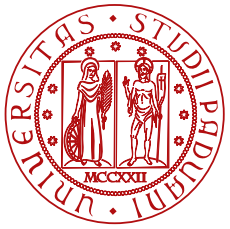

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Adaptive Robust Controller for handling Unknown Uncertainty of Robotic Manipulators

Master Candidate

Batuhan Bilen

Student ID 2071302

Supervisor

Prof. Carli Ruggero

University of Padova

Academic Year
2024/2025

To my parents
and friends,
for always believing in me.

Abstract

This thesis introduces a new adaptive robust control scheme, specially designed to improve the stability of robotic manipulators and the accuracy of their performance, whenever it takes place under uncertain and dynamic circumstances. Classical control approaches have an acceptable performance in structured environments, however, they usually have problems with model inaccuracy, tolerating external disturbances, and real-time parameter variations. In order to solve these problems, this research has produced a robust Adaptive Feedback Linearization Controller (ARFLC) which is able to make real-time correction of both structured and unstructured uncertainties.

The approach proposed in this study is to apply the method to two benchmark robotic systems, namely a 2-DOF RR manipulator and a 4-DOF SCARA manipulator. The construction of kinematic and dynamic models of both robots which includes the Denavit-Hartenberg parameters and Lagrangian formulation has been conducted. The controller contains an adaptive gain feature, whereby the adaptation coefficient is updated selectively, triggered only when the system faces atypical grinds in a particular way. This kind of pure adaptation of functions does not merely contribute to computational efficiency but also prevents overcompensing due to minor disturbances.

Simulation results confirmed the supremacy of the proposed ARFLC approach the problems of trajectory tracking, robustness, and speed of convergence, as compared to traditional ones. The results get to the root of the issue and confirm the use of adaptive robust control as one of the most practical ways in modern-day robot applications, where there is always the chance of uncertainty and the need for real-time adaptability is key.

Contents

List of Figures

List of Tables

List of Algorithms

List of Code Snippets

List of Acronyms

ARFL Adaptive Robust Feedback Linearization

BIBO Bounded Input Bounded Output

CM Center of Mass

DH Denavit Hartenberg

DOF Degrees of Freedom

FK Forward Kinematics

IK Inverse Kinematics

LQR Linear Quadratic Regulator

MATLAB Matrix Laboratory (Computational Tool)

MIMO Multiple Input Multiple Output

PD Proportional Derivative (Controller)

PID Proportional Integral Derivative (Controller)

RR Revolute-Revolute (2-DOF) Manipulator

SCARA Selective Compliance Assembly Robot Arm

1 Introduction

1.1 Motivation

1.1.1 Historical Background

Figure 1.1: Raymond Goertz's Teleoperated Mechanical Arm [12]

Starting the design of robotic manipulators was early in the 1950s because of a need for having machinery remotely controlled to protect human beings while undertaking operations in dangerous environments. One of the first tele-

operated mechanical arms, which were developed by Raymond Goertz, was used to handle radioactive substances at Argonne National Laboratory in 1951 [10], [11]. The invention opened the door for modern robotic manipulators as it was useful with precision and even for security applications. Robotic manipulators have been an integral part of modern industrial systems year after year, performing tasks such as assembly, welding, material handling, and precision manufacturing.

Despite such advancements, accurate and robust control of robot manipulators continues to be a significant challenge, particularly in dynamic and uncertain situations. Some of the reasons for this include external disturbances, model uncertainties, and changing operating conditions, among others, which continue to hinder optimal performance and hence the necessity for more advanced control schemes [14].

1.1.2 Challenges in Controlling Robotic Manipulators

Traditional control techniques, such as PID, LQR, PD, and robust control, have been widely used in robotic manipulator systems due to their mature theoretical backgrounds and simplicity for use . PID controllers, for example, are simple and effective for well-tuned systems with modest performance in trajectory tracking and disturbance rejection under normal operating conditions [6], [26], [24].

Linearization techniques of feedback enhance control precision by transforming non-linear dynamics into a linear model for optimizing control for improved tracking performance. Alternatively, robust control techniques are formulated to offer greater stability for bounded uncertainties through worst-case approximations in the system model. Adaptive control techniques have also been developed in an effort to real-time adapt system parameters based on observed dynamics changes. Although functioning satisfactorily, these conventional control strategies do possess some deficiencies when used with actual robotic manipulator systems, particularly in dynamic and uncertain environments [14], [17].

Among the major issues of such methods is their extensive dependence on accurate system models. Most conventional controllers are designed on the

basis of accurate information about the robots' dynamics, e.g., mass, friction, and external forces, which is hardly achievable in real situations [24], [26]. In the event of unpredictable disturbances or parameter variations, the performance of these controllers deteriorates severely, leading to trajectory errors and instability [17].

In addition, traditional methods are not dynamic enough to express dynamic adaptation capability against varying conditions [14], [17]. For instance, industrial robots would need diverse activities with divergent force and movement requirements and continuous manual adjusting of control parameters as a prerequisite. This restrictiveness disables traditional controllers to operate under uncertain and time-varying conditions. Thus, there is an increasing requirement for sophisticated control methods that incorporate both robustness and adaptability in a manner that enables robotic manipulators to drive under dynamic and uncertain environments [14].

1.1.3 Justification for Choosing Robust Adaptive Control

In consideration of the limitations offered by conventional control approaches, there is a demand for a better approach that incorporates adaptability, as well as robustness. Robust adaptive control enables real-time handling of uncertainties, together with external disturbances, thus ensuring stability and precision [14], [17]. As compared to conventional controllers, which depend on prior information concerning system parameters and continuous readjustments through manual means, a robust adaptive controller possesses the capability to adapt in real time, thereby enhancing performance in dynamic conditions. This thesis, therefore, focuses on formulating a robust adaptive control scheme with the objective of enhancing both the effectiveness and reliability of robotic manipulators subjected to uncertain environments.

1.2 Research Objectives and Contributions

The main aim of this master's thesis is to address the control challenges caused by uncertainties and disturbances in robotic manipulators through the investigation and development of robust control strategies. By enhancing the stability, precision, and adaptability of robotic systems, this research contributes

to improving industrial processes and advancing the field of robotics.

To achieve this end, the research first examines how changes in dynamic parameters and unexpected disturbances affect the stability and performance of robotic manipulators in industrial environments. The study then explores existing control methodologies, identifying their shortcomings in handling uncertainties. Consequently, novel adaptive and robust control methods are formulated and implemented to enable real-time compensation for these uncertainties. Furthermore, the performance of the proposed control strategies is evaluated through simulations and experimental validations, comparing them with conventional techniques. Additionally, the practical applicability of the developed control strategies is demonstrated in real-world scenarios, reinforcing their relevance to robotic systems.

Secondly, this research focuses on optimizing the adaptive control approach by refining the adaptation coefficient ρ. Instead of continuously applying adaptive modifications, the control strategy is designed to activate adaptation only when the level of uncertainty is significant enough to improve the tracking error beyond a predefined threshold. More specifically, the adaptation rate ρ is programmed to decrease when the tracking error remains within acceptable limits, thereby preventing unnecessary parameter updates and reducing computational overhead. The introduction of this selective adaptation mechanism enhances both the stability and efficiency of the controller while mitigating the risk of overfitting to transient disturbances. As a result, adaptive mechanisms are employed only when necessary within the control framework, thus proving the overall robustness and reliability of the system.

1.3 Scope and Structure of the Thesis

The thesis is divided into three major sections, each focusing on separate domains of modeling and control in robotics.

Part I introduces the fundamental concepts in robotic modeling and control, covering kinematic and dynamic modeling. These models establish the groundwork for subsequent simulations and the development of control strategies.

Part II deals with the control strategies for robotic manipulators, includ-

ing design and simulation, which includes feedback linearization and adaptive robust control, which are validated through RR and SCARA simulations.

Part III concludes the thesis by summarizing key findings, discussing their implications, and offering recommendations for future research directions.

The thesis is structured as follows:

Section I

Chapter 1: Introduction

- Motivation
- Research objectives and contributions
- Scope and structure of the thesis

Chapter 2: Theoretical Background

- Linear algebra review
- Kinematics of manipulators
- Feedback linearization
- Primary types of controllers in robotic systems
- Stability

Section II

Chapter 3: Mathematical Modeling of the Manipulator

- Modelling of RR manipulator
- Modelling of SCARA manipulator
- Deriving dynamic equations methods

Chapter 4: Control of Robotic Manipulators

- Motion equations
- Inverse dynamics control
- Feedback linearization control
- Designing adaptive robust control
- Designing ρ parameter

Chapter 5: Implementation,Simulation and Analysis

- RR manipulator simulation
- SCARA manipulator Simulation

Section III

Chapter 6: Conclusion

- Summary of key findings
- Future works

2

Theoretical Background

2.1 Linear Algebra

2.1.1 Overview

Linear algebra is a necessary mathematical equipment of robotics and it provides the setting for prescribing and analyzing robot manipulator movement, forces, and transformation. Vectors and matrices and their use allow precise models of and control of robot mechanisms that guarantee that it travels to specific locations and interacts with the environment [5], [24].

2.1.2 Vectors and Their Role in Robotics

Vectors are used to represent positions, velocities, and forces in robotic systems. A position vector defines the location of a point in three-dimensional space relative to a reference frame [6], [20]. Given a coordinate system, a position vector is expressed as:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.1}$$

where the Cartesian coordinates of the point are represented by x, y, and z. Additionally, velocity and force vectors describe the rate of change of position and the interaction forces acting on a manipulator.

2.1.3 Matrices and Transformations

Matrices are extremely useful to robot kinematics and dynamics since they allow for the coordinate transformation between two frames [6], [26], [7]. The rotation matrix R can be utilized to describe the orientation of a reference frame relative to another:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.2}$$

Each entry represents a directional cosine describing the orientation of the new frame relative to the original one. A homogeneous transformation matrix is used to combine both rotation and translation.

$$T = \begin{bmatrix} R & d \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

In this matrix, d represents the translation vector. Its a fundamental tool for calculating forward kinematics and for converting coordinates in robotic arms.

2.1.4 Applications in Robot Control

Linear algebra plays a fundamental role in the control of robotic manipulators. By using matrix operations, engineers can create control methods to move the robot, keep it balanced, and place its end point exactly where needed. Tools like Jacobians, rotation matrices, and transformation matrices help turn general movement commands into signals for each joint of the robot. This allows the robot to work quickly and correctly in real time.

2.2 Robotic Manipulator Kinematics and Analysis

2.2.1 Overview

A robotic manipulator consists of rigid bodies called links, connected by either rotational (revolute) or sliding (prismatic) joints. Typically, one end of the manipulator is attached to a fixed base, while the other carries the tool or end-effector. The movement of the system results from the sequential motion of these

links, which must be described mathematically to determine the end-effectors position and orientation.

Forward kinematics is used to compute the position of the end-effector based on known joint values, whereas inverse kinematics involves determining the joint angles required to reach a desired position. Since inverse kinematics often includes complex and nonlinear equations, it can be challenging to solve; therefore, alternative approaches are commonly used. One widely adopted method is the Denavit-Hartenberg (DH) convention, which simplifies kinematic modeling by defining four parameters for each link. Understanding these principles is essential for effective motion control. The following section introduces differential kinematics, which describes the relationship between joint velocities and the motion of the end-effector.

2.2.2 Forward Kinematics

Forward kinematics calculates the position and orientation of the end-effector given the joint parameters (e.g., angles for revolute joint or displacement for prismatic joint). It is a simple procedure in which the position of each link is computed serially from the base to the end-effector.

Mathematically, for an open-chain robotic manipulator with n joints, the pose of the end-effector with respect to the base frame is calculated by the transformation matrix as:

$$T_n^0 = A_1^0(q_1)A_2^1(q_2)\dots A_n^{n-1}(q_n) \tag{2.4}$$

where A_{i+1}^i represents the homogeneous transformation matrix between consecutive joints [2].

2.2.3 DenavitHartenberg (DH) Convention

In order to standardize the geometric representation of robotic manipulators, the Denavit Hartenberg (DH) convention was developed [7]. This method provides a standard method for specifying spatial relationships between links by assigning a particular local coordinate frame to each joint. Based on four basic parameters for each joint, the DH convention provides systematic derivation of transformation matrices that specify the position and orientation of each link with respect to the previous one. Although other methods of frame assignment

do exist, the DH convention is still very popular because of its simplicity and success in kinematic analysis.

- d_i: The offset along the previous z-axis, representing the distance from one link to the next along the shared axis.
- θ_i: The joint angle, which is the rotation about the previous z-axis, typically used for revolute joints.
- a_i: The link length, defined as the distance between two joints measured along the current x-axis.
- α_i: The link twist, or the angle between the previous and current z-axes, measured about the current x-axis.

Using these parameters, a homogeneous transformation matrix is constructed as:

$$A_i^{i+1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Applying this transformation iteratively from the base to the end-effector, the forward kinematics of the manipulator can be determined.

2.2.4 Inverse Kinematics

Inverse kinematics is the process of calculating the joint variables that lead to a desired position and orientation of the end-effector. It is a more complex problem than forward kinematics because it is nonlinear and may have numerous or no solutions.

For a non-redundant manipulator, the inverse kinematics can be expressed as solving for q in:

$$f(q) = x_d \tag{2.6}$$

where $f(q)$ is the forward kinematics function and x_d is the desired end-effector pose. The complexity comes from the fact that this equation is typically nonlinear and can need numerical methods, e.g., Newton-Raphson methods or optimization methods, whenever no analytical solution is available. In order to continue investigating the motion properties, differential kinematics provides a

way to connect joint velocities to end-effector velocities, as will be presented in the next section.

To further investigate the characteristics of motion, differential kinematics provides a way to relate end-effector velocities and joint velocities.

2.3 Jacobian Matrix and Differential Kinematics

The Jacobian matrix is utilized to relate the joint velocities and the resultant linear and angular end-effector velocities. For an n-degree-of-freedom (DOF) robotic manipulator, the Jacobian matrix $J(q)$ is defined as:

$$v = J(q)\dot{q} \tag{2.7}$$

where v is the velocity of the end-effector (linear and angular), and $\dot{q}$ is the vector of joint velocities. The Jacobian plays a crucial role in velocity kinematics, singularity analysis, trajectory planning, and force transmission [3].

2.3.1 Jacobian Decomposition

The Jacobian matrix can be decomposed into two components:

$$J(q) = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \tag{2.8}$$

where:

- J_v relates joint velocities to the linear velocity of the end-effector.
- J_ω relates joint velocities to the angular velocity of the end-effector.

2.3.2 Jacobian for Revolute and Prismatic Joints

For a revolute joint i, the Jacobian columns are given by:

$$J_v^i = z_{i-1} \times (o_n - o_{i-1}) \tag{2.9}$$

$$J_\omega^i = z_{i-1} \tag{2.10}$$

where z_{i-1} is the unit vector along the joint axis, and o_n and o_{i-1} are the positions of the end-effector and joint $i-1$, respectively.

For a prismatic joint, the Jacobian columns are defined as:

$$J_v^i = z_{i-1}, \quad J_\omega^i = 0 \tag{2.11}$$

Understanding these properties is fundamental for precise motion control.

2.4 Feedback Linearization

Feedback linearization is a nonlinear control strategy that transforms a nonlinear system into an equivalent linear one through an appropriate choice of control input. Unlike conventional linearization methods, which approximate the system behavior around an equilibrium point, feedback linearization aims for exact cancellation of nonlinearities, enabling the application of well-established linear control techniques [23], [25].

Mathematically, consider a nonlinear system represented as:

$$\dot{x} = f(x) + g(x)u \tag{2.12}$$

where x is the system state, u is the control input, and $f(x)$, $g(x)$ define the system dynamics. The key idea behind feedback linearization is to design u such that the transformed system behaves like a linear system:

$$\dot{z} = Az + Bu \tag{2.13}$$

where z is a new set of variables obtained through state transformation, and A, B are constant matrices representing a linear system.

This method is particularly useful in systems with strong nonlinearities, such as robotic systems, power electronics, and aerospace applications. By effectively canceling nonlinear terms, feedback linearization allows the use of classical linear controllers like PID, LQR, and pole placement, leading to improved stability and precise control.

However, feedback linearization relies on an accurate mathematical model of the system. If model uncertainties or external disturbances exist, the control performance may degrade, requiring robust or adaptive techniques to compensate for deviations.

In summary, feedback linearization is a powerful approach for nonlinear control, offering a systematic way to achieve linear-like behavior in complex systems while facilitating the use of well-established linear control strategies.

2.5 Fundamentals of Robotic Control

Once the kinematic relationships of a robotic manipulator are established, the next crucial step is ensuring accurate motion through control strategies [9], [22]. Robot controllers are responsible for regulating joint positions, velocities, and forces to achieve the desired task. The primary types of controllers in robotic systems include:

- **Proportional-Derivative (PD) Control:** A straightforward and common technique that adjusts control inputs proportionally to the error and its derivative, yielding quick response with modest stability.
- **Proportional-Integral-Derivative (PID) Control:** A more sophisticated version with an integral term to remove steady-state errors, such that it is suitable for precise tracking.
- **Linear Quadratic Regulator (LQR) Control:** A model-based optimal control approach that minimizes a quadratic cost function in an effort to balance system performance and control effort.

Among these, PID controllers are the most commonly implemented due to their balance of simplicity and effectiveness.

Additionally, control system performance is linked to stability, ensuring the manipulator remains within safe operating limits and converges to the desired state without oscillations or instability.

2.6 Stability Analysis in Robotic Control

The stability analysis is the foundation of the robotic manipulator control theory, where the main aim is to secure that the system persists at the required level of performance and recovers from its disturbances [19].

Using these smooth and precise movements avoids complications such as oscillations, energy losses, and hardware damage. It is mostly used in the high-precision industrial automation and medical robotics fields to allow sm ooth and precise trajectory tracking. Among others, engineers, by using the stability concept, can come up with robust and fault-tolerant control approaches which can help to keep the system intact even in the case of changing conditions.

2.6.1 Types of Stability

- **Bounded Input Bounded Output (BIBO) Stability:** A system is BIBO stable if, for every bounded input, the output remains bounded. This is critical for ensuring predictable system responses to external inputs.
- **Asymptotic Stability:** A stronger form of stability, where system trajectories not only remain bounded but also converge to the equilibrium point as time approaches infinity.
- **Exponential Stability:** A system is exponentially stable if the rate of convergence to equilibrium is exponential, meaning disturbances dissipate quickly over time.
- **Lyapunov Stability:** A system is Lyapunov stable if, for any small perturbation, its state remains within a bounded region around the equilibrium. This ensures that the system does not diverge uncontrollably [25].
- **Nyquist and Routh-Hurwitz Stability Criteria:** These classical methods provide analytical tools for assessing the stability of control systems by analyzing the frequency response or characteristic polynomial of the system.

3

Mathematical Modeling of the Manipulator

3.1 Overview

The process of mathematical modeling of a manipulator serves the purpose of unraveling its movement, control, and connection with the crowded ambience [27]. It means establishing equations that express the manipulator's kinematics and dynamics that are basic for path planning, control design, and performance analysis.

The kinematic modeling deals with the geometrical relations among joint variables and end-effector position and orientation. Forward kinematics gives a straightforward link from joint space to Cartesian space, whereas the inverse kinematics establishes the prediction of joint configurations that lead to the desired end-effector position [13]. These are mainly used in the field of robotics for motion planning and ensuring precise positioning.

Dynamic modeling, on the flip side, considers the forces and torques that the manipulator encounters. It takes the form of deriving the equations of motion using either the Newton-Euler formulation or the Lagrangian approach. The first approach derives on the basis of Newtons laws, while energy principles are the root for the second approach. The obtained equations contain the inertia matrix $M(q)$, Coriolis and centrifugal forces $C(q, \dot{q})$, and the gravity vector $G(q)$,

building the basement of control strategies such as PID control, model-based control, and adaptive control [24].

An organized mathematical model is the driving force behind the manipulator's efficiency and stability, which are essentially what cause it to be the best in robotics, automation, and industrial systems.

3.2 Mathematical Modeling of RR Manipulator

The RR manipulator is a two-joint robotic arm with rotational joints that are connected by rigid links. The goal of the forward kinematics problem is to determine the position of the end-effector given the joint angles and the lengths of the links. To solve this, we use the Denavit-Hartenberg (DH) parameters, which provide a systematic way to model the manipulator geometrically [7].

3.2.1 Denavit-Hartenberg Parameters

For a two-link planar manipulator, the Denavit-Hartenberg parameters are defined as:

- θ_1 and θ_2: The joint angles of the first and second joints.
- $d_1 = 0$, $d_2 = 0$: The offsets in the z-direction are zero because both joints are revolute.
- a_1 and a_2: The lengths of the first and second links.
- $\alpha_1 = 0$, $\alpha_2 = 0$: No twists between the links.

The transformation matrix for the first joint is:

$$A_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & a_1\sin\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, for the second joint:

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total transformation matrix from the base to the end-effector is the product of these two matrices:

$$T = A_1 \cdot A_2 \tag{3.1}$$

This matrix gives the position and orientation of the end-effector relative to the base frame. From the transformation matrix we can place the end-effector with the required accuracy in the cartesian coordinate system.

The position of the end-effector in the x and y directions is described by:

$$x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \tag{3.2}$$

$$y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2) \tag{3.3}$$

Thus, the forward kinematics provides a relationship between the joint angles and the end-effector position. It is necessary to use it for motion planning and robot trajectory control in three-dimensional space.

3.2.2 Visual Representation of Forward Kinematics

The primary way to understand is visualizing the manipulator as a chain of connecting links [27]. The first link (length a_1) is attached to the base, and the second link (length a_2) is connected to the first one. The angles θ_1 and θ_2 display the rotation of these links about their own axes. The workspace is given by the difference $|a_1 - a_2|$ and the sum $a_1 + a_2$ of the link lengths. The area in which the RR manipulator can move is the workspace heuristics which can be evaluated in a few simple issues as follows.

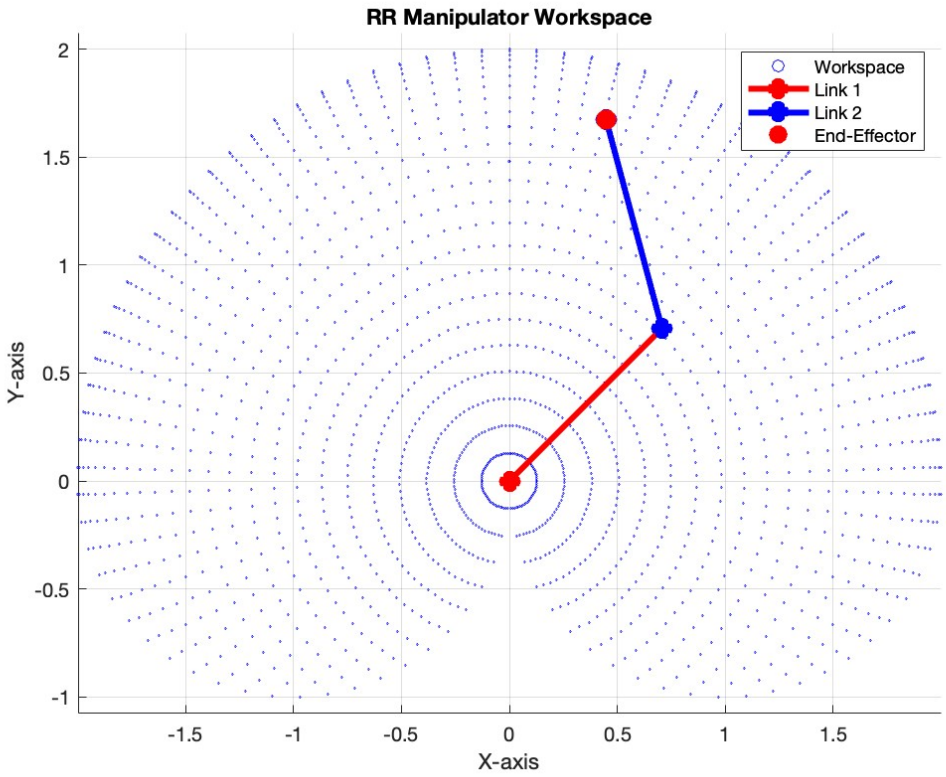

Figure 3.1: Workspace of the RR manipulator

3.3 Inverse Kinematics of the RR Manipulator

Inverse kinematics is a method used for evaluating the joint variables θ_1 and θ_2 that are required for a specific end-effector position (x, y). In this problem the objective is to find the adequate link movements that will allow the manipulator to reach this position.

The forward kinematics equations help us to solve forθ_1 and θ_2 starting by using those unique values as a fundamental step.

$$x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \tag{3.4}$$

$$y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2) \tag{3.5}$$

To isolate θ_2, we use trigonometric identities. The equation for θ_2 is derived

as:

$$\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1a_2}\right) \tag{3.6}$$

Once θ_2 is known, we can substitute it into the equation for x or y to solve for θ_1. The solution for θ_1 is:

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{a_2 \sin\theta_2}{a_1 + a_2\cos\theta_2}\right) \tag{3.7}$$

The joint angles of θ_1 and θ_2 can be used to acquire the position of end-effectors in the space. The inverse kinematics solution is substantial for robot motion control, which means that a manipulator can be located at any desired point in the workspace.

3.4 Dynamic Modeling of the RR Manipulator

The prediction of behavior and the exact control of a robot arm will be impossible without a deep insight into the robotic manipulator mechanisms. The kinetic models describe the joint and the end effector motion, while the dynamic models show the forces, torques, and energy transfer among the components of the system. Dynamic models are used to explore not only the presence of disturbances from the outside but also the interactions between the joints and the stability of the whole system. The Newtonian and Lagrangian methods constitute the two basic methods to derive dynamic equations. Both approaches demonstrate unique features in terms of the computational cost and simplicity of the formulation [24].

3.4.1 Methods for Deriving Dynamic Equations

The Newtonian and Lagrangian approaches are the most important methods that are mainly used in complexity of dynamic equations of robot manipulators [18], [21].

Newtonian Formulation

Newtonian Approach: This procedure evaluates the forces and torques on each robot link by using Newton's second law of motion. By solving the equi-

librium and free-body diagrams, the robot's motion equations are found. When this method is implemented, the recurrent approach needs only the real-time inputs, this is how physics can help the process.

Lagrangian Formulation

Lagrangian Approach: This method is mainly based on the energy concept, which is not directly addressed as the main force. The amortized in Euler-Lagrange function is replaced by the potential energy of the system and the kinetic energy for the equations of motion. This procedure is a systematic and wide-ranging approach to complex robotic systems.

3.4.2 Inertia Matrix

The inertia matrix M demonstrates the interaction arising between the joint accelerations and the torques appealed to the manipulator for it to move. The inertia matrix of the RR manipulator can be calculated as follows:

$$M = \begin{bmatrix} I_1 + I_2 + m_2 l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos(\theta_2) & I_2 + m_2 l_2^2 + m_2 l_1 l_2 \cos(\theta_2) \\ I_2 + m_2 l_2^2 + m_2 l_1 l_2 \cos(\theta_2) & I_2 + m_2 l_2^2 \end{bmatrix}$$

Where:

- m_2 is the mass of the second link.
- l_1 and l_2 are the lengths of the two links.
- I_1 and I_2 are the moments of inertia of the links.

This inertia matrix is significant in calculating the torques required to accelerate the robot and in determining the effect of link masses on robot motion.

3.4.3 Coriolis Forces

The Coriolis forces arise due to the interaction between the links' velocities. These forces are given by the following expression:

$$C = \begin{bmatrix} -2m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_2 & -m_2 l_1 l_2 \sin(\theta_2)(\dot{\theta}_1 + \dot{\theta}_2) \\ m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_1 & 0 \end{bmatrix}$$

Where $\dot{\theta}_1$ and $\dot{\theta}_2$ are the joint velocities. These forces must be accounted for when designing controllers, as they affect the manipulator's response to input torques.

3.4.4 Gravitational Forces

The gravitational forces acting on the manipulator can be expressed as:

$$G = \begin{bmatrix} (m_1 l_1 + m_2 l_1) g \cos(\theta_1) + m_2 l_2 g \cos(\theta_1 + \theta_2) \\ m_2 l_2 g \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Where g is the acceleration due to gravity. The gravitational forces depend on the joint angles and the lengths of the links.

3.5 Mathematical Modeling of SCARA

A SCARA manipulator, a 4-DoF robotic arm, has two rotational joints, one prismatic joint, and one final rotational joint for end-effector orientation. In the forward kinematic problem, the main task is to find the end-effector location and orientation depending on the joint variables like two rotational angles (θ_1, θ_2), one linear displacement (d_3), and a final rotational angle (θ_4).

If we want to handle the forward kinematics problem in an ordered manner, we must write down our robot kinematically as the Denavit-Hartenberg (DH) parameters that in a structured manner through a geometric representation displays the manipulators configuration.

3.5.1 Denavit-Hartenberg Parameters

For a four-degree-of-freedom SCARA manipulator, the Denavit-Hartenberg (DH) parameters are given by the following form:

- $\theta_1, \theta_2, \theta_4$: The joint angles of the first, second, and fourth joints (rotational).
- d_3: The prismatic joint variable, representing vertical movement.
- a_1, a_2: The lengths of the first and second links.
- $\alpha_1, \alpha_2, \alpha_3, \alpha_4$: The twist angles between the links.

Joint	θ_i	d_i	a_i	α_i
1	θ_1	d_1	a_1	0
2	θ_2	0	a_2	0
3	0	d_3	0	0
4	θ_4	0	0	0

Table 3.1: Denavit-Hartenberg Parameters for the SCARA manipulator

The DH parameter table for the SCARA manipulator is given below:
The transformation matrix for the first joint is:

$$A_1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & a_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & a_1\sin\theta_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, the transformation matrix for the second joint is:

$$A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & a_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The transformation matrix for the third (prismatic) joint is:

$$A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And for the fourth joint:

$$A_4 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The total transformation matrix from the base to the end-effector is given by:

$$T = A_1 \cdot A_2 \cdot A_3 \cdot A_4$$

This matrix indicates where the end effector of the SCARA manipulator (in its base frame) is and how it is oriented. From this transformation matrix, picking up the associated parts that are used in this transformation to calculate the Cartesian coordinates of the end-effector.

3.5.2 Visual Representation of Forward Kinematics

For a clearer picture, imagine the SCARA manipulator as a chain of connected links with both turning and linear sliding joints. The first link (length a_1) is attached to the base, and the second link (length a_2) is attached to the first one. The angles θ_1 and θ_2 describe the rotation of these links about their respective axes, while d_3 determines the vertical displacement of the end-effector. The final joint, with angle θ_4, controls the orientation of the end-effector.

The workspace of the SCARA manipulator is bounded between the absolute difference $|a_1 - a_2|$ and the sum $a_1 + a_2$ of the link lengths in the XY-plane. The vertical range is determined by the limits of d_3. By extracting the relevant components from the transformation matrix, we can analyze the SCARA manipulators workspace as follows.

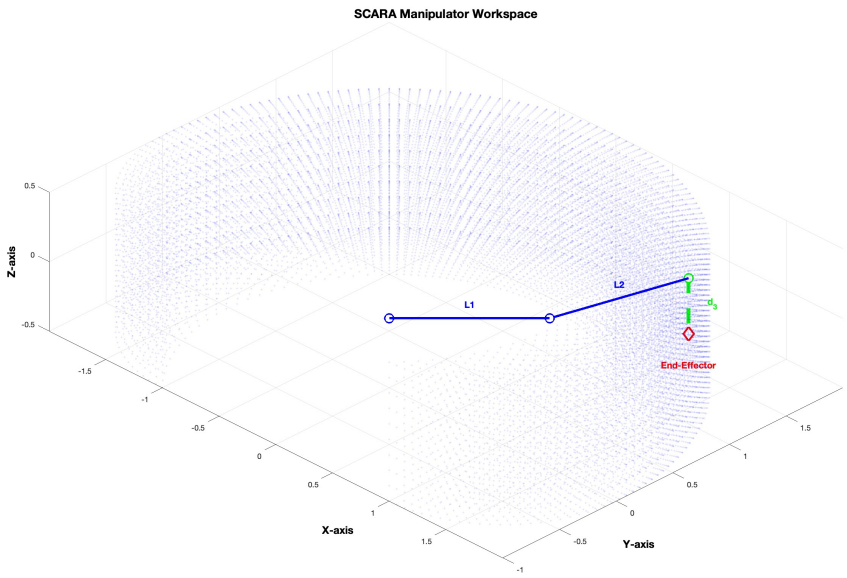

Figure 3.2: Workspace of the SCARA manipulator

3.5.3 Inverse Kinematics of the SCARA Manipulator

Inverse kinematics is the process of determining the joint variables required to place the end-effector at a desired position and orientation. For the 4DOF SCARA manipulator, we need to solve for the joint angles $\theta_1, \theta_2, \theta_4$ and the prismatic displacement d_3 given the desired Cartesian coordinates (X, Y, Z) and end-effector orientation.

Solving for θ_1 and θ_2

The first two rotational joints determine the position of the end-effector in the XY-plane. Using geometric relations, the inverse kinematics equations for θ_1 and θ_2 are:

$$\cos\theta_2 = \frac{X^2 + Y^2 - a_1^2 - a_2^2}{2a_1a_2}$$

$$\theta_2 = \tan^{-1}\left(\frac{\sin\theta_2}{\cos\theta_2}\right)$$

$$\theta_1 = \tan^{-1}\left(\frac{Y}{X}\right) - \tan^{-1}\left(\frac{a_2\sin\theta_2}{a_1 + a_2\cos\theta_2}\right)$$

Solving for d_3

The prismatic joint determines the vertical displacement:

$$d_3 = d_1 - Z$$

where d_1 is the initial height of the base along the Z-axis.

Solving for θ_4

The final rotation θ_4 is determined based on the desired end-effector orientation:

$$\theta_4 = \theta_{\text{desired}} - (\theta_1 + \theta_2)$$

By solving these equations, we can determine the required joint variables to position the SCARA manipulators end-effector at the specified Cartesian

coordinates.

Inertia Matrix

The inertia matrix (M) for the 4DOF SCARA manipulator, which accounts for the mass distribution and link lengths, is given by:

$$M = \begin{bmatrix} M_{11} & M_{12} & 0 & 0 \\ M_{21} & M_{22} & 0 & 0 \\ 0 & 0 & M_{33} & 0 \\ 0 & 0 & 0 & M_{44} \end{bmatrix}$$

where:

$$M_{11} = I_1 + I_2 + m_2 a_1^2 + m_2 a_2^2 + 2 m_2 a_1 a_2 \cos(\theta_2)$$

$$M_{12} = M_{21} = I_2 + m_2 a_2^2 + m_2 a_1 a_2 \cos(\theta_2)$$

$$M_{22} = I_2 + m_2 a_2^2$$

$$M_{33} = m_3$$

$$M_{44} = I_4$$

where:

- m_2, m_3 are the masses of the second and third links.
- a_1, a_2 are the lengths of the first and second links.
- I_1, I_2, I_4 are the moments of inertia of the links.
- M_{33} corresponds to the mass of the prismatic joint.

This matrix is essential for calculating the torques needed for robot motion and understanding the effects of link masses.

Coriolis Forces

The Coriolis forces arise due to the interaction between link velocities and affect the system dynamics. The Coriolis matrix (C) for the SCARA manipulator is:

$$C = \begin{bmatrix} -2 m_2 a_1 a_2 \sin(\theta_2)\dot{\theta}_2 & -m_2 a_1 a_2 \sin(\theta_2)(\dot{\theta}_1 + \dot{\theta}_2) & 0 & 0 \\ m_2 a_1 a_2 \sin(\theta_2)\dot{\theta}_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\dot{\theta}_1, \dot{\theta}_2$ are the joint angular velocities. Coriolis forces must be considered in control design as they affect the manipulators response to input torques.

Gravitational Forces

The gravitational forces acting on the SCARA manipulator are expressed as:

$$G = \begin{bmatrix} (m_1 a_1 + m_2 a_1) g \cos(\theta_1) + m_2 a_2 g \cos(\theta_1 + \theta_2) \\ m_2 a_2 g \cos(\theta_1 + \theta_2) \\ m_3 g \\ 0 \end{bmatrix}$$

where:

- g is the acceleration due to gravity.

The gravitational forces depend on the joint angles, lengths of the links and mass of the links.

These equations are critical for calculating the required torques to compensate for gravity in the control system of the SCARA robot.

4

Control of Robotic Manipulators

4.1 Overview

The control of robot manipulators is a main aspect of robotics engineering, where the ability to be precise, to be stable, and to be efficient in doing actions is very important. A robot manipulator is a highly complicated multi-input, multi-output (MIMO) system that presents nonlinear behavior because of strong coupling effects, imprecise parameters, and uncertain disturbances. Control strategies must be effective and be able to deal with these issues and in this way be sure that the trajectory is controlled with high accuracy as well as the rejecting of disturbances and maintaining system robustness.

Manipulator control systems are aimed at producing the appropriate control inputs to move the system towards a desired state as well as compensating for uncertainties. More generally, robot tasks can be included in two groups: regulation tasks, where the manipulator keeps the same position, and trajectory tracking tasks, where the end effector runs on a fixed predefined path within the time period. The task of managing a robot arm becomes much more complicated when the following elements are included: disturbances, the inaccuracy of models, and the variation of payload.

Many control methodologies have been developed to cope with these challenges, each with unique advantages and disadvantages. Among traditional control methods like Proportional-Integral-Derivative (PID) control, computed

torque control, and feedback linearization have shown up largely due to their simplicity and easy fitting. These methods are proper for organized space directions where systems work under natural dynamic conditions and yet fail to deliver high-quality results in the real world, where model uncertainties and external perturbations are included.

To transcend these restrictions, advanced control techniques have been developed. Adaptive control updates system parameters in real time and thus is able to cope with uncertainties in robotic manipulators. Adaptive controllers, however, depend on persistent excitation conditions and are computationally expensive for real-time implementation. Robust control, by contrast, guarantees stability and performance under bounded disturbances and parameter variations but does not adapt to unknown changes in the system dynamics.

A more advanced and potent strategy is adaptive robust control, which combines the adaptability of adaptive control and the disturbance-rejection attribute of robust control. By this method, the manipulator is enabled to handle both structured and unstructured uncertainties so that accurate trajectory tracking is still possible even under unknown model variations. One of the fundamentals of adaptive robust control is Lyapunov-based stability analysis, which guarantees the stability of the control system in different conditions. The adaptation of gain parameters, i.e., ρ, also imparts the feature of dynamic adaptation to the controller in response to changes in the behavior of a system.

This chapter provides a detailed examination of manipulator control methods, with particular emphasis on adaptive robust control [6],[18],[24]. The chapter begins with the dynamics modeling of manipulator mathematics using the Lagrangian approach [8],[16]. Then, it continues with the discussion of conventional control methods and their limitations. A step-by-step derivation of adaptive robust control laws and stability proofs based on the Lyapunov function are provided [1],[4],[15]. The chapter also discusses the real-time implementation of the control schemes and evaluates their performance by simulating RR and SCARA manipulator systems. By introducing a control strategy that can successfully reject uncertainties and external disturbances, the goal of this research is to improve the accuracy, stability, and robustness of robotic manipulator control to widen its application in real industrial and research environments.

4.2 Lagrangian Formulation and Equations of Motion

The dynamic model of a robot manipulator defines the mathematical relation between the joint torques produced by actuators and the ensuing motion of the system. The formulation is fundamental to the understanding and control of the behavior of robotic multi-degree-of-freedom (DOF) systems.

Through Lagranges approach, the equations of motion can be systematically derived in a manner that is independent of any reference coordinate frame. This is accomplished by introducing a set of generalized coordinates q_i $(i = 1, \ldots, n)$ that conveniently describe the link positions of an n-DOF manipulator. The Lagrangian function of the system is then expressed as:

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{4.1}$$

where:

- T represents the total kinetic energy of the system,
- U represents the total potential energy of the system.

The Lagrange equation of motion is derived as:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \xi_i, \quad i = 1, \ldots, n \tag{4.2}$$

where ξ_i denotes generalized force associated with the generalized coordinate q_i.

The application of this equation to robotic manipulators results in the well-known rigid body dynamic equation:

$$M(q(t)\ddot{q}(t) + C(q(t), \dot{q}(t))\dot{q}(t) + G(q(t)) = \tau(t) \tag{4.3}$$

where:

- $M(q(t)) \in \mathbb{R}^{N\times N}$ represents the inertia matrix, which is a definite positive matrix for any $q(t)$;
- $C(q(t), \dot{q}(t)) \in \mathbb{R}^{N\times N}$: Represents the Coriolis and centrifugal forces.
- $G(q) \in \mathbb{R}^N$: Represents the gravitational effects.

- $= \tau$: Represents the generalized torques applied to system.

4.2.1 Kinetic Energy

For a robotic manipulator with *n rigid links*, the total kinetic energy is determined by the contributions from both the motion of each link and the motion of the actuators driving the joints.

It is expressed as:

$$\mathcal{T} = \sum_{i=1}^{n} \left(\mathcal{T}_{\ell_i} + \mathcal{T}_{m_i} \right) \tag{4.4}$$

$\mathcal{T}_{\ell_i}$: kinetic energy of Link i

$\mathcal{T}_{m_i}$: kinetic energy of the motor actuating Joint i

The kinetic energy contribution of Link i consists of translational and rotational components:

$$\mathcal{T}_{\ell_i} = \frac{1}{2} m_{\ell_i} \dot{p}_{\ell_i}^T \dot{p}_{\ell_i} + \frac{1}{2} \boldsymbol{\omega}_i^T \mathbf{R}_i \mathbf{I}_{\ell_i}^i \mathbf{R}_i^T \boldsymbol{\omega}_i. \tag{4.5}$$

- $\dot{p}_{\ell_i}$ = velocity of the center of mass of link i
- $\boldsymbol{\omega}_i$ = angular velocity of link i
- $\mathbf{I}_{\ell_i}$ = inertia tensor of link i relative to its center of mass
- m_{ℓ_i} = mass of link i

Using Jacobian matrices, the velocity components can be expressed as:

$$\dot{p}_{\ell_i} = J_{P_{\ell_i}} \dot{q}, \quad \boldsymbol{\omega}_i = J_{O_{\ell_i}} \dot{q} \tag{4.6}$$

Where $\mathbf{J_P}$ and $\mathbf{J_O}$ are the Jacobian matrices for translational and rotational velocity, respectively.

By substituting the velocity expressions from **(4.6)** into the kinetic energy equation **(4.5)**, we obtain:

$$\mathcal{T}_{\ell_i} = \frac{1}{2} m_{\ell_i} \dot{q}^T J_{P_{\ell_i}}^T J_{P_{\ell_i}} \dot{q} + \frac{1}{2} \dot{q}^T J_{O_{\ell_i}}^T \mathbf{R}_i \mathbf{I}_{\ell_i}^i \mathbf{R}_i^T J_{O_{\ell_i}} \dot{q}. \tag{4.7}$$

The derivation of the kinetic energy of motors follows a similar approach, incorporating the mass and inertia of the motor rotors.

By substituting the velocity expressions, the kinetic energy of Rotor i can be rewritten as:

$$\mathcal{T}_{m_i} = \frac{1}{2} m_{m_i} \dot{q}^T \mathbf{J}_{P_{m_i}}^T \mathbf{J}_{P_{m_i}} \dot{q} + \frac{1}{2} \dot{q}^T \mathbf{J}_{O_{m_i}}^T \mathbf{R}_{m_i} \mathbf{I}_{m_i} \mathbf{R}_{m_i}^T \mathbf{J}_{O_{m_i}} \dot{q}. \tag{4.8}$$

where $\mathbf{R}_{m_i}$ is the rotation matrix of the rotor.

Finally, by summing the various contributions relative to the single links (7.20) and single rotors (7.30) as in (7.4), the total kinetic energy of the manipulator with actuators is given by the quadratic form:

$$\mathcal{T} = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} b_{ij}(\mathbf{q}) \dot{q}_i \dot{q}_j = \frac{1}{2} \dot{\mathbf{q}}^T \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}}. \tag{4.9}$$

where

$$\begin{aligned} \mathbf{M}(\mathbf{q}) = \sum_{i=1}^{n} \Bigg(& m_{\ell_i} \mathbf{J}_P^{(\ell_i)T} \mathbf{J}_P^{(\ell_i)} + \mathbf{J}_O^{(\ell_i)T} \mathbf{R}_i \mathbf{I}_{\ell_i}^i \mathbf{R}_i^T \mathbf{J}_O^{(\ell_i)} \\ & + m_{m_i} \mathbf{J}_P^{(m_i)T} \mathbf{J}_P^{(m_i)} + \mathbf{J}_O^{(m_i)T} \mathbf{R}_{m_i} \mathbf{I}_{m_i}^{m_i} \mathbf{R}_{m_i}^T \mathbf{J}_O^{(m_i)} \Bigg) \end{aligned} \tag{4.10}$$

is the $(n \times n)$ *inertia matrix* which is:

- *symmetric*,
- *positive definite*,
- *(in general) configuration-dependent*.

4.2.2 Potential Energy

The total potential energy is determined by the sum of the contributions relative to each link as well as to each rotor.

$$\mathcal{U} = \sum_{i=1}^{n} \left(\mathcal{U}_{\ell_i} + \mathcal{U}_{m_i} \right). \tag{4.11}$$

The contribution due only to gravitational forces is expressed by

$$\mathcal{U}_{\ell_i} = -m_{\ell_i} \mathbf{g}_0^T \mathbf{p}_{\ell_i} \tag{4.12}$$

As regards the contribution of Rotor i,

$$\mathcal{U}_{m_i} = -m_{m_i} \mathbf{g}_0^T \mathbf{p}_{m_i}. \tag{4.13}$$

By substituting equations (4.12) and (4.13) into (4.11), the potential energy can be expressed as follows.

$$\mathcal{U} = -\sum_{i=1}^{n} \left(m_{\ell_i} \mathbf{g}_0^T \mathbf{p}_{\ell_i} + m_{m_i} \mathbf{g}_0^T \mathbf{p}_{m_i} \right). \tag{4.14}$$

4.3 Inverse Dynamics Control

Now, let's consider the challenge of tracking a trajectory in joint space. The control strategy is based on nonlinear multivariable systems. The dynamic equation governing an n-joint manipulator is given by (8.7) and can be rewritten as:

$$\mathbf{B}(q)\ddot{q} + \mathbf{n}(q, \dot{q}) = \mathbf{u}, \tag{4.15}$$

where, for simplicity, the function $\mathbf{n}(q, \dot{q})$ has been defined as:

$$\mathbf{n}(q, \dot{q}) = \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{F}\dot{q} + \mathbf{g}(q). \tag{4.16}$$

The following control method is designed to determine a control vector $\mathbf{u}$ as a function of the systems state, ensuring a structured input-output behavior of linear nature. The goal is not merely an approximate linearization, but rather an *exact linearization* of the system dynamics achieved through *nonlinear state*

feedback. The feasibility of this approach is ensured by the intrinsic structure of the system dynamics. Specifically, equation (8.55) is linear in the control input $\mathbf{u}$, and the matrix $\mathbf{B}(q)$ is full-rank, meaning it can be inverted for any manipulator configuration.

By defining the control input $\mathbf{u}$ as a function of the system state in the form:

$$\mathbf{u} = \mathbf{B}(q)\mathbf{y} + \mathbf{n}(q, \dot{q}), \tag{4.17}$$

the system reduces to:

$$\ddot{q} = \mathbf{y}. \tag{4.18}$$

4.4 Feedback Linearization Control

The robotic system under consideration follows the Lagrangian dynamics, expressed as:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{4.19}$$

where:

- q is the N-dimensional vector of generalized coordinates (positions or angles depending on the degrees of freedom),
- $M(q)$ is the positive definite inertia matrix,
- $C(q, \dot{q})$ represents the Coriolis and centrifugal forces,
- $g(q)$ is the gravity effects vector,
- τ is the vector of control torques applied to the system.

The goal is to ensure that the system's position, velocity, and acceleration $(q, \dot{q}, \ddot{q})$ track a desired trajectory $(q_d, \dot{q}_d, \ddot{q}_d)$.

A feedback linearization approach is used to simplify the nonlinear system dynamics. This control scheme consists of an outer loop and an inner loop.

To eliminate system nonlinearities, the following control law is introduced:

$$\tau = M(q)a + n(q, \dot{q}) \tag{4.20}$$

where:

- $n(q, \dot{q}) = C(q, \dot{q})\dot{q} + g(q)$ represents the Coriolis and gravitational effects.

- a is an auxiliary control input designed to make the system behave as a double integrator.

Substituting this equation into the system dynamics:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = M(q)a + n(q,\dot{q}) \tag{4.21}$$

simplifies to:

$$\ddot{q} = a \tag{4.22}$$

This transformation allows us to design a simple control law in the inner loop.

The auxiliary control input a is chosen as:

$$a = \ddot{q}_d + K_P e + K_D \dot{e} \tag{4.23}$$

where:

- $e = q_d - q$ is the position error,
- $\dot{e} = \dot{q}_d - \dot{q}$ is the velocity error,
- K_P and K_D are $(n \times n)$ symmetric and positive definite proportional and derivative gain matrices.

Plugging this into the outer loop equation gives the final control law:

$$\tau = M(q)(\ddot{q}_d + K_P e + K_D \dot{e}) + n(q,\dot{q}) \tag{4.24}$$

This results in the error dynamics:

$$\ddot{e} + K_D \dot{e} + K_P e = 0 \tag{4.25}$$

With appropriately chosen K_P and K_D matrices, the error e converges exponentially to zero.

4.5 Model Uncertainty and Adaptive Robust Control

In real-world scenarios, the exact model parameters of the system M, C, g are often unknown. Instead, approximated estimates $\hat{M}, \hat{C}, \hat{g}$ are used:

- $\hat{M}(q) \neq M(q)$
- $\hat{C}(q, \dot{q}) \neq C(q, \dot{q})$
- $\hat{g}(q) \neq g(q)$

Thus, the applied control law becomes:

$$\tau = \hat{M}(q)(\ddot{q}_d + K_P e + K_D \dot{e}) + \hat{n}(q, \dot{q}) \tag{4.26}$$

This introduces an error term η in the system dynamics:

$$\ddot{e} + K_D \dot{e} + K_P e = \eta \tag{4.27}$$

where:

$$\eta = (I - M^{-1}\hat{M})a - M^{-1}\tilde{n} \tag{4.28}$$

with $\tilde{n} = \hat{n} - n$ representing the mismatch between the estimated and actual system dynamics.

This error is due to the model uncertainty.

To handle this, Adaptive Robust Feedback Linearization (ARFL) is introduced.

4.6 Adaptive Robust Feedback Linearization Control

The Adaptive Robust Feedback Linearization Control (ARFBL) method is an advanced approach designed to handle model uncertainties in robotic manipulators [25]. Traditional feedback linearization techniques assume a perfect knowledge of the system dynamics, allowing them to transform the nonlinear system into a linear double-integrator model. However, real-world implementations often involve unmodeled dynamics, parameter variations, and unpredictable disturbances, making it necessary to introduce robust compensatory mechanisms.

This section presents a modified robust feedback linearization control framework, which adapts to model uncertainties dynamically. The key innovation lies in the adaptive update mechanism, which modifies the compensatory control term based on Lyapunov stability principles to ensure accurate trajectory tracking without requiring predefined uncertainty bounds.

To compensate for model uncertainty, the control input a is modified to include an adaptive robust term w:

$$a = \ddot{q}_d + K_P e + K_D \dot{e} + w \tag{4.29}$$

where w is an adaptive term that counteracts the effects of uncertainty. The vector w is formulated based on the Lyapunov direct approach to ensure stability.

Defining the state vector as:

$$\xi = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{4.30}$$

the system dynamics can be rewritten in state-space form:

$$\dot{\xi} = H\xi + D(\eta - w) \tag{4.31}$$

where:

- $H = \begin{bmatrix} 0 & I \\ -K_P & -K_D \end{bmatrix}$ is $(2n \times 2n)$ matrix,
- $D = \begin{bmatrix} 0 \\ I \end{bmatrix}$ is $(2n \times n)$ matrix.

A Lyapunov function is introduced to ensure stability:

$$V(\xi) = \xi^T Q \xi \tag{4.32}$$

where Q is a positive definite matrix.

Using Lyapunovs direct method, if w is designed as:

$$w = \rho \frac{z}{\|z\|} \tag{4.33}$$

where:

- $z = D^T Q \xi$,
- ρ is a adaptive gain satisfying:

$$\rho \geq \|\eta\| \tag{4.34}$$

then stability is ensured, and the tracking error e converges to zero even in the presence of model uncertainties.

Then, the time derivative of V, i.e.,

$$\dot{V}(\xi) = \dot{\xi}^T Q\xi + \xi^T Q\dot{\xi}, \tag{4.35}$$

This leads to Equation (15), which provides a robust adaptation strategy to handle unknown uncertainties in robotic systems.

The control objective is to ensure:

$$\dot{V} \leq 0 \tag{4.36}$$

for global stability. The adaptive update law for ρ is derived based on this condition.

4.7 Updating Law for ρ in Robust Control

4.7.1 Overview

In Adaptive Robust Feedback Linearization Control (ARFBL), the adaptive gain ρ is dynamically adjusted to compensate for uncertain disturbances and modeling errors. Unlike fixed-gain robust controllers, which assume predefined upper bounds for uncertainties, the adaptive approach modifies ρ in real-time based on the systems behavior [1],[24].

The main objective of the adaptive law is to ensure that the control system remains stable and robust while avoiding excessive oscillations (chattering). This is achieved by increasing ρ when needed and preventing unnecessary growth when disturbances are minimal.

4.7.2 Mathematical Formulation of Adaptive ρ

To ensure that the adaptive gain ρ is modified effectively based on the systems dynamic behavior, an update law is formulated using Lyapunov stability principles. The fundamental idea is to ensure that ρ increases when the system encounters large uncertainties and decreases when robustness is no longer needed.

The update law is derived from the Lyapunov function (4.35):

$$\dot{V}(\xi) = \dot{\xi}^T Q\xi + \xi^T Q\dot{\xi}, \tag{4.37}$$

For stability, the system must satisfy:

$$\dot{V} \leq 0 \tag{4.38}$$

Based on this, the update law for ρ is formulated as:

$$\dot{\rho} = \begin{cases} k_\rho, & \text{if } \dot{V} > 0 \text{ and } \|z\| \geq \epsilon \\ k_\rho, & \text{if } -\epsilon_1 \leq \dot{V} \leq 0 \text{ and } \|z\| \geq \epsilon \\ -k_\rho, & \text{if } \dot{V} \leq -\delta \text{ and } \|z\| < \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{4.39}$$

where:

- k_ρ is the adaptation gain controlling the rate of change of ρ.
- $z = D^T Q E$ represents the projected system error.
- ϵ is a threshold preventing unnecessary adaptation.
- $\delta = \delta_{\text{threshold}} \cdot \|\xi\|^2$ ensures adaptive robustness scaling.
- ϵ_1, as a small positive constant, prevents unwanted oscillations in ρ.

4.7.3 Adaptive ρ Behavior Under Different Conditions

The behavior of ρ varies based on system states and disturbances:

Case 1: When Model Uncertainties Are Large

If tracking errors are significant and the system struggles to stabilize, the Lyapunov function derivative $\dot{V}$ becomes positive, indicating a need for increased robustness. If the error norm $\|z\|$ exceeds threshold ϵ, the update law increases ρ:

$$\dot{\rho} = k_\rho \tag{4.40}$$

This strengthens the control response, allowing the system to counteract uncertainties effectively.

Case 2: When the System Is Approaching Stability

If tracking errors begin to decrease and $\dot{V}$ moves toward zero, robustness is still necessary, but excessive growth should be prevented. The update law ensures that ρ is either maintained or slowly increased:

$$\dot{\rho} = k_\rho, \quad \text{if } -\epsilon_1 \leq \dot{V} \leq 0 \text{ and } \|z\| \geq \epsilon \tag{4.41}$$

Case 3: When the System Is Fully Stabilized

Once tracking errors are minimal and $\dot{V}$ becomes significantly negative, maintaining high robustness is unnecessary. In this case, ρ is gradually reduced to optimize control energy:

$$\dot{\rho} = -k_\rho, \quad \text{if } \dot{V} \leq -\delta_{\text{threshold}} \cdot \|\xi\|^2 \text{ and } \|z\| < \epsilon \tag{4.42}$$

This adjustment prevents excessive control effort while preserving stability.

Case 4: When No Significant Changes Occur

In cases where tracking errors and system states remain unchanged, modifying ρ is unnecessary. Thus, the update law keeps ρ constant:

$$\dot{\rho} = 0 \tag{4.43}$$

4.7.4 Advantages of the Adaptive Update Law

This approach offers several benefits:

- **Dynamic Robustness**: ρ increases only when needed, avoiding excessive control effort.
- **Energy Efficiency**: ρ decreases automatically when robustness is not required, minimizing energy consumption.
- **Reduced Chattering**: Thresholds ϵ and ϵ_1 prevent undesired oscillations.
- **Improved Stability**: Ensures that tracking errors remain bounded while maintaining optimal robustness.

4.7.5 Conclusion

By refining the adaptive update law, the system maintains an optimal balance between robustness and efficiency. This approach guarantees better tracking performance compared to conventional robust control methods. The next section will validate these theoretical insights through simulation results.

5

Implementation,Simulation and Analysis

5.1 Introduction

In this chapter, the simulation and implementation of two robot manipulators, SCARA and RR manipulator, are considered to compare their performance based on a single control strategy explained in Chapter 4. The objective is to analyze how these robots respond to the imposed control strategy under various operational conditions. Comparisons are made to evaluate the effectiveness and limitations of the selected approach when applied to different manipulator structures. The results contribute to a better understanding of the control strategy's applicability and provide insights for further improvement in robotic system control and design.

To ensure a realistic evaluation, two sets of parameters are defined: nominal dynamic parameters, representing the actual system, and perturbed dynamic parameters, which introduce uncertainties in the model. These perturbed parameters are intentionally incorporated to assess the robustness of the controller against variations in mass, inertia, and center of mass locations.

In practical applications, exact system parameters are rarely known due to factors such as manufacturing tolerances, wear, and external disturbances. Therefore, the controller must operate effectively despite these uncertainties. To

evaluate this, both nominal and perturbed dynamic models are used to compute the inertia matrix $M(q)$, Coriolis and centrifugal forces $C(q, \dot{q})$, and gravitational effects $G(q)$. The deviation between the estimated and actual dynamics, denoted as $\tilde{n}$, provides insights into model uncertainty and its impact on system performance.

By implementing Adaptive Robust Feedback Linearization (ARFBL), the control algorithm compensates for these uncertainties without prior knowledge of their bounds. The effectiveness of this approach is evaluated through comparative analysis of tracking accuracy, robustness to model mismatch, and the evolution of the adaptive gain ρ. The findings contribute to enhancing adaptive control strategies for robotic systems, ensuring improved stability and performance under uncertain conditions.

5.2 RR Manipulator

5.2.1 Nominal Kinematic and Dynamic Parameters

The kinematic and dynamic parameters of the RR manipulator are determined using the Denavit-Hartenberg (DH) convention. Table 5.1 presents the DH parameters, including the joint angles, link lengths, and twist angles, which define the manipulator's structure and motion.

Link	θ	d	a	α
1	θ_1	0	0.3	0
2	θ_2	0	0.15	0

Table 5.1: DH-table for RR manipulator (Nominal)

In addition to kinematic properties, the dynamic parameters of the manipulator, including mass and center of mass (C.O.M) locations for each link, are given in Table 5.2. These parameters are essential for modeling the systems motion and formulating the equations of motion used in control design.

Parameter	Symbol	Value	Unit
Mass of link 1	m_1	7.848	kg
Mass of link 2	m_2	4.49	kg
Length of link 1	l_1	0.3	m
Length of link 2	l_2	0.15	m
C.O.M of link 1	l_{c_1}	0.1554	m
C.O.M of link 2	l_{c_2}	0.0411	m
Inertia of link 1	I_1	0.176	kg m^2
Inertia of link 2	I_2	0.0411	kg m^2
Gravitational acceleration	g	9.8	m s^{-2}

Table 5.2: Nominal Dynamic parameters for the RR manipulator

To accurately model the dynamics of the 2-link RR manipulator, the system's inertia, Coriolis, and gravitational effects must be determined. The following code computes the inertia matrix $M(q)$, the Coriolis and centrifugal matrix $C(q, \dot{q})$, and the gravity vector $G(q)$ based on the manipulator's physical parameters. These matrices are essential for formulating the system's equations of motion and implementing control strategies, and they are updated in each iteration to reflect changes in the system's state.

```
% Inertia matrix M(q)
M = [I1 + I2 + m1 * lc1^2 + m2 * (l1^2 + lc2^2 + 2 * l1 * lc2 * cos(q(2))), ...
     I2 + m2 * (lc2^2 + l1 * lc2 * cos(q(2)));
     I2 + m2 * (lc2^2 + l1 * lc2 * cos(q(2))), ...
     I2 + m2 * lc2^2];

% Coriolis and centrifugal matrix C(q, q_dot)
C = [-2 * m2 * l1 * lc2 * sin(q(2)) * q_dot(2), -m2 * l1 * lc2 * sin(q(2)) * (q_dot(1) + q_dot(2));
     m2 * l1 * lc2 * sin(q(2)) * q_dot(1), 0];

% Gravity vector g(q)
G = [(m1 * lc1 + m2 * l1) * g_const * cos(q(1)) + m2 * lc2 * g_const * cos(q(1) + q(2));
     m2 * lc2 * g_const * cos(q(1) + q(2))];
```

Code 5.1: Inertia, Coriolis, and Gravity Matrices (Nominal)

5.2.2 Perturbed Kinematic and Dynamic Parameters

To evaluate the robustness of the control strategy, perturbed parameters are introduced by slightly modifying the nominal values. These perturbations account for potential uncertainties in mass, inertia, and link dimensions due to modeling errors or physical variations in the system. The updated Denavit-Hartenberg parameters and perturbed properties are presented in Table 5.3 and Table 5.4, respectively.

Link	θ	d	a	α
1	θ_1	0	0.33	0
2	θ_2	0	0.12	0

Table 5.3: DH-table for RR manipulator (Perturbed)

Similar to the nominal parameters presented in Table 5.2, the following table presents the updated dynamic parameters that account for the perturbations introduced in Table 5.4. These modified values reflect potential uncertainties in mass, inertia, and link dimensions and will be used in the subsequent calculations and control implementation.

Parameter	**Symbol**	**Value**	**Unit**
Mass of link 1	$m_{1,\text{est}}$	7.848	kg
Mass of link 2	$m_{2,\text{est}}$	4.49	kg
Length of link 1	$l_{1,\text{est}}$	0.3	m
Length of link 2	$l_{2,\text{est}}$	0.15	m
C.O.M of link 1	$l_{c1,\text{est}}$	0.1554	m
C.O.M of link 2	$l_{c2,\text{est}}$	0.0411	m
Inertia of link 1	$I_{1,\text{est}}$	0.176	kg m^2
Inertia of link 2	$I_{2,\text{est}}$	0.0411	kg m^2
Gravitational acceleration	g_{est}	9.8	m s^{-2}

Table 5.4: Dynamic parameters for the 2-link RR planar manipulator (Perturbed)

```
% Estimated Inertia matrix M_est(q)
M_est = [I1_est + I2_est + m1_est * lc1_est^2 + m2_est * (l1_est^2 +
    lc2_est^2 + 2 * l1_est * lc2_est * cos(q(2))), ...
          I2_est + m2_est * (lc2_est^2 + l1_est * lc2_est * cos(q(2)))
    ;
          I2_est + m2_est * (lc2_est^2 + l1_est * lc2_est * cos(q(2)))
    , ...
          I2_est + m2_est * lc2_est^2];

% Estimated Coriolis and centrifugal matrix C_est(q, q_dot)
C_est = [-2 * m2_est * l1_est * lc2_est * sin(q(2)) * q_dot(2), -
    m2_est * l1_est * lc2_est * sin(q(2)) * (q_dot(1) + q_dot(2));
           m2_est * l1_est * lc2_est * sin(q(2)) * q_dot(1), 0];

% Estimated Gravity vector g_est(q)
G_est = [(m1_est * lc1_est + m2_est * l1_est) * g_const * cos(q(1)) +
     m2 * lc2 * g_const * cos(q(1) + q(2));
          m2_est * lc2_est * g_const * cos(q(1) + q(2))];
```

Code 5.2: Estimated Inertia, Coriolis, and Gravity Matrices

5.2.3 Desired Trajectory

In order to evaluate the tracking performance of the control strategy, a reference trajectory is designed for each joint of the manipulator. The desired trajectory for the i-th joint, denoted as $q_i^d(t)$, is generated as the sum of multiple sinusoidal components with randomly selected frequencies, ensuring a sufficiently rich excitation for controller evaluation. The trajectory is defined as follows:

$$q_i^d(t) = A \sum_{i=1}^{N} \sin(\omega_i t), \tag{5.1}$$

where:

- A is the amplitude of the sinusoidal components.
- N is the total number of sinusoids used in the trajectory.
- ω_i represents the angular frequency of the i-th sinusoid, which is randomly selected from a uniform distribution:

$$\omega_i \sim U([\omega_{\min}, \omega_{\max}]), \tag{5.2}$$

with $\omega_{\min} = \pi$ rad/s and $\omega_{\max} = 3\pi$ rad/s.

For implementation, the frequency values ω_i are sampled randomly within the defined range for each joint.

```
% Desired Trajectory
A = 1; N = 6;
omega_min = pi; omega_max = 3 * pi;

% Randomly generating frequency components
omega1 = omega_min + (omega_max - omega_min) * rand(1, N);
omega2 = omega_min + (omega_max - omega_min) * rand(1, N);

% Computing the desired trajectory for each joint
qd1 = A * sum(sin(omega1' .* t), 1);
qd2 = A * sum(sin(omega2' .* t), 1);
```

Code 5.3: MATLAB implementation of the desired trajectory

This approach ensures a diverse range of frequency components in the trajectory, making it suitable for testing the systems response under dynamic conditions. Below, an example of the desired trajectory is provided.

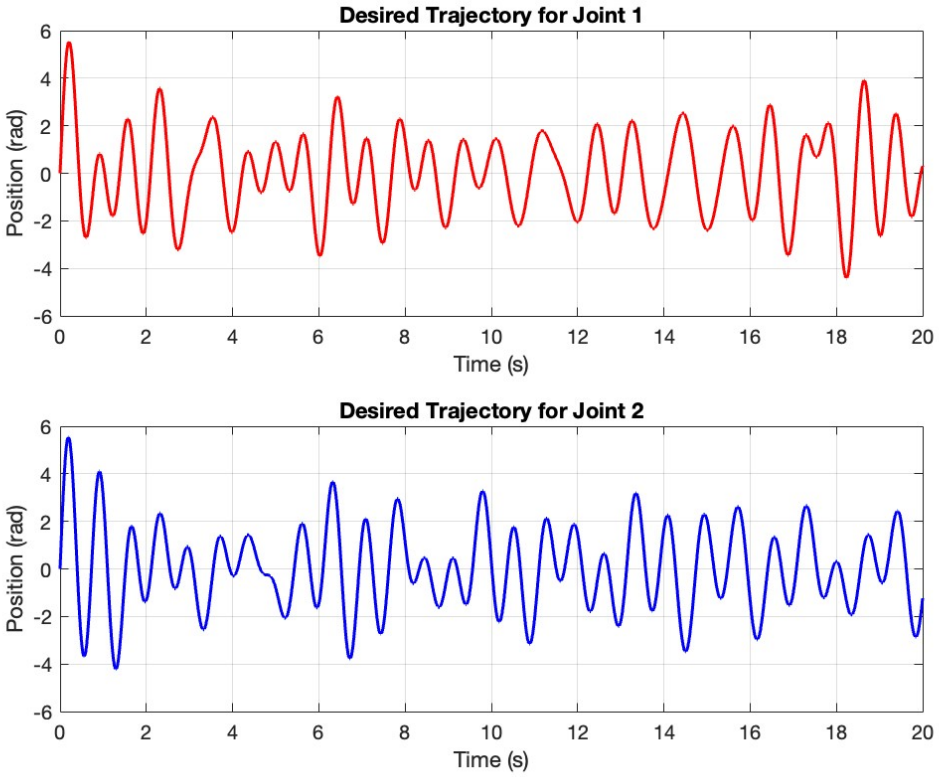

Figure 5.1: Desired trajectories for Joint 1 and Joint 2

5.2.4 Feedback Linearization Control

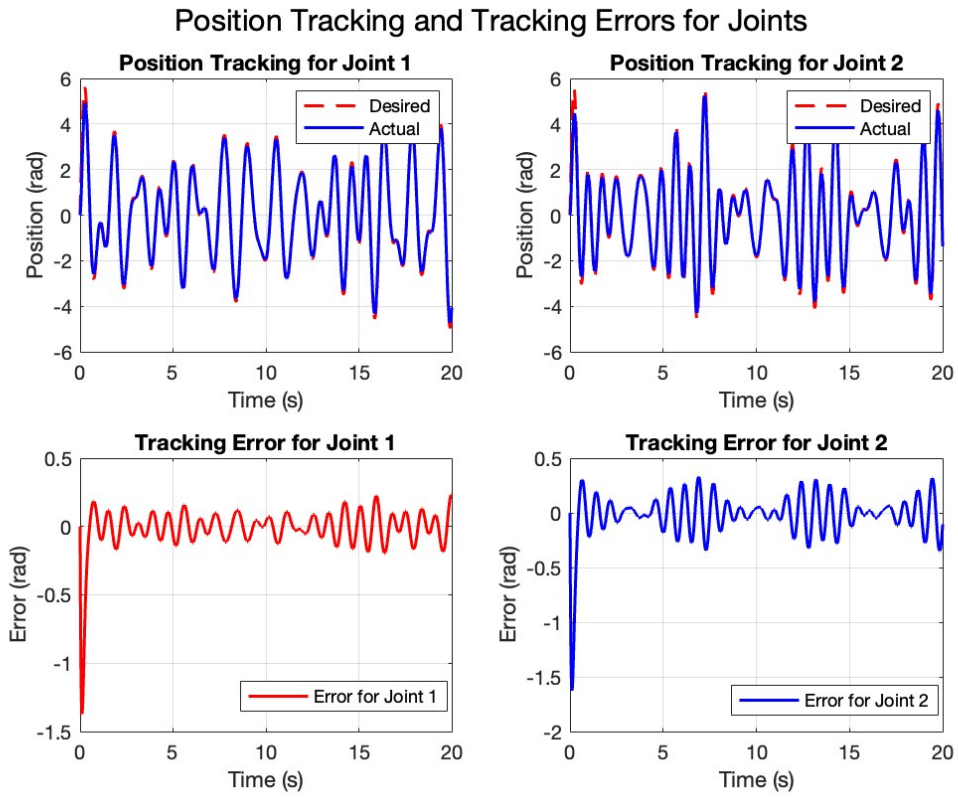

Figure 5.2: Feedback linearization control

Figure 5.2 illustrates the performance of the feedback linearization control strategy applied to the robotic system. The control gains were selected as:

$$K_p = \begin{bmatrix} 100 & 0 \\ 0 & 100 \end{bmatrix} \tag{5.3}$$

and

$$K_d = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{5.4}$$

to ensure a well-tuned system response. As shown in Figure 5.2, while the tracking performance is acceptable, there are persistent oscillations and minor deviations between the desired and actual trajectories. The tracking error does not fully vanish and remains within a bounded range, indicating that the system is sensitive to modeling inaccuracies.

The results validate the effectiveness of feedback linearization in compensating for the nonlinear dynamics of the system. By appropriately selecting the control gains, the tracking performance improves, leading to minimal steady-state error and smooth joint torques. However, since the feedback linearization approach assumes an accurate model of the system, any uncertainties or modeling errors may degrade the tracking performance. In the following section, adaptive robust feedback linearization will be investigated to evaluate its effectiveness in handling parametric uncertainties and external disturbances.

5.2.5 Adaptive Robust Feedback Linearization Control

Adaptive Robust Feedback Linearization (ARFBL) extends the conventional Feedback Linearization control method by incorporating an adaptive gain parameter (ρ) to effectively handle system uncertainties. This approach ensures improved robustness and adaptability in dynamic environments.

For $\epsilon = 0.05$

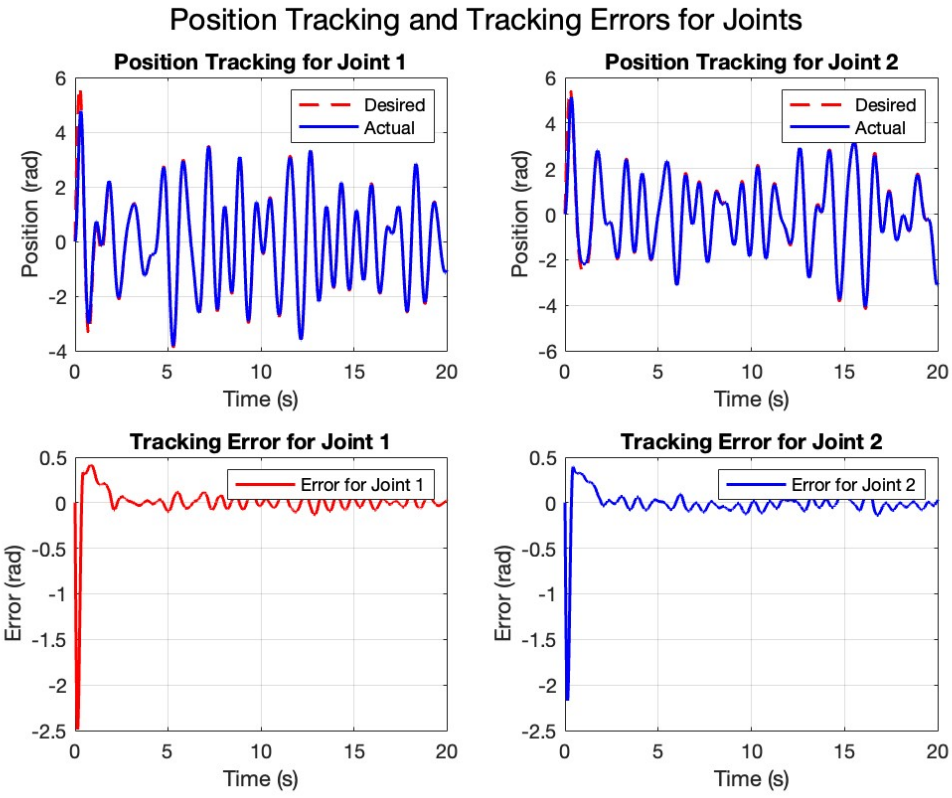

Figure 5.3: Adaptive Robust control

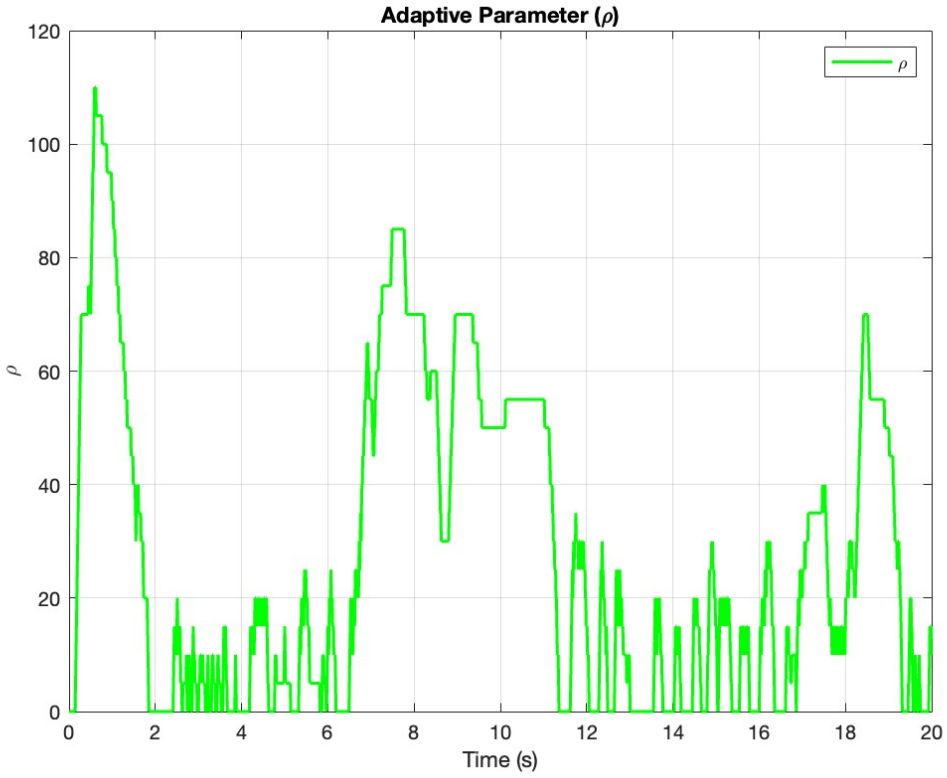

Figure 5.4: Time evolution of the adaptive parameter ρ

At the beginning, the system experiences a significant tracking error due to the presence of uncertainties and external disturbances. This can be observed in the position tracking plots (Figure 5.3), where the actual trajectory initially deviates substantially from the desired one. In response, the adaptive controller rapidly increases the adaptation parameter (ρ), as shown in Figure 5.4, to compensate for the unknown variations in the system dynamics.

As the adaptive control mechanism takes effect, the tracking error begins to decrease, stabilizing the motion of the joints. This is reflected in the tracking error plots (Figure 5.3), where the error gradually converges toward zero. Consequently, as the error reduces and the system behavior aligns with the desired trajectory, the need for aggressive adaptation diminishes. As a result, the controller automatically decreases ρ over time, ensuring that unnecessary parameter updates are avoided. This adaptive regulation prevents excessive control effort and computational overhead while maintaining robustness.

The evolution of ρ in Figure 5.4 further highlights this process. Initially, ρ exhibits large variations as the controller actively compensates for the uncertainties. However, once the system reaches a more stable state, ρ fluctuates at much lower magnitudes, indicating that only minor adjustments are needed to maintain performance. Once the error reduces, the adaptive parameter also decreases, since less correction is needed. This process ensures that the system remains both stable and efficient throughout the operation. By dynamically adjusting ρ based on tracking performance, the system achieves an optimal balance between adaptability and control efficiency, avoiding unnecessary adaptations when the system is already performing well.

5.3 SCARA

5.3.1 Nominal Kinematic and Dynamic Parameters

The kinematic and dynamic parameters of the SCARA manipulator are determined using the Denavit-Hartenberg (DH) convention, similar to how it was done for the RR manipulator. Table 5.5 presents the DH parameters, including the joint angles, link lengths, and twist angles, which define the manipulators structure and motion.

Link	θ	d	a	α
1	θ_1	0	0.3	0
2	θ_2	0	0.15	0
3	0	d_3	0	0
4	θ_4	0	0.1	0

Table 5.5: DH-table for SCARA manipulator (Nominal)

In addition to kinematic properties, the dynamic parameters of the manipulator, including mass and center of mass (C.O.M) locations for each link, are given in Table 5.6. These parameters are essential for modeling the system's motion and formulating the equations of motion used in control design.

Parameter	**Symbol**	**Value**	**Unit**
Mass of link 1	m_1	7.8	kg
Mass of link 2	m_2	4.5	kg
Mass of link 3	m_3	3.0	kg
Mass of link 4	m_4	2.0	kg
Length of link 1	l_1	0.3	m
Length of link 2	l_2	0.15	m
Length of link 3 (prismatic)	l_3	0.2	m
Length of link 4	l_4	0.1	m
C.O.M of link 1	lc_1	0.1554	m
C.O.M of link 2	lc_2	0.0341	m
C.O.M of link 3	lc_3	0.1	m
C.O.M of link 4	lc_4	0.05	m
Inertia of link 1	I_1	0.176	kg m^2
Inertia of link 2	I_2	0.0411	kg m^2
Inertia of link 3 (prismatic)	I_3	0.0	kg m^2
Inertia of link 4	I_4	0.01	kg m^2
Gravitational acceleration	g	9.8	m s^{-2}

Table 5.6: Nominal Dynamic Parameters for the 4-DOF SCARA Manipulator

To accurately model the dynamics of the 4-DOF SCARA manipulator, the systems inertia, coriolis, and gravitational effects must be determined. The following computation defines the inertia matrix $M(q)$, the Coriolis and centrifugal matrix $C(q, \dot{q})$, and the gravity vector $G(q)$ based on the manipulators physical properties.

These matrices are essential for formulating the systems equations of motion and designing effective control strategies. They are updated iteratively to adapt to changes in the systems state and ensure accurate trajectory tracking.

```
% Inertia matrix M(q) for 4 DOF
M = [I1+I2+I3+m2*l1^2+m3*(l1^2+l2^2)+m4*(l1^2+l2^2+l3^2), I2+I3+m3*l2^2+m4*(l2^2+l3^2), 0, 0;
     I2+I3+m3*l2^2+m4*(l2^2+l3^2), I3+m3*l2^2+m4*(l2^2+l3^2), 0, 0;
     0, 0, m3, 0;
     0, 0, 0, I4];

% Coriolis and centrifugal matrix C(q, q_dot)
C = [ -2 * m2 * l1 * lc2 * sin(q(2)) * q_dot(2), -m2 * l1 * lc2 * sin(q(2)) * (q_dot(1) + q_dot(2)), 0, 0;
       m2 * l1 * lc2 * sin(q(2)) * q_dot(1), 0, 0, 0;
       0, 0, 0, 0;
       0, 0, 0, 0];

% Gravity vector G(q)
G = [ (m1*lc1 + m2*l1 + m3*(l1+l2) + m4*(l1+l2+l3)) * g_const * cos(q(1));
      (m2*lc2 + m3*l2 + m4*(l2+l3)) * g_const * cos(q(1) + q(2));
      m3 * g_const + m4 * g_const;
      0];
```

Code 5.4: Inertia, Coriolis, and Gravity Matrices for SCARA (Nominal)

5.3.2 Perturbed Kinematic and Dynamic Parameters

To evaluate the robustness of the control strategy, perturbed parameters are introduced by slightly modifying the nominal values. These perturbations account for potential uncertainties in mass, inertia, and link dimensions due to modeling errors or physical variations in the system. The updated Denavit-Hartenberg parameters and dynamic properties are presented in Table 5.7 and Table 5.8, respectively.

Link	θ	d	a	α
1	θ_1	0	0.32	0
2	θ_2	0	0.14	0
3	0	d_3	0	0
4	θ_4	0	0.12	0

Table 5.7: DH-table for perturbed SCARA manipulator

Similar to the nominal parameters presented in Table 5.6, the following table presents the updated dynamic parameters that account for the perturbations

introduced in Table 5.7. These modified values reflect potential uncertainties in mass, inertia, and link dimensions and will be used in the subsequent calculations and control implementation.

Parameter	Symbol	Value	Unit
Mass of link 1	$m_{1,est}$	8.0	kg
Mass of link 2	$m_{2,est}$	4.3	kg
Mass of link 3 (prismatic)	$m_{3,est}$	3.2	kg
Mass of link 4 (wrist)	$m_{4,est}$	2.1	kg
Length of link 1	$l_{1,est}$	0.32	m
Length of link 2	$l_{2,est}$	0.14	m
Length of link 3 (prismatic)	$l_{3,est}$	0.18	m
Length of link 4 (wrist)	$l_{4,est}$	0.12	m
C.O.M of link 1	$lc_{1,est}$	0.16	m
C.O.M of link 2	$lc_{2,est}$	0.035	m
C.O.M of link 3	$lc_{3,est}$	0.09	m
C.O.M of link 4	$lc_{4,est}$	0.045	m
Inertia of link 1	$I_{1,est}$	0.18	kg m^2
Inertia of link 2	$I_{2,est}$	0.04	kg m^2
Inertia of link 3 (prismatic)	$I_{3,est}$	0.0	kg m^2
Inertia of link 4 (wrist)	$I_{4,est}$	0.012	kg m^2
Gravitational acceleration	g_{est}	9.8	m s^{-2}

Table 5.8: Dynamic parameters for the perturbed 4-DOF SCARA manipulator

```
% Estimated Inertia matrix M_est(q)
M_est = [I1_est + I2_est + I3_est + m2_est*l1_est^2 + m3_est*(l1_est^2 + l2_est^2) + ...
      m4_est*(l1_est^2 + l2_est^2 + l3_est^2), I2_est + I3_est + m3_est*l2_est^2 + ...
      m4_est*(l2_est^2 + l3_est^2), 0, 0;
      I2_est + I3_est + m3_est*l2_est^2 + m4_est*(l2_est^2 + l3_est^2), ...
      I3_est + m3_est*l2_est^2 + m4_est*(l2_est^2 + l3_est^2), 0, 0;
      0, 0, m3_est, 0;
      0, 0, 0, I4_est];

% Estimated Coriolis and centrifugal matrix C_est(q, q_dot)
C_est = [ -2 * m2_est * l1_est * lc2_est * sin(q(2)) * q_dot(2), ...
         -m2_est * l1_est * lc2_est * sin(q(2)) * (q_dot(1) + q_dot(2)), 0, 0;
          m2_est * l1_est * lc2_est * sin(q(2)) * q_dot(1), 0, 0, 0;
```

```
            0, 0, 0, 0;
            0, 0, 0, 0];

% Estimated Gravity vector G_est(q)
G_est = [(m1_est*lc1_est + m2_est*l1_est + m3_est*(l1_est + l2_est) + ...
          m4_est*(l1_est + l2_est + l3_est)) * g_const * cos(q(1));
         (m2_est*lc2_est + m3_est*l2_est + m4_est*(l2_est + l3_est)) * g_const * cos(q(1) + q(2));
         m3_est * g_const;
         0];
```

Code 5.5: Inertia, Coriolis, and Gravity Matrices for SCARA (Perturbed)

5.3.3 Adaptive Robust Feedback Linearization for SCARA

As observed in the RR manipulator, Adaptive Robust Feedback Linearization (ARFBL) enhances the conventional Feedback Linearization method by introducing an adaptive gain parameter (ρ), allowing the system to respond dynamically to uncertainties. Applying this approach to the SCARA manipulator, we aim to compensate for variations in mass, inertia, and unmodeled dynamics, ensuring stable and precise motion control.

Unlike in the RR manipulator, where the control strategy mainly handled revolute joints, the SCARA system introduces additional complexity due to its mixed revolute and prismatic joints. Thus, the adaptive gain ρ is adjusted accordingly to maintain robustness in both rotational and translational movements.

- The adaptive gain ρ dynamically adjusts to account for varying joint dynamics in SCARA.
- The approach compensates for uncertainties in both revolute and prismatic joint interactions.
- Improved disturbance rejection is achieved in hybrid motion control scenarios.

The control gains were selected as:

$$K_p = \begin{bmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}$$

and

$$K_d = \begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 30 & 0 \\ 0 & 0 & 0 & 30 \end{bmatrix}$$

to ensure a well-tuned system response.

For $\epsilon = 0.05$,

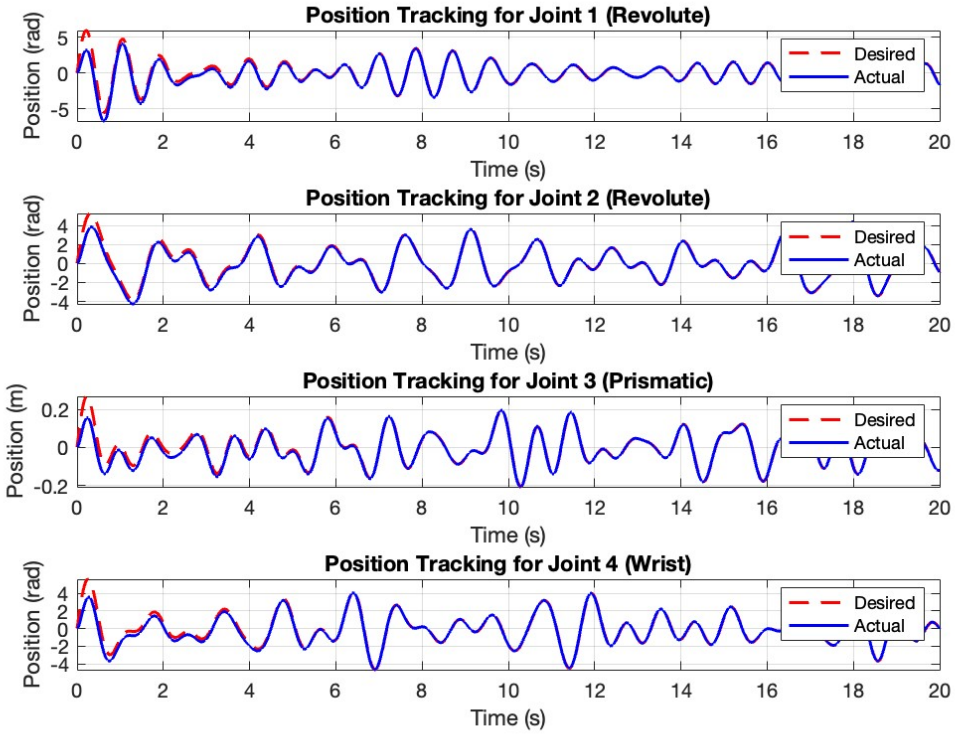

Figure 5.5: Adaptive Control Performance for SCARA Joint Tracking

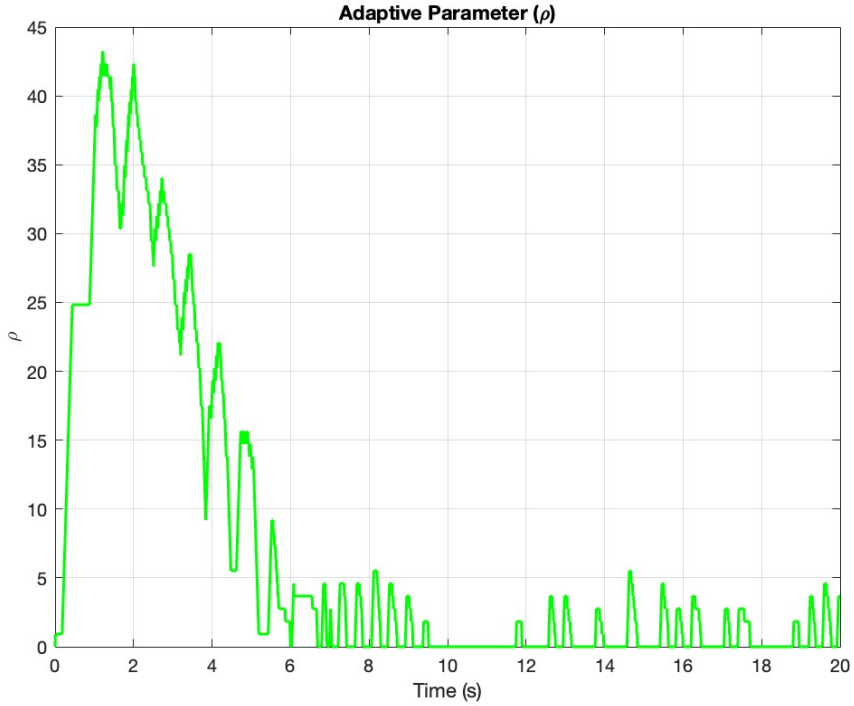

Figure 5.6: Time evolution of the adaptive parameter ρ

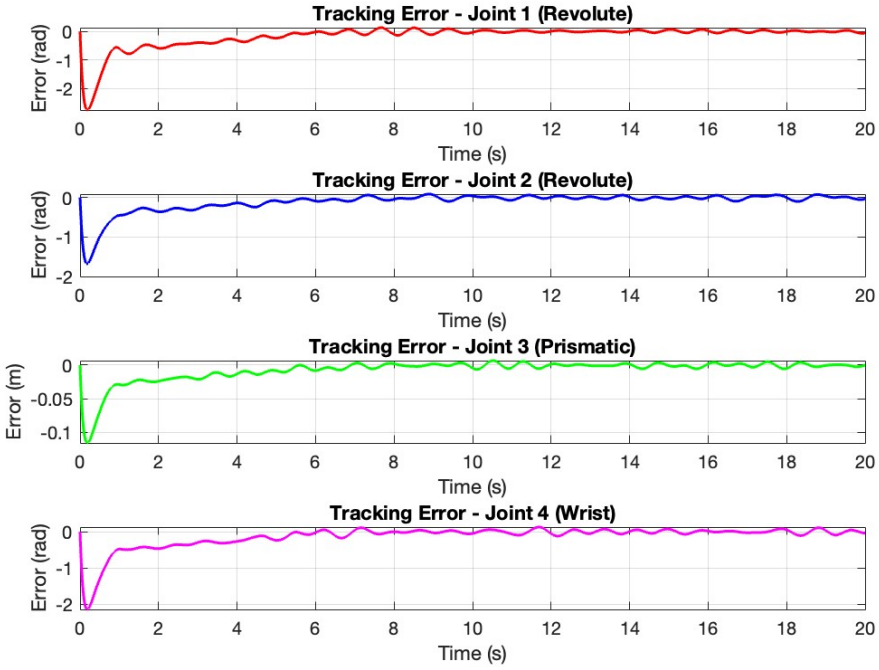

Figure 5.7: Time evolution of the adaptive parameter ρ

Similar to the analysis in Section 5.2.5, at the beginning of the operation, the system exhibits a significant tracking error due to initial uncertainties and external disturbances, as observed in the tracking error plots (Figure 5.7). This deviation between the desired and actual trajectories can also be seen in the position tracking plots (Figure 5.5), where all joints initially fail to accurately follow the reference path. In response, the adaptive control mechanism increases the adaptation gain parameter ρ sharply, as shown in Figure 5.6, to counteract the unknown variations in the system dynamics.

As the adaptation takes effect, the tracking error for each joint gradually decreases, indicating that the controller successfully compensates for system uncertainties. This convergence is clearly visible in the error curves, where each joints error steadily approaches zero. As a result, the controller begins to reduce the adaptation parameter ρ over time, minimizing unnecessary adaptation and control effort. This dynamic adjustment ensures efficient and stable trajectory tracking.

Eventually, all joints closely follow the desired trajectory with minimal deviation, as seen in the later phases of the position tracking plots. The decline in both the tracking error and ρ illustrates that the adaptive controller achieves a balance between robustness and efficiency, adapting only when needed and maintaining performance once the system stabilizes.

6 Conclusion

6.1 Summary of Key Findings

This paper thoroughly explores common problems in the control of robotic manipulators under uncertain conditions, particularly in dynamic environments. Special attention has been given to developing adaptive robust control strategies to enhance the precision and stability of these robots.

In Chapter 4, an adaptive control algorithm is presented in which the adaptation parameter ρ is dynamically adjusted based on the systems tracking performance. Specifically, the updating law is designed such that ρ increases when the tracking error grows, allowing the controller to respond more aggressively to disturbances or uncertainties. However, once the actual trajectory begins to follow the desired path within a predefined error threshold, the algorithm starts to reduce ρ. This strategy aims to minimize unnecessary adaptation effort, ensuring that parameter updates occur only when needed, thereby improving both the efficiency and robustness of the controller.

In Chapter 5, the control strategies introduced in Chapter 4 were validated through simulation studies conducted on two different robotic manipulator configurations: a two-link RR planar manipulator and a SCARA manipulator. The results obtained from these simulations were thoroughly analyzed. The simulation plots clearly demonstrated the controllers ability to maintain accurate trajectory tracking despite the presence of modeling inaccuracies and external

disturbances.

These results emphasize the relevance of adaptability in robotic control systems, especially in applications requiring consistent performance under uncertain and dynamic conditions. The findings suggest that the proposed adaptive robust feedback linearization controller can offer a reliable approach to handling various types of uncertainties across different manipulator structures. Moreover, this study may serve as a useful reference for future research aiming to explore or refine adaptive control methods in both academic and applied robotics contexts.

6.2 Future Works

Future research can entail the integration of machine learning algorithms, for improving dynamic uncertainty estimation in the adaptive control scheme. In addition, more complex adaptation laws can be researched for additional robustness to partially unmodeled and time-varying dynamics without sacrificing system stability.

References

[1] Mohamed Mahmoud Abdelwahab et al. "Adaptive Robust Controller for Handling Unknown Uncertainty of Robotic Manipulators". In: *arXiv preprint arXiv:2406.14338* (2024). URL: https://doi.org/10.48550/arXiv.2406.14338.

[2] J. Angeles. "Kinematics and Dynamics of Robotic Manipulators". In: *IEEE Transactions on Robotics and Automation* 16.3 (2000), pp. 281–289. DOI: 10.1109/70.849428.

[3] J. Angeles. *Robotic Manipulators: Analysis and Control*. London: Springer, 2000.

[4] A. Astolfi and R. Ortega. "A New Adaptive Control Design for Robotic Manipulators: Applications to Adaptive Robust Control". In: *IEEE Transactions on Automatic Control* 48.7 (2003), pp. 1153–1157. DOI: 10.1109/TAC.2003.815126.

[5] Stephen Barnett. *Matrix Methods for Engineers and Scientists*. McGraw-Hill, 1979.

[6] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. 3rd. Pearson Education, 2005.

[7] J. Denavit and R. S. Hartenberg. "A kinematic notation for lower pair mechanisms". In: *Journal of Applied Mechanics* 22 (1955), pp. 215–221.

[8] R. Featherstone. *Rigid Body Dynamics*. Springer, 2008.

[9] Gene F. Franklin, J. Da Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, 1994. ISBN: 9780201503325.

[10] R. C. Goertz. "Fundamentals of general-purpose remote manipulators". In: *Nucleonics* 10.11 (1952), pp. 36–42.

[11] R. C. Goertz. "Mechanical masterslave manipulator". In: *Nucleonics* 12.11 (1954), pp. 45–46.

[12] Raymond Goertz. *Goertz Early Manipulator*. Image retrieved from `https://historyimages.com/goertz-early-manip`. 1950.

[13] J. J. Hah and J. Ploeg. *The Robotics of Manipulation*. Cambridge University Press, 2014.

[14] Yichi Huang et al. "Robust adaptive control for robotic system with external disturbance and guaranteed parameter estimation". In: *IFAC PapersOnLine* 55.38 (2022), pp. 178–183. DOI: `10.1016/j.ifacol.2022.10.390`.

[15] P. A. Ioannou and J. Sun. *Robust Adaptive Control*. Prentice Hall, 1996.

[16] R. N. Jazar. *Vehicle Dynamics: Theory and Application*. Springer, 2008.

[17] Rafael Kelly, Víctor Santibañez, and Antonio Loria. *Control of Robot Manipulators in Joint Space*. Springer, 2005. DOI: `10.1007/b138235`.

[18] Hassan K. Khalil. *Nonlinear Systems*. 3rd. Upper Saddle River, NJ: Prentice Hall, 2015.

[19] D.G. Luenberger. *Stability and Control of Dynamical Systems with Applications*. John Wiley Sons, 1979. ISBN: 9780471024247.

[20] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. 1st. USA: Cambridge University Press, 2017.

[21] Richard M. Murray, Zexiang Li, and Shankar S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[22] N.S. Nise. *Fundamentals of Linear Control*. Wiley, 2011. ISBN: 9780470617004.

[23] H. G. Sage, M. F. De Mathelin, and E. Ostertag. *Robust control of robot manipulators: A survey*. Vol. 72. 16. International Journal of Control, 1999, p. 14981522.

[24] Bruno Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer, 2009.

[25] Jean-Jacques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice-Hall, 1991.

[26] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2006.

[27] H. Zhou and Z. Li. *Robot Manipulator Control: Theory and Practice*. Springer, 2010.

Acknowledgments

I would like to express my heartfelt gratitude to my thesis advisor, Prof. Carli Ruggero, for their invaluable support, guidance, and encouragement throughout this research. His insightful advice and academic expertise were fundamental in shaping the direction and quality of my work.

A debt of gratitude is also owed to my family and friends, with special thanks to my girlfriend, Ebru, whose unwavering support, love, and belief in me have been a constant source of strength throughout this academic journey.