SSY156 - Modelling and Control of Mechatronic Systems
Peer-2-Peer Homework 02
Group06 Solution

Question 1 (1 point)

At a first glance, this problem may seem easy. However, it is not possible to solve this inverse differential kinematics problem using the analytical Jacobian for the given end-effector position. Since the matrix $T_A(\phi)$ relating the analytical to the geometrical Jacobian ($J_A = T_A^{-1}(\phi) * J$) is singular for the desired end-effector Euler-angle orientation, it will not be possible to calculate the analytical Jacobian and the simulations will lead to unstable and wrong results.

The tricky part to solve this question is to formulate the inverse differential kinematics problem using Quaternions, as explained in the slide number 19 of the lecture notes or page 140 of the main course book. In this way, since the geometric Jacobian will be used instead of the analytical Jacobian, the occurrence of representation singularities will be avoided.

At this point, a Jacobian inverse solution can be computed as:

$$\dot{q} = J^{-1}(q) \begin{bmatrix} \dot{p}_d + K_P\, e_P \\ \omega_d + K_O\, e_O \end{bmatrix} \tag{1}$$

where the orientation error e_O is expressed in terms of quaternions:

$$\begin{aligned} e_O &= \Delta\epsilon \\ \Delta Q &= \{\Delta\eta, \Delta\epsilon\} = Q_d * Q_e^{-1} \end{aligned} \tag{2}$$

This problem can be solved by using the following Matlab code:

```
% Desired end-effector position
x_d_P = [-0.1;
     0.2;
     0.7];
% Desired end-effector Euler-angle orientation
x_d_O = [0;
         pi;
         0];
% Desired end-effector velocity
v_d_P = [0;
         0;
         0];
% Desired end-effector Euler-angle velocity
v_d_O = [0;
         0;
         0];
```

```
% Calculate forward kinematics
T_0_e = KUKA_FKin(q);
x_e_P = T_0_e(1:3,4);
x_e_O = rotm2eul(T_0_e(1:3,1:3),'ZYZ')';

% Calculate orientations in Quaternions
x_e_O_quat = rotm2quat(T_0_e(1:3,1:3));
x_d_O_quat = eul2quat(x_d_O','ZYZ');

% Calculate desired angular velocity
Tphi =@(eulang) [0 -sin(eulang(1))  cos(eulang(1))*sin(eulang(2)) ;
                 0  cos(eulang(1))  sin(eulang(1))*sin(eulang(2)) ;
                 1  0               cos(eulang(2))           ];
w_d = Tphi(x_e_O)*v_d_O;

% Define gain matrices
K_P = eye(3)*10;
K_O = eye(3)*10;

% Calculate position error
e_P = x_d_P - x_e_P;

% Calculate orientation error based on Quaternions
dQ = quatmultiply(x_d_O_quat, quatinv(x_e_O_quat));
e_O = dQ(2:4)';

% Calculate geometric Jacobian
J_geom = Jacobian_KUKA7(q);

% Calculate joint velocities
v = [ v_d_P + K_P * e_P;
        w_d   + K_O * e_O];
dq = J_geom\ v;
```

After running the model, it can be seen that the position and orientation error go very fast to zero, meaning that the end-effector has reached the desired position.

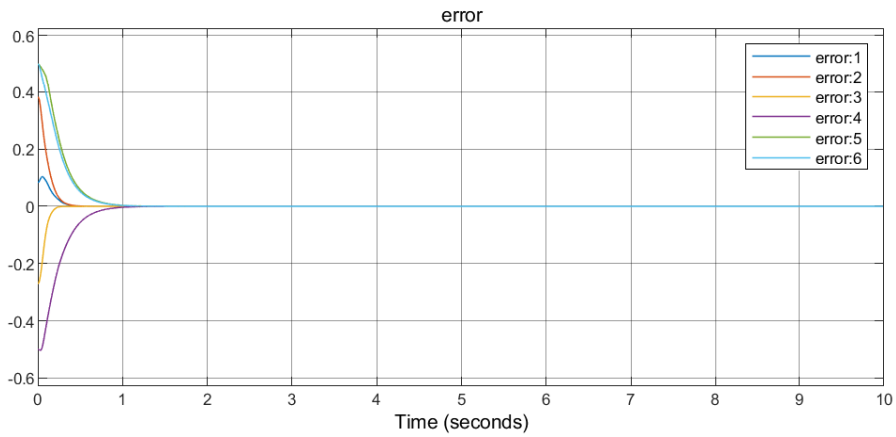

Figure 1: *Position and orientation error. (x,y,z) are the first 3 lines and the the Euler angles are the last 3*

At the end of the simulation the following joint angles are obtained as answer to the question:

```
q = [0.9799 -0.5521 0.5183 -1.7641 0.2743 1.8582 1.5116]
```

Question 2 (1 point)

In order to solve this problem we need to use the kineto-statics duality, which will give the joint force/torques that can balance force/torques at the endeffector:

$$\tau = J^T(q) \cdot \gamma_e \tag{3}$$

First, we need to calculate the geometric jacobian based on the joint angles that will result in the desired end-effector position. This can be done using the answer from question 1, which resulted in `q=[0.9799,-0.5521,0.5183,-1.7641,0.2743,1.8582,1.5116]`.

Next, the acting force on the end-effector γ_e is obtained by simply calculating the reaction force that the block is causing in the end-effector.The tricky part is to know the orientation of this force, since it must be defined in base-frame coordinates. This will correspond to $\gamma_e = [0, 0, -98]N$. This will result in the following calculation:

$$\tau = \begin{bmatrix} -0.2000 & 0.1894 & -0.3184 & -0.0971 & -0.1254 & -0.0078 & 0 \\ -0.1000 & 0.2824 & 0.0142 & 0.0227 & 0.0074 & -0.1308 & 0 \\ 0 & -0.1104 & -0.1020 & 0.3692 & -0.0000 & -0.0000 & 0 \\ 0 & -0.8304 & -0.2922 & 0.9564 & 0.0567 & -0.9982 & 0.0001 \\ 0 & 0.5571 & -0.4355 & -0.1337 & 0.9573 & 0.0592 & 0.0001 \\ 1.0000 & 0 & 0.8514 & 0.2598 & 0.2835 & -0.0001 & -1.0000 \end{bmatrix}^T \cdot \begin{bmatrix} 0 \\ 0 \\ -98 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

$$\tau = \begin{bmatrix} 0 \\ 10.8159 \\ 9.9954 \\ -36.1862 \\ 0.0009 \\ 0.0011 \\ 0 \end{bmatrix} \tag{5}$$

From this result we can infer that the joint 1 and the joint 7 are parallel to the force, so there's no resultant torque. This can be confirmed in the figure 2 where the blue arrow in the 3-axis frame of reference points out the z direction.

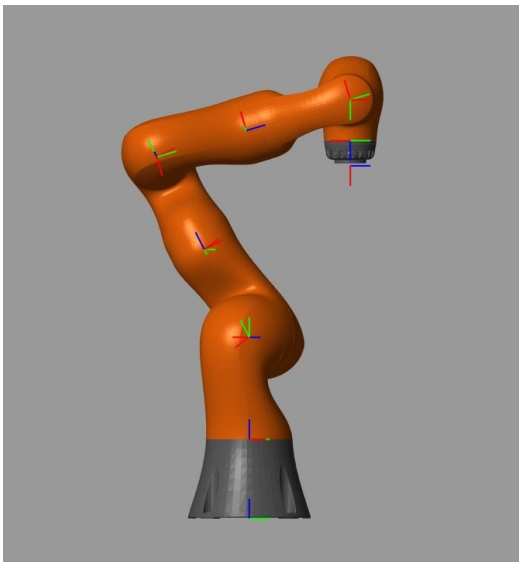

Figure 2: *Reference frame of the joints, direction z is the blue arrow*