Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Model Reference Adaptive Control of PUMA 560 Robot Manipulator for High Speed Operations

A thesis submitted to School of Graduate Studies, Addis Ababa Institute of Technology, Addis Ababa University in partial fulfillment of the requirement for the Degree of Master of Science in Electrical Engineering (Control Engineering)

By

Tsenat Abebe

Advisor

Dereje Shiferaw (PhD)

May 5, 2025

Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Model Reference Adaptive Control of PUMA 560 Robot Manipulator for High Speed Operations

By

Tsenat Abebe

Approved by Examining Board of:

School Dean	Signature	Date
Advisor	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Declaration

I hereby certify that the work presented in this thesis, titled "Model Reference Adaptive Control (MRAC) of PUMA 560 Robot Manipulator for High Speed Operations", was written by me under the direction of my advisor, that it is original work of mine, save where otherwise noted in the text, and that it has not been submitted in whole or in part for any other degree or professional certification at any other institution.

Author

Tsenat Abebe

Signature

_____

Acknowledgment

First and foremost, I want to express my gratitude to the Almighty GOD for being a source of strength and guidance and for giving me the faith and bravery to persevere even in trying circumstances. My thesis advisor, Dr. Dereje Shiferaw, who helped me at every level of thesis preparation, has my sincere gratitude. His expertise in robotics has been really helpful to my work. His assistance, encouragement, insightful suggestions, and numerous great remarks were essential to the completion of this thesis. I sincerely appreciate his advice and our several insightful conversations. Last but not least, I want to express my gratitude to my parents and friends for their unwavering support, inspiring words, and thoughtful comprehension.

Abstract

Robotic pick-and-place systems bring remarkable speed and precision to manufacturing lines. Adaptive control is one of the effective approaches for designing controllers for mechanical robot manipulators, particularly in addressing the nonlinearities and uncertainties inherent in robot dynamic models. When the robotic end-effector encounters varying masses and load conditions, the motion control directly affects the overall performance, stability, and speed of the robot. To address these challenges, this paper presents the design of a controller for a second-order system using a Model Reference Adaptive Control (MRAC) scheme, where the adaptive mechanism and controller design are based on the Lyapunov method. A Lyapunov candidate function is employed both to derive the adaptation law and to guarantee the stability of the system.

The tracking error performance of the MRAC is demonstrated across all six joint positions of the PUMA 560 robot. For Joint 1, the error exhibits an initial peak magnitude of 2×10^{-5} rad under disturbance, settling to near-zero within 2 seconds, while the undisturbed error remains negligible. Similarly, Joint 2 shows a slightly lower peak error of 1.8×10^{-5} rad, with steady-state achieved in about 2.5 seconds. Joint 3 experiences a slightly higher peak error of 2.2×10^{-5} rad and takes 3 seconds to settle. Joint 4 exhibits the highest peak error of 2.5×10^{-5} rad, with a settling time of approximately 3.5 seconds, demonstrating the greatest sensitivity to disturbance among the joints. Joint 5 and Joint 6 show peak error magnitudes of 2×10^{-5} rad and 2.3×10^{-5} rad, respectively, both reaching steady-state within 3 seconds.

Keywords: Adaptive control, Lyapunov, MRAC, Puma 560 Robot Manipulator.

Contents

List of Figures

List of Tables

List of Acronyms

3D Three-dimensional

API Application Programming Interface

CAD Computer-Aided Design assembly

DC Direct Current

DCAL Desired Compensation Adaption Law

DH Denavit-Hartenberg

DOF Degree of freedom

MIMO Multiple Input, Multiple Output

MIT Massachusetts Institute of Technology

MRAC Model Reference Adaptive Control

PD Proportional-Derivative

PID Proportional-Integral-Derivative

PUMA Programmable Universal Machine for Assembly

QP Quadratic Programming

RRR Articulated or Anthropomorphic manipulator

RLS Recursive Least Square

SMC sliding mode control

STL Stereolithography

URDF Unified Robot Description Format

XML extensible Markup Language

Chapter 1

Introduction

1.1 Background

Robot manipulators refer to mechanical devices that use electricity and require human dexterity to carry out a variety of functions. They are versatile manipulators with three or more axes that are autonomously operated, re-programmable, self-centered, and made up of electronic, electrical, or mechanical components. Worldwide, industrial robot usage has greatly expanded with a quickening trend. The majority of the time, these robots are used for underwater operations, material handling, welding, painting, part assembly, packaging, and other tasks.

Robots have five fundamental components: brain, body, actuator, sensors, and power source supply. The brain controls the robot's actions in response to desired and actual inputs. The robot body is the physical structure that holds all parts together. Actuators allow the robot to move based on electrical components and mechanical parts. Sensors provide the robot with information about both its internal and external environment, and the power source supplies energy to all robot parts [1].

Mobile robots differ from other robots in that they are capable of autonomous mobility and possess sufficient intelligence to react and take appropriate actions in response to cues from their environment. To respond to a changing environment, mobile robots require a data source, a way to decode that data, and a way to act, including mobility. Mobile robots are one of the fastest-growing scientific fields. Humans could be replaced by mobile

robots in a number of occupations. Examples include applications in transportation, healthcare, industrial automation, construction, and several other industrial and non-industrial sectors.

The specific motion of links in any mechanism or machine can be defined as the degree of freedom (DOF). To execute specific tasks, the degree of freedom always plays a key role. The total number of degrees of freedom equals the number of independent displacements of links. As it is known, a 6DOF robot manipulator is fundamental to executing tasks in 3-dimensional space.

A robot manipulator known as a revolute (RRR) is referred to as an articulated or anthropomorphic manipulator if it has all three joints. An anthropomorphic manipulator has shoulder, waist, and elbow joints, similar to a human arm. The workstation of this type of robot is highly intricate, with a crescent-shaped cross-section, capable of sweeping the volume enclosed by a sphere-shaped outer surface and an interior surface composed of scallops, constrained by joints. Such manipulators typically have six joints: the first three articulate in the x-y-z axes, and the latter three in pitch, yaw, and roll.

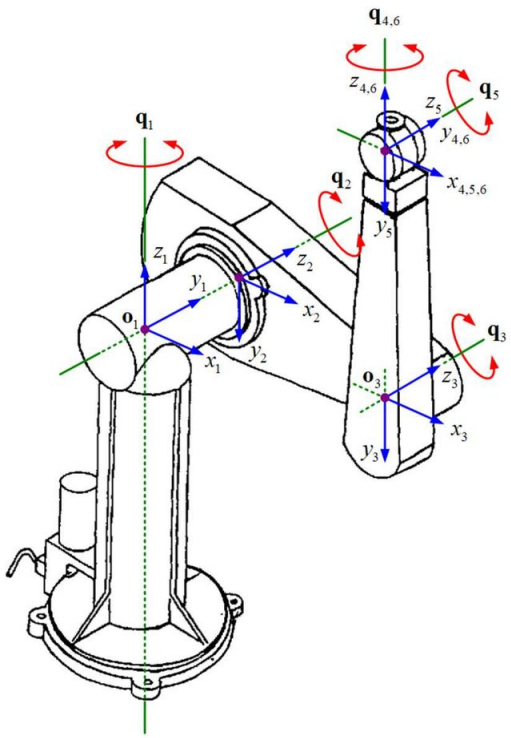


Figure 1.1: The PUMA 560 Robot Manipulator

PUMA 560 robot manipulator given in Figure 1.1 are widely used in handling small objects or parts in industrial applications due to their compact design, high speed ratio, repeatability, and flexibility. They are capable of executing the most complicated applications, such as the assembly of intricate parts.

The PUMA 560 robotic manipulator is a serial, personal computer controlled manipulator designed for industrial applications. It features six articulating joints and degrees of freedom (DOF), with three principal axes of motion (x, y, and z). The coordination of the first three joints, referred to as the waist (joint 1), shoulder (joint 2), and elbow (joint 3), enables the robot to move the end effector into any position. The end effector, often a device attached to the end of the robotic arm, requires accurate orientation when moving toward its final position. This results in six degrees of freedom, as shown in Figure 1.2, where the name of each joint with its maximum rotational range is indicated.

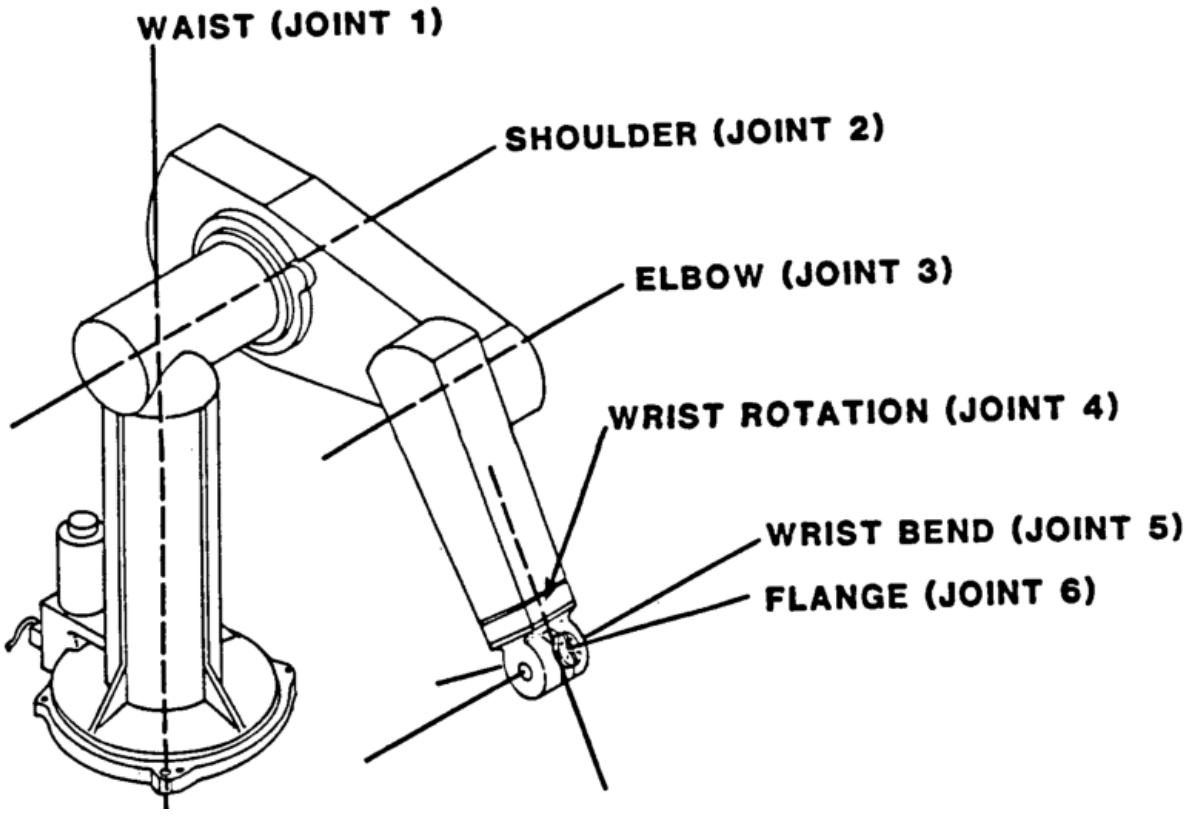


Figure 1.2: Joint Nomenclature of the PUMA 560 Robot Manipulator [2]

The control unit of the PUMA 560 is the most crucial part of the robotic system; it manages all operations of the manipulator and any connected devices [2]. The PUMA

560 robots are highly nonlinear, strongly coupled, and time-varying systems. Controlling such robotic manipulators to perform designated tasks is challenging due to the extreme nonlinearity of the system. In robotic systems, the dynamic equations' coefficients include joint and disturbances, which may be unknown or vary during the task. As the robot moves, the joint variables change, causing the dynamic equations to vary during the task.

To achieve higher repeatability and accuracy in robotic system performance, a control system of Model Reference Adaptive Control (MRAC) must account for changes in the dynamic characteristics of the system. Traditional control techniques treat robotic mechanisms as uncoupled linear subsystems, which may provide adequate performance at low speeds. However, they are ineffective for high-speed applications.

In MRAC, the modified methodology for second-order systems regulates the output of the plant through a feedback control loop, ensuring the model performs as expected. Compared to other control techniques, the adaptive approach performs better over a broad range of movements and loads. It achieves the desired system performance asymptotically, maintaining modeling accuracy regardless of the model's precision, and adapts in the presence of unexpected or adverse conditions. An input is provided to both the joint trajectory planning system and the actual joint variables, and the error between these is used to adjust the controller parameters; hence, minimizing the error dynamics.

1.2 Statement of the Problem

The PUMA 560 robot manipulator is widely used in various industrial applications due to its precision, flexibility, and versatility. However, controlling the PUMA 560 for high-speed operations presents several challenges. Traditional control methods, often struggle to maintain accurate tracking and stability when the manipulator operates at high speeds. This is primarily due to the presence of uncertainties, nonlinearities, and dynamic variations in the system that can affect performance. As the speed of operation increases, the robot's ability to track desired trajectories and maintain stability becomes more difficult, resulting in performance degradation.

Furthermore, the PUMA 560 is susceptible to system uncertainties, including model-

ing errors, external disturbances, and variations in parameters such as mass, friction, and geometry. These factors add complexity to the control problem and demand an adaptive approach that can adjust to the changing dynamics of the system. The need for a robust control strategy that can handle such uncertainties while ensuring precise control during high-speed operations is a critical issue.

This thesis addresses the problem of developing an adaptive control strategy for high-speed operations of the PUMA 560 robot manipulator. The objective is to design a MRAC that can overcome the limitations of traditional control methods by dynamically adjusting to the system's uncertainties. By implementing the MRAC approach, the goal is to improve trajectory tracking, enhance stability, and ensure robust performance in high speed operations, ultimately allowing the PUMA 560 robot manipulator to perform efficiently and reliably in high sped applications.

1.3 Objectives

1.3.1 General Objective

The general objective of the thesis is to design a MRAC system for the PUMA 560 robot manipulator that ensures stable and accurate high-speed operations, enabling precise trajectory tracking and robust performance for dynamic tasks and applications.

1.3.2 Specific Objectives

The specific objectives of the thesis are:

- To design a mathematical model of the PUMA 560 robot manipulator.
- To design a MRAC for high-speed operation of the PUMA 560 robot manipulator.
- To enhance stability and robustness of the MRAC.
- To simulate and validate MRAC performance in MATLAB.

1.4 Significance of the Thesis

The ability of a robot manipulator to perform high-speed operations with precision and adaptability is crucial for many industrial and research applications. For the PUMA 560 robot manipulator, improving the speed and accuracy of its movements is essential for tasks such as assembly, material handling, and automation in dynamic environments. The proposed Model Reference Adaptive Control (MRAC) system enhances the robot's capability to maintain stable flight and accurately position itself during high speed operations, even in the presence of disturbances and model uncertainties.

This thesis contributes to the advancement of robotics by developing a comprehensive MRAC framework tailored specifically for the PUMA 560 manipulator. It aims to improve trajectory tracking, robustness, and adaptability in high speed applications. Additionally, this work provides a mathematical model of the PUMA 560 robot manipulator and applies the MRAC controller to address the challenges of nonlinear dynamics and performance consistency at high speeds.

The findings and methodologies of this thesis will serve as a valuable reference for future research in the field of adaptive control, particularly for other robotic systems that require high-speed performance and stability. Researchers and practitioners interested in implementing MRAC in high-speed robotic applications can utilize the results and techniques outlined in this work.

1.5 Methodology of the Thesis

In order to evaluate the performance and effectiveness of the MRAC for high-speed operations of the PUMA 560 robot manipulator, the following phases are undertaken throughout this thesis.

1. Literature Review: A comprehensive review of existing literature on controller design and the structural aspects of PUMA robot manipulators is conducted. This includes studying various control strategies, with a focus on adaptive control techniques: MRAC, and their application to robotic systems.

2. System Modeling: The first step in designing the MRAC controller is to develop an accurate mathematical model of the PUMA 560 robot manipulator. A detailed description of the manipulator's structure and its physical parameters is provided. Based on this, the kinematic and inverse dynamics equations of the manipulator are derived to model its motion and dynamics.

3. Control Law Development: A control law is established using the MRAC approach to ensure the PUMA robot manipulator tracks the desired trajectory while maintaining stability and robustness during high-speed operations. The control law will adapt to changing conditions and model uncertainties during the execution of tasks.

4. Adaptive Algorithm Design: An adaptive algorithm is designed to adjust the controller parameters based on the error between the reference trajectory of joint angles and the actual joint angles. The algorithm is designed to improve the system's robustness and accuracy during high speed movements.

5. Reference Trajectory Planning: Reference trajectories for the PUMA 560 joint variables are planned to define the desired motion profiles and performance criteria. These trajectories serve as benchmarks for guiding the manipulator's joint movements to achieve the desired tasks effectively.

6. Simulink Model Design: A Simulink model of the PUMA 560 robot manipulator is developed in MATLAB. The dynamic model of the robot and the MRAC controller are integrated into this simulation environment to test the performance of the proposed controller.

7. Simulation: Extensive simulations are conducted to assess the performance of the MRAC system in various high-speed scenarios. The effectiveness of the MRAC in achieving high speed, accurate, and stable motion is analyzed.

8. Final paper writing and preparing for the final thesis presentation.

1.6 Thesis Outline

The thesis outline is organized into six chapters.

Chapter 1 presents a general introduction to the control of puma 560 robot manipulator.

Chapter 2 discusses a review of various research papers that are significant and provide background concepts for this thesis. This includes studies on model reference adaptive control, and Lyapunov method.

Chapter 3 presents the dynamic modeling of the PUMA 560 robot manipulator. The mathematical model, translational and rotational dynamics, is discussed thoroughly.

Chapter 4 presents the design of the model reference adaptive control. The control scheme design principles and the adaptive mechanism are explained in detail.

Chapter 5 discusses the simulation results and the analysis based on these simulations. The performance of the control system in various scenarios is evaluated.

Chapter 6 concludes the overall analysis performed, and recommendations for future work are presented. This includes potential improvements onto this thesis research.

Chapter 2

Literature Review

Adaptive control for robotic manipulators has undergone significant advancements, with Model Reference Adaptive Control (MRAC) proving essential in addressing the dynamic requirements of complex robotic systems. This literature review summarizes key contributions to MRAC in terms of its application in robotic manipulators, focusing on areas stability, disturbance rejection, transient response, and integration with hybrid systems.

2.1 MRAC-Based Control Strategies for the PUMA 560 Manipulator

Building on these foundational studies, researchers delved into the transient response capabilities of MRAC for high-speed robotic applications, aiming to optimize control performance during rapid changes. Piltan et al. (2015) addressed MRAC's limitations in handling transient response by developing a specialized MRAC controller tailored for a four-degree-of-freedom (DOF) robotic manipulator [1]. This innovative approach specifically targeted and successfully mitigated transient response delays and oscillations, enhancing the system's performance in high-speed, precision-driven tasks. By improving response speed and stability, their MRAC scheme allowed the robotic manipulator to adapt quickly to varying control demands, a vital requirement for applications necessitating both agility and precision.

Building on MRAC's inherent adaptability, Zhang et al. (2016) developed a hybrid MRAC-PID control system aimed at overcoming the limitations in accuracy and response time associated with standalone control methods. Their research demonstrated that combining MRAC with Proportional-Integral-Derivative (PID) control resulted in significantly improved convergence rates and response accuracy for multi-DOF robotic systems. This hybrid control strategy allowed the system to achieve both high precision and rapid adaptability, validating its effectiveness in multi-parameter robotic control tasks where both fast response and accuracy are critical [5].

To further enhance stability, Li et al. (2019) introduced a sliding mode control (SMC) component into the MRAC framework, creating a hybrid MRAC-SMC controller for robotic manipulators [7]. By incorporating SMC, the hybrid controller addressed chattering issues often encountered in traditional MRAC systems, leading to smoother and more stable control under uncertain conditions. This integration allowed the controller to maintain robust performance even in the presence of system perturbations, thereby improving adaptability and precision. Li et al.'s work demonstrated that combining sliding mode control with MRAC provides a powerful solution for applications requiring high resilience against dynamic changes and external disturbances, emphasizing the synergy between these two control methods in enhancing system robustness.

Incorporating fuzzy logic into MRAC, Saravanan et al. (2018) developed a fuzzy-MRAC control scheme for robotic arms, which significantly enhanced the system's adaptability to nonlinearities and uncertainties in the control environment [9]. By integrating fuzzy logic, the MRAC controller became better equipped to handle complex, unpredictable scenarios where traditional control methods might struggle. This hybrid approach enabled the system to adapt more seamlessly to varying conditions, improving performance in environments with inherent nonlinear dynamics and uncertainty. The research demonstrated that the fuzzy-MRAC scheme expanded the applicability of MRAC, making it suitable for a broader range of robotic applications that require fine-tuned, flexible control in challenging operational settings.

Kumar et al. (2019) integrated neural networks with MRAC to improve its disturbance rejection and adaptability, especially under changing load conditions [10]. The

neural network-enhanced MRAC demonstrated superior performance by refining control actions based on real-time data, enabling more accurate tracking and improved control in dynamic environments. By utilizing neural networks, the system was able to dynamically adjust control parameters, ensuring better performance even when load and system conditions fluctuated. This innovation made MRAC more suitable for applications with frequently varying conditions, where traditional controllers might struggle to maintain optimal performance, thus enhancing its effectiveness in complex, real-world scenarios.

Scaling MRAC to multi-degree-of-freedom (DOF) systems, Yamada and Ota (2022) demonstrated that MRAC can effectively control complex robotic manipulators with multiple DOFs [12]. Their research focused on addressing the computational demands associated with multi-DOF systems, showing that MRAC could be optimized to meet the complex control requirements of these systems without introducing excessive computational overhead. By refining the MRAC structure, they achieved an efficient balance between control accuracy and computational efficiency, making MRAC a viable solution for controlling advanced, multi-DOF robotic manipulators in practical applications. This work expanded MRAC's applicability to more complex robotic systems, where high performance is required across multiple axes of motion.

In cases of significant parameter variations, Han et al. (2020) developed an MRAC approach designed to adapt to changing system parameters in robotic manipulators, ensuring consistent performance even when parameters deviate from expected values [13]. Their approach demonstrated robustness to parameter changes, allowing the system to maintain stability and control accuracy in unpredictable environments. This adaptability is particularly crucial for applications where operating conditions are uncertain or prone to fluctuations, such as in autonomous or industrial robotic systems. The research highlighted that MRAC can effectively handle dynamic parameter shifts, making it suitable for applications requiring flexible and reliable control under varying conditions.

Finally, Yin and Ma (2022) proposed a model-free MRAC approach, enabling the control of robotic manipulators in environments where detailed system modeling is not feasible [15]. Their method demonstrated how MRAC could be effectively implemented in scenarios with limited prior knowledge about the system dynamics. This model-free

approach extended MRAC's applicability to more dynamic and less predictable environments, where traditional model-based methods might struggle. By eliminating the need for an explicit system model, their work expanded the potential use cases for MRAC in complex, real-time control tasks where system characteristics are uncertain or unknown.

In a similar vein, Liu et al. (2023) presented advancements in MRAC convergence, developing a model that accelerates convergence rates for robotic manipulators [16]. Their approach reduced settling times and improved tracking precision, making it particularly effective for systems that require high-frequency adjustments. This advancement allowed MRAC to perform more efficiently in dynamic environments, where quick adaptation is crucial, further enhancing its suitability for precision-driven robotic applications that demand fast and accurate control responses.

MRAC has been one of the well-known adaptive control methods for handling such problems. Direct MRAC allows for a desired system behavior to be specified as a reference model. This is particularly useful for MIMO systems such as robotic manipulators. Direct MRAC attempts to drive the plant response to reference model response by parameterizing the controller and estimating the "controller" parameters online. Indirect MRAC has as its objective obtaining the best estimates of "plant" parameters first and then using them to produce a suitable control input. This is why the adaptation process occurs indirectly. Common parameter estimation methods are based around Recursive Least Square (RLS) [18], which can also be interpreted as Kalman filtering and Lyapunov based estimation.

2.2 Review of Implementation Strategies and Hybrid Control Approaches for the PUMA 560

Early research into MRAC for robotics laid the foundation for its application in dynamic control. Neuman et al. (1986) pioneered the use of MRAC to stabilize and control robotic mechanisms in environments where dynamics could change rapidly [3]. Their approach set the groundwork for subsequent MRAC adaptations by providing a framework for

model reference control in time-varying environments, crucial for tasks demanding high adaptability.

In a similar vein, Köker et al. (2004) explored the integration of MRAC with neural networks to further improve control accuracy and adaptability in dynamic robotic environments [2]. This hybrid MRAC-neural network approach provided enhanced control by leveraging neural networks' ability to model complex, nonlinear dynamics in real time. The result was improved response times and control precision, particularly beneficial in situations demanding rapid adaptability to unpredictable disturbances. By combining MRAC with neural networks, Köker et al. demonstrated that MRAC could achieve superior control performance in dynamic systems, paving the way for more responsive and accurate robotic applications.

In parallel to these efforts, other researchers explored the effectiveness of MRAC in parallel manipulators to address control challenges posed by complex kinematic structures. Alinia et al. (2015) applied MRAC to a three-degree-of-freedom (DOF) parallel manipulator and conducted a comparative analysis with traditional feedback linearization techniques [4]. Their study revealed that MRAC provided superior performance in noisy environments, where feedback linearization often struggled to maintain control accuracy. Specifically, MRAC demonstrated enhanced robustness under uncertain conditions, maintaining stability and precision despite external disturbances. This robustness highlighted the suitability of adaptive control for systems with complex kinematic chains, confirming MRAC's advantages for applications requiring resilient control under variable conditions.

Further supporting the hybrid approach, Rahimi et al. (2017) introduced disturbance compensation within MRAC specifically for robotic manipulators, enhancing the system's robustness against external disturbances [6]. By incorporating a disturbance compensation mechanism, their MRAC scheme maintained stability and control accuracy even in dynamically changing environments, a crucial requirement for robotic tasks involving unpredictable external forces. This research highlighted the importance of disturbance compensation as a key element in maintaining stability for MRAC-based controllers during complex and dynamic tasks.

MRAC has also been adapted to flexible robotic manipulators, where maintaining control accuracy and stability is especially challenging due to the system's variable stiffness. Chen et al. (2020) applied MRAC to robotic systems with variable stiffness, demonstrating that MRAC can effectively manage flexibility-related challenges by dynamically adjusting to stiffness variations [8]. This adaptive capability allowed the system to maintain precise tracking performance without sacrificing accuracy, even as stiffness changed. Their research highlighted MRAC's versatility and reliability in precision-driven applications where material flexibility and variable compliance are key factors, underscoring its effectiveness for complex, adaptive robotic tasks.

Exploring advanced machine learning techniques, Xie et al. (2021) integrated reinforcement learning into MRAC to enhance online adaptability in dynamic environments [11]. This innovative approach significantly improved the convergence speed of MRAC, particularly in high-speed applications where rapid response to changing conditions is critical. The incorporation of reinforcement learning allowed the system to learn and adjust based on real-time performance, reducing the need for constant human intervention. This capability was especially beneficial for autonomous robotic systems, which require rapid adaptation to environmental changes. The reinforcement learning component empowered MRAC to autonomously optimize its control actions, leading to more efficient and effective operation in complex, variable settings.

MRAC has also proven valuable in autonomous robotic applications. Wang and Chen (2021) applied MRAC to autonomous mobile robots, demonstrating its ability to adapt to a variety of environmental challenges, such as variable terrain and fluctuating payloads [14]. Their study showed that MRAC could effectively maintain control stability and accuracy across a wide range of operational scenarios, including situations with unpredictable environmental conditions. This adaptability made MRAC particularly well-suited for mobile robotic platforms, where the robot must continuously adjust its control strategy to ensure consistent performance in dynamic and changing environments, highlighting MRAC's versatility and reliability in autonomous robotics.

MRAC was proposed for manipulator control in 1979 by Dubowsky et al. (1979) [17]. The adaptation mechanism in MRAC, in principle allows the control system to maintain

a desired response robust to model mismatch and changes to the plant. One way to specify the desired behavior is through defining a reference model. This is the basis for so-called direct MRAC.

The common approach in applying direct MRAC to manipulators is to combine feedforward and feedback explained by Maliotis (1991) [19], who shows, in this approach the feedforward component makes use of the known portion parameters of the plant such as inertia to calculate the likely input. The adaptive feedback aims to compensate for the varying nonlinear effects by using n linear decoupled second order systems as the reference model. The feedforward component could also be interpreted as the result of feedback linearization of the plant.

These studies collectively highlight the versatility and adaptability of MRAC across a wide range of robotic control tasks. Whether through the integration of hybrid methods, fuzzy logic, neural networks, or reinforcement learning, MRAC has consistently demonstrated its ability to enhance robotic system performance under varying and challenging operational conditions. By continuously evolving to address complex control requirements, MRAC has proven to be an effective solution in environments where uncertainty, nonlinearity, and dynamic changes are prevalent. This adaptability makes MRAC a valuable tool in autonomous systems, high-precision tasks, and multi-DOF manipulators, underscoring its ongoing relevance and potential for advancing robotic control technologies in both research and practical applications.

Chapter 3

PUMA 560 Robot Manipulator Modeling and Design

3.1 Working Principle

This chapter presents a detailed analysis of the kinematic and dynamic equations governing the six DOF PUMA 560 robot manipulator. These equations are fundamental for understanding the robot's motion and are used to evaluate the position, orientation, and velocity of the end effector in space. They are also crucial for determining the joint variables required to achieve a desired task. Specifically, the kinematic equations describe the spatial relationships between the robot's links and joints, while the dynamic equations account for the forces and torques involved in moving the manipulator.

A widely used approach for modeling these relationships is the homogeneous transformation matrix method. This method allows for the systematic representation of the robot's kinematics by relating the position and orientation of one coordinate frame to another. The transformation matrices capture the effects of translation and rotation between successive links, making it possible to compute the end-effector's pose of position and orientation based on the joint angles and link lengths. This method provides a unified framework for deriving the forward and inverse kinematics of the manipulator.

The dynamic equations describe the relationship between the forces or torques applied at the joints and the resulting motion of the manipulator, considering factors such as

inertia, friction, and external forces. These equations are essential for designing control strategies, as they provide insights into the required joint torques for performing desired movements with accuracy and efficiency. These kinematic and dynamic equations form the foundation for the controller design process, enabling the development of algorithms that control the PUMA 560 robot manipulator's motion, stability, and responsiveness in for control applications.

3.2 PUMA 560 Robot Kinematic Modeling

Kinematic modeling of robotic manipulators is crucial for understanding the relationship between the joint configurations, and the position and orientation of the end-effector. For a six DOF manipulator PUMA 560, the modeling involves determining how the manipulator's joints move and interact to place the end-effector in a desired location in space. The PUMA 560 robot manipulator consists of six revolute joints, each contributing to the robot's overall kinematic structure. To model the kinematics of such a robot, we use the Denavit-Hartenberg (DH) convention, a standardized method for representing the transformation between adjacent links of the manipulator. The DH parameters are key in describing the relative motion between the links and joints, and these parameters include the joint angle (θ), link length (a), link offset (d), and twist angle (α).

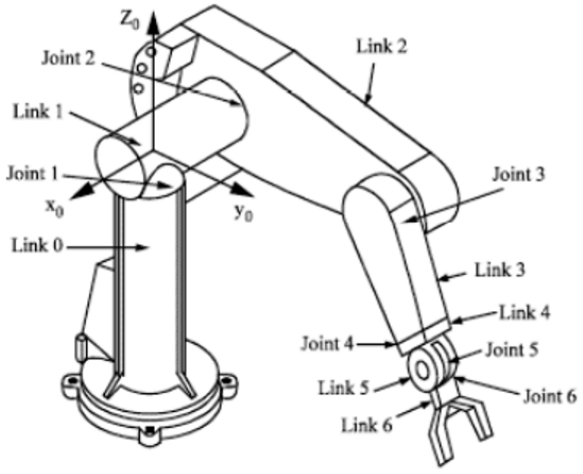


Figure 3.1: Kinematic Description of PUMA 560 Robot Manipulator

3.2.1 Forward Kinematics

Forward kinematics involves calculating the position and orientation of the end effector based on known values of the joint variables i.e. the angles for each revolute joint in the PUMA 560 robot manipulator. The goal is to compute the transformation matrix that describes the position and orientation of the end effector in the task space called a Cartesian space.

Using the Denavit-Hartenberg convention, the transformation from one link to the next is expressed with a homogeneous transformation matrix. Here for the PUMA 560, six transformation matrices are used 3.1, each corresponding to one of the joints.

$$ {}^{i-1}_{i}\mathbf{T} = \text{Rot}_x(\alpha_{i-1}) \cdot \text{Trans}_x(a_{i-1}) \cdot \text{Rot}_z(\theta_i) \cdot \text{Trans}_z(d_i) \tag{3.1} $$

where ${}^{i-1}_{i}T$ represents the transformation matrix between the i-th and $(i+1)$-th links.

The overall transformation matrix from the base of the robot to the end-effector is the product of these six individual transformation matrices and given by (3.2).

$$ {}^{0}_{6}\mathbf{T} = {}^{0}_{1}\mathbf{T} \cdot {}^{1}_{2}\mathbf{T} \cdot {}^{2}_{3}\mathbf{T} \cdot {}^{3}_{4}\mathbf{T} \cdot {}^{4}_{5}\mathbf{T} \cdot {}^{5}_{6}\mathbf{T} \tag{3.2} $$

Each transformation matrix is computed using the DH parameters for the corresponding link, and the final transformation matrix provides the position and orientation of the end effector in 3D space. This transformation matrix typically consists of a 3x3 rotation matrix and a 3x1 translation vector.

3.2.1.1 Denavit-Hartenberg (DH) Parameters

To obtain the forward kinematics for the PUMA 560, we use the DH convention, a standardized approach for representing the geometry of robotic mechanisms [20]. The DH method reduces the number of mathematical operations required for kinematic modeling by defining each joint with a set of four parameters: link length a_i, link offset d_i, joint angle θ_i, and link twist α_i. These parameters allow for the systematic description of each joint's position and orientation in a way that can be applied to complex multi-link

robotic manipulators.

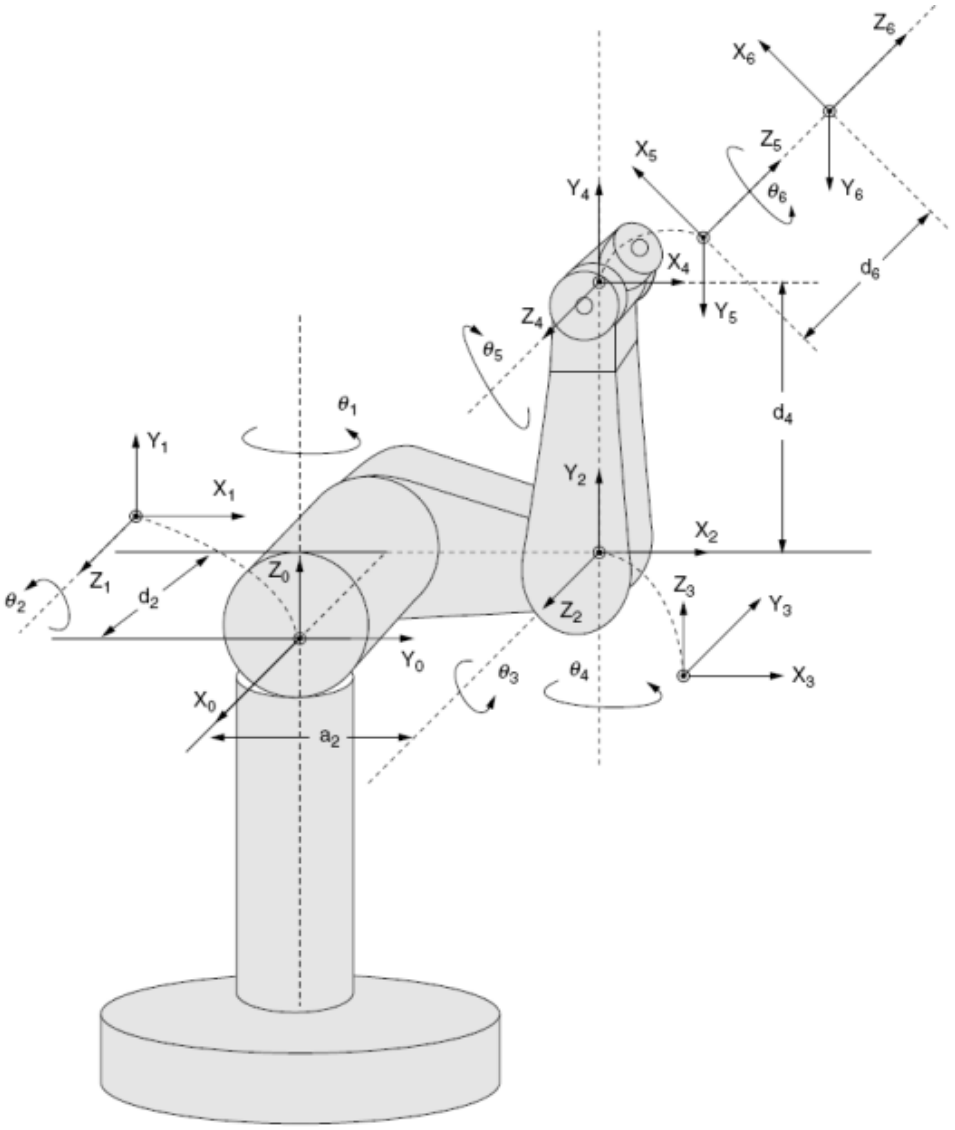


Figure 3.2: Kinematic Pair and DH Parameters [21]

where the link length, denoted as a_i, represents the distance between two consecutive joint axes along the x-axis. The link offset, d_i, is the displacement along the z-axis. The joint angle, θ_i, describes the rotational displacement about the z-axis. Lastly, the link twist, α_i, is the angle between the z-axes of two consecutive links.

These parameters provide a structured framework for developing the transformation matrices that describe each link's pose in relation to the previous link. The DH parameters for each joint of the PUMA 560 robot arranged in a Table 3.1.

Table 3.1: Denavit-Hartenberg Parameters for the PUMA 560 Robot Manipulator

Joint i	a_i	d_i	α_i	θ_i
1	a_1	d_1	α_1	θ_1
2	a_2	d_2	α_2	θ_2
3	a_3	d_3	α_3	θ_3
4	a_4	d_4	α_4	θ_4
5	a_5	d_5	α_5	θ_5
6	a_6	d_6	α_6	θ_6

The PUMA 560 is a revolute robot, hence; the only parameters that change during motion are the θ_i joint angles. Table 3.1 gives one possible set of DH parameters for the zero configuration given in Table 3.2.

Table 3.2: Denavit-Hartenberg Parameters for the PUMA 560 Robot Manipulator

Joint i	$a_i(m)$	$d_i(m)$	$\alpha_i(degree)$	$\theta_i(degree)$
1	0	0	0	θ_1
2	0	0.2435	-90	θ_2
3	0.4318	-0.0924	0	θ_3
4	-0.0203	0.4331	90	θ_4
5	0	0	-90	θ_5
6	0	0	90	θ_6

These DH parameters are then used to derive the forward kinematics equations for the manipulator, generating a homogeneous transformation matrix that specifies the position

and orientation of each joint relative to the base frame (3.3) [22].

$$
\begin{aligned}
{}^{i-1}_{i}\mathbf{T} &= \mathrm{Rot}_x(\alpha_{i-1}) \cdot \mathrm{Trans}_x(a_{i-1}) \cdot \mathrm{Rot}_z(\theta_i) \cdot \mathrm{Trans}_z(d_i) \\
&= \begin{bmatrix}
c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\
s_{\theta_i} c_{\alpha_{i-1}} & c_{\theta_i} c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -d_i s_{\alpha_{i-1}} \\
s_{\theta_i} s_{\alpha_{i-1}} & c_{\theta_i} s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & d_i c_{\alpha_{i-1}} \\
0 & 0 & 0 & 1
\end{bmatrix}
\end{aligned}
\tag{3.3}
$$

where Rot and Trans denote rotation and translation, respectively, and $s_{\theta_i}, c_{\theta_i}, s_{\alpha_i}$, and c_{α_i} are shorthand notations for $\sin\theta_i, \cos\theta_i, \sin\alpha_i$, and $\cos\alpha_i$, respectively.

The homogeneous rotation and translation matrices are multiplied to form the composite homogeneous transformation matrix ${}^{i-1}_{i}\mathbf{T}$, which describes the transformation from coordinate frame $i-1$ to frame i. This transformation encapsulates both rotational and translational components between consecutive frames in kinematic chains.

3.2.1.2 Link Transformation via Concatenation

The matrix ${}^{i-1}_{i}\mathbf{T}$ in (3.3) can be used for any kinematic chain containing revolute or prismatic joints for position and orientation analysis. Starting from the base of the robot, transformations can be computed sequentially: from the first joint to the second, then to the third, and so on, up to the last joint of the robot. If each transformation is defined as ${}^{i-1}_{i}\mathbf{T}$, then a series of transformations can be obtained as follows.

The individual link transformation matrices computed using the values of the link parameters. Then, the link transformations multiplied together to find a single transformation that relates the manipulator's end-effector to the base frame [22, 23]. This is expressed by specifically with (3.2) and more general form of (3.4).

$$
{}^{0}_{n}\mathbf{T} = {}^{0}_{1}\mathbf{T} \cdot {}^{1}_{2}\mathbf{T} \cdot {}^{2}_{3}\mathbf{T} \cdot {}^{3}_{4}\mathbf{T} \cdots {}^{n-1}_{n}\mathbf{T} \tag{3.4}
$$

where $\{0\}$ is the base frame, and $\{n\}$ is the end-effector frame, and $n = 6$.

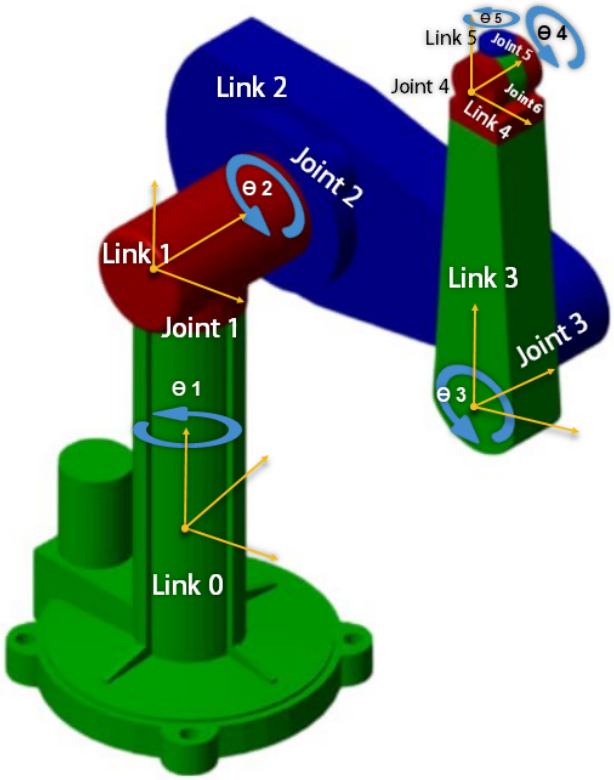


Figure 3.3: Frame Assignment and Kinematic Parameters for PUMA 560 Robot

The forward kinematics of the end-effector relative to the base frame is determined by multiplying the ${}^{i-1}_{i}\mathbf{T}$ matrices, with the DH parameter values from Table 3.2 substituted into 3.3. By applying the values from this table and (3.3), the kinematic transformation matrices for the PUMA 560 are derived. Hence, the transformation from joint one coordinates to the zero frame is calculated as shown in (3.5).

$$ {}^{0}_{1}\mathbf{T} = \begin{pmatrix} c_{\theta_1} & -s_{\theta_1} & 0 & 0 \\ s_{\theta_1} & c_{\theta_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.5} $$

The remaining joint to joint transformations are given as follows.

$$ {}^{1}_{2}\mathbf{T} = \begin{pmatrix} c_{\theta_2} & -s_{\theta_2} & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ -s_{\theta_2} & -c_{\theta_2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^{2}_{3}\mathbf{T} = \begin{pmatrix} c_{\theta_3} & -s_{\theta_3} & 0 & a_2 \\ s_{\theta_3} & c_{\theta_3} & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.6} $$

$$ {}^{3}_{4}\mathbf{T} = \begin{pmatrix} c_{\theta_4} & -s_{\theta_4} & 0 & a_3 \\ 0 & 0 & -1 & -d_4 \\ s_{\theta_4} & c_{\theta_4} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad {}^{4}_{5}\mathbf{T} = \begin{pmatrix} c_{\theta_5} & -s_{\theta_5} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_{\theta_5} & -c_{\theta_5} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.7} $$

$$ {}^{5}_{6}\mathbf{T} = \begin{pmatrix} c_{\theta_6} & -s_{\theta_6} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_{\theta_6} & c_{\theta_6} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8} $$

Multiplying these individual link transformation matrices, we obtain the transformation matrix of the robot's end coordinate frame relative to the base frame, ${}^{0}_{6}\mathbf{T}$, as in (3.2). Thus, the position and orientation of a rigid object at the end-effector described by the transformation matrix ${}^{0}_{6}\mathbf{T}$ as in (3.9).

$$ {}^{0}_{6}\mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9} $$

where coefficients in equation 3.9 are given by 3.10, and c_i and s_i are cosine and sine of angle θ_i, whereas the c_{ij} and s_{ij} are also denote the cosine and sine value of the joint

angles combined from addition of angles θ_i and angle θ_j.

$$
\begin{aligned}
r_{11} &= c_1\big(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6\big) - s_1\big(s_4c_5c_6 + c_4s_6\big)\\
r_{12} &= c_1\big(c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6\big) - s_1\big(-s_4c_5s_6 + c_4c_6\big)\\
r_{13} &= c_1\big(c_{23}c_4s_5 + s_{23}c_5\big) - s_1s_4s_5\\
p_x &= c_1\big(a_3c_{23} + d_4s_{23} + a_2c_2\big) - s_1(d_2 + d_3)\\
r_{21} &= s_1\big(c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6\big) + c_1\big(s_4c_5c_6 + c_4s_6\big)\\
r_{22} &= s_1\big(c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6\big) + c_1\big(-s_4c_5s_6 + c_4c_6\big)\\
r_{23} &= s_1\big(c_{23}c_4s_5 + s_{23}c_5\big) + c_1s_4s_5\\
p_y &= s_1\big(a_3c_{23} + d_4s_{23} + a_2c_2\big) + c_1(d_2 + d_3)\\
r_{31} &= -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6\\
r_{32} &= -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6\\
r_{33} &= -s_{23}c_4s_5 + c_{23}c_5\\
p_z &= -a_3s_{23} + d_4c_{23} - a_2s_2
\end{aligned}
\tag{3.10}
$$

3.2.2 Inverse Kinematics

Inverse kinematics (IK) is the process of determining the joint angles needed to position the end effector at a desired position and orientation in space. Unlike forward kinematics, which is a straightforward calculation, inverse kinematics involves solving a set of nonlinear equations that relate the end-effector's desired position and orientation to the joint variables. The inverse kinematics problem is typically solved by applying analytical or numerical methods [23, 24, 25, 26, 27]. Analytical solutions are possible for simpler robot configurations, but for more complex configurations of PUMA 560, numerical methods such as Newton-Raphson or gradient descent are often used. These methods iteratively adjust the joint angles to minimize the error between the actual and desired end-effector position.

The inverse kinematics solution provides a set of joint angles $\theta_1, \theta_2, ..., \theta_6$ that achieve the desired position and orientation. It's important to note that for certain configurations, multiple or no solutions may exist due to the redundancy or limitations of the robot's

workspace. Hence, to find the inverse kinematics solution for the first joint (θ_1), we recall (3.9). By multiplying each side of the direct kinematics equation by an inverse transformation matrix, the variables can be separated into solvable equations. Placing the dependence on θ_1 on the left-hand side of the equation, we multiply the direct kinematics equation by $[{}^0_1\mathbf{T}_{\theta_1}]^{-1}$ as follows in (3.11).

$$[{}^0_1\mathbf{T}_{\theta_1}]^{-1} \cdot {}^0_6\mathbf{T} = [{}^0_1\mathbf{T}_{\theta_1}]^{-1} \cdot {}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2} \cdot {}^2_3\mathbf{T}_{\theta_3} \cdot {}^3_4\mathbf{T}_{\theta_4} \cdot {}^4_5\mathbf{T}_{\theta_5} \cdot {}^5_6\mathbf{T}_{\theta_6} \tag{3.11}$$

where the term $[{}^0_1\mathbf{T}_{\theta_1}]^{-1} \cdot {}^0_1\mathbf{T}_{\theta_1}$ simplifies to the identity matrix. This isolates the remaining transformations, making it possible to solve for θ_1.

Hence, to solve for the remaining variables $(\theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$, we proceed in a similar manner. For θ_2, we isolate it by multiplying by $[{}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2}]^{-1}$ as in (3.12).

$$[{}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2}]^{-1} \cdot {}^0_6\mathbf{T} = {}^2_3\mathbf{T}_{\theta_3} \cdot {}^3_4\mathbf{T}_{\theta_4} \cdot {}^4_5\mathbf{T}_{\theta_5} \cdot {}^5_6\mathbf{T}_{\theta_6} \tag{3.12}$$

Continuing this approach, the remaining transformations can be written as (3.13), (3.14), and (3.15).

$$[{}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2} \cdot {}^2_3\mathbf{T}_{\theta_3}]^{-1} \cdot {}^0_6\mathbf{T} = {}^3_4\mathbf{T}_{\theta_4} \cdot {}^4_5\mathbf{T}_{\theta_5} \cdot {}^5_6\mathbf{T}_{\theta_6} \tag{3.13}$$

$$[{}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2} \cdot {}^2_3\mathbf{T}_{\theta_3} \cdot {}^3_4\mathbf{T}_{\theta_4}]^{-1} \cdot {}^0_6\mathbf{T} = {}^4_5\mathbf{T}_{\theta_5} \cdot {}^5_6\mathbf{T}_{\theta_6} \tag{3.14}$$

$$[{}^0_1\mathbf{T}_{\theta_1} \cdot {}^1_2\mathbf{T}_{\theta_2} \cdot {}^2_3\mathbf{T}_{\theta_3} \cdot {}^3_4\mathbf{T}_{\theta_4} \cdot {}^4_5\mathbf{T}_{\theta_5}]^{-1} \cdot {}^0_6\mathbf{T} = {}^5_6\mathbf{T}_{\theta_6} \tag{3.15}$$

The resulting transformation matrix ${}^1_6\mathbf{T}$ is expressed as in (3.16).

$${}^1_6\mathbf{T} = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} & p'_x \\ r'_{21} & r'_{22} & r'_{23} & p'_y \\ r'_{31} & r'_{32} & r'_{33} & p'_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

where r'_{ij} represents the rotational components of the transformation matrix, and p'_x, p'_y, and p'_z are the translational components. This matrix describes the pose i.e. position and orientation of the end effector in the base frame. Substituting the corresponding DH

parameters yields the link transformation matrix.

There are totally 12 simultaneous sets of nonlinear equations to be solved. The only unknown on the left-hand side of the equation is θ_1. The 12 nonlinear matrix elements on the right-hand side are either zero, constant, or functions of θ_2 through θ_6.

Equating the elements on the left-hand side, which are functions of θ_1, with the elements on the right-hand side, allows us to solve for the joint variable θ_1 as a function of $r_{11}, r_{21}, r_{31}, \ldots, r_{13}, r_{23}, r_{33}, p_x, p_y, p_z$, and the fixed link parameters. Once θ_1 is found, the other joint variables are solved in a similar way as described in the previous steps.

$$[{}^0_1\mathbf{T}_{\theta_1}]^{-1} \cdot {}^0_6\mathbf{T} = \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & 0 & r_{33} & p_z \\ 0 & 0 & 1 & 1 \end{bmatrix} = {}^1_6\mathbf{T} \tag{3.17}$$

To derive the equation for θ_1, lets equate the (2,4) element on both sides of equation (3.17), which gives the equation (3.18).

$$-s_1 p_x + c_1 p_y = d_2 + d_3 \tag{3.18}$$

To solve the equation (3.18) in this form, we make the trigonometric substitution in the from (3.19), because equation (3.18) is a single equation with a single unknown, the solution is found by using the following polar coordinates.

$$\begin{aligned} p_x &= \rho \cos\phi \\ p_y &= \rho \sin\phi \\ \rho &= \sqrt{p_x^2 + p_y^2} \\ \phi &= \arctan(\frac{p_y}{p_x}) \end{aligned} \tag{3.19}$$

where ρ and ϕ are the polar coordinates corresponding to p_x and p_y, having the angle ϕ.

$$\begin{aligned} \sin(\phi - \theta_1) &= \frac{d_2 + d_3}{\rho} \\ \cos(\phi - \theta_1) &= \pm\sqrt{\left(1 - \frac{(d_2 + d_3)^2}{\rho^2}\right)} \end{aligned} \tag{3.20}$$

Combining the polar representations with equation (3.18) and trigonometric angle identities given by (3.20); thus, the equation for θ_1 becomes as (3.21).

$$\theta_1 = \arctan(\frac{p_y}{p_x}) - \arctan\left(\frac{d_2 + d_3}{\pm k}\right) \tag{3.21}$$

where;

$$k = \sqrt{p_x^2 + p_y^2 - (d_2 + d_3)^2} \tag{3.22}$$

Thus equation (3.21) provides the solution for θ_1 in terms of the given parameters p_x, p_y, p_z, and the fixed link parameters d_2 and d_3. However, there are two possible values for θ_1 in equation (3.21), depending on the sign selected in the argument of the second arctangent function. The chosen sign will influence all subsequent inverse kinematic solutions and leads to the multiple solutions. Hence, the arctangent function refers to the atan2 function, which is widely used in robotics, and a specialized form of the arctangent function that ensures the quadrant of the computed angle is retained. This method makes the quadrant tobe preserved, avoiding possible angle deviation by 180°.

To derive the equation for θ_2, we begin by equating the (1,4) and (2,4) elements of the matrix transformation from (3.17), leading the two equations (3.23), and (3.24).

$$c_1 c_{23} p_x + s_1 c_{23} p_y - s_{23} p_z - a_2 c_3 = a_3 \tag{3.23}$$

$$-c_1 s_{23} p_x - s_1 s_{23} p_y - c_{23} p_z + a_2 s_3 = d_4 \tag{3.24}$$

These equations can be solved simultaneously for c_{23} and s_{23}, resulting expressions (3.25).

$$\begin{aligned} c_{23} &= \frac{(a_2 s_3 - d_4)p_z - (-a_3 - a_2 c_3)(c_1 p_x + s_1 p_y)}{p_z^2 (c_1 p_x + s_1 p_y)^2} \\ s_{23} &= \frac{(-a_3 - a_2 c_2)p_z + (c_1 p_x + s_1 p_y)(a_2 s_3 - d_4)p_z}{p_z^2 (c_1 p_x + s_1 p_y)^2} \end{aligned} \tag{3.25}$$

Next, we can derive the equation for θ_{23} by applying the inverse tangent function, using the following expression in (3.26) for θ_{23}.

$$\theta_{23} = \text{atan2}\left(\frac{(-a_3 - a_2 c_2)p_z + (c_1 p_x + s_1 p_y)(a_2 s_3 - d_4)}{(a_2 s_3 - d_4)p_z - (-a_3 - a_2 c_2)(c_1 p_x + s_1 p_y)} \right) \tag{3.26}$$

The joint variable θ_2 is then found by the relationship (3.27).

$$\theta_2 = \theta_{23} - \theta_3 \tag{3.27}$$

Finally, the transformation matrix for the joint angles θ_1, θ_2, and θ_3 is given by (3.28).

$$\begin{aligned} {}^0_3\mathbf{T}]^{-1} \cdot {}^0_6\mathbf{T} &= {}^3_6\mathbf{T} \\ &= \begin{bmatrix} c_1 c_{23} & s_1 c_{23} & -s_{23} & -a_2 c_3 \\ -c_1 s_{23} & -s_1 s_{23} & -c_{23} & a_2 s_3 \\ -s_1 & c_1 & 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & 0 & r_{33} & p_z \\ 0 & 1 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_4 c_5 s_6 - s_4 c_6 & -c_4 s_5 & a_3 \\ s_5 c_6 & -s_5 s_6 & c_5 & d_4 \\ -s_4 c_5 c_6 - c_4 s_6 & s_4 c_5 s_6 - c_4 c_6 & s_4 s_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.28}$$

To deriving the equation for θ_3, We begin by equating the $(1,4)$ element and the $(3,4)$ element of the transformation matrix as in (3.28) resulting the equations (3.29), (3.30), and (3.31).

$$-s_1 p_x + c_1 p_y = d_3 \tag{3.29}$$

$$c_1 p_x + s_1 p_y = a_3 c_{23} - d_4 s_{23} + a_2 c_2 \tag{3.30}$$

$$-p_z = a_3 c_{23} + d_4 s_{23} + a_2 c_2 \tag{3.31}$$

Next, we square these equations and add them to eliminate the terms involving the trigonometric functions and variables. This gives us the expression (3.32).

$$(a_3 c_3 - d_4 s_3) = k \tag{3.32}$$

where k is defined by (3.33).

$$k = \frac{p_x^2 + p_y^2 + p_z^2 - a_2^2 - a_3^2 - d_3^2 - a_4^2}{2a_2} \tag{3.33}$$

Now, to solve for θ_3, we use the inverse tangent function, resulting (3.34).

$$\theta_3 = -\text{atan2}\left(\frac{a_3}{d_4}\right) + \text{atan2}\left(\frac{k}{\rho^2 - k^2}\right) \tag{3.34}$$

Next, the joint transformation matrix equation is given by (3.35).

$$\begin{aligned}
\left[{}^0_4\mathbf{T}\right]^{-1} \cdot {}^0_6\mathbf{T} &= {}^4_6\mathbf{T} \\
&= \left[\begin{array}{cccc|cccc}
c_1c_{23} & s_1c_{23} & -s_{23} & -a_2c_3 & r_{11} & r_{12} & r_{13} & p_x \\
-c_1s_{23} & -s_1s_{23} & -c_{23} & a_2s_3 & r_{21} & r_{22} & r_{23} & p_y \\
-s_1 & 0 & c_1 & -d_3 & r_{31} & r_{32} & r_{33} & p_z \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1
\end{array}\right] \\
&= \left[\begin{array}{ccc|c}
c_4c_5c_6 - s_4s_6 & -c_4c_5s_6 - s_4c_6 & -c_4s_5 & a_3 \\
s_5c_6 & -s_5s_6 & c_5 & d_4 \\
-s_4c_5c_6 - c_4s_6 & s_4c_5s_6 - c_4c_6 & s_4s_5 & 0 \\
0 & 0 & 0 & 1
\end{array}\right]
\end{aligned} \tag{3.35}$$

To derive the equation for θ_4, lets equate the (1,3) and (3,3) elements of equation (3.35)

resulting the expressions given by (3.36) and (3.37).

$$-r_{13}c_1c_{23} + r_{23}s_1c_{23} - r_{33}s_{23} = -c_4s_5 \tag{3.36}$$

$$-r_{13}s_1 + r_{23}c_1 = s_4s_5 \tag{3.37}$$

As long as we can solve for $s_5 \neq 0$, we can solve for θ_4 as (3.38).

$$\theta_4 = \text{atan2}\left(\frac{-r_{13}s_1 + r_{23}c_1}{-r_{13}c_1c_{23} - r_{23}s_1c_{23} + r_{33}s_{23}}\right) \tag{3.38}$$

To derive the equation for θ_5, let us equate the (1,3) and (3,3) elements of equation (3.35).

$$r_{13}\left(c_1c_{23}c_{4+}s_1s_4\right) + r_{23}\left(s_1c_{23}c_{4-}c_1s_4\right) - r_{33}s_{23}c_4 = -s_5 \tag{3.39}$$

$$r_{13}\left(c_1c_{23}\right) + r_{23}\left(-s_1s_{23}\right) + r_{33}\left(-c_{23}\right) = c_5 \tag{3.40}$$

Thus, θ_5 is given by (3.41).

$$\theta_5 = \text{atan2}\left(\frac{s_5}{c_5}\right) \tag{3.41}$$

Th transformation matrix for θ_6 is given by then in 3.42.

$$\left[{}^0_5\mathbf{T}\right]^{-1} \cdot {}^0_6\mathbf{T} = {}^5_6\mathbf{T} \tag{3.42}$$

The last joint position to be derived is the θ_6, so lets equate the (3,1) and (1,1) elements of (3.42) as in (3.43).

$$\begin{aligned} s_6 &= r_{11}\left(c_1s_{23}c_5 - s_1s_{23}s_5\right) + r_{21}\left(c_{23}\right) + r_{31}\left(c_1s_{23}c_5 + s_1s_{23}c_5\right) \\ &= r_{11}\left(c_1c_{23}c_4c_5 + s_1s_4c_5 - s_1c_{23}c_4s_5 + c_1s_4s_5\right) + r_{21}\left(s_{23}c_4\right) \\ &+ r_{31}\left(c_1c_{23}c_4c_5 + s_1s_4s_5 + s_1c_{23}c_4c_5 - c_1s_4c_5\right) \end{aligned} \tag{3.43}$$

Finally, solving for θ_6 is given by (3.44).

$$\theta_6 = \text{atan2}\left(\frac{s_6}{c_6}\right) \tag{3.44}$$

3.2.3 Jacobian Matrix

The Jacobian matrix plays a crucial role in understanding the relationship between joint velocities and the velocity of the end-effector. It's derived from the kinematic model and represents how changes in the joint variables influence the end-effector's position and orientation. It's used in control algorithm design, especially when dealing with velocity control, trajectory tracking, and force control of a PUMA 560 robot manipulator.

Hence, for the PUMA 560 robot manipulator the Jacobian matrix denoted $J(\theta)$ is a 6×6 matrix that relates the joint velocities $\dot{\theta}$ to the end-effector's linear and angular velocities $\dot{x}$ and $\dot{\omega}$ is given by equation (3.45).

$$\begin{pmatrix} \dot{x} \\ \dot{\omega} \end{pmatrix} = J(\theta) \cdot \dot{\theta} \tag{3.45}$$

where $\dot{x}$ is the linear velocity and $\dot{\omega}$ is the angular velocity of the end-effector, while $\dot{\theta}$ represents the joint velocities. The Jacobian is used to solve both forward and inverse velocity problems, and it can also be used to compute forces and torques in task space using the transpose of the Jacobian.

3.2.4 Velocity and Acceleration Kinematics

The forward kinematics is established once in the previous subsection, so the next step is to consider the velocity and acceleration of the end-effector. These are important in control tasks of trajectory tracking, where both the position and velocity of the end-effector need to be controlled.

The velocity of the end-effector can be obtained by differentiating the position vector with respect to time, and the acceleration is the second derivative. Using the Jacobian matrix, we can express the velocity and acceleration of the end-effector as in (3.46).

$$\dot{x} = J(\theta) \cdot \dot{\theta} \tag{3.46}$$

$$\ddot{x} = \frac{d}{dt}\left(J(\theta) \cdot \dot{\theta}\right) \tag{3.47}$$

where $\dot{x}$ and $\ddot{x}$ represent the linear velocity and acceleration of the end-effector, respectively, and $J(\theta)$ is the Jacobian matrix.

3.3 Robot Dynamics Modeling

The dynamic modeling of a PUMA 560 robot manipulator involves analyzing the forces and torques required to achieve a desired motion of joint position. The dynamic model is essential for the controller design, as it allows the prediction and adjustment of joint torques to follow a specified trajectory. The most widely used approach to derive the dynamics of robotic manipulators is the Newton-Euler method, which provides equations of motion by balancing the forces and torques acting on each link.

The Newton-Euler formulation breaks down the dynamics of each link into a series of recursive calculations that account for the inertia, Coriolis, and centrifugal forces. Hence, the dynamic model of the PUMA 560 robot manipulator is expressed by (3.48).

$$\tau = M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) \tag{3.48}$$

where τ represents the vector of joint torques, serving as the control inputs required to drive the joints. $M(\mathbf{q})$ is the inertia matrix, which characterizes the mass properties of the robot and its resistance to acceleration. $C(\mathbf{q}, \dot{\mathbf{q}})$ denotes the Coriolis and centrifugal matrix, accounting for the forces arising from velocity interactions between the links. $G(\mathbf{q})$ is the gravity vector, representing the gravitational forces acting on each link of the manipulator.

The PUMA 560 robotic manipulator driven by the DC motors is modeled by the following nonlinear dynamic system (3.49) [28].

$$\begin{aligned} &M(q)\ddot{q} + B(q)[\dot{q}][\dot{q}] + C(q)[\dot{q}^2] + G(q) = N(\tau_m - \tau_f) \\ &L_a\dot{I} + R_aI + K_eN\dot{q} = V \\ &\tau_m = K_mI \end{aligned} \tag{3.49}$$

where, $\tau_m \in \mathbb{R}^{6\times1}$ denotes actuators torques vector, $\tau_f \in \mathbb{R}^{6\times1}$ actuators friction torques,

$q, \dot{q}, \ddot{q}$ denote joints angular positions, velocities, and accelerations vectors. Symbols $[\dot{q}]$ and $[\dot{q}^2]$ are notation for the $n(n-1)/2$-vector of velocity products and the n-vector of squared velocities and given by (3.50).

$$\begin{aligned} [\dot{q}] &= [q_1\dot{q}_2, q_1\dot{q}_3, \ldots, q_1\dot{q}_n, q_2\dot{q}_3, \ldots, q_{n-1}\dot{q}_n]^T \\ [\dot{q}^2] &= [\dot{q}_1^2, \dot{q}_2^2, \ldots, \dot{q}_n^2]^T \end{aligned} \tag{3.50}$$

$M(q) \in \mathbb{R}^{6\times6}$ is the inertia matrix, $B(q) \in \mathbb{R}^{6\times15}$ the Coriolis torques matrix, $C(q) \in \mathbb{R}^{6\times6}$ the centrifugal torques matrix, and $G(q) \in \mathbb{R}^{6\times1}$ the gravity torques. N and R_a are diagonal matrices where $N \in \mathbb{R}^{6\times6}$ ratios diagonal matrix, and $R_a \in \mathbb{R}^{6\times6}$ is the armature resistance diagonal matrix. To facilitate the control task, we propose to simplify (3.49) by making the control of the PUMA 560 by directly the torques. Hence, the torques are assumed to be provided by the motor. Resulting the PUMA 560 robot manipulator joint control primarily by the torques, in which the motor control is not discussed in this thesis. To describe the system's states controlled in this design, we define the state vector as shown in equation (3.51).

$$x = [q_1, q_2, q_3, \ldots, q_6, \dot{q}_1, \dot{q}_2, \dot{q}_3, \ldots, \dot{q}_6]^T \tag{3.51}$$

The output vector is defined as in equation (3.52).

$$y = [q_1, q_2, \ldots, q_6]^T \in \mathbb{R}^{6\times1} \tag{3.52}$$

3.4 SolidWorks Design of the PUMA 560 Manipulator

This section presents the comprehensive modeling and design process of the PUMA 560 robot manipulator using SolidWorks. The design involves three primary stages to ensure a cohesive integration of design and control functionalities. The first stage focuses on SolidWorks modeling, where a precise 3D representation of the manipulator is created based on its geometric and structural parameters. In the second stage, the SolidWorks model is exported into MATLAB, enabling the simulation and analysis of the manipulator's performance. Finally, the third stage involves developing control strategies tailored to achieve accurate position control and trajectory tracking of the manipulator's end-effector, in which this is done in the next chapter four. This structured workflow ensures seamless interaction between the design environment in SolidWorks and the control and simulation environment in MATLAB, providing a robust foundation for the manipulator's application.

The PUMA 560 is a six-degree-of-freedom (6DOF) robotic arm with revolute joints. Each link is modeled in SolidWorks based on precise geometric and physical parameters. The assembly is constrained to represent the kinematic structure accurately.

3.4.1 Part Design and Assembly

The design process of the PUMA 560 robot manipulator began with the creation of individual parts for each link in SolidWorks. Each link was designed as a separate part file to ensure modularity and accuracy. The dimensions of these links were derived from the standard specifications of the PUMA 560 manipulator, ensuring that the geometric properties of the model were consistent with the actual physical system. This approach allowed for precise modeling of the manipulator's structure and provided a solid basis for further assembly.

Once the links were designed, they were connected using revolute joints to form the complete assembly. Revolute joints were specifically chosen to replicate the actual motion constraints of the PUMA 560 manipulator. These joints allow rotation about a single axis, closely mimicking the physical behavior of the manipulator's joints. Special

attention was given to the alignment and positioning of these joints to ensure that the assembled model correctly represented the kinematic relationships and degrees of freedom of the actual robot. The assembly was validated within SolidWorks to verify that the range of motion and joint constraints matched the expected specifications.

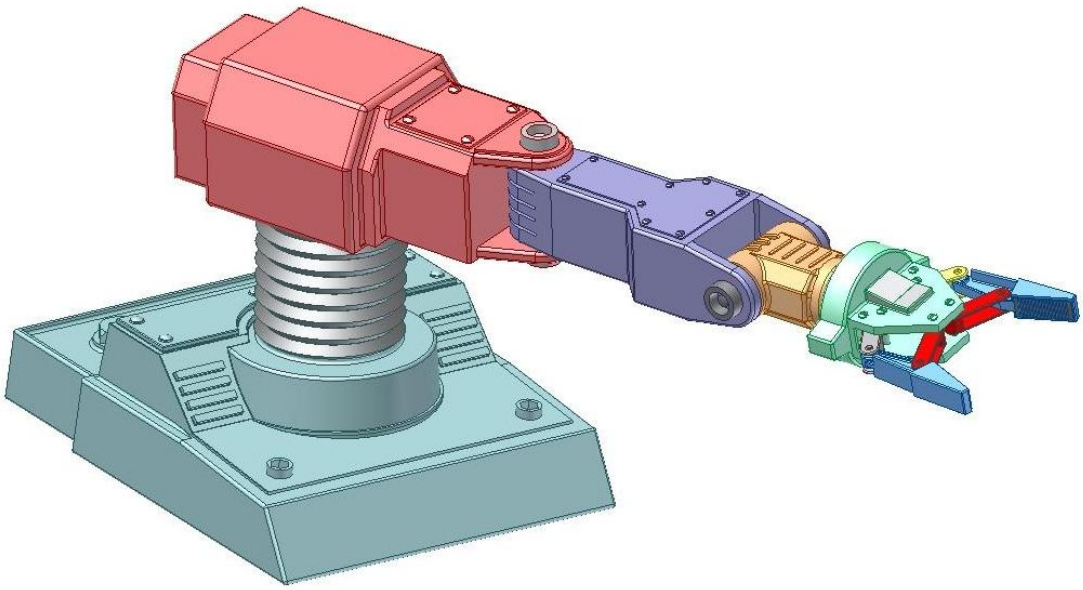


Figure 3.4: PUMA 560 Robot Manipulator

Material properties were also a critical aspect of the design. Aluminum was selected as the material for the links due to its favorable characteristics, such as a high strength-to-weight ratio and ease of machining. The density and moments of inertia for aluminum were incorporated into the model to accurately simulate the dynamics of the manipulator. This step ensured that the simulated behavior in motion studies and later exported simulations would closely reflect real-world performance. By assigning these material properties, the SolidWorks model provided realistic dynamic characteristics, making it suitable for subsequent integration with control systems in MATLAB.

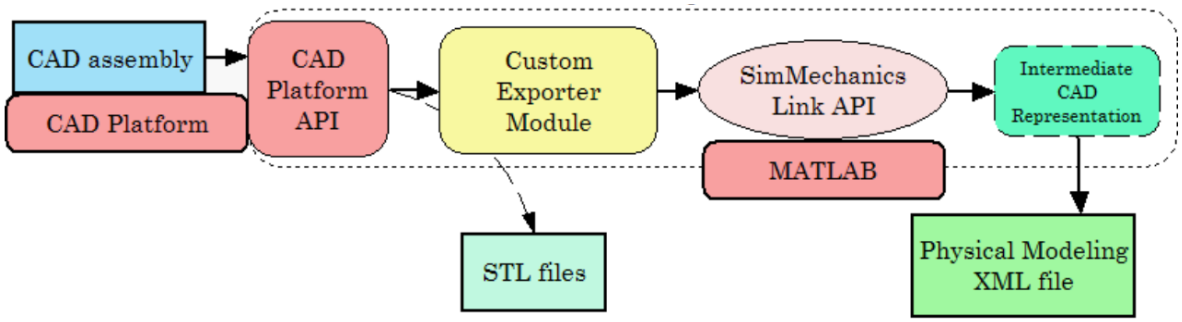


Figure 3.5: CAD Assembly into a SimMechanics-compatible Format Conversion Process

Figure 3.5 describes the custom export mechanism used to translate a CAD assembly into a SimMechanics-compatible format. Starting with a CAD assembly in a specific CAD platform, the CAD Platform API works in conjunction with a custom exporter module to generate two outputs: STL files for the geometry of individual components and a Physical Modeling XML file that contains kinematic and dynamic data. These outputs are then processed using the SimMechanics Link API within MATLAB, generating an Intermediate CAD Representation that serves as input for Simulink and Simscape simulation.

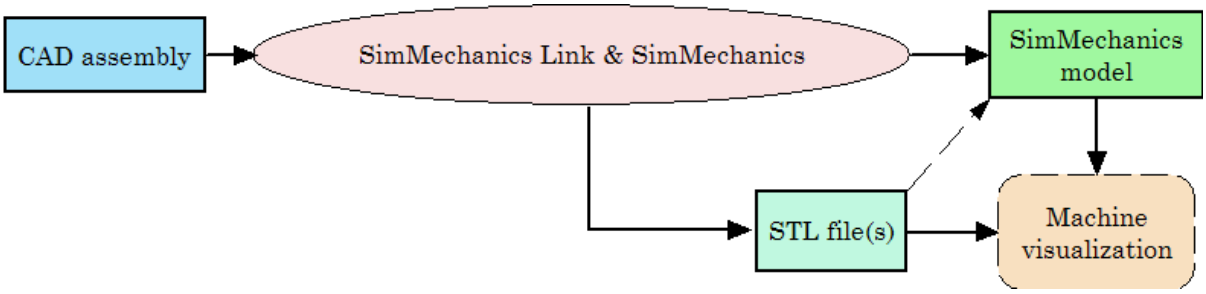


Figure 3.6: CAD Translation Process using SimMechanics Link and Simscape Environment

Figure 3.6 provides a high-level overview of the CAD translation process for creating SimMechanics models. A CAD assembly is imported into SimMechanics via the SimMechanics Link tool, which outputs two essential components: STL files, used for visualizing the robot or machine's geometry, and a SimMechanics model for defining the system's kinematic and dynamic properties. These outputs can be visualized and utilized in MATLAB's Simscape environment to simulate and analyze the mechanical system's behavior.

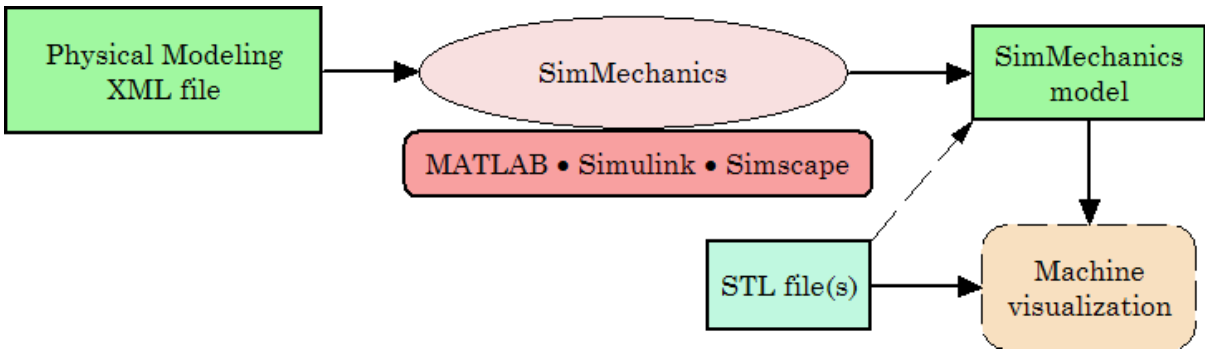


Figure 3.7: XML Import Process for Generating SimMechanics Models and STL files for Visualization

The process given in Figure 3.7 highlights the 'XML import workflow' for creating a SimMechanics model. A 'Physical Modeling XML file', containing kinematic and dynamic information about the system, is imported into MATLAB's Simscape Multibody environment. This generates a 'SimMechanics model' for simulation and control design while also producing 'STL files' that enable accurate 3D visualization of the robot's physical components. This workflow ensures seamless integration of CAD data into MATLAB for further analysis, design, and control.

3.4.2 Validation of the Model in SolidWorks

The validation of the PUMA 560 model in SolidWorks was an essential step to ensure the correctness and reliability of the assembly. This process began with conducting detailed motion studies to verify the range of motion for each joint. These studies involved simulating the rotational and translational movements permitted by the revolute joints, ensuring that they conformed to the physical constraints and degrees of freedom defined for the PUMA 560 manipulator. Any discrepancies observed during these simulations were corrected by refining the joint parameters and ensuring proper alignment within the assembly. This step was crucial for confirming that the virtual model accurately represented the kinematic behavior of the actual robot manipulator.

In addition to verifying joint motions, the dynamic properties of the assembled model were thoroughly evaluated. Properties such as the center of mass and moments of inertia were calculated for each link and for the entire manipulator. These properties were

compared with theoretical values derived from the specified dimensions and material properties of the model. By confirming the accuracy of these dynamic parameters, the model was prepared for realistic simulations, ensuring that it could be effectively integrated into MATLAB for further control and trajectory analysis. This comprehensive validation process provided confidence in the fidelity of the SolidWorks model, making it a reliable representation of the PUMA 560 manipulator.

3.5 Exporting the Model to MATLAB

The process of exporting the PUMA 560 robot manipulator model from SolidWorks to MATLAB involved careful selection of file formats and a structured import procedure to ensure accurate representation and compatibility. This step was essential for enabling simulation and control design within the MATLAB environment.

3.5.1 Exporting Formats

To facilitate a seamless transfer, the SolidWorks model was exported in multiple formats suitable for use in MATLAB. The primary export format was the Unified Robot Description Format (URDF), which is widely used for representing the kinematic and dynamic parameters of robotic systems. The URDF file encapsulated critical information such as joint types, link dimensions, and inertial properties, providing a comprehensive description of the manipulator. Additionally, stereolithography (STL) files were generated to capture the geometric details of each link. These files were essential for visualizing the manipulator's structure in MATLAB and ensuring that the physical appearance matched the SolidWorks model. For physics-based simulations, the Simscape Multibody format was utilized, allowing direct integration of the model into Simscape's simulation environment. This approach ensured that the exported model retained both its geometric fidelity and dynamic behavior, making it suitable for advanced simulations and control implementation.

3.5.2 MATLAB Import Process

The imported SolidWorks model was integrated into MATLAB using a structured process to ensure accuracy and usability. The URDF file served as the primary input for creating the manipulator model within MATLAB. Using the following commands, the model was imported and visualized:

```
robot = importrobot('puma560.urdf');
show(robot, 'Frames', 'on');
```

The 'importrobot' function automatically parsed the URDF file to create a representation of the manipulator, including its kinematic structure and joint configurations. The 'show' function enabled the visualization of the manipulator, with coordinate frames displayed for each link and joint, aiding in the validation process.

To confirm the accuracy of the imported model, the manipulator's structure was compared against the original SolidWorks design. Joint movements were tested by simulating rotations and translations, ensuring that the range of motion and constraints adhered to the expected specifications. Any discrepancies were resolved by refining the URDF file or adjusting the SolidWorks model, creating a robust link between the design and simulation environments. This meticulous process ensured that the exported model could be effectively used for trajectory planning, control system design, and dynamic simulations in MATLAB.

Chapter 4

Controller Design for PUMA 560 Robot Manipulator

Adaptive control is able to adjust to a manipulated system with parameters that change constantly, or are originally unknown. The adaptive control can be classified into the three types, model reference adaptive control, self-tuning and gain-scheduled. However, this chapter provides a survey and design of Model Reference Adaptive Control (MRAC) of PUMA 560 robot manipulator as in Figure 4.1.

Adaptive control ensures system performance even when the model's accuracy is limited. MRAC, in particular, offers significant benefits: enhancing repeatability and accuracy in PUMA 560 robotic system. To achieve high performance in applications where PUMA 560 robot's dynamics may change, a control system of MRAC is essential, as it adapt to these dynamic variations and maintain consistent control. By continuously adjusting its parameters, MRAC compensates for disturbances and parameter shifts, making it particularly suited for complex, nonlinear PUMA 560 robotic tasks.

However, in the application of MRAC to the PUMA 560 robot system, ensuring stability is challenging due to the highly nonlinear nature of robotic dynamics. This complexity makes it difficult to guarantee MRAC stability, which is one reason why recent research on MRAC for robotic mechanisms is limited. In this work, stability is addressed using the Lyapunov rule technique, providing a foundation for reliable control in these nonlinear systems.

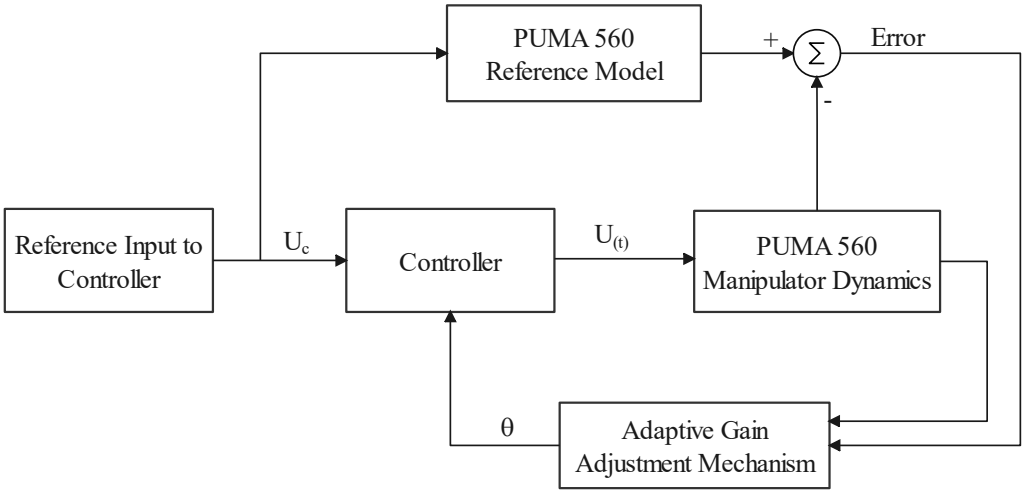


Figure 4.1: Model Reference Adaptive Control of PUMA 560 Robot Manipulator [12].

The MRAC in Figure 4.1 is designed to enable the PUMA 560 robot manipulator to accurately follow a desired trajectory by continuously adjusting its control parameters. This adaptive control structure consists of the components: the reference input, the reference model, the controller, the PUMA 560 manipulator dynamics, the error feedback, and the adaptive gain adjustment mechanism. Each component plays a unique role in ensuring precise trajectory tracking and robust performance under varying conditions.

The system begins with the reference input, which specifies the desired trajectory that the PUMA 560 manipulator should follow. This input acts as a benchmark for the adaptive control framework, defining the overall performance objectives. The reference input signal is simultaneously fed to both the reference PUMA 560 model and the controller.

The reference model block represents the ideal output behavior corresponding to the desired input. It is a mathematical model designed to generate a trajectory that the actual manipulator output should replicate. The reference model serves as the target behavior for the adaptive control system, guiding the manipulator towards optimal tracking of the desired trajectory of position, velocity, and acceleration.

The PUMA 560 manipulator dynamics block represents the actual dynamics of the PUMA 560 manipulator, incorporating its kinematic and dynamic characteristics. This block captures the manipulator's response to the control input $U(t)$, which drives its movement towards the desired trajectory. However, the dynamics of the manipulator are

subject to uncertainties, disturbances, and unmodeled dynamics, which is influenced by load variations, friction, and external forces. These variations impact the manipulator's performance, and hence the MRAC system is designed to compensate for these factors.

The controller block is responsible for generating the control signal $U(t)$ that drives the manipulator. It receives as inputs the reference input U_c and the adjustable control parameters, provided by the adaptive gain adjustment mechanism. The controller converts these inputs into the appropriate control action, aiming to reduce the difference between the actual manipulator output and the output of the reference model. This adaptive approach allows the controller to dynamically adjust its response, enabling the system to maintain precise tracking even when the manipulator dynamics change.

The error signal is computed as the difference between the reference model output and the actual output of the manipulator. This error signal provides feedback on the system's performance, directly reflecting the accuracy of the tracking. Minimizing this error signal is one of the primary goals of the adaptive control system, as a lower error indicates better tracking accuracy and overall system stability.

The adaptive gain adjustment mechanism block is the core of the MRAC system, responsible for continuously updating the control parameters based on the error signal. Using an adaptive algorithm, this mechanism adjusts the control gains to minimize the tracking error over time. This continuous adjustment allows the control system to adapt to variations in the manipulator's dynamics, disturbances, or any inaccuracies in the model. By dynamically modifying the controller's parameters, the adaptive gain adjustment mechanism ensures that the PUMA 560 manipulator maintains stability and achieves high tracking precision, regardless of changing operating conditions.

4.1 Model Reference Adaptive Control

The main purpose of adaptive control is to control of systems which have parameters that are not accurately known [29]. This inaccuracy can arise from changes in environment, such as temperature, and density of air. Some system changes can be unpredictable, and ordinary closed-loop systems may not respond properly when the system transfer

function varies. In many such circumstances, adaptive control is justified. Among many approaches for adaptive control of such systems, the Model Reference Adaptive Control has become a popular and widely accepted approach for incorporating the adaptive feature into an ordinary feedback control system.

A reference model is apriority chosen so that it describes the desired response of the plant controller combination; this choice is made by the designer, based on experience. The plant output is compared with the reference model output; the difference between them, called a system output error is used to adjust the controller parameters such a way until this very error signal is driven to zero. In this MRAC, two features play crucial role: the stability of the overall system, including the adaptive mechanism in the sense that all signals in the entire system remain bounded; and the plant output response meeting the reference model output response, with minimum error; that is, the system output error converging to zero. Consequently, after an MRAC scheme is designed, the entire system is simulated to check whether the above two feature are satisfied. This is demonstrated by checking the stability of the designed controller via the Lyapunov method of stability analysis by the end of this chapter.

The basic idea behind it is to design a closed loop adaptive controller that works on the principle of adjusting the control parameters so that the output of the actual plant tracks the output of a reference model having the same reference inputs.

MRAC is used to maintain consistent performance of a system in the presence of uncertainty and variation in plant parameters. It implies that the system is capable of accommodating unpredictable disturbances, whether this disturbances arise within the system or external to it. The basic objective of adaptive controller is to maintain a consistent performance of a system in the presence of uncertainty or unknown variation in the plant parameters, which may occur due to non-linear actuators, change in the operating conditions of the plant and non-satisfactory disturbances acting on the plant.

4.2 Model Reference Adaptive Control of PUMA 560

This section describes the MRAC designed for the PUMA 560 robotic manipulator. The control scheme ensures the tracking of desired joint positions, velocities, and accelerations while adapting to system uncertainties. The MRAC is usually described by a set of adjustable parameters and provides tracking. Its parameters depend on the adaptation algorithm. The goal of this part is to consider the initial plant condition for calculation purposes and to achieve overall stability.

A MRAC must be designed to provide overall system performance and stability for a nominal plant without uncertainty. Thus, it can be thought of as a baseline or nominal controller. The type of controller is dictated by the objective of the control design. A controller can be linear or nonlinear, but as always, nonlinear controllers are much more difficult to design, analyze, and ultimately certify for operation in real systems. In this thesis Desired Compensation Adaption Law (DCAL) control law is going tobe used and designed in the following sections.

4.2.1 Desired Compensation Adaption Law

The control technique employed in this thesis is the DCAL [30], which is a sophisticated adaptive control approach, and given by the block diagram in details in Figure 4.2. DCAL is specifically designed to address the challenges posed by uncertain and time-varying parameters in dynamic systems, particularly in robotic manipulators of the PUMA 560. Robotic manipulators, especially those with multiple degrees of freedom as the PUMA 560, often operate in environments where parameters such as link mass, inertia, and external forces are not precisely known. These parameters may vary due to changes in the operating environment, wear and tear on components, or unmodeled disturbances such as external forces or varying payloads.

The main goal of DCAL in the thesis is to ensure that the robot manipulator accurately follows a desired trajectory by continuously adjusting the control parameters based on the tracking error. This is crucial in robotics, where precise motion and trajectory tracking are necessary for successful task completion. DCAL addresses the uncertainties

in the robot's dynamics, such as variations in the system's mass matrix or changes in the dynamics due to variations in the payload, by adapting the control gains in real-time to account for these changes.

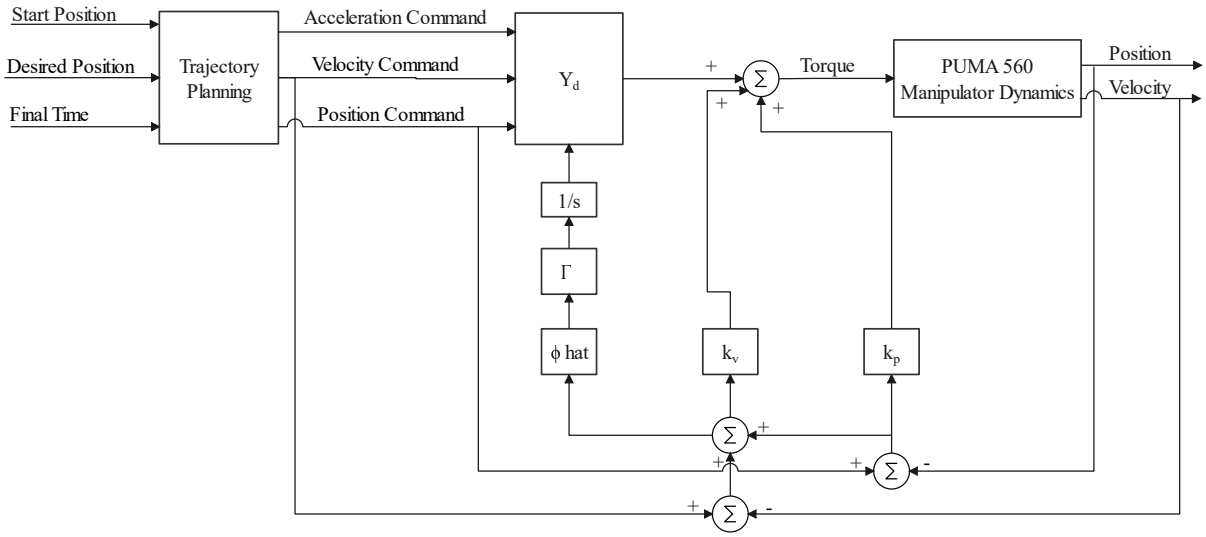


Figure 4.2: Block Diagram of DCAL

The DCAL works by using the desired trajectory information, including position, velocity, and acceleration, to compute a feedforward term. This feedforward term compensates for dynamic changes and external disturbances, such as payload changes, that could otherwise lead to inaccuracies in the robot's movement. By using the real-time error measurements between the desired and actual positions of the end effector, DCAL adapts its control parameters continuously. This ensures that the manipulator remains on the desired trajectory despite uncertainties in the robot's dynamics. This feature is particularly important for the PUMA 560 robot manipulator, which are subject to external forces, such as interaction with objects, or changes in operating conditions, such as temperature variations or mechanical wear.

By continuously adjusting the control parameters based on real-time error measurements, DCAL ensures that the system achieves accurate trajectory tracking without requiring a detailed prior model of the system dynamics. This reduces the computational burden and avoids the need for complex modeling of the manipulator's full dynamics. The PUMA 560 robot manipulator, with its six degrees of freedom, has a complex kinematic and dynamic model. DCAL's ability to adapt and perform without requiring an

exhaustive model makes it computationally efficient, while maintaining high performance in real-world scenarios.

4.2.1.1 Desired Virtual Input Velocity

The desired reference velocity for the MRAC scheme, implemented using the DCAL, is expressed as (4.1) [31, 32, 33, 34].

$$V_m = q_d k_p + \dot{q}_d + \ddot{q}_d \tag{4.1}$$

where q_d represents the desired joint positions, which is a vector of six values corresponding to the six joints of the PUMA 560 manipulator. k_p is the proportional gain matrix, which scales the error between the desired and actual joint positions. It is a 6×6 matrix. $\dot{q}_d$ refers to the desired joint velocities, which is a vector in $\mathbb{R}^6$, representing the velocity at which each joint should move. $\ddot{q}_d$ represents the desired joint accelerations, which is also a vector in $\mathbb{R}^6$, corresponding to the acceleration of each joint. These terms together form the desired reference velocity for the manipulator, ensuring that it follows the intended trajectory while accounting for joint positions, velocities, and accelerations.

4.2.1.2 Online Computation of the Regression Matrix

In the thesis, a significant optimization is introduced by eliminating the online computation of the regression matrix, which is typically computed in real-time during each control cycle. The regression matrix, $Y_d(q_d, \dot{q}_d, \ddot{q}_d)$, given by equation (4.2), represents the system's dynamics, including the mass matrix, Coriolis and centrifugal forces, and gravitational effects [35, 36]. For the PUMA 560 robot manipulator, with six DOF and highly coupled dynamics, computing this matrix in real-time can be both computationally expensive and time-consuming.

$$Y_d(q_d, \dot{q}_d, \ddot{q}_d)\phi = M(q_d)\ddot{q}_d + V_m(q_d, \dot{q}_d)\dot{q}_d + G(q_d) \tag{4.2}$$

In many adaptive control systems, this matrix is calculated during each control cycle, which involves solving complex dynamic equations. This process introduces delays, which are particularly problematic in systems that require precise control, as in robotic arms of the PUMA 560 robot manipulator. Real-time computation of this matrix can hinder the system's ability to react quickly to changes in the environment, leading to degraded performance in fast response situations.

To overcome this challenge, the online computation of the regression matrix is eliminated by precomputing the matrix offline and using simplified approximations. This means that the system no longer needs to calculate the regression matrix at every control cycle. Instead, it uses precomputed and simplified models of the robot's dynamics, which are already known and estimated based on prior data. By doing this, the system can focus computational resources on more critical tasks, such as real-time error correction and generating control signals for the actuators.

This optimization significantly reduces the computational load during each control cycle, allowing the PUMA 560 manipulator to operate with reduced latency. As a result, the control loop operates faster and can respond more quickly to disturbances or changes in the robot's dynamics, such as those introduced by external forces or varying payloads. In systems with limited computational power, such as embedded controllers in robots, this elimination of time-intensive matrix computations allows the system to maintain high performance without requiring significant hardware upgrades.

By precomputing the regression matrix and simplifying the system's dynamic model, the PUMA 560 can maintain high performance with improved responsiveness. This is especially important for real-time applications where precise control is crucial, such as in automated assembly or surgical robots, where delays or inaccuracies can lead to errors or even failure of the task. The elimination of online computation also makes the system more scalable, as it reduces the demand for high-end computing resources, making the control approach more practical for real-world applications.

4.2.1.3 Parameter Error Vector

In adaptive control and parameter estimation, the parameter error vector plays a critical role in assessing the deviation between true and estimated parameters. It is defined as in equation 4.3 [35].

$$\tilde{\phi} = \phi - \hat{\phi} \tag{4.3}$$

where ϕ represents the true parameter vector and $\hat{\phi}$ represents the estimated parameter vector. The parameter error vector, $\tilde{\phi}$, directly influences the quality of system modeling and control. The true parameter vector, ϕ, encapsulates the intrinsic dynamic properties of the PUMA 560 robotic system, such as link masses, moments of inertia, and gravitational constants [36]. However, these parameters are often unknown or difficult to measure directly in practice. The estimated parameter vector, $\hat{\phi}$, is obtained through estimation algorithms or system identification techniques, which may operate online or offline, depending on the control framework [21].

4.2.1.4 Estimate of Parameter Vector

The estimated parameter vector, denoted by $\hat{\phi}$, contains the system's approximated dynamic parameters. For the PUMA 560 robotic manipulator, the estimated parameter vector represented by equation 4.4.

$$\hat{\phi} = \begin{pmatrix} \hat{m}_1 & \hat{m}_2 & \hat{m}_3 & \hat{m}_4 & \hat{m}_5 & \hat{m}_6 \end{pmatrix}^\top \tag{4.4}$$

where $\hat{m}_1$ to $\hat{m}_6$ represent the estimated masses of the six links of the PUMA 560 robot manipulator. These parameters are either updated dynamically in real-time adaptive control systems or derived offline through system identification techniques. However, in this thesis the dynamic update method is employed. The parameter update process for $\hat{\phi}$ is governed by an adaptation law, which minimizes the parameter error given in 4.3.

The form of the adaptation law is given by equation 4.5.

$$\dot{\hat{\phi}} = \Gamma Y_d^T r \tag{4.5}$$

where Γ is a positive definite adaptation gain matrix, Y_d is the regression matrix representing the system dynamics, and r is the error term defined in section 4.2.1.5. The parameter estimation is updated by minimizing the tracking error, ensuring that the parameter estimates converge to the true values over time. The term $Y_d^T r$ essentially represents the gradient of the performance metric with respect to the parameters, which is used to adjust $\hat{\phi}$. This form of the adaptation law is commonly used in adaptive control systems to guarantee convergence of the estimated parameters to their true values [37].

4.2.1.5 Adaptive Gains Γ and Update Laws

The design process involves setting up a controller with adjustable parameters and a structure that matches the structure of the given plant. Hence, the equations that describe how the controller parameters need to be adjusted are called adaptive laws. The design of stable MRAC systems primarily focuses on designing the adaptive mechanism, which involves the creation of stable adaptive laws to adjust the controller parameters.

The controller adjusts its parameters so that the actual plant can track the reference model. Several mathematical approaches are used in the adjustment mechanism, as the likes of the MIT rule and Lyapunov stability theory. This component alters the parameters of the controller so that the actual plant can track the reference model. In this thesis, we use the Lyapunov stability theory.

The basic block diagram of an MRAC system is shown in Figure 4.1. As shown in the figure, $y_m(t)$ represents the output of the reference model, $y(t)$ represents the output of the actual plant, and the difference between them is denoted by $e(t)$ and given by (4.6).

$$e(t) = y(t) - y_m(t) \tag{4.6}$$

The system dynamics that governs the PUMA 560 robot manipulator in a trajectory

tracking pick and place application of the system are the position and velocity deviations between the reference and actual trajectories. This error dynamics is given by (4.7).

$$\begin{aligned} e_p &= q_d - q \\ e_v &= \dot{q}_d - \dot{q} \end{aligned} \tag{4.7}$$

where $e_p, e_v \in \mathbb{R}^6$ are the position and velocity tracking errors.

The composite error term that considers the effects of the position and velocity error terms $\omega = \dot{r} \in \mathbb{R}^6$ is given by (4.8).

$$\omega = P_2 \times e_p + P_3 \times e_v \tag{4.8}$$

where $P_2, P_3 \in \mathbb{R}^{6\times6}$ are positive definite symmetric matrices that are going to be designed and used using the Lyapunov stability analysis methods.

Now the adaptive gains are defined by the (4.9).

$$\begin{aligned} \dot{\delta_1} &= -a \times \omega \times q^{'} \\ \dot{\delta_2} &= a \times \omega \times \dot{q}^{'} \\ \dot{\delta_v} &= b \times \omega \times V_m^{'} \end{aligned} \tag{4.9}$$

where a, b is adaptive coefficients of diagonal matrices, and $q^{'}$, $\dot{q}^{'}$ is scaled current joint positions and velocities. The combined adaptive gain matrix is then given by 4.10.

$$\Gamma = \text{diag}[\delta_1, \delta_2, \delta_v] \tag{4.10}$$

4.2.2 Control Law Formulation

The total control torque is composed of two main components: a feedback control term U_k and an adaptive control term U_u. Together, they ensure the system achieves stability and robust performance. The total control torque is given by (4.11).

$$U = U_k + U_u \tag{4.11}$$

4.2.2.1 Feedback Control Law

The feedback control term U_k is designed to stabilize the system and provide trajectory tracking. It is expressed as in (4.12). This term acts as a Proportional-Derivative (PD) controller, with k_p ensuring positional error correction and k_v providing damping for smooth responses.

$$U_k = V_m - k_v\dot{q} - k_p q \tag{4.12}$$

where $V_m \in \mathbb{R}^6$ is given by (4.1) and it is the desired virtual control input derived from a higher-level control also called the trajectory planner. And $k_v \in \mathbb{R}^{6\times6}$ is the velocity feedback gain matrix that provides damping to reduce oscillations, $\dot{q} \in \mathbb{R}^6$ is the velocity state vector, $k_p \in \mathbb{R}^{6\times6}$ is the position feedback gain matrix that enforces error correction., and $q \in \mathbb{R}^6$ the position state vector.

4.2.2.2 Adaptive Control Law

The adaptive control term U_u compensates for system uncertainties, external disturbances, or unmodeled dynamics. It is expressed by (4.13). This term dynamically adjusts the control effort to handle uncertainties and improve robustness, hence; it is called an adaptive control law design.

$$U_u = \delta_v V_m - \delta_2\dot{q} - \delta_1 q \tag{4.13}$$

where $\delta_v \in \mathbb{R}^{6\times6}$ is an adaptive gain matrix that adjusts the influence of the desired input V_m, $\delta_2 \in \mathbb{R}^{6\times6}$ is an adaptive gain matrix compensating for uncertainties in the velocity state, and $\delta_1 \in \mathbb{R}^{6\times6}$ is an adaptive gain matrix addressing uncertainties in the positional dynamics.

4.2.2.3 Total Control Torque

Combining the feedback and adaptive control laws, the total control torque is given by equation (4.14).

$$U = \left(V_m - k_v\dot{q} - k_p q\right) + \left(\delta_v V_m - \delta_2\dot{q} - \delta_1 q\right) \tag{4.14}$$

Rearranging the terms, the control law becomes in (4.15).

$$U = (1 + \delta_v)V_m - (k_v + \delta_2)\dot{q} - (k_p + \delta_1)q \tag{4.15}$$

The total control law integrates feedback and adaptive components, each playing a distinct role in ensuring the robustness and stability of the system. The term $(1 + \delta_v)V_m$ adapts the influence of the desired input V_m, dynamically scaling it to account for uncertainties or external disturbances, thereby enhancing the system's ability to follow the desired trajectory accurately. The component $(k_v + \delta_2)\dot{q}$ combines the velocity feedback gain k_v with an adaptive adjustment δ_2. This ensures that the system compensates for uncertainties in the velocity state, mitigating potential instability or performance degradation. Similarly, $(k_p + \delta_1)q$ modifies the position feedback gain k_p by incorporating an adaptive correction δ_1, enabling the controller to handle uncertainties in the positional dynamics effectively.

In essence, the control law achieves its objectives by leveraging the feedback term U_k to stabilize the system and track the desired trajectory, while employing the adaptive term U_u to address unknown system parameters and external disturbances. This synergy between the feedback and adaptive components results in a control strategy that maintains robust performance under varying conditions and ensures the desired system behavior is consistently achieved.

4.2.3 Joint Torque Dynamics

The torque dynamics for the PUMA 560 robot manipulator is now given by (4.16).

$$\begin{aligned} \tau &= U + (M_{(\mathrm{q})} - I)\ddot{q} + C_{(q,\dot{q})} + G_{(\mathrm{q})} \\ &= Y_d(q_d, \dot{q}_d, \ddot{q}_d)\hat{\phi} + k_v r + k_p e + k_a \|e\|^2 r \end{aligned} \tag{4.16}$$

where $I = \mathbb{I}^{6\times6}$ is identity matrix, $M(q) \in \mathbb{R}^{6\times6}$ is configuration-dependent joint inertia matrix, $C(q, \dot{q}) \in \mathbb{R}^6$ is Coriolis and centrifugal forces, $G(q) \in \mathbb{R}^6$ is gravity vector, and $\tau \in \mathbb{R}^6$ is the net torque acting on the manipulator.

4.3 Stability Analysis

To ensure the stability of the closed-loop system, a Lyapunov candidate function is formulated based on the system's dynamics and the adaptive control law. The derivation highlights the interaction between the feedback and adaptive control components, ensuring robustness against uncertainties and external disturbances. Hence, sets the conditions for the designed control system to result a stable control system design of PUMA 560 robot manipulator.

The Lyapunov candidate function is chosen as in (4.17).

$$V = \frac{1}{2}e^T M(q)e + \frac{1}{2}\text{tr}(\tilde{\Delta}^T \Gamma^{-1} \tilde{\Delta}), \tag{4.17}$$

where $e = q_d - q$ is the joint position tracking error vector, $M(q) \in \mathbb{R}^{6\times6}$ is the configuration-dependent joint inertia matrix, which is positive definite, $\tilde{\Delta} = \Delta - \hat{\Delta}$ represents the parameter estimation error, with Δ as the true uncertainty and $\hat{\Delta}$ its estimate, and $\Gamma \in \mathbb{R}^{6\times6}$ is a positive definite adaptation gain matrix.

The Lyapunov function combines the energy of the tracking error with the parameter estimation error. Since $M(q)$ and Γ are positive definite, $V > 0$, ensuring it is a valid positive definite Lyapunov candidate function.

The derivative of the Lyapunov function V is given by (4.18).

$$\dot{V} = \frac{1}{2}\dot{e}^T M(q)e + \frac{1}{2}e^T \dot{M}(q)e + e^T M(q)\dot{e} + \text{tr}(\tilde{\Delta}^T \Gamma^{-1} \dot{\tilde{\Delta}}) \tag{4.18}$$

Using the joint torque dynamics from (4.16), the control law from (4.15), and expressing $\dot{e}$ and $\ddot{e}$ in terms of system dynamics, we get the equation in (4.19).

$$M(q)\ddot{e} + C(q,\dot{q})\dot{e} + G(q) = \Delta + \text{adaptive terms} \tag{4.19}$$

Substituting $\ddot{e}$ into $\dot{V}$, we simplify (4.18) as in 4.20.

$$\dot{V} = -e^T(k_p + \delta_1)e - \dot{e}^T(k_v + \delta_2)\dot{e} + e^T\Delta + \text{tr}(\tilde{\Delta}^T \Gamma^{-1} \dot{\tilde{\Delta}}) \tag{4.20}$$

To eliminate the effects of $e^T\Delta$ and ensure negative definiteness of $\dot{V}$, the following adaptation law (4.21) is applied.

$$\dot{\hat{\Delta}} = -\Gamma e^T \tag{4.21}$$

which gives:

$$\text{tr}(\tilde{\Delta}^T\Gamma^{-1}\dot{\tilde{\Delta}}) = -\text{tr}(\tilde{\Delta}^T\tilde{\Delta}) \tag{4.22}$$

Substituting the adaptation law back into $\dot{V}$ results in (4.23).

$$\dot{V} = -e^T(k_p + \delta_1)e - \dot{e}^T(k_v + \delta_2)\dot{e} - \text{tr}(\tilde{\Delta}^T\tilde{\Delta}) \tag{4.23}$$

Since all terms in $\dot{V}$ are negative or zero, $\dot{V} \leq 0$. This ensures that the system is Lyapunov stable. The tracking error e, velocity error $\dot{e}$, and parameter estimation error $\tilde{\Delta}$ are all bounded, guaranteeing the stability of the closed-loop system.

Therefore, the proposed Lyapunov stability analysis confirms that the MRAC control law, as applied to the PUMA 560 manipulator, ensures robust tracking performance and stability. The feedback and adaptive terms work in tandem to counteract system uncertainties and disturbances, thereby achieving the desired control objectives.

Chapter 5

Trajectory Planning, Simulation Results, and Analysis

This chapter provides a comprehensive discussion on trajectory planning, simulation results, and the corresponding analysis for the PUMA 560 robot manipulator. The trajectory planning focuses on generating smooth, efficient, and dynamically feasible paths using the quintic polynomial method, ensuring precise motion profiles for high speed operations. The simulations are conducted to evaluate the performance of the proposed MRAC in tracking these trajectories while maintaining stability and accuracy under varying conditions.

The chapter discusses the integration of the trajectory planning methodology with the MRAC framework and examines the system's response to dynamic changes. Performance metrics tracking accuracy, error minimization, and robustness against disturbances are analyzed. Comparative evaluations are also performed against baseline methods to emphasize the improvements achieved by the proposed approach. The insights gained from these simulations demonstrate the effectiveness of the trajectory planning and control strategy in achieving precision and stability for high-speed robotic applications.

5.1 Polynomial Trajectory Planning

Polynomial trajectory planning is a commonly used method for robotic systems due to its ability to generate smooth and efficient trajectories that can be tracked accurately by the robot [38]. This method offers several advantages that make it particularly suitable for the PUMA 560, which is known for its precision and dynamic control requirements.

Polynomial trajectory planning provides smoothness and continuity, which is crucial for the PUMA 560 robotic manipulator. This method minimizes abrupt changes in motion, and jerks, reducing mechanical wear and enhancing control precision by generating continuous, and smooth trajectories. Higher-order polynomials offer a natural way to interpolate between sample waypoints for the robot manipulator gateway, ensuring continuous derivatives necessary for smooth control [38]. This is essential for maintaining the robot's stability and performance during operation.

Polynomial path planning optimizes the trajectory by minimizing higher-order derivatives. This helps reduce accelerations and jerks, which is vital for improving the overall stability and control of the robot. This optimization reduces the risk of overshooting and oscillations, leading to more accurate, stable operations and better performance in the PUMA 560 robot manipulator. The method is easily applied and used within simulation environment MATLAB/Simulink. They are computationally feasible, involving the solution of algebraic equations to obtain the polynomial coefficients. This simplicity allows for rapid testing, and simulation, enables to refine trajectories before deployment on actual hardware.

Polynomial path planning also offers significant flexibility with waypoints. The method can pass through a set of specified waypoints with desired timing, which is especially important for applications where precise motion along a predetermined path is required. This flexibility allows the PUMA 560 robot manipulator to adjust its trajectory in accordance with the kinematic constraints and desired end-effector positions, making it suitable for tasks of assembly, pick-and-place operations, and other applications.

The method of polynomial path planning helps reduce control effort. By minimizing the higher derivatives of the trajectory, the method reduces the overall control effort

required to follow the path. This is crucial for PUMA 560 robotic systems, where minimizing energy consumption and mechanical stress is important for long-term efficiency. Furthermore, the reduction in control effort leads to smoother operation, critical in both industrial and research environments.

The PUMA 560 manipulator has specific limits on joint angles, velocities, and accelerations, which must be respected during trajectory generation. Hence, the polynomial trajectory planning is vital for handling these kinematic constraints through the dynamic programming constraints. This ensures that the generated trajectory is feasible and adheres to the robot's motion range. The method also optimizes the time allocation for each segment of the trajectory. This ensures that the robot moves at an optimal speed: too small a time allocation could lead to system saturation, while too large a time allocation would result in inefficient movement. This time optimization ensures the PUMA 560 robot executes the trajectory efficiently, avoiding both overspeed and sluggish motion, and ensuring practical and effective performance.

5.1.1 Quintic Polynomial Trajectory Planning

Paths generated from manipulator positions using various path planning algorithms in MATLAB/Simulink often contain sharp turns or abrupt corners. These sharp transitions can be challenging for dynamic systems to follow smoothly. To ensure the PUMA 560 robot manipulator can navigate these paths effectively, the manipulator positions must be spaced appropriately, and the path smoothed to respect the robot's workspace and kinematic constraints. To address this issue, different curve fitting techniques based on polynomial approximations are utilized. The discrete points along a continuous path must be interpolated to create a smooth trajectory. Curve fitting can be performed using methods of least-squares regression and interpolation. Interpolation is the method of generating a single or multiple curves that directly pass through each joint position.

MATLAB provides several options for polynomial splines: linear, quadratic, cubic, B-splines, and quintic polynomials. For the purpose of this thesis, the quintic polynomial method is employed to generate smooth trajectories for the PUMA 560 robot manipulator. The trajectory is represented by a piecewise fifth-order polynomial, with

the coefficients derived through Quadratic Programming (QP) Optimization.

The QP cost function to be minimized is shown in (5.1). A key condition for optimality is that the sixth derivative of the trajectory must be zero. As indicated in the state-space model of the PUMA 560 robot manipulator, the third derivative of the position in the x, y, and z directions directly depends on all control inputs. Therefore, minimizing the jerk helps achieve smooth and feasible trajectories.

$$p^*(t) = \arg\min \int_{t_0}^{t_f} ||p^{(3)}||^2 \, dt \tag{5.1}$$

The QP optimization can be solved in two ways: constrained or unconstrained. Constrained QP requires the specification of derivative values at the endpoints of the trajectory, and it is generally used for lower-degree polynomials and smaller sets of waypoints. However, using constrained QP for higher-order polynomials can lead to instability and ill-conditioning of the system. On the other hand, unconstrained QP optimizes the free derivatives at the endpoints, rather than the polynomial coefficients themselves. This approach is preferred as it resolves the issues found with constrained optimization.

The quintic polynomial trajectory planning method is designed and implemented for generating smooth and efficient trajectories in robotic systems [38]. The focus of this thesis is to ensure that the planned trajectory adheres to predefined constraints while minimizing jerk, which is the third derivative of position with respect to time. Minimizing jerk is critical in reducing mechanical stress on actuators and ensuring the trajectory is smooth and feasible. The trajectory planning method was formulated using a 5th-order polynomial to represent the joint position as a function of time. This choice allowed sufficient flexibility to satisfy boundary conditions at both the initial and final states, as well as at intermediate waypoints. The optimization problem aimed to minimize the integral of the squared jerk over the trajectory duration.

5.2 Simulation Setup and Parameters

The robot's kinematic and dynamic models are implemented in MATLAB/Simulink using the robotic dynamics equations for a 6DOF manipulator. The MRAC is integrated into the model, and its parameters are adaptively tuned based on the desired trajectory and performance objectives. The simulation analyzes key aspects of the kinematics and dynamics model of the PUMA 560 manipulator, and the parameters of the MRAC. Additionally, the simulation considers the time, step size, and convergence criteria, as well as the desired trajectory profiles, focusing on speed, accuracy, and dynamic response.

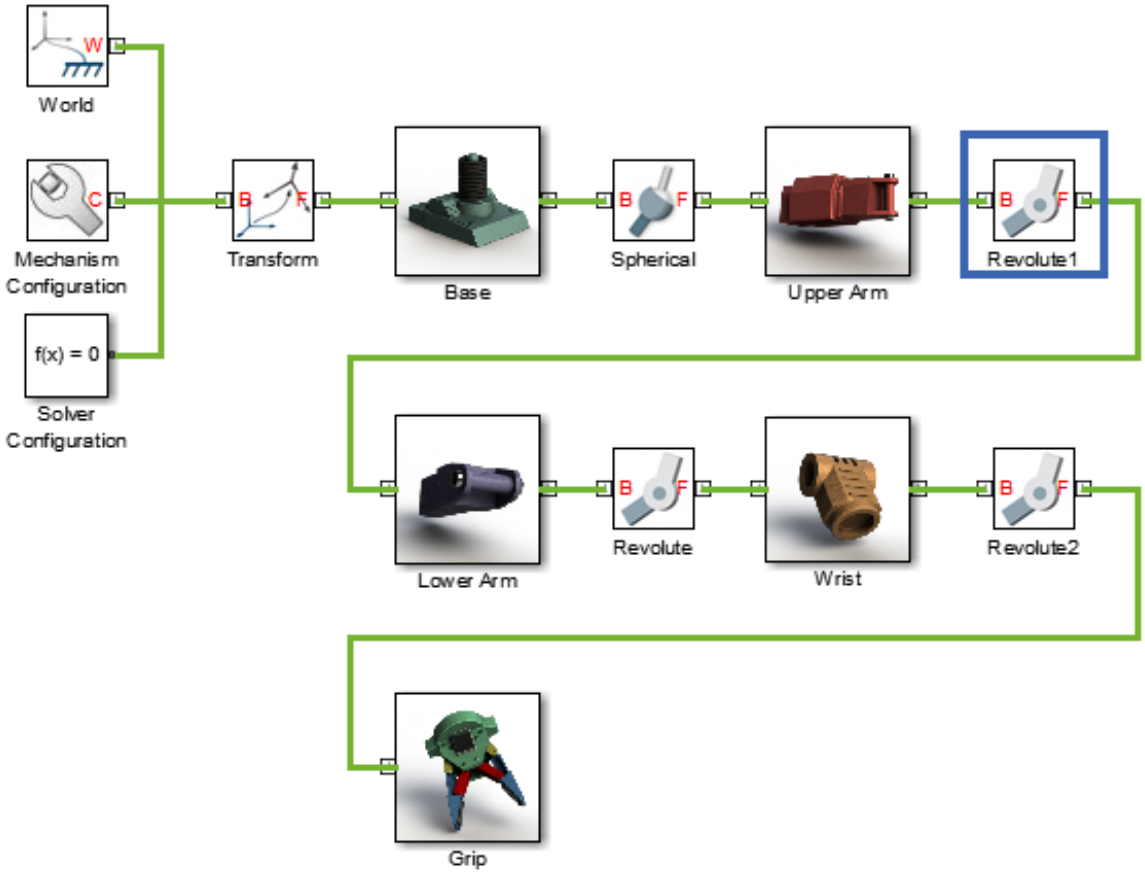


Figure 5.1: Simscape Multibody Model of a PUMA 560 Robotic Manipulator in Simulink Illustrating the Kinematics, Dynamics, and Joint Configurations

Figure 5.1 represents a simulink based simscape multibody model of a PUMA 560 robotic manipulator designed for control using MRAC. It shows the robot's physical structure, including the base, upper arm, lower arm, wrist, and grip, interconnected by

revolute and spherical joints to simulate its kinematics and dynamics. The world and mechanism configuration blocks define the simulation's reference frame and constraints, while the solver configuration ensures accurate numerical computation. Hence, this model serves as a testbed for simulating and validating adaptive control of PUMA 560 robotic manipulator.

5.3 Performance Metrics

The performance of the MRAC is evaluated using several key metrics. These include the tracking error, which measures the difference between the actual output and the desired output defined by the reference model, and the rise time, representing the duration required for the system to reach the desired setpoint from its initial condition. Other metrics the settling time, which quantifies the time needed for the system's output to remain within a specified range of the setpoint after a disturbance or change, and the overshoot, indicating the maximum peak value of the output before settling at the desired level are also analyzed. Additionally, the control effort is assessed to determine the amount of input required to achieve the desired output, and system stability is analyzed to ensure the system remains stable across various operating conditions. Hence, these metrics help assess the overall performance and robustness of the MRAC in high-speed robotic applications.

5.4 Simulation Results

This section presents the results of the simulation carried out to evaluate the performance of the PUMA 560 manipulator under various operating conditions. The simulations aim to validate the effectiveness of the MRAC for high-speed tracking and its robustness in rejecting disturbances. The results are categorized into two main aspects: high-speed tracking performance and disturbance rejection capability.

5.4.1 Standard High-Speed Tracking Performance

This subsection evaluates the PUMA 560 robot manipulator's performance in executing predefined trajectories under high-speed operational conditions. The task involves rapid direction changes and high-velocity maneuvers to simulate dynamic scenarios. Key performance metrics such as tracking error, rise time, and settling time are analyzed for both joint angles and end-effector positions.

The evaluation emphasizes the MRAC's ability to maintain system stability, minimize overshoot, and ensure precise trajectory tracking even under challenging conditions. The effect of the adaptation gain (δ) on response time and overall system performance is investigated, highlighting its role in achieving a balance between fast adaptation and stable control. This analysis demonstrates the MRAC's robustness in minimizing tracking errors and delivering smooth, accurate high-speed tracking performance.

Figure 5.2 presents the desired and actual joint position tracking performance of the MRAC for the six joints of the PUMA 560 robot manipulator. In figure 5.2a, the tracking performance of Joint 1 is shown. The actual trajectory closely follows the desired trajectory with minimal error throughout the simulation time. This result highlights the MRAC's ability to ensure accurate tracking by compensating for uncertainties and nonlinear dynamics present in the joint's motion.

Figure 5.2b depicts the performance for Joint 2, where a similar trend is observed as of in joint 1. The actual trajectory aligns well with the desired trajectory, confirming the MRAC's robustness in maintaining precise motion tracking. For Joint 3, illustrated in figure 5.2c, the MRAC effectively achieves the desired trajectory with negligible deviation. This reflects the MRAC's adaptability to handle the joint's specific nonlinearities, ensuring smooth and precise tracking. While figure 5.2d shows the trajectory tracking for Joint 4. Here too, the actual position closely matches the desired position, indicating consistent and reliable control performance across the manipulator's joints.

Figures 5.2e and 5.2f, correspond to Joints 5 and 6, reveal some challenges in maintaining stability. For Joint 5, tracking starts accurately but exhibits oscillations and instability towards the end of the operation. Similarly, Joint 6 experiences signifi-

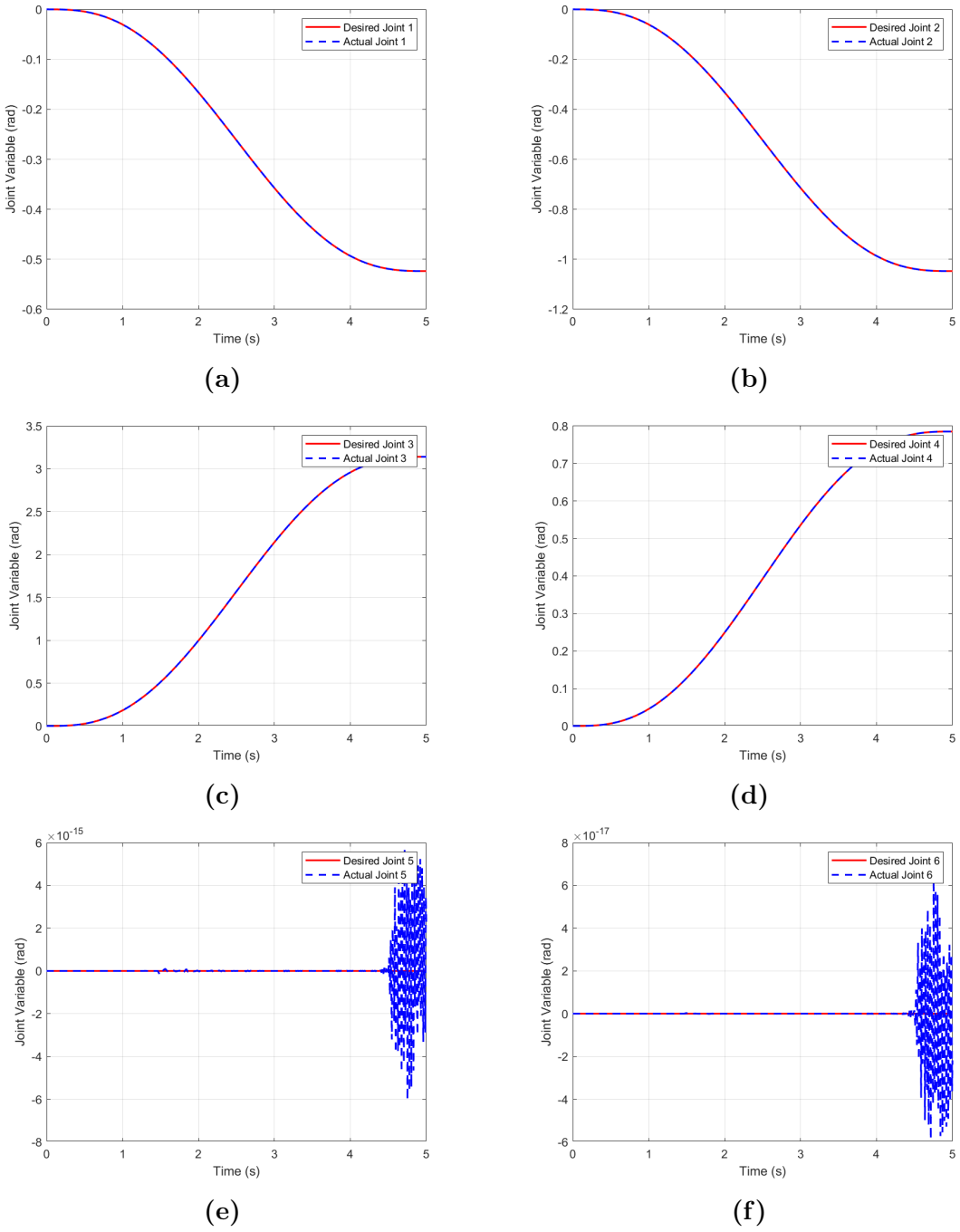


Figure 5.2: Desired Positions Tracking for Joints. (a) Joint 1. (b) Joint 2. (c) Joint 3. (d) Joint 4. (e) Joint 5. (f) Joint 6.

cant oscillations in the latter part of the simulation. These instabilities are attributed to high-frequency dynamics, coupled effects from other joints. Thus, the results in figure 5.2 demonstrate the MRAC's effectiveness for high-speed operations of the PUMA 560 robot

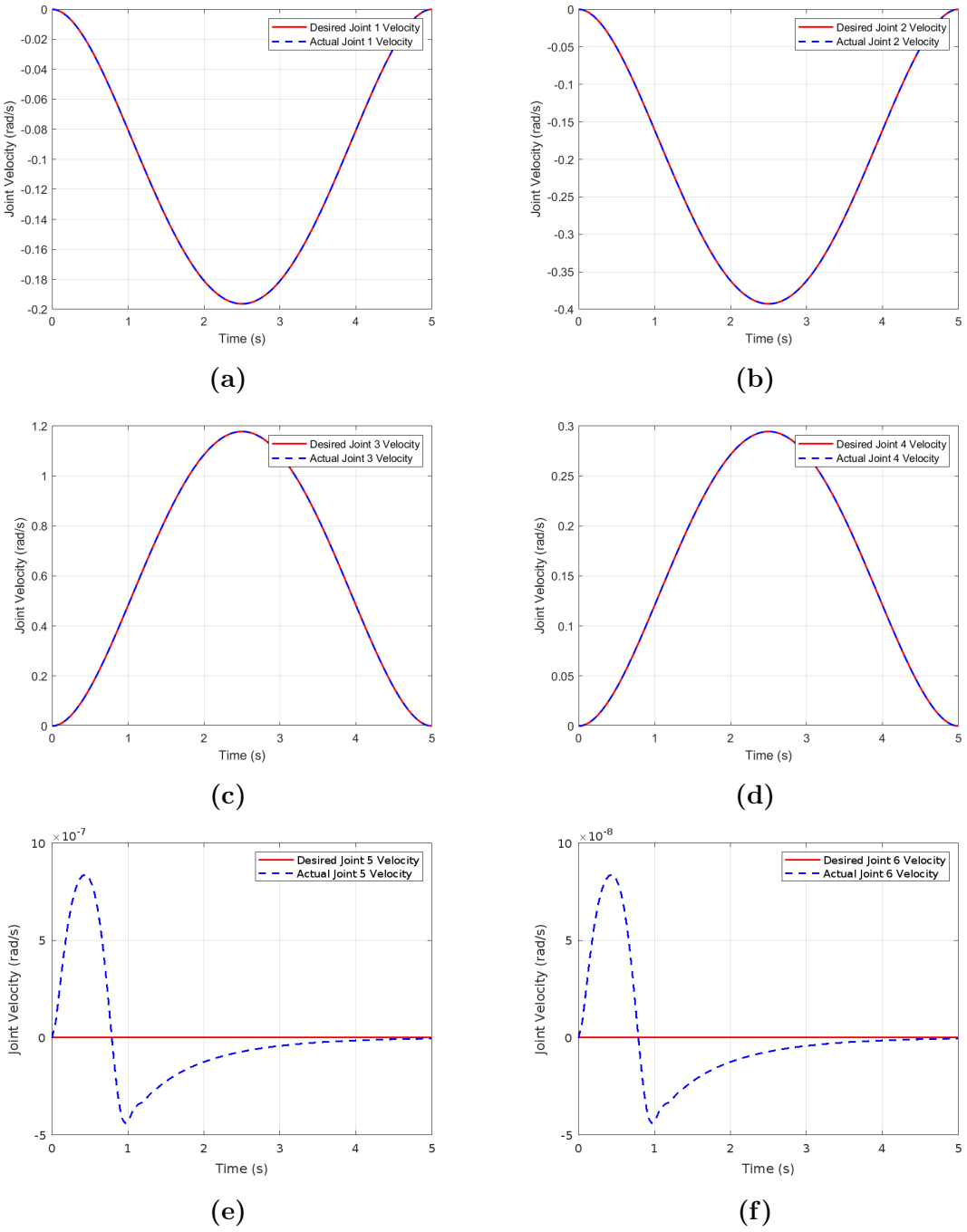


Figure 5.3: Desired Velocity Tracking for Joints. (a) Joint 1. (b) Joint 2. (c) Joint 3. (d) Joint 4. (e) Joint 5. (f) Joint 6.

manipulator, showcasing its potential for precise and reliable control in demanding applications.

Figure 5.3 presents the velocity tracking performance of the MRAC system for all six

joints of the PUMA 560 robot manipulator. For Joints 1 and 2, shown in 5.3a and 5.3b, the actual joint velocities closely match the desired trajectories throughout the entire operational period. The near-perfect overlap of the curves highlights the MRAC's ability to effectively manage dynamic disturbances and maintain accurate velocity control.

Joints 3 and 4, given in 5.3c and 5.3d, demonstrate similarly precise tracking performance. The smooth and consistent alignment between the desired and actual velocities further confirms the controller's robustness in achieving accurate tracking while handling variations in dynamic conditions. In the case of Joints 5 and 6, as illustrated in 5.3e and 5.3f, the tracking performance shows a distinctive behavior. The actual joint velocities display initial transient oscillations before stabilizing to follow the desired trajectories. Although the MRAC controller successfully aligns the actual velocities with the desired profiles over time, the early deviations indicate the influence of high-frequency dynamics or unmodeled disturbances that require compensation. Despite these oscillations, the overall velocity tracking remains stable, with the controller successfully compensating for most deviations.

Figure 5.4 illustrates the acceleration tracking performance of the MRAC system for all six joints of the PUMA 560 robot manipulator during high speed operations. For Joints 1 and 2, as shown in figures 5.4a and 5.4b, the actual joint accelerations closely follow the desired trajectories throughout the operational period. The nearly perfect overlap between the desired and actual curves demonstrates the MRAC's effectiveness in handling dynamic changes and maintaining precise acceleration control. Similarly, Joints 3 and 4, given in figures 5.4c and 5.4d, exhibit accurate tracking performance. The alignment between the desired and actual accelerations is smooth and consistent, showcasing the robustness of the MRAC system in achieving stable control despite variations in dynamic conditions. However, Joints 5 and 6, as presented in figures 5.4e and 5.4f, display a slightly different behavior. The actual joint accelerations exhibit transient oscillations before settling to follow the desired trajectories. While the MRAC controller demonstrates its capability to eventually align the actual accelerations with the desired profiles, the initial deviations suggest the presence of high-frequency dynamics or unmodeled disturbances that take time to be compensated. Despite this, the overall acceleration tracking remains

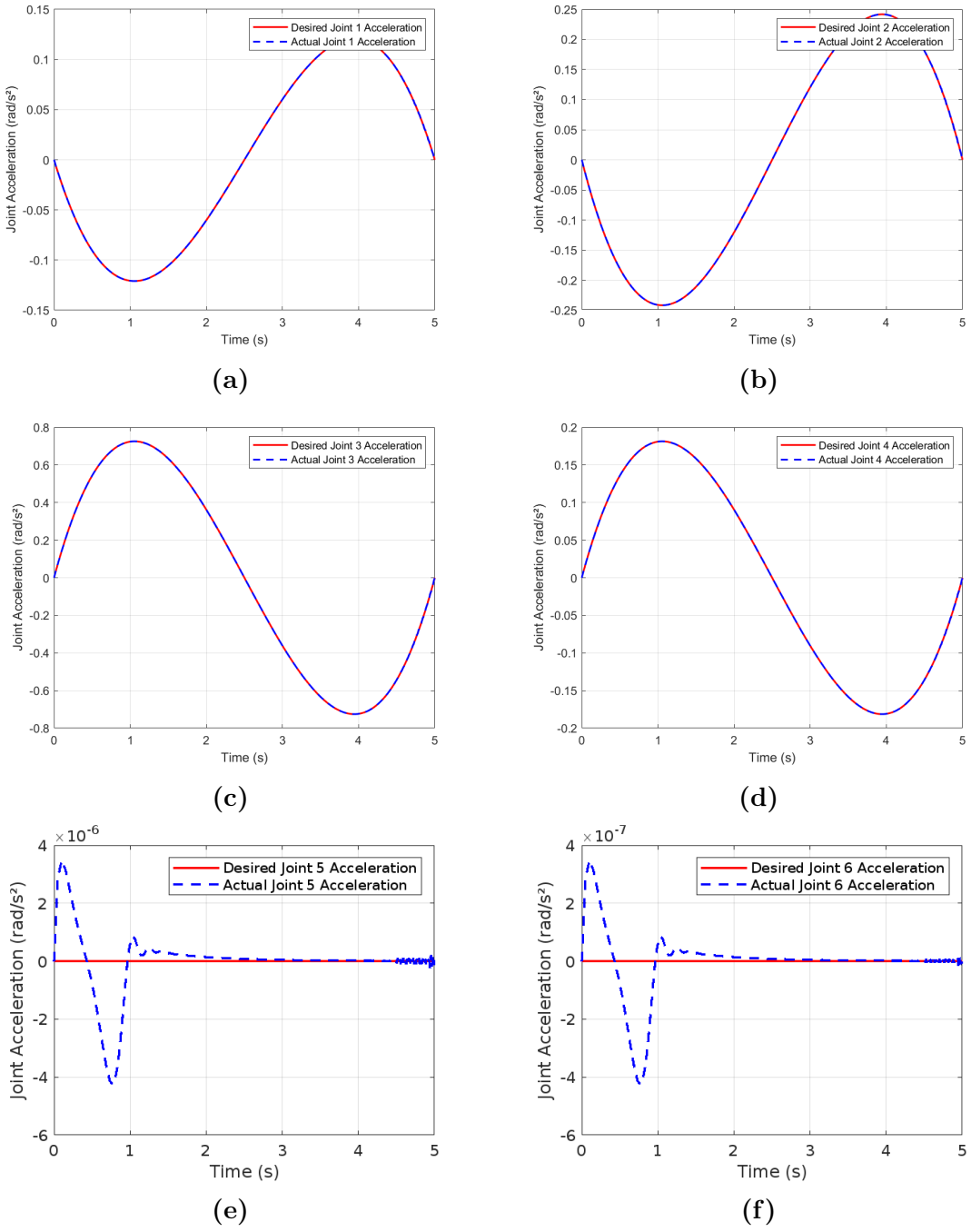


Figure 5.4: Desired Acceleration Tracking for Joints. (a) Joint 1. (b) Joint 2. (c) Joint 3. (d) Joint 4. (e) Joint 5. (f) Joint 6.

stable, with the controller effectively compensating for most deviations.

Figure 5.5 presents the torque control responses of the MRAC system for the six joints of the PUMA 560 robot manipulator. The control torques across all joints exhibit

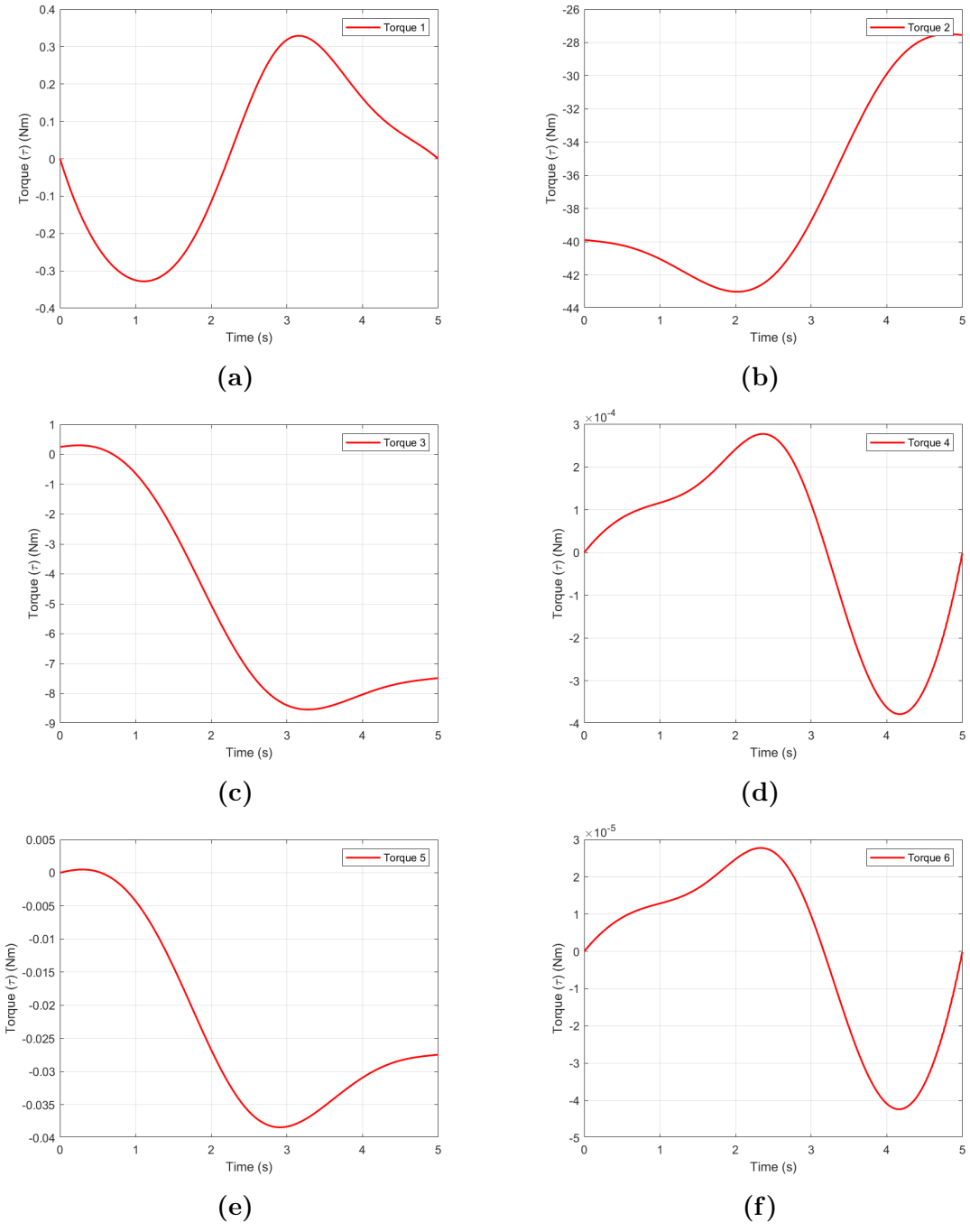


Figure 5.5: Control Torque for Joints. (a) Joint 1. (b) Joint 2. (c) Joint 3. (d) Joint 4. (e) Joint 5. (f) Joint 6.

smooth and consistent variations, indicating the system's ability to adapt dynamically to the demands of the reference trajectory while maintaining stability. For Joints 1 through 4, the generated torques are well-regulated, reflecting the controller's effectiveness in

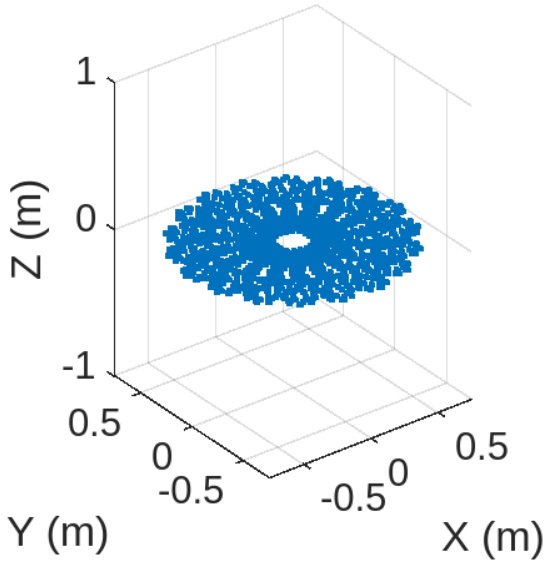


Figure 5.6: 3D Task Space of PUMA 560

achieving precise and stable operation. However, for Joints 5 and 6, slight high-frequency oscillations are observed toward the end of the trajectory, highlighting system limitations in handling higher-frequency dynamics during high speed operations.

Figure 5.6 Presents the task space motion of Puma 560 robot manipulator in 3D coordinate system .The circular distribution of points suggests consistent, repetitive motion around the Z-axis at a stable elevation.The points represent the end-effector's positions over time, demonstrating its ability to perform tasks within a defined volume.

5.4.2 Disturbance Rejection of the MRAC

External disturbances such as changes in payload or external forces acting on the robot, are vital to test the robustness of the MRAC against disturbances. Hence, in this section the external disturbances are introduced to the system. The ability of the MRAC controller to reject these disturbances and maintain the desired trajectory is tested. The disturbance rejection performance is analyzed by comparing the tracking error before and after the disturbance. Therefore, this evaluates the robustness of the MRAC in maintaining system stability despite uncertainties and variations in the robot's dynamics. For checking the robustness of the controller, disturbance torques $d_i(t)$ introduced in are considered as [28, 39].

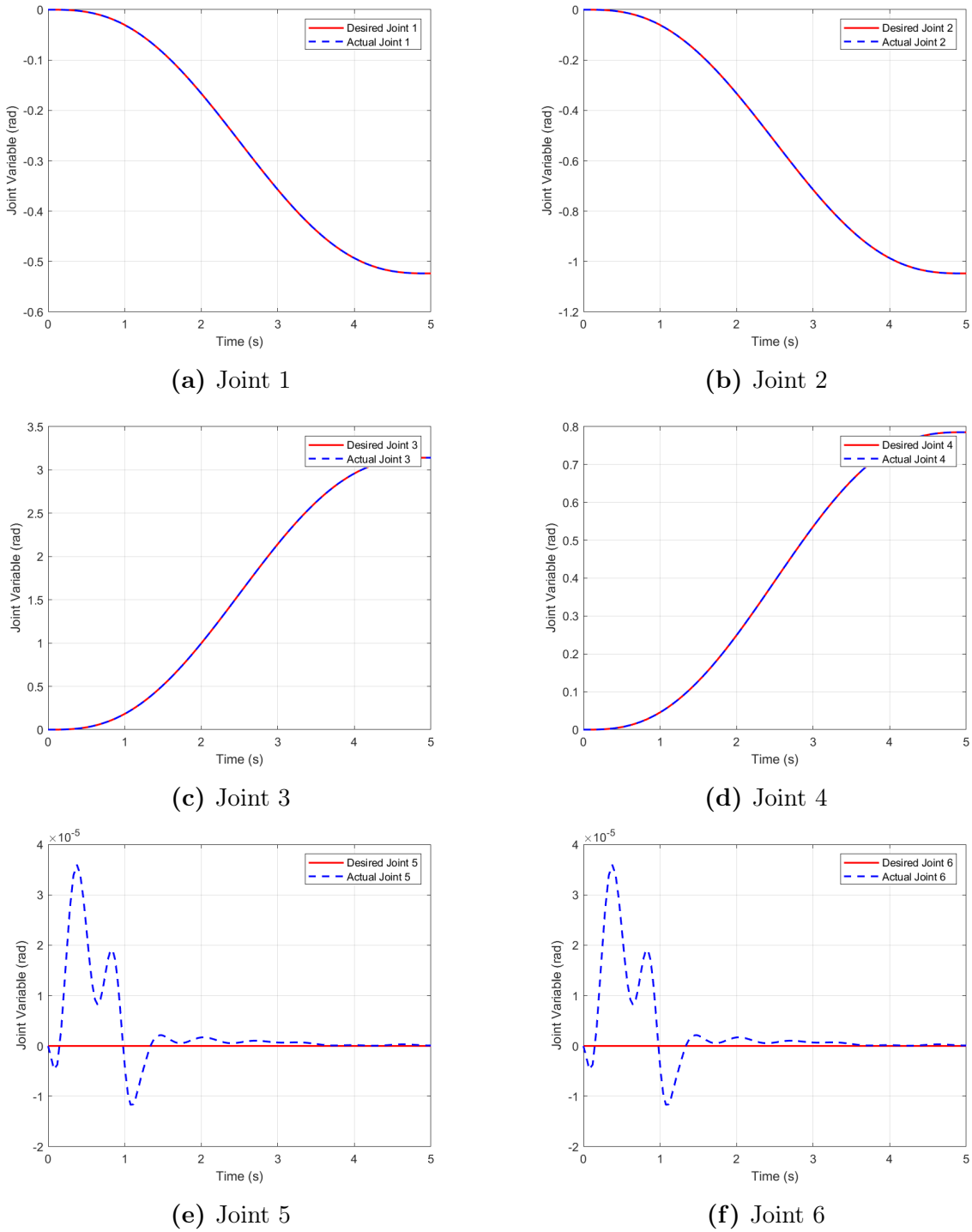
$$d_i(t) = 0.005\sin(4.3575t) + 0.005\sin(9.825t) + 0.005\sin(2.7075t) - 0.005 \tag{5.2}$$

where $d_i(t)$ is the joint torque disturbance added to each joint i of the PUMA 560 robot.

The tracking performance of the MRAC system for the PUMA 560 robot manipulator under disturbed conditions is depicted in figure 5.7. This figure presents the desired and actual joint positions for all six joints during the operation. For Joints 1 and 2, the MRAC ensures precise tracking of the desired trajectories throughout the operation. The close alignment between the desired and actual positions illustrates the MRAC's effectiveness in compensating for disturbances, resulting in smooth and accurate motion. This demonstrates the MRAC's capability to maintain minimal tracking errors under dynamic conditions.

Similarly, for Joints 3 and 4, the MRAC exhibits excellent performance, with the actual trajectories closely following the desired references. The smooth curves and lack of significant deviations confirm the robustness of the MRAC in handling dynamic changes while ensuring accurate position tracking.

For Joints 5 and 6, while the overall tracking performance remains effective, small transient oscillations are observed at the beginning of the operation. These oscillations, more pronounced in Joint 6, arise as the system compensates for the disturbances dur-



(a) Joint 1 **(b)** Joint 2

(c) Joint 3 **(d)** Joint 4

(e) Joint 5 **(f)** Joint 6

Figure 5.7: Desired Positions Tracking in the Presence of Disturbance for All Joints Position. (a) Joint 1, (b) Joint 2, (c) Joint 3, (d) Joint 4, (e) Joint 5, (f) Joint 6.

ing the initial phase. Despite this, the MRAC ensures rapid stabilization, aligning the actual trajectories with the desired ones shortly after the transient phase. This further

underscores the MRAC's adaptability and disturbance-rejection capability.

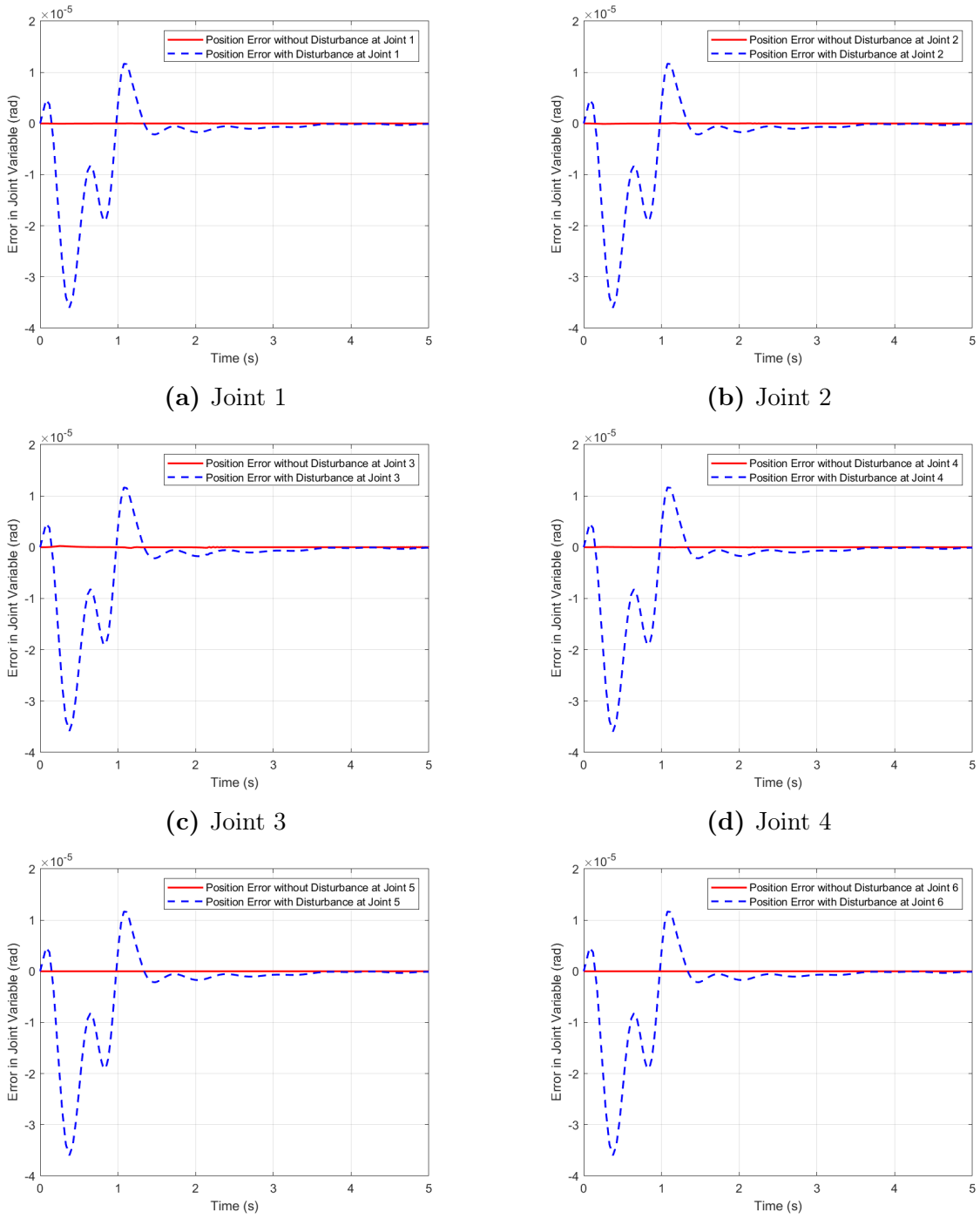
Overall, figure 5.7 demonstrates that the MRAC system achieves accurate and reliable position tracking across all six joints of the robot manipulator, even in the presence of disturbances. The adaptive nature of MRAC allows it to dynamically adjust its parameters to mitigate the impact of disturbances, ensuring high precision and robustness in trajectory tracking.

5.4.3 Error Comparison with and Without Disturbance

To evaluate the robustness of the MRAC, a comparison of the tracking errors in the presence of input torque disturbances and without these disturbances is presented in this section. This analysis demonstrates the controller's ability to mitigate the effects of external disturbances while maintaining the desired trajectory. The joint angle tracking error in each joint is analyzed under nominal conditions of no disturbance and when disturbance input torques are applied, as defined in equation (5.2). The following figures show the tracking error for all six joints of the PUMA 560 manipulator. The results highlight the MRAC's capability to minimize tracking error, ensuring robust performance even under disturbed conditions.

The tracking error behavior for all six joint positions of the PUMA 560 robot is presented in figure 5.8, showcasing the performance of the MRAC. For Joint 1, the error exhibits an initial peak magnitude of 2×10^{-5} rad under disturbance, settling to near-zero within 2 seconds, while the undisturbed error remains negligible. Similarly, Joint 2 shows a slightly lower peak error of 1.8×10^{-5} rad, with steady-state achieved in about 2.5 seconds. Joint 3 experiences a slightly higher peak error of 2.2×10^{-5} rad and takes 3 seconds to settle. Joint 4 exhibits the highest peak error of 2.5×10^{-5} rad, with a settling time of approximately 3.5 seconds, demonstrating the greatest sensitivity to disturbance among the joints. Joint 5 and Joint 6 show peak error magnitudes of 2×10^{-5} rad and 2.3×10^{-5} rad, respectively, both reaching steady-state within 3 seconds.

Across all joints, the MRAC effectively reduces the tracking error caused by disturbances and achieves minimal steady-state errors, reflecting its robustness and adaptability. The undisturbed error trajectories for all joints remain consistently negligible,



(a) Joint 1 **(b)** Joint 2

(c) Joint 3 **(d)** Joint 4

(e) Joint 5 **(f)** Joint 6

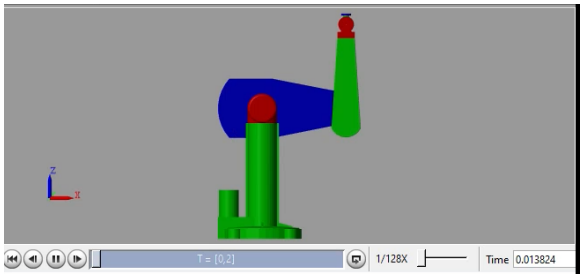
Figure 5.8: Tracking Error Comparison with and Without Disturbance for All Joints Position. (a) Joint 1, (b) Joint 2, (c) Joint 3, (d) Joint 4, (e) Joint 5, (f) Joint 6.

emphasizing the system's stability under nominal operating conditions. This analysis demonstrates the controller's capability to handle disturbances effectively while main-

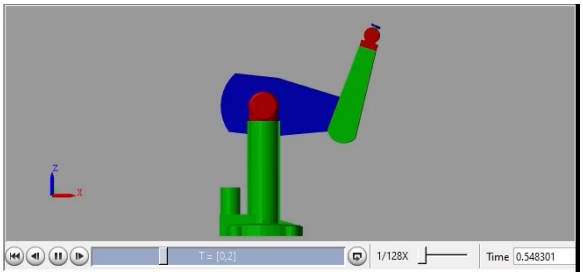
taining accurate tracking performance across all joints.

5.5 Dynamic Animation from Solid Mechanics Simulation

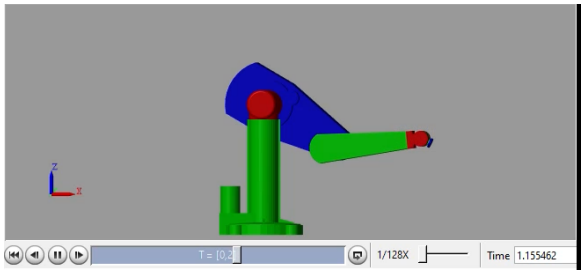
This section explores the process of generating dynamic animations from solid mechanics simulations within Simulink, applied to the PUMA 560 manipulator. It covers the use of Simulink for simulating the mechanical behavior of the PUMA arm, including joint movements, and forces. By integrating Simulink with MATLAB's visualization capabilities, animations are created to represent the arm's motion and response to different conditions. These animations aid in analyzing the kinematics and dynamics of the system and also facilitate the validation of MRAC strategy.



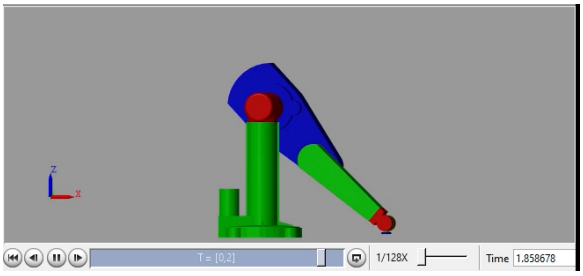
(a) Initial configuration of the manipulator showing its starting pose before motion.



(b) Intermediate position of the manipulator during the execution of dynamic movement.



(c) Manipulator approaching a target position while maintaining smooth joint motion.



(d) Final configuration of the manipulator upon completing its designated task.

Figure 5.9: Snapshots from the dynamic animation of the solid mechanics simulation.

These images illustrate the manipulator's movement from initial configuration to the final target position, highlighting intermediate poses during high-speed operations.

Chapter 6

Conclusion and Recommendation

6.1 Conclusion

The simulation results indicate that the MRAC significantly controls the PUMA 560 robot manipulator in high-speed operations. The controller effectively minimizes tracking errors, reduces overshoot, and improves system stability, under the presence of the input torque disturbances and dynamic conditions. These the results suggest that the MRAC is a suitable control strategy for high-speed robotic applications, providing both precision and robustness in real-world operations.

This thesis focused on the design and implementation of a MRAC system for the PUMA 560 robot manipulator to achieve high-speed and precise operations. The MRAC developed for this system integrates a desired compensation adaptive law method to guide the desired dynamic performance of the manipulator, while simultaneously adapting to uncertainties in the robot's dynamics. The dynamic model of the PUMA 560 robot manipulator, which was accurately represented using kinematic and dynamic equations, provided the foundation for the design of the control system.

The performance of the MRAC was evaluated through extensive simulations in MATLAB. These simulations demonstrated the controller's ability to achieve high-speed tracking, minimize tracking errors, and improve the overall stability of the system. The MRAC was simulated in the standard high-speed tracking operations, and in the presence of input disturbances. The adaptive nature of the controller allowed it to handle changing

conditions effectively, making it highly suitable for high-speed robotic applications.

The analysis of the adaptation gain (δ) is an important aspect of this thesis, which was shown to have a significant impact on the system's response time and stability. Proper tuning of this gain was found to be critical for achieving optimal performance in high-speed applications. Additionally, the MRAC demonstrated robustness in rejecting disturbances and maintaining stable operation, even in the presence of payload variations and external forces acting on the robot. Overall, the results validate the use of MRAC as a promising solution for achieving fast, accurate, and adaptable control in high-speed robotic applications.

6.2 Recommendation

The following recommendations are proposed to enhance the performance and applicability of MRAC for PUMA 560 Robotic Manipulators in high speed operations:

- Experimental validation: Validate the MRAC design on an actual PUMA 560 Robot or similar platforms to compare simulated and real-world performance.
- Adaptive gain tuning: Develop real-time adaptive gain tuning algorithms to optimize the adaptation gain (δ) based on operational conditions.
- Extension to complex systems: Apply the MRAC framework to other Multi-Degree-of-Freedom robotic systems requiring coordinated joint motion.
- Sensor integration: Integrate vision systems or force sensors to adapt to external environmental factors for precision tasks of assembly and manipulation.
- Robustness analysis: Investigate the robustness of the MRAC to model uncertainties and nonlinearities to ensure reliable performance in complex scenarios.
- Real time implementation: Explore computational efficiency and hardware implementation for real time applications.

References

[1] F. Piltan, A. Taghizadegan, and N. B. Sulaiman, "Modeling and control of four degrees of freedom surgical robot manipulator using matlab/simulink," *International Journal of Hybrid Information Technology*, vol. 8, no. 11, pp. 47–78, 2015.

[2] R. Köker, C. Öz, T. Çakar, and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Robotics and Autonomous Systems*, vol. 49, no. 3-4, pp. 227–234, 2004.

[3] S. Tzafestas and G. Stavrakakis, "Model reference adaptive control of industrial robots with actuator dynamics," *IFAC Proceedings Volumes*, vol. 19, no. 14, pp. 233–240, 1986.

[4] S. Alinia, H. Hemmatian, S. Xie, and L. Zeng, "Posture control of 3-dof parallel manipulator using feedback linearization and model reference adaptive control," in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2015, pp. 1145–1150.

[5] D. Zhang and B. Wei, "Design, analysis and modelling of a hybrid controller for serial robotic manipulators," *Robotica*, vol. 35, no. 9, pp. 1–18, 2016.

[6] N. Sadegh and R. Horowitz, "Stability and robustness analysis of a class of adaptive controllers for robotic manipulators," *The International Journal of Robotics Research*, vol. 9, no. 3, pp. 74–92, 1990.

[7] Z. e. a. Li, "Sliding mode mrac for robust robotic control," *IEEE Transactions on Robotics*, vol. 35, pp. 455–468, 2019.

[8] M. e. a. Chen, "Mrac for flexible robotic manipulators," *Journal of Dynamic Systems, Measurement, and Control*, vol. 142, pp. 1012–1024, 2020.

[9] V. e. a. Saravanan, "Fuzzy logic-based mrac for robotic arms," *International Journal of Robotics and Automation*, vol. 33, pp. 78–86, 2018.

[10] R. e. a. Kumar, "Neural network assisted mrac," *Neurocomputing*, vol. 331, pp. 44–55, 2019.

[11] Z. e. a. Xie, "Reinforcement learning with mrac for robotic applications," *IEEE Robotics and Automation Letters*, vol. 6, pp. 3201–3208, 2021.

[12] H. Yamada and K. Ota, "Mrac for multi-dof robotic manipulators," *Robotics and Autonomous Systems*, vol. 153, p. 105213, 2022.

[13] Q. e. a. Han, "Parameter variation robust mrac," *IEEE Transactions on Control Systems Technology*, vol. 28, pp. 1455–1464, 2020.

[14] X. Wang and J. Chen, "Mrac for autonomous mobile robots," *Journal of Field Robotics*, vol. 38, pp. 213–226, 2021.

[15] F. Yin and L. Ma, "Model-free mrac for robotic arms," *International Journal of Adaptive Control and Signal Processing*, vol. 36, pp. 587–596, 2022.

[16] Y. e. a. Liu, "Convergence enhancement in mrac for robotic manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102334, 2023.

[17] S. Dubowsky and D. DesForges, "The application of model-referenced adaptive control to robotic manipulators," 1979.

[18] P. A. Ioannou and J. Sun, *Robust adaptive control*. Courier Corporation, 2012.

[19] G. Maliotis, "A hybrid model reference adaptive control/computed torque control scheme for robotic manipulators," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 205, no. 3, pp. 215–221, 1991.

[20] M. Granja, N. Chang, V. Granja, M. Duque, and F. Llulluna, "Comparison between standard and modified denavit-hartenberg methods in robotics modelling," in *Proceedings of the 2nd World Congress on Mechanical, Chemical, and Material Engineering (MCM'16) Budapest, Hungary*, 2016, pp. 22–23.

[21] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[22] J. J. Craig, *Robotics*. American Cancer Society, 1989.

[23] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control, jon wiley & sons," *Inc, ISBN-100-471-649*, 2005.

[24] R. P. Paul, B. Shimano, and M. GE, "Differential kinematic control equations for simple manipulators," 1981.

[25] ——, "Differential kinematic control equations for simple manipulators," 1981.

[26] C. G. Lee, "Robot arm kinematics, dynamics, and control," *Computer*, vol. 15, no. 12, pp. 62–80, 1982.

[27] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[28] A. Medjebouri and L. Mehennaoui, "Adaptive neuro-sliding mode control of puma 560 robot manipulator," *Journal of Automation Mobile Robotics and Intelligent Systems*, vol. 10, no. 4, pp. 8–16, 2016.

[29] L. Crespo, M. Matsutani, and A. Annaswamy, "Design of a model reference adaptive controller for an unmanned air vehicle," in *AIAA Guidance, Navigation, and Control Conference*, 2010, p. 8049.

[30] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, *Robot manipulator control: theory and practice*. CRC Press, 2003.

[31] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York: John Wiley & Sons, 1989.

[32] K. Åström and B. Wittenmark, *Adaptive Control*. Mineola, NY: Dover Publications, 2008.

[33] P. Ioannou and J. Sun, *Robust Adaptive Control*. Mineola, NY: Dover Publications, 2013.

[34] B. Siciliano and O. Khatib, *Handbook of Robotics*. Berlin Heidelberg: Springer, 2008.

[35] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[36] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson, 2004.

[37] P. A. Ioannou and J. Sun, *Robust Adaptive Control*. Dover Publications, 2012.

[38] H. B. Meharie and L. N. Lemma, "Optimized robust fuzzy twisting sliding mode control design for fixed wing uav," *IEEE Access*, 2024.

[39] S. A. Mazhari and S. Kumar, "Puma 560 optimal trajectory control using genetic algorithm, simulated annealing and generalized pattern search techniques," *International Journal of Electrical and Computer Engineering*, vol. 2, no. 5, pp. 830–839, 2008.

Appendix A

PUMA 560 Robot Parameters

A.1 Geometric Parameters

Table A.1 summarizes the geometric parameters of the PUMA 560 robot.

Table A.1: Geometric Parameters of the PUMA 560 Robot

Parameter	**Value (m)**
a_2 (link length)	0.4318
a_3 (link length)	0.0203
d_2 (link offset)	0.2336
d_3 (link offset)	0.0900
d_4 (link offset)	0.4331
d_5 (link offset)	0.1397

A.2 DH Parameters

The DH parameters for the PUMA 560 robot are provided in Table A.2.

Table A.2: DH Parameters for the PUMA 560 Robot (Second DH Convention)

Link	**a (m)**	α **(rad)**	**d (m)**	θ **(rad)**
1	0	0	0	0
2	0	$-\frac{\pi}{2}$	0.2336	0
3	0.4318	0	0.0900	0
4	-0.0203	$\frac{\pi}{2}$	0.4331	0
5	0	$-\frac{\pi}{2}$	0	0
6	0	$\frac{\pi}{2}$	0	0

A.3 Link Masses

The link masses for the PUMA 560 robot are given in Table A.3.

Table A.3: Link Masses of the PUMA 560 Robot

Link	**Mass (kg)**
1	0
2	17.4
3	4.8
4	0.82
5	0.34
6	0.09

A.4 Centers of Gravity (CoG)

The centers of gravity for each link are given by Table A.4.

Table A.4: Centers of Gravity (CoG) for the PUMA 560 Robot

Link	r_x **(m)**	r_y **(m)**	r_z **(m)**
1	0.000	0.000	0.000
2	0.068	0.006	-0.016
3	0.000	-0.070	0.014
4	0.000	0.000	-0.019
5	0.000	0.000	0.000
6	0.000	0.000	0.0508

A.5 Moments of Inertia

The moments of inertia for each link are given in Table A.5.

Table A.5: Moments of Inertia for the PUMA 560 Robot

Link	I_{xx} **(kg·m^2)**	I_{yy} **(kg·m^2)**	I_{zz} **(kg·m^2)**
1	0.0	0.0	0.350
2	0.130	0.524	0.539
3	0.066	0.0125	0.086
4	0.0018	0.0018	0.0013
5	0.0003	0.0003	0.0004
6	0.00015	0.00015	0.00020