Xuan - Pu Autumn Hong
03749607
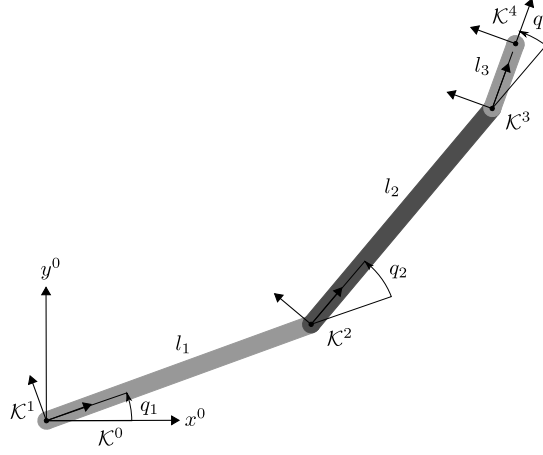
2 Theory

2.1 Kinematics and dynamics

Figure 1: Kinematic structure of the 3DOF planar manipulator

In order to demonstrate the basic concept of impedance control, the model of a planar manipulator with three rotational degrees of freedom (DOF) will be addressed. The robot model consists of three rigid bodies which are connected via hinge joints. The kinematic structure of the robot is shown in Fig. 1. To describe the manipulator configuration, joint position variables $\mathbf{q} \in \mathbb{R}^n$ (where $n = 3$ is the number joint coordinates) are introduced. The relative position and orientation of frame $\mathcal{K}^{i+1}$ w.r.t. $\mathcal{K}^i$ is described by the homogeneous transformation

$$^i\mathbf{T}_{i+1} = \begin{bmatrix} ^i\mathbf{R}_{i+1} & ^i\mathbf{p}_{i+1} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{1}$$

where $^i\mathbf{R}_{i+1} \in SO(3)$ and $^i\mathbf{p}_{i+1} \in \mathbb{R}^3$ are the rotation matrix and position vector, respectively. By choosing a specific representation for the rotation (e.g., Cardan angles, Euler angles, etc.), Cartesian coordinates of the end-effector pose (i.e., position and orientation)

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \in \mathbb{R}^m \tag{2}$$

can be derived from the homogeneous transformation between end effector and base frame. The vector function $\mathbf{f}(\mathbf{q})$ gives a mapping between the n joint coordinates and the m Cartesian coordinates, where $n > m$ for redundant kinematic chains. For impedance control (and other Cartesian control schemes), the relation

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\,\dot{\mathbf{q}} \tag{3}$$

is needed, which maps joint velocities to Cartesian velocities. Here, $\mathbf{J}(\mathbf{q}) = \frac{\partial \mathbf{f}(\mathbf{q})}{\partial \mathbf{q}} \in \mathbb{R}^{m \times n}$ is the Jacobian matrix. Notice that the Jacobian is expressed w.r.t. a coordinate frame, which depends on the choice of the reference frame for $\mathbf{f}(\mathbf{q})$. The components of $\mathbf{f}(\mathbf{q})$ and $\mathbf{J}(\mathbf{q})$ for the case of the 3DOF planar manipulator (see Fig.1) are listed in appendix A.

The dynamic model of the rigid body robot manipulator can be written:

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} \tag{4}$$

Here $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ the Coriolis/centrifugal matrix, and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^3$ the vector of gravity forces/torques. For the case of the 3DOF planar manipulator, the components of $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{g}$ are listed in the appendix A.

From a control point of view, two important properties of the matrices in (4) have to be remarked. The mass matrix is *symmetric* and *positive definite*, i.e.

$$\mathbf{M}(\mathbf{q}) = \mathbf{M}(\mathbf{q})^T, \quad \mathbf{y}^T \mathbf{M}(\mathbf{q})\,\mathbf{y} > 0, \quad \forall \mathbf{y}, \mathbf{q} \neq \mathbf{0} \in \mathbb{R}^n, \tag{5}$$

2

where $\mathbf{y} \in \mathbb{R}^n$ is arbitrary. This property is important for stability analyses. If $\mathbf{C}$ is derived via _Christoffel symbols_, the matrix $\boxed{\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})}$ is _skew-symmetric_, i.e.

$$\mathbf{y}^T \left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \right) \mathbf{y} = 0, \quad \forall \mathbf{y}, \mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^n, \tag{6}$$

which guarantees _passivity_ of the dynamic system (4).

2.2 Joints control*

The simplest way to control the configuration of a manipulator is to control directly the joints positions. Given a vector of desired joints angles $\mathbf{q}_d \in \mathbb{R}^n$, the error between the current and desired manipulator configuration is:

$$\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q} \tag{7}$$

Problem* 2.2.1: _Considering only the case of regulation ($\dot{\mathbf{q}}_d = 0$), write the law of the PD controller for reaching the desired joints position $\mathbf{q}_d$._

$$\tau = k_P(q_d - q) - k_D \dot{q} + g(q)$$
$$\underbrace{\qquad}_{=\,0}$$

$$\tau = k_P \tilde{q} + g(q)$$

Problem* 2.2.2: _Using the control law derived in Problem 2.2.1, write the equation of the dynamics of the controlled system._

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = k_P \tilde{q} + g(q)$$
$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} - k_P \tilde{q} = 0$$

Neglecting the centrifugal and Coriolis forces and assuming a _quasi-stationary_ variation, the controlled dynamics can be written:

$$\mathbf{M}(\mathbf{q}_0)\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}} = 0 \tag{8}$$

Problem** 2.2.3: _Considering a constant stiffness matrix $\mathbf{K}_p$, derive a damping matrix $\mathbf{K}_d$ for achieving a given damping factor ζ of the system (8)._

$$K_P = K_{P_1} K_{P_1}$$
$$M(q_0) = M_1 M_1$$

$$K_d = \xi (M_1 K_{P_1} + K_{P_1} M_1)$$
$$0 \leq \xi \leq 1$$

3

2.3 Inverse kinematics control**

The robot tasks need to be specified most of the time in the operational space. A way is therefore needed in order to command the robot to reach a desired EE position in the Cartesian space.

A possible scheme of Cartesian control is represented in Fig.2. In this approach, the desired Cartesian position of the EE is inverted in order to obtain a desired joint position $\mathbf{q}_d$. The desired position $\mathbf{q}_d$ is then commanded to the robot by means of a joint controller, as developed in 2.2.
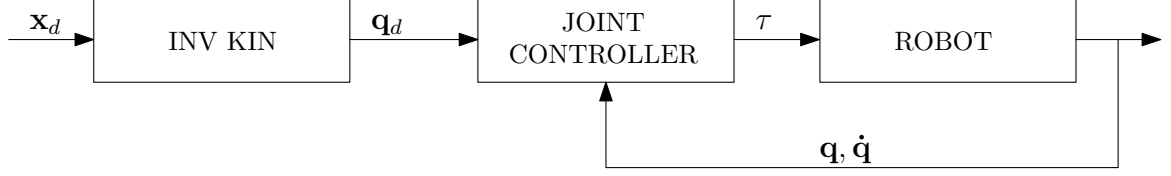
Figure 2: Scheme of Cartesian control using inverse kinematics.

Inverting the kinematics of a robot requires the solution of the equation:

$$\mathbf{x}_d = \mathbf{f}(\mathbf{q}_q) \tag{9}$$

where $\mathbf{x}_d = [x, y, \phi]^T$ is the desired EE position for the 3DOF planar robot, $\mathbf{f}(\mathbf{q})$ is the forward kinematics function, and $\mathbf{q}_d$ is the unknown joints position.

The problem of the inverse kinematics is complex and not always solvable in closed form. For complicated robot structures a numerical procedure is therefore needed in order to find one of the multiple solutions of (9).

Considering the monodimensional case of a continuos and regular function $y = f(x)$, one solution could be found by means of the Newton's method. The function $f(x)$ can be *locally* approximated by its tangent. The intersecation of this tangent and the abscissa axis gives a better approximation of the local solution x_d (i.e. such that $y_d = f(x_d)$):

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad k = 0, 1, \dots, n \tag{10}$$

Under certain hyphothesis, the iterative algorithm converges to the solution x_d. The iterations can be stopped when a convergence criterion has been satisfied:

$$|y_d - f(x)| \le \epsilon \tag{11}$$

Problem **2.3.1:** *Extend the Newton's method in order to derive an algorithm for solving the inverse kinematics problem.*

Problem* 2.3.2: *What happens in the vicinity of a singularity?*

Another approach for solving (9) is the gradient method. The desired joints position is found minimizing a cost function which indicates the distance of the calculated Cartesian position from the desired Cartesian position:

$$H(\mathbf{q}) = \frac{1}{2}\|\mathbf{x}_d - \mathbf{f}(\mathbf{q})\|^2 \tag{12}$$

Problem** 2.3.3: *Minimize equation* (12) *and derive the corresponding expression for the algorithm for solving the inverse kinematics problem.*

Problem* 2.3.4: *What happens in the vicinity of a singularity?*

Problem* 2.3.5: *Compare the two methods in terms of computational effort, behavior on singularity, convergence rate and application to redundant robots.*

Problem* **2.3.6:** *Draw the equivalent loop scheme that implements the algorithm based on the transposed Jacobian.*

2.4 Translational Cartesian Impedance control*

An alternative way to control the TCP pose is by means of the operational space formulation, as represented in Fig.3
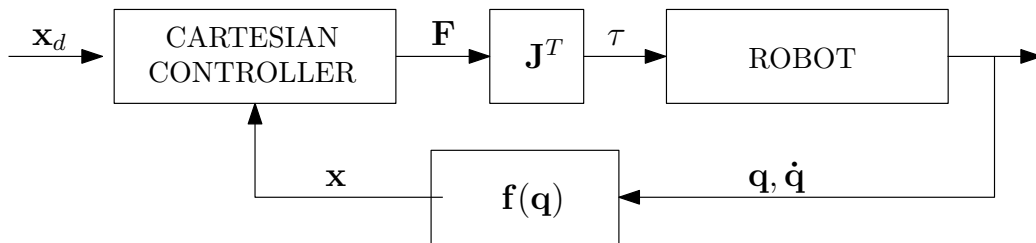

Figure 3: Scheme of cartesian control using the transpose Jacobian

Problem* **2.4.1:** *The mapping from joint to Cartesian velocity space is given by (3). Derive the mapping between Cartesian forces and joints torques.*

$$\dot{x} = J(q)\, \dot{q}$$

$$\tau = J^T_{(q)} F + g(q) + \underline{\underline{\tau_N}}$$

$$\hookrightarrow \text{null-space torque}$$

The following steps will lead to the Cartesian impedance control law.

Problem* **2.4.2:** *Considering a translational error $\tilde{x} = x - x_d$ and a constant, positive definite, symmetric stiffness matrix $K_t \in \mathbb{R}^2$, write the law of a spring force F_K between the TCP of the manipulator and the desired point.*

$$F_k = -K_t \cdot \tilde{x}$$

Problem* 2.4.3: *Calculate the planar* <mark>*Cartesian force* $\mathbf{F}_D$</mark> *resulting from a viscous damper with the constant symmetric damping matrix* $\mathbf{D}_t \in \mathbb{R}^2$.

$$F_D = -D_t \cdot \dot{\tilde{x}}$$

Problem* 2.4.4: *The* <mark>*Cartesian impedance control law*</mark> *for the translational case can now be set up as the superposition of the solutions of the Problems 2.4.2 and 2.4.3. The transformation from Cartesian forces to joint torques was derived in Problem 2.4.1.* <mark>*Write the complete control law for the translational Cartesian impedance controller such that it can be commanded to the joint actuators.*</mark>

$$\tau = J^T_{(q)}(-K_t \cdot \tilde{x} - D_t \cdot \dot{\tilde{x}}) + g(q) + \tau_N$$

$$K_t, D_t \in R^{2 \times 2}$$

2.5 Nullspace optimizations**

In Problem 2.4.4 a control law was treated which determines the translational Cartesian forces, i.e., the task space has $m = 2$ DOF. Since the considered manipulator has $n = 3$ joints, the configuration (joint angles) could change, while the end-effector is in a fixed position. The resulting motion is called a *nullspace motion*. In order to control the nullspace motion, a control law for the $r = n - m = 1$ *redundant* DOF will be derived. The complete impedance controller can be obtained summing up the TCP impedance torque $\boldsymbol{\tau}_i$ derived in Eq.(2.4.4) and a nullspace torque $\boldsymbol{\tau}_n$.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_i + \boldsymbol{\tau}_n \tag{13}$$

Problem** 2.5.1: *Write a control law* $\boldsymbol{\tau}_n$ *for the nullspace and demonstrate that doesn't interfere with the TCP torque.*

Problem** 2.5.2: *Write the expression of the pseudo-inverse* $\mathbf{J}^+$ *and demonstrate that* $\mathbf{J}\mathbf{J}^+ = \mathbf{I}$.

Hint 1: *Use the simplest form of pseudo-inverse (Moore–Penrose)*

Problem* 2.5.3: *Define a damping nullspace torque $\boldsymbol{\tau}_0$*

The nullspace additional degree of freedom can be also used for making some optimizations on the configuration of the manipulator. A possible use-case is the avoidance of singularities by means of additional nullspace torques that don't interfere with the TCP motion.

Problem* 2.5.4: *Using the mapping from Cartesian forces to joint torques $\boldsymbol{\tau} = \mathbf{J}^T\mathbf{F}$, give a qualitative interpretation of the joint torques which result from forces acting in singular directions (how does the robot behave at a singularity?)*

An index of manipulability is $\mathbf{m}(\mathbf{q}) = \sqrt{\det(\mathbf{J}\mathbf{J}^T)}$ and for the considered 3DOF manipulator it's given by:

$$m_{\text{kin}}(\mathbf{q}) = \sqrt{l_1^2 l_2^2 \sin^2 q_2}. \tag{14}$$

As the determinant vanishes for singular $\mathbf{J}(\mathbf{q})$, this measure gives locally the distance to the singularity. A possible approach is to define a force field which repels the manipulator from singularities when the manipulability is too low.

Problem* 2.5.5: *Define a quadratic potential function with a scalar coefficient k_s controlling the gain of the singularity avoidance. Use a piecewise definition to restrict the potential to the vicinity of the singularity.*

Problem** 2.5.6: *Write a control torque that implements the singularity avoidance by means of the potential defined in 2.5.5.*

Problem* 2.5.7: *Write the complete control law for the Cartesian impedance controller from Problem 2.4.4 and the nullspace optimizations (nullspace damping and singularity avoidance).*

2.6 Full Cartesian Impedance controller***

The translational Cartesian impedance controller from Section 2.4 allows only to set the position of the TCP and not its orientation. Therefore, the controller should be generalized to also allow for rotational stiffnesses. The general controller structure is maintained but extended for a rotational component. Although our 3DOF planar model can be described by $\mathbf{x} \in \mathbb{R}^3$, in the following Cartesian coordinates $\mathbf{x} \in \mathbb{R}^6$ are considered for generality.

The full controller can be designed splitting up $\tilde{\mathbf{x}}$ into two components $\tilde{\mathbf{x}}_t \in \mathbb{R}^3$ and $\tilde{\mathbf{x}}_r \in \mathbb{R}^3$ describing respectively the end effector position and orientation.

Problem* 2.6.1: *Write the Cartesian stiffness matrix $\mathbf{K}_x \in \mathbb{R}^6$, partitioned into the translational stiffness $\mathbf{K}_t$, a rotational stiffness $\mathbf{K}_r$, and the coupling stiffnesses $\mathbf{K}_c$.*

$$\mathbf{K}_x = \begin{bmatrix} K_t & K_c \\ K_c & K_r \end{bmatrix} \in \mathbb{R}^{6 \times 6}$$

$$K_t, K_r, K_c \in \mathbb{R}^{3 \times 3}$$

The coupling stiffness $\mathbf{K}_c$ is omitted in the following, therefore $\mathbf{K}_c = 0$. In contrast to the vector difference as used in Section 2.4, the offset between the TCP frame and the desired frame is expressed by a homogeneous transformation matrix.

Problem* **2.6.2:** *Write the transformation $^{TCP}\mathbf{T}_d$ from the TCP frame $\mathcal{K}^{TCP}$ to the desired frame $\mathcal{K}^d$.*

given $^{0}T_{TCP}$ & $^{0}T_d$

$$^{0}T_d = {}^{0}T_{TCP} \cdot {}^{TCP}T_d$$

$$^{TCP}T_d = {}^{0}T_{TCP}^{-1} \cdot {}^{0}T_d$$

The translational part of $^{TCP}\mathbf{T}_d$ can be directly used for the translational impedance controller. For the rotational stiffness, the situation is more complex. As there is no global minimal representation of $SO(3)$, the choice of orientation coordinates for the use in Cartesian controllers is not straight forward. In the following, a solution based upon unit quaternions will be worked out.

Unit quaternions are a generalization of complex numbers and can represent rotations in $SO(3)$ in a similar way as complex numbers represent planar rotations. The advantage of unit quaternions is that they give a global parameterization and are therefore singularity free in $SO(3)$. A unit quaternion consists of a scalar component η and a vector component ϵ. The components can be calculated by

$$
\begin{aligned}
\eta &= \frac{1}{2}\sqrt{1 + r_{11} + r_{22} + r_{33}} \\
\epsilon_1 &= \frac{r_{32} - r_{23}}{4\eta} \\
\epsilon_2 &= \frac{r_{13} - r_{31}}{4\eta} \\
\epsilon_3 &= \frac{r_{21} - r_{12}}{4\eta}
\end{aligned}
\tag{15}
$$

where

$$
\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}.
$$

The rotational error and the Cartesian rotational elastic force can then be computed as:

$$
\tilde{\mathbf{x}}_r := 2\epsilon \quad \Longrightarrow \quad \mathbf{F}_r = -\mathbf{K}_r \tilde{\mathbf{x}}_r
\tag{16}
$$

This choice of coordinates uses again only three parameters and is therefore also only a local parametrization. Its advantage is that it has no discontinuity and the only drawback is an unstable equilibrium point at $\eta = \pm\pi$.

Problem** **2.6.3:** *Derive a Jacobian for mapping the rotational forces to joints torques consistently with the quaternion representation used.*

$$\dot{x}_j = J_j(q)\dot{q}$$
$$\tau = J_j^T(q) F_j$$
with $\dot{x}_j = \begin{bmatrix} v_j \\ \omega_j \end{bmatrix}$; $F_j = \begin{bmatrix} f_j \\ m_j \end{bmatrix}$

$$J(q) = \frac{\partial f(q)}{\partial q} \text{ with } x = f(q)$$

$$\dot{x}_R = J_R \omega$$

$$\dot{x} = \begin{bmatrix} v \\ \dot{x}_R \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0 \\ 0 & J_R \end{bmatrix}\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} I_{3\times3} & 0 \\ 0 & J_R \end{bmatrix} J(q)\dot{q}$$

x-Cartesian

$$J_R = \begin{bmatrix} \eta & \epsilon_3 & -\epsilon_2 \\ -\epsilon_3 & \eta & \epsilon_1 \\ \epsilon_2 & -\epsilon_1 & \eta \end{bmatrix}$$

10

Problem** 2.6.4: *Assume a unit axis pointing along one of the Cartesian axes.* ~~*Why is the choice of the quaternions for the spring torque*~~ *convenient compared to a linear torque function? Draw and compare qualitatively the plots of the joints torques for both cases.*

The problem description of a unit axis pointing is on $[0, 2\pi]$ in one dimensional system. While a linear torque function must have a discontinuity on $[0, 2\pi]$, the quaternions' choice won't have this problem.

Linear Spring Torque :

Quaternions Spring Torque :

As in the case of the PD joints controller, the damping matrix $\mathbf{D}$ shouldn't be held constant because the Cartesian mass matrix $\mathbf{M}_x(\mathbf{q})$ is not constant.

Problem* 2.6.5: *Use (3) and the result from Problem 2.4.1 to transform the joint space dynamics ((4)) to Cartesian coordinates.* ==*Give an expression for the Cartesian mass matrix* $\mathbf{M}_x(\mathbf{q})$. *and Coriolis matrix* $\mathbf{C}_x(\mathbf{q})$==

$$\begin{cases} \dot{x} = J(q)\, \dot{q} \\ \tau = J^{\mathsf{T}}(q) F + g(q) + \tau_N \end{cases}$$

$$M_x(q) = J^{-\mathsf{T}}(q)\, M(q)\, J^{-1}(q)$$

$$C_x(q)\,\dot{x} = J^{-\mathsf{T}}(q)\, C(q,\dot{q})\,\dot{q} - M_x(q) J(q)\dot{q}$$

Assuming that the gravity is compensated and neglecting velocity dependent terms, the equation of motion of the robot can be written as:

$$\mathbf{M}_x(\mathbf{q})\ddot{\mathbf{x}} = \mathbf{F} \tag{17}$$

where $\mathbf{M}_x(\mathbf{q}) \in \mathbb{R}^{6\times 6}$ is the Cartesian mass matrix and $\mathbf{F} \in \mathbb{R}^6$ is the generalized forces vector acting on the mass. Precisely, ==$\mathbf{F} = \mathbf{F}_c + \mathbf{F}_{\text{ext}}$ where $\mathbf{F}_c$ are the controller forces== and $\mathbf{F}_{\text{ext}}$ ==are the external forces.== In order to ensure a desired damping of the controller dynamics derived in Eq.(17), the ==matrix $\mathbf{D}$ should be accordingly designed.==

Problem** 2.6.6: *Considering a constant stiffness matrix* $\mathbf{K}$*, derive a damping matrix* $\mathbf{D}$ *for achieving a given damping factor* ζ *of the complete system derived in Eq.(17).*

$$D_x = \zeta \left(M_x K_x + K_x M_x \right)$$

Problem* 2.6.7: *Write the* ==*expression of the damping torque*== *of the full impedance controller.*

$$F_D = -D_x (\dot{x} - \dot{x}_d)$$

$$\tau_D = J^{\mathsf{T}} F_D = -J^{\mathsf{T}} D_x (\dot{x} - \dot{x}_d)$$

Problem* 2.6.8: *Write the Full Cartesian Impedance controller, consisting of the sum of the translational and rotational stiffness, and damping derived in Problem 2.6.7. Define the impedances in TCP coordinates. Make sure all matrices are relating to the correct frames.*

$$F_k = -K_x \widetilde{x}$$

$$\tau_k = J^T F_k = -J^T K_x \widetilde{x}$$

$$\tau = \tau_k + \tau_D = -J^T (D \dot{\widetilde{x}} + K_x \widetilde{x})$$

2.7   Collision detection**

One of the features needed for allowing a safe human-robot interaction is the ability to detect collisions. A possible way to detect a collision is to compare the commanded torques and the effective torques deduced from the current state of the system.

τ_F

τ

Robot dynamics

τ_K

1. Collision force
2. Friction force
3. Joint Error

$\tau_k = J_k^T(q) F_k$

q $\dot{q}$

Disturbance detector

$\hat{\tau}_K \approx \tau_K + \tau_F$
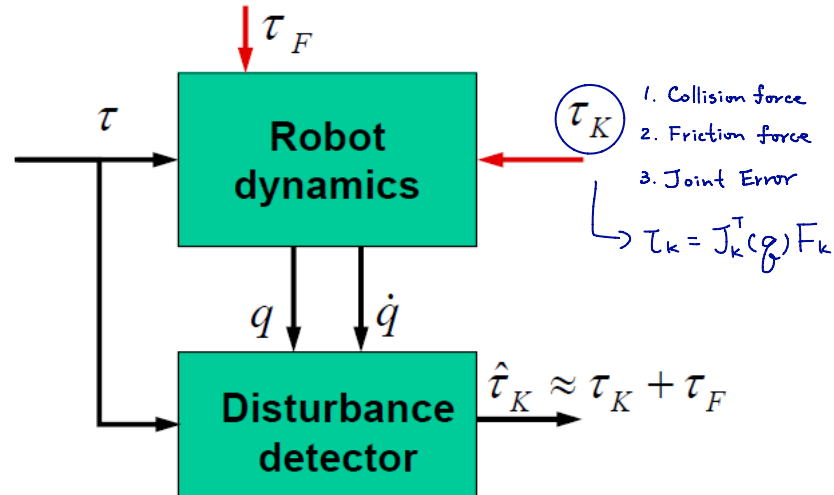
Figure 4: Scheme of estimation of the external torques

This scheme has the disadvantage that the estimated torque is the sum of both the external torques and the friction forces.

Problem* 2.7.1: *Draw a scheme for estimating only the external torques.*

τ_F

τ_m motor dynamics τ_J rigid body dynamics τ_{ext}

q $\dot{q}$

disturbance observer $\hat{\tau}_{ext} \approx \tau_{ext}$
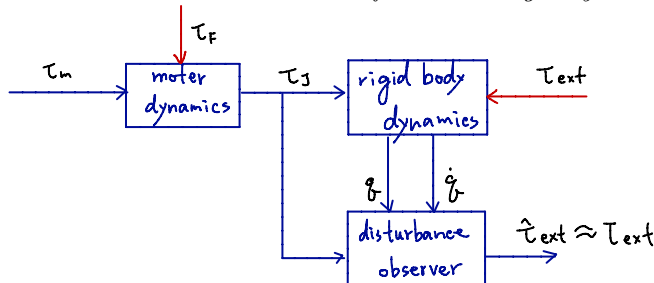
The effective torque $\boldsymbol{\tau}_e$ can be deduced either from the measurements or from the desired trajectory using the dynamic model of the manipulator:

$$\boldsymbol{\tau}_e = \mathbf{M}(\mathbf{q}_n)\ddot{\mathbf{q}}_n + \mathbf{C}(\mathbf{q}_n, \dot{\mathbf{q}}_n)\mathbf{q}_n + \mathbf{g}(\mathbf{q}_n) \qquad \text{from measurements} \qquad (18a)$$

12

$$\boldsymbol{\tau}_e = \mathbf{M}(\mathbf{q}_d)\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q}_d, \dot{\mathbf{q}}_d)\mathbf{q}_d + \mathbf{g}(\mathbf{q}_d) \qquad \text{from desired trajectory} \qquad (18\text{b})$$

Problem* 2.7.2: *Explain the disadvantages of both approaches*

1. Measurement : Noisy acceleration signal

2. Desired Trajectory : Controller dynamics is not considered.

A better approach is to use the impulse instead than directly the torques. The impulse can be estimated comparing the expected momentum caused by the commanded torques and the effective momentum of the system (obtained from the measurement of the state of the system). The generalized momentum of the manipulator is given by: $\mathbf{p} = \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}.$

Problem** 2.7.3: *Considering the manipulator dynamics in Eq.(4), derive the equation of the variation of the momentum.*

$$\begin{cases} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau \\ p = M(q)\dot{q} \end{cases}$$

momentum : $\dot{p} = M(q)\ddot{q} + \dot{M}(q)\dot{q} = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + C^{T}(q,\dot{q})\dot{q}$

$\dot{p} = \boxed{\tau_{tot}} g(q) + C^{T}(q,\dot{q})\dot{q}$

$\rightarrow \tau_{tot} = \tau + \tau_k$

Problem** 2.7.4: *Draw the scheme of a collision estimator based on the impulse.*

momentum
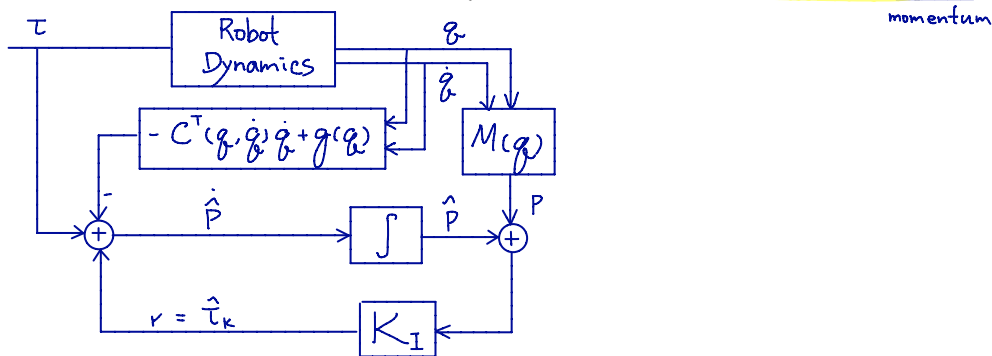
Problem** 2.7.5: *Derive the equation of the dynamics of the estimator.*

$$\hat{\tau}_k(t) = K_1 \left[p(t) - \int_0^t \underline{(\tau + \hat{\tau}_k + C(q,\dot{q})\dot{q} - g(q))} dt - p(0) \right]$$

τ_{tot}

with $\begin{cases} \hat{\tau}_k(0) = 0 \\ K_1 > 0 \text{ diagonal} \end{cases}$

13

Problem* 2.7.6: *Which reaction strategies can be adopted in case of collision?*

1. Stop the robot

2. Switch to gravity compensation $\boxed{\tau = g(q)}$

3. "reflex reaction", fast movement in collision force direction $\boxed{\tau = g(q) + K_R \hat{\tau}_K}$

4. Slowing down, or reversing the trajectory by modifying the interpolator in position control mode.

$$\boxed{\dot{\Theta}d = K_R \hat{\tau}_K}$$

3   Simulation

This experimental section provides an introduction to the simulation environment of the 3DOF planar manipulator and the controllers presented in the theory section. Therefore it is mandatory to solve the theory section first.

The simulation should be implemented in Simulink. The library "m3dof_library.slx" is given to the students. It contains the implementations of the principal matrices needed for the implementation of the algorithms (forward kinematics, Jacobian, mass matrix, Coriolis and centrifugal matrix, gravity vector). Together with these matrices is given also a visualization utility for the 3DOF planar manipulator, which facilitates the understanding of the system behavior.

MATLAB function $A_1 A_1 = A$

3.1   Kinematics and dynamics**   sqrtm(m) $\Rightarrow$ $A_1 = \underline{real} (sqrtm(A))$

to extract Real part

Task 1: *Implement the dynamics of the 3DOF manipulator described in Sec. 2.1 as a standalone block. The inputs are the commanded torques τ and the outputs are the joints angles and velocities $\mathbf{q}$, $\dot{\mathbf{q}}$.*

   Hint: Add a dissipative term for simulating the friction at joints: $\boldsymbol{\tau}_f = -\mathbf{K}_f \dot{\mathbf{q}}$

The block will be used throughout the entire simulation as the base for applying the treated control algorithms. Therefore, a correct implementation of the dynamics is required before continuing with the design of the controllers.

Task 2: *Simulate the system from the initial position $\mathbf{q}_i = [-60\,^\circ, -30\,^\circ, 20\,^\circ]$, plot the joint angles and comment the results.*
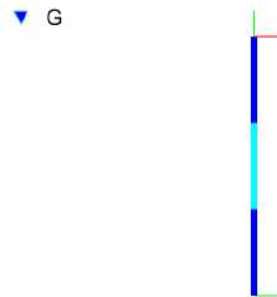
Task 3: *Implement the calculation of the Cartesian pose and velocity by means of the forward kinematics and the Jacobian matrices.*
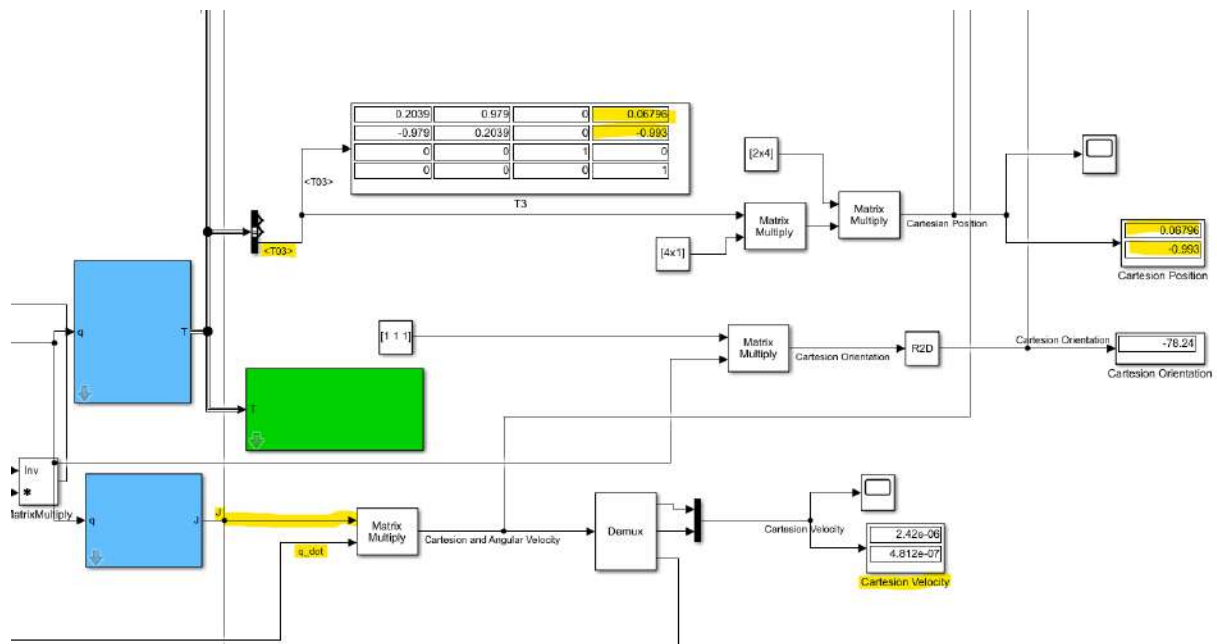
3.2   Joints control*

For the implementation of the control laws create a "Controller" block which should include all the control algorithms considered. The input of the block is the state of manipulator (joint positions and velocities, forward kinematics, jacobians, dynamics matrices, etc...) and the outputs are the commanded torques. The closed loop between the Dynamics block and the Controller block constitutes a simulation of the behavior of the controlled manipulator, as showed in Fig. 5(a)

It's highly suggested to implement all the following controllers as standalone subsystems and to sum up their torques in a unique signal (see Fig. 5(b)), thanks to the superposition principle for the impedances. The subsystems could be individually activated or deactivated during the simulation by means of the use of an "Enable" input block which can be found in the "Port & Subsystems" Simulink standard library. In this way it's possible to compare the effect of each individual functionality of the controller.

The first step of the implementation is to correctly counterbalance the gravity torques which act on the manipulator.

Task 4: *Implement a gravity compensation using the gravity torques from the online model. Start the simulation again and compare the robot behaviour.*

3.1 Kinematics and dynamics**

Task 1: Implement the dynamics of the 3DOF manipulator described in Sec. 2.1 as a standalone block. The inputs are the commanded torques τ and the outputs are the joints angles and velocities $q, \dot{q}$.

Hint: Add a dissipative term for simulating the friction at joints: $\tau_f = -K_f \dot{q}$

The block will be used throughout the entire simulation as the base for applying the treated control algorithms. Therefore, a correct implementation of the dynamics is required before continuing with the design of the controllers.

$$K_f = 5 \times I_3$$

Task 2: Simulate the system from the initial position q_i = [−60°, −30° , 20°], plot the joint angles and comment the results.

Because of gravity and the friction at joints, the 3DOF manipulator eventually points in the direction of gravity. q_i finally converges to [−90°, 0° , 0°]. ($\tau = 0$)
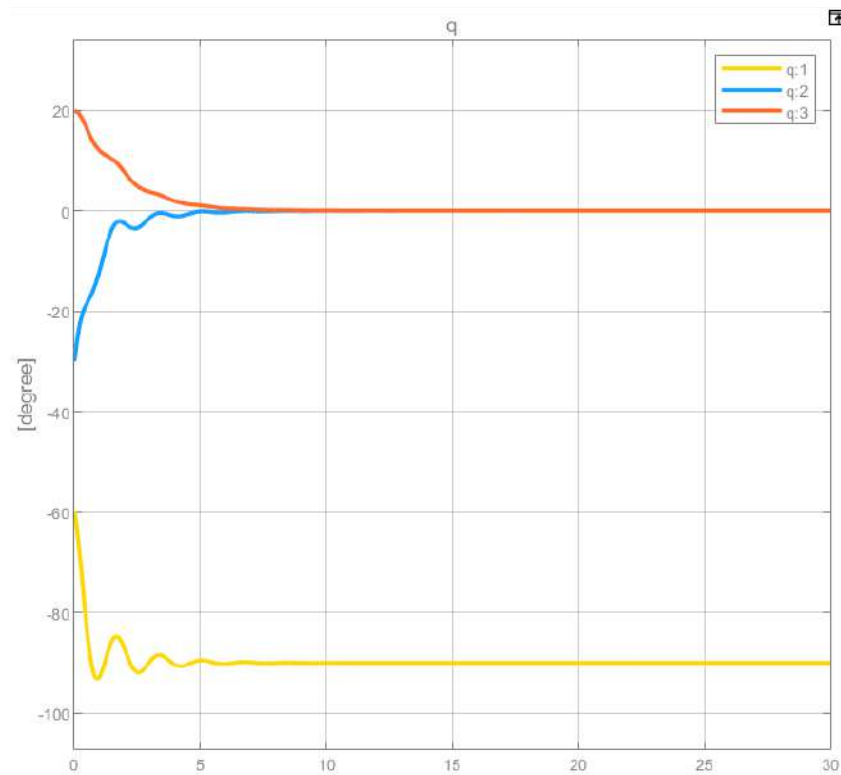
Figure 1. The change of the joint angles in 20 time steps

Figure 2. the final position

Task 3: Implement the calculation of the Cartesian pose and velocity by means of the forward kinematics and the Jacobian matrices.

Figure 3. calculation of the Cartesian pose and velocity

3.2 Joints control*

Task 4: Implement a gravity compensation using the gravity torques from the online model. Start the simulation again and compare the robot behaviour.

Because of the gravity compensation, the final position at 30th time step is as same as the initial position and the joint angles keep in the 30 time steps.
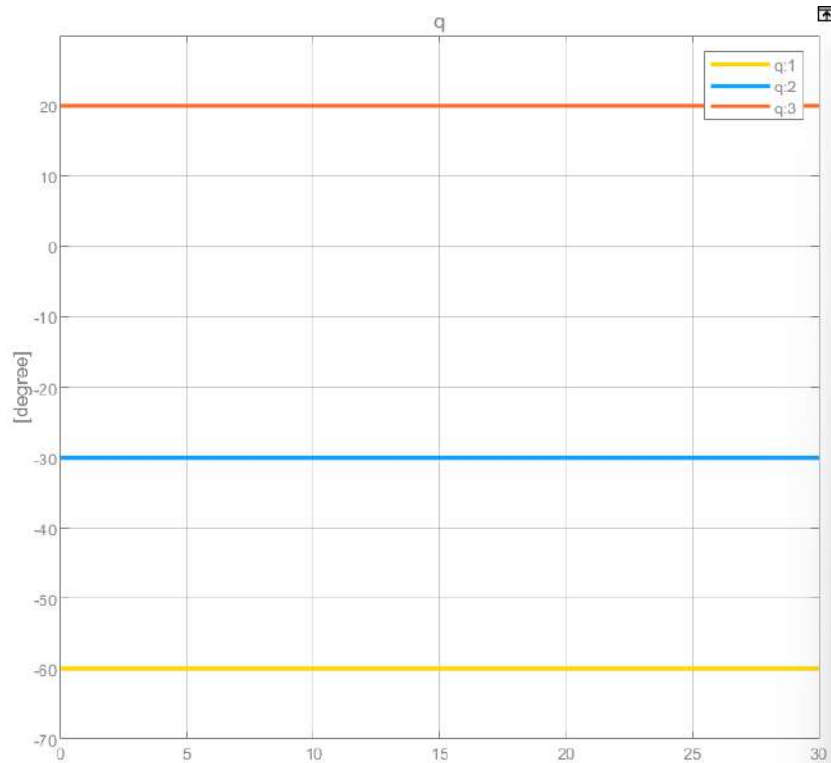
Figure 4. The change of the joint angles in 30 time steps
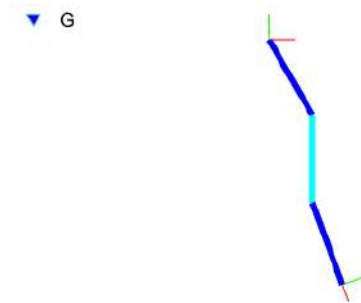
Figure 5. the final position

The gravity compensation is easy to implement in theory but is very important in practice. Keep the gravity compensation on throughout the tutorial.

Task 5: Implement the joints PD controller described in (2.2) with a costant damping. Tune the stiffness and damping matrices K_P, K_d so as to achieve a fast and well damped response.

$K_P = 1 \times I_3$ and $K_d = 1 \times I_3$

Task 6: Run the simulation again with the gravity compensation and the PD controller, what happens if the gravity compensation is deactivated?

If the gravity compensation is deactivated, the robot cannot reach the desired position.

Task 7: Plot the joints angles and torques with respect to time for some representative cases of K_P and K_d. In order to achieve a desired damping factor ζ, the damping matrix K_d should be appropriately designed.
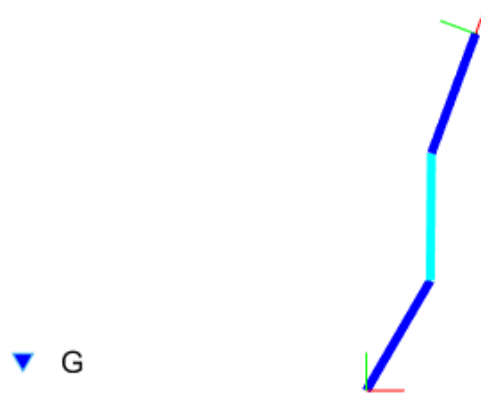
Figure 6. the robot reaches the desired position
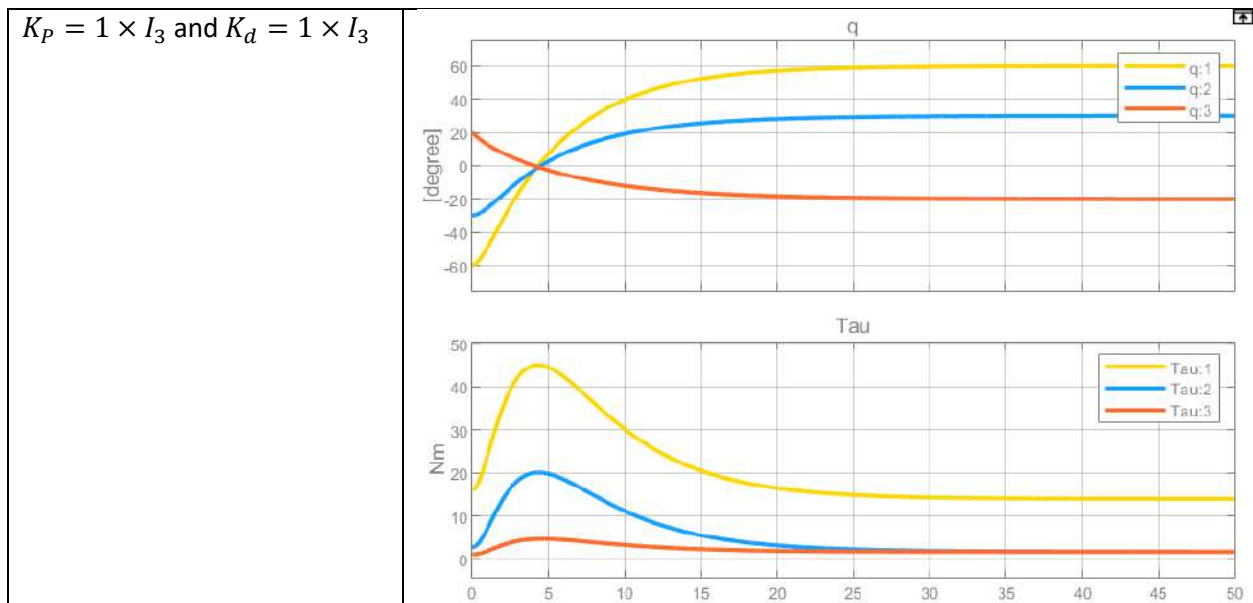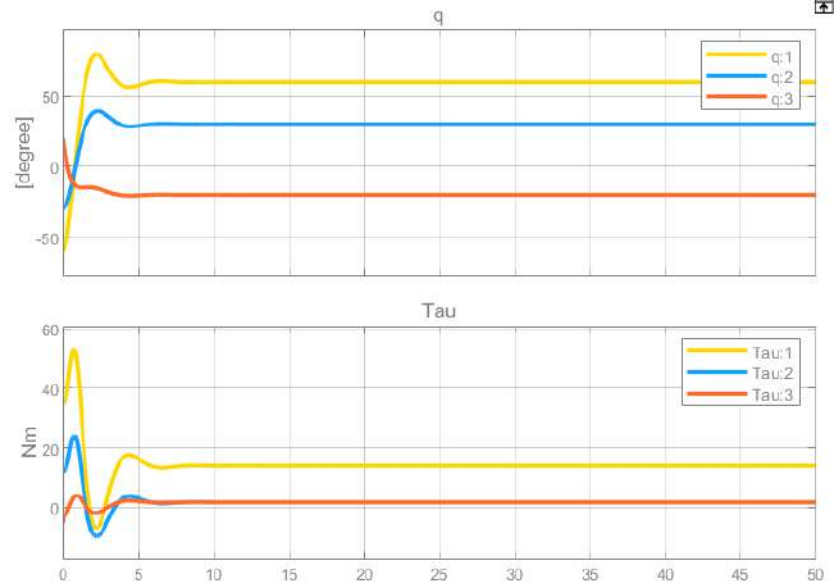
$K_P = 10 \times I_3$ and $K_d = 1 \times I_3$	
$K_P = 1 \times I_3$ and $K_d = 10 \times I_3$	
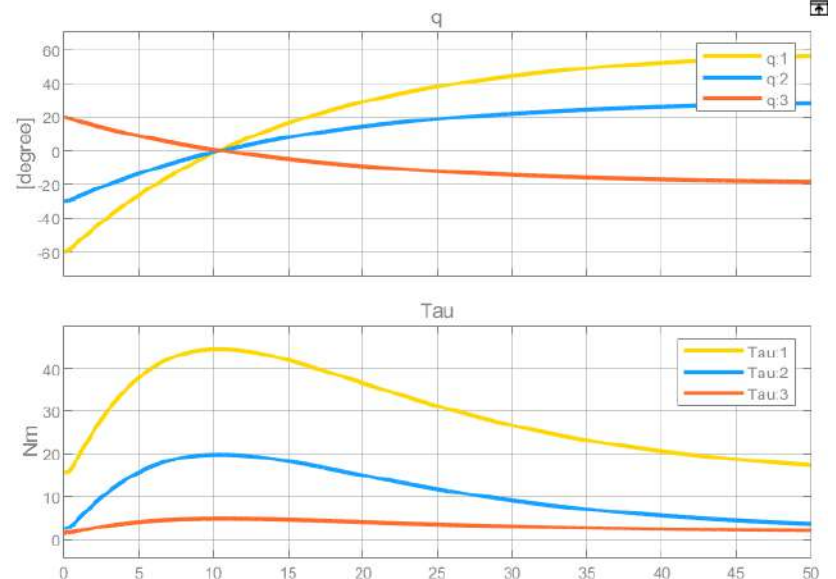
Task 8: Implement the damping design with the method of the square root matrices.
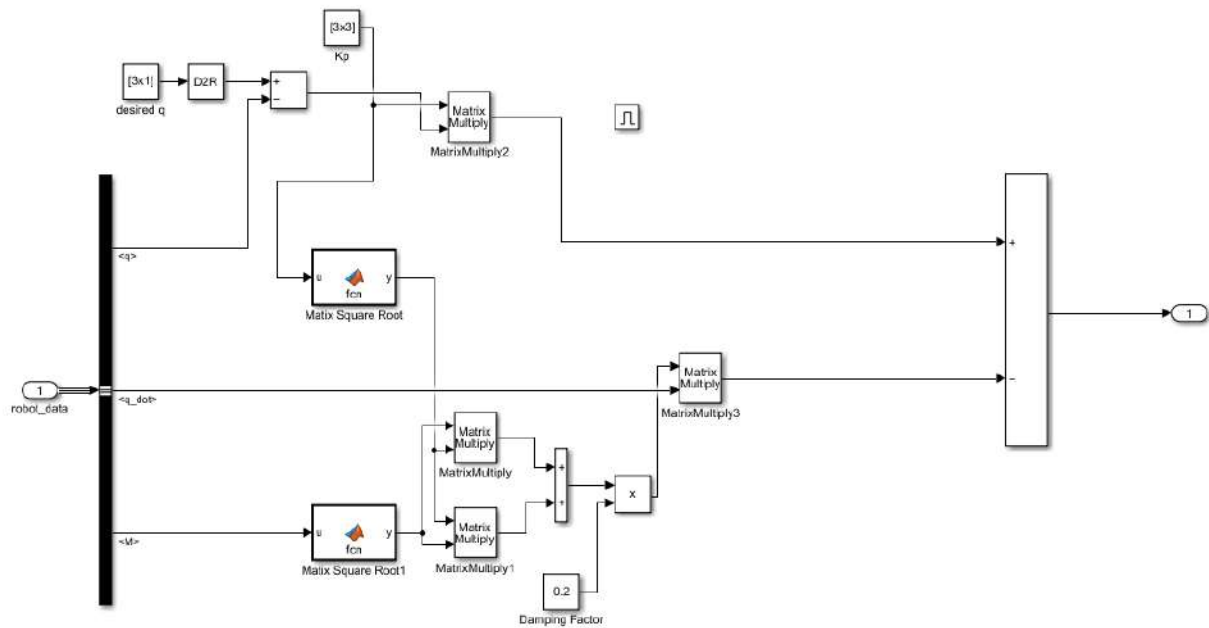
Figure 7. damping design with the method of the square root matrices

Task 9: Plot and compare the temporal response of the joint angles for three different damping factors ζ.
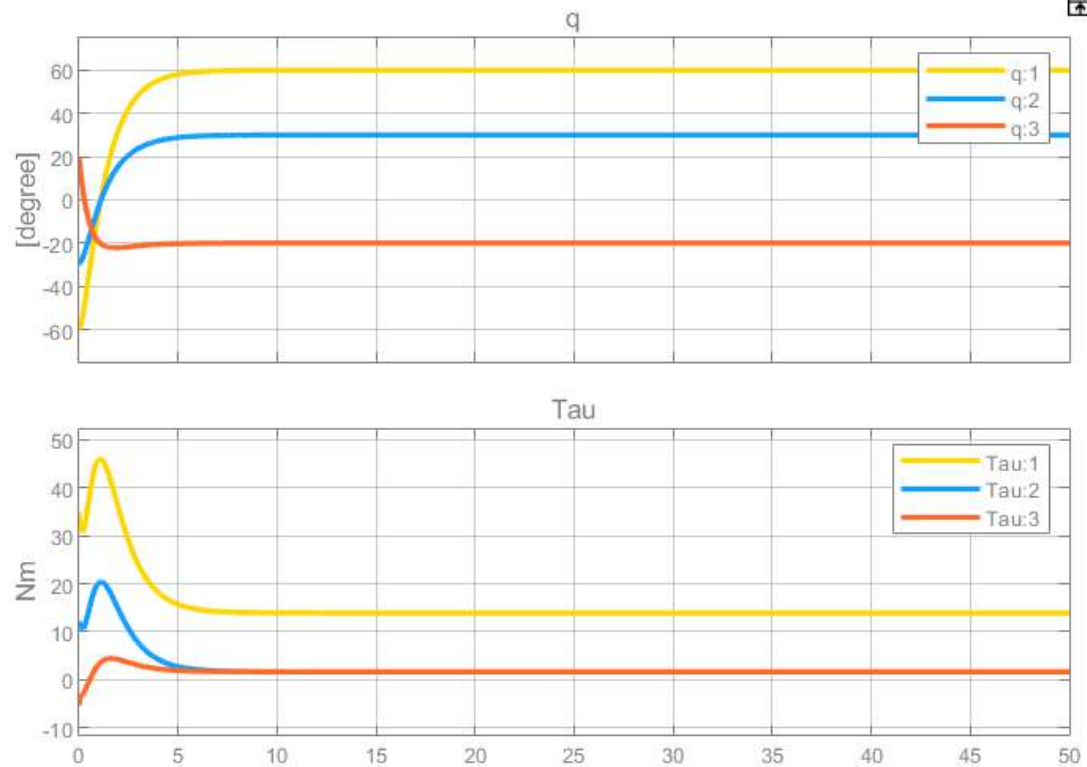
$$K_P = 1 \times I_3$$

| ζ = 0.2 | |
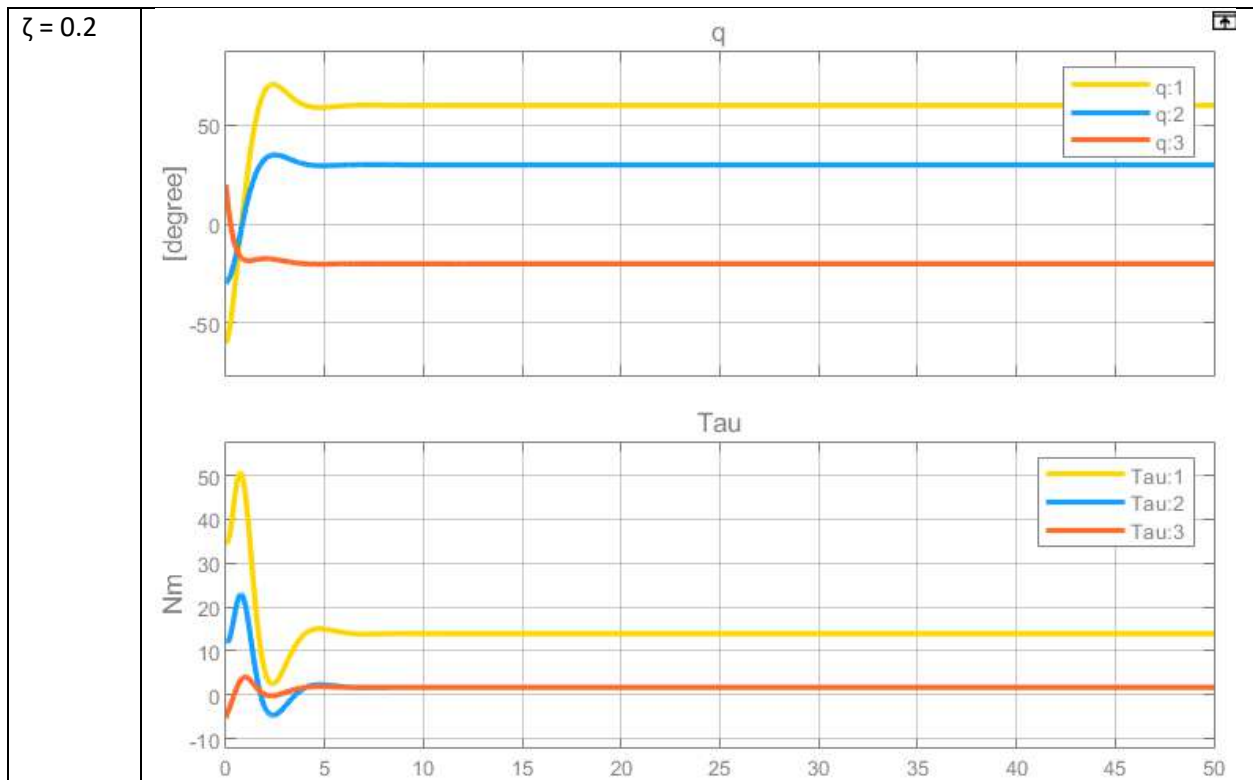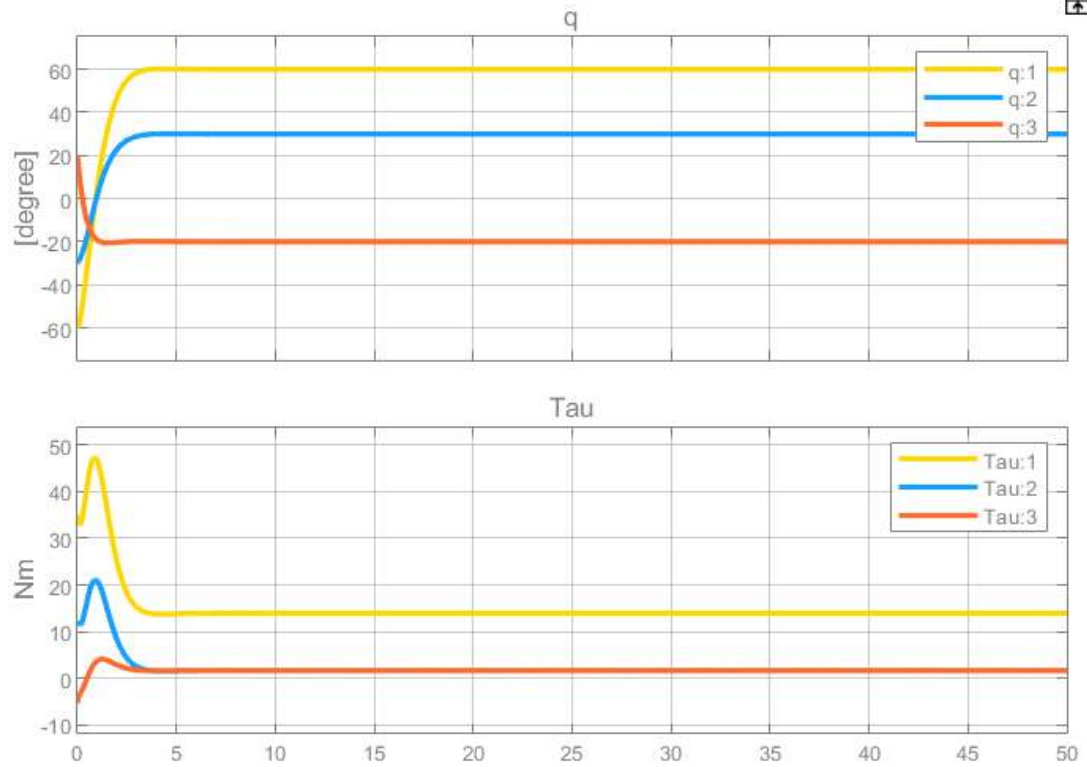
ζ = 0.5	
ζ = 0.8	

When the damping factor is too small, the overshoot happens. When the damping factor is too large, the settling time will take longer.

3.4 Translational Cartesian Impedance control*

Task 13: Implement the translational Cartesian controller derived in the problem 2.4.4 and tune the stiffness and damping matrices. After trying various desired positions, command the position [x, y] = [0, 1.5]. What happens in this case?
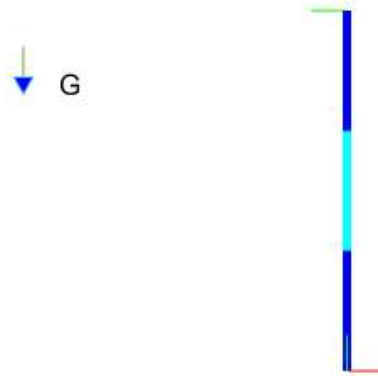
Figure 11. The final position with the desired Cartesian position [x, y] = [0, 1.5]

Because [x, y] = [0, 1.5] is the point that the robot cannot reach, the robot appears as a straight line.

Task 14: Plot the position of the TCP with respect to time for some representative cases of K_p and K_d.
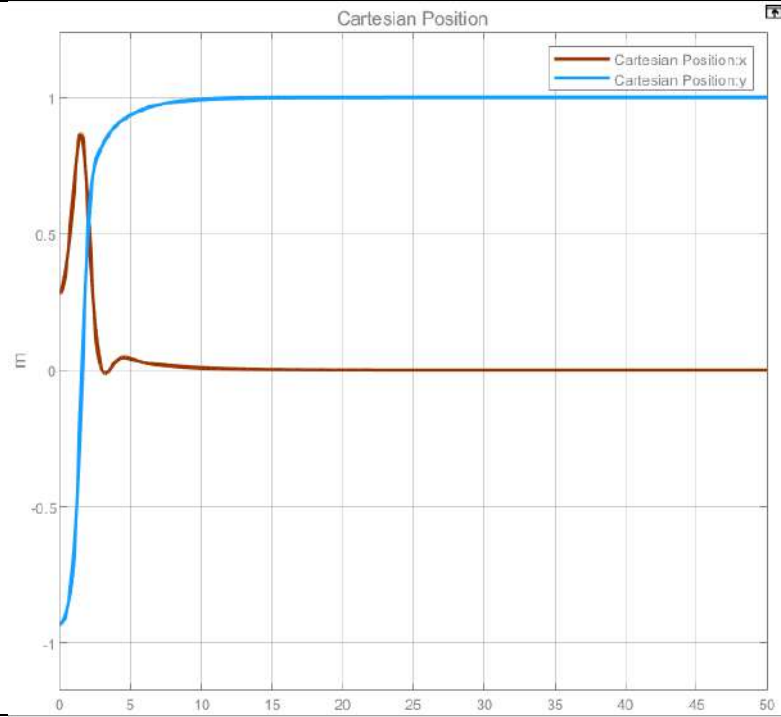
$K_p = 10 \times I_2$ and $K_d = 5 \times I_2$	
$K_p = 10 \times I_2$ and $K_d = 10 \times I_2$	
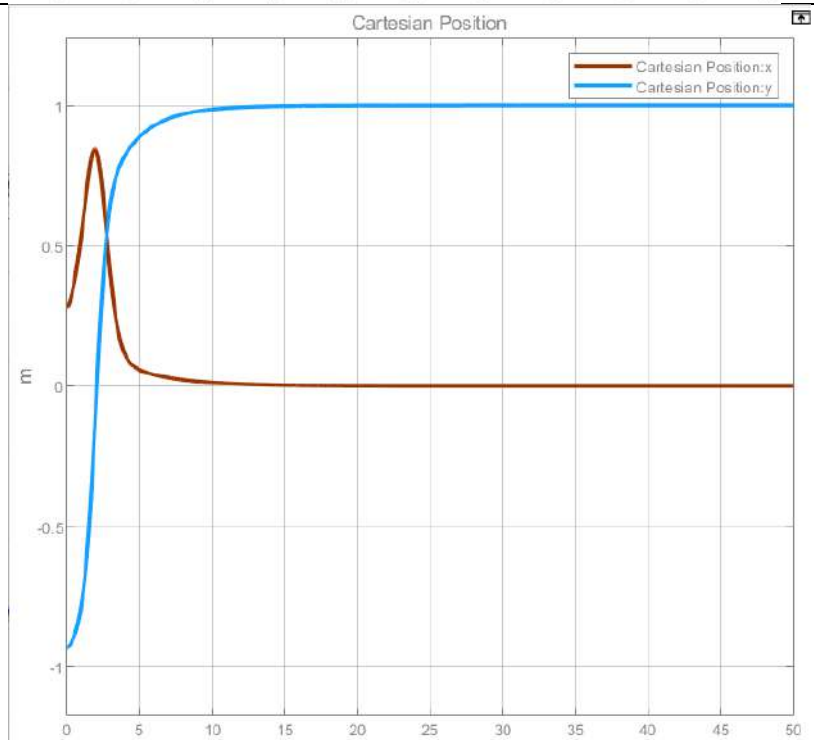
$K_p = 5 \times I_2$ and $K_d = 5 \times I_2$	
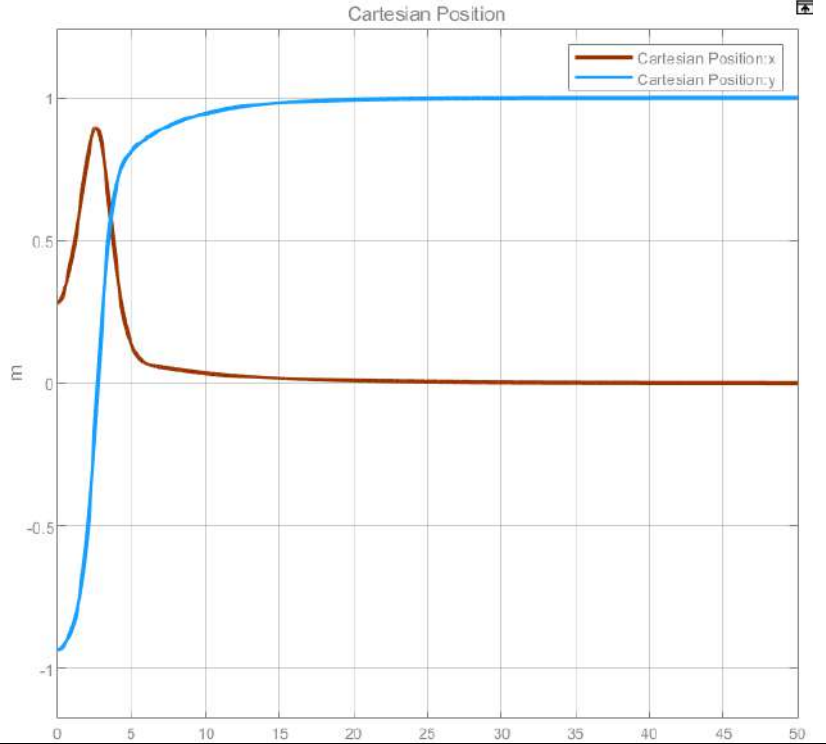
3.6 Full Cartesian Impedance controller***

Task 18: Calculate the rotational error matrix and implement an Embedded Matlab function for transforming it into the quaternion representation.

Task 19: Implement the full Cartesian controller derived in Problem 2.6.8.

Task 20: Implement the damping design for the full Cartesian controller.

Task 21: Plot the position of the TCP with respect to time for three different damping factors.

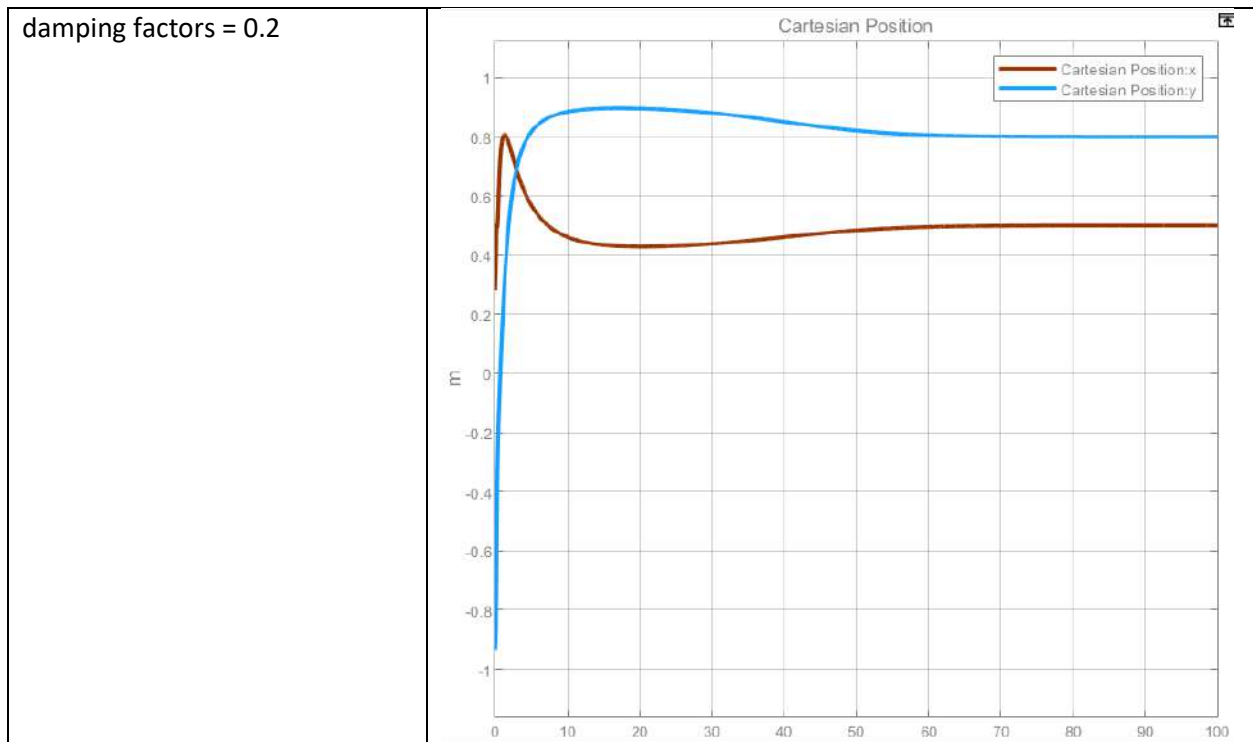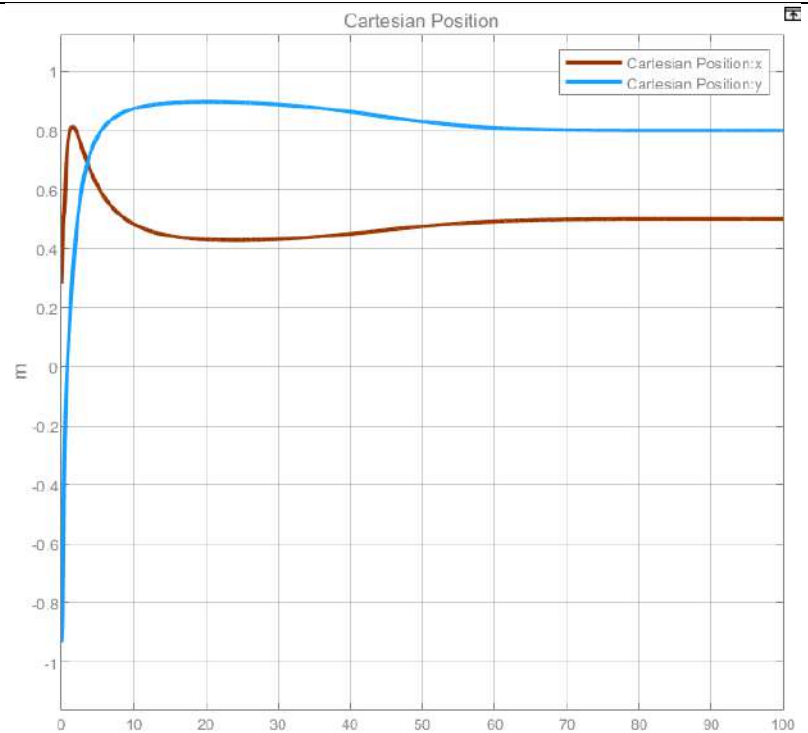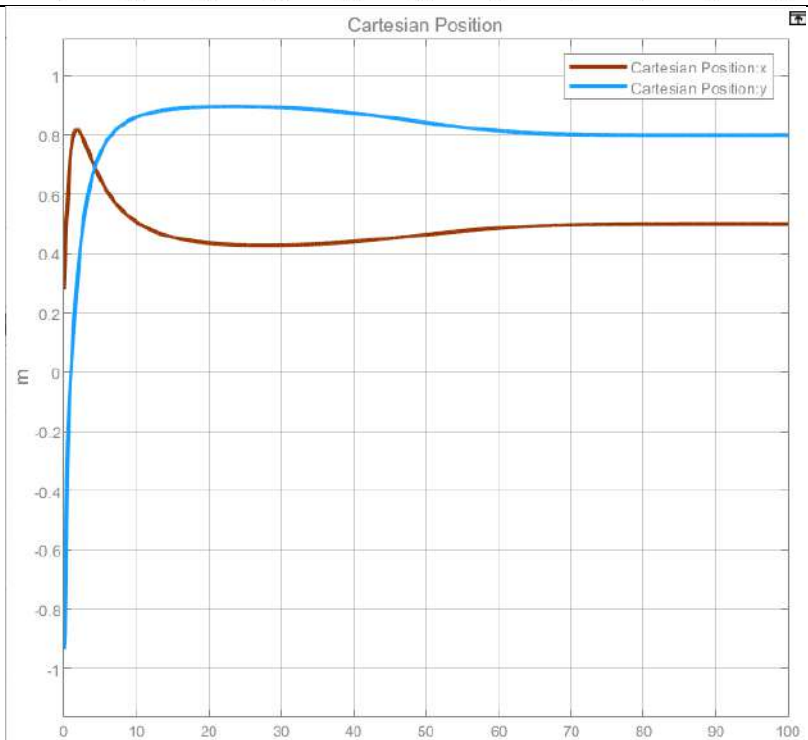$$K_p = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 40 \end{bmatrix}$$
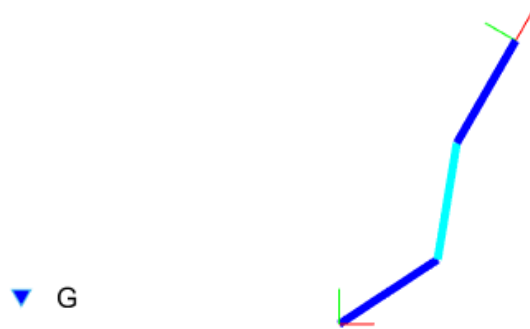
Figure 15. Desired position $[x, y] = [0.5, 0.8]$ and desired orientation $w = 60°$

damping factors = 0.2	

damping factors = 0.5	
damping factors = 0.8	

3.7 Collision detection**

with Full Cartesian Impedance controller and

$$K_p = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 40 \end{bmatrix}$$

Task 22: Implement the external torque observer derived in Problem 2.7.4.

Task 23: Make some representative plots that the observer is correctly detecting the impacts.
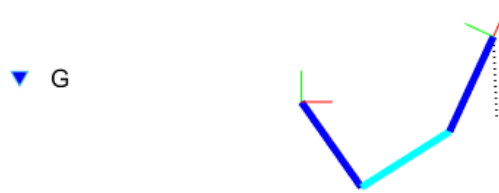
Figure 19. Position of the wall [0.6 0] and Orientation of the wall [-1 0]

Task 24: Add source of noise on the measurements of the joints angles and velocities $q, \dot{q}$. Design the gain K_i so as to obtain a fast response of the estimator and reject the noisy components of the measurements.

$$K_i = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}; \text{ noisy components(white noise): Noise power 0.0001}$$
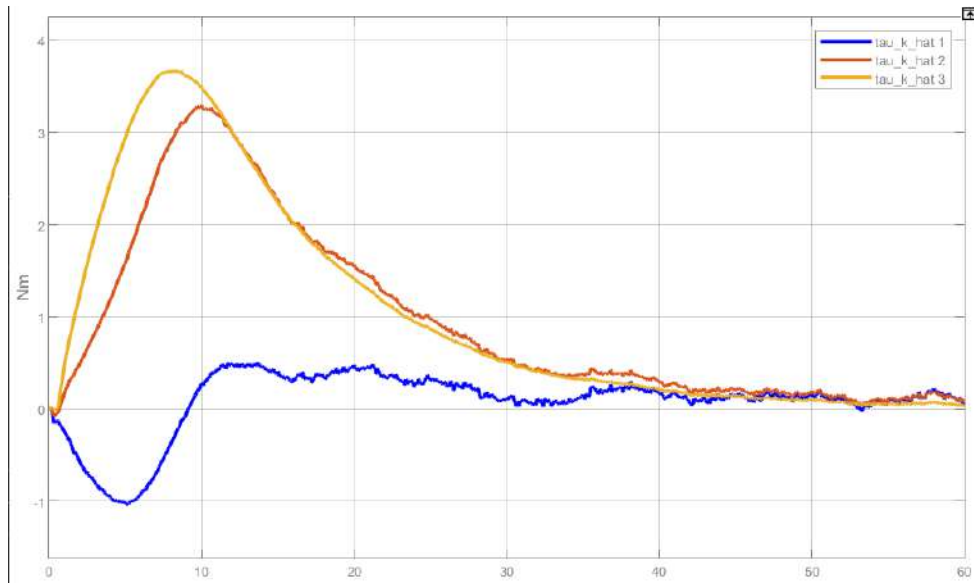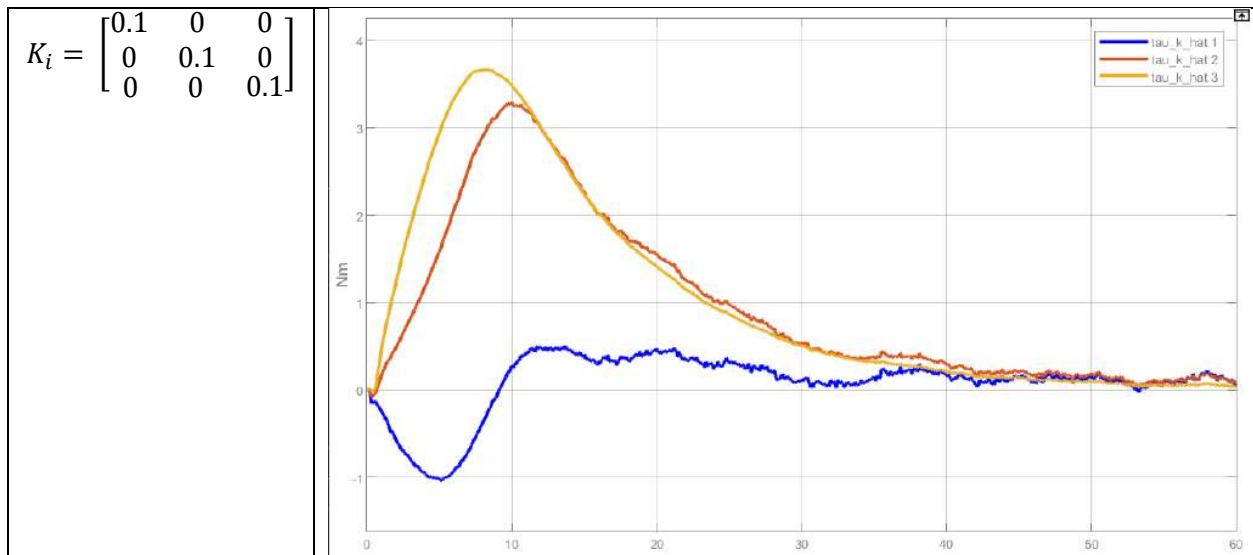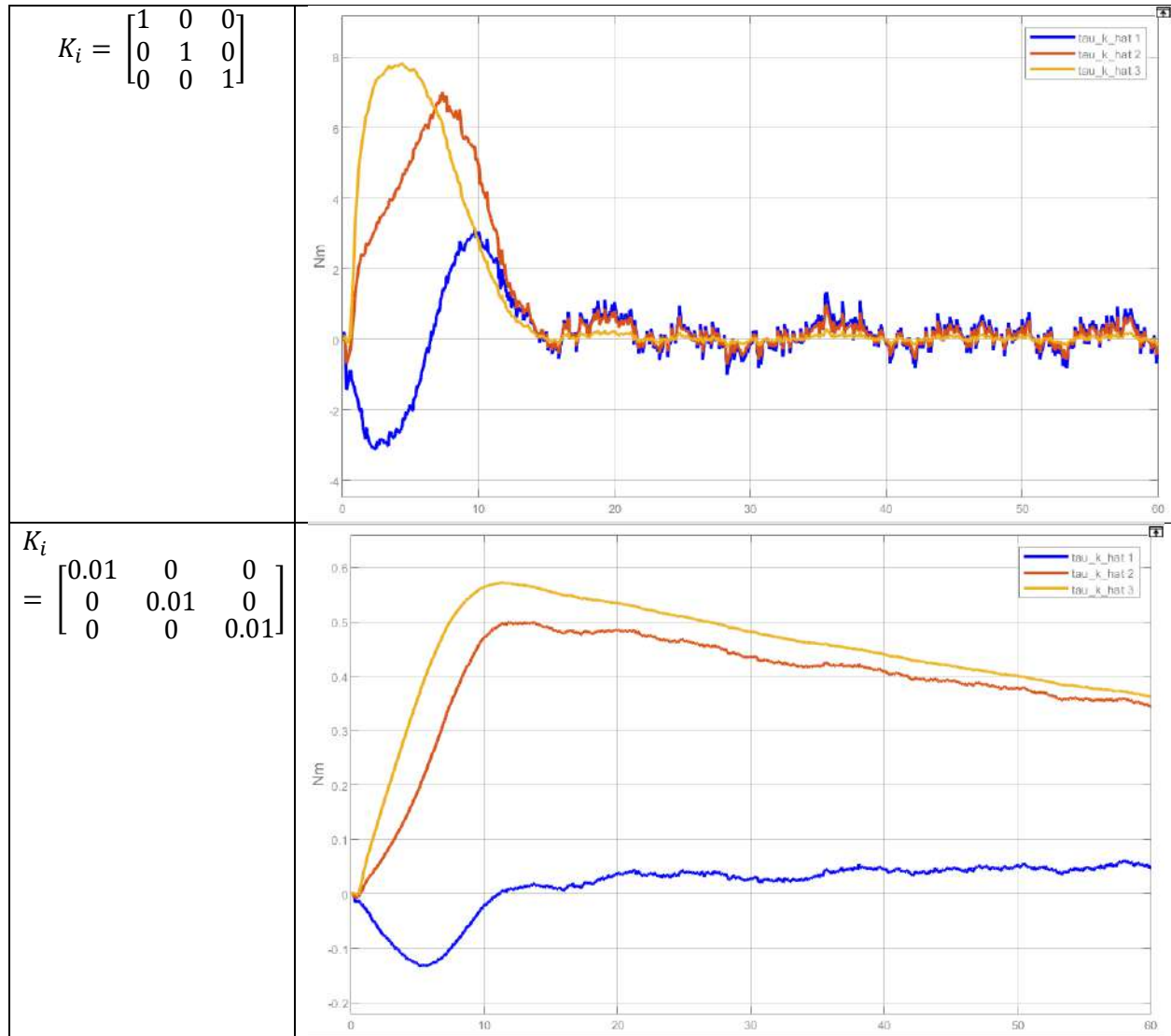
Figure20. Tau_k_hat

Task 25: Plot the behavior of the observer for different cases of K_I and comment the differences.

$$K_i = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

$K_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
$K_i = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$	

The $K_i = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$ is relatively a better choice. If the Ki is too large, the noise on Tau_k_hat can be clearly observed when the robot hit the wall. And if the Ki is too small, the Tau_k_hat value is still relatively to large after the robot has left the wall.