Link i	θ_i [rad]	d_i [m]	a_i [m]	α_i [rad]
1	θ_1	0.35	0.46	0
2	θ_2	0.25	0.25	π
3	0	d_i	0	0
4	θ_4	0	0	$\pi/2$
5	θ_5	0	0	$-\pi/2$
6	θ_6	0	0	0

Let us suppose that this robot is used for inspection. To this end, a camera is used as end-effector. For a given task, it is important to reorient the camera so that it is always pointing to a fixed point in the workspace. Then, let us suppose that the robot performs a straight line motion between the points (0.4, -0.4, 0.3) and (0.2, 0.3, 0.0). Find how the wrist should be reoriented along the trajectory so that the camera is always pointing to (0, 0, 0).

SOLUTION

```
%%% First we clean the workspace

clc;
clear;

%%% Now, we define the six links

L(1) = Link([0, 0.35, 0.46,    0, 0]);
L(2) = Link([0, 0.25, 0.25,   pi, 0]);
L(3) = Link([0,    0,    0,    0, 1]);
L(4) = Link([0,    0,    0, pi/2, 0]);
L(5) = Link([0,    0,    0,-pi/2, 0]);
L(6) = Link([0,    0,    0,    0, 0]);

Scara = SerialLink(L);

Scara.name = 'Scara';

%%% We define the intial and final locations

TRini = transl(0.4, -0.4, 0.3);
TRend = transl(0.2, 0.3, 0.0);

%%% We represent the robot and the locations to check if they have
%%% been correctly been defined
```

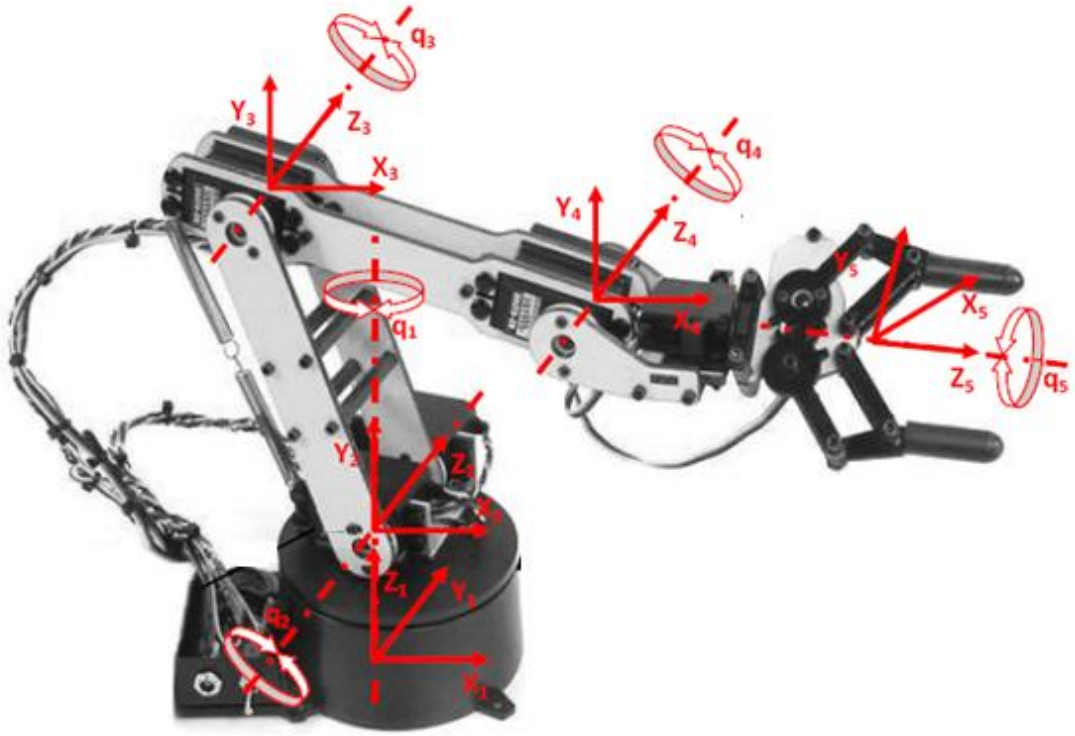
```
axis equal;

Scara.plot([0 0 1 0 0 0]);

hold on;

trplot(TRini, 'color', 'r', 'arrow');
trplot(TRend, 'color', 'g', 'arrow');
```




```
%%% We generate the desired trajectory by interpolating 100
%%% samples between the initial and final locations

samples = 100;

TRint = inv(TRini)*TRend;

for i=1:samples+1
    TRinter = TRini*transl((i-1)/samples*transl(TRint));
    ax = -TRinter(1,4);
    ay = -TRinter(2,4);
    az = -TRinter(3,4);
    a = [ax ay az]/norm([ax ay az]);
    o = [-a(2) a(1) 0];
    TRAJ(:,:,i) = TRinter*oa2tr(o,a);
end

Q = Scara.ikine(TRAJ);
Scara.plot(Q);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 17

Consider the Lynx 6 robot arm shown below.



This robot has 5 degrees of freedom. Thus, its gripper cannot attain any arbitrary orientation.

1) Despite it is a 5R robot, how can we still use the build-in function *ikine6s* to solve its inverse kinematics?

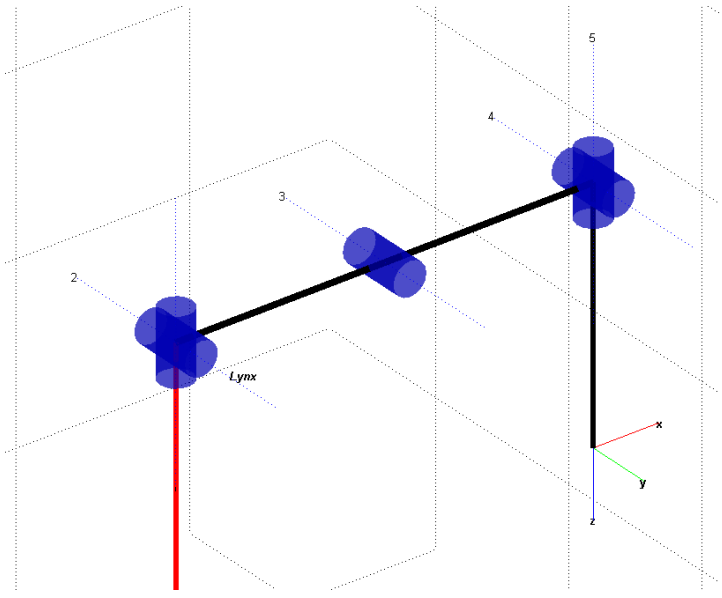
2) Compute the forward kinematics for all joint angles set to zero and obtain all other set of joint angles that take the robot hand to the same place up to rotations about the missing axis.

SOLUTION

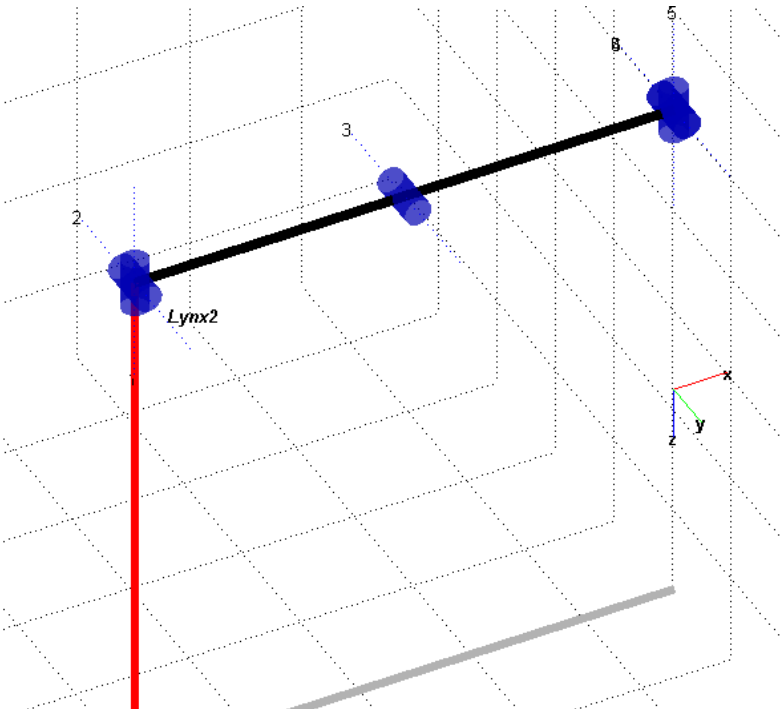
It can be checked that a set of valid DH parameters is:

Joint/Link i	θ_i [rad]	d_i [cm]	a_i [cm]	α_i [rad]
1	θ_1	0	0	$\pi/2$
2	θ_2	0	12	0
3	θ_3	0	12	0
4	θ_4	0	0	$\pi/2$
5	θ_5	14	0	0

```
%%% First we clean the workspace

clc;
clear;

%%% Now, we define the five links

L(1) = Link([0,  0,  0, pi/2]);
L(2) = Link([0,  0, 12,    0]);
L(3) = Link([0,  0, 12,    0]);
L(4) = Link([0,  0,  0, pi/2]);
L(5) = Link([0, 14,  0,    0]);

Lynx = SerialLink(L);
Lynx.name = 'Lynx';

%%% We represent the robot to check if it has correctly been defined

Lynx.plot([0 0 0 0 0]);
```



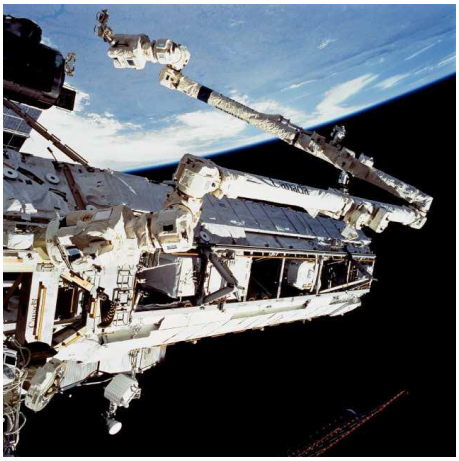
```
%%% Now, we introduce one extra degree of freedom to transform
%%% this robot into a 6R robot with a spherical wrist. To this
%%% end, the 14 cm translation along the tool axis has also
%%% to be incorporated to the tool transformation.

L(1) = Link([0, 0, 0, pi/2]);
L(2) = Link([0, 0, 12, 0]);
L(3) = Link([0, 0, 12, 0]);
L(4) = Link([0, 0, 0, pi/2]);
L(5) = Link([0, 0, 0, -pi/2]);
L(6) = Link([0, 0, 0, 0]);

ModifiedLynx = SerialLink(L);
ModifiedLynx.name = 'Modified Lynx';
ModifiedLynx.tool = trotx(pi/2)*transl(0, 0, 14);

%%% We represent the robot

ModifiedLynx.plot([0 0 0 0 0 0]);
```

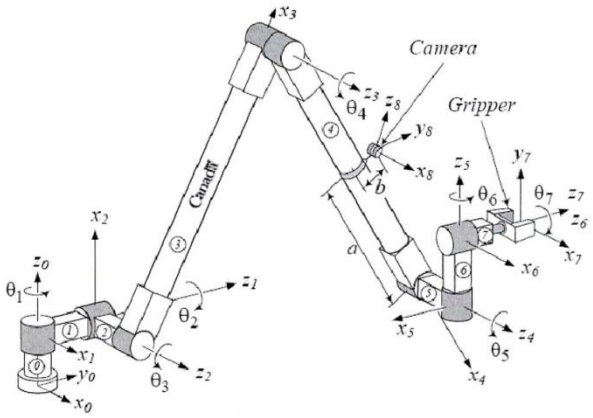



```
%%% Now, we can use the ikine6s function

T = ModifieldLynx.fkine([0 0 0 0 0 0]);

q1 = ModifiedLynx.ikine6s(T, 'run');
q2 = ModifiedLynx.ikine6s(T, 'rdn');

ModifiedLynx.plot(q1);
ModifiedLynx.plot(q2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

PROBLEM 18

The 7-dof Canadarm-2 robot, mounted on the International Space Station, is shown below on the left. The kinematic chain of this robot is symmetric with respect to the mid revolute axis. This symmetry allows the base and the end-effector of this robot to exchange roles, a feature that enables the robot to move along the space station. The kinematic chain of the Canadarm-2 and its DH parameters are also shown below.





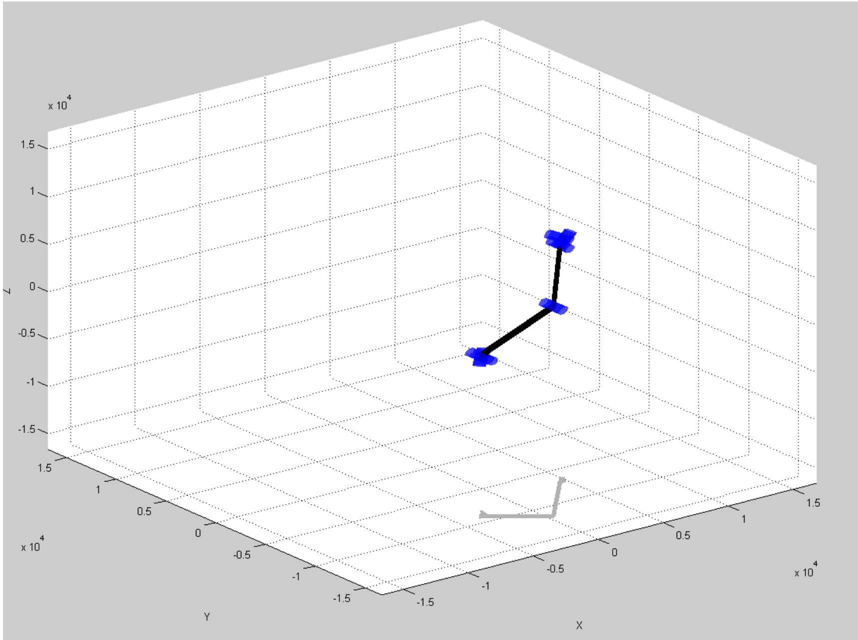
Joint/Link i	θ_i [rad]	d_i [mm]	a_i [mm]	α_i [rad]
1	θ_1	380	0	$-\pi/2$
2	θ_2	635	0	$\pi/2$
3	θ_3	504	6850	0
4	θ_4	0	6850	0
5	θ_5	504	0	$-\pi/2$
6	θ_6	635	0	$\pi/2$
7	θ_7	380	0	0

1) Create the kinematic model of the above robot and its symmetric one with respect to the mid revolute axis.

2) Generate a random walk of the robot in which the robot hand moves to an arbitrary location, then the hand is fixed and the base is released so that the base is moved to another arbitrary location, and the process is repeated.

SOLUTION

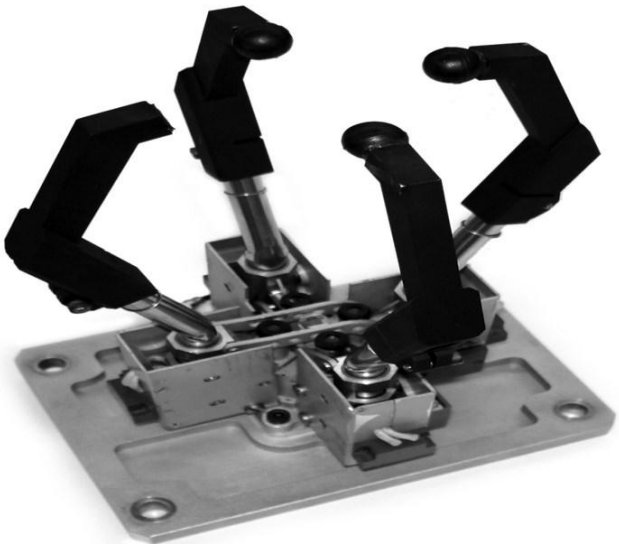
```
clear;
clc;

L(1)= Link([0 380    0 -pi/2]);
L(2)= Link([0 635    0  pi/2]);
L(3)= Link([0 504 6850     0]);
L(4)= Link([0   0 6850     0]);
L(5)= Link([0 504    0 -pi/2]);
L(6)= Link([0 635    0  pi/2]);
L(7)= Link([0 380    0     0]);

R(1)= Link([0 -380     0 -pi/2]);
R(2)= Link([0 -635     0  pi/2]);
R(3)= Link([0 -504 -6850     0]);
R(4)= Link([0    0 -6850     0]);
R(5)= Link([0 -504     0 -pi/2]);
R(6)= Link([0 -635     0  pi/2]);
R(7)= Link([0 -380     0     0]);

RobotD = SerialLink(L);
RobotD.name= 'D';

RobotI = SerialLink(R);
RobotI.name= 'I';

n =10;

samples= 50;

angR = [0 0 0 0 0 0 0];

for i=1:n
    angD =  random('unif',-pi,pi,1,7);
    RobotD.plot(jtraj(angR, angD, samples), 'nobase', 'noname', ...
        'nojaxes', 'nowrist', 'cylinder', [0 0 1]);
    angR = -1*fliplr(angD);
    RobotI.base = RobotD.fkine(angD);
    angD =  random('unif',-pi,pi,1,7);
    RobotI.plot(jtraj(angR, angD, samples), 'nobase', 'noname', ...
        'nojaxes', 'nowrist', 'cylinder', [0 0 1]);
    angR = -1*fliplr(angD);
    RobotD.base = RobotI.fkine(angD);
end
```

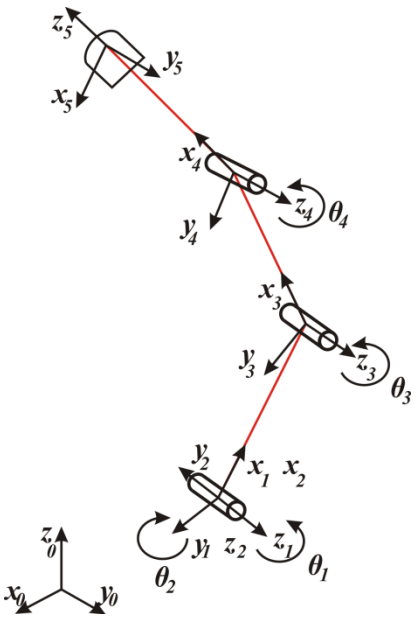

x 10^4
1.5
1
0.5
0
-0.5
-1
-1.5
Z
Y
X

%%
%%
%%

PROBLEM 19

Consider the following four-fingered robotic hand. Each finger has 4 actuated revolute joints as shown below on the right.





The DH parameters of each finger are:

Joint/Link i	θ_i [rad]	d_i [mm]	a_i [mm]	α_i [rad]	Ranges [rad]
1	θ_1	0	0	$-\pi/2$	[0, π/2]
2	θ_2	0	55	0	[-π/6, π/6]
3	θ_5	0	25	0	[0, π/2]
4	θ_4	0	25	0	[0, π/2]

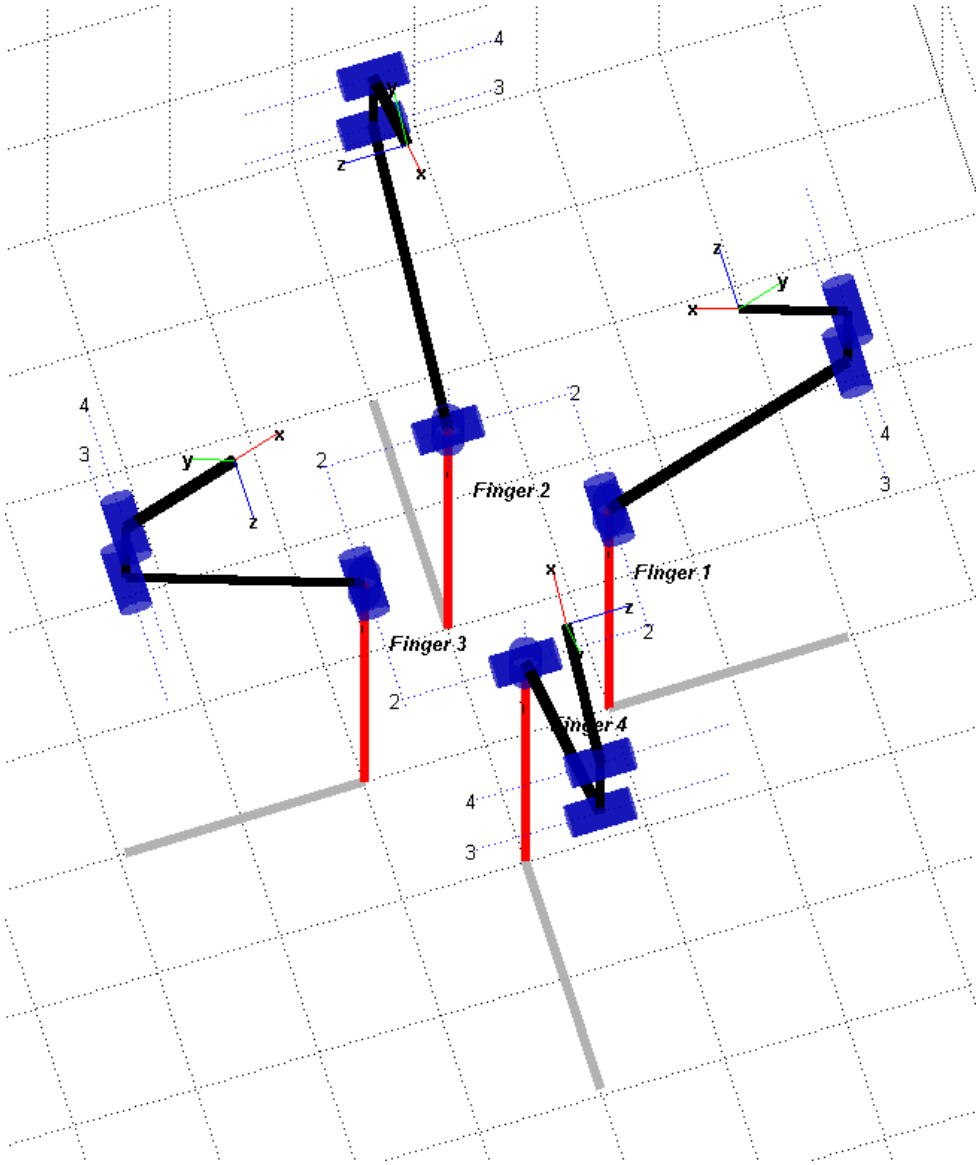
Assuming that the fingers are anchored at the vertices of a square of side length of $\sqrt{2}$ mm,

1) Build a kinematic model of the hand.

2) Simulate a coordinated motion in which the hand is holding a square of side length 10 mm, which rotates randomly, so that each finger distal phalanges are perpendicular and coplanar with one edge of the square.

SOLUTION

```
%%% QUESTION 1:
%%%
%%% First we clean the workspace

clear;
clc;

%%% We create the links according to the DH parameters

L(1)= Link([0 0  0 -pi/2]);
L(2)= Link([0 0 55  0]);
L(3)= Link([0 0 25  0]);
L(4)= Link([0 0 25  0]);

%%% We create the four fingers and locate them in the base frame

Finger1 = SerialLink(L);
Finger1.name = 'Finger 1';
Finger1.base= trotz(0)*transl(20, 0, 0);

Finger2 = SerialLink(L);
Finger2.name = 'Finger 2';
Finger2.base= trotz(pi/2)*transl(20, 0, 0);

Finger3 = SerialLink(L);
Finger3.name = 'Finger 3';
Finger3.base= trotz(pi)*transl(20, 0, 0);

Finger4 = SerialLink(L);
Finger4.name = 'Finger 4';
Finger4.base= trotz(3*pi/2)*transl(20, 0, 0);

%%% We plot the four fingers to check if they have been correctly
%%% defined

figure;
axis([-100 100 -100 100 -100 100]);
grid on;
hold on;

Finger1.plot([0 -pi/4 -pi/4 -pi/4]);
Finger2.plot([0 -pi/4 -pi/4 -pi/4]);
Finger3.plot([0 -pi/4 -pi/4 -pi/4]);
Finger4.plot([0 -pi/4 -pi/4 -pi/4]);
```

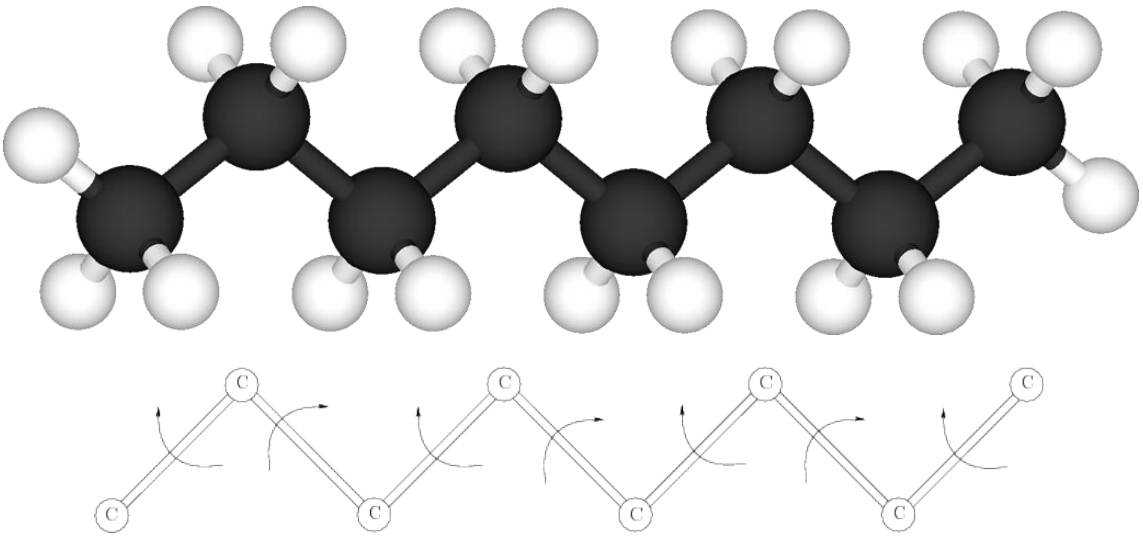


%%% QUESTION 2:
%%%

PROBLEM 20

The following figure shows a model of the octane molecule. There is a linear chain of eight carbon atoms, and a bond exists between each consecutive pair of carbons in the chain. There are also numerous hydrogen atoms, but we will ignore them. Each bond between a pair of carbons is capable of twisting. It is assumed that there are seven degrees of freedom, each of which arises from twisting a bond.

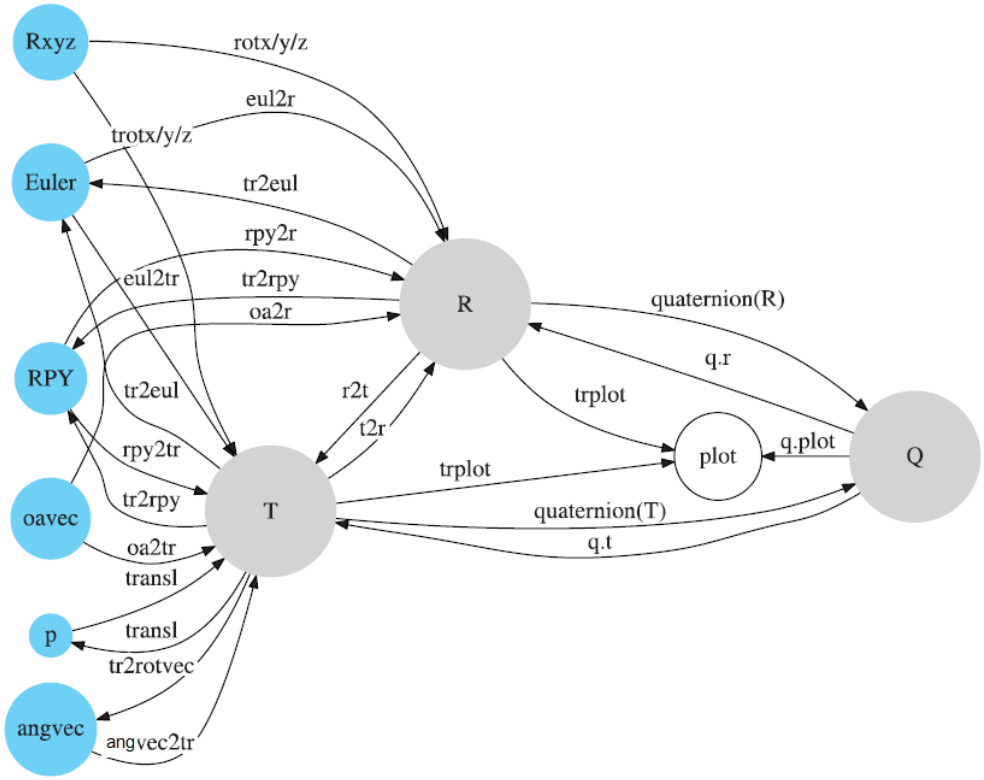
Express the distance between the carbons in the extremes of the molecule as a function of the seven degrees of freedom.



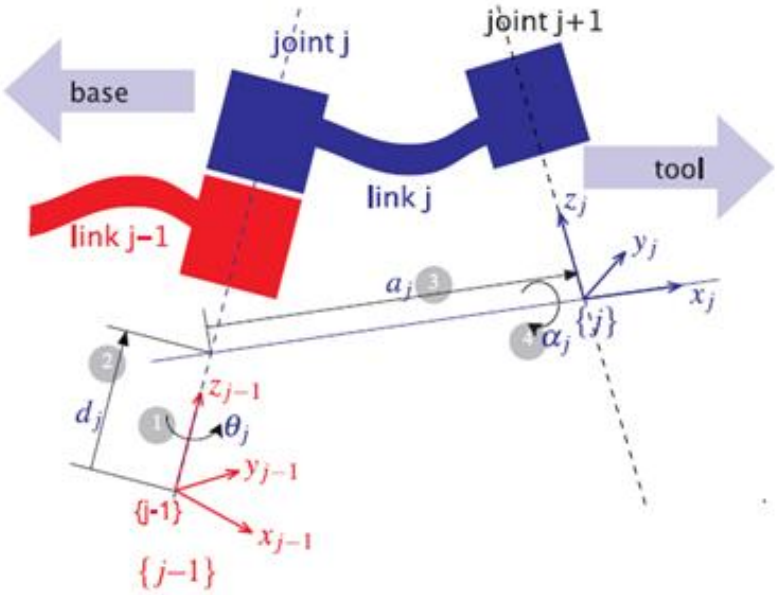
SOLUTION

Each bond connecting two carbons may be interpreted as a link of length d_i that is aligned with the z_i axis.

APPENDIX I: ROBOTICS TOOLBOX FUNCTIONS TO MANIPULATE TRANSFORMATIONS



APPENDIX II: DEFINITION OF THE STANDARD DH PARAMETERS




```
Tj-1j = trotz(theta_j)*transl(0,0,d_j)*transl(a_j,0,0)*trotx(alpha_j);
```