DIETI UNINA UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

FOUNDATIONS OF ROBOTICS

TECHNICAL PROJECT: CONTROL OF A S.C.A.R.A MANIPULATOR

Prof.

Bruno Siciliano

Fiorella Maria Romano

P38000265

Claudia Panza

P38000266

Anno Accademico 2023/2024

Contents

1. Chapter 1. Modelling the robot

This chapter investigated the direct kinematic of a SCARA robot.

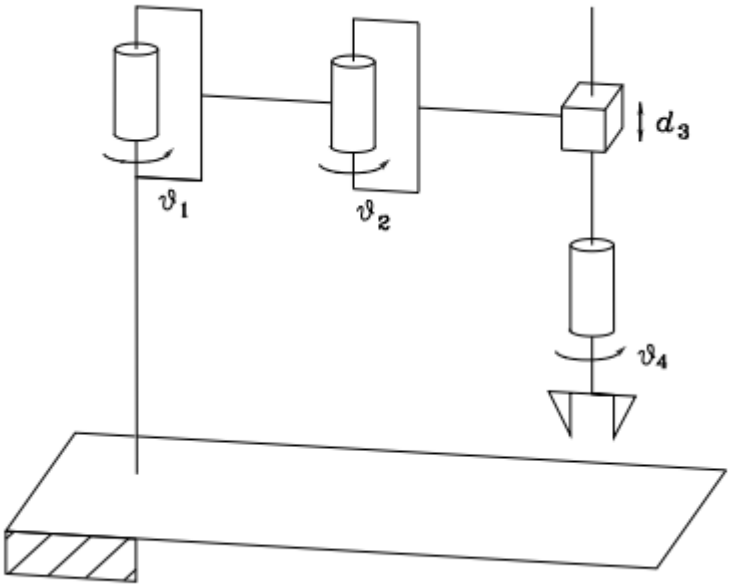


Figure 1.1: Scara Robot

S.C.A.R.A. is a special type of manipulator consisting of five links connected by four rotoidal or prismatic joints. The strength of the manipulator lies in the fact that the axes of the joints are all parallel to each other. The order in which the joints are presented is:

1. rotoidal joint

2. rotoidal joint

3. prismatic joint

4. rotoidal joint

The first step in studying the kinematics, that is the relationship that links the positions of the

joints to the placement of the end organ. The choice was made using the Denavit-Hartenberg convention.

Denavit-Hartenberg parameters were derived as follow:

LINK	a	α	d	θ
1	0.5	0	0	θ_1
2	0.5	0	0	θ_2
3	0	0	d_3	0
4	0	0	0	θ_4

Given the nature of joints, the vector of the joint variables is defined as:

$$\boldsymbol{q} = [\theta_1(t)\ \theta_2(t)\ \mathrm{d}_3(t)\ \theta_4(t)]^T$$

Remember that the pose of a rigid body with respect to a reference frame is completely described by:

• The position vector of the origin of the frame attached to the body, with respect to the reference frame.

• The components of the unit vectors of the frame attached to the body, with respect to the reference frame. Therefore, direct kinematics can be expressed by a transformation matrix such as:

$$T_e^b = \begin{bmatrix} n_e & s_e & a_e & p_e \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Such a matrix can be obtained as follows:

$$T_e^b = \boldsymbol{T_0^b}(\boldsymbol{A_1^0 A_2^1 A_3^2 A_n^{n-1}})\boldsymbol{T_e^n}$$

Where the homogeneous transformation matrices:

$$A_i^{i-1}(q_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \text{sen}(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\sin(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \text{sen}(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In the end, we obtain:

$$T_0^4(q) = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_4) & -\sin(\theta_1 + \theta_2 + \theta_4) & 0 & a_2\cos(\theta_1 + \theta_2) + a_1\cos(\theta_1) \\ \sin(\theta_1 + \theta_2 + \theta_4) & \cos(\theta_1 + \theta_2 + \theta_4) & 0 & a_2\sin(\theta_1 + \theta_2) + a_1\sin(\theta_1) \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To obtain $T_e^b(q)$, it is necessary to compute the transformation matrix that describes the position of the frame 0 with respect to the base frame T_0^b, and the transformation matrix that describes the position of the end-effector frame with respect to the last frame in this case T_e^4:

$$T_0^b = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad T_e^4 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Now it is possible to compute $T_e^b(q)$:

$$T_e^b(q) = \begin{bmatrix} -\sin(\theta_1 + \theta_2 + \theta_4) & cos(\theta_1 + \theta_2 + \theta_4) & 0 & a_2\cos(\theta_1 + \theta_2) + a_1\cos(\theta_1) \\ \cos(\theta_1 + \theta_2 + \theta_4) & \sin(\theta_1 + \theta_2 + \theta_4) & 0 & a_2\sin(\theta_1 + \theta_2) + a_1\sin(\theta_1) \\ 0 & 0 & -1 & d_0 + d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.1 Differential Kinematics

Differential kinematics allows to create a linear link between the joint velocities and the linear and angular velocities of the end-effector:

Where J(q) is the geometric Jacobian of the manipulator, which can be computed as follows:

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

which reveals that it is inherently impossible to rotate about axes x and y. So, if a four-dimensional operational space (r = 4) is of concern, then the (4 × 4) reduces version of the Jacobian is computed:

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

In this case the number of variables of the end-effector is equal to the number of joints. So, we can conclude that this manipulator is not intrinsically redundant. However, for some tasks, it may become functionally redundant. We should remember that in the case of a SCARA robot, the geometric Jacobian and the analytic Jacobian coincides

2. Chapter 2. Robot Manipulability

The differential following kinematics and statics equations, together with the duality property, allow the definition of indices for the evaluation of manipulator performance. They are algebraic tools that make it possible to assess the manipulator's ability to exercise speed and force for a given posture.

2.1 Velocity Manipulability Ellipsoid and Force Manipulability Ellipsoid

$$\boldsymbol{v}_e = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}$$

$$\boldsymbol{\tau} = \boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{\gamma_e}$$

Such indices can be helpful both for mechanical manipulator design and for determining suitable manipulator postures to execute a given task in the current configuration.

Velocity manipulability ellipsoid represents the attitude of a manipulator to arbitrarily change end effector position and orientation. In the general case of a redundant manipulator at a nonsingular configuration, the minimum-norm solution can be considered. This yields to:

$$\boldsymbol{v}_e^T(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^T(\boldsymbol{q}))^{-1}\boldsymbol{v_e} = 1$$

Which is the equation of the points on the surface of an ellipsoid in the end effector velocity space.

The *force manipulability ellipsoid* characterizes the end effector forces that can be generated with the given set of joint torques, with the manipulator in a given posture. It is given by:

$$\gamma_e^T(\boldsymbol{J}(\boldsymbol{q})\boldsymbol{J}^T(\boldsymbol{q})\gamma_{\boldsymbol{e}} = 1$$

It's important to observe that the directions of the principal axes s of the force manipulability ellipsoid coincide with those of the velocity manipulability ellipsoid, while their dimensions is inverse proportional.

2.2 Simulation

Let's consider the 2x2 Jacobian and the following configurations:

$$\boldsymbol{J}(\boldsymbol{q}) = \begin{bmatrix} -a_1\sin(\theta_1) - a_2\sin(\theta_1 + \theta_2) & -a_2\sin(\theta_1 + \theta_2) \\ a_1\cos(\theta_1) + a_2\cos(\theta_1 + \theta_2) & a_2\cos(\theta_1 + \theta_2) \end{bmatrix}$$

$$q_1 = \begin{bmatrix} \frac{\pi}{4} \\ -\frac{\pi}{2} \end{bmatrix} \quad q_2 = \begin{bmatrix} \frac{\pi}{6} \\ -\frac{\pi}{3} \end{bmatrix}$$

$$q_3 = \begin{bmatrix} \frac{\pi}{12} \\ -\frac{\pi}{6} \end{bmatrix} \quad q_4 = \begin{bmatrix} \frac{\pi}{3} \\ -\frac{2\pi}{3} \end{bmatrix}$$

$$q_5 = \begin{bmatrix} \frac{2\pi}{5} \\ -\frac{4\pi}{5} \end{bmatrix} \quad q_6 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

We are going to appreciate how the manipulability changes with different robot postures. The manipulability analysis helps us understand how the SCARA robot transforms joint velocities and torques into end-effector velocities and forces.

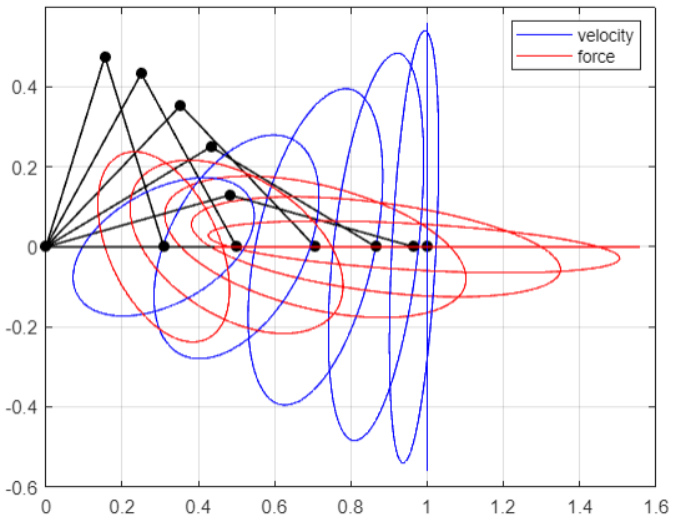


Figure 2.1: Manipulability ellipsoids

As said before, the principal axes of the ellipsoids are the same, but the dimension between the two ellipsoids are inversely proportional. Indeed, a big velocity transformation ratio corresponds to a small force transformation ratio. It can be observed that getting close to the singularity the ellipsoid tends to collapse to a line.

Manipulability measures for the previous configurations have been computed.

$$w_1 = 0.2500 \quad w_2 = 0.2165 \quad w_3 = 0.1250 \quad w_4 = 0.2165 \quad w_5 = 0.1469 \quad w_6 = 0$$

3. Chapter 3. Trajectory Planning

This section focuses on defining the robot's trajectory, which is composed of 20 distinct points. The motion between two consecutive points can either follow a straight line or a circular arc, extending across different planes in three-dimensional space. Out of these points, four are identified as via points, meaning the trajectory will pass near them without directly crossing through. The entire trajectory is designed to be completed within a total duration of 46 seconds.

Point	**x**	**y**	**z**	ϕ
p1	-0.4	-0.1	0	0
p2	-0.6	0.1	0	$\pi/6$
p3	-0.5	0.2	0.1	$\pi/4$
p4	-0.3	0.2	0.1	$\pi/4$
p5	-0.2	0.2	0.15	$\pi/4$
p6	-0.1	0.3	0.2	$\pi/4$
p7	0.0	0.3	0.2	$\pi/11$
p8	0.1	0.4	0.3	$\pi/2$
p9	0.2	0.4	0.35	$\pi/2$
p10	0.3	0.45	0.4	$\pi/2$
p11	0.3	0.4	0.3	$3\pi/7$
p12	0.3	0.2	0.25	$3\pi/10$
p13	0.4	0.15	0.4	$\pi/4$
p14	0.3	0.1	0.1	$\pi/4$
p15	0.3	-0.2	0.1	$\pi/4$
p16	0.3	-0.2	0.1	$\pi/4$
p17	-0.1	-0.3	-0.2	$2\pi/9$
p18	-0.2	-0.1	-0.1	$\pi/8$
p19	-0.3	-0.1	-0.1	0
p20	-0.3	-0.2	-0.1	0

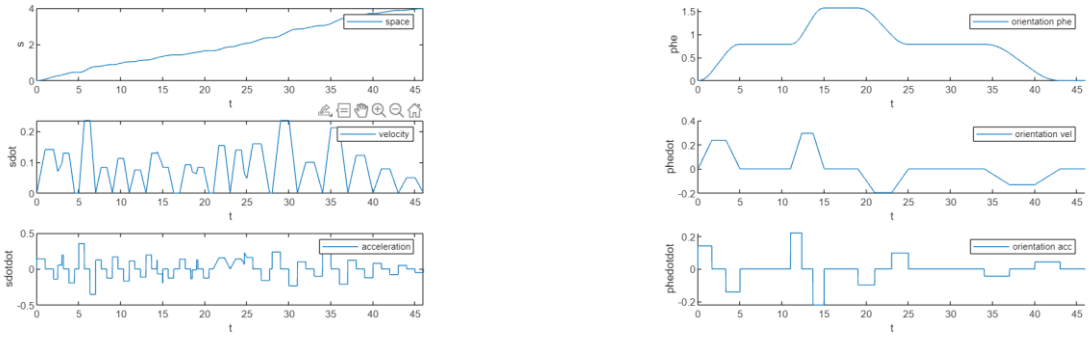


Figure 3.1: s, sdot, sdotdot and phe, phedot, phedotdot performance

At this stage, we can determine the desired trajectory in the operational space, along with its first and second derivatives, by applying the following equations based on a trapezoidal velocity profile:

$$x_d(t) = P(s(t))$$
$$\dot{x}_d(t) = \frac{dP(s(t))}{ds}\dot{s}(t)$$
$$\ddot{x}_d(t) = \frac{d^2P(t)}{ds^2}\dot{s}(t)$$

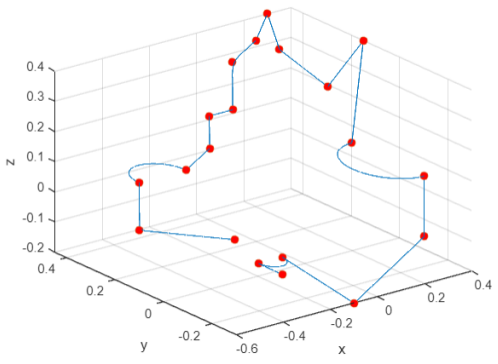


Figure3.2: 3D end effector trajectory

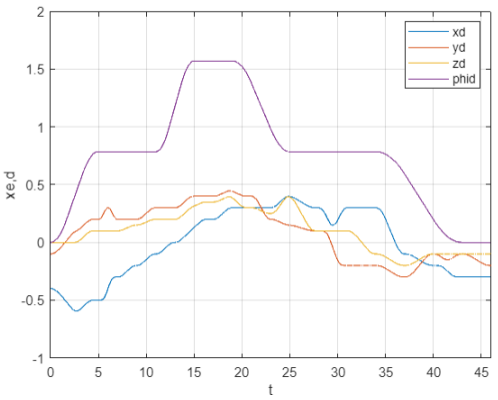


Figure 3.3: xe,d

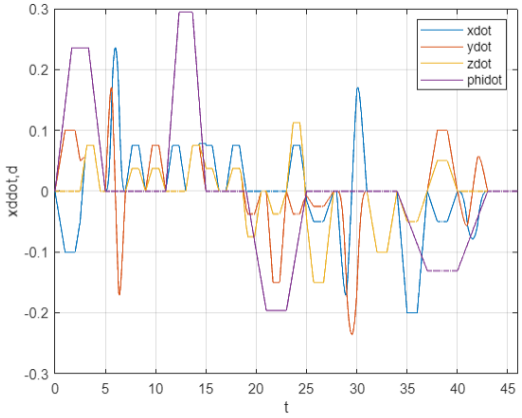


Figure 3.4: xdot,d

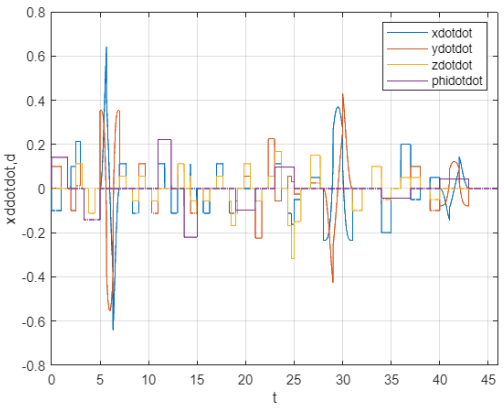


Figure 3.5: xdotdot

4. Chapter 4. Inverse Kinematics

Inverse kinematics problems can be highly complex and labor-intensive to solve for robotic manipulators with intricate structures. The non-linear relationship between the operational space and joint positions makes it challenging to find closed-form solutions. Consequently, it becomes necessary to use the **differential kinematic equation**, which employs the Jacobian matrix to create a linear mapping between joint velocities and velocities in the operational space.

By knowing the joint velocities, the desired joint positions can be determined through numerical integration techniques such as Euler integration. However, these methods may introduce drift over time, which can be minimized using solution schemes that account for the error between the actual and desired poses in the operational space. Such methods are the CLIK (Closed-Loop Inverse Kinematics) algorithms, which can be classified as follows:

- CLIK algorithm based on the inverse of the Jacobian.
- CLIK algorithm based on the transposition of the Jacobian.
- Second-order CLIK algorithm.

All these algorithms ensure that the error converges to zero in a stable manner.

$$e = xd - xe$$

To ensure e(0)=0 and improve the algorithm's performance, the initial manipulator position q0 was calculated using the **CLIK algorithm based on the inverse Jacobian**, with xd=p(0) and xddot=0 as constant references. The integrations were performed using the **forward Euler method** with a sampling time of 1 ms.

4.1 Jacobian Inverse Algorithm

The first algorithm to be analyzed is the Jacobian inverse algorithm, which enables the system to be linearized via feedback linearization. However, this method is computationally intensive. To apply this algorithm, we must assume that the manipulator is non-redundant, meaning there are no extra degrees of freedom beyond what is necessary to achieve the desired motion. Under this assumption, we can suppose:

$$\dot{\mathbf{q}} = \mathbf{J}_A^{-1}(\mathbf{q})(\mathbf{x}_d + \mathbf{K}\mathbf{e})$$

With this choice, the equation governing the error dynamics reduces to:

$$\dot{e} + \mathbf{K}\mathbf{e} = \mathbf{0}$$

It's easy to observe that the error converges to 0 if K is a positive definite Matrix. The matrix K is chosen as:

$$K = \begin{bmatrix} 1000 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 \\ 0 & 0 & 700 & 0 \\ 0 & 0 & 0 & 500 \end{bmatrix}$$

4.1.1 Simulation

The conceptual scheme of CLIK with inverse Jacobian is shown below, followed by the implementation in Simulink.

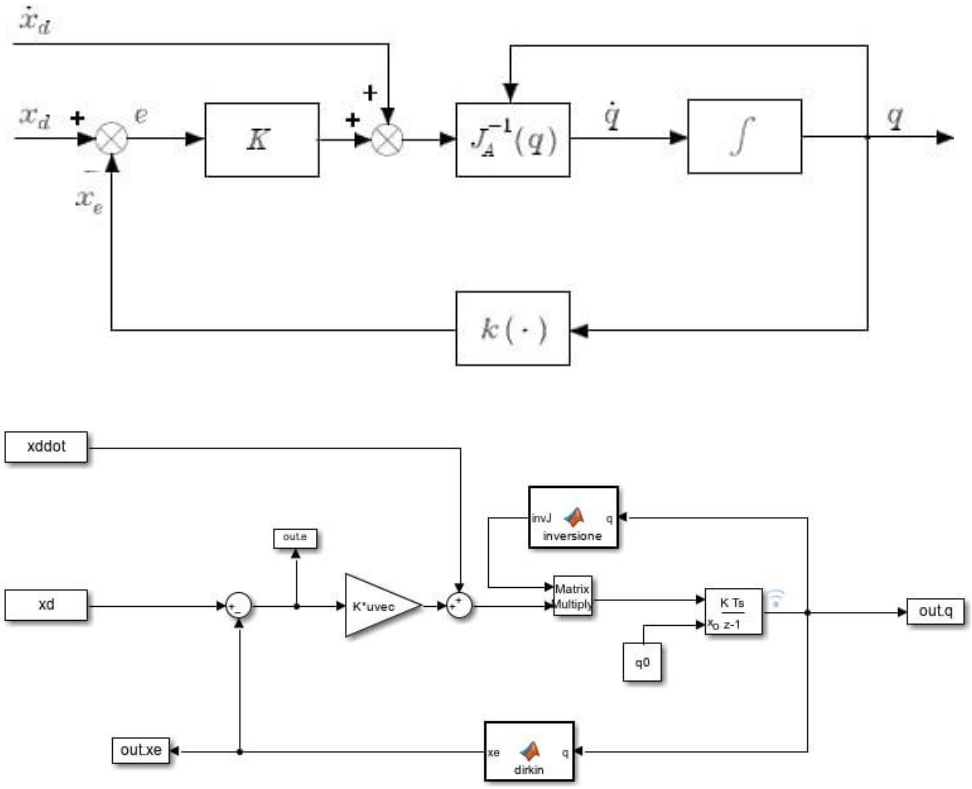


Figure 4.1: Simulink implementation of the CLICK algorithm with Jacobian inverse

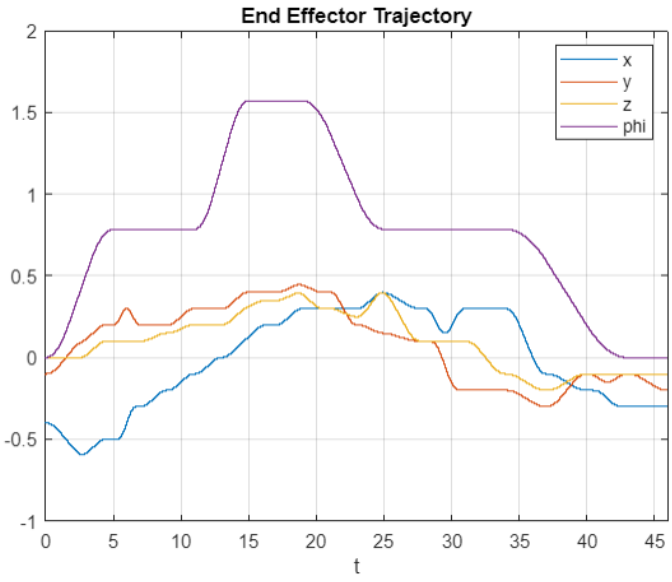


Figure 4.2: operational space variables with Jacobian inverse algorithm

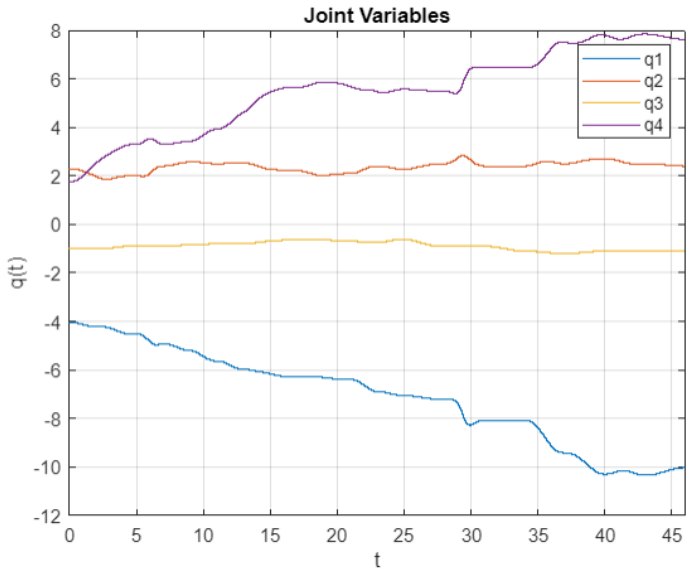


Figure 4.3: joint variables with Jacobian inverse algorithm

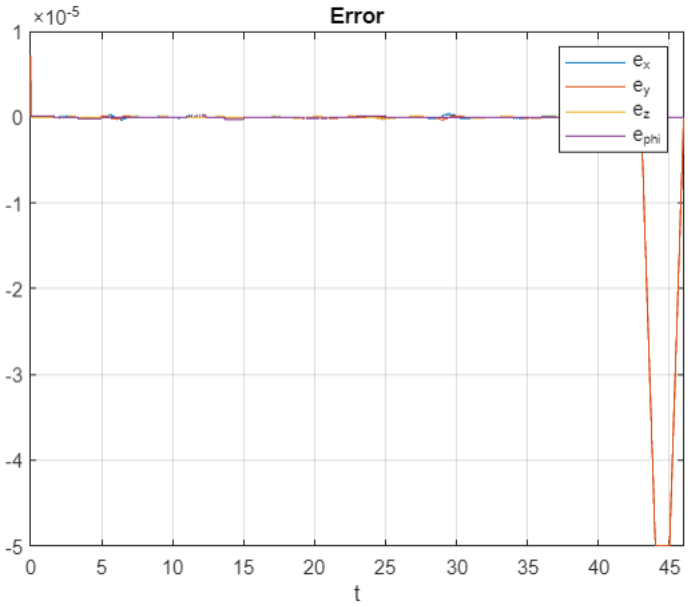


Figure 4.4: position and orientation error with Jacobian inverse algorithm

4.2 Jacobian Transpose Algorithm

The algorithm based on the **Jacobian transpose** is computationally simpler than the Jacobian inverse algorithm, as it establishes a relationship between *qdot* and *e* that ensures the error converges to zero without needing explicit linearization. The error dynamics are governed by a *non-linear differential equation*. By applying *Lyapunov's method*, it becomes straightforward to determine a relationship between *qdot* and e that guarantees the **asymptotic stability** of the error. In this case, we can suppose:

$$\dot{q} = J_A^T(q)Ke$$

With K chosen as such:

$$K = \begin{bmatrix} 550 & 0 & 0 & 0 \\ 0 & 800 & 0 & 0 \\ 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 200 \end{bmatrix}$$

We can observe that despite having a lower computational complexity compared to others, the Jacobian transpose algorithm produces less accurate predictions than those involving the computation of the Jacobian inverse.

4.2.1 Simulation

The conceptual scheme of CLIK with inverse Jacobian is shown below, followed by the implementation in Simulink.

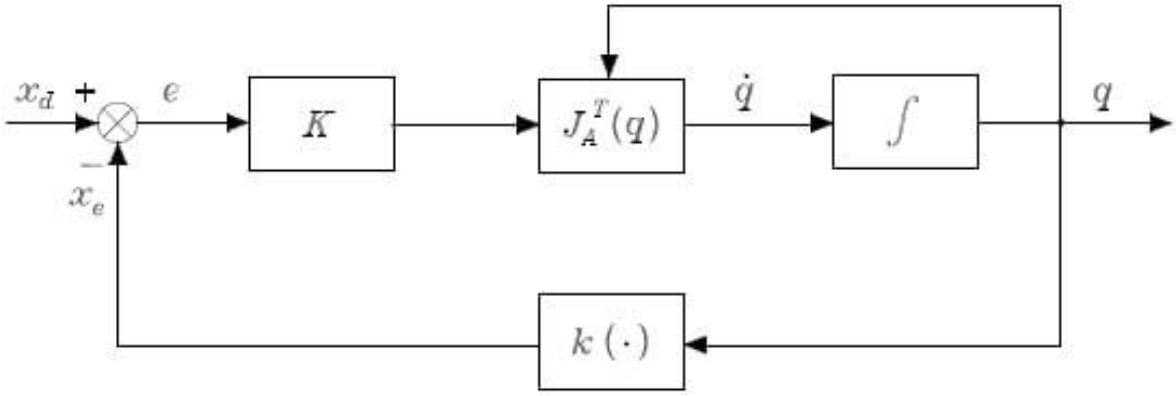


Figure 4.5: Conceptual Transpose scheme

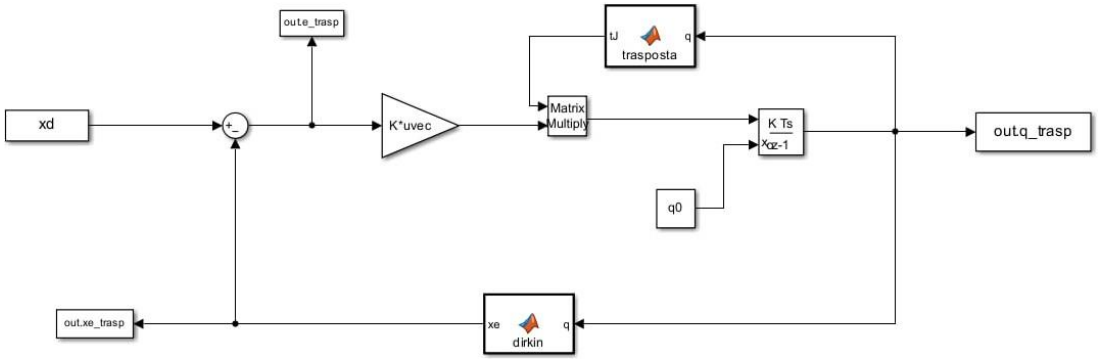


Figure 4.6: Simulink Transpose scheme

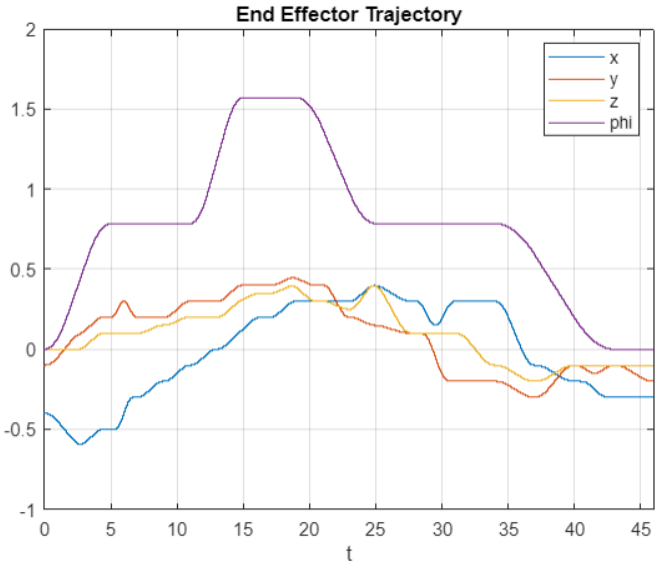


Figure 4.7: operational space variables with Jacobian transpose algorithm

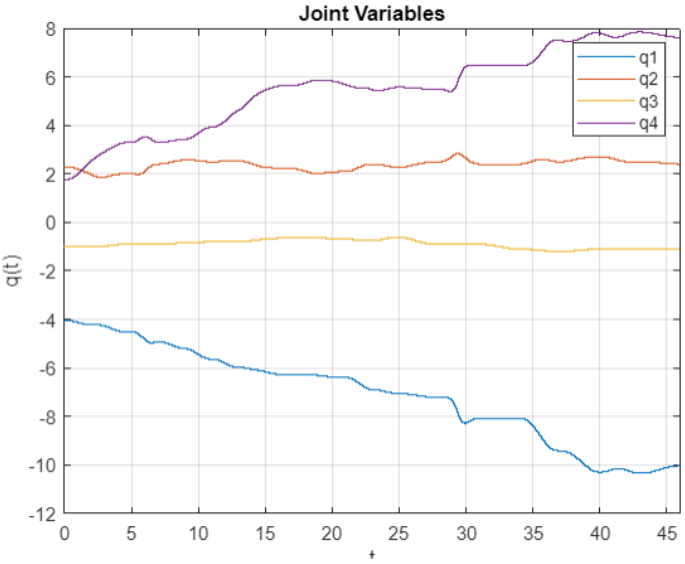


Figure 4.8: joint variables with Jacobian transpose algorithm

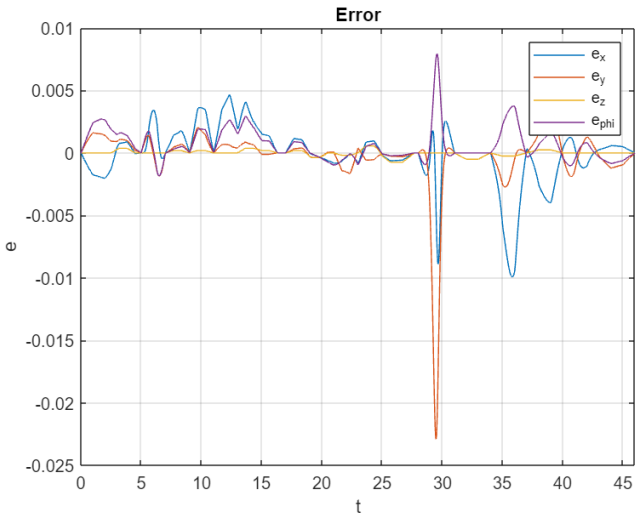


Figure 4.9: position and orientation error with Jacobian transpose algorithm

4.3 Jacobian Pseudo-Inverse Algorithm

To test the performance of the CLIK algorithm using the pseudo-inverse of the Jacobian, the SCARA robot was treated as a functionally redundant system by loosening one of the operational space constraints, specifically relaxing the y-axis. In the case of redundant manipulators (where the number of joints r is less than the number of operational space dimensions n), the Jacobian matrix J is not square, making it impossible to directly invert. As a result, the pseudo-inverse of the Jacobian is used in such scenarios.

The pseudo-inverse, which provides a solution for non-square matrices, is applied in place of the Jacobian inverse, allowing the system to compute joint velocities. This approach ensures the robot can still follow the desired trajectory despite the redundancy.

4.3.1 Simulation

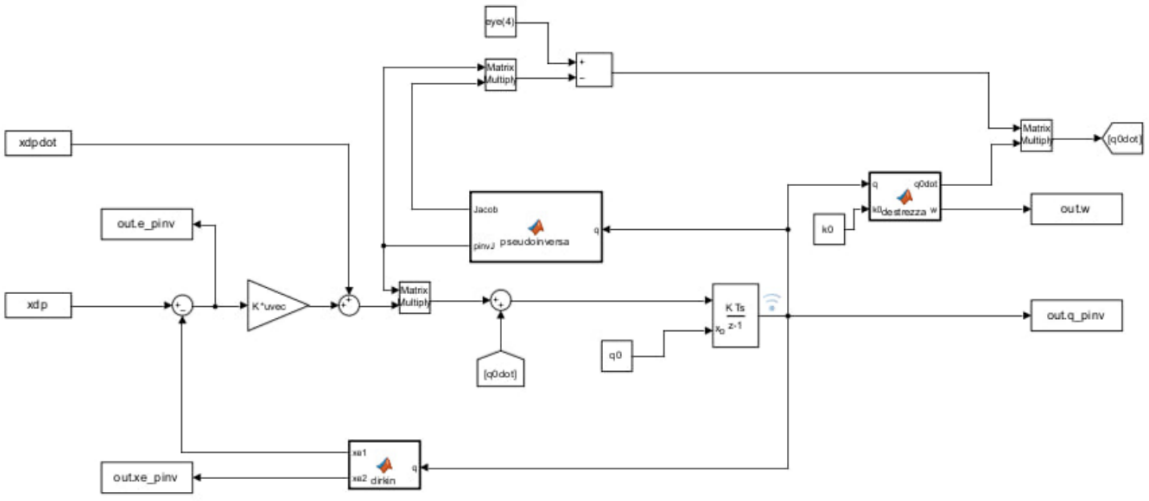


Figure 4.10: Simulink Pseudo-inverse scheme

The choise made in this case is:

$$\mathbf{q} = \mathbf{J}_A^{\dagger}(\mathbf{q})(\mathbf{x}_d + \mathbf{K}\mathbf{e}) + (\mathbf{I}_n - \mathbf{J}_A^{\dagger}\mathbf{J}_A)\mathbf{q}_0$$

With regard to the second term of this expression it represents a projector in the null space ofthe Jacobian.

K is chosen as:

$$K = \begin{bmatrix} 70 & 0 & 0 \\ 0 & 70 & 0 \\ 0 & 0 & 70 \end{bmatrix}$$

The use of the pseudo-inverse is crucial because it enables us to leverage kinematic redundancy to assign a secondary task to the robotic system. Specifically, we aim to maximize the manipulability measure, which is an index that indicates how far the robot is from kinematic singularities. This measure helps ensure that the robot operates in configurations where it has better control and flexibility. For the initial joint configuration q0 we have:

$$\mathbf{q}_0 = k_0 \left(\frac{\partial \mathbf{w}(\mathbf{q})}{\partial \mathbf{q}} \right)$$

Where

$$k_0 = 5 \quad \text{and} \quad \mathbf{w}(\mathbf{q}) = \sin^2(\theta_2)$$

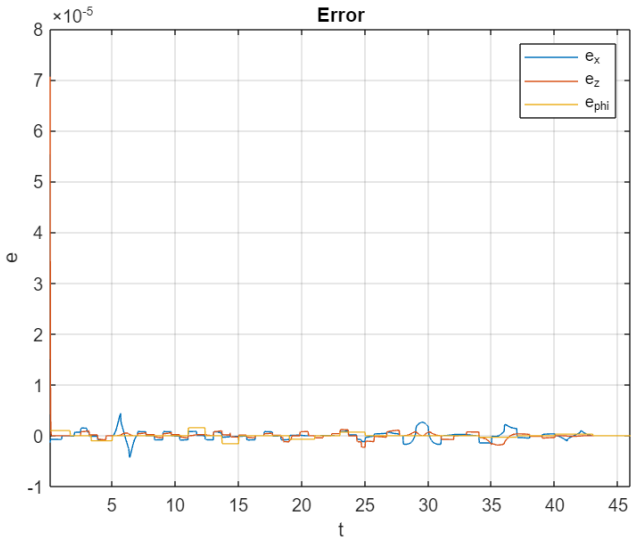


Figure 4.11: position error with Jacobian Pseudo-Inverse algorithm

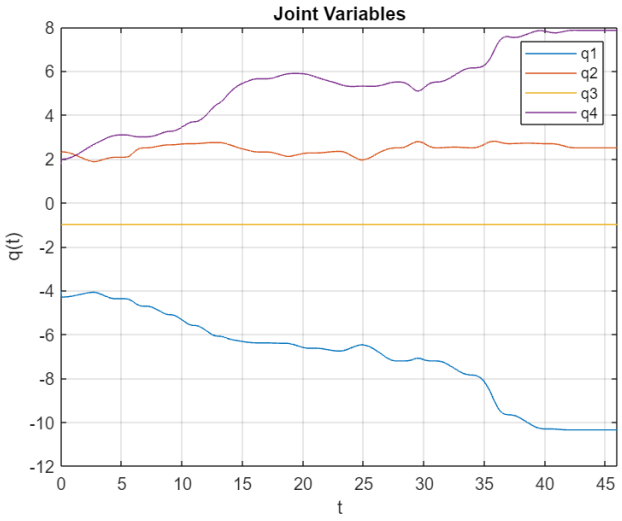


Figure 4.12: variable in the joint space

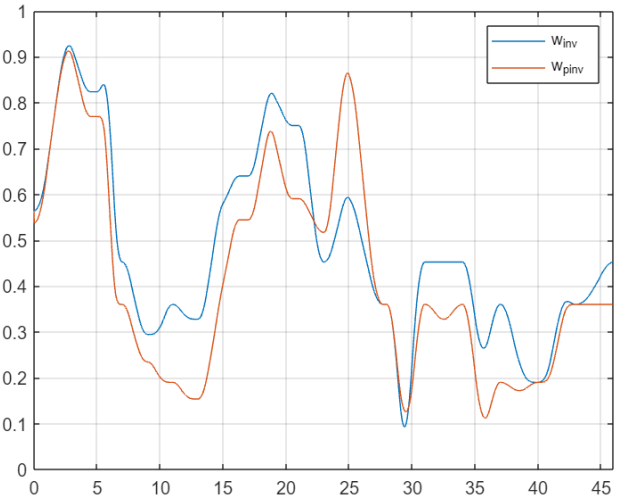


Figure 4.13: manipulability measure

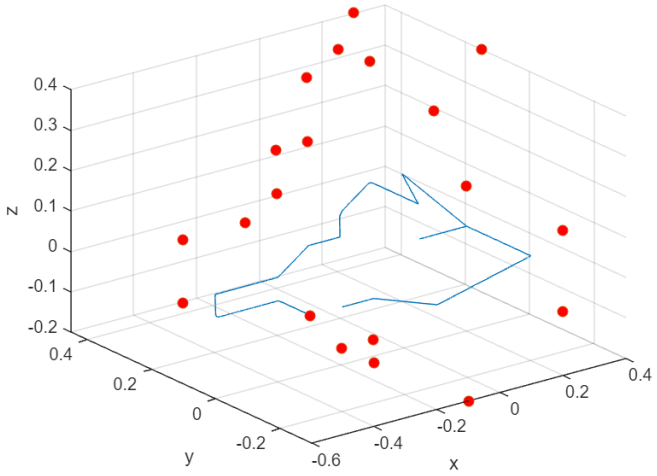


Figure 4.14: 3D end-effector trajectory

It can be immediately noted that the trajectory does not pass through all the chosen work points. We expected this having relaxed the component concerning the y-axis of the operating space. To evaluate the effectiveness of the solution we must therefore compare the temporal trend of the dexterity measure with the one obtained from the CLIK scheme with the inverse of the Jacobian.

In the case of redundant manipulator, the manipulability index is not always better because the optimization of this index has lower priority compared to the trajectory constraint. Probably, by relaxing the *y*-axis and moving primarily in the *xz*-plane, the manipulator is often required to pass closer to singularities. The system primarily focuses on achieving the desired motion in the operational space, and only secondarily tries to optimize the manipulability through redundancy. As a result, the manipulability index may not always be maximized due to the need to balance these competing objectives.

4.4 Second order Algorithm

When performing torque control, it is frequently required to reverse the motion path with respect to position, velocity, and acceleration. As a result, second-order kinematic inversion algorithms have been developed. The block diagram of the second-order algorithm is shown in the figure below:

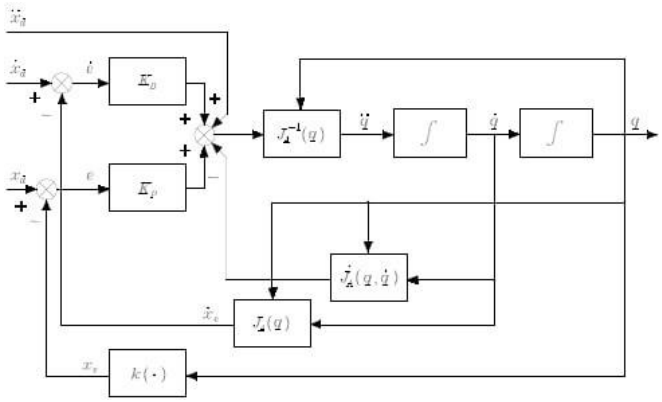


Figure 4.15: second order CLIK scheme

A second-order solution for kinematic inversion involves the following choice for the vectorof accelerations at the joints:

$$\mathbf{q} = \mathbf{J}_A^{-1}(\mathbf{q})(\mathbf{x}_d + \mathbf{K}_d\mathbf{e} + \mathbf{K}_p\mathbf{e} + \mathbf{J}_A(\mathbf{q}, \mathbf{q})\mathbf{q})$$

4.4.1 4.4.1 Simulation

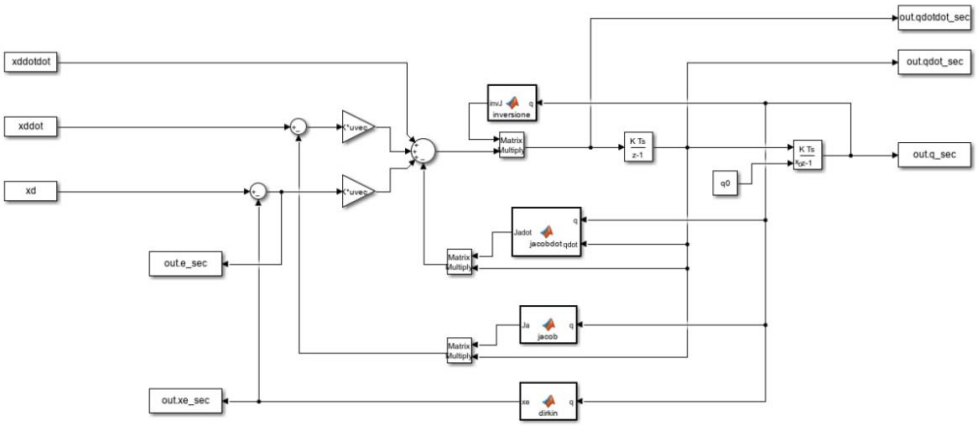


Figure 4.16: Simulink 2-order scheme

The error dynamics is described by:

$$\ddot{\mathbf{e}} = \mathbf{K_D}\dot{\mathbf{e}} + \mathbf{K_P}\mathbf{e} + \mathbf{0}$$

Where $\mathbf{K_D}$ and $\mathbf{K_P}$ have been chosen as follows:

$$K_D = \begin{bmatrix} 3500 & 0 & 0 & 0 \\ 0 & 3000 & 0 & 0 \\ 0 & 0 & 3000 & 0 \\ 0 & 0 & 0 & 3500 \end{bmatrix}$$

$$K_P = \begin{bmatrix} 70 & 0 & 0 & 0 \\ 0 & 70 & 0 & 0 \\ 0 & 0 & 70 & 0 \\ 0 & 0 & 0 & 70 \end{bmatrix}$$

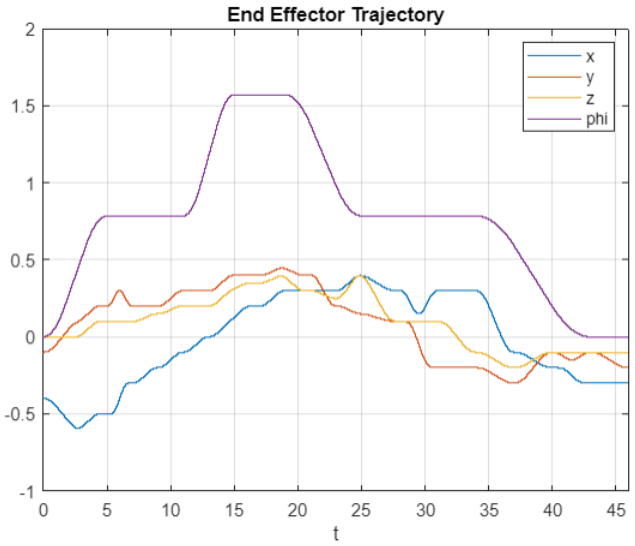


Figure 4.17: Position and Orientation in the Operational space

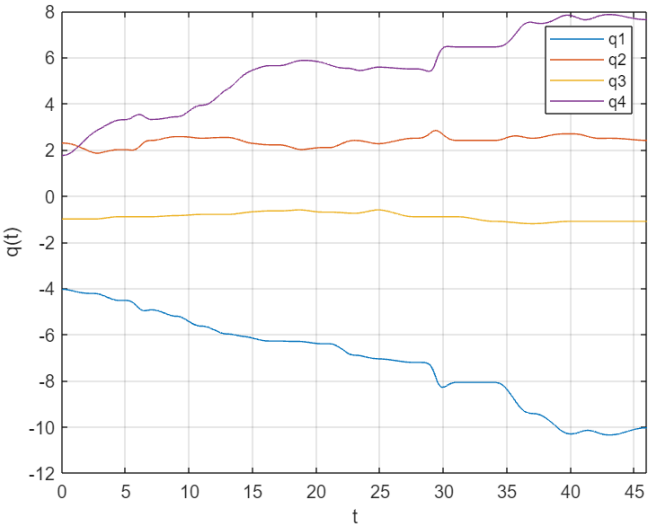


Figure 4.18: Variables in the Joint space

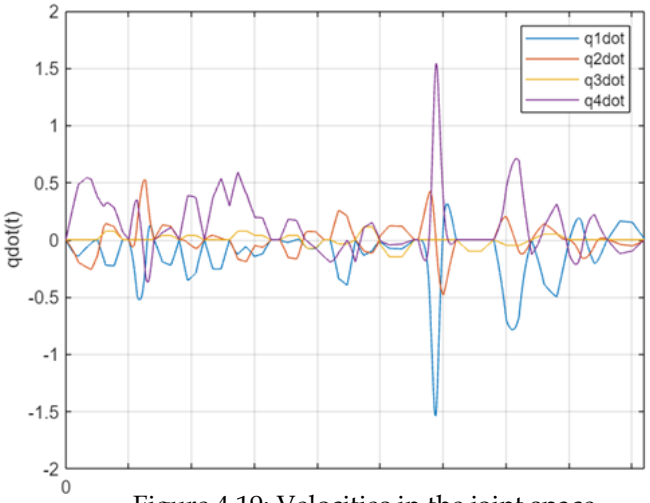


Figure 4.19: Velocities in the joint space

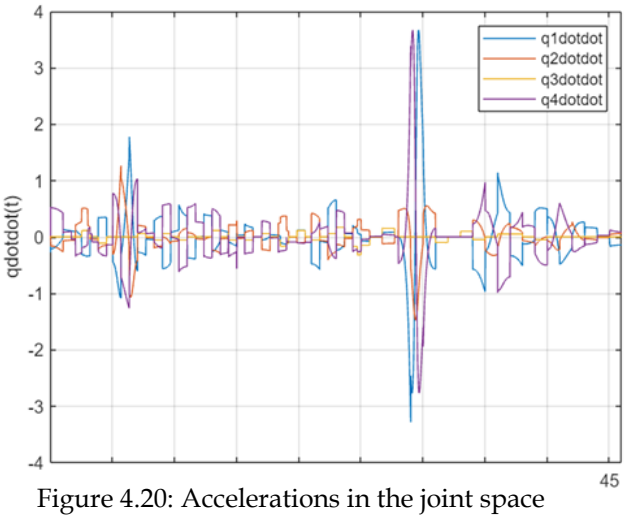


Figure 4.20: Accelerations in the joint space

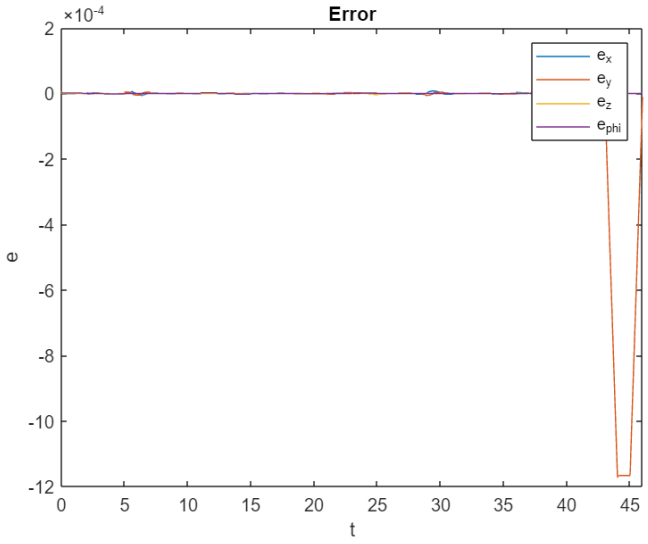


Figure 4.21: Position and Orientation Error with Click of second Order

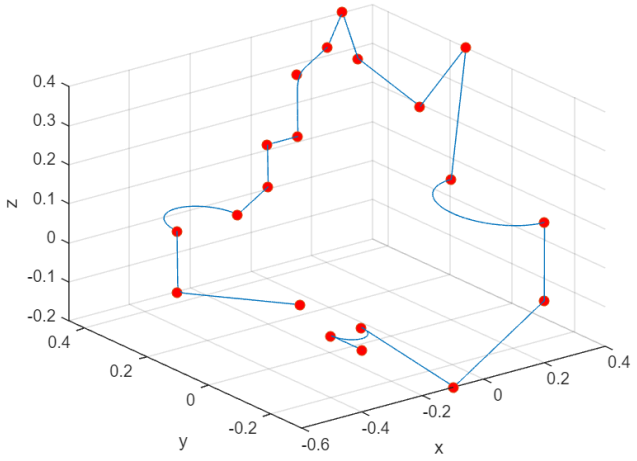


Figure 4.22: 3D end-effector trajectory

5. Chapter 5. Motion Control

This chapter analyzes various control strategies in both joint space and operational space.

In particular, we will see:

- Robust Control
- Adaptive Control
- An operational space controller with the addition of an integral action to recover the steady-state error due to payload

The controllers implemented in joint space are classified as inverse dynamics controllers, which aim to accurately compensate for nonlinearities and coupling effects in the model through a centralized action, enabling the system to be controlled as a decoupled linear system using a PD action. It is important to note that the reference trajectory is described in operational space. Therefore, in the joint space algorithm, it is necessary to invert the kinematics using a second-order kinematic algorithm that we have seen in previous chapters.

First of all, we have to describe the dynamic model of an n-joints manipulator.

$$B(q)\ddot{q} + n(q,\dot{q}) = u$$

where:

$$n(q,\dot{q}) = C(q,\dot{q}) + F\dot{q} + g(q)$$

If u is selected as follows:

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

since B(q) is invertible, the system is equivalent to:

$$\ddot{q} = y$$

The control law y has been chosen to stabilize the feed-back system is:

$$y = -K_p\tilde{q} - K_d\dot{\tilde{q}} + \ddot{q}_d$$

It leads to the system of second-order equations:

$$K_p\tilde{q} + K_d\dot{\tilde{q}} + \ddot{\tilde{q}} = 0$$

which, under the assumption of matrices KP e KD defined as positive, is asymptotically stable. The general control scheme of a controller based on inverse dynamics is described:

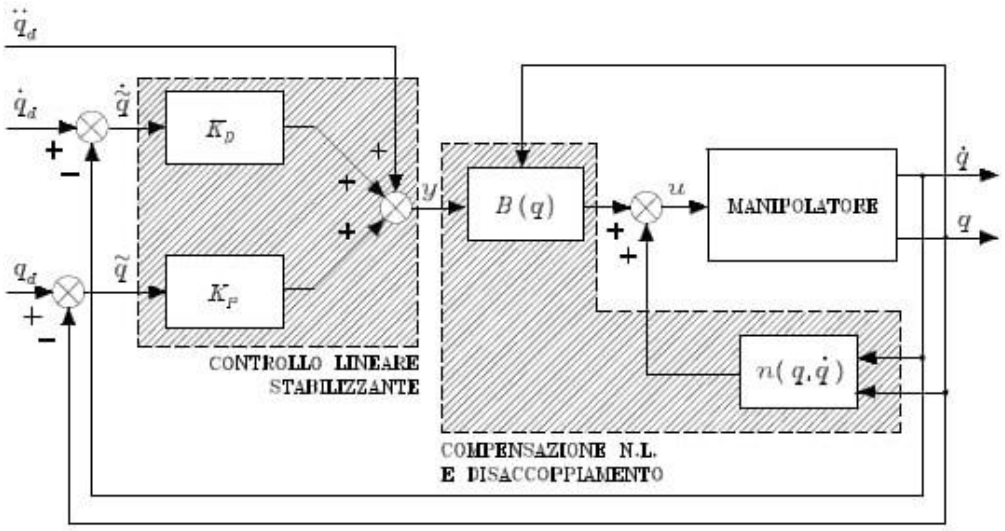


figure 5.1 general control scheme of a controller based on inverse dynamics

Control laws based on inverse dynamics calculation have limitations. Indeed, they are based on the assumption of perfect cancellation. In order for them to be perfectly cancelled, the dynamic model must be perfectly known, and they must be calculated in real time. To counter these uncertainties, robust and adaptive control techniques are used.

5.1 Robust Control

In the case of imperfect compensation, the control input on which we can act is the vector of pairs. Let us assume that we provide such a control input:

$$\tau = u = \hat{B}(q)y + \hat{n}(q,\dot{q})$$

The two terms $\hat{B}(q)$ and $\hat{n}(q,\dot{q})$ represent estimates of the model terms. We choose to use only the terms on the diagonal of the inertia

matrix for terms $\hat{B}$ and neglect the centrifugal and Coriolis terms for $\hat{n}$.

In turn, the y is chosen with a feedforward and a feedback action, like this:

$$y = -K_p\tilde{q} - K_d\dot{\tilde{q}} + \ddot{q}_d + w$$

As far as the choice of w is concerned, we have not directly adopted the classical unit vector law, but have set a boundary layer to avoid the problem of chattering:

$$w = \begin{cases} \rho\dfrac{z}{||z||}, & ||z|| \geq \epsilon \\ \rho\dfrac{z}{\epsilon}, & ||z| < \epsilon \end{cases}$$

The introduction of the threshold ϵ is crucial because, once the sliding space is reached, further splitting becomes impossible. Without this threshold, the resulting control law would oscillate at an infinite frequency, causing the chattering phenomenon to occur.

The parameters we have chosen are:

$$\rho = 3\ \epsilon = 0.01$$

$$K_P = \begin{bmatrix} 1500 & 0 & 0 & 0 \\ 0 & 1500 & 0 & 0 \\ 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 4500 \end{bmatrix} K_D = \begin{bmatrix} 100 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 & 0 & 1000 \\ 1000 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The matrix Q is structured to give a relatively lower weight to the velocity error compared to the position error. In all the aforementioned matrices, a more aggressive control strategy was specifically applied to the vertical position of the end-effector. This focused control is necessary due to the unavoidable impact of gravitational forces from the payload. In conclusion, while this control strategy does not fully compensate for the presence of the payload at the tip, with a proper selection of gains, the errors are still kept minimal.

5.1.1 Simulation

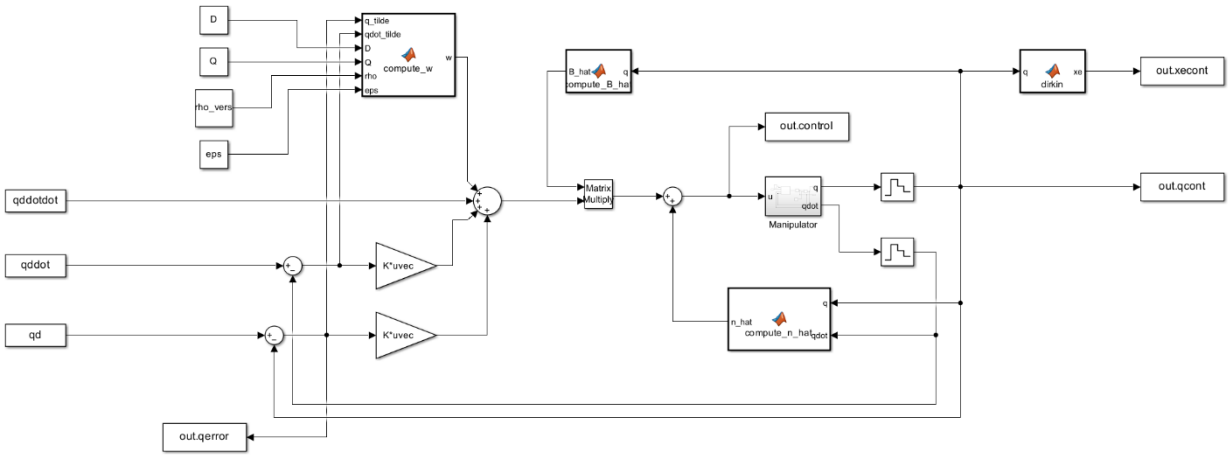


Figure 5.2: Simulink Robust control scheme

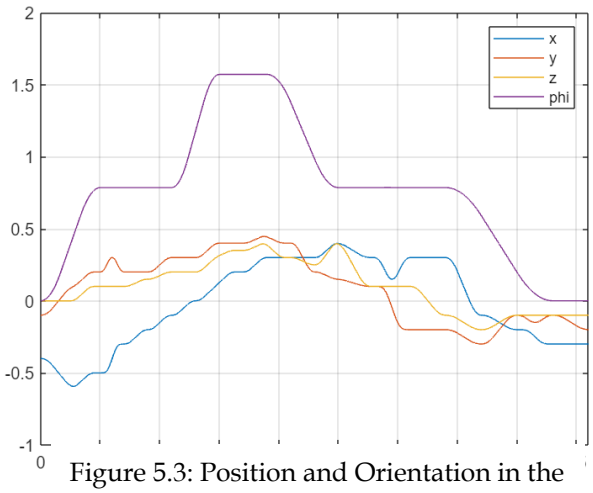


Figure 5.3: Position and Orientation in the Operational space

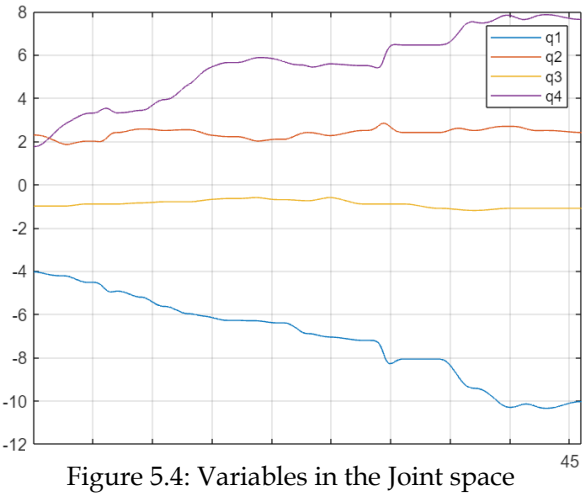


Figure 5.4: Variables in the Joint space

It's also possible to see that by decreasing the value of epsilon, theeffects of chattering are perfectly visible on the pair. Comparison results will be shown below:

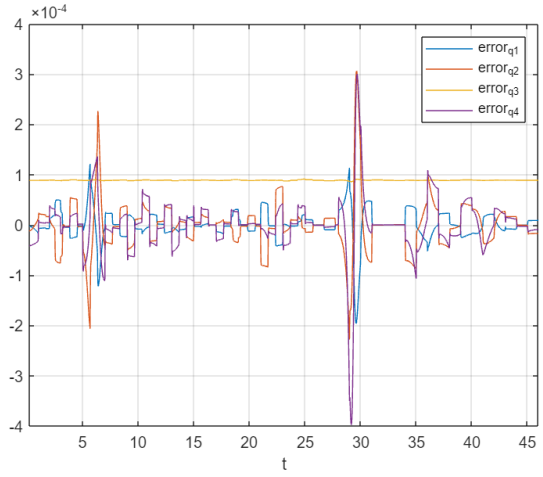


Figure 5.5: Position error with ϵ = 0.01

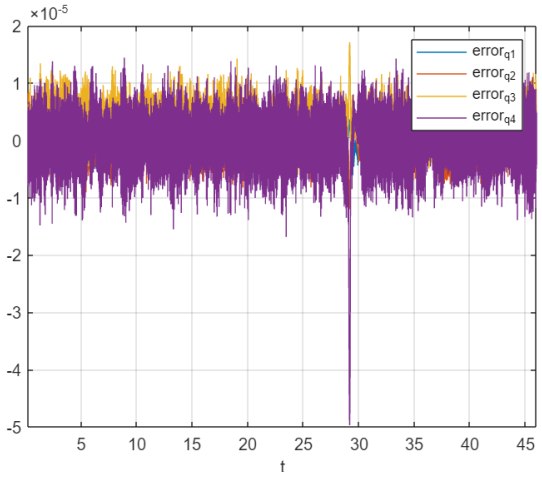


Figure 5.6: Position error with ϵ = 0.00001

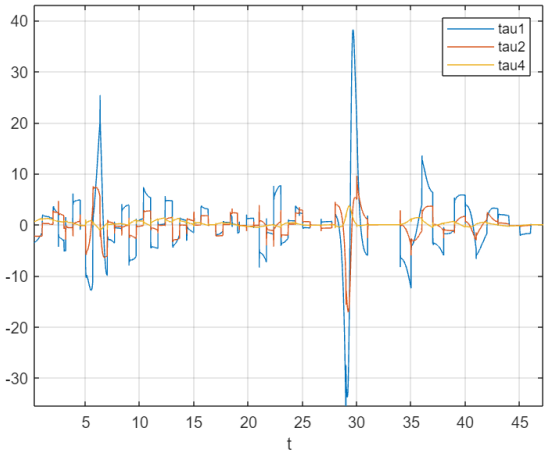


Figure 5.7: Value of the joint torques with ϵ = 0.01

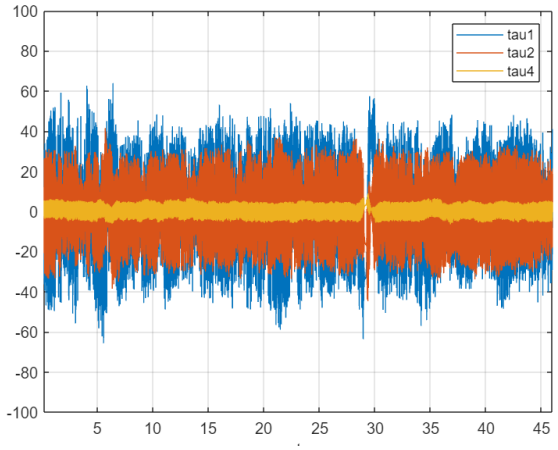


Figure 5.8: Value of the joint torques with ϵ = 0.00001

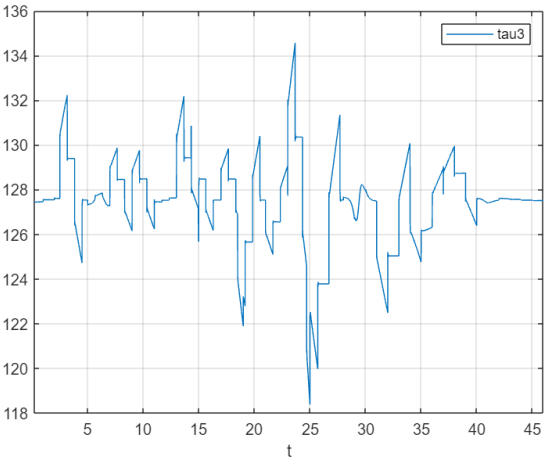


Figure 5.9: Value of the joint torque along z-axis with ϵ = 0.01

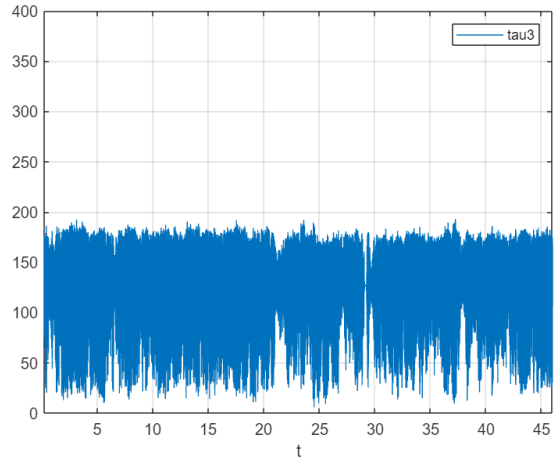


Figure 5.10: Value of the joint torque along z-axis with ϵ = 0.00001

	$\epsilon = 0.01$	$\epsilon = 0.00001$
Max error	1.9577e-04 3.0592e-04 1.0745e-04 3.9768e-04	7.0402e-06 1.1971e-05 1.7094e-05 4.9700e-05
Max torque	38.2161 17.0595 137.7181 3.6998	65.6695 44.7161 193.0186 8.9553

5.2 Adaptive Control

Unlike robust control, which attempts to counteract parametric uncertainties with robustifying action, adaptive control attempts to estimate the value of model parameters online. This is made possible by the linearity property of the dynamic model. We choose a control law in the form:

$$u = Y(q, \dot{q}, q_r, \dot{q}_r)\hat{\pi} + K_D\sigma$$

Using the Lyapunov method it can be noticed that the system is asymptotically stable if:

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

In particular, the matrices were chosen as:

```
lambda=diag([200,200,200,200]);
K_pi=diag([10,10,10,10,10,10,10,10,10,10,10,0.001,10,10,10]);
Kd_c_ad=diag([400,400,800,400]);
```

We can observe that in the matrix K_Π, a greater weight (because we are referring to the inverse) has been given to the most uncertain component, (the tip load).

5.2.1 Simulation

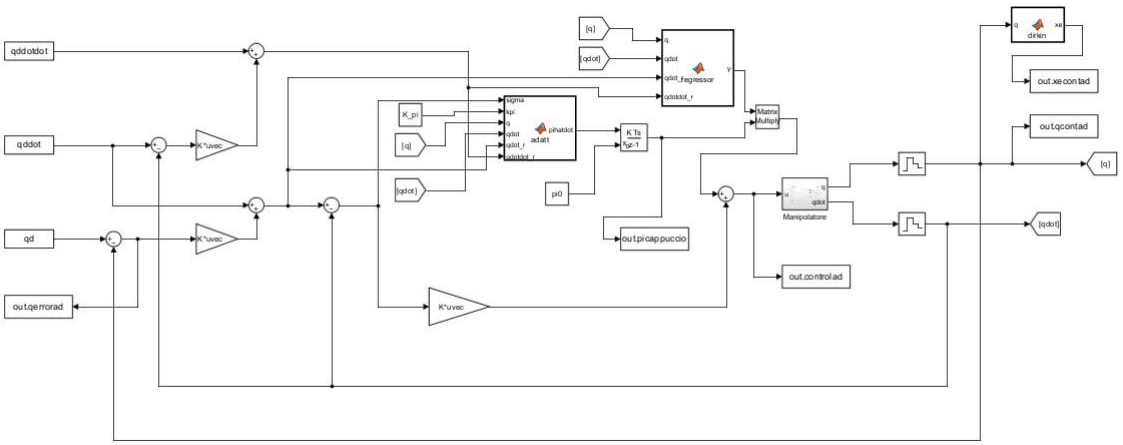


Figure 5.11: Adaptive Control Scheme

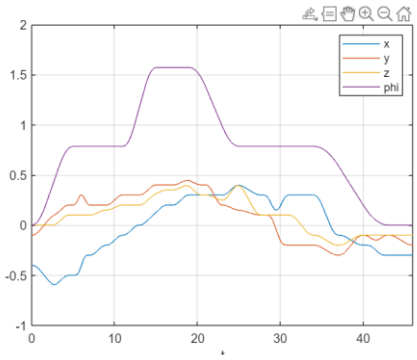


Figure 5.12: Position and Orientation in the operational space

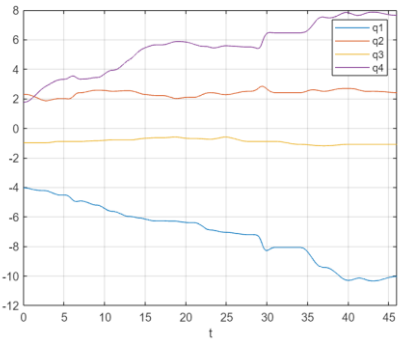


Figure 5.13: Variable in the Joint Space

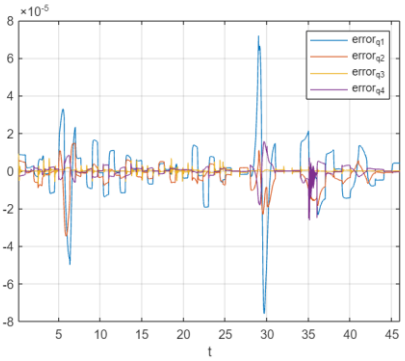


Figure 5.14: Position error with adaptive control

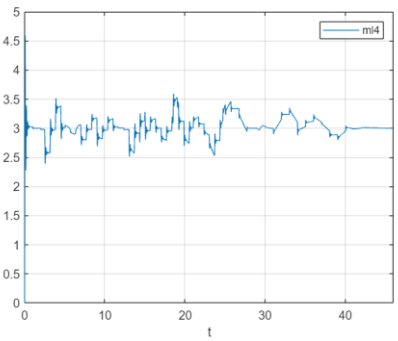


Figure 5.15: Payload estimation

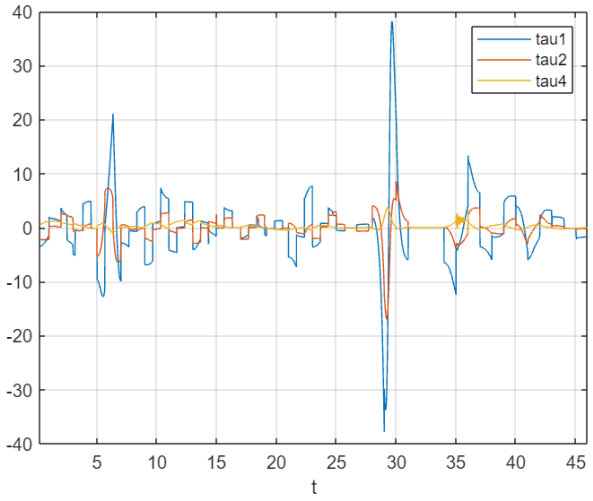


Figure 5.16: Value of the joint torques with adaptive control

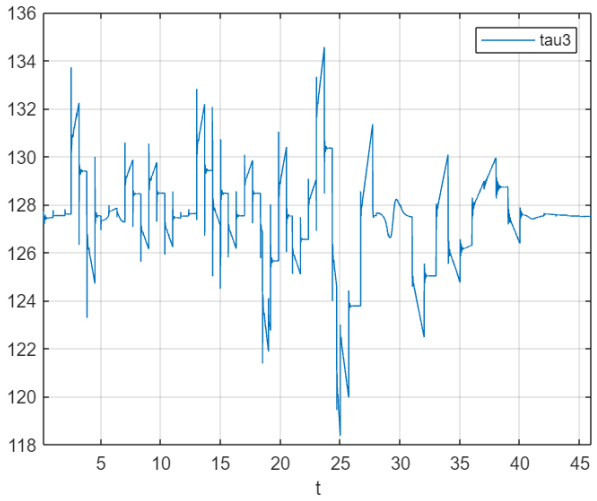


Figure 5.17: Value of the joint torque along z-axis with

Control:	ROBUST ($\epsilon = 0.01$)	ADAPTIVE
Max error	1.9577e-04 3.0592e-04 1.0745e-04 3.9768e-04	7.5877e-05 3.4675e-05 4.8182e-05 2.6234e-05
Max torque	38.2161 17.0595 137.7181 3.6998	38.1932 16.9898 134.5640 3.7253

We can see that both control systems shown above give satisfactory results, in particular, very similar joint actuation torque values were obtained, while the position error, due to the choices made, is slightly better in the adaptive control.

5.3 Operational Space Control

Below we will show a control approach that involves considering the control scheme directly in the operating space. This type of control uses a feedback linearization approach, directly in the operating space to counteract the contribution

due to the uncompensated load at steady state. We choose the control input as:

$$B(q)\ddot{q} + n(q,\dot{q}) = u$$

In this way, the controlled system is reduced to a double integrator:

$$\ddot{q} = y$$

Which can be controlled appropriately by making the choice:

$$y = J_A^{-1}(q)\left(x_d + K_P\tilde{x} + K_D\dot{\tilde{x}} + K_I\int_0^t \tilde{x}(\tau)d\tau\right) - \dot{J}_A(q,\dot{q})\dot{q}$$

We note that an integral action has also been added to meet the specification.Let us therefore see the choices of gain:

```
Kp_op = diag([5000 10000 10000 5000]);
Kd_op = diag([1000 1000 1000 1000]);
Ki_op = diag([2000 4000 2000 2000]);
```

5.3.1 Simulation

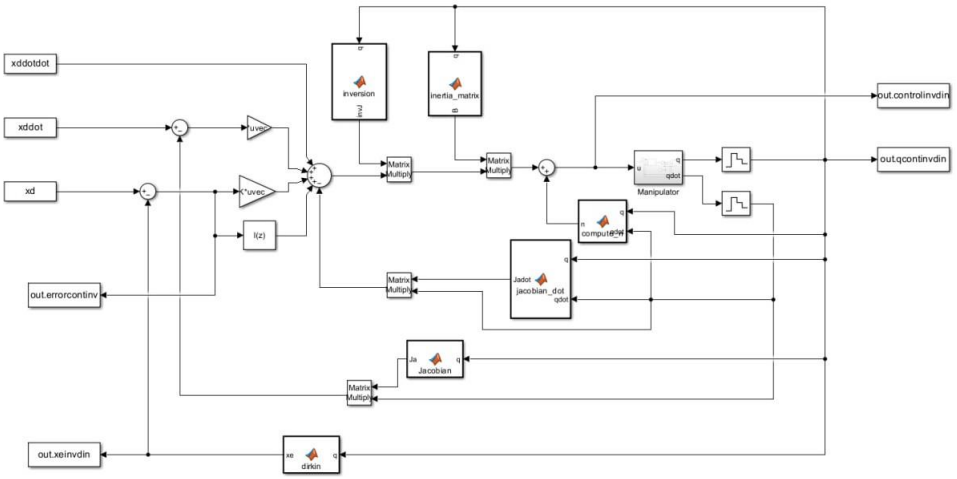


Figure 5.18 : Simulink scheme of Operational space control

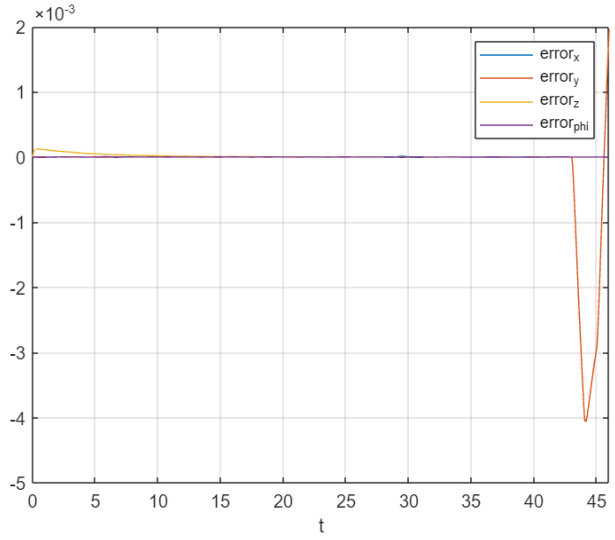


Figure 5.19 : Position error in the operational space inversedynamics control

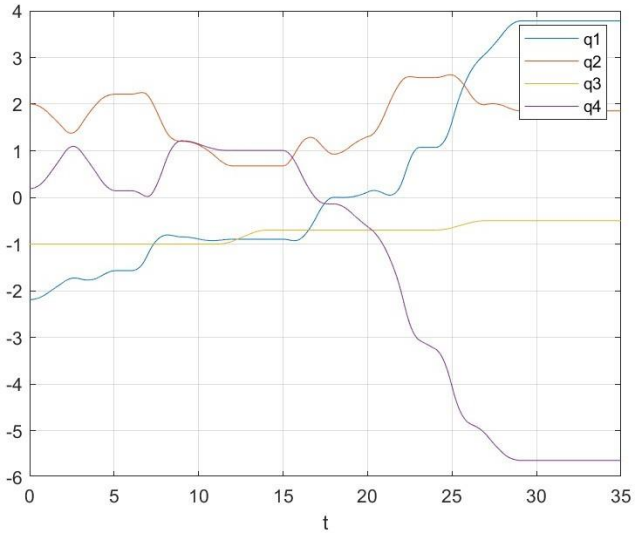
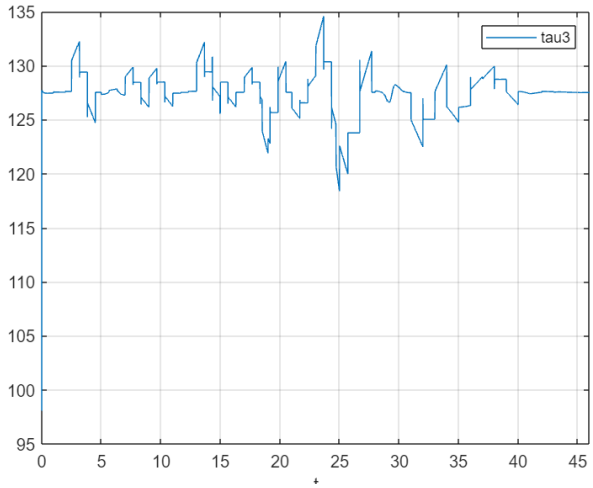


Figure 5. 20: Position and Orientation in the operational space with operational space inverse dynamics control



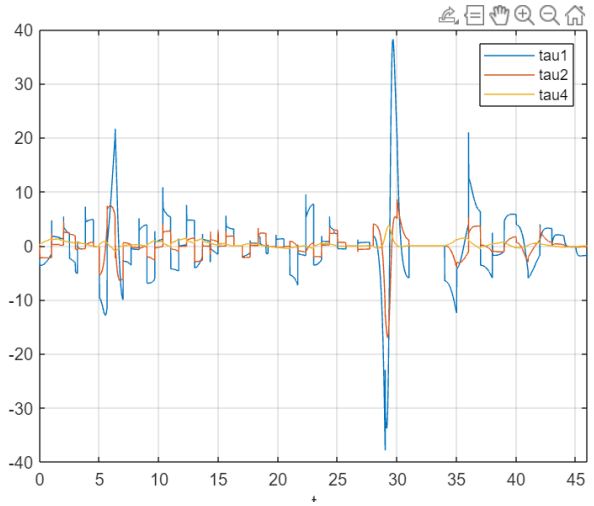


Figure 5.21 : Value of the joint torques with operational space inverse dynamics control

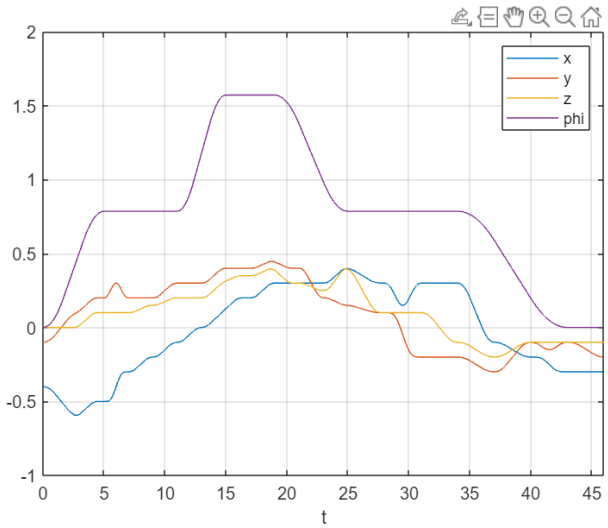


Figure 5.22 : Position in the operational space with inverse dynamics control

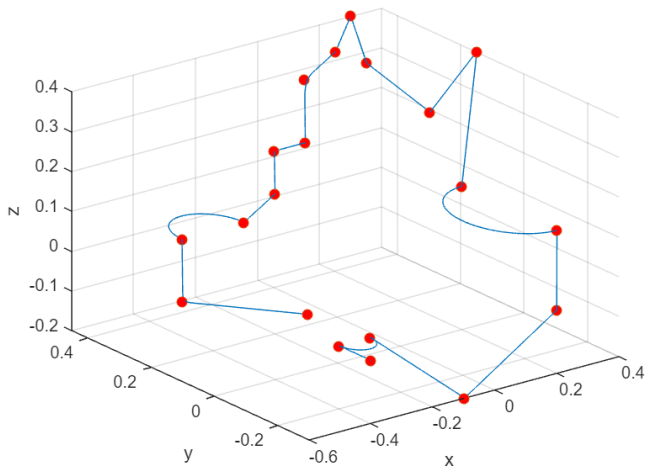


Figure 5.23 : 3D end-effector trajectory