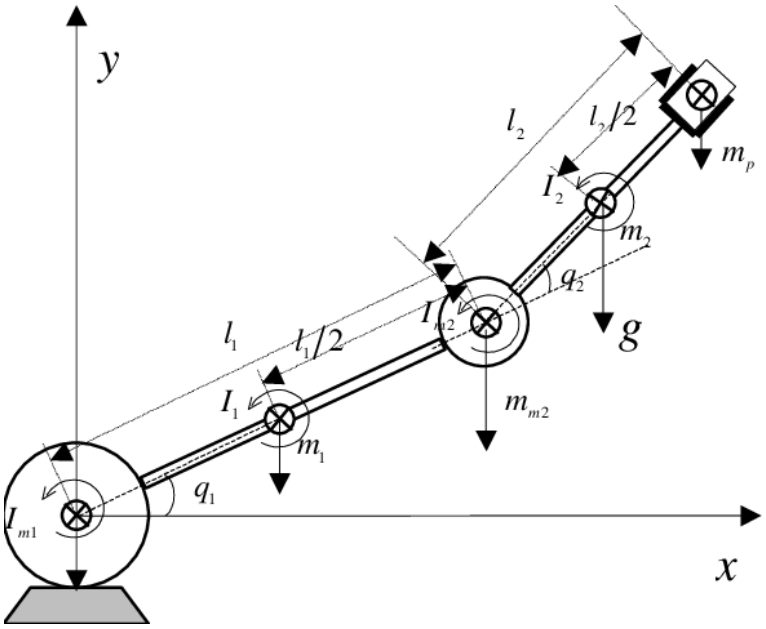
Politecnico di Bari

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE

Robotics - Industrial Handling

"Implementation and comparison of centralized control algorithms for two-arm planar industrial manipulators, in MATLAB/Simulink environment"



Submitted by: Vatinno Simona

Supervisor: Prof. Eng. Paolo Lino

Master Degree in Automation Engineering
Curriculum "Cyber-Physical Systems", A.Y. 24/25

Contents

1 Introduction

The objective of this year's topic is the implementation of **centralized** control strategies in the **joint space**, in a MATLAB/Simulink environment, for industrial manipulators.
The strategies considered are listed below:

- model predictive control;
- robust control;
- adaptive control.

Given the references for the joint variables, as well as their first and second derivatives, we want to control the motion of the system so that it follows the *setpoints* and meets the required specifications.
This involves determining the generalized forces, which also include torques, that the actuators must apply to the joints in order to ensure that the robot behaves as desired. This type of control is cheaper to implement than that in the operating space, since the sensors needed to close the control loop (e.g., rotary and linear encoders) are less expensive than those needed to detect the position of the end effector. However, joint space control requires the kinematic inversion of the x_d trajectory upstream of the loop and does not allow direct control of the position and orientation of the end-effector.
Consequently, any uncertainties in the model or errors in processing result in a loss of accuracy and deficiencies in final performance. On the other hand, however, control in operational space is generally more complex because it requires the calculation of the Jacobian and must include a method for managing redundancies.

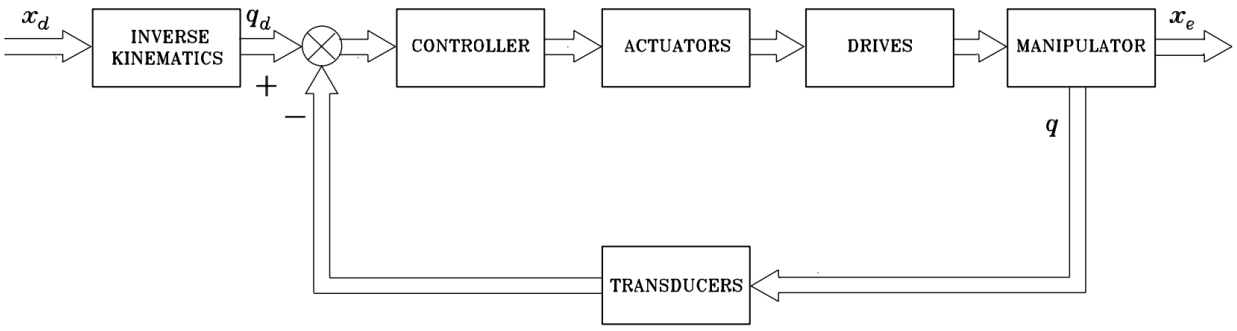


Figure 1.1: Control loop in joint space

The proposed family of strategies is called centralized control, since each joint is not adjusted independently of the others, but they are considered jointly, also evaluating the dynamic effects due to mutual interactions.

Finally, the controllers were compared with a series of tests to evaluate their strengths and weaknesses.

2 Manipulator model

The **two-armed planar** manipulator, equipped with two rotoidal joints ($n = 2$), is described by a generalized coordinate vector $\mathrm{q} = [\theta_1\ \theta_2]^T$, which defines its position.

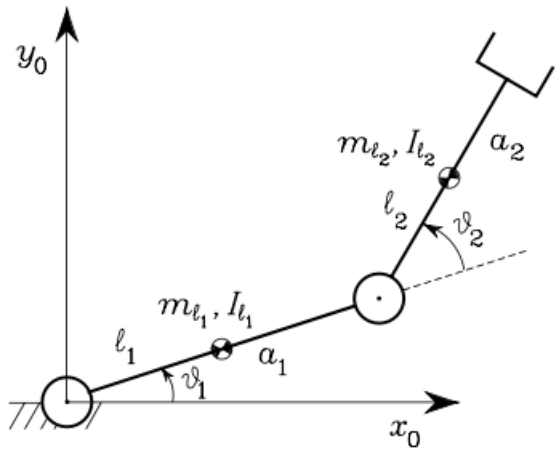


Figure 2.1: Two-arm planar manipulator

In this year's topic, the following manipulator was considered:

	a_i	l_i	m_{li}	I_{li}	k_{ri}	m_{mi}	I_{mi}
arm no. 1	1m	0.5m	50kg	10 $kg \cdot m^2$	100	5kg	0.01 $kg \cdot m^2$
arm no. 2	1m	0.5m	50kg	$10kg \cdot m^2$	100	5kg	0.01 $kg \cdot m^2$

Table 2.1: Manipulator parameters

Where, assuming that the motors are positioned so that the axis of their joint has its center of mass at the origin of the respective reference system, we have:

- a_i is the length of the arm [m];
- l_i is the distance from the center of mass of the i-th arm to joint i [m];
- m_{li} is the mass of the i-th arm [kg];
- I_{li} is the moment of inertia of the i link with respect to the center of mass of the arm from the z_0 axis [$kg \cdot m^2$];
- k_{ri} is the reduction ratio of the i-th rotor;
- m_{mi} is the mass of the i-th motor [kg];
- I_{mi} is the moment of inertia of the rotor i with respect to the motor axis [$kg \cdot m^2$].

2.1 Kinematics

The reference systems of each joint have been fixed in accordance with the Denavit-Hartenberg conventions, where:

- a_i is the distance between O_i and $O_{i'}$;
- α_i is the angle between the axes z_{i-1} and z_i, rotating around x_i (positive if counterclockwise);
- d_i is the coordinate of $O_{i'}$ along z_{i-1};
- θ_i is the angle between the axes x_{i-1} and x_i, rotating around z_{i-1} (positive if counterclockwise).

Being characterized by two **rotoidal joints**, for the manipulator under consideration a_i, α_i and d_i are constants, while θ_i are the joint variables, for i = 1, 2.

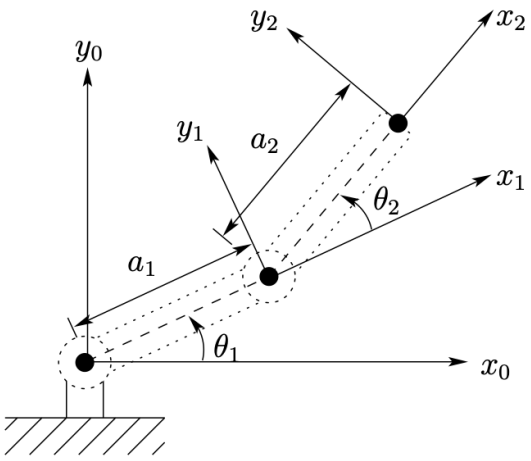


Figure 2.2: Reference systems according to Denavit-Hartenberg

	a_i	α_i	d_i	θ_i
arm no. 1	a_1	0	0	θ_1
arm no. 2	a_2	0	0	θ_2

Table 2.2: Denavit-Hartenberg parameters

The **homogeneous transformation** matrix is:

$$T_n^0(q) = \begin{bmatrix} cos(\theta_1+\theta_2) & -sin(\theta_1+\theta_2) & 0 & a_1 \cdot cos(\theta_1) + a_2 \cdot cos(\theta_1+\theta_2) \\ sin(\theta_1+\theta_2) & cos(\theta_1+\theta_2) & 0 & a_1 \cdot sin(\theta_1) + a_2 \cdot sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Therefore, the vector of the terminal organ coordinates with respect to the base triplet is:

$$p^0(q) = \begin{bmatrix} a_1 \cdot cos(\theta_1) + a_2 \cdot cos(\theta_1 + \theta_2) \\ a_1 \cdot sin(\theta_1) + a_2 \cdot sin(\theta_1 + \theta_2) \\ 0 \end{bmatrix}$$

This vector will be used later to impose constraints regarding collisions with obstacles (section 4.1).

2.2 Dynamics

The robot can be described by the following equation that regulates its dynamics, assuming the absence of external forces:

$$\tau = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + F_s sgn(\dot{q}) + G(q)$$

Where:

- τ is the vector (n x 1) of the **torques** applied to the joints;
- $B(q)$ is the **inertia matrix** (n x n), symmetric, positive definite and, in general, dependent on the configuration:

$$B(q) = \sum_{i=1}^{n} (m_{l_i} J_P^{(l_i)T} J_P^{(l_i)} + J_O^{(l_i)T} I_{l_i} J_O^{(l_i)} + m_{m_i} J_P^{(m_i)T} J_P^{(m_i)} + J_O^{(m_i)T} R_{m_i} I_{m_i}^{m_i} R_{m_i}^T J_O^{(m_i)})$$

  With $I_{l_i} = R_i I_{l_i}^i R_i^T$ symmetric matrix representing the **inertia tensor** relative to the center of gravity **of the i-th arm**, expressed with respect to the base triad:

$$I_{l_i} = \begin{bmatrix} \int (r_{iy}^2 + r_{iz}^2) \cdot \rho \, dV & -\int r_{ix} \cdot r_{iy} \cdot \rho \, dV & -\int r_{ix} \cdot r_{iz} \cdot \rho \, dV \\ * & \int (r_{ix}^2 + r_{iz}^2) \cdot \rho \, dV & -\int r_{iy} \cdot r_{iz} \cdot \rho \, dV \\ * & * & \int (r_{ix}^2 + r_{iy}^2) \cdot \rho \, dV \end{bmatrix} = \begin{bmatrix} I_{l_i xx} & -I_{l_i xy} & -I_{l_i xz} \\ * & I_{l_i yy} & -I_{l_i yz} \\ * & * & I_{l_i zz} \end{bmatrix}$$

  Assuming that the **rotor** is a solid of revolution around its axis of rotation, its **inertia tensor** $I_{m_i}^{m_i}$, expressed in a triplet R_{m_i} with origin in the center of gravity and axis z_{m_i} coinciding with the axis of rotation, can be written as:

$$I_{m_i}^{m_i} = \begin{bmatrix} I_{m_i xx}^{m_i} & 0 & 0 \\ 0 & I_{m_i yy}^{m_i} & 0 \\ 0 & 0 & I_{m_i zz}^{m_i} \end{bmatrix}$$

  The rotation matrices R_i and R_{m_i} express, respectively, the orientation of the triplet

integral with arm i and of the triplet integral with mass i with respect to the base triplet. However, since ω_i (i = 1, 2) is aligned with z_0, R_i and R_{m_i} have no effect. In fact, it was also assumed previously that the motors are positioned so that the axis of their joint has its center of mass at the origin of the respective reference system. Consequently, reference can be made to the scalar moments of inertia I_{l_i} and I_{m_i}, associated with the axis z_0.
The **Jacobians** are necessary to express the kinetic energy as a function of the generalized coordinates of the system, i.e., the joint variables. In the calculation, the motor of joint i is considered to be located on arm $i-1$, and the geometric method that characterizes the intermediate arm instead of the terminal member has been adopted.

The matrices (3 x n) $J_P^{(l_i)}$ and $J_P^{(m_i)}$ are the geometric Jacobians that take into account the contribution of the arm and rotor speeds of joint i to the linear speed:

$$J_P^{(l_1)} = \begin{bmatrix} -l_1 \cdot sin(\theta_1) & 0 \\ l_1 \cdot cos(\theta_1) & 0 \\ 0 & 0 \end{bmatrix}, \quad J_P^{(l_2)} = \begin{bmatrix} -a_1 \cdot sin(\theta_1) - l_2 \cdot sin(\theta_1 + \theta_2) & -l_2 \cdot sin(\theta_1 + \theta_2) \\ a_1 \cdot cos(\theta_1) + l_2 \cdot cos(\theta_1 + \theta_2) & l_2 \cdot cos(\theta_1 + \theta_2) \\ 0 & 0 \end{bmatrix}$$

$$J_P^{(m_1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad J_P^{(m_2)} = \begin{bmatrix} -a_1 \cdot sin(\theta_1) & 0 \\ a_1 \cdot cos(\theta_1) & 0 \\ 0 & 0 \end{bmatrix}$$

The matrices (3 x n) $J_O^{(l_i)}$ and $J_O^{(m_i)}$, on the other hand, are the geometric Jacobians that take into account the contribution of the arm and rotor speeds of the i-th joint to the angular velocity:

$$J_O^{(l_1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad J_O^{(l_2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

$$J_O^{(m_1)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ k_{r1} & 0 \end{bmatrix}, \quad J_O^{(m_2)} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & k_{r2} \end{bmatrix}$$

Performing the calculations yields the following matrix:

$$B(q) = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$b_{11} = I_{l1} + m_{l1} \cdot l_1^2 + k_{r1}^2 \cdot I_{m1} + I_{l2} + m_{l2} \cdot [a_1^2 + l_2^2 + 2 \cdot a_1 \cdot l_2 \cdot cos(\theta_2)] + I_{m2} + m_{m2} \cdot a_1^2$$

$$b_{12} = b_{21} = I_{l2} + m_{l2} \cdot [l_2^2 + a_1 \cdot l_2 \cdot cos(\theta_2)] + k_{r2} \cdot I_{m2}$$

$$b_{22} = I_{l2} + m_{l2} \cdot l_2^2 + k_{r2}^2 \cdot I_{m2}$$

- $C(q, \dot{q})$ is the (n x n) matrix that takes into account the contribution of centrifugal and **Coriolis** accelerations due to the speeds of the joints:

$$h = -m_{l2} \cdot a_1 \cdot l_2 \cdot sin(\theta_2)$$

$$C(q, \dot{q}) = \begin{bmatrix} h \cdot \dot{\theta}_2 & h \cdot (\dot{\theta}_1 + \dot{\theta}_2) \\ -h \cdot \dot{\theta}_1 & 0 \end{bmatrix}$$

- F_v and F_s are two diagonal matrices (n x n), containing, on the main diagonal, the viscous and static friction contributions, respectively, for each joint. The simplest friction model has the following form as a function of velocity:

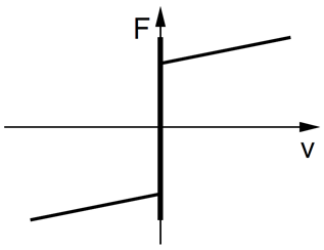


Figure 2.3: Coulombian Friction

- $G(q)$ is the vector (n x 1) containing the pairs generated by the force of **gravity**, with $g = 9.81 \ \frac{m}{s^2}$ and assuming $g_0 = \begin{bmatrix} 0 & -g & 0 \end{bmatrix}^T$:

$$G(q) = \begin{bmatrix} (m_{l1} \cdot l_1 + m_{m2} \cdot a_1 + m_{l2} \cdot a_1) \cdot g \cdot cos(\theta_1) + m_{l2} \cdot l_2 \cdot g \cdot cos(\theta_1 + \theta_2) \\ m_{l2} \cdot l_2 \cdot g \cdot cos(\theta_1 + \theta_2) \end{bmatrix}$$

In this year's exam, viscous and static friction have been neglected for simplicity.

3 Control strategies

Centralized control is implemented by adjusting the torque and, therefore, in turn the current, when the interaction between the joints cannot be assimilated to a disturbance and/or is not negligible. This usually occurs when high speeds and/or accelerations and, consequently, torques are required, but also in the presence of direct couplings ($K_r = I$). In this case, the nth order system cannot be decoupled into n independent and linearized SISO systems; this assumption would simplify the control, which could then be implemented with n PIDs, in the most general case, to regulate the position, speed, and/or acceleration of each joint. The following diagram is therefore implemented:

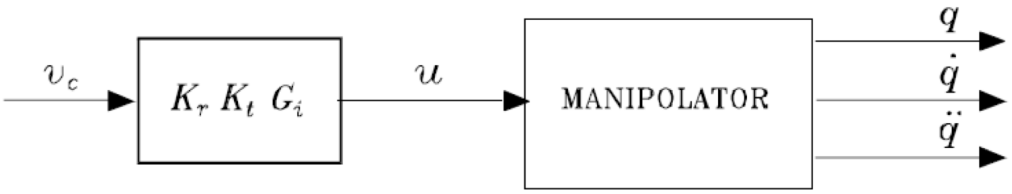


Figure 3.1: Centralized control

Where, taking into account the transmission ratio matrices K_r, the torque constants K_t, the gains G_i and the control voltage v_c, we have:

$$u = \tau = K_r \tau_m = K_r K_t i_a = K_r K_t G_i v_c$$

3.1 Model Predictive Control

Let us consider the system to be controlled, expressed in the form $\dot{x} = f(x, u)$, $y = g(x)$, where $x \in \mathbb{R}^n$ is the state (n = number of states), $u \in \mathbb{R}^m$ is the control input (m = number of inputs) and $y \in \mathbb{R}^p$ is the output (p = number of outputs), with x, u and y functions of time. In this specific case:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_3 \\ x_4 \\ B(q)^{-1}[u - C(q, \dot{q})\dot{q} - G(q)] \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

In ***Model Predictive Control*** (MPC), at each sampling step T_c, the set of control actions (present and future) $\{u^*(k), ..., u^*(k+H_c-1)\}$, where H_c is the control horizon, which optimizes the objective function V(k) at step k, subject to the imposed constraints, at the instants $\{t_{k+1}, ..., t_{k+H_p}\}$. Once this is done, only the sample $u^*(k)$ is applied to the process input, and the procedure is repeated iteratively at the next step. H_p is the prediction horizon, which is generally much larger than the control horizon and influences the performance of the controller. The cost function generally takes this form [1]:

$$V(k) = \sum_{i=H_w}^{H_p} ||y(k+i|k) - r(k+i|k)||^2_{Q(i)} + \sum_{i=0}^{H_c-1} ||\Delta u(k+i|k)||^2_{R(i)}$$

That is, the objective is to determine the optimal control action that minimizes the cost $V(k)$, which balances the tracking error, i.e., the difference between the controlled outputs and their respective references, and the variations in u, evaluating their norm-2 weighted by the matrices Q and R.

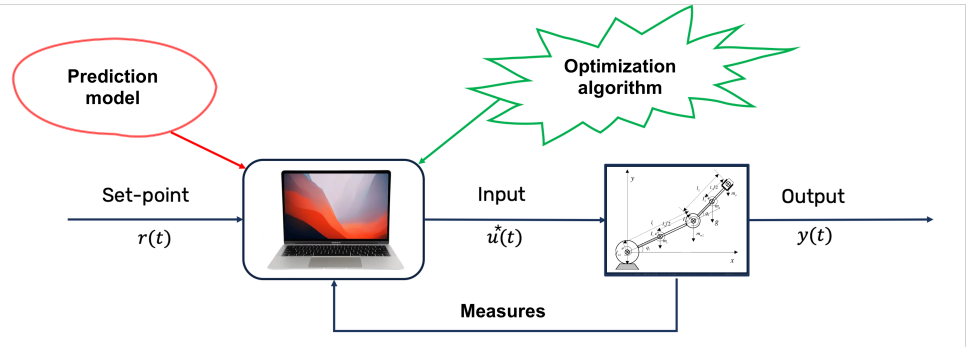


Figure 3.2: MPC-based control scheme

Various constraints can be imposed on inputs, outputs, and states (for example, maximum and minimum values can be set, etc.).

3.1.1 Passivity-based MPC

MPC is not always able to guarantee the stability of a nonlinear system, which is why strategies based on **passivity** are introduced. The two schemes can be combined into one, maintaining the advantages they offer individually (i.e., feasibility and closed-loop stability of passivity, and optimality with MPC constraints).
Passivity is the property whereby, at any given moment, the amount of energy that a system can supply to the environment does not exceed the amount that has been supplied to it. This implies that, as time evolves, it absorbs a fraction of the energy supplied and transforms it into another form. When a system is passive, it may be advantageous to use information about the

physical structure of the system and design a controller to model its energy. Similar to PD with gravity compensation, a constant equilibrium position q_d must be specified. The structure of the control action that guarantees global asymptotic stability must be defined. Since the reference is constant ($\tilde{q} = q - q_d = q$), the following applies:

$$\tau = B(q)\ddot{\tilde{q}} + C(q,\dot{q})\dot{\tilde{q}} + F\dot{\tilde{q}} + G(q)$$

The candidate Lyapunov function, the sum of two quadratic forms, is [2]:

$$V(\tilde{q},\dot{\tilde{q}}) = \frac{1}{2}\dot{\tilde{q}}^T B(q)\dot{\tilde{q}} + \frac{1}{2}\tilde{q}^T K_P \tilde{q} > 0 \qquad \forall \tilde{q}, \dot{\tilde{q}} \neq 0$$

Where $B(q)$ and K_P are positive definite matrices, while F is the diagonal matrix of viscous and electrical frictions.

$$\begin{aligned}\dot{V}(\tilde{q},\dot{\tilde{q}}) &= \frac{1}{2}\dot{\tilde{q}}^T \dot{B}(q)\dot{\tilde{q}} + \dot{\tilde{q}}^T B(q)\ddot{\tilde{q}} + \dot{\tilde{q}}^T K_P \tilde{q} \\ &= \frac{1}{2}\dot{\tilde{q}}^T [\dot{B}(q) - 2C(q,\dot{q})]\dot{\tilde{q}} - \dot{\tilde{q}}^T F + \dot{\tilde{q}}^T [\tau - G(q) + K_P \tilde{q}] \\ &= -\dot{\tilde{q}}^T F + \dot{\tilde{q}}^T u \leq \dot{\tilde{q}}^T u\end{aligned}$$

having set $\mathbf{u} = \boldsymbol{\tau} + \mathbf{G(q)} - \mathbf{K_P}\tilde{\mathbf{q}}$, where τ is the model input. Therefore, the system is passive with respect to the input $u_p = u$ and the output $y_p = \dot{\tilde{q}}$, since V (*storage function)* is non-increasing. The system will tend to dissipate energy and reach a stable steady-state equilibrium point. At each step, the optimization algorithm will impose an additional constraint to ensure stability:

$$y_p^T u_p + \rho y_p^T y_p = \dot{\tilde{q}}^T u + \rho \dot{\tilde{q}}^T \dot{\tilde{q}} \leq 0 \qquad \rho > 0$$

Therefore, a proportional action on the position error and a gravity compensation are introduced, which requires online calculation and exact knowledge of the vector g(q).

3.2 Robust Control

Typically, the strategies used for centralized control are PD with gravity compensation and dynamic inversion. However, these methods are based on imperfect models, due to approximations necessary to lighten the computational load and/or due to imperfect knowledge of the manipulator model. As a result, they do not always guarantee satisfactory performance, due to inadequate compensation.

Robust control, as well as adaptive control, is therefore introduced to counterbalance these effects. The former involves the introduction of an additional term, with respect to the control

loop implemented for inverse dynamics, which guarantees greater robustness.
The control vector takes the form $u = \hat{B}(q)y + \hat{n}(q, \dot{q})$, having only an estimate of the dynamic model of the manipulator available, which ensures the compensation of nonlinear effects and the decoupling between the joints, in an approximate manner. Therefore, given the reference trajectory $q_d(t)$ in joint space and defining the position error as $\tilde{q} = q_d - q$, we obtain:

$$\ddot{\tilde{q}} = \ddot{q}_d - y + \eta$$

where $\boldsymbol{\eta}$ represents an approximation of the variability of the uncertainty, assumed to be realistically known, while y is the control law. The trajectory tracking problem is transformed into one based on **error stabilization**, which is assumed as the new state of the system:

$$\xi = \begin{bmatrix} \tilde{q} \\ \dot{\tilde{q}} \end{bmatrix}; \qquad \dot{\xi} = H\xi + D(\ddot{q}_d - y + \eta)$$

$$H = \begin{bmatrix} O & I \\ O & O \end{bmatrix}_{(2nx2n)} \qquad D = \begin{bmatrix} O \\ I \end{bmatrix}_{(2nxn)}$$

By choosing $\mathbf{y} = \ddot{\mathbf{q}}_\mathbf{d} + \mathbf{K_D}\dot{\tilde{\mathbf{q}}} + \mathbf{K_P}\tilde{\mathbf{q}} + \mathbf{w}$, PD control is implemented to stabilize the dynamics with which the error evolves, with linear feedforward compensation of the second derivative of the position reference. In fact, with the gain matrices, the error trend can be adjusted; since $K_P = diag\{w_{n1}^2, ..., w_{nn}^2\}$ and $K_D = diag\{2\delta_1 w_{n1}, ..., 2\delta_n w_{nn}\}$, we obtain n independent equations. Instead, the term $\boldsymbol{w}$ contributes to ensuring the robustness of the control:

$$w = \begin{cases} \frac{\rho}{||\mathbf{z}||}\mathbf{z} & for \quad ||z|| \geq \epsilon \\ \frac{\rho}{\epsilon}\mathbf{z} & for \quad ||z|| < \epsilon. \end{cases}$$

It is implemented in this way to ensure error convergence and avoid points of discontinuity, which would impose instantaneous variations in the control signal. This would require switching at frequencies that are not feasible, violating the physical limits of the actuators. The higher the frequency, the closer we get to the ideal case (Fig. 3.3); therefore, this law does not guarantee the convergence of the error to zero, but its limitation in norm (the error falls within a range whose amplitude depends on ϵ). Furthermore, $\dot{V} < 0$ along all directions of ξ, if $\rho \geq ||\eta||$, $\forall q, \dot{q}, \ddot{q}_d$, with ρ a positive scalar.
Setting:

$$K = \begin{bmatrix} K_P & K_D \end{bmatrix} \qquad \tilde{H} = H - DK = \begin{bmatrix} O & I \\ -K_P & -K_D \end{bmatrix}$$

we have $\dot{\boldsymbol{\xi}} = \tilde{\mathbf{H}}\boldsymbol{\xi} + \mathbf{D}(\boldsymbol{\eta} - \mathbf{w})$. Since $\tilde{H}$ is a matrix with negative real eigenvalues and choosing any symmetric and positive definite matrix P, with Lyapunov's equation ($\tilde{H}^T Q + Q\tilde{H} = -P$) it is possible to determine the unique, symmetric and positive definite matrix Q. Regardless of the initial error, the trajectory will converge on the sliding subspace $z = D^T Q\xi = 0$, thanks to the conditions imposed, with a transient that depends on P, K_P, and K_D. This strategy is generally used for systems that are able to withstand stresses and oscillations, which will have limited amplitude as the frequency increases. In fact, the error converges to zero with an oscillatory trend due to delays introduced by digital implementation and physical components (chattering):

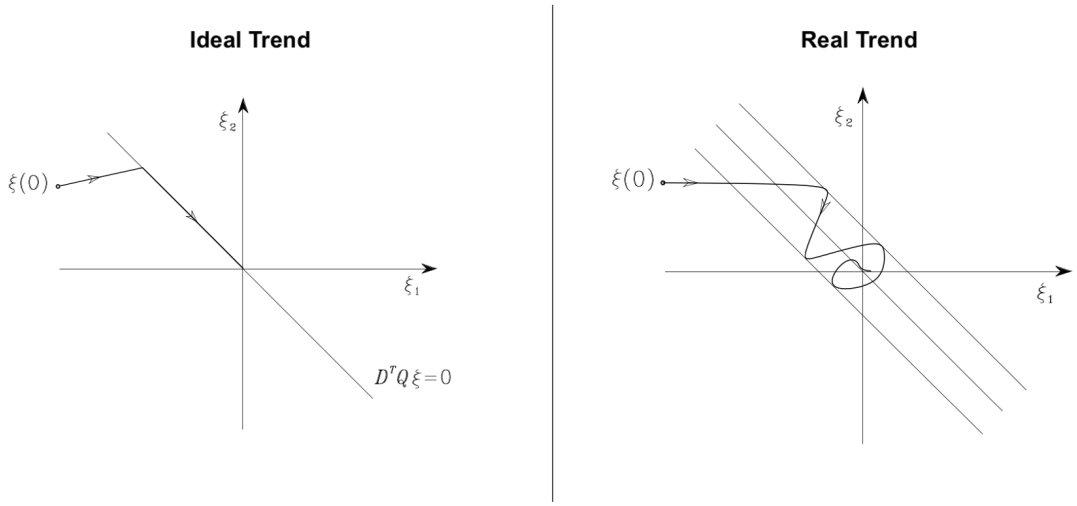


Figure 3.3: Error trajectory with the robust control scheme

3.3 Adaptive Control

Unlike robust control, **adaptive control** adjusts the parameters of the model used for inverse dynamics *online* to those of the real system, using an appropriate update law. As a result, the computational cost increases, but so does performance, if the adaptation is successful. This is possible because the dynamics can be expressed in a linear form in the parameters (**LIP**):

$$u = \tau = Y(q, \dot{q}, \ddot{q})\pi$$

$$Y(q, \dot{q}, \ddot{q}) = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & y_{15} & y_{16} & y_{17} & y_{18} \\ 0 & 0 & 0 & 0 & y_{25} & y_{26} & y_{27} & y_{28} \end{bmatrix}$$

$$y_{11} = a_1^2 \cdot \ddot{\theta}_1 + a_1 \cdot g \cdot cos(\theta_1); \qquad y_{12} = 2 \cdot a_1 \cdot \ddot{\theta}_1 + g \cdot cos(\theta_1); \qquad y_{13} = \ddot{\theta}_1; \qquad y_{14} = k_{r1}^2 \cdot \ddot{\theta}_1;$$

$$y_{15} = (a_1^2 + 2 \cdot a_1 \cdot a_2 \cos\theta_2 + a_2^2)\ddot{\theta}_1 + (a_1 \cdot a_2 \cos\theta_2 + a_2^2)\ddot{\theta}_2 - 2 \cdot a_1 \cdot a_2 \sin\theta_2 \dot{\theta}_1 \dot{\theta}_2 - a_1 \cdot a_2 \sin\theta_2 \dot{\theta}_2^2 + a_1 \cdot g \cos\theta_1 + a_2 \cdot g \cos(\theta_1 + \theta_2);$$

$$y_{16} = (2 \cdot a_1 \cdot cos(\theta_2) + 2 \cdot a_2) \cdot \ddot{\theta}_1 + (a_1 \cdot cos(\theta_2) + 2 \cdot a_2) \cdot \ddot{\theta}_2 - 2 \cdot a_1 \cdot sin(\theta_2) \cdot \dot{\theta}_1 \cdot \dot{\theta}_2 - a_1 \cdot sin(\theta_2) \cdot \dot{\theta}_2^2 + g \cdot cos(\theta_1 + \theta_2);$$

$$y_{17} = \ddot{\theta}_1 + \ddot{\theta}_2; \qquad y_{18} = k_{r2} \cdot \ddot{\theta}_2;$$

$$y_{25} = (a_1 \cdot a_2 \cdot cos(\theta_2) + a_2^2) \cdot \ddot{\theta}_1 + a_2^2 \cdot \ddot{\theta}_2 + a_1 \cdot a_2 \cdot sin(\theta_2) \cdot \dot{\theta}_1^2 + a_2 \cdot g \cdot cos(\theta_1 + \theta_2);$$

$$y_{26} = (a_1 \cdot cos(\theta_2) + 2 \cdot a_2) \cdot \ddot{\theta}_1 + 2 \cdot a_2 \cdot \ddot{\theta}_2 + a_1 \cdot sin(\theta_2) \cdot \dot{\theta}_1^2 + g \cdot cos(\theta_1 + \theta_2);$$

$$y_{27} = \ddot{\theta}_1 + \ddot{\theta}_2; \qquad y_{28} = k_{r2} \cdot \ddot{\theta}_1 + k_{r2}^2 \cdot \ddot{\theta}_2.$$

$$\pi = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 & \pi_6 & \pi_7 & \pi_8 \end{bmatrix}^T$$

$$\pi_1 = m_{l1} + m_{m2}; \qquad \pi_2 = m_{l1} \cdot (l_1 - a_1); \qquad \pi_3 = I_{l1} + m_{l1} \cdot (l_1 - a_1)^2 + I_{m2}; \qquad \pi_4 = I_{m1};$$

$$\pi_5 = m_{l2}; \qquad \pi_6 = m_2 l_{C2} = m_{l2} \cdot (l_2 - a_2); \qquad \pi_7 = \hat{I}_2 = I_{l2} + m_{l2} \cdot (l_2 - a_2)^2; \qquad \pi_8 = I_{m2}.$$

Where π is a vector (p x 1) of constant parameters, with respect to which the method is adaptive, while Y is the matrix (n x p) of **regressors**, a function of the positions, velocities, and accelerations of the joints.
The control law includes estimates of the matrices characterizing the manipulator model and parameters, as well as a PD action on the error $\boldsymbol{\sigma} = \dot{q}_d + \Lambda\tilde{q} - \dot{q} = \dot{\tilde{q}} + \Lambda\tilde{q}$, where Λ is a positive definite matrix for filtering, in a form that allows the position error and its first derivative to be weighted:

$$u = \hat{B}\ddot{q}_r + \hat{C}(q, \dot{q})\dot{q}_r + \hat{F}\dot{q}_r + \hat{g} + K_D\sigma = Y(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\hat{\pi} + K_D\sigma$$

The first term contributes to compensation and decoupling, approximately, in a similar way to robust control, while the second stabilizes the error (K_D must be defined as positive). If the estimate vector is updated over time according to the **gradient-type adaptive law**:

$$\dot{\hat{\pi}} = K_\pi^{-1} Y^T(q, \dot{q}, \dot{q}_r, \ddot{q}_r)\sigma$$

the manipulator trajectories will converge, globally and asymptotically, to $\sigma = 0$ and $\tilde{q} = 0$, with a speed that depends on K_π (p x p) (matrix of learning rates); therefore, correct compensation will only be achieved when operating at full capacity. However, although it is more accurate

than robust control, the computational cost increases and a complete model of the system is required. Furthermore, it is not able to counteract any external disturbances. On the other hand, it does not require direct measurement of acceleration; the latter is an advantage, as accelerometers are not always available.

4 Implementation in MATLAB/Simulink

The reference trajectories in joint space have the following behavior over time, setting $t_i = 0$s, $t_f = 5$s, $T_c = 5$ms, $q_i = [0 \quad \frac{\pi}{4}]^T$ and $q_f = [\frac{\pi}{2} \quad \frac{\pi}{2}]^T$:

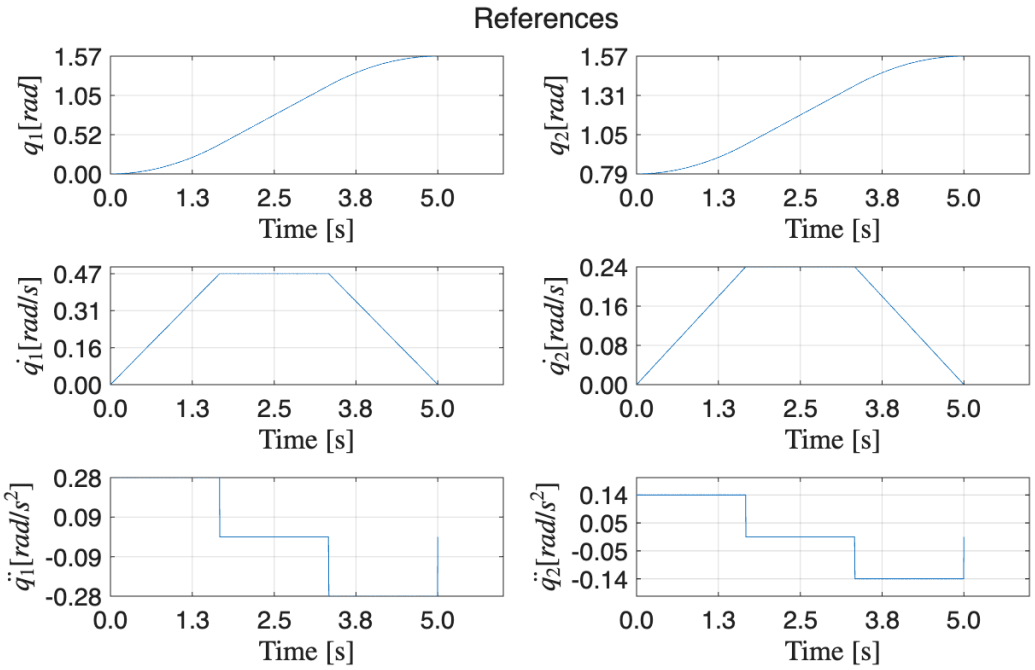


Figure 4.1: Reference trajectories

The position reference is obtained by implementing a trajectory with a trapezoidal velocity profile, to move from q_i to q_f, setting the final instant. High torques will be required, as the acceleration has a step profile, which involves sudden changes.

4.1 Model Predictive Control

4.1.1 Nonlinear MPC

The saturation of the actuators, i.e., the minimum and maximum limits of the torques applicable to the joints (**manipulable variables**), and the presence of obstacles limiting the movements of the manipulator were considered as **constraints**. The **measurable outputs** are the positions and velocities of the joints, as introduced above. In the first case, a **ceiling** was considered:

For simplicity, we assume that only the tip of the terminal organ must avoid collisions with the obstacle; the nonlinear constraint [3] will therefore be:

$$f(q) = a_1 \cdot sin(q_1) + a_2 \cdot sin(q_1 + q_2) \leq l_c$$

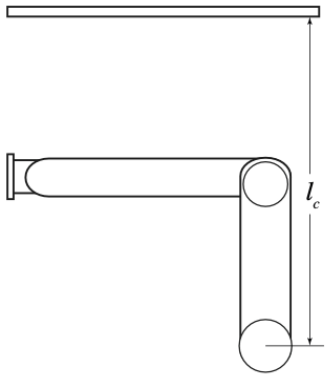


Figure 4.2: 1st obstacle considered for testing

$$f(q_i) = 0.707m; \qquad f(q_f) = 1m$$

The constraint must be compatible with the points of the trajectories, otherwise the desired position would never be reached. Consequently, it is necessary to define a custom constraint using an appropriate function with the extension ".m" in MATLAB. In addition, the state functions, output functions, and related Jacobians have been made explicit.
Finally, various tests were conducted, modifying parameters, constraints, and weights, to evaluate the performance of the controller. The terminal device successfully avoids the ceiling in every case tested. However, it was found that the time required to perform the calculations always exceeds the sampling time; therefore, in order to control the system in real time, T_c should be increased, H_P should be decreased, or the complexity of the problem should be reduced. However, it is not always possible to act on the sampling step; in fact, we are often limited by the specifications of the microcontroller and/or the dynamics of the system.

	H_P	H_c	T_c	t_f	l_c	W_{MV}	W_{MVR}	W_{OV}	Constraints	max$\|\|\tilde{q}\|\|$	max$\|\|\dot{\tilde{q}}\|\|$	t_{sim}	$V(t_f)$
1st	20	2	5ms	5s	2m	[0.1,0.1]	[0.1,0.1]	[1,1,1,1]	Yes	4.5rad	$3.3\frac{rad}{s}$	23s	506
2nd	20	2	5ms	5s	2m	[0,0]	[0.1,0.1]	[1,1,1,1]	Yes	3.8rad	$3.1\frac{rad}{s}$	33s	401
3°	20	2	5ms	5s	2m	[0,0]	[0,0]	[1,1,1,1]	Yes	0.4rad	$0.4\frac{rad}{s}$	47s	2.5
4°	20	2	5ms	5s	2m	[0,0]	[0,0]	[1,1,1,1]	No	0.03rad	$0.05\frac{rad}{s}$	30s	0.008

Table 4.1: First tests with MPC

- By increasing the **weights on the inputs** (W_{MV}) and their variations (W_{MVR}), the controller fails. On the other hand, however, this takes into account the control effort and makes the controller less efficient but closer to real-world operation.
- Reducing $\mathbf{H_P}$ makes the control action less aggressive and decreases the computational cost (so the test takes less time, t_{sim} decreases), but also worsens performance;
- Increasing $\mathbf{H_C}$ makes the control more responsive and accelerates convergence, but at the same time amplifies the variability of the outputs;

	H_P	H_c	T_c	t_f	l_c	W_{OV}	max$\|\|\tilde{q}\|\|$	max$\|\|\dot{\tilde{q}}\|\|$	t_{sim}	$V(t_f)$
5°	10	2	5ms	5s	2m	[1,1,1,1]	0.3rad	$0.2\frac{rad}{s}$	13s	0.6
6th	50	2	5ms	5s	2m	[1,1,1,1]	0.2rad	$0.22\frac{rad}{s}$	4min	0.26
7th	20	10	5ms	5s	2m	[1,1,1,1]	0.2rad	$0.16\frac{rad}{s}$	27s	0.4
8°	20	2	10ms	5s	2m	[1,1,1,1]	0.13rad	$0.11\frac{rad}{s}$	30s	0.1
9°	20	2	5ms	10s	2m	[1,1,1,1]	0.1rad	$0.09\frac{rad}{s}$	1min	0.05
10°	20	2	5ms	5s	2m	[5,5,1,1]	0.15rad	0.2rad/s	48s	0.007
11°	20	2	5ms	5s	2m	[1,1,5,5]	0.11rad	$0.1\frac{rad}{s}$	48s	0.18
12°	20	2	5ms	5s	2m	[5,5,5,5]	0.1rad	$0.1\frac{rad}{s}$	48s	3
13°	20	2	5ms	5s	2m	[10,10,1,1]	0.15rad	$0.38\frac{rad}{s}$	53s	0.003
14°	15	2	5ms	5s	2m	[5,5,1,1]	0.14rad	$0.16\frac{rad}{s}$	29s	0.007
15°	20	2	5ms	5s	2m	[10,10,5,5]	0.1rad	$0.09\frac{rad}{s}$	41s	2.5
16°	15	2	5ms	5s	2m	[1,1,5,5]	0.1rad	0.1rad/s	30s	0.14
17°	15	5	5ms	5s	2m	[2,2,2,2]	0.1rad	$0.1\frac{rad}{s}$	24s	0.3
18°	15	2	5 ms	5 s	1 m	[2.2,2.2,2.2]	1.6 rad	$0.5\frac{rad}{s}$	32 s	147.1
19°	15	2	5ms	5s	1.8m	[10,10,5,5]	0.1rad	$0.4\frac{rad}{s}$	29s	8

Table 4.2: Latest tests with MPC

- By prolonging $\mathbf{T_C}$, we note that the cost function at the last step decreases, as does the maximum position error in norm;
- Slowing down the reference trajectory, i.e., increasing $\mathbf{t_f}$, results in improvements, even if the experiment takes twice as long. However, it is not always possible to change the final instant of the trajectory, especially when the manipulator is integrated into a system where a certain timing and synchronization must be guaranteed.
- Without **constraints**, the computational load is lightened and tracking improves, but this does not represent a situation closer to reality. If the limitations on the applicable pairs become more stringent, the constrained optimization problem becomes more complex and performance worsens;
- The higher the ceiling (as l_c increases), the better the tracking becomes.
- Increasing the **weights on the outputs** (W_{OV}) makes the implementation more aggressive and speeds up convergence, but the y also become more oscillatory. It should be noted that increasing only the weights related to the joint variables worsens the tracking of the velocity trajectory (*trade-off*), and vice versa. Furthermore, it is not advisable to increase these weights, because there is no noticeable improvement. The smaller the weight on a certain output, the less it will be subject to optimization, as it has less importance in the cost function. If the weights relative to the variables of one joint increase, the performance of the other joint worsens.

By halving H_P, the simulation lasts half as long, but the maximum position error in norm and

$V(t_f)$ double; therefore, even in this case, the trade-off must be evaluated. In conclusion, it is necessary to calibrate the hyperparameters in order to determine their optimal values.

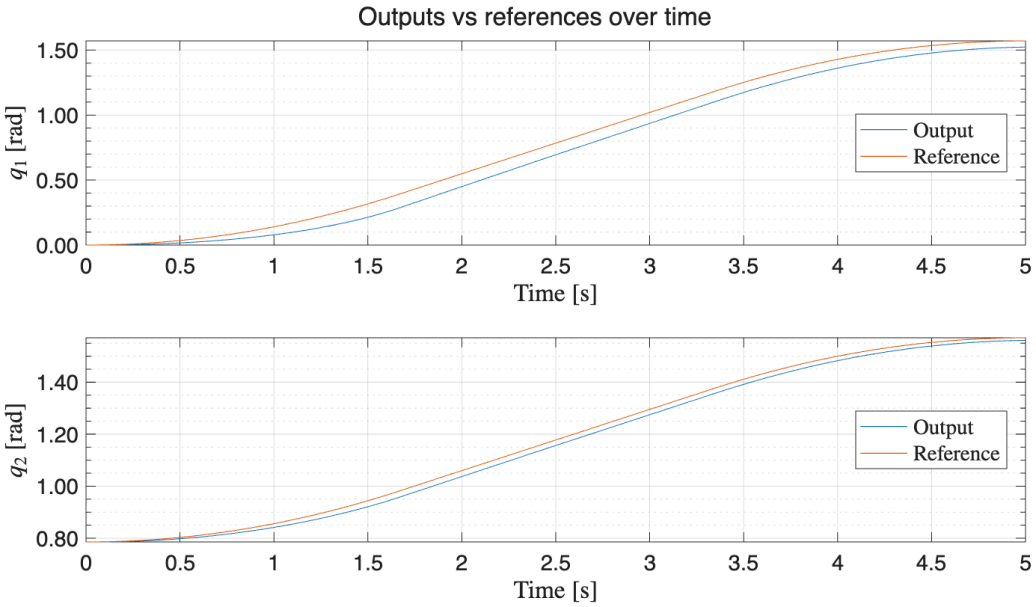


Figure 4.3: Nonlinear MPC, 19th test: joint variables over time

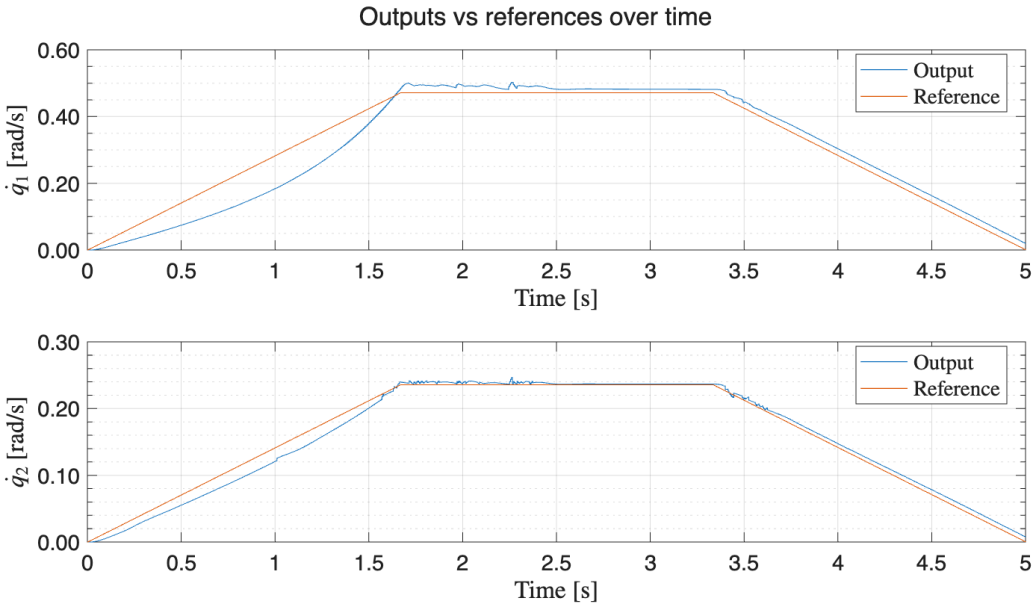


Figure 4.4: Nonlinear MPC, 19th test: speed over time

In addition to the ceiling, tests were also conducted with another obstacle, namely a circle in the (x, y) plane:
The constraint was modeled as follows [3]:

$$f(q) = -[a_1 \cdot cos(q_1) + a_2 \cdot cos(q_1 + q_2) - c_x]^2 - [a_1 \cdot sin(q_1) + a_2 \cdot sin(q_1 + q_2) - c_y]^2 \leq -r_c^2$$

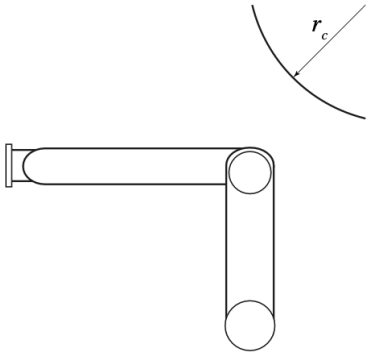


Figure 4.5: 2nd obstacle considered for testing

The performance is comparable to that obtained in the previous case:

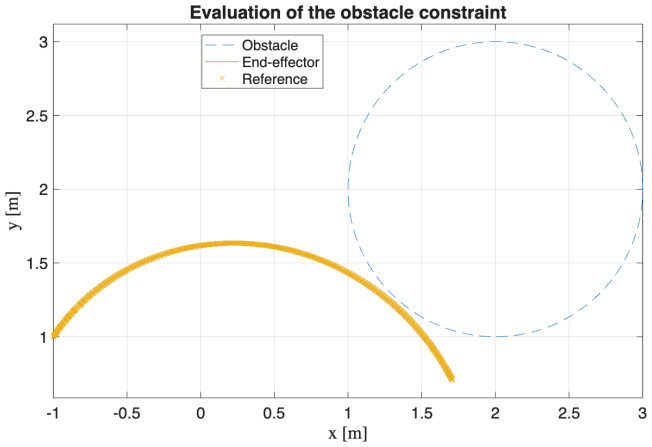


Figure 4.6: nlMPC: collisions with a circle

4.1.2 Linear MPC

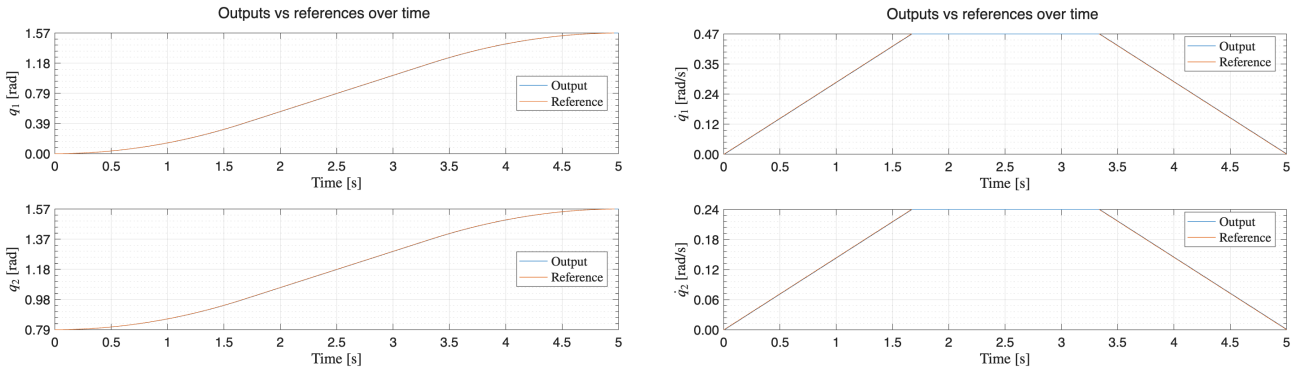
To reduce the simulation time by 33% , the code can be run with the **"*Run As Linear MPC*"** option, for example in adaptive mode. By doing so, the model prediction can be adapted in *real-time* to compensate for the nonlinearities of the system. Performance comparable to that guaranteed by the nonlinear controller was found, except for acceleration, which deviates from the desired value.

The alternative that allows t_{sim} to be reduced to $1s$, errors of the order of 10^{-3} to be obtained, and cost functions of approximately 10^{-4} to be achieved, is to exploit the Jacobians of the states and outputs to obtain the **linearized system**. In this way, we obtain the typical representation in the form of states (A, B, C, D), which, however, depends on the state around which the approximation is calculated. The further away we are from this point, the worse the performance becomes. Furthermore, even in this case, the accelerations are very different from

the reference ones. However, apart from a few peaks in the first $0.5s$, the calculation time does not exceed the sampling rate, so the system can be controlled in real time, albeit not robustly. In this case, however, the nonlinear constraint relating to the collision with the obstacle cannot be imposed. A possible solution would be to approximate the constraints as well; however, even in this circumstance, this technique is only acceptable when working close to the linearization point.

	H_P	H_c	W_{MV}	W_{MVR}	W_{OV}	$\max\|\|\tilde{q}\|\|[rad]$	$\max\|\|\dot{\tilde{q}}\|\|[\frac{rad}{s}]$	t_{sim}	$maxV(k)$	$V(t_f)$
1°	4	2	[0,0]	[0,0]	[1,1,1,1]	$2.8 \cdot 10^{-3}$	$1.7 \cdot 10^{-3}$	1s	$9.5 \cdot 10^{-5}$	$2.6 \cdot 10^{-7}$
2nd	25	2	[0,0]	[0.1,0.1]	[10,10,10,10]	$4.1 \cdot 10^{-2}$	$5.7 \cdot 10^{-2}$	1s	5.9	2.5

Table 4.3: Test lMPC



(a) Linear MPC: 1st test, outputs over time

(b) Linear MPC: 2nd test, control signals

Figure 4.7: Comparison of Linear MPC results in two test scenarios

4.1.3 Passivity-based MPC

In the presence of a constant reference, performance deteriorated when tracking the equilibrium position. In particular, prolonged oscillations were triggered, requiring an increase in H_P and weights on the tracking error, but with an increase in t_{sim}. The nonlinear MPC was therefore replaced with one based on **passivity**. This strategy allows the energy of the system to be controlled by modifying the control law. The passive approach helps to keep the system closer to its equilibrium state by reducing oscillations and/or ensuring stability. This also makes the system more robust to external disturbances, such as friction, changes in system parameters, or load. Passivity inputs and outputs must be defined in the relevant functions (*"Passivity.InputFcn"* and *"Passivity.OutputFcn"*), while the constraint is specified with *"Passivity.EnforceConstraint=true"*, with ρ to be entered in the *"Passivity.InputPassivityIndex"* field. Compared to previous versions, there is an additional constraint, so the computational cost increases.

	H_P	H_c	W_{OV}	t_{sim}	K_P	max $OS_{\%}$	max $t_{a2\%}$
1st	20	2	[5,5,0,0]	50s	50	3 %	4s
2nd	20	2	[5,5,0,0]	43s	300	22 %	3.9s
3rd	20	2	[5.5,0.0]	37s	750	36 %	3.9s
4th	20	2	[5.5,0.0]	29s	1500	24 %	3.1s
5th	20	2	[5.5,0.0]	29s	3750	13%	3s
6th	20	2	[5.5,0.0]	26s	16750	8%	2.3s

Table 4.4: Passivity-based MPC test

As K_P increases, it becomes faster in locking onto the output (reducing t_{sim} and $t_{a2\%}$), but also more oscillatory and aggressive in terms of torque and acceleration.

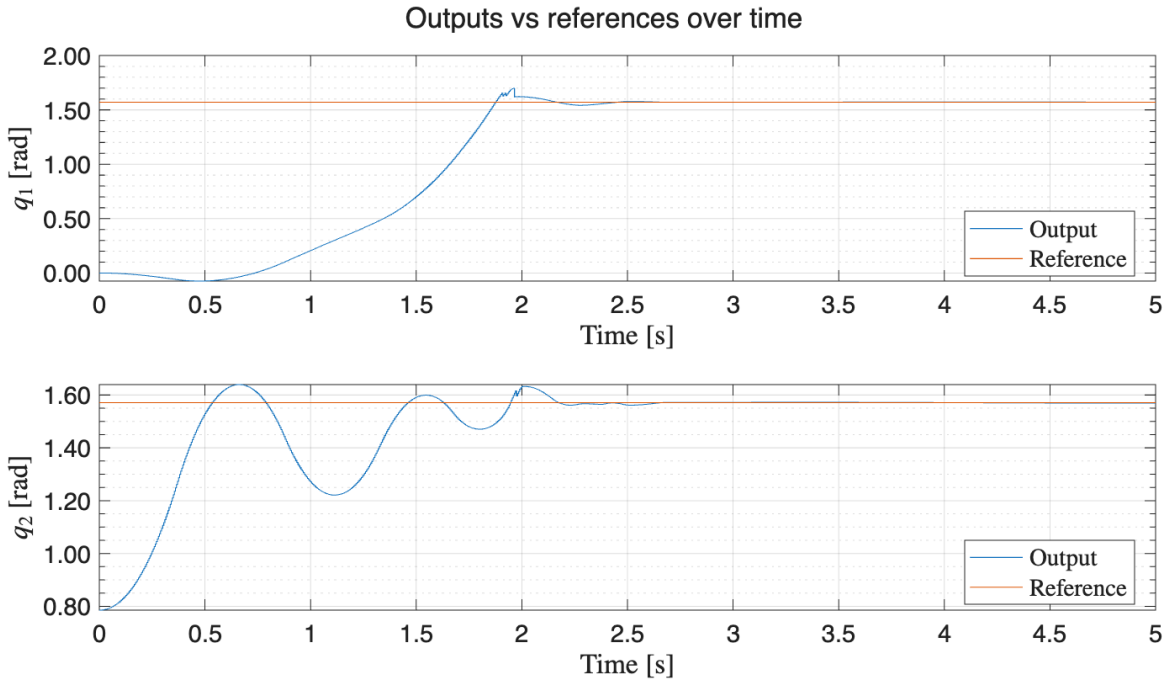


Figure 4.8: Passivity-based MPC: 6th test, outputs over time

4.2 Robust Control

The following diagram was implemented on Simulink:
Inertia ($\hat{B}(q) = \bar{B}$) was set constant and partial compensation ($\hat{n}(q, \dot{q}) = G(q)$) was considered, while the parameters were set equal to [4]:

Choosing a small threshold value (ϵ) determines the presence of high-frequency components in the joint pairs (Fig. 4.12), resulting in a limited tracking error. This occurs because w increases and we approach the ideal case, therefore requiring higher control signal switching frequencies.

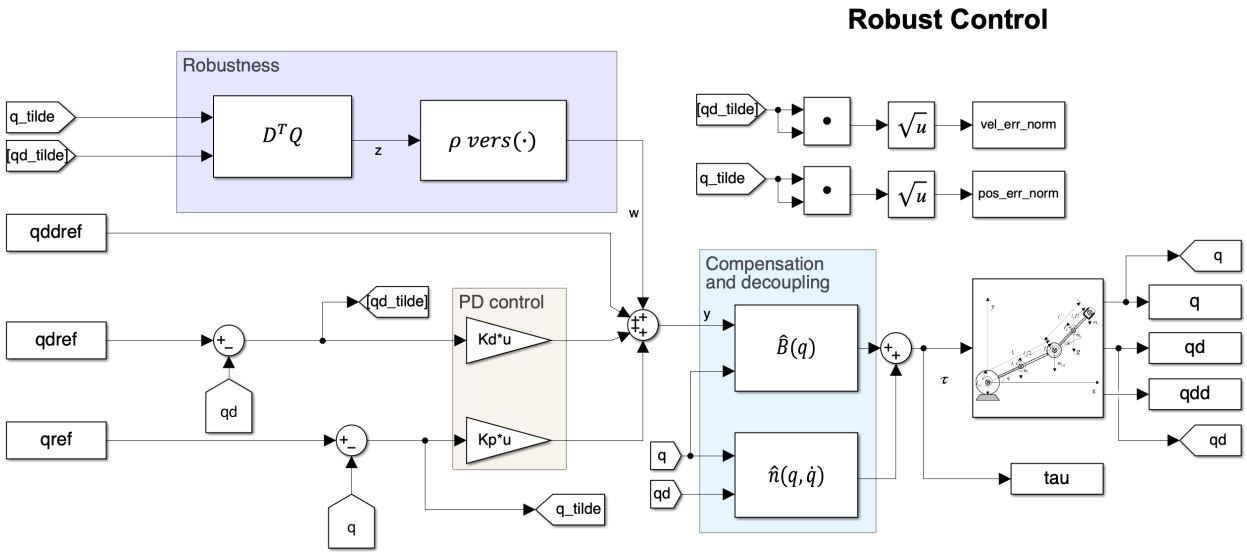
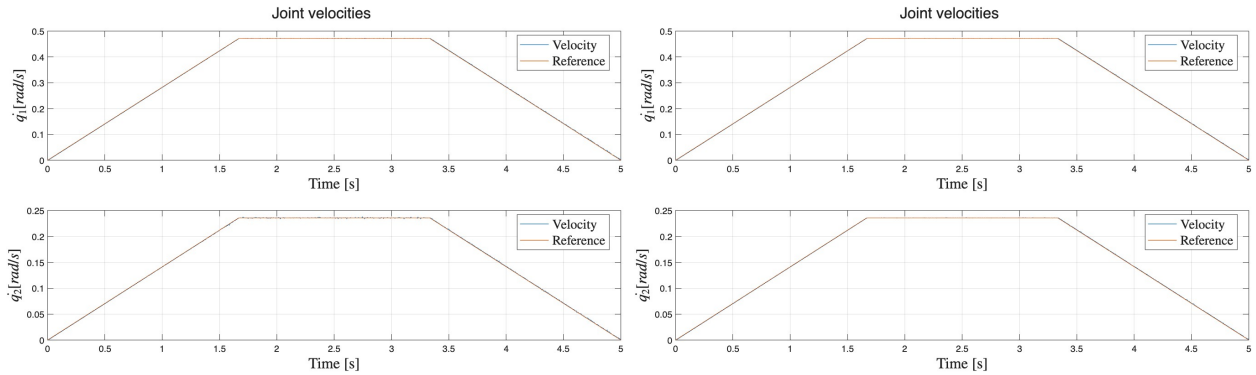


Figure 4.9: Robust control diagram at the joints

	K_P	K_D	P	ρ	ϵ
1st test	$25I_2$	$5I_2$	I_2	70	0.004
2nd test	$25I_2$	$5I_2$	I_2	70	0.01

Table 4.5: Robust control test



(a) 1st Test

(b) 2nd Test

Figure 4.10: Speed in robust control

Meanwhile, if the threshold increases, the torque takes on a more regular and smoothed trend (Fig. 4.13), but the tracking error increases slightly. With a smaller ϵ, accelerations and torques reach higher values and exhibit more oscillatory trends.

In the first test, the position trajectory is correctly tracked, as is the velocity trajectory, while the accelerations are greater than the reference ones. However, as can be seen in Figure 4.10, the velocity has a more oscillatory trend for a smaller threshold. Since no limits can be imposed on the torques, they assume higher values than in the case with MPC.

The performance in the two tests is almost equivalent.

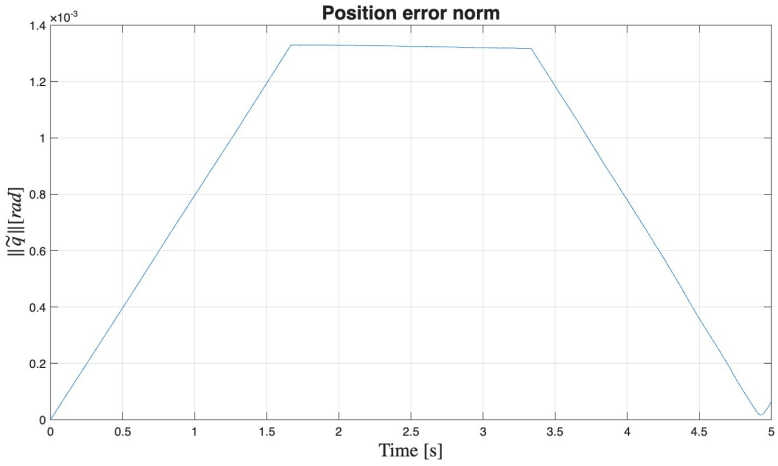


Figure 4.11: Robust control: 1st test, position error over time

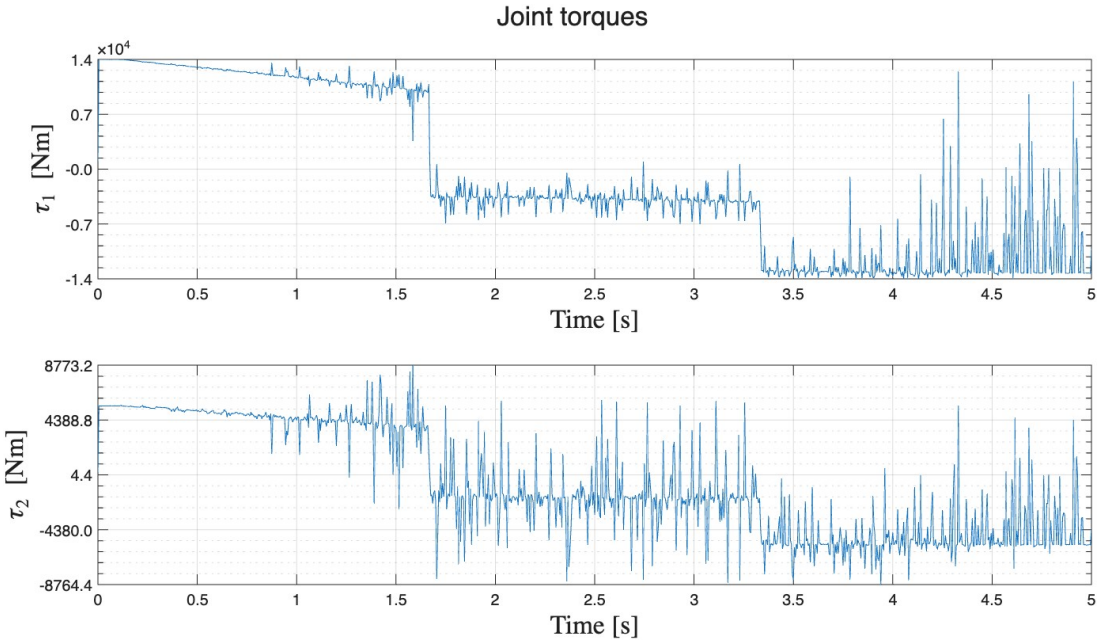


Figure 4.12: Robust control: 1st test, torque at the joints over time

As ρ decreases, the position error increases, while accelerations and torques follow a more damped trend and reach lower values, and the velocity error becomes less oscillatory.
Furthermore, the accelerations approach the reference values (Fig. 4.14); this happens because w has been reduced proportionally to ρ, which, however, cannot take on values that are too small, as it must counterbalance the level of uncertainty η.
By increasing K_P, the torques show greater and more sudden variations, but the speed error decreases. By increasing K_D, on the other hand, $\tilde{q}$ and $\dot{\tilde{q}}$ tend to increase, but the system speeds up.

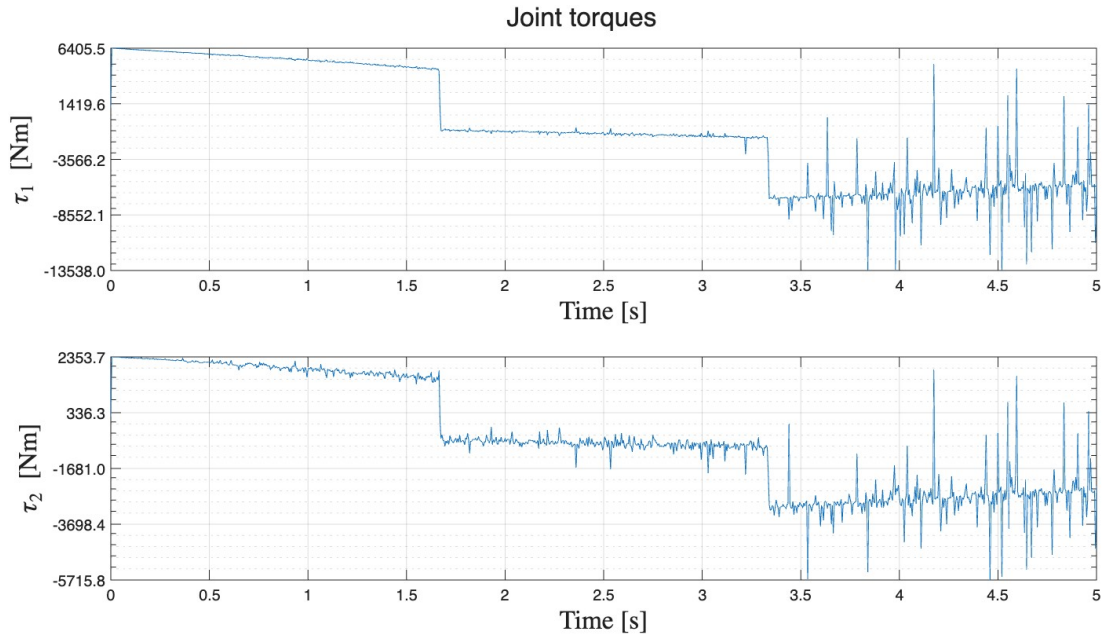


Figure 4.13: Robust control: 2nd test, torque at joints over time

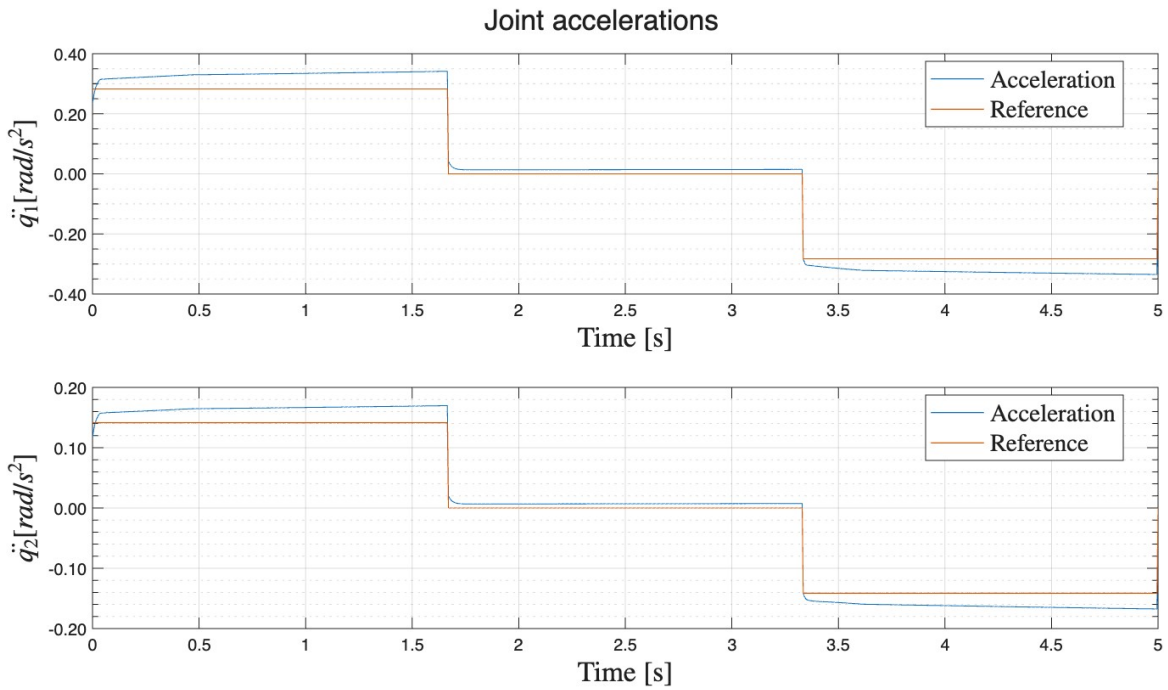


Figure 4.14: Accelerations in robust control

4.3 Adaptive Control

The following control loop was created in Simulink:

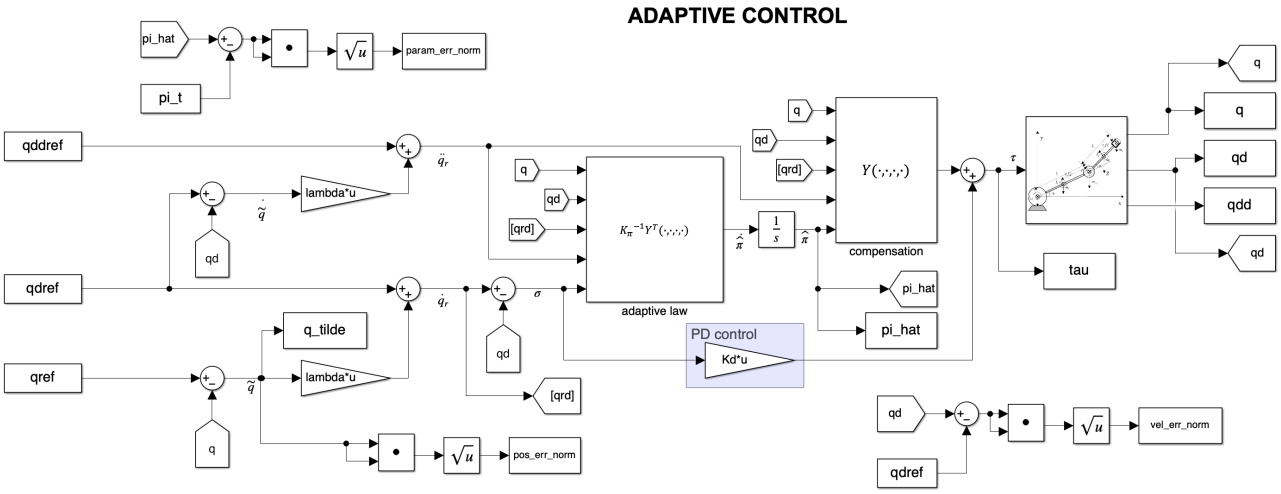


Figure 4.15: Adaptive control scheme for joints

The initial estimate of the parameter vector was set equal to that calculated with the nominal data of the manipulator. The characteristic matrices used in the tests are as follows [4]:

$$\Lambda = 5I_2; \quad K_D = 750I_2; \quad K_\pi = 0.01I_8$$

For the parametric variations, with respect to the nominal values, of the simulated system alone, the following was considered:

$$\Delta m_{l2} = 10kg; \quad \Delta m_2 l_{C2} = 11kg \cdot m; \quad \Delta \hat{I}_2 = 12.12kg \cdot m^2.$$

Thanks to the effectiveness of the adaptive action, a small tracking error can be appreciated; however, the parameters do not converge to their nominal values, as confirmed by the trend of the error norm on the parameters, which settles at a value other than zero. To improve learning, the adaptation law could be replaced with one based on a neural network, for example, capable of ensuring better performance and capturing nonlinear dynamics. In the implemented strategy, some parameters can be adapted, while others make an error that tends to decrease and then stabilizes. The position error is slightly higher than with robust control and does not faithfully follow the speed reference at each step.

The evolution of the parametric error is strongly dependent on the matrices used (K_D, K_π, Λ), the initial conditions of the estimate $\hat{\pi}(0)$, and the reference trend. In fact, if the trajectory sufficiently stimulates the system, i.e., it is a persistently exciting signal, convergence will be facilitated, as the estimator will have more information available.

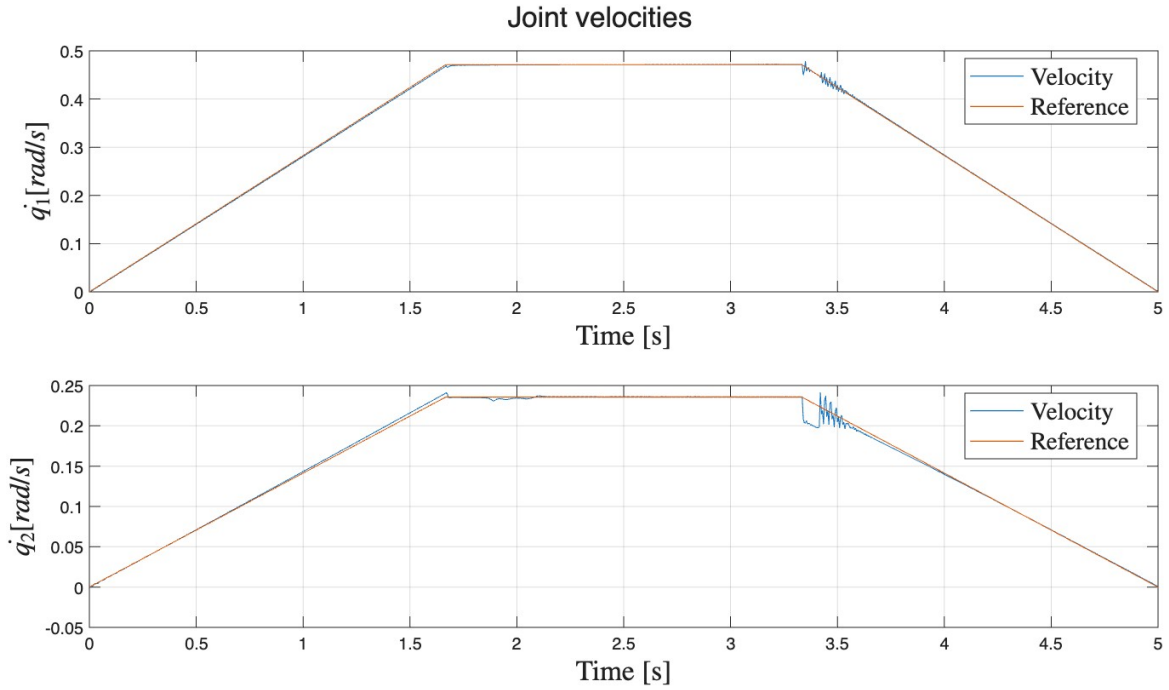


Figure 4.16: Adaptive control: speed over time

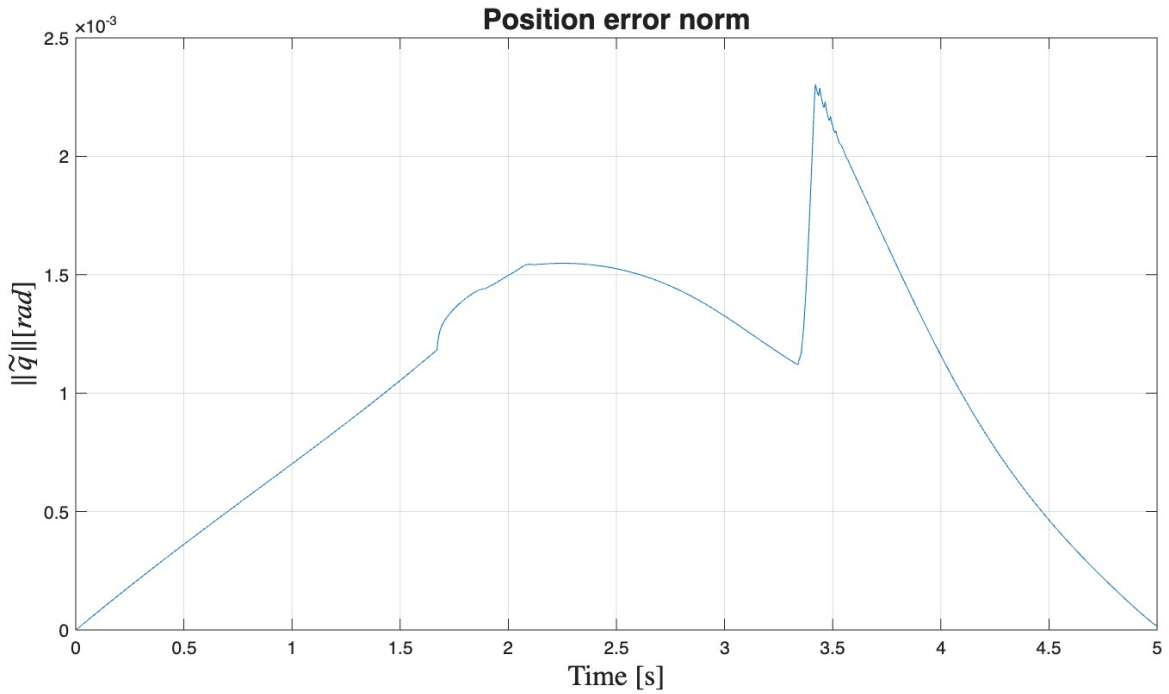


Figure 4.17: Adaptive control: position error over time

By increasing K_P, the error gradually becomes more stable and the maximum value of $||\tilde{q}||$ is reduced. The position error increases at the steps where the parametric error is amplified. By decreasing Λ, the parametric and tracking errors increase, having less weight in the calculation of σ. If, on the other hand, Λ increases, torques, velocities, and accelerations take on more

oscillatory trends. If K_π decreases, since K_π^{-1} appears in the adaptation law, $\hat{\pi}$ takes on a more oscillatory trend and the updates are more variable. Convergence is faster, but not guaranteed, since the estimates can also diverge. Conversely, if K_π increases.
The parametric error tends to decrease, then increases in the section where the speed is constant, until it starts to decrease again in the descending section. In fact, as mentioned above, the reference trajectories must be as variable as possible over time.

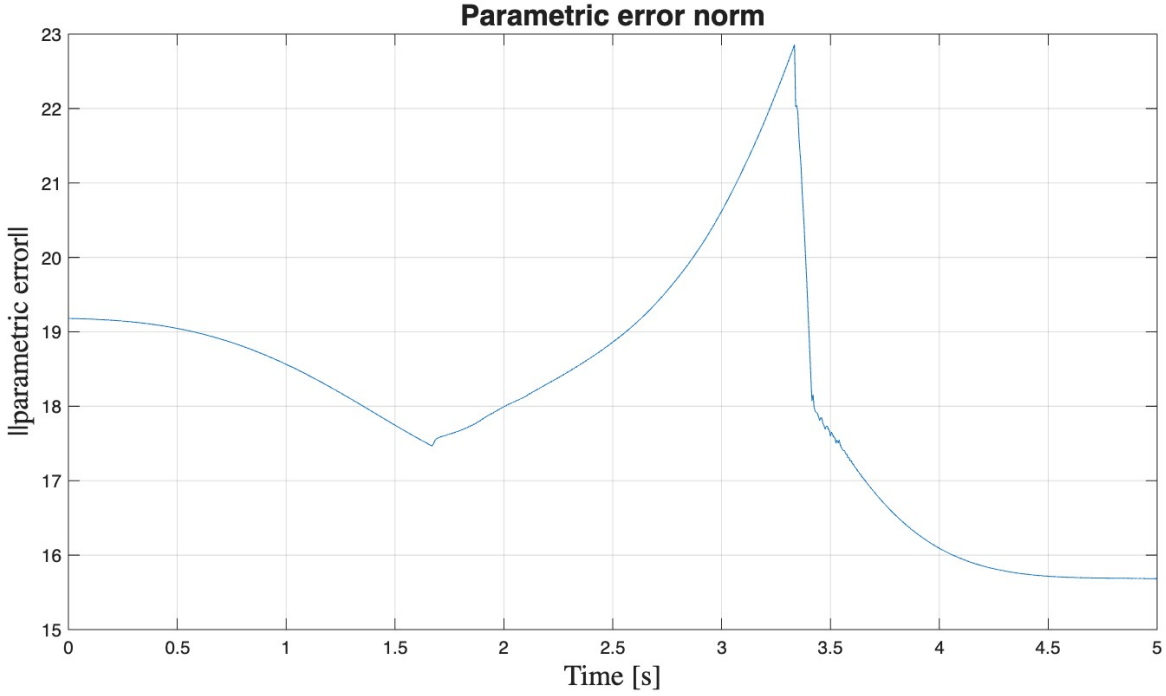


Figure 4.18: Adaptive control, trapezoidal speed profile: parametric error

With a 3rd order polynomial trajectory, the parametric error tends to decrease, except near $t = 2.5s$, where the acceleration is zero. It is also possible to estimate $\hat{\pi}(0)$ with a more variable trajectory, and then perform the simulations with the desired reference.

The initial value of the parametric error norm is approximately equal to 19, that is:

$$||\hat{\pi}(0) - \pi(0)|| = \sqrt{(\Delta m_{l2})^2 + (\Delta m_2 l_{C2})^2 + (\Delta \hat{I}_2)^2} = \sqrt{10^2 + 11^2 + 12.12^2} \approx 19.2$$

In any case, this type of control does not imply that $\hat{\pi}$ tends towards π, since the convergence of the parameters to their real values depends on the structure of the matrix Y, as well as on the desired and actual trajectories. In fact, the proposed approach aims to solve a **direct adaptive** control problem, i.e., to find a control law that guarantees limited tracking errors, so it is not aimed at determining the real parameters of the system (as in an indirect adaptive control problem).

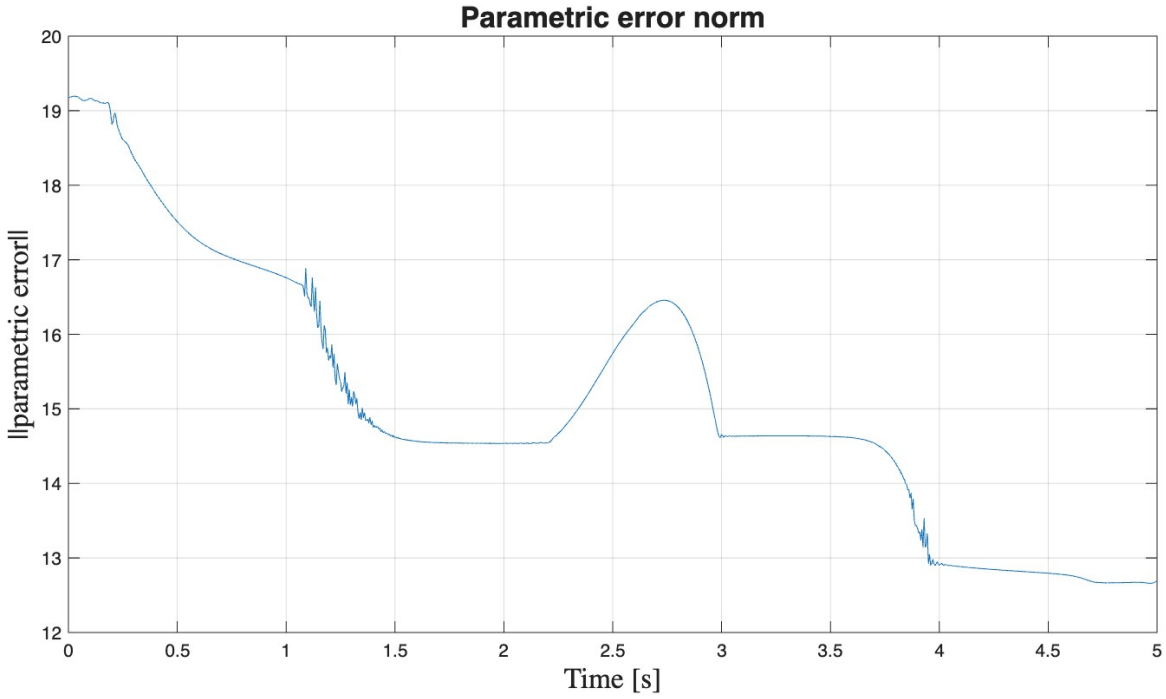


Figure 4.19: Adaptive control, cubic trajectory: parametric error

4.4 Comparisons

Following the dynamic modeling of the manipulator, it was possible to implement various centralized control algorithms. All of them guarantee the constraint on trajectory tracking, but with different performances.
MPC offers the possibility of imposing constraints, such as avoiding an obstacle and saturating control actions, which allow physically feasible experiments to be mirrored. On the other hand, the other controllers focus on controlling systems under conditions of uncertainty. The latter is the most common working condition, as it is not possible to know the model of a physical system with certainty.
Furthermore, it is not possible to implement any control algorithm due to the hardware limitations of the devices, such as available memory and sampling frequency. Consequently, if the algorithm is particularly complex and computationally demanding, robust control is chosen instead of full dynamic inversion, which could be too burdensome, for example.
Some tests were conducted to compare the performance of the various calibrated controllers, providing the same references:

The **robust** and **adaptive** controls incorporate an uncertainty component that is absent in MPC, in which, however, the presence of measurable and unmeasurable disturbances can be managed.
In general, if the trajectory is slower, i.e., if T_c and t_f increase, and/or lower speeds and accel-

	$\max\|\|\tilde{q}\|\|$	$\max\|\|\dot{\tilde{q}}\|\|$	t_{sim}
Nonlinear MPC	$10^{-1} rad$	$10^{-1} \frac{rad}{s}$	28s
Linear MPC	$2.8 \cdot 10^{-3} rad$	$1.7 \cdot 10^{-3} \frac{rad}{s}$	1s
Robust control	$1.4 \cdot 10^{-3} rad$	$4 \cdot 10^{-3} \frac{rad}{s}$	3s
Adaptive control	$2.5 \cdot 10^{-3} rad$	$3 \cdot 10^{-2} \frac{rad}{s}$	3s

Table 4.6: Comparison between controllers

erations are required, performance improves.
MPC requires longer simulations, as the computational cost is higher than that of other controllers. Furthermore, it causes a position error that is, in any case, greater than that of the others, given the presence of nonlinear constraints. To make the performance of MPC comparable to that of the other two controllers, it is necessary to linearize the system. However, in that case, nonlinear constraints can no longer be imposed, and performance depends on the proximity to the point around which the approximation is made.
To improve the adaptation of adaptive control, the LIP model, which is not always available and known with accuracy, could be replaced with an algorithm based on Radial Basis Function Neural Network (RBF-NN), for example.
Experiments were also carried out with constant position references to evaluate the effectiveness of **MPC** control based on the **passivity** property of the system. The method proved to be optimal and ensures greater robustness.
In conclusion, each method has advantages and weaknesses, so the control loop design must be selected based on the operating conditions and project objectives.

List of Figures

List of Tables

Bibliography

[1] J. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[2] M. Lemmon, "Passivity based control." Course of Nonlinear Control Systems, University of Notre Dame.

[3] P. S. G. Cisneros, A. Sridharan, and H. Werner, "Constrained predictive control of a robotic manipulator using quasi-lpv representations," *Elsevier*, 2018.

[4] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[5] P. Lino, "Kinematics - part #2." Course of Robotics - Industrial Handlings, Polytechnic of Bari, 2024.

[6] P. Lino, "Dynamics." Course of Robotics - Industrial Handlings, Polytechnic of Bari, 2024.

[7] P. Lino, "Motion control - part #1." Course of Robotics - Industrial Handlings, Polytechnic of Bari, 2024.

[8] P. Lino, "Motion control - part #2." Course of Robotics - Industrial Handlings, Polytechnic of Bari, 2024.

[9] P. Falugi, "Model predictive control: a passive scheme," *IFAC*, 2014.