SSY156 - Modelling and Control of Mechatronic systems
Assignment A01

Lucas Rath

Question 1

Making use of the Denavit-Hartenberg Convention, we can define the following parameters for the 3 links of the OmniBundle manipulator, as seen in Table 1.

Table 1: *DH parameters for the OmniBundle manipulator*

Link	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	0	q_1
2	L_1	0	0	q_2
3	L_2	0	0	$q_3 - \frac{\pi}{2}$

To obtain the DH table above, the DH convention had to be followed. Since the axes are already defined, the remaining task consist of defining the translation and rotation displacements between the axes related to each link. The first link is a virtual link, due to its zero length and the only non-zero values are the parameters α_1 (-pi/2 angle from z0 to z1 about axis x1) and θ_1 (angle q1 from x0 to x1 about axis z0). Link 2 and 3 are similar from the kinematics point of view and the nonzero values correspond to a_i (translation O_{i-1}-O_i along axis -x_i) and the angle displacement θ_i. Please check Appendix for a more detailed explanation.

Using the DH table we are then able to derive the homogeneous transformation matrices between each link: $A_1^0(q_1)$, $A_2^1(q_2)$ and $A_3^2(q_3)$. The resultant transformation of the end-effector in respect to the base can be calculated as $A_3^0(q_1, q_2, q_3) = A_1^0(q_1) * A_2^1(q_2) * A_3^2(q_3)$, which represents the forward kinematics of the OmniBundle. Given the angles $q = [q_1, q_2, q_3]$, we can then calculate A_3^0 numerically. As a result, the position of the end-effector $F3^0$ can be calculated by transforming a null vector in the end-effector coordinates $\{3\}$ to the base coordinate system $\{0\}$:

$$F3^0 = A_3^0(0.67, -0.15, 2.7) \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.16 \\ 0.1268 \\ -0.08984 \\ 1 \end{bmatrix} \tag{1}$$

$$F3^0 = A_3^0(-0.73, 0.25, 1.5) \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.1921 \\ -0.1719 \\ -0.05619 \\ 1 \end{bmatrix} \tag{2}$$

Question 2

Given a desired end-effect position in respect to the base frame $F3^0$, we can find the value of the joint angles by solving the inverse kinematics problem. One simple and effective approach is to make

use of the iterative Newton method, such that the same equation as in (1), but now give the desired end-effector position $F3^0$, calculate the joint angles q_1, q_2, and q_3 that will satisfy the same equation:

$$\tilde{F3}^0 = A_3^0(q_1, q_2, q_3) \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

Using the numerical solver in Matlab, the following results were obtained:

```
F3 = [0.06 0.04 0.02 1]';           % in {0} coordinates
% vpasolve( T_0_3_fun(q)*[0 0 0 1]' == F3 )
    q1:  0.588
    q2:  1.013
    q3: -0.996

F3 = [0.0847 0.0123 -0.005 1]';     % in {0} coordinates
% vpasolve( T_0_3_fun(q)*[0 0 0 1]' == F3 )
    q1:  0.144
    q2:  1.298
    q3: -0.909
```

Question 3

In order to find the analytical jacobian J_A, one can first derive the geometrical jacobian J, which is quite straight forward to calculate, and then apply a known transformation $J = T_A(\phi)\, J_A$. Since the the OmniBundle is composed of 3 revolute joints, the geometric Jacobian is given as follows:

$$J = \begin{bmatrix} z_0 \times p_e - p_0 & z_1 \times p_e - p_1 & z_2 \times p_e - p_2 \\ z_0 & z_1 & z_2 \end{bmatrix} \tag{4}$$

where:

$$\begin{aligned} p_0 &= [0\ 0\ 0\ 1]^T \\ p_1 &= A_1^0 \cdot p_0; \\ p_2 &= A_1^0 \cdot A_2^1 \cdot p_0; \\ p_e &= A_1^0 \cdot A_2^1 \cdot A_3^2 \cdot p_0 \end{aligned} \qquad \textit{and} \qquad \begin{aligned} z0 &= [0\ 0\ 1]^T \\ z1 &= R_1^0 \cdot z0 \\ z2 &= R_1^0 \cdot R_2^1 \cdot z0 \end{aligned}$$

Finally:

$$J = \begin{bmatrix} -0.132\sin(q_1)\,(\sin(q_2+q_3)+\cos(q_2)) & 0.132\cos(q_1)\,(\cos(q_2+q_3)-1.0\sin(q_2)) & 0.132\cos(q_2+q_3)\cos(q_1) \\ 0.132\cos(q_1)\,(\sin(q_2+q_3)+\cos(q_2)) & 0.132\sin(q_1)\,(\cos(q_2+q_3)-1.0\sin(q_2)) & 0.132\cos(q_2+q_3)\sin(q_1) \\ 0 & -0.132\sin(q_2+q_3)-0.132\cos(q_2) & -0.132\sin(q_2+q_3) \\ 0 & -1.0\sin(q_1) & -1.0\sin(q_1) \\ 0 & \cos(q_1) & \cos(q_1) \\ 1.0 & 0 & 0 \end{bmatrix} \tag{5}$$

Once calculated J properly, we use the transformation matrix $T_A(\phi)$ that will transform the end-effector angular velocities ω_e to Euler-ZYZ angle derivatives $\dot{\phi}_e$:

$$J = \underbrace{\begin{bmatrix} I_3 & & \mathbf{0}_{3,3} & \\ & 0 & -s_\phi & c_\phi s_\theta \\ \mathbf{0}_{3,3} & 0 & c_\phi & s_\phi s_\theta \\ & 1 & 0 & c_\theta \end{bmatrix}}_{T(\phi_e)} \cdot J_A \quad \Rightarrow \quad J_A = T\,(\phi_e)^{-1} \cdot J = \begin{bmatrix} J_{A,p}(q) \\ \hline J_{A,\phi}(q) \end{bmatrix} \tag{6}$$

Important to notice is that $J_A(q, \phi)$ is not only a function of the joint angles q, but also of the Euler angles $\phi = [\varphi, \vartheta, \psi]$, which can be obtained from some well-known algebraic relations using the homogeneous transformation matrix A_3^0.

Question 4

a)

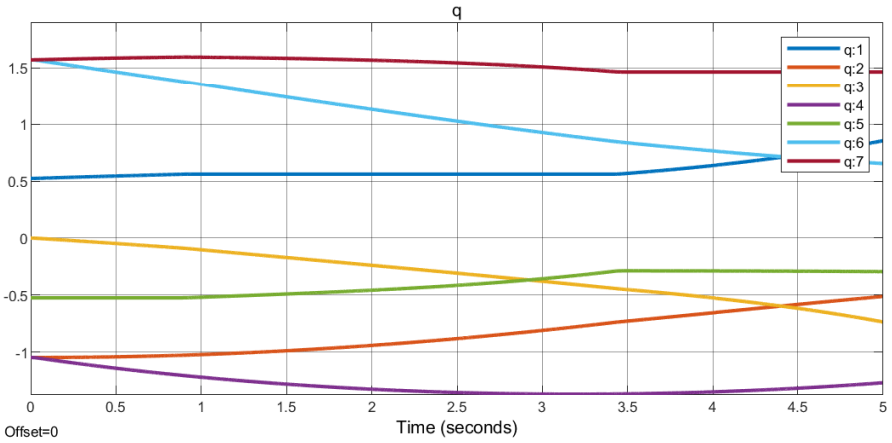
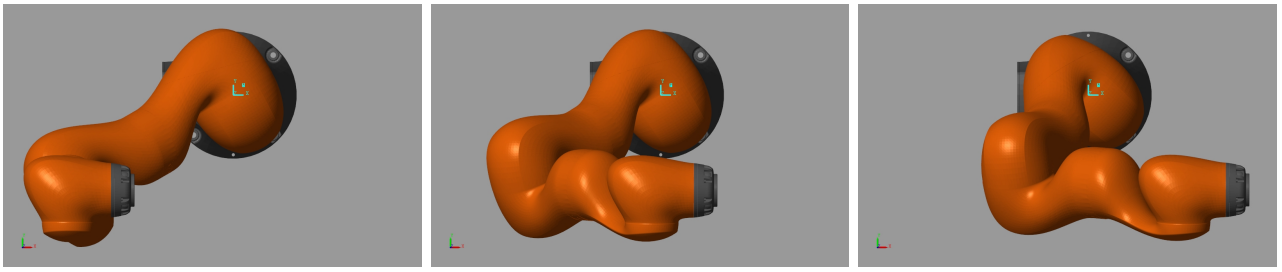


Figure 1: *Joint angles of the KUKA robot for constant x velocity in base frame coordinates, keeping end-effector orientation constant*



(a) *t=0s* ***(b)*** *t=3s* ***(c)*** *t=5s*

Figure 2: *Top view of the KUKA robot for 3 time instants moving with a constant velocity in the base frame x direction*

b) In order to achieve the desired position and velocity profile, a close-loop inverse kinematics formulation can be used to avoid integration drift. As stated in the lecture notes, one can use the following formula to calculate the joint velocities for each time step:

$$\dot{q} = J_A^{-1}(q)\left(\dot{x}_d + K \cdot (x_d - x_e)\right) \tag{7}$$

where x_d and $\dot{x}_d$ are the desired position and velocity respectively, K is a positive definite matrix and x_e is the end-effector position obtained by the direct kinematics using the joint angles q and $A_e^0(q)$.

$$x_d = \begin{bmatrix} -0.2 \\ -0.0008\,t^3 + 0.012\,t^2 - 0.2 \\ 1 \end{bmatrix} \qquad \dot{x}_d = \frac{dx_d}{dt} = \begin{bmatrix} 0 \\ -0.0024\,t^2 + 0.024\,t \\ 0 \end{bmatrix} \tag{8}$$

However, since the operational space consist only of the three translational coordinates (x,y,z), we only need to use the first three rows of the analytical jacobian in equation 9. Since the operational space consists of 3 DOF and there are 7 joints in KUKA, then the problem is said to be redundant.

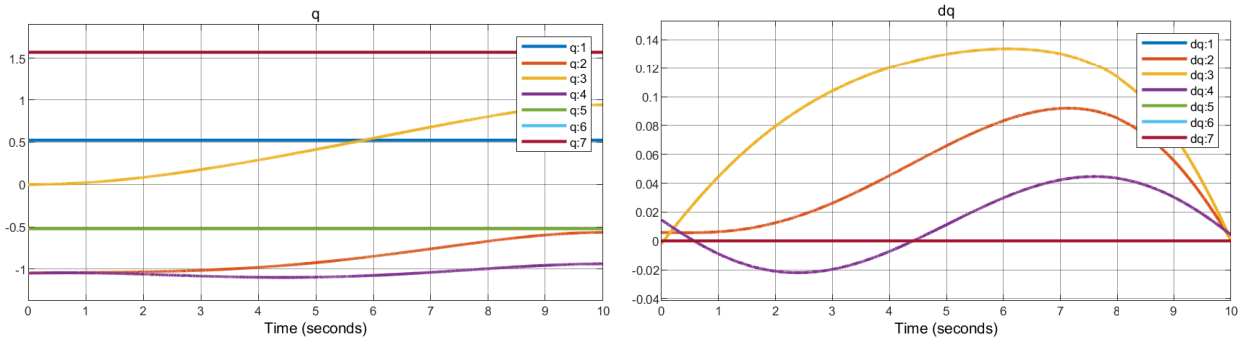
As a result, following the method of Lagrangian multipliers described in page 124 of the text book, the following formula should be applied:

$$\dot{q} = J_P^{\dagger}(q)\,(\dot{x}_d + K \cdot (x_d - x_e)) \tag{9}$$

where $J_P^{\dagger}(q)$ is the pseudo inverse of the linear part (first three rows) of the analytical Jacobian. Next, joint positions can be obtained by integrating velocities over time:

$$q(t) = \int_{t_0}^{t} \dot{q}(t)\; dt + q(t_0) \tag{10}$$

After simulation, the following results are obtained:



(a) *Joint angles* ***(b)*** *Joint velocities*

Figure 3: *Joint angles and velocities for the desired end-effector path*

c) Since there is no need to keep the orientation constant, the operational space has dimension equal 3, corresponding to the (x,y,z) desired end-effector positions. As explained in the last question, our manipulator is redundant and there are an infinity of configurations that can reach the desired position x_d=[-0.2,0.2,1]^T.

One easy way to obtain two solutions to the desired position is to run the close-loop inverse kinematics model, but starting the KUKA robot with different joint angles initial conditions. Moreover, if we set the desired velocity $\dot{x}_e = 0$, and the desired position x_d=[-0.2,0.2,1]^T, the end-effector will converge to the desired position $x_e \to x_d$ in the course of time.

If we choose q0 = [pi/6; -pi/3; 0; -pi/3; -pi/6; pi/2; pi/2]; we get:

$$q = [0.5236, -0.5571, 0.9493, -0.8944, -0.5236, 1.5708, 1.5708] \tag{11}$$

And if we choose q0 = [pi; -pi/3; 0; -pi/3; -pi/6; pi/2; pi/2]; then:

$$q = [2.9671, -0.1980, -0.2614, -0.8944, -0.5236, 1.5708, 1.5708] \tag{12}$$

1 Apendix

1.1 OmniBundle DH parameters:

Link 1: (virtual link)

- a1: no translation (O_0-O_1) along -x1
- alpha1: -pi/2 angle from z0 to z1 about x1
- d1: no translation (O_0-O_1) along z0
- theta1: angle q1 from x0 to x1 about z0

Link 2:

- a2: L1 translation (O_1-O_2) along -x2
- alpha2: 0 angle from z1 to z2 about x2
- d2: no translation (O_1-O_2) along z1
- theta2: angle q2 from x1 to x2 about z1

Link 3:

- a3: L2 translation (O_2-O_3) along -x3
- alpha3: 0 angle from z2 to z3 about x3
- d3: no translation (O_2-O_3) along z2
- theta3: angle q3-90 from x2 to x3 about z2

1.2 Matlab code for questions 1, 2 and 3

```
clear all; close all; clc;
%% OmniBundle parameters

L1 = 0.132;
L2 = 0.132;
syms q1 q2 q3
q = [q1 q2 q3];

T_0_1 = rb.T_DH(0,-sym(pi)/2,0,q1);
T_1_2 = rb.T_DH(L1,0,0,q2);
T_2_3 = rb.T_DH(L2,0,0,q3-sym(pi)/2);
T_0_3 = T_0_1 * T_1_2 * T_2_3;


%% Question 1

T_0_3_fun = matlabFunction(T_0_3,'Vars',{[q1 q2 q3]});

q_val = [0 0 0];
T_0_3_fun(q_val)*[0 0 0 1]'

q_val = [0.67 -0.15 2.7];
T_0_3_fun(q_val)*[0 0 0 1]'

q_val = [-0.73 0.25 1.5];
T_0_3_fun(q_val)*[0 0 0 1]'
```

```
%% Question 2

% Match the position of the end-effector (T_0_3_fun(q)*[0 0 0 1]') in the
% base coordinates to a specific position p3

p3 = [0.06 0.04 0.02 1]';
p3_resp = vpasolve( T_0_3_fun(q)*[0 0 0 1]' == p3 );

fprintf("q1: %.3f\nq2: %.3f\nq3: %.3f\n\n", p3_resp.q1, p3_resp.q2, p3_resp.q3 )
vpa(T_0_3_fun([p3_resp.q1, p3_resp.q2, p3_resp.q3]))*[0 0 0 1]'

p3 = [0.0847 0.0123 -0.005 1]';
p3_resp = vpasolve( T_0_3_fun(q)*[0 0 0 1]' == p3 );

fprintf("q1: %.3f\nq2: %.3f\nq3: %.3f\n\n", p3_resp.q1, p3_resp.q2, p3_resp.q3 )
vpa(T_0_3_fun([p3_resp.q1, p3_resp.q2, p3_resp.q3]))*[0 0 0 1]'


%% Question 3

% 3 revolute joints
z0 = [0 0 1]';
z1 = T_0_1(1:3,1:3)*z0;
z2 = T_0_1(1:3,1:3)*T_1_2(1:3,1:3)*z0;

p0 = [0 0 0 1]';            % selection of 4th column
p1 = T_0_1 * p0;
p2 = T_0_1 * T_1_2 * p0;
pe = T_0_1 * T_1_2 * T_2_3 * p0;

sel = [eye(3) zeros(3,1)];  % remove last row from vector

J_geom = [cross(z0,sel*(pe-p0)), cross(z1,sel*(pe-p1)), cross(z2,sel*(pe-p2));
          z0, z1, z2];
J_geom = simplify(J_geom)

syms phi theta psi
eulZYZ = [phi theta psi].';     % rotm2eul(T_0_3(1:3,1:3),'ZYZ')

J_analit = rb.T_zyz2geom(eulZYZ) \ J_geom;
J_analit = simplify(J_analit)
```

1.3 Matlab code for question 4

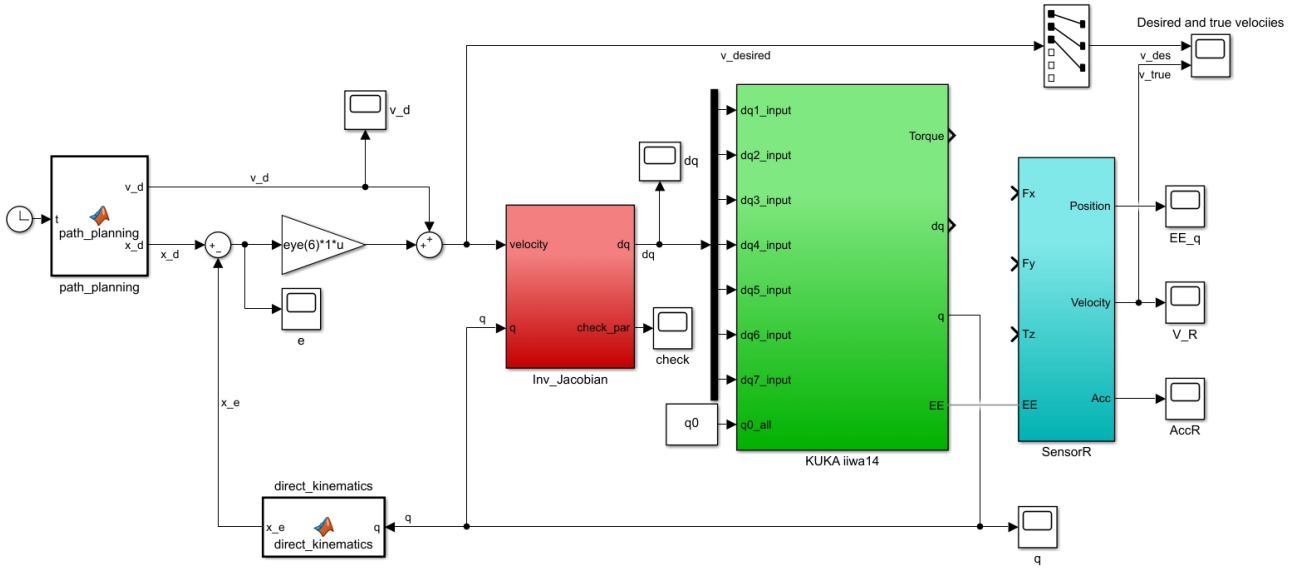


Figure 4: *Simulink block diagram for the KUKA robot*

```
function [dq,check_par] = Inv_Jacobian(q,v)
    J_geom = Jacobian_KUKA7(q);
    T_0_e = KUKA_FKin(q);
    eulZYZ = rotm2eul(T_0_e(1:3,1:3),'ZYZ');
    J_analyt = rb.T_zyz2geom(eulZYZ) \ J_geom;
    check_par = min(svd(J_analyt)) ;%< 1e-4;

    %% Question 4a - Operational space = 6
    % dq = J_analyt \ v;

    %% Question 4b and 4c  - Operational space = 3
    dq = J_analyt(1:3,:) \ v(1:3);


function x_e = direct_kinematics(q)
    T_0_e = KUKA_FKin(q);
    eulZYZ = rotm2eul(T_0_e(1:3,1:3),'ZYZ')';
    x_e = [T_0_e(1:3,4);
           eulZYZ];

function [v_d,x_d] = path_planning(t)
    %% Question 4a
    % x_d = [0.1*t;
    %        zeros(5,1)];
    % v_d = [0.1;
    %        zeros(5,1)];

    %% Question 4b
    % x_d = [-0.2;
    %        -0.0008*t^3+0.012*t^2-0.2;
    %        1;
    %        zeros(3,1)];
    % v_d = [0;
    %       (3*t)/125 - (3*t^2)/1250;
    %        0;
    %        zeros(3,1)];
```

```
%% Question 4c
x_d = [-0.2;
       0.2;
       1;
       zeros(3,1)];
v_d = zeros(6,1);
```