```
% MEC529 Matlab Homework 5 Problem 4, IK for SCARA Codes Created by Yongxin Guo
addpath('/Users/guoyongxin/Desktop/Assignment_Academics/Senior_Second semester/MEC529/Myfunctions');
% Codes summary:
% assign the given joint angles for computing the given gst transformation
% matrix via forward kinematics algorithms. The given gst matrix was fed into the IK algorithm codes for
% computing the joint angles and we can be able to compare one of the
% solutions we got with the given joint angles at the beginning, and the
% rest of the solutions will be passed into the forward kinematics algorithms to see if we can get back
% the original gst matrix. The above will be the verification process for
% the IK algorithm codes.

close all
clear
clc

% assign configuration variables
l0 = 0.2;
l1 = 0.6;
l2 = 0.3;
% create gst0 matrix.
R0 = eye(3);
P0 = [0;l1+l2;l0];
gst0 = [R0,P0;[0 0 0],1];
% create axis of rotation.
axis_joints = [0;0;1]*ones(1,4);
% create q_matrix.
q1 = [0;0;0];
q2 = [0;l1;0];
q3 = [0;l1+l2;0];
q4 = [0;0;0];
q_joints = [q1,q2,q3,q4];
% create matrix for the type of joints
type_joints = ["R";"R";"R";"P"];
% assign the joint angles for SCARA.
theta = [(pi/4) * ones(3,1); 0.1];
% compute gst(theta)
gst_theta = manipdkin(gst0, axis_joints, q_joints, type_joints, theta);
disp("The given gst(theta) transformation matrix is shown below: ");
disp(gst_theta);

%****************************IK algorithms code*************************
% gst_theta is given transformation matrix, which will be used for future
% verification!
pt_p = q3; % choose a point p along axis of xi_3, which is also point p' as illustrated in my HW solns.
pt_q = q1; % choose a point q along axis of xi_1, which is also point p'' as illustrated in my HW solns.
pt_r1 = q2; % choose a point r1 along axis of xi_2 for computing theta2 in SP3, as illustrated in my HW solns.
pt_r2 = q1; % choose a point r2 along axis of xi_1 for computing theta1 in SP1, as illustrated in my HW solns.
pt_r3 = q3; % choose a point r3 along axis of xi_3 for computing theta3 in SP1, as illustrated in my HW solns.
% get theta4 first
theta4 = gst_theta(3,4) - l0;
%---------------------get theta2------------------------
R0t = transpose(R0);
gst0_inv = [R0t, -1*R0t*P0; [0 0 0], 1];
gst_theta4_inv = [eye(3),-1*axis_joints(:,4)*theta(4);[0 0 0], 1]; % transformation matrix for only negative xi4 axis
delta = gst_theta*gst0_inv*gst_theta4_inv*[pt_p;1] - [pt_q;1]; % compute the vector delta. Note that we need homogenous representation for points.
delta_mag = sqrt(transpose(delta)*delta); % compute the magnitude of delta
% use SP3 to get theta2. There will be 2 possible solns for theta2.
theta2 = PadenKahanSP3(axis_joints(:,2), pt_p, pt_q, pt_r1, delta_mag); % Note that we don't need homogenous representation of point in this funct
pt_q_prime = gst_theta*gst0_inv*gst_theta4_inv*[pt_p;1]; % compute point q prime shown in HW soln paper. Note this is homo. rep. of a point.
for m = 1:length(theta2)
    %---------------------get theta1-------------------------
    R2 = AxisAngle_to_Rot(axis_joints(:,2),theta2(m)); % Rotation matrix for axis of xi2 with angle theta2.
    P2 = (eye(3)-R2)*q2; % Position vector for transformation around xi2.
    pt_p_temp = [R2,P2;[0 0 0],1] * [pt_p;1]; % compute temporary point p after rotation around xi2 by theta2.
    % use SP1 to get theta1.
    theta1_m = PadenKahanSP1(axis_joints(:,1),pt_p_temp(1:3),pt_q_prime(1:3),pt_r2); % Note the points here are not homo. rep.
    theta1(m,1) = theta1_m; % assign theta1 for corresponding theta2.
    %---------------------get theta3-------------------------
    % compute exp(-xi2*theta2).
    R2t = transpose(R2);
    gst_theta2_inv = [R2t,-1*R2t*P2;[0 0 0], 1]; % gst(theta2) matrix inverse, which is just exp(-xi2*theta2).
    % compute exp(-xi1*theta1).
    R1 = AxisAngle_to_Rot(axis_joints(:,1),theta1_m);
    P1 = (eye(3)-R1)*q1; % Position vector for transformation around xi1.
    R1t = transpose(R1);
    gst_theta1_inv = [R1t,-1*R1t*P1;[0 0 0], 1]; % gst(theta1) matrix inverse, which is just exp(-xi1*theta1);
    pt_q_doublePrime = gst_theta2_inv*gst_theta1_inv*gst_theta*gst0_inv*gst_theta4_inv*[pt_q;1]; % compute point q double prime
    % use SP1 to get theta3.
    theta3_m = PadenKahanSP1(axis_joints(:,3),pt_q,pt_q_doublePrime(1:3),pt_r3); % Note that no homo. rep. of point here
    theta3(m,1) = theta3_m; % assign theta3 for corresponding theta2 and theta1.
end


%---------------------concatenating all the possible solns-----------------
theta_IK = transpose([theta1,theta2,theta3,theta4*ones(2,1)]);
[rows,solnsNum] = size(theta_IK);

validSoln = 0; % initialize a counter for counting the valid solutions.
for i = 1:solnsNum
    disp("No." + num2str(i) + " solution is: ");
    disp(theta_IK(:,i));
    % verification starts.
    gst_theta_IK = manipdkin(gst0, axis_joints, q_joints, type_joints, theta_IK(:,i)); % compute gst to see if we get the identical gst as given.
    disp("Its corresponding transformation matrix is: ");
    disp(gst_theta_IK);
    diff = abs(gst_theta_IK-gst_theta);
```

```
    disp("The corresponding difference with the given matrix is: ");
    disp(diff);
    if norm(diff) < 1.0e-10   % set a criterion for checking the consistence with the given matrix.
        disp("No." + num2str(i) + " solution is valid!");
        validSoln = validSoln + 1; % update the validSoln counter by 1 if the solution is valid.
    else
        disp("No." + num2str(i) + " solution is invalid!");
    end
    disp("---------------------------------------------------------");
    % verification ends.
end

disp("Conclusion: There are " + num2str(validSoln) + " possible solutions in total");
```

```
The given gst(theta) transformation matrix is shown below:
   -0.7071   -0.7071         0   -0.7243
    0.7071   -0.7071         0    0.4243
         0         0    1.0000    0.3000
         0         0         0    1.0000

No.1 solution is:
    1.2964
    5.4978
    1.8452
    0.1000

Its corresponding transformation matrix is:
   -0.7071   -0.7071         0   -0.7243
    0.7071   -0.7071         0    0.4243
         0         0    1.0000    0.3000
         0         0         0    1.0000

The corresponding difference with the given matrix is:
   1.0e-15 *

    0.4441    0.2220         0         0
    0.2220    0.4441         0    0.2220
         0         0         0         0
         0         0         0         0

No.1 solution is valid!
---------------------------------------------------------
No.2 solution is:
    0.7854
    0.7854
    0.7854
    0.1000

Its corresponding transformation matrix is:
   -0.7071   -0.7071         0   -0.7243
    0.7071   -0.7071         0    0.4243
         0         0    1.0000    0.3000
         0         0         0    1.0000

The corresponding difference with the given matrix is:
   1.0e-15 *

    0.1110    0.1110         0    0.1110
    0.1110    0.1110         0    0.0555
         0         0         0         0
         0         0         0         0

No.2 solution is valid!
---------------------------------------------------------
Conclusion: There are 2 possible solutions in total
```

Published with MATLAB® R2018b