# SSY156 - Modelling and Control of Mechatronic systems
# Assignment A03

Lucas Rath

## Question 1

For each desired end-effector position of the Omniblundle, the joint angles measured by the sensors were recorded. The algorithm chosen for the trajectory planning was the *trapezoidal velocity profile*, with maximum acceleration of $1 rad/s^2$. An initial time of 2 seconds was given to the robot to stabilize in the initial position. The result of the trajectory planning can be seen in figure 1.

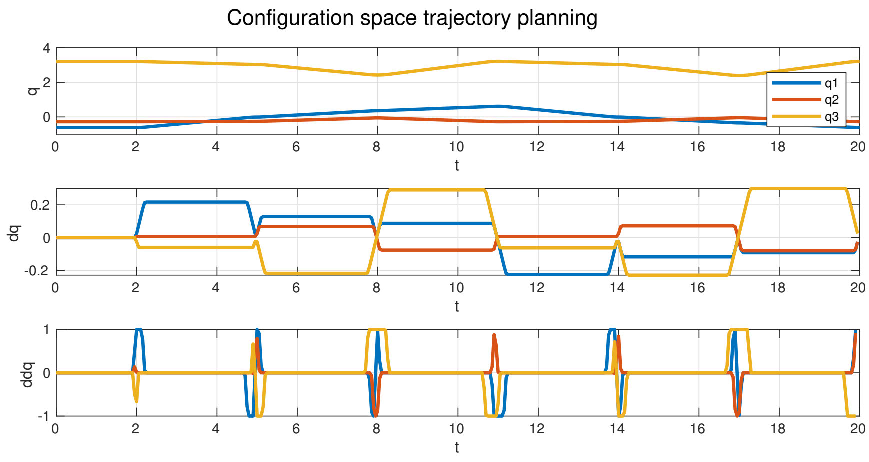


***Figure 1:*** *Trajectory planning for the recorded configuration space*

## Question 2

**A)** The linear model used to design the PID controller is the linear and decoupled representation of the robot dynamics. Starting from the original non-linear dynamics:

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F\dot{q} + g(q) = \xi \tag{1}$$

we first decompose the inertia matrix $B(q) = \bar{B} + \Delta B(q)$, such that $\bar{B}$ is constant and diagonal. Second, we put aside all the non-linear and not constant terms $d = \Delta B(q) + C(q,\dot{q})\dot{q} + g(q)$ and consider them as disturbances in the system. At the end, we arrive in the following linear and decoupled model:

$$\bar{B}\ddot{q} + F\dot{q} + d = \xi \tag{2}$$

Thereafter, we rewrite the linearized and decoupled equation above in the Laplace domain:

$$q_i = \frac{F_i^{-1}}{s\left(1 + \frac{\bar{B}_i}{F_i}s\right)}(\xi_i - d_i) = \frac{K_m}{s\,(1 + T_m s)}(\xi_i - d_i) \tag{3}$$

where the constants $K_m$ and $T_m$ were introduced in this case to facilitate the notation. We then propose a PID controller of the form:

$$PID(s) = \frac{\xi}{e} = (K_v + K_p K_v T_v) + \frac{(K_p K_v)}{s} + (K_v T_v)s \tag{4}$$

where $T_v, K_v, K_p$ are constants of the controller and $e$ is the error between the measured $q_i$ and the reference value. We proceed by calculating the close-loop transfer function $H_{cl}(s)$. By choosing $T_v = T_m$ we manage to cancel one pole and one zero of $H_{cl}(s)$, resulting in a well-known second order close-loop transfer function:

$$H_{cl}(s) = \frac{1}{1 + \frac{s}{K_p} + \frac{s^2}{K_m K_p K_v}} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5}$$

where $\omega_n$ and $\zeta$ are the natural frequency and damping coefficient. Finally, we then choose the controller coefficients to match the desired close-loop performance, which results in:

$$K_v = \frac{2\zeta\omega_n}{K_m} \qquad K_p = \frac{\omega_n}{2\zeta} \tag{6}$$

As a last improvement, in order to enhance the performance when the reference trajectory is subject to high speed and accelerations we add a feed-forward compensation, by modifying the reference value (for sake of simplicity, the derivation will be omitted in this report):

$$\theta_{md}^* = \theta_{md} + \frac{\dot{\theta}_{md}}{K_p} + \frac{\ddot{\theta}_{md}}{K_p K_m K_v} \tag{7}$$

**B)** The simulation results can be verified in figure 2 As seen, the trajectory is followed and the errors presented are very low.

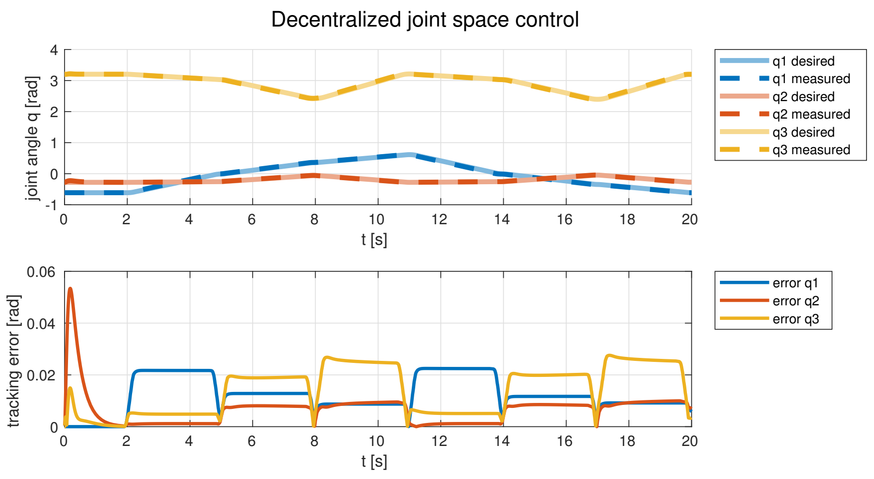


***Figure 2:*** *Decentralized joint-space control - Simulation*

## Question 3

**A)** For the simulation part, the parameters $\omega_n = 20$ and $\zeta = 1$ led to good simulation results. According to equations 4 and 6, the chosen parameters resulted in a PID with the following gains:

$$K_P = \begin{bmatrix} 3.3469 \\ 2.7819 \\ 0.7272 \end{bmatrix} \quad K_I = \begin{bmatrix} 3.5600 \\ 6.8000 \\ 2.3200 \end{bmatrix} \quad K_D = \begin{bmatrix} 0.2991 \\ 0.2102 \\ 0.0495 \end{bmatrix} \tag{8}$$

The chosen parameters, when implemented in the lab, resulted in very aggressive control, leading to unstable behavior. The best tune found was when $\omega_n = 8$ and $\zeta = 1$ for all joints, resulting in:

$$K_P = \begin{bmatrix} 0.6210 \\ 0.6083 \\ 0.1720 \end{bmatrix} \quad K_I = \begin{bmatrix} 0.5696 \\ 1.0880 \\ 0.3712 \end{bmatrix} \quad K_D = \begin{bmatrix} 0.1196 \\ 0.0841 \\ 0.0198 \end{bmatrix} \tag{9}$$

as might be noticed, these gains are much smaller than the original ones designed.

**B)** As can be seen in figure 3, the Omnibundle managed to follow the trajectory very well. The tracking errors are mostly contained around 0.05 rad.

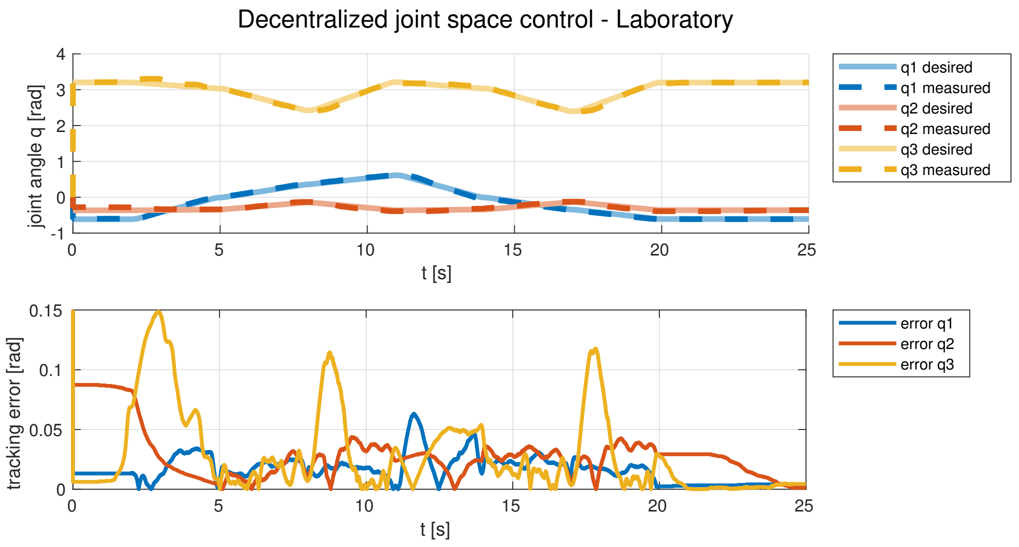


***Figure 3:*** *Decentralized joint-space control - Laboratory measurements*

## Question 4

The centralized control using inverse dynamics was implemented in Simulink. It was decided to use the coefficients $\zeta = 1$ and $\omega_n = 20$ for the error dynamics for all the three joints. The simulation results obtained are shown in figure 4a.

In the lab, however, the chosen parameters presented very bad control behaviour and had to be returned independently for each joint. The best parameters were $\zeta = [0,\ 0.3,\ 0.8]$ and $\omega_n = [35,\ 50,\ 70]$ for each joint angle. The experiment results are shown in figure 4b. As can be seen, the tracking errors were kept very low, around 0.03 rad.

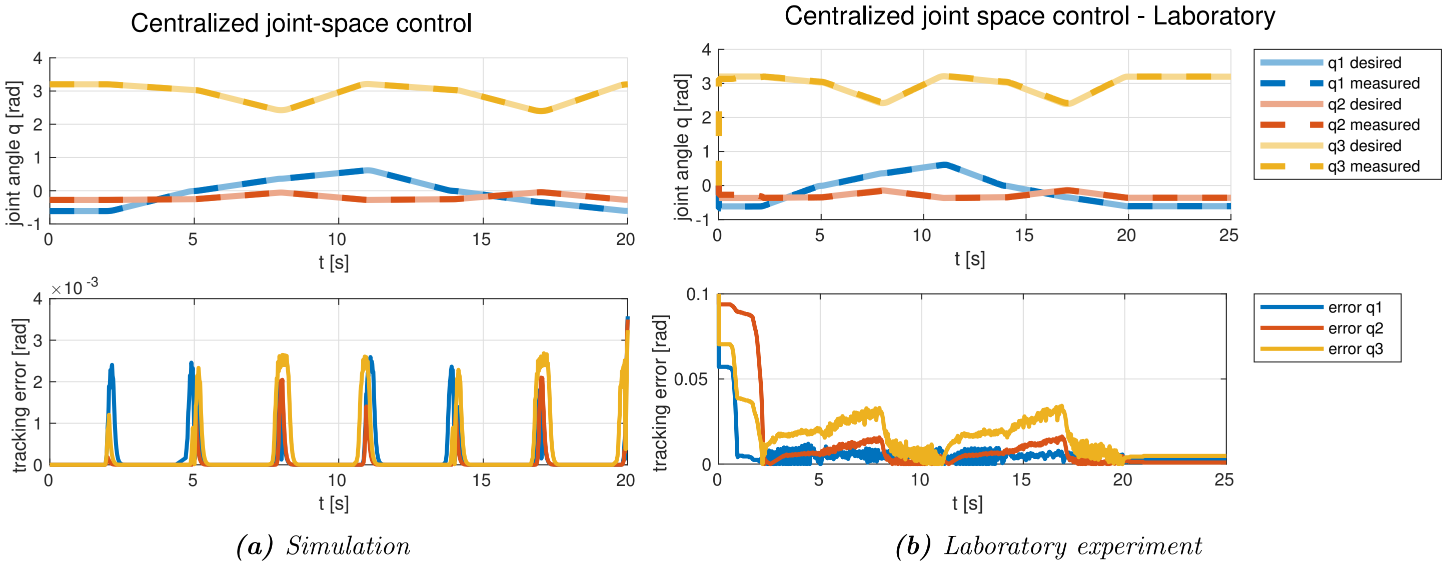


***Figure 4:*** *Centralized control using inverse dynamics*

## Question 5

The control input of the centralized control with inverse dynamics subjected to a load disturbance $\tau_d$ is given by:

$$u = n(q,\dot{q}) + B(q)y + \tau_d, \qquad \text{where} \qquad y = \ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q} \tag{10}$$
$$\Rightarrow \quad u = n(q,\dot{q}) + B(q)(\ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}) + \tau_d \tag{11}$$

where the joint position error $\tilde{q} = q_d - q$. Considering that we have modelled the perfect dynamics, we can write it as $B(q)\ddot{q} + n(q,\dot{q}) = u$. Coupling then the dynamics with the equation 11, we get:

$$B(q)\ddot{q} + n(q,\dot{q}) = n(q,\dot{q}) + B(q)(\ddot{q}_d + K_D\dot{\tilde{q}} + K_P\tilde{q}) + \tau_d \tag{12}$$
$$\Rightarrow \quad \ddot{\tilde{q}} + K_D\dot{\tilde{q}} + K_P\tilde{q} = -B^{-1}(q)\tau_d \tag{13}$$

As can be seen in equation 13, the error dynamics now contains a constant input $-B^{-1}(q)\tau_d$, which will avoid the error to converge to zero in steady state. However, when considering $\tau_d = [0,\ d,\ 0]^\mathsf{T}$, the first position of the vector $-B^{-1}(q)\tau_d$ will be zero, meaning that there will be no stationary error for the first joint. This can be further checked with simulation results for $d = 0.1$, see figure 5. The error of joint 1 is always kept around 0 without any stationary error.

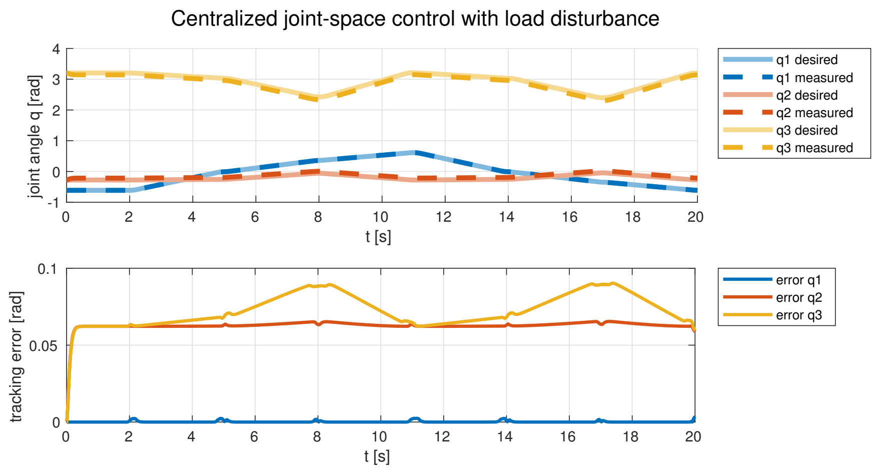


***Figure 5:*** *Centralized control using inverse dynamics subject to load disturbance - Simulation*

## Question 6

Choosing $T = 5s$ and implementing again the *trapezoidal velocity profile* with maximum acceleration of 0.5 m/s, we get the following trajectory in figure 6. Two seconds in the beginning of the simulation were given so the robot could stabilize first in its initial position.

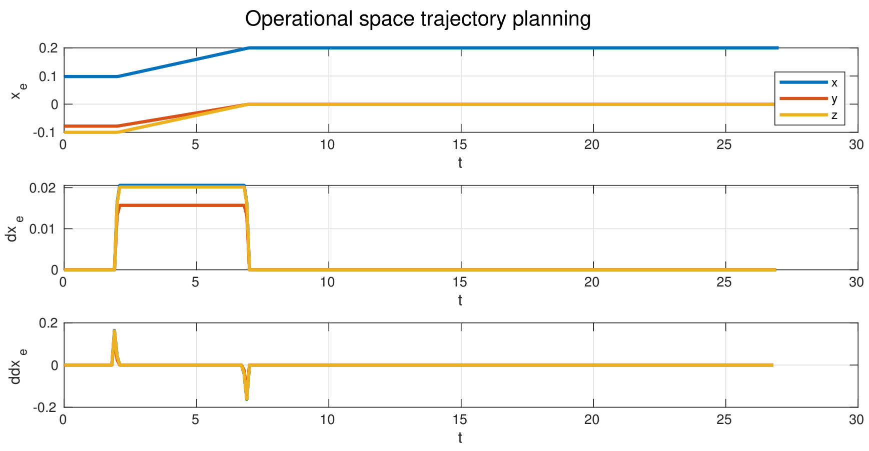
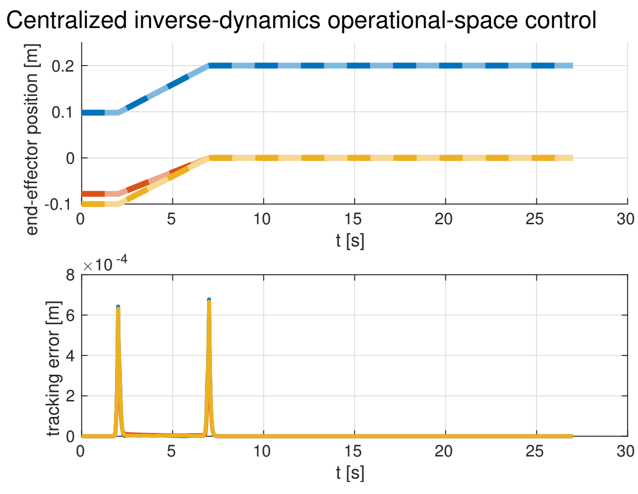


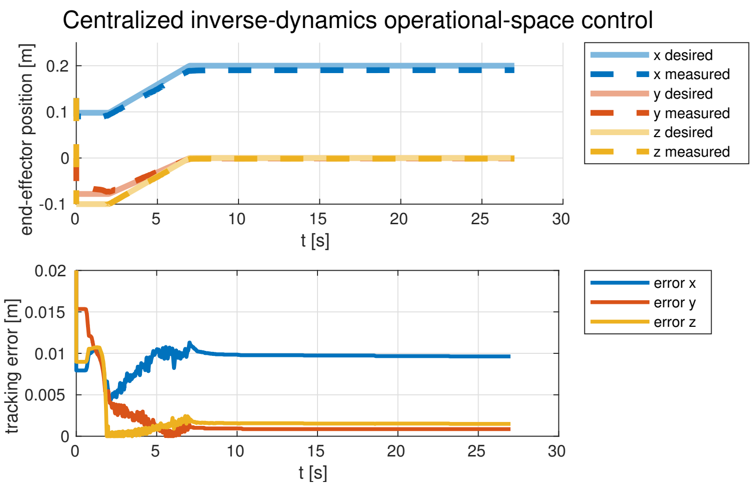
***Figure 6:*** *Trajectory planning for the operational space*

## Question 7

Please, see code in the appendix.



*(a) Simulation*



*(b) Laboratory experiment*

***Figure 7:*** *Operational space centralized control using inverse dynamics*

## Question 8

First, we calculate the desired end-effector acceleration, which will lead to an impedance behaviour:

$$z = M_D^{-1}\left(M_D * \ddot{x}_d + K_P * (x_d - x_e) + K_D * (\dot{x}_d - \dot{x}_e)\right) \tag{14}$$

then we calculate the desired joint acceleration by applying inverse differential dynamics:

$$y = J_{A,P}^{-1}\left(z - \dot{J}_{A,P}\,\dot{q}\right) \tag{15}$$

And finally we calculate the joint torques by means of inverse dynamics:

$$u = n(q,\dot{q}) + B(q)y \tag{16}$$

Gathering these equations with the dynamics equation, results in an error with the following dynamics:

$$M_D\ddot{\tilde{x}} + K_D\dot{\tilde{x}} + K_P\tilde{x} = M_D J B^{-1} J^{\mathsf{T}} h_e \tag{17}$$

where $h_e$ is an external force applied to the system. Equation 17 is then a second order differential equation, which can be characterized by $M_D^{-1}K_D = 2\zeta\omega_n$ and $M_D^{-1}K_P = \omega_n^2$.

Considering the error inertia matrix $M_D$ as identity matrix, an under-damped behavior for the error can be achieved by choosing $\zeta < 1$ and over-damped by choosing $\zeta > 1$. The natural frequency $\omega_n$ might be designed arbitrarily and represents the system responsiveness.

## Question 9

Replacing $h_e$ in equation 17 by an force in X direction $h_e = [h_x,\ 0,\ 0]$ and multiplying both sides by $M_D^{-1}$ we get:

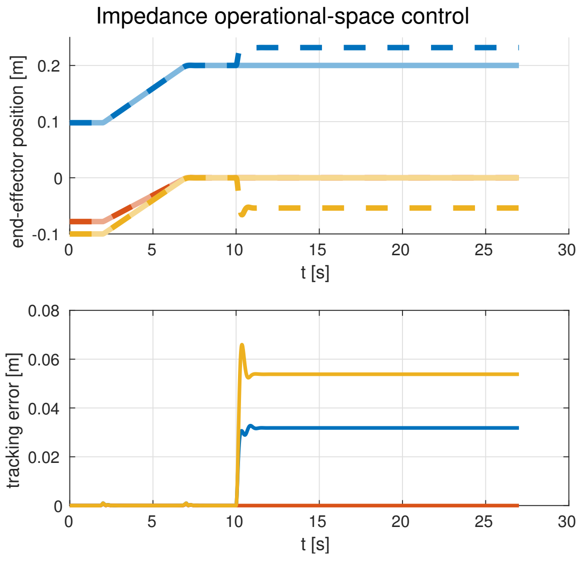
$$\ddot{\tilde{x}} + M_D^{-1}\,K_D\dot{\tilde{x}} + M_D^{-1}\,K_P\tilde{x} = JB^{-1}J^{\mathsf{T}}\,[h_x,\ 0,\ 0]^{\mathsf{T}} \tag{18}$$

The analytic expression of the forces affecting the error dynamics of the X-coordinate is however to long to be displayed in this report. The following code may be then used:

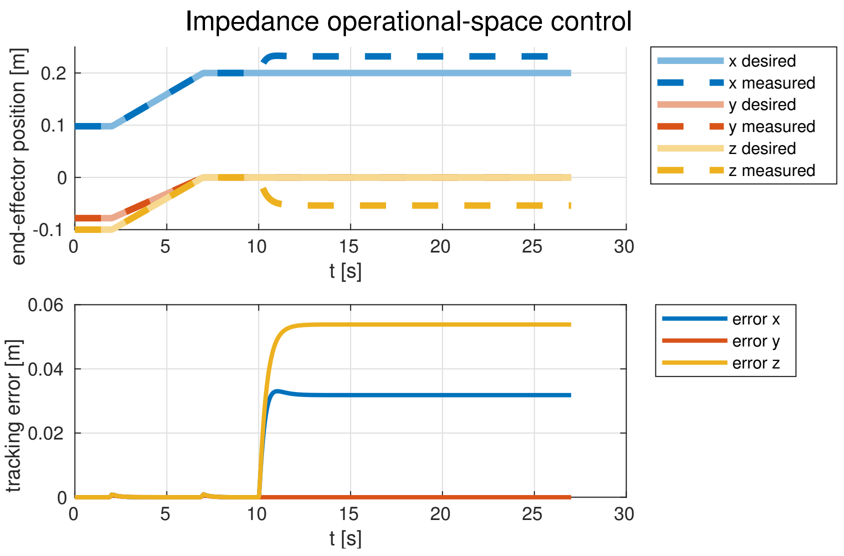
$\ddot{\tilde{x}}_1 + M_D^{-1}\,K_D\dot{\tilde{x}}_1 + M_D^{-1}\,K_P\tilde{x}_1 =$ `[1 0 0] * J_A(1:3,1:3) * inv(B) * J_A(1:3,1:3).' * [hx 0 0].'`

## Question 10

Choosing $M_D = I$, $\omega_n = 10$ for both controllers, $\zeta = 2$ for the over-damped and $\zeta = 0.5$ for the under-damped impedance control, the following simulation results are obtained, see figure 8. Clearly, the tracking error behaved as under and over-damped as expected according to the chosen $\zeta$.



***(a)*** *Under-damped behaviour*

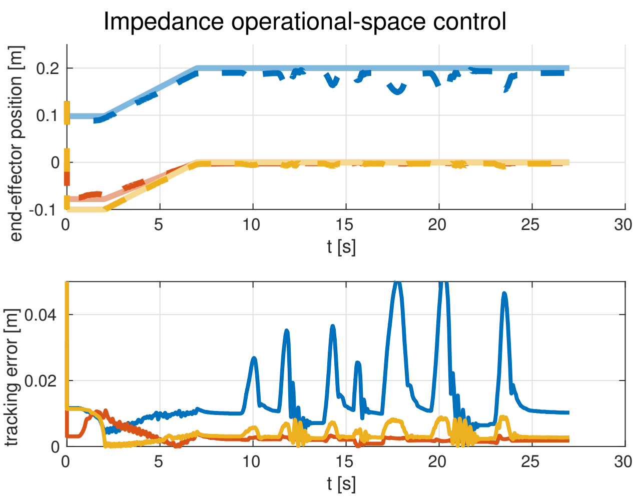


***(b)*** *Over-damped behaviour*

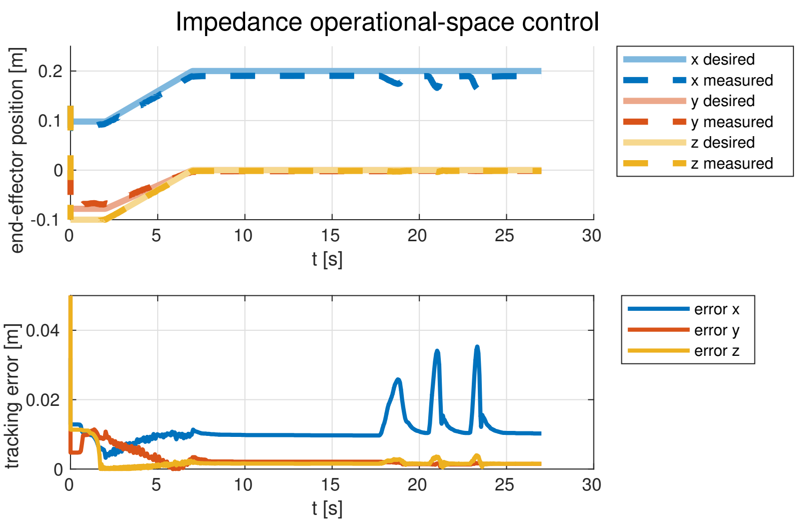
***Figure 8:*** *Operational space impedance control - Simulation*

## Question 11

In the lab, under and over-damped parameters were tested. In both cases, a small force was applied in the -X direction of the end-effector and then released after 1 second. This procedure was done repeatedly for some times. Clearly, the under-damped controller made the end-effector oscillate when returning to the desired position while the over-damped returned without any overshoot. Although the results are not exactly the same when compared to the simulation, the observed behaviour matches to what was expected.



***(a)*** *Under-damped behaviour*



***(b)*** *Over-damped behaviour*

***Figure 9:*** *Operational space impedance control - Laboratory Experiment*

## Appendix

***Listing 1:*** *Centralized joint-space control*

```
function xi = centr_control_joint_space(ddq_d, dq_d, q_d, q, dq)

    % initialize output - torque
    xi = zeros(3,1);

    % controller gains
    zeta = 1;
    wn = 20;
    K_D = diag(repmat(2*zeta*wn,3,1));
    K_P = diag(repmat(wn^2,3,1));

    % desired joint acceleration
    y = ddq_d + K_P * ( q_d - q ) + K_D * ( dq_d - dq );

    % inverse dynamics
    xi = inv_dyn_omni_bundle_fun(9.8, q, dq, y);
end
```

***Listing 2:*** *Centralized operational-space control*

```
function [xi, x_e] = centr_control_operational_space(ddx_d, dx_d, x_d, q, dq)

    % initialize output
    xi = zeros(3,1);

    % direct dynamics
    x_e = dir_kin_omni_bundle_fun(q);

    % direct differential kinematics
    A_0_e = A_0_e_omni_bundle_fun(q);
    eulZYZ = rotm2eul(A_0_e(1:3,1:3),'ZYZ')';
    J_A = anal_jacob_omni_bundle_fun(q,eulZYZ);
    dx_e = J_A(1:3,1:3) * dq;

    % dJA
    dJ_A = dJ_analit_P_omni_bundle_fun(q,dq);

    % controller gains
    zeta = 1;
    wn = 20;
    K_D = diag(repmat(2*zeta*wn,3,1));
    K_P = diag(repmat(wn^2,3,1));

    % desired end-effector acceleration
    z = ddx_d + K_P * (x_d - x_e) + K_D * (dx_d - dx_e);

    % inverse differential kinematics - desired joints acceleration
    y = J_A(1:3,1:3) \ ( z - dJ_A * dq ); %

    % inverse dynamics
    xi = inv_dyn_omni_bundle_fun(9.8, q, dq, y);
end
```

***Listing 3:*** *Impedance operational-space control*

```
function [xi, x_e] = control_impedance_operspace(ddx_d, dx_d, x_d, q, dq, t)

    % initialize output
    xi = zeros(3,1);

    % direct dynamics
    x_e = dir_kin_omni_bundle_fun(q);

    % direct differential kinematics
    A_0_e = A_0_e_omni_bundle_fun(q);
    eulZYZ = rotm2eul(A_0_e(1:3,1:3),'ZYZ')';
    J_A = anal_jacob_omni_bundle_fun(q,eulZYZ);
    dx_e = J_A(1:3,1:3) * dq;

    % dJA
    dJ_A = dJ_analit_P_omni_bundle_fun(q,dq);

    % controller gains
    % Over-damped
    zeta = 2;
    wn = 10;

    % Under-damped
    %     zeta = 0.5;
    %     wn = 10;

    M_D = eye(3);
    K_D = M_D * diag(repmat(2*zeta*wn,3,1));
    K_P = M_D * diag(repmat(wn^2,3,1));


    % desired end-effector acceleration
    z = M_D \ ( M_D * ddx_d + K_P * (x_d - x_e) + K_D * (dx_d - dx_e) );

    % inverse differential dynamics - desired joints acceleration
    y = J_A(1:3,1:3) \ ( z - dJ_A * dq ); %

    % inverse dynamics
    xi = inv_dyn_omni_bundle_fun(9.8, q, dq, y);


    if t>=10
        xi = xi + J_A(1:3,1:3)*[1 0 0]';     % apply disturbance in x direction
    end
end
```