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°]. ($\tau = 0$)

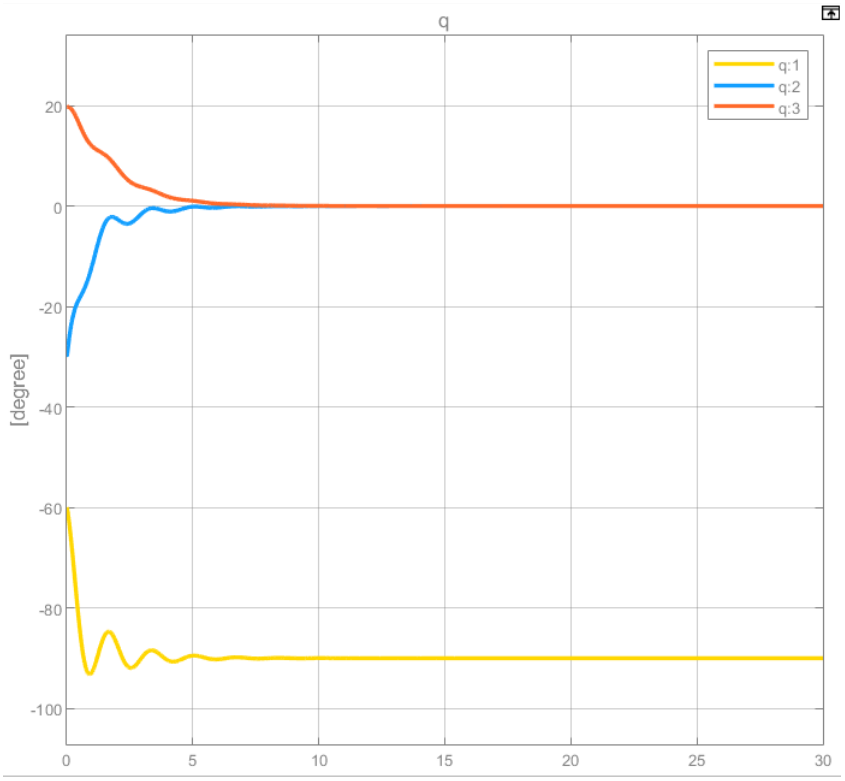


Figure 1. The change of the joint angles in 20 time steps

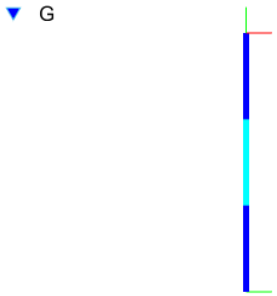


Figure 2. the final position

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

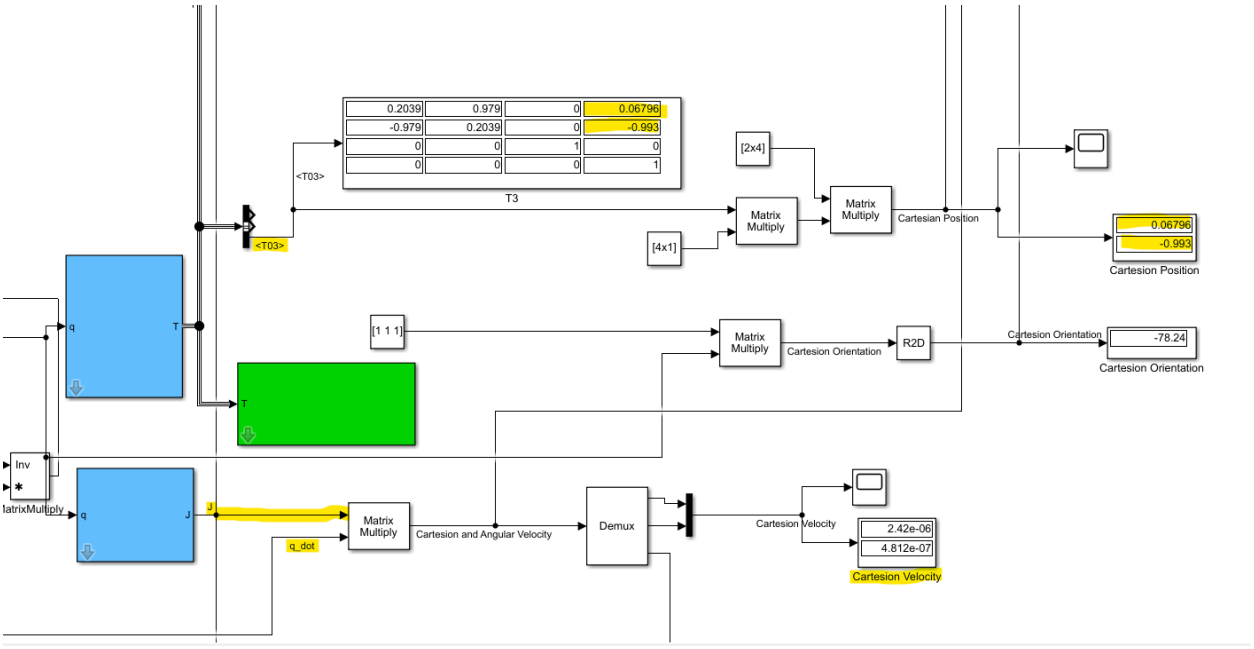


Figure 3. calculation of the Cartesian pose and velocity

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.

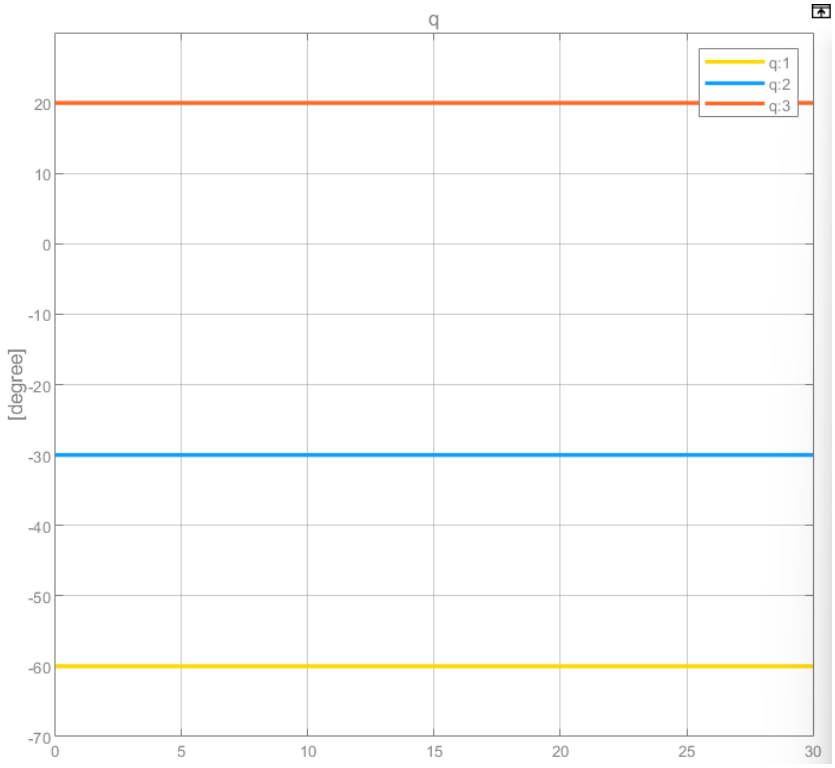


Figure 4. The change of the joint angles in 30 time steps

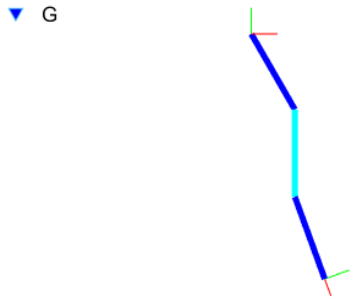


Figure 5. the final position

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.

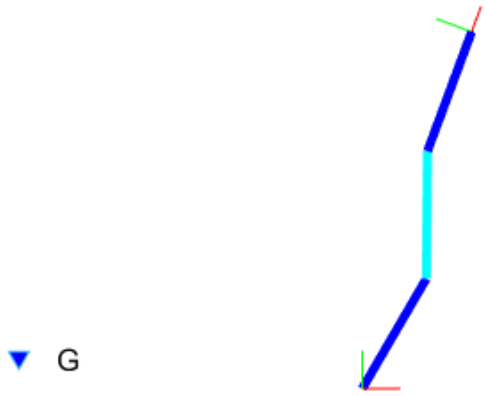
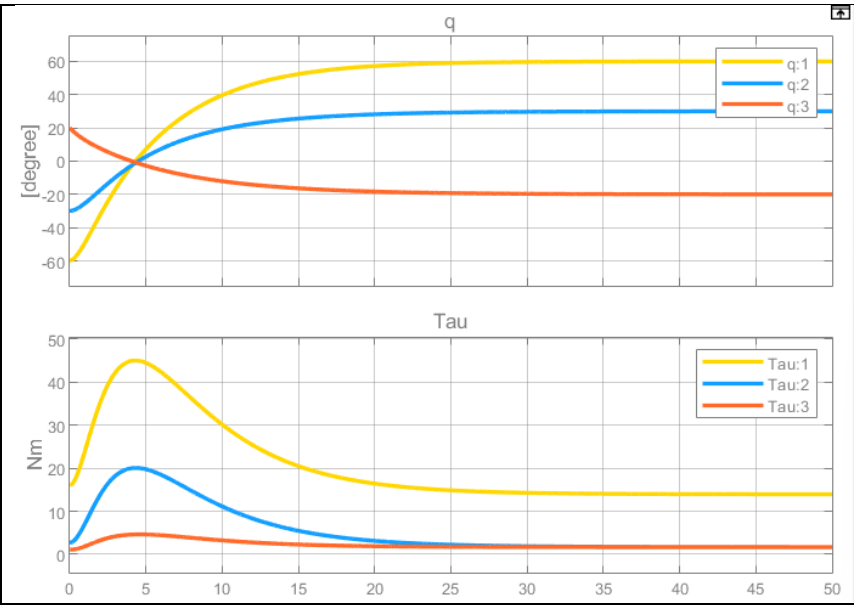
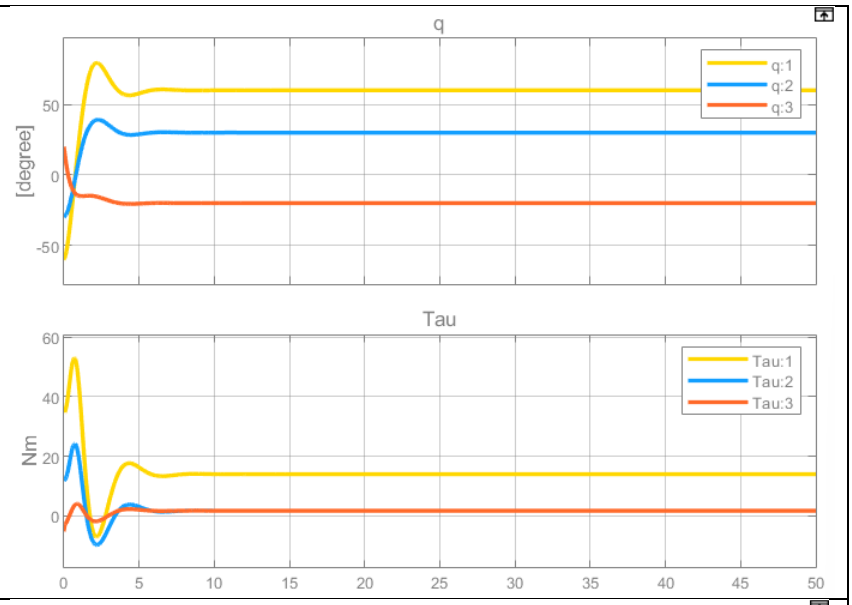
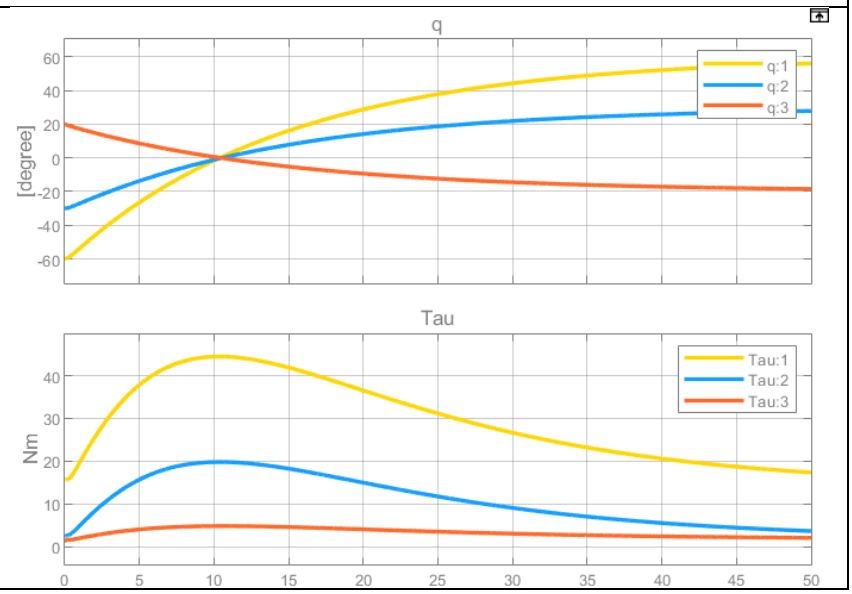


Figure 6. the robot reaches the desired position

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

$K_P = 10 \times I_3$ 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.

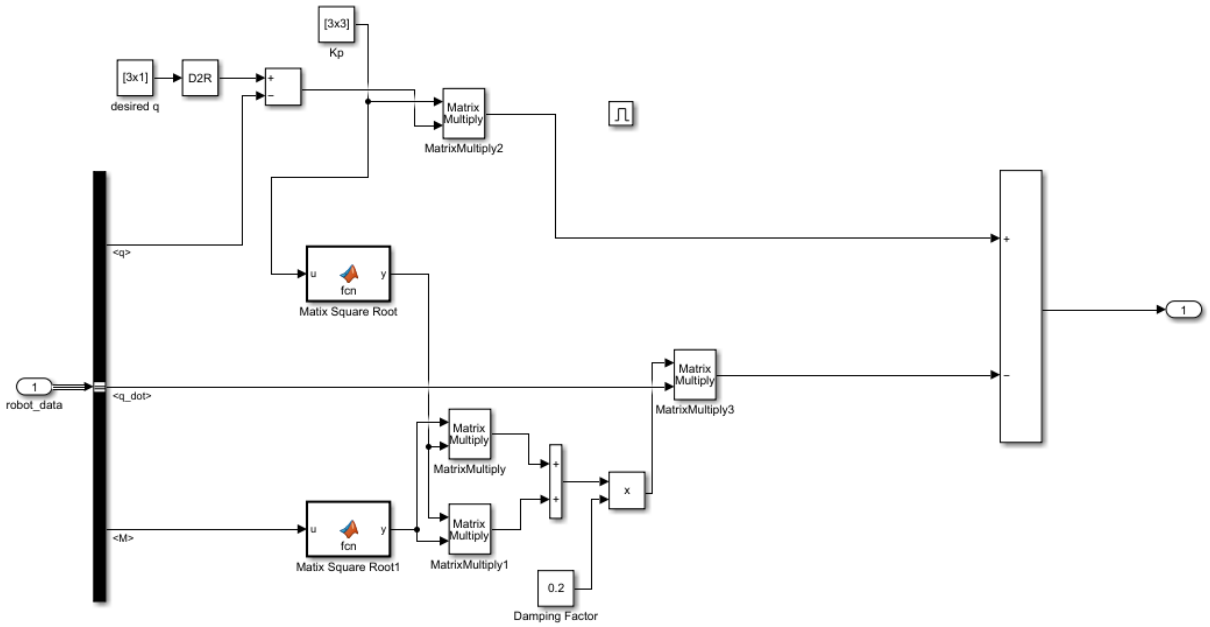
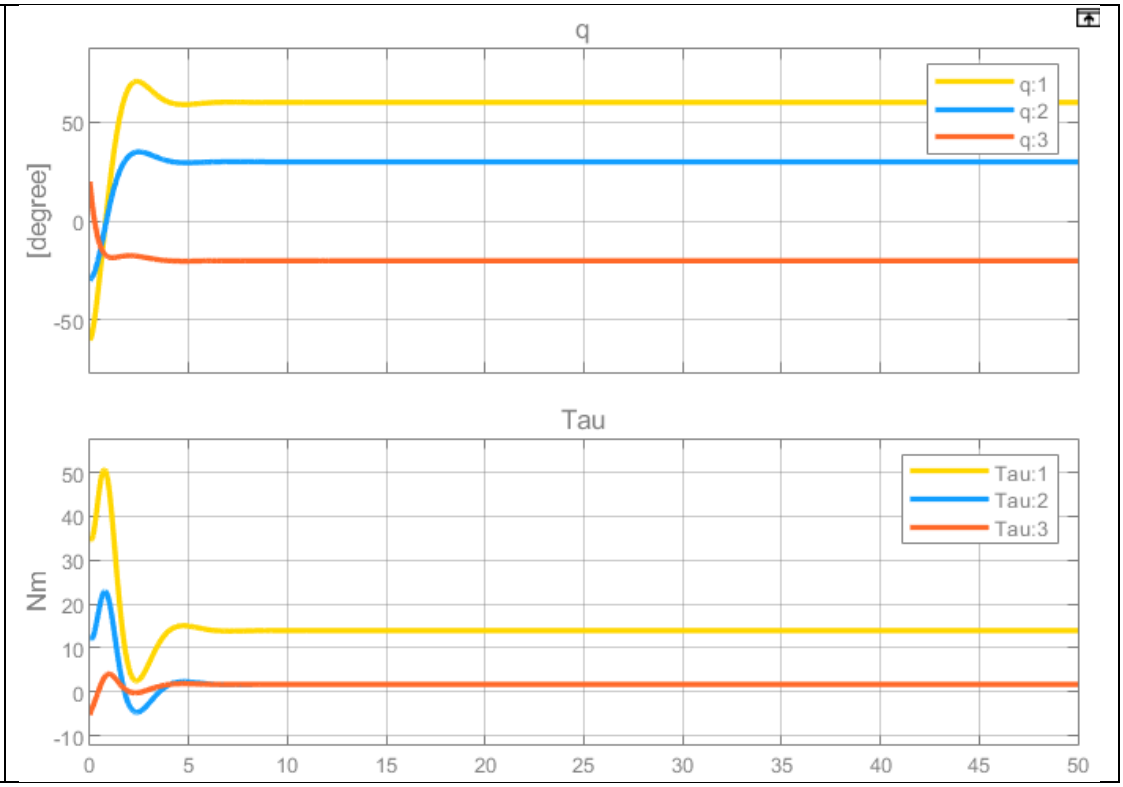
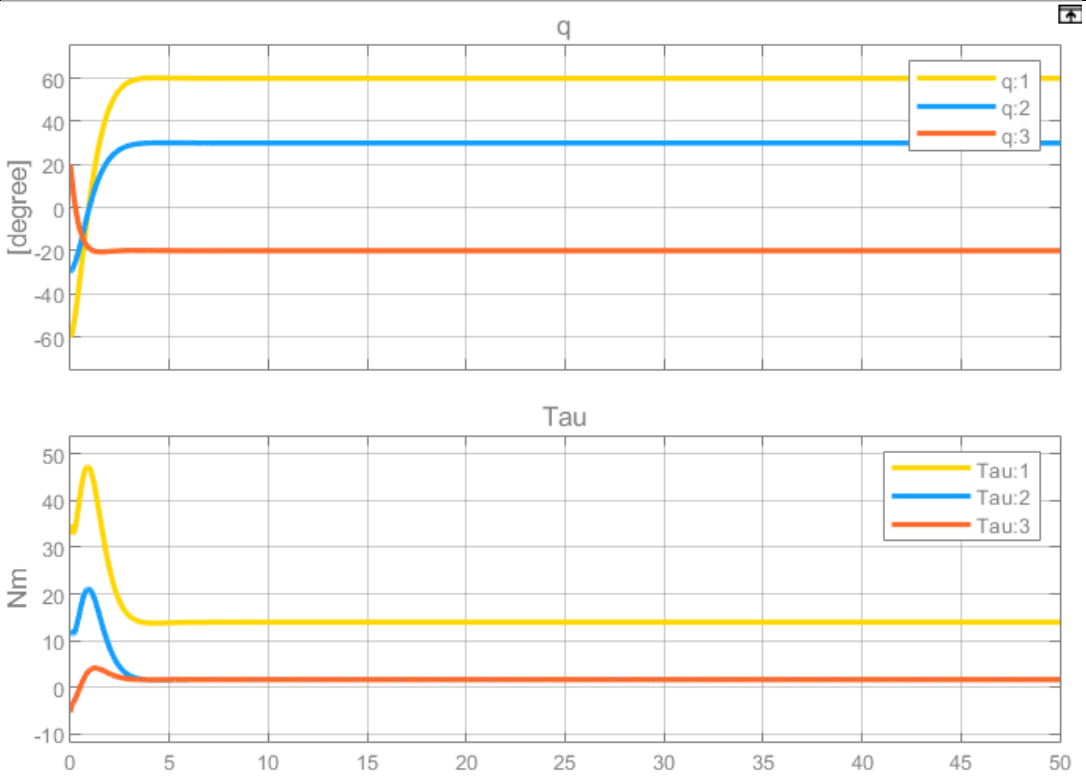
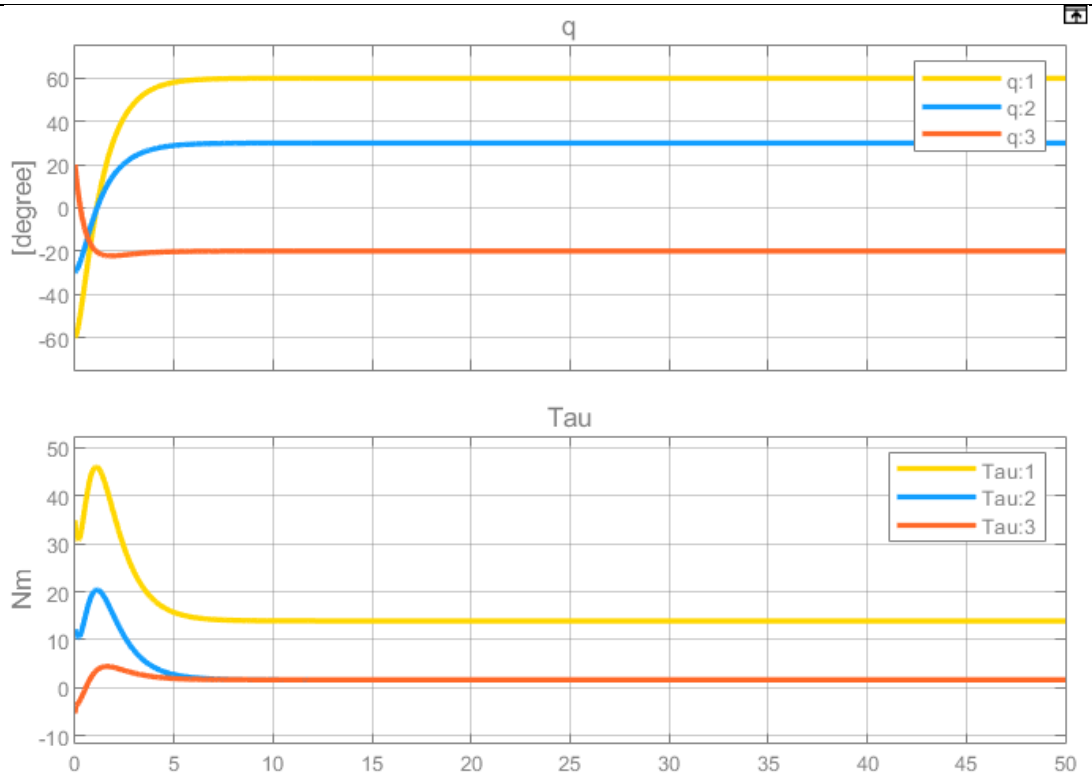


Figure 7. damping design with the method of the square root matrices

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?

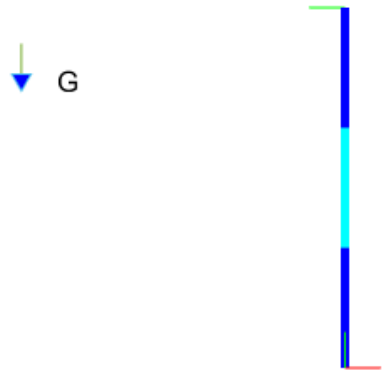
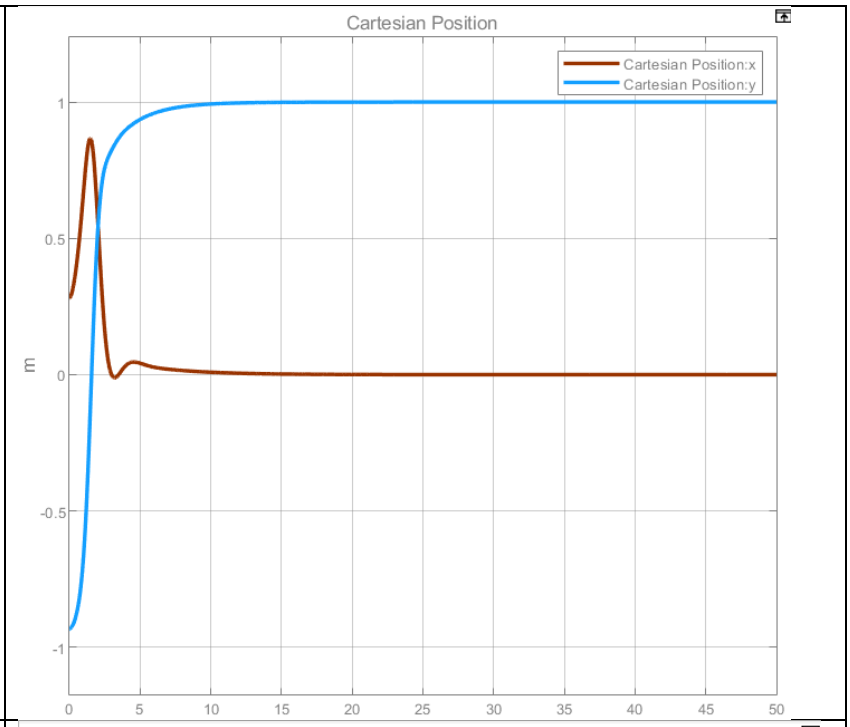
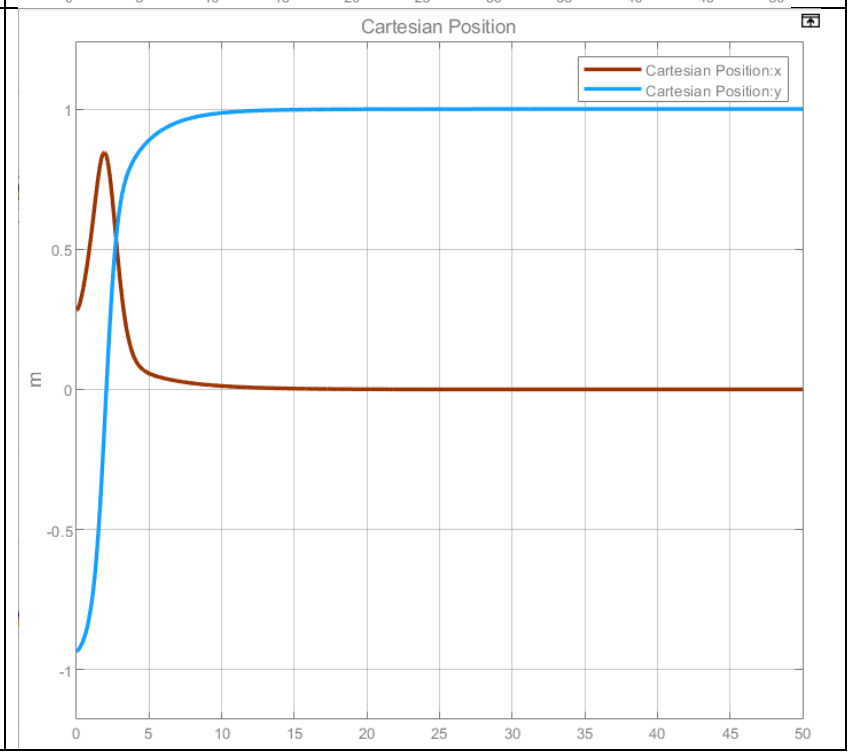
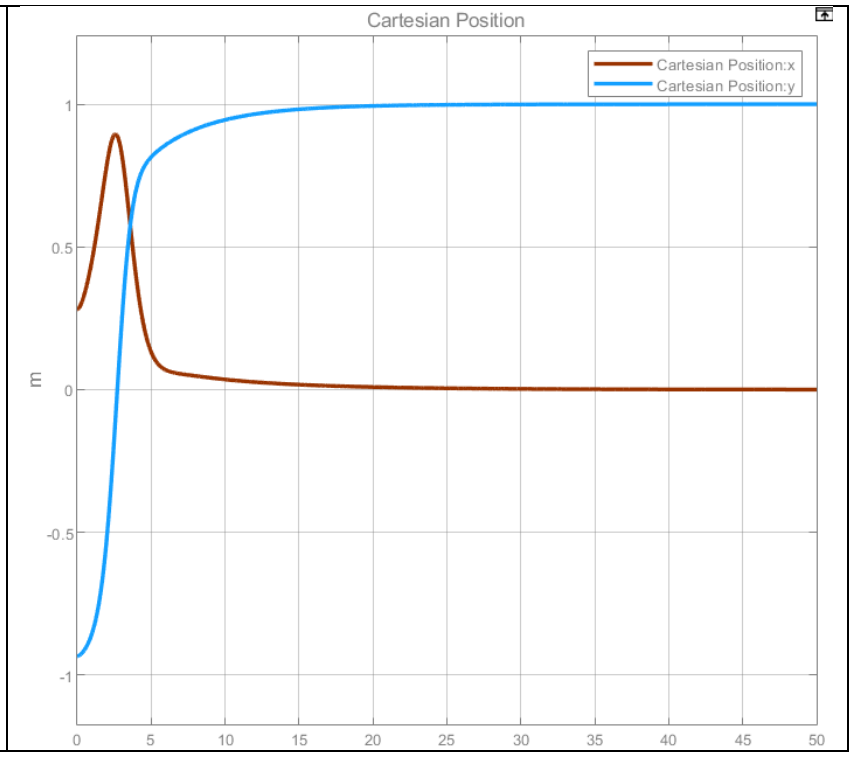


Figure 11. The final position with the desired Cartesian position [x, y] = [0, 1.5]

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.

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

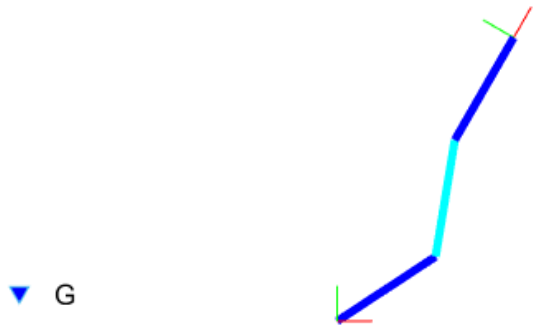
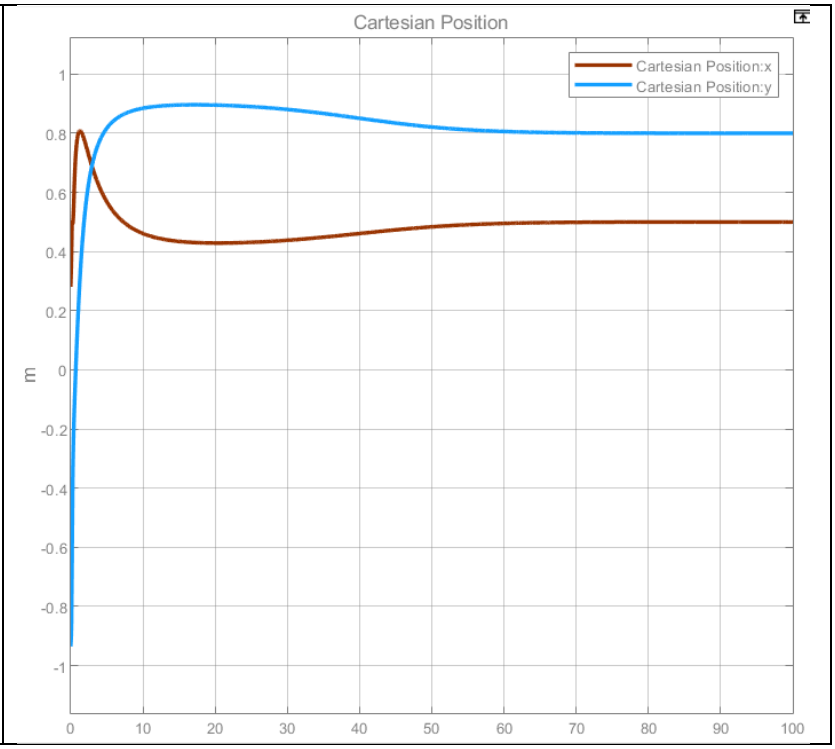
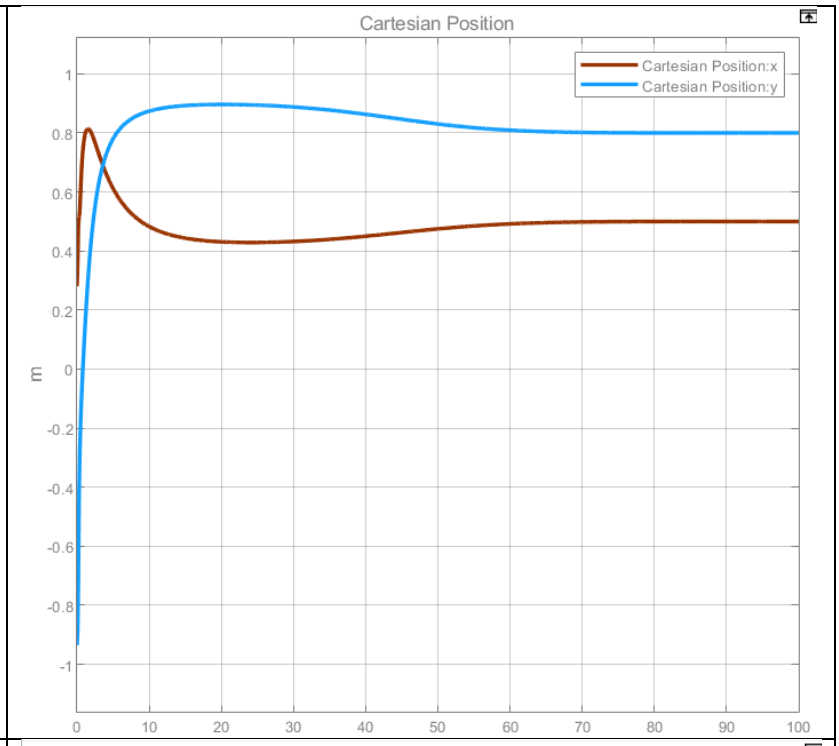
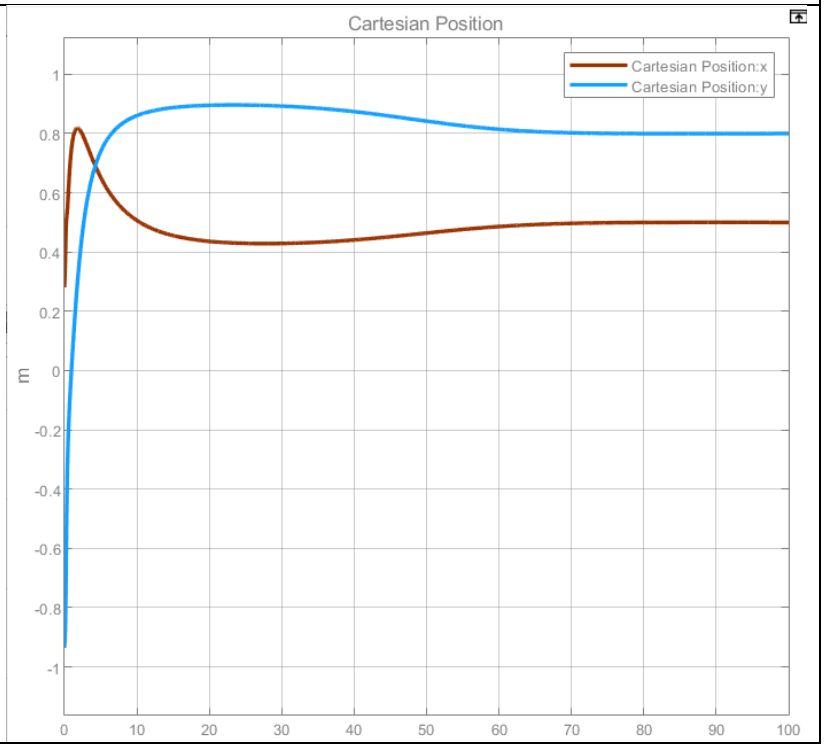


Figure 15. Desired position $[x, y] = [0.5, 0.8]$ and desired orientation $w = 60^{\circ}$

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.

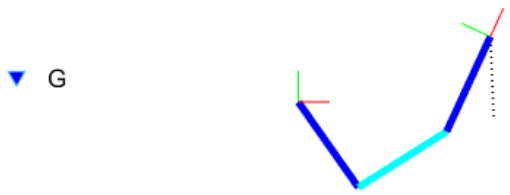


Figure 19. Position of the wall [0.6 0] and Orientation of the wall [-1 0]

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.

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

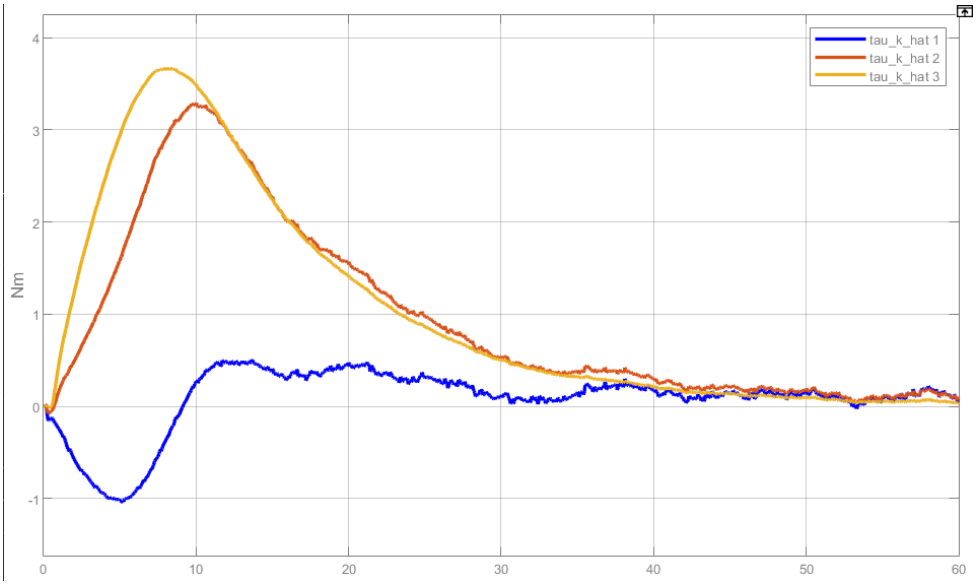
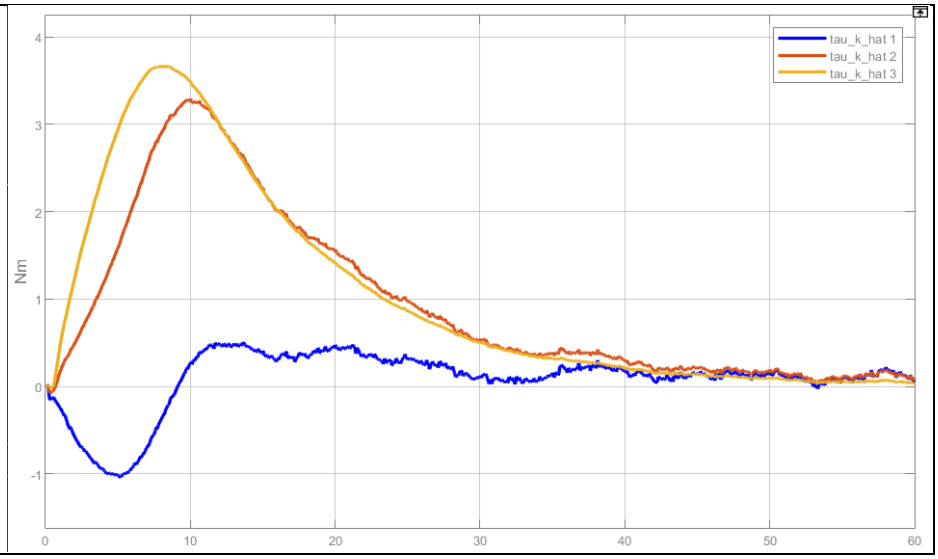
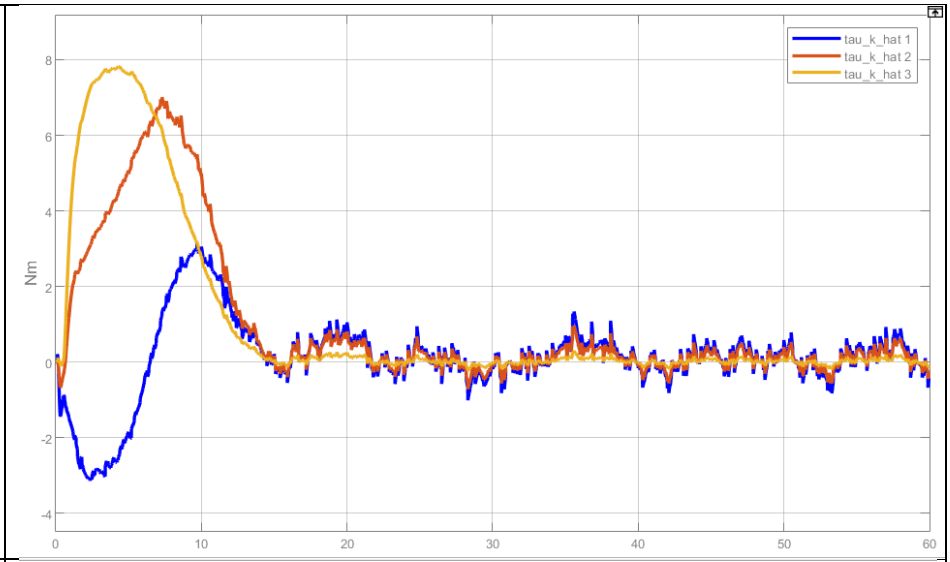
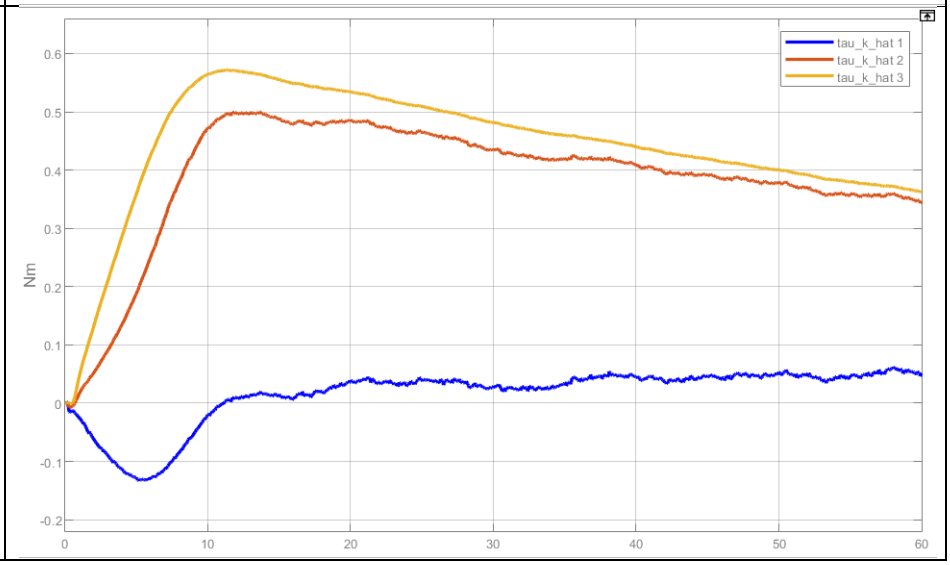


Figure20. Tau_k_hat

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.