```
% Elbow Manipulator Inverse Alogrithms. Take given configuration as the input, and output the joint angles solution and corresponding message, Cre
% Codes summary:
% Given configuration of the peg and final configuration of the hole,
% which are all in the form of transformation matrix, and feed them into the IK algorithm codes for
% computing the joint angles. For verification, we can passed the joint angles we obtained into
% the forward kinematics algorithms to see if we can get back
% the original given configuration.
function joint_angles = getElbowManipIK(configuration,gst0,axis_joints,q_joints,type_joints,info)

% given transformation matrix.
T_given = configuration;
% assign axis of motion.
axis1 = axis_joints(1:3,1);
axis2 = axis_joints(1:3,2);
axis3 = axis_joints(1:3,3);
axis4 = axis_joints(1:3,4);
axis5 = axis_joints(1:3,5);
axis6 = axis_joints(1:3,6);
% assign q vectors.
q1 = q_joints(1:3,1);
q2 = q_joints(1:3,2);
q3 = q_joints(1:3,3);
q4 = q_joints(1:3,4);
q5 = q_joints(1:3,5);
q6 = q_joints(1:3,6);

% display the given transformation matrix.
disp("The given "+ info +" transformation matrix is shown below: ");
disp(T_given);

%****************************IK algorithms code*************************
% gst_theta and gst0 are given transformation matrices, gst_theta will be used for future
% verification!
pt_p = q4; % select a point at the intersecting of axis 4,5 and 6.
pt_q = q1; % select a point along axis 1 and 2.
pt_p_prime = pt_q; % same as point q.
pt_p_doublePrime = [0;0;0]; % select a point that is not along axis 6, which can be the origin.
pt_r1 = q3; % for theta3.
pt_r2 = q1; % for theta1 and 2.
pt_r3 = q4; % for theta4 and 5.
pt_r4 = q4; % for theta6.
%-------------------------get theta3-----------------------------
% calculate gst*g(0)^-1, which is g.
R0 = gst0(1:3,1:3);
P0 = gst0(1:3,4);
R0t = transpose(R0);
gst0_inv = [R0t, -1*R0t*P0; [0 0 0], 1];
g = T_given*gst0_inv;
delta = g*[pt_p;1]-[pt_q;1];
delta_mag = sqrt(transpose(delta)*delta);
% use SP3.
theta3 = PadenKahanSP3(axis3, pt_p, pt_q, pt_r1, delta_mag);
% There will be 2 possible solns for theta3.
sz = length(theta3); % get the number of possible theta3 solns.
% allocating the size for theta first.
theta1 = zeros(sz,1);
theta2 = theta1;
theta4 = theta1;
theta5 = theta1;
theta6 = theta1;
% declare 3 by 3 indentity matrix for future use.
I = eye(3);
counter = 0; % initialize a counter for counting the total soln number
dof = 6; % the d.o.f for the elbow manipulator.
for i = 1:sz
    R3 = AxisAngle_to_Rot(axis3,theta3(i));
    P3 = (eye(3)-R3)*q3;
    %--------------------get theta1 and theta2 (SP2)-------------------
    temp_pt = [R3,P3;[0 0 0],1]*[pt_p;1];
    gp = g*[pt_p;1];
    % use SP2
    theta1and2 = PadenKahanSP2(axis1,axis2,temp_pt(1:3),gp(1:3),pt_r2);
    sz1 = length(theta1and2); % get the length of theta1and2. number of solns.
    for j = 1:sz1
        theta1(j) = theta1and2(1,j);
        theta2(j) = theta1and2(2,j);

        %--------------------get theta4 and theta5 (SP2)-------------------
        % compute gst_theta1_inv.
        R1 = AxisAngle_to_Rot(axis1,theta1(j));
        R1t = transpose(R1);
        P1 = (I-R1)*q1;
        gst_theta1_inv = [R1t,-1*R1t*P1; [0 0 0], 1];
        % compute gst_theta2_inv.
        R2 = AxisAngle_to_Rot(axis2,theta2(j));
        R2t = transpose(R2);
        P2 = (I-R2)*q2;
        gst_theta2_inv = [R2t,-1*R2t*P2; [0 0 0], 1];
        % compute gst_theta3_inv.
        R3t = transpose(R3);
        P3 = (I-R3)*q3;
        gst_theta3_inv = [R3t,-1*R3t*P3; [0 0 0], 1];
        % multiplying together
        gst_theta321_inv = gst_theta3_inv*gst_theta2_inv*gst_theta1_inv;
        % compute point q_prime.
        pt_q_prime = gst_theta321_inv*g*[pt_p_prime;1];
```

```
        % use SP2
        theta4and5 = PadenKahanSP2(axis4,axis5,pt_p_prime,pt_q_prime(1:3),pt_r3);
        sz2 = length(theta4and5); % get the length of theta4and5.
        for k = 1:sz2
            counter = counter + 1; % update the counter by 1.
            theta4(k) = theta4and5(1,k);
            theta5(k) = theta4and5(2,k);
            %--------------------get theta6-------------------
            % compute gst_theta5_inv.
            R5 = AxisAngle_to_Rot(axis5,theta5(k));
            R5t = transpose(R5);
            P5 = (I-R5)*q5;
            gst_theta5_inv = [R5t,-1*R5t*P5; [0 0 0], 1];
            % compute gst_theta4_inv.
            R4 = AxisAngle_to_Rot(axis4,theta4(k));
            R4t = transpose(R4);
            P4 = (I-R4)*q4;
            gst_theta4_inv = [R4t,-1*R4t*P4; [0 0 0], 1];
            % multiplying together
            gst_theta54_inv = gst_theta5_inv*gst_theta4_inv;
            % get gst_theta54321_inv
            gst_theta54321_inv = gst_theta54_inv*gst_theta321_inv;
            % compute point q_doublePrime
            pt_q_doublePrime = gst_theta54321_inv*g*[pt_p_doublePrime;1];
            % use SP1
            theta6 = PadenKahanSP1(axis6,pt_p_doublePrime,pt_q_doublePrime(1:3),pt_r4);
            %-------------------Assign all the theta solns-------
            theta_temp = [theta1(j);theta2(j);theta3(i);theta4(k);theta5(k);theta6];
            for l = 1:dof
                theta_IK(l,counter) = theta_temp(l);
            end
        end
    end
end


[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.
    T_given_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(T_given_IK);
    diff = abs(T_given_IK-T_given);
    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");
joint_angles = theta_IK;


end
```

```
Not enough input arguments.

Error in getElbowManipIK (line 11)
T_given = configuration;
```

---

*Published with MATLAB® R2018b*