الجمهوريـــة الجزائريـــة الديمقراطيـــة الشعبيـــة

**République Algérienne Démocratique et Populaire**

وزارة التعليم العـــالي والبحث العلمـــي

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université MUSTAPHA Stambouli**

**Mascara**

**جامعة مصطفى اسطبولي**

**معسكر**

**Faculté des Sciences et de la Technologie**

**Département d'Électrotechnique**

**Laboratoire Signaux et Systèmes – Université Abdelhamid Ibn Badis - Mostaganem**

# THÈSE DE DOCTORAT

**Spécialité : Automatique**

**Option : Automatique et Informatique Industrielle**

**Intitulé**

# Utilisation des techniques du soft-computing pour la commande des systèmes industriels : application sur un robot industriel

**Présentée par : REZALI Baghdadi**

**Devant le jury :**

| | | | |
|---|---|---|---|
| **Président** | **EL Kebir Abdelkader** | **Pr** | **Univ. Mustapha stambouli-Mascara** |
| **Co-Directeur de thèse** | **Ahmed-Foitih Zoubir** | **Pr** | **Univ. Sciences et de la Technologie-Oran** |
| **Directeur de thèse** | **IBARI Benaoumeur** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |
| **Examinateur** | **BOUGUENNA Farouk Ibrahim** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |
| **Examinateur** | **BENZOUAOUI Ahmed** | **M.C.A** | **Univ. Hassiba Benbouali-Chlef** |
| **Examinateur** | **BOUREGUIG Kada** | **M.C.A** | **Univ. Ibn Khaldoun-Tiaret** |
| **Examinateur** | **LARBAOUI Ahmed** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |

**Soutenue le : 04 juin 2025**

الجمهوريـــة الجزائريـــة الديمقراطيـــة الشعبيـــة

**People's Democratic Republic of Algeria**

وزارة التعليم العـــالي والبحث العلمـــي

**Ministry of Higher Education and Scientific Research**

**University of Mustapha Stambouli**

**Mascara**

**جامعة مصطفى اسطبولي**

**معسكر**

**Faculty of Science and Technology**

**Department of Electrotechnical**

**Signals and Systems Laboratory – University of Abdelhamid Ibn Badis - Mostaganem**

# DOCTORAL THESIS

**Specialist: Automatic**

**Option : Automatic and Industrial Computing**

**Titled**

## Use of soft computing techniques for the control of industrial systems: application on an industrial robot

**Carried out by : REZALI Baghdadi**

**In the presence of the committee :**

| | | | |
|---|---|---|---|
| **President** | **EL Kebir Abdelkader** | **Pr** | **Univ. Mustapha stambouli-Mascara** |
| **Thesis Co-Supervisor** | **Ahmed-Foitih Zoubir** | **Pr** | **Univ. cience and Technology-Oran** |
| **Thesis Supervisor** | **IBARI Benaoumeur** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |
| **Examiner** | **BOUGUENNA Farouk Ibrahim** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |
| **Examiner** | **BENZOUAOUI Ahmed** | **M.C.A** | **Univ. Hassiba Benbouali-Chlef** |
| **Examiner** | **BOUREGUIG Kada** | **M.C.A** | **Univ. Ibn Khaldoun-Tiaret** |
| **Examiner** | **LARBAOUI Ahmed** | **M.C.A** | **Univ. Mustapha stambouli-Mascara** |

**Defended on : June 4, 2025**

# Use of soft computing techniques for the control of industrial systems: application on an industrial robot

THESIS
SUBMITED TO
UNIVERSITY OF MUSTAPHA STAMBOULI MASCARA
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

AUTHOR :
REZALI BAGHDADI

2025

*To my beloved parents, whose unwavering support and guidance have shaped my journey.*

*To my brothers and sisters, for their constant encouragement and companionship.*

*To my dear friends and neighbors, whose kindness and inspiration have been invaluable.*

*And to myselfa reflection of resilience, determination, and an unyielding belief in my dreams.*

*If the plan doesnt work, change the plan, but never the goal.*

— Unknown

**Acknowledgments**

First of all, I would like to thank my God (**Allah**), my Creator, for granting me the strength and patience to complete this work.
I would like to express my deepest gratitude to all those who have supported and guided me throughout the journey of my doctoral research.
First and foremost, I would like to thank my advisors, **Dr. IBARI Benaoumeur** and **Pr. Zoubir AHMED FOITIH** for their unwavering guidance, invaluable feedback, and continuous support. Their knowledge, patience, and encouragement have been crucial to the completion of this thesis.
I also would like to express my heartfelt appreciation to the members of the defense jury:

- **Prof. EL KEBIR Abdelkader**, President of the jury University of Mustapha Stambouli, Mascara
- **Dr. BOUGUENNA Farouk Ibrahim**, Examiner University of Mustapha Stambouli, Mascara
- **Dr. BENZOUAOUI Ahmed**, Examiner University of Hassiba Benbouali, Chlef
- **Dr. BOUREGUIG Kada**, Examiner University of Ibn Khaldoun, Tiaret
- **Dr. LARBAOUI Ahmed**, Examiner University of Mustapha Stambouli, Mascara

Their insightful comments and evaluations have significantly enriched this work.
I am also grateful to the **University of Mustapha Stambouli, Mascara**, for providing the academic environment and resources necessary for my research.
My sincere appreciation goes to the **Signals and Systems Laboratory**, **University of Abdelhamid Ibn Badis, Mostaganem**, where I had the privilege of collaborating with talented colleagues.
I am profoundly grateful to my family for their constant love, support, and understanding throughout this journey. To my parents, thank you for your unending belief in me. Your sacrifices and encouragement have been my greatest source of strength.
Lastly, I would like to thank my friends for their emotional support and for always being there for me, especially during the most challenging times.
To all those who have contributed to my personal and academic growth, I extend my heartfelt thanks.

**Abstract**

This thesis presents advancements in industrial robotics using soft computing techniques, focusing on kinematic and dynamic modeling, sensorless collision detection and optimal trajectory planning. Chapter 3 provides a comprehensive study of robot manipulators, covering fundamental principles of kinematic and dynamic modeling. A detailed case study on the Fanuc M-710iC/70 industrial robot examines its kinematic structure, dynamic model, and control aspects, offering practical insights into robotic motion and system behavior. Chapter 4 addresses the limitations of traditional model-based collision detection methods and introduces a novel sensorless approach using a fuzzy momentum observer. By dynamically adjusting observer parameters through fuzzy logic, this method enhances detection accuracy, improving both sensitivity and robustness. Extensive simulations validate its effectiveness in detecting collisions with high precision. Chapter 5 focuses on optimal trajectory planning for industrial robots to minimize energy consumption while ensuring smooth motion. A deep learning-based energy model, utilizing a Long Short-Term Memory (LSTM) network, accurately predicts energy consumption. Additionally, a Genetic Algorithm (GA) optimizes robot trajectories by considering execution time, jerk, and energy efficiency. The integration of deep learning and evolutionary optimization enables the generation of energy-efficient trajectories, enhancing industrial robot performance. Overall, this research contributes to improving industrial robots by enhancing modeling accuracy, safety, and energy efficiency, making them more adaptive and intelligent in real-world applications.

# Contents

# List of Figures

# List of Tables

# Notations

In this thesis, the notation used in each chapter is defined independently, without being constrained by definitions in other chapters. This means that the same symbol may have different meanings depending on the context in which it appears. Such an approach allows each chapter to introduce and use notations in a manner best suited to its specific topic, ensuring clarity and logical consistency within each section. Readers are advised to refer to the relevant chapters notation definitions for precise meanings.

# Chapter 1

# Introduction

Abstract

This chapter provides the motivation behind this thesis, emphasizing the critical role of robotics in modern industry and the challenges associated with their implementation. It highlights the potential advantages of leveraging soft computing techniques to address these challenges effectively. Section 1.2 outlines the specific aims and objectives of this research, followed by an overview of the thesis structure to guide the reader through the subsequent chapters. Finally, Section 1.3 presents the publications directly related to this work.

## 1.1 Motivation

For decades, technological applications have been used in the industrial sector in the form of manufacturing system automation to improve operations. Indeed, automation technologies represent a significant revolution in industrial manufacturing. This is why many manufacturing companies have fully automated their production processes. In an industrial perspective, systems automation are computer-controlled equipment and machinery to enhance productivity, quality and flexibility in production processes, thus reducing the need for human interaction. In view of that, Robotics lies in the category of automation systems. It is a distinct field of automated machines designed with human-like traits or capabilities. A key feature of industrial robotics is their mechanical arms, which are engineered to mimic the movement of human body limbs. Aaccording to the Robotics Industries Association (RIA), an industrial robot can be defined as “a multi-functional manipulator, can be reprogrammed designed to move through a sequence of motions to carry out useful tasks such as welding, assembly, material handling and packaging ”[1].

Nowadays, robots are sophisticated mechanical devices to perform different tasks with high precision and their movement are controlled by advanced computers. The demand from manufacturers for increased production has necessitated using robots to accomplish various tasks quickly [2], the integration of robotics into these industries also leads to improved safety and enhanced capabilities, driving innovation and economic growth. Unlike human workers, these

robots are not affected by physical or psychological limitations. They do not experience fear, fatigue, or lack of interest, which allows them to perform their tasks consistently and without interruption.

According to [3], robotic applications in the industrial sector are divided into three fundamental categories: material handling, object processing operations and inspection. In material handling, robots transfer objects from one location to another, which is recognized as pick-and-place operations. In this category, robots are also capable of sorting parts and packaging products. Object processing operations involve specific processes performed on objects with high precision to obtain the desired final result. These operations can be welding, painting, deburing and assembly and much more. In the manufacturing process, some parts require inspection to ensure product quality, making robots ideal candidates for these operations. These applications can include material inspection or the detection of manufacturing imperfections.

Although there have been significant advances in industrial robots in current-day, they still face numerous challenges. The primary interests related to industrial robots involve various aspects, including tracking control, trajectory planning, human-robot interaction and energy consumption. The tracking control system is the mastermind of the robot, as it is responsible for generating the appropriate commands for the actuators to achieve the desired movement. In industrial robots, the purpose of tracking control is to ensure that the robot's position, velocity, and other state variables follow a desired trajectory by appropriately generating the driving torques for each joint, Consequently, precise control of each joint is essential [4]. Industrial robots are characterized by flexibility, varying loads, and unknown disturbances [5]. Therefore, it is essential to develop a control strategy with high robustness and adaptability. Various robust control strategies have been employed, including proportionalintegralderivative (PID) control [6], [7], neural network control [8], fuzzy logic control [9], and sliding mode control (SMC) [10]. Despite these advancements, the development of control techniques remains a primary area of interest for researchers, as achieving higher robustness and adaptability in industrial robots continues to be a key challenge. The aspect of planning trajectories for industrial robots is just as important as the control aspect. Planning the trajectory of an industrial robot in an optimal way balancing time efficiency, trajectory precision and other performance metrics remains a critical challenge that continues to engage domain specialists. This problem involves not only determining the most efficient path for the robot to execute its tasks but also ensuring the trajectory adheres to constraints such as kinematics limits. The common evaluation criteria for trajectory planning of industrial robots include: (1) minimum execution time [11], [12] to enhance productivity, (2) minimum energy consumption [13]–[15] to reduce power usage and operational costs and (3) minimum jerk [16], [17] to ensure smooth motion while minimizing mechanical shocks and vibrations. Human-Robot Interaction (HRI) in industrial settings is a crucial field that focuses on ensuring robots operate safely in environments where human presence is necessary. Unlike traditional industrial robots, which are confined within safety cages, collaborative robots (cobots) are designed to physically interact with human workers in shared workspaces. Cobots combine the strengths of both humans and robots to enhance

productivity. However, without robust safety measures for physical interaction, robots would remain restricted to enclosed environments [18]. Last but not least, the energy consumption of industrial robots is also a significant challenge, especially with the increasing demand for energy resources worldwide. Therefore, finding solutions to reduce energy consumption is of great interest to the industrial sector.

Industrial robotic systems are inherently complex, requiring sophisticated solutions to address challenges such as precise motion control, adaptive decision-making, and real-time response in dynamic environments. Solving these problems demands the integration of advanced technologies capable of handling uncertainty, nonlinearity, and variability in industrial processes. Several studies [19]–[22] highlight the significant role of soft computing techniques in developing intelligent models to tackle these challenges across various domains. Soft computing is a branch of artificial intelligence that encompasses a range of intelligent methodologies, including fuzzy logic, which enables reasoning under uncertainty; neural computing, which facilitates learning from data and pattern recognition; and evolutionary computing, which provides optimization capabilities through bio-inspired algorithms. These techniques collectively enhance the adaptability, robustness, and efficiency of industrial robots, making them more capable of operating autonomously in unstructured and dynamic environments. As a result, these techniques offer significant advantages in industrial robotics, where robots must operate in uncertain conditions, adapt to changing tasks, and optimize their performance in real time. Soft computing can be applied to various challenges, such as trajectory planning, force control, sensor fusion, fault detection, and energy efficiency optimization. By incorporating these intelligent methodologies, industrial robots can achieve greater autonomy, flexibility, and robustness, ultimately enhancing their overall efficiency and reliability in manufacturing and automation processes. Researchers have extensively explored various soft computing techniques in robotics, leveraging their ability to handle uncertainty, adapt to dynamic environments, and optimize performance. These techniques include, but are not limited to:

- Fuzzy logic [23]–[25], which enables reasoning under uncertainty by modeling imprecise information and making flexible, human-like decisions. It has been widely applied in robotic control, trajectory planning and collision avoidance.
- Neural networks [26]–[29], which provide learning capabilities by recognizing patterns from data, making them effective for tasks such as robot perception, system identification, and adaptive control.
- Machine learning [30], [31], which allows robots to improve their performance through experience, enabling applications in predictive maintenance, autonomous navigation, and real-time decision-making.
- Evolutionary algorithms [32]–[36], which employ bio-inspired optimization strategies to solve complex problems in robotic path planning, parameter tuning, and multi-objective optimization.

By integrating these soft computing techniques, industrial robots can achieve greater adaptability, robustness, and efficiency, enhancing their ability to perform complex tasks in unstructured and dynamic environments.

## 1.2 Aims and Objectives

The general objective of using soft computing in industrial robots is to enhance their adaptability, precision, and decision-making capabilities in dynamic and uncertain environments. Soft computing techniques, such as fuzzy logic, neural networks, and evolutionary algorithms, enable robots to handle imprecise data, learn from experience, and optimize complex tasks. This approach improves the efficiency and flexibility of industrial operations, making robots more intelligent and responsive to changing conditions.
In order to apply soft computing techniques to industrial robots in this work, a comprehensive mathematical and simulation model of the Fanuc M-710iC/70 industrial robot is designed. This robot is widely used in various manufacturing applications due to its high precision, flexibility, and payload capacity, making it a suitable platform for investigating advanced control strategies.
The first contribution involves constructing the kinematics, dynamics, and control models of the robot:

- Kinematics Model: Both forward and inverse kinematics are formulated, establishing the mathematical relationships between the robots joint variables and its end-effector position and orientation. This allows precise trajectory planning and motion control.
- Dynamic Model: A detailed dynamic model of the robot is built using the Simscape/Matlab environment. This model captures the effects of robot link masses, joint torques, inertia, and external forces, providing a realistic representation of the robots motion.
- Control Framework: The developed dynamic model serves as a foundation for testing and implementing various soft computing techniques, for optimizing robot performance in trajectory tracking .

Construction a high-fidelity simulation model, this work enables the application and evaluation of soft computing strategies in an industrial robot, enhancing its performance.
The second contribution of this thesis addresses the safety challenges in physical human/environment-robot interactions, which are critical for ensuring safe and efficient collaboration between robots and their surroundings. Collision detection methods are generally classified into two categories: Model-based methods and Model-independent methods. The later rely on external sensors to detect collisions. While effective, these methods increase system costs and complicate the manufacturing of robotic manipulators due to additional hardware requirements. To overcome these limitations, our contribution focuses on a sensorless approach, adopting a model-based method for collision detection. However, model-based approaches face a fundamental challengethere is

an unavoidable trade-off between collision sensitivity and the reduction of the peaking value, which affects detection accuracy and robustness. To address this issue, we propose an improved momentum observer based on fuzzy logic for collision detection and force estimation in industrial robots. This contribution includes:

- Designing a *generalized momentum observer* derived from the robots dynamic model to estimate external forces.
- Constructing a *fuzzy logic system* that intelligently adjusts observer parameters in real time, enhancing adaptability.
- Integrating the fuzzy system with the observer, enabling an intelligent tuning mechanism that improves detection sensitivity while minimizing peak disturbances.

Through combination momentum-based estimation with fuzzy logic adaptability, this approach enhances collision detection effectiveness, achieving a better balance between sensitivity and peak reduction, making it a promising solution for safer human-robot interaction in industrial settings.

The third contribution of this work focuses on the optimal trajectory planning of industrial robots by leveraging deep learning techniques and evolutionary algorithms. Given the continuous rise in energy costs, optimizing energy consumption has become a critical concern in modern industry. This contribution introduces an efficient and intelligent approach to trajectory planning, optimizing execution time, jerk, and energy consumption, while adhering to the robots kinematic constraints. The key highlights of this contribution are as follows:

- A 5th-order B-spline is employed alongside a multi-objective optimization technique to construct smooth and continuous trajectories up to jerk. This ensures precise and efficient robot motion while reducing vibrations and mechanical stress.
- Since an explicit mathematical model relating trajectory parameters to energy consumption is not readily available, a predictive model is developed using a Long Short-Term Memory (LSTM) neural network. This model learns from historical data to estimate the robots energy consumption profile and is incorporated as an objective function in the optimization process. Additionally, it enables energy prediction before real-time execution, allowing for pre-optimized motion planning.
- The Non-Dominated Sorting Genetic Algorithm II (NSGA-II) is employed to optimize the robot trajectory by simultaneously minimizing execution time, jerk, and energy consumption. This ensures an optimal trade-off among motion efficiency, smoothness, and energy savings.

Integration deep learning for predictive modeling and evolutionary algorithms for optimization, this approach significantly enhances the efficiency, precision, and sustainability of industrial robotic motion planning, making it a practical solution for energy-aware automation.

## 1.3 Thesis Outline

This thesis is structured into five chapters as follows:

**Chapter 1** serves as the introduction, beginning with the background and motivations behind this research. It then outlines the key challenges faced in industrial robotics and concludes by presenting the main contributions of this study.

**Chapter 2** presents a comprehensive state-of-the-art review, focusing on the application of soft computing techniques in industrial robotics. It explores key methodologies, including fuzzy logic, artificial neural networks and evolutionary algorithms, highlighting their roles in various aspects of robotics, such as trajectory tracking and control, optimal trajectory planning, energy efficiency optimization, and safety enhancement. This chapter provides a detailed analysis of how these intelligent techniques contribute to improving the performance, adaptability and reliability of industrial robots.

**Chapter 3** covers the fundamental principles of robot manipulators, including kinematic and dynamic modeling. It provides a detailed analysis of the mathematical formulations governing robot motion and dynamics. Additionally, this chapter includes an in-depth case study on the Fanuc M-710iC/70 industrial robot, examining its kinematic structure, dynamic model and control aspects.

**Chapter 4** addresses the limitations of traditional model-based collision detection methods for industrial robots and proposes a novel sensorless approach based on a fuzzy momentum observer. This approach enhances collision detection accuracy by intelligently adjusting observer parameters using fuzzy logic, improving both sensitivity and robustness. The effectiveness of the proposed method is demonstrated through comprehensive simulation results.

**Chapter 5** addresses the problem of optimal trajectory planning for industrial robots, with a particular focus on minimizing energy consumption. It presents an effective approach for generating optimal trajectories by considering execution time, jerk, and energy consumption as key optimization criteria. The proposed method integrates deep learning for energy modeling, utilizing a Long Short-Term Memory (LSTM) network to accurately predict energy consumption. Additionally, a Genetic Algorithm (GA) is employed as an optimization technique to refine the robots trajectory based on the three aforementioned criteria.

**Chapter 6** concludes the thesis by summarizing the key findings and contributions of this work. Additionally, it discusses potential future research directions, highlighting areas for further exploration and improvement in the field of industrial robotics and soft computing techniques.

## 1.4 Author's Publications

- **Rezali, B**., Ibari, B., Hebali, M., Berka, M., Bennaoum, M., Bouzgou, K., ... & Benchikh, L. (2025). Optimal trajectory planning for industrial robots: Minimizing time, jerk, and energy consumption using LSTM for energy profile modeling. Journal of Vibration and Control, 10775463251333481.

- B. Ibari, M. Hebali, **B. Rezali**, and M. Bennaoum, Collision detection and external force estimation for robot manipulators using a composite momentum observer, *AIMS Electronics and Electrical Engineering*, vol. 8, no. 2, pp. 237254, 2024.

- B. Ibari, M. Hebali, **B. Rezali**, M. Bennaoum, K. Boureguig, and K. Bouzgou, Enhanced trajectory tracking of robotic manipulators using velocity observer-integrated computed torque control, Studies in Engineering and Exact Sciences, vol. 5, no. 2, e11119e11119, 2024.

- **B. Rezali**, B. Ibari, M. Hebali, M. Berka, M. Bennaoum, and H. A. Azzedine, Sensorless robot collision detection based on fuzzy momentum observer, *Transactions of the Institute of Measurement and Control*, p. 01 423 312 241 262 538, 2024.

- **B. Rezali**, B. Ibari, M. Hebali, and H. A. Azzeddine, An optimal fractional order pi$\lambda$d$\mu$ for robotic manipulators control under constrained torque., *Journal of Engineering Science & Technology Review*, vol. 16, no. 5, 2023.

# Chapter 2

# State of the Art

Abstract

After a brief introduction to soft computing techniques, the fundamentals of the fuzzy logic approach are presented, along with a review of the relevant literature discussing its applications in industrial robotics (IRs). Following this, Section 2.3 provides into artificial neural networks, exploring their principles and recent advancements in IRs. Finally, the chapter concludes by highlighting evolutionary algorithms, with a focus on their methodologies and practical applications within the field of IRs.

## 2.1 Introduction

In recent years, the popularity of soft computing techniques has increased significantly for solving engineering problems. This has attracted the attention of researchers, particularly in addressing challenges in robotics [37]. Given the multidisciplinary nature of research in robotic systems, there is an urgent demand for the development of subfields such as kinematics, control, path planning, humanrobot interaction (HRI) and industrial robot energy consumption. As a result, current research trends focus on implementing soft computing in robotics and developing intelligent systems to manage rule bases or knowledge bases, such as fuzzy logic systems, artificial neural networks, swarm intelligence and evolutionary algorithms. They are suitable for systems with highly complex dynamics that lack a precise mathematical model, as well as for robot navigation where knowledge of the environment is inherently imprecise, unpredictable, and incomplete. Another advantage of soft computing techniques is that they can reduce the dependency on physical devices such as sensors in robotics by enabling robots to make more intelligent decisions based on incomplete, imprecise or noisy data.
This chapter covers the current state of the art in various soft computing techniques applied to robotics, including kinematics, control, humanrobot interaction, trajectory planning and collision avoidance and more.

## 2.2 Fuzzy Logic

With the development of electronic technologies and the increase in computational capabilities, there has been significant growth in the development of soft computing techniques across various applications. Relying on these techniques provides an excellent alternative tool for modeling and controlling systems. In particular, the fuzzy logic approach is a common method in intelligent systems due to its simplicity, offering efficient solutions and attracting the interest of researchers in robotic applications.
Zadeh is considered the first to introduce the concept of fuzzy logic [38], presenting it as a method that mimics human brain reasoning and thinking to solve ambiguous problems.Fuzzy logic is a form of multi-valued logic that deals with reasoning that is approximate rather than fixed or exact. Unlike classical binary logic, which operates with clear-cut, discrete values such as "true/false"or "high/low", fuzzy logic allows for a continuum of values between these extremes. It captures the nuances of human reasoning by expressing values like "fast", "very fast", "slow"and "very slow"rather than simply "fast"or "slow"[39].
The main concept of fuzzy logic involves establishing the relationship between the antecedent (premise) and the consequent (conclusion) through IF-THEN rules. The antecedent refers to the current situation or condition being evaluated, while the consequent outlines the action or outcome that should follow in response. The basic structure of a fuzzy logic system is composed of four primary components, as illustrated in Figure 2.1.

- *Fuzzy knowledge base*: represents the central part of a fuzzy logic system (FLS). It consists of knowledge data provided by expert users in the form of if-then rules. When the rules are accurate, the system's control is optimized, making it an essential component of the FLS. This knowledge base is typically constructed from expert insights and practical experiences.

- *Fuzzification interface*: in this module, real-world input values are transformed into fuzzy sets. This process allows continuous or precise input data to be interpreted in a way that can be compared with the fuzzy rules in the knowledge base. By converting crisp inputs into degrees of membership within fuzzy sets, the system can handle uncertainty and approximate reasoning, enabling more flexible decision-making.

- *Inference engine*: the inference engine evaluates fuzzy inputs by applying the rules stored in the knowledge base. It processes the inputs through a series of "if-then "rules to derive conclusions or make decisions. By combining and interpreting the results of these rules, the inference engine generates fuzzy outputs, which represent the system's response.

- *Defuzzification interface*: this component converts the fuzzy outputs generated by the inference engine into a precise, quantifiable result in crisp logic. It translates the fuzzy sets into a single, actionable value using one of several defuzzification methods, such as the centroid method or mean of maximum.

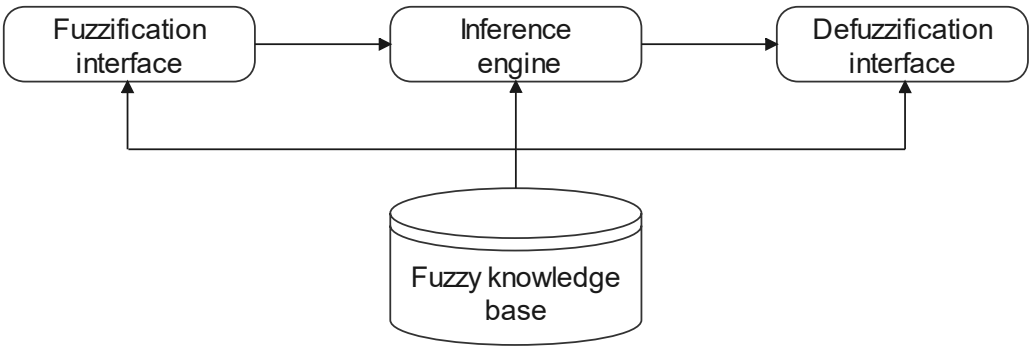



Figure 2.1: The basic structure of a fuzzy logic

Fuzzy logic has been applied across various domains in robotics, including (such as) control, path planning and collision avoided. In [40], the authors presented the application of fuzzy logic control in robotic manipulators. In [41], A fuzzy PD controller is used for trajectory tracking of the Rhino robot, and its performance is compared with that of a conventional PD controller. The fuzzy controller demonstrates the ability to suppress overshoot during the initial phase of robot trajectory tracking and performs better than the basic PD controller under all tested scenario. In [42], the author proposes a fuzzy logic technique to adaptively tune the gains of a sliding mode controller. The proposed controller is applied for position control of a 2-DOF polar manipulator, achieving significant improvements in terms of reduced reaching time and minimized chattering. Anupam et al. in [43] used a Type-2 fuzzy logic controller, an extension of Type-1 fuzzy logic, to control a 2-DOF robot manipulator with a payload. Type-2 fuzzy controllers provide an additional degree of freedom in their membership functions, making them particularly well-suited for handling complex and highly uncertain systems, such as robot manipulators. The fuzzy logic technique was integrated into hybrid force/position control of a robot manipulator to enhance its performance [44], a fuzzy-based computed torque control (CTC) method is employed to regulate the robot's end-effector position, ensuring precise movement and trajectory tracking. Simultaneously, a fuzzy proportional-integral (PI) controller is used for force control, enabling the manipulator to adapt to varying interaction forces with its environment. In reference [45], a fault-tolerant control system is developed using fuzzy logic to enhance the tracking performance of robotic manipulators. This approach does not require complete prior knowledge of the robot's dynamic model.

Another area in robotics where fuzzy logic has been widely applied is trajectory planning. Hentout et al. [46] conducted a comprehensive review of fuzzy logic technique for the path planning of robotic manipulators. Path planning of a robot manipulator remains a significant challenge in the field of robotics. Path planning involves identifying a series of configurations for the robot that allow it to move from an initial configuration to a final configuration while considering the mechanical constraints of the robot and the possible presence of obstacles in the robot's workspace. Lian [47] noted that it is challenging to establish an exact mathematical model of a robot to generate an effective path. In [48], authors proposed an offline path planning approach using fuzzy inference for robotic manipulators in a static environment. Their method leverages the Transition-based Rapidly-exploring Random Tree (T-RRT) algorithm,

which assigns a cost to each configuration. The cost function is defined by the distance to configurations that result in collisions. Fuzzy function approximation is used to evaluate this cost, providing an effective way to compute the cost throughout the configuration space. In order to achieve obstacle avoidance, Beheshti et al. [49] presented an adaptive fuzzy logic approach to solve the inverse kinematics problem for redundant robots, considering mechanical joint limits. Collision detection between humans and robots based on fuzzy identification was investigated by [50], in this approach, the robot is trained to recognize expected collisions, and the magnitude of the resulting forces is identified using a fuzzy logic technique. This method enables the robot to detect and respond to collisions more effectively, improving safety during human-robot interaction.

On the other hand, fuzzy logic has been used as an effective technique for addressing kinematic problems in robotics. In particular, the study presented in [51] demonstrates the application of fuzzy logic to solve the inverse kinematics problem for a 7-DoF redundant-serial robot. The paper [52] explores the use of Adaptive Neuro-Fuzzy Inference Systems (ANFIS) to address the inverse kinematics problem for multiple robotic arms. The work by Mary et al. [53] proposed an alternative method for solving the inverse kinematics of a 3-joint robotic manipulator. This method is based on closed-loop theory using fuzzy PD control. The proposed approach aims to obtain the IK model by minimizing the difference between the desired Cartesian position of the end-effector and its actual position, thereby improving the accuracy of the IK solution. In [54], fuzzy logic is merged with the ant colony algorithm to solve the inverse kinematics of a robot arm. Shihabudheen and Pillai [55] proposed an Extreme Learning Machine (ELM) model that integrates knowledge from fuzzy systems to predict the inverse kinematics solutions of robotic arms, this approach offers an efficient method for addressing the complexities of inverse kinematics in robotic systems.

## 2.3 Artificial Neural Networks

Nowadays, Artificial Neural Networks (ANNs) are considered a major researche trend in the scientific community. This area of research has had a profound impact across multiple fields, including robotics. A substantial number of publications have explored the application of artificial neural networks in robotics. This section provides a concise overview of artificial neural networks and highlights significant research contributions that apply ANNs approaches to key areas within robotics. These areas include kinematics, dynamics, trajectory planning, sensing and control, where neural networks have been utilized to address complex challenges and enhance robotic performance. By exploring these foundational studies, we gain insights into the growing impact of neural networks in advancing robotics technology.

### 2.3.1 An Overview of Artificial Neural Networks

#### What is a ANNs ?

A artificial neural networks is a modeling technique that uses data to make decisions in a way that mimics the human brain. It works by simulating the behavior of biological neurons, which collaborate to recognize patterns, evaluate options and reach conclusions.

#### How do ANNs work?

Similar to the human brain, which consists of a vast network of biological neurons, an artificial neural network is made up of a large network of interconnected nodes, as shown in Figure 2.2. In Figure 2.2, the group of square nodes represents the input layer, which transmits

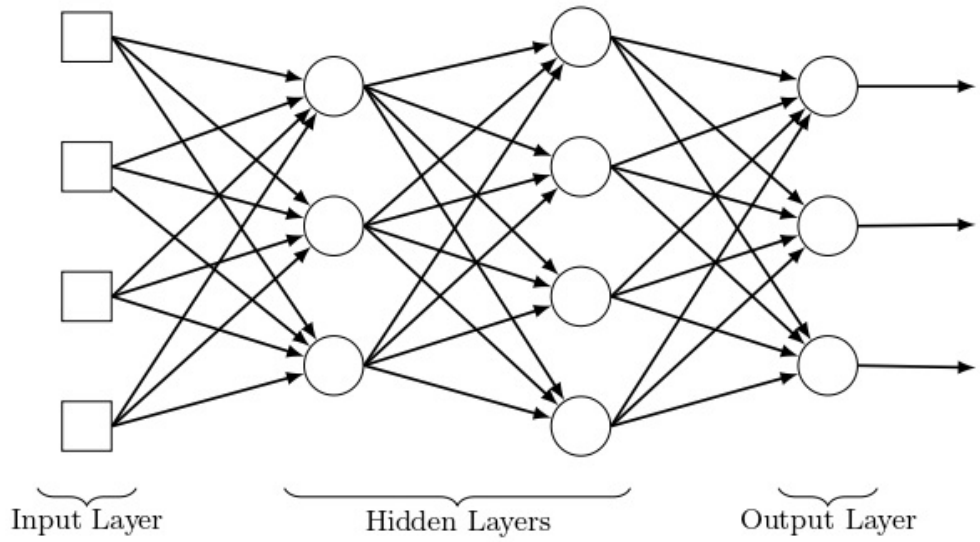



Figure 2.2: Basic structure of neural network.

the input signals to the subsequent nodes. On the rightmost side of the network is the output layer. The output from these nodes represents the final result produced by the neural network after processing the input through various layers. The layers located between the input and output layers are known as hidden layers. These hidden layers play a crucial role in processing and transforming the input data by extracting features and identifying patterns that help the network make accurate predictions or classifications in the output layer.

In a layered neural network, the signal flows in a structured, step-by-step manner. It first enters through the input layer, where raw data is introduced into the network. The signal then moves through one or more hidden layers, where it undergoes a series of transformations. Each hidden layer extracts relevant features from the data, enabling the network to recognize patterns and refine its understanding. At each layer, all nodes process the incoming signal simultaneously and pass their outputs to the nodes in the next layer, continuing this process until reaching the output layer. The final processed signal in the output layer represents the network's result, such as a prediction or classification, based on the input data. This layer-

by-layer progression allows the network to gradually refine and enhance the signal, leading to more accurate outcomes.

#### Types of ANNs

There are several types of neural networks, each designed for specific purposes based on the problem being addressed. While this list is not exhaustive, the following types represent the primary neural networks commonly encountered in various applications.

#### Multi-Layer Perceptrons (MLP)

Multi-layer perceptrons, also known as feedforward neural networks created by Frank Rosenblatt [56], are the simplest type of neural network. In this architecture, information flows in one direction, from the input layer to the output layer. This type of network is commonly used for basic pattern recognition and classification tasks and serves as the foundation for more complex neural network architectures.

#### Convolutional neural networks (CNNs)

These networks leverage principles from linear algebra, particularly matrix multiplication, to identify patterns within images. This makes them ideal for tasks such as image classification, object detection, and other applications in computer vision.

#### Recurrent neural networks (RNNs)

These RNNs are characterized by their feedback loops, which allow information to persist across time steps. These learning algorithms are primarily utilized with time-series data to make predictions about future outcomes. Variants such as Long Short-Term Memory (LSTM) networks address the limitations of traditional RNNs by effectively retaining long-term dependencies.

### 2.3.2 Applications of Artificial Neural Networks in Robotics

#### Neural networks in kinematics

The development of inverse kinematics models is one of the challenging problems in robotics due to its computational intensity and the existence of multiple solutions. This computational complexity can be reduced by utilizing neural networks to address the inverse kinematics problem. However, while analytical solutions for inverse kinematics yield numerically accurate results, neural network solutions may not achieve the same level of precision.

Köker et al. [57] have used a designed neural network (NN) to solve the inverse kinematics problem for a three-joint robotic manipulator. In reference [58], the authors investigated forward kinematics for hybrid robots with a parallelserial structure by employing a neural network based on radial basis functions (RBF). Their study explores the effectiveness of this approach in accurately modeling the kinematic behavior of this type of robots. A new neural network model for solving inverse kinematics is proposed in [59]. The novelty of this approach lies in

incorporating the current joint angle configuration into the input pattern of the neural network, alongside the desired position and orientation. This additional input improves the accuracy of the NN estimation of joint angles, enhancing its ability to generate precise outputs for complex robotic movements. In [60], Xuehong Sun employs a dynamic neural network, rather than a static one, using new learning rules known as a fast-learning neural network. This approach, characterized by high identification accuracy for both linear and non-linear systems, is applied to identify the non-linear kinematics model of robots. According to to [61], experimental results have shown that geometry-based inverse kinematics for Delta robots may not achieve sufficient accuracy in reaching desired positions due to measurement errors, joint flexibility, and backlash. To address this, a data-driven model using a neural network is employed to approximate the inverse kinematics of such robots equipped with stepper motors.

### Neural networks in control

For effective control of robots, an accurate understanding of the robot's dynamics is essential. However, in practice, the robot's dynamic model is complex and challenging to derive, not to mention the manufacturing uncertainties, friction, and disturbances that affect the robot's dynamics. Therefore, intelligent control methods are often required. Artificial neural networks offer effective solutions to these challenges, and much of the research in robotics has focused on their application in control systems.

Adaptive control using neural networks is an effective approach for managing uncertainties in a robot's dynamic model, particularly when both the structure and parameters are unknown [62]–[64]. In [65], a neural network-based PD control in cascade is designed to compensate for both structured and unstructured uncertainties in a robot manipulator. To enable precise tracking control for industrial robot tasks, a deep neural network is used to approximate the Jacobian matrix of a robot with unknown kinematic parameters [66]. Friction is an undesirable phenomenon in robotic control that can disrupt trajectory tracking, making it essential to estimate and compensate for friction within the control law to achieve high performance. To address this challenge, [67] proposes an approach in which a Radial Basis Function (RBF) neural network is used to estimate unknown nonlinear friction online, providing compensation signals that are then integrated into the controller. Li et al. [68] proposed a neural network-based approach for correcting positioning errors. The neural network is designed to predict the positioning errors of the robot, which are then used to adjust target points within the robots workspace for improved accuracy. For more research on this topic, see reference [69].

### Neural networks in motion planning

To address issues like slow motion planning, low accuracy, high path calculation costs and collision avoidance in robots, several neural network-based approaches have been proposed. In [70], A dynamic robot motion planning using NNs method is proposed, the robot to move autonomously in unknown environments by dynamically adapting its path in real time, leveraging the neural networks capacity to respond to unpredictable surroundings. A single-layer neural

network is used to predict or control aspects of the robot's movement [71], it is utilized to generate the desired base position, or optimal starting point, which helps suppress residual vibrations of the manipulator by minimizing displacement in the workspace. Ramya and Supriya [72] used a recurrent neural network (RNN) to generate a robot's path from a source to a destination while avoiding obstacles. The RNN model was trained on images of environments that include obstacles, enabling it to recognize and navigate around them effectively. Additionally, the algorithm is designed to find the shortest path among nodes, optimizing the robot's path for efficiency. A novel method called Neural-Network-Driven Prediction (NEED) [73] is presented for robot path planning problems. By training the NEED model on numerous successful path planning cases, it can analyze and predict the search region, serving as a heuristic to guide the search direction of path planning algorithms. A Bi-directional Rapidly-exploring Random Tree combined with Long Short-Term Memory (LSTM-BiRRT) technique for planning tasks involving dual-arm assembly robots in three-dimensional environments [74].

## 2.4 Evolutionary Algorithms

Evolutionary (metaheuristic) optimization algorithms are computational techniques inspired by various natural processes [75]. These algorithms are among the most widely used methods for addressing diverse and complex applications, particularly in scenarios where conventional optimization methods fall short. One notable area of application is in robotics systems, where evolutionary algorithms can effectively tackle intricate optimization challenges. In this review, we explore the application of evolutionary algorithms in industrial robotics. Specifically, we investigate widely used evolutionary computation techniques, such as Particle Swarm Optimization (PSO), Genetic Algorithms (GA) and Grey Wolf Optimization (GWO). Our focus will be on how these methods can enhance robotic performance, improve efficiency and solve complex optimization problems in various industrial robotics.

### 2.4.1 Particle Swarm Optimization

It is one of the oldest evolutionary computation techniques; however, it is still extensively studied and used in various applications. This method is inspired by the foraging behavior observed in bird flocking and fish schooling. In this method, a population of particles iteratively searches for an optimal solution by adjusting their positions within the search space, based on knowledge of the positions of other particles in the swarm [76]. The basic mechanism of PSO is

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{2.1}$$

where $x_i^{(t)}$ is current position of particle $i$ at iteration $t$ and $v_i^{(t+1)}$ represents velocity of particle $i$ at iteration $t+1$, which is calculated based on best position of particles $p_i^{(t)}$ and the best position in the swarm $g^{(t)}$ as

$$v_i^{(t+1)} = w.x_i^{(t)} + c_1.r_1(p_i^{(t)} + x_{(i)}^t + c_2.r_2(g^{(t)} + x_i^{(t)}) \tag{2.2}$$

where $w$ represent the inertia weight, $c_1$ and $c_2$ are acceleration coefficients and $r_1$ and $r_2$ are random numbers between 0 and 1.

Optimal trajectory planning is one of the most extensive applications of the PSO algorithm in solving motion planning problems for robotic arms. A 3-5-3 polynomial interpolation method based on PSO is proposed for time-optimal trajectory planning in the joint space of a 5-DOF manipulator [77]. An enhanced Hybrid Particle Swarm Optimization (Hybrid-PSO) algorithm is proposed to achieve time-optimal trajectory planning for industrial robots [78]–[80], refer to the review paper [81] for information on evolutionary computation algorithms in the trajectory planning of industrial robotics. The PSO algorithm has been also applied to address aspects of inverse kinematics in robotics. The authors in [82] propose an improved PSO algorithm for calculating inverse kinematics, which is applicable to various types of robots. The particle swarm optimization algorithm is susceptible to local minima. To address this issue, David et al.[83] propose a modification of Fully Resampled PSO for solving the inverse kinematics of robot arms. In the field of robotic control, a self-tuning fuzzy PID controller is designed using PSO [84], a super-twisting sliding mode controller with adaptive capabilities is proposed for an exoskeleton robot, this adaptation utilizes the PSO algorithm, focusing on online tuning of the parameters to minimize the objective function [85]. A novel PSO technique is used to tune the PID controller for position control of the joints of a robotic manipulator [35]. A control method based on Nonlinear Active Disturbance Rejection Control (NADRC) using an extended state observer is proposed for robotic manipulators. An Improved Particle Swarm Optimization (IPSO) algorithm, based on chaos theory, is employed to optimize the key parameters of the controller [86]. Table 2.1 lists the reviewed some papers that applied PSO as the optimization method for various industrial robotics applications since 2020.

Table 2.1: Some literature on industrial robotics problems using PSO since 2020.

| Ref. | Year | Type of robot | Problem |
|---|---|---|---|
| [87] | 2023 | 6-DoF serial robot manipolator | Trajectory planning |
| [88] | 2020 | 6-DoF serial robot manipolator | Energy consumption |
| [89] | 2022 | Dual-arm robot | Energy consumption |
| [90] | 2021 | 7-DoF serial robot manipulator | Inverse Kinematics |
| [91] | 2021 | 6-DoF serial robot manipolator | Inverse Kinematics |
| [92] | 2020 | 7-DoF serial robot manipulator | Inverse Kinematics |
| [93] | 2021 | Humanoid robot | Robot control |
| [94] | 2022 | 5-DoF serial robot manipulator | Inverse Kinematics |
| [95] | 2022 | 5-DoF serial robot manipulator | Jerk minimization |
| [96] | 2020 | 3-DoF articulated robot arm | Robot control |
| [97] | 2021 | 3-DoF articulated robot arm | collision avoidance |
| [98] | 2022 | 2-DoF serial robot manipulator | Parameters identification |
| [99] | 2023 | 6-DoF serial robot manipulator | Parameters identification |

### 2.4.2 Genetic Algorithm

A genetic algorithm (GA) is an optimization method inspired by the process of natural selection and is used for optimization and search problems in various fields, including computer science, engineering and robotics. The GA performs its search based on a population of potential solutions and applies the concept of "survival of the fittest ". It simulates the process of natural selection to evolve better solutions over time. The algorithm typically follows these steps [100]:

- Population Initialization: Generate an initial population consisting of a set of individuals (potential solutions).
- Fitness Evaluation: Calculate the fitness score for each individual based on a defined criterion or objective function.
- Termination Check: If the termination condition (such as a fixed number of generations) is met, end the process. Otherwise, proceed to the next step.
- Parent Selection: Choose individuals for reproduction, favoring those with higher fitness scores.
- Crossover and Mutation: Modify the selected parents through crossover and mutation to produce offspring (children).
- Population Update: Form the new population by combining the offspring with some unchanged individuals from the previous generation.
- Loop: Repeat from step 2 until the termination condition is met.

Since the 2000s, the number of publications on the application of genetic algorithms (GAs) in robot manipulators has increased several-fold. Path planning is a key area in robotics that often benefits from the application of genetic algorithms. Garg and Kumar [101] noted that the paths of robot manipulators could be optimized using genetic algorithms (GA) by considering joint torque for a 2-DoF robot manipulator. They suggested that this approach could be extended to more complex robot manipulators. In [102], an evaluation was conducted on the efficiency of time-optimal and smooth trajectory optimization using a multiple population genetic algorithm (GA). The authors concluded that the approach is valid and yields promising results. Andrea et al. [103] present a method for trajectory planning and optimization that prevents collisions with human operators in a dual-arm robot's work-cell, using the occupancy data of the human operator, the optimization algorithm employed is a genetic algorithm (GA). Optimal time and continuous jerk for trajectory planning of industrial robots have been achieved using a hybrid approach combining the Whale Optimization Algorithm (WOA) and the GA [104]. By combining non-dominated sorting genetic algorithm-II (NSGA-II) with achievement scalarizing function (ASF), a multi-objective optimization technique achieves higher positional accuracy for a 6-DoF industrial robot, while minimizing the time-jerk-torque rate of the joint trajectory [105]. In order to establish an accurate dynamic model of a robot, which is key to

effective robot motion control, robot parameter identification is important. This has prompted many researchers to work within this framework. Feng et al. [106] propose a hybrid algorithm combining GA and PSO to identify the inertia parameters of the Selective Compliance Assembly Robot Arm (SCARA). In [107], the parameters of the Coulomb viscous friction and Stribeck friction models are identified using a genetic algorithm to enhance the dynamic model of a 6-DoF modular robot manipulator. Claudio and José [108] examined various parameter identification methods for the SCARA robot, including least squares, neural networks and genetic algorithms and evaluate their comparative performance. The study in [109] employs an adaptive genetic algorithm with a simulated annealing algorithm to determine the structural parameters of a six-degree-of-freedom industrial robot, where the authors observed that this method could be widely applied to the calibration of other types of robots. For robot control systems, genetic algorithms are commonly used as a tool for tuning and optimizing controller parameters, as [110] where GA-based fuzzy PID control is employed to control a robotic arm in the presence of external disturbances. A GA-based optimal computed torque control, along with an appropriate cost function, is proposed for controlling a tracker robot [111]. A controller integrating fuzzy logic, neural networks, and genetic algorithm techniques is developed for controlling a robotic manipulator, GAs are employed to optimize the fuzzy subsets [112]. GA-based nonlinear sliding mode control is applied to trajectory tracking of a 2-DoF robot arm [113]. In the work [114], a Multi-Objective Genetic Algorithm (MOGA) is utilized to optimize the sliding mode control trajectory for accurate tracking in a SCARA robot. A position control of a 3-RevoluteRevoluteRevolute (3-RRR) parallel robot using a PID controller optimized by a genetic algorithm [115]. Eltayeb et al. [116] presented a comparative analysis of a fractional-order proportionalintegralderivative (FO-PID) controller versus a standard PID controller, both optimized using a GA. Table 2 provides an overview of additional studies involving the application of genetic algorithms across various disciplines within industrial robotics.

Table 2.2: Some literature on industrial robotics problems using GAs since 2020.

| Ref. | Year | Type of robot | Problem |
|---|---|---|---|
| [117] | 2021 | SCARA robot | Robot control |
| [118] | 2021 | 6-DoF serial robot manipolator | Energy consumption |
| [119] | 2024 | Robotic Cell | Path planning |
| [120] | 2020 | 7-DoF redundant manipulator | Trajectory planning |
| [121] | 2020 | 2-DoF serial robot manipulator | Robot control |
| [122] | 2022 | 6-DoF serial robot manipulator | Trajectory planning |
| [123] | 2020 | 6-DoF serial robot manipulator | Trajectory planning |
| [124] | 2022 | Serial-parallel hybrid manipulator | Trajectory planning |
| [125] | 2021 | 2-DoF serial robot manipulator | Robot control |
| [126] | 2022 | SCARA robot | Robot control |
| [127] | 2022 | 6-DoF serial robot manipulator | Robot control |
| [128] | 2020 | 2-DoF serial robot manipulator | Energy consumption |

### 2.4.3 Grey Wolf Optimization

The Grey Wolf Optimization (GWO) algorithm, introduced by Mirjalili et al.[129], is a new metaheuristic optimization technique inspired by the natural hunting behavior of grey wolves. Gray wolves are organized into a hierarchy with four dominance levels: alpha ($\alpha$) as the primary leaders, beta ($\beta$) as second-in-command, delta ($\delta$) as third-level leaders, and omega ($\omega$) as the lowest-ranked members of the pack. The GWO algorithm starts by randomly generating a population of gray wolves. These wolves estimate the prey's (optimal solution's) location through an iterative process. During each iteration, the $\alpha$, $\beta$, and $\delta$ wolves determine the approximate position and distance to the prey. This process of encircling the prey continues until it becomes stationary, at which point the wolves launch their attack. The mathematical model for encircling the prey is as follows

$$\vec{D} = |\vec{C}\vec{X}_p^{(t)} - \vec{X}^{(t)}| \tag{2.3}$$

$$\vec{X}^{(t+1)} = \vec{X}_p^{(t)} - \vec{A}\vec{D} \tag{2.4}$$

$\vec{X}_p^{(t)}$ and $\vec{X}^{(t)}$ are current position vector of prey and grey wolf, respectively. The $\vec{A}$ and $\vec{C}$ are are coefficient vectors, they are calculated as

$$\vec{A} = 2\vec{a}\vec{r}_1 - \vec{a} \tag{2.5}$$

$$\vec{C} = 2\vec{r}_2 \tag{2.6}$$

$\vec{r}_1$ and $\vec{r}_2$ are random vectors in the range $[0, 1]$. The vector $\vec{a}$ linearly decreases from 2 to 0 over the iterations, as shown in the following equation

$$\vec{a} = 2\left(1 - \frac{t}{t_{max}}\right) \tag{2.7}$$

and the mathematical modeling of grey wolf hunting is represented as follows

$$\begin{cases} \vec{D}_\alpha = |\vec{C}_1\vec{X}_\alpha - \vec{X}| \\ \vec{D}_\beta = |\vec{C}_2\vec{X}_\beta - \vec{X}| \\ \vec{D}_\delta = |\vec{C}_3\vec{X}_\delta - \vec{X}| \end{cases} \tag{2.8}$$

$$\begin{cases} \vec{X}_1 = \vec{X}_\alpha - \vec{A}_1\vec{D}_\alpha \\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2\vec{D}_\beta \\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3\vec{D}_\delta \end{cases} \tag{2.9}$$

$$\vec{X}^{(t+1)} = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2.10}$$

Initially, the first three best solutions ($\alpha$, $\beta$ and $\delta$) are considered optimal parameters, as they have superior knowledge about the location of the prey. Throughout the iterations, $\alpha$, $\beta$ and

$\delta$ update their positions relative to the prey. If $|A| > 1$, the potential solutions (wolves) move away from the prey; otherwise, they converge toward the prey. The GWO algorithm concludes when a termination criterion is met or the specified number of iterations is reached.

It is witnessed that the Grey Wolf Optimization (GWO) algorithm has been widely applied in robotic field, demonstrating its effectiveness in areas such as kinematics, trajectory planning, and robot control. Inverse kinematics (IK) is a critical challenge in robotics. In [130], the GWO algorithm was applied to solve the IK problem for a 7-joint robot manipulator. The work [131] employed GWO and other bio-inspired algorithms to solve the structural parameter problem of a 6-DOF serial robot arm, and also in [132], GWO was applied to solve and optimize the forward and inverse kinematics problem of a SCARA robot. A randomness-enhanced Grey Wolf Optimizer (REGWO) is introduced to solve the inverse kinematics of redundant manipulators [133]. This enhanced version of GWO mitigates issues such as premature convergence and limited population diversity. Zafar et al.[134] propose a Deep Neural Network (DNN) model to compute the inverse kinematics of a 3-DoF robot manipulator. This DNN is optimized using the GWO technique. The GWO has also been applied in robotics trajectory planning, in [135], an approach based on GWO was presented for calculating the jerk-optimal trajectory of a 7-DOF redundant robot manipulator. An approach based on the GWO algorithm is proposed in [136] to generate a smooth, error-free continuous path motion for a 6-DOF parallel manipulator and its performance is compared with a GA technique. A new multi-strategy GWO algorithm is introduced for path planning in hydraulic robotic arms [137]. Beyond trajectory planning, GWO has proven to be effective in aspect of robotics control. Its capabilities extend to optimizing control strategies. As shown in [138], GWO has been applied to enhance the control law for trajectory tracking of a 3-DOF robotic manipulator. The primary objective of using GWO is to estimate the inertia matrix of the robot, which has a essential role in designing control laws for robotics. In [139], [140], to control a 2-DOF robot arm, an extended gray wolf optimizer was employed for tuning the parameters of sliding mode controller. In work of Zhou et al.[141], an extreme learning machine (ELM) model optimized by the gray wolf optimization algorithm is used to address the problem of external disturbances and uncertainties in robotics control. In the study [142], a self-adaptive fuzzy control system based on the gray wolf optimizer is designed for the PUMA 560 robotic arm. In this system, the GWO is utilized to adapt the fuzzy membership functions, enabling improved performance and responsiveness in the control of the robotic arm. In [143], a computed torque control technique optimized by the GWO is proposed for a 3-DoF robot manipulator. In [144], the parameters of an LQR-PID controller are tuned using GWO for trajectory control of a quadruped robot. Further recent studies involving the application of GWO in industrial robotics are provided in Table 2.3.

### 2.4.4 Other Methods

Beyond the aforementioned techniques, various other evolutionary computation algorithms have been applied in industrial robotics. This section provides a brief overview of the reviewed papers that discuss these methods.

Table 2.3: Recent studies of GWO in industrial robotics.

| Ref. | Year | Type of robot | Problem |
|---|---|---|---|
| [145] | 2019 | Parallel robot | Energy consumption |
| [146] | 2024 | 2-DoF Robot arm | Robot control |
| [147] | 2022 | 6-Dof Robot manipulator | Inverse kinematics |
| [148] | 2020 | 2-DoF Robot arm | Robot control |
| [149] | 2024 | 6-DOF robotic manipulator | Energy consumption |
| [150] | 2022 | 2-DOF robot arm | Robot control |
| [151] | 2023 | 3-DoF robot arm | Robot control |

The Cuckoo Search (CS) algorithm is a bio-inspired optimization algorithm based on the brood parasitism behavior of cuckoo birds. In robotics, the cuckoo search algorithm, combined with the imperialist competitive algorithm, was employed to solve the inverse kinematics problem of robotic manipulators [152]. An adaptation of CS algorithm, referred to as the adaptive cuckoo search (ACS) algorithm, was applied on 6-DOF robot manipulator for path planning to minimize the total motion time under strict dynamic constraints [153]. In [154], trajectory planning for a dual-arm robot was performed using a modified cuckoo search algorithm, taking into account multiple objectives to optimize the robot's performance. In the work by Tlijani et al.[155], a sliding mode control (SMC) approach, optimized using the CS algorithm, was implemented to enhance the accuracy of joint position control in a two-degree-of-freedom robotic system.

The Beetle Antennae Search (BAS) algorithm is an optimization technique that mimics the foraging behavior of beetles using their antennae. In [156], motion planning of a redundant robot with joint velocity constraints using the BAS algorithm. An inverse kinematic model for a redundant robot using a neural network optimized by BAS algorithm was proposed by [157]. Zhibin et al. [158] present a new industrial robot calibration approach based on the beetle antennae search method. Ameer Hamza et al. [159] proposed a control scheme combining a metaheuristic-based BAS algorithm with a recurrent neural network for tracking control and obstacle avoidance in robotic manipulators. The enhanced BAS optimization algorithm was employed to identify the dynamic parameters of the Zhichang Kawasaki RS010N industrial robot [160].

The Whale Optimization Algorithm (WOA) is a metaheuristic optimization method inspired by the hunting behavior of humpback whales. It was introduced by Seyedali Mirjalili and Andrew Lewis in 2016 [161]. Tuning PID parameters for robot manipulator control using the whale optimization algorithm (WOA) [162] offers an efficient approach to enhancing control performance. The WOA is capable of providing optimal controller parameters by leveraging its global search capabilities. In [163], an improved whale optimization algorithm (IWOA) is used for planning an optimal trajectory for a six-axis industrial robot. The work by Yufei et al. [164] used the method WOA to identify the nonlinear friction model of a hyper-redundant manipulator. This friction model plays an important role in achieving an accurate dynamic model for the robot, which is essential for precise control and efficient performance in complex

tasks.

The Artificial Bee Colony (ABC) algorithm is an optimization tool introduced by Dervis Karaboga in 2005 [165]. It mimics the foraging behavior of honeybee swarms. Zhenyong et al. [166] proposed the ABC algorithm for the path planning of a dual-chain robot. The work [167] proposed an artificial bee colony algorithm for solving the inverse kinematics of 7-degree-of-freedom robotic arm. In [168], the authors proposed the development of an intelligent controller based on the Artificial Bee Colony (ABC) algorithm to optimize the tuning of PID parameters. This method was applied to control a two-link flexible robot, addressing the problem of optimal tunning of controller. In the work by Yibing et al. [169], a novel approach combining reinforcement learning (RL) with the artificial bee colony algorithm was applied to robot path planning, this hybrid method leverages the exploration and exploitation capabilities of the ABC algorithm along with the decision-making prowess of reinforcement learning to optimize the robot's path.

The Differential Evolution (DE) algorithm is a robust and simple evolutionary optimization technique that is capable of handling nonlinear, non-differentiable and multi-modal objective functions effectively. In the work [170], an optimization of geometrically constrained path planning for a 6-joint robot using the differential evolution technique. In [171], the differential evolution algorithm was applied to solve the inverse kinematics of a 5-DoF robot manipulator. The authors in [172] proposed an energy-efficient path planning method for an industrial robot operating in a dynamic environment. The method is based on the DE algorithm, which was used to optimize the robot's movement paths by minimizing energy consumption while adapting to changes in the environment. According to [173], a control model based on an improved differential evolution algorithm is proposed to achieve accurate trajectory tracking control for the robots.

Another optimization algorithm inspired by nature is Ant Colony Optimization (ACO), which mimics the foraging behavior of ants to solve computational problems. The work [174] used ACO method to obtain optimal path planning of a 2-DOF robot. Huadong et al.[175] proposed an enhanced version of the ACO algorithm, known as the dynamic recursive ant colony optimization (DRACO) algorithm to optimize the motion of a SCARA robot. An adjustment mechanism for PID controller parameters using the ACO technique is proposed for trajectory tracking control of a robotic arm [176]. An optimal trajectory planning approach for pick-and-place operations in an industrial robot was presented in [177], the optimization was carried out using the ACO algorithm, initially within a digital twin environment and subsequently implemented on the real robot following an inspection.

Among the nature-inspired algorithms is the Butterfly Optimization Algorithm (BOA), which mimics how butterflies search for food or mates and has been applied in various robotics fields. The BOA was used to achieve an optimal time-jerk trajectory for 3-DoF Delta robot [178], this approach aims to enhance the robot's motion efficiency by minimizing both the travel time and jerk, ensuring smoother and more precise movements. Hung Quang et al. [179] proposed a hybrid approach for structural parameter identification of a parallel robot. The approach

combines a neural network with the Butterfly Optimization Algorithm (BOA). In [180], the authors designed a new robot manipulator controller based on an advanced butterfly optimization algorithm for accurate trajectory tracking control.
The table 2.4 also presents further review of research papers that explore the application of these methods in industrial robotics.

Table 2.4: Using evolutionary algorithm to solve industrial robotics problem.

| Ref. | Year | Problem | The method used |
|---|---|---|---|
| [181] | 2022 | Trajectory planning | CS |
| [182] | 2022 | Trajectory planning | WOA |
| [183] | 2023 | Robot control | WOA |
| [184] | 2020 | Robot control | ABC |
| [185] | 2014 | Robot control | DE |
| [186] | 2024 | Robot control | DE |
| [187] | 2022 | Path planning | ACO |
| [188] | 2024 | Model identification | BOA |

## 2.5 Conclusion

As computing power continues to increase and the cost of processing devices decreases, soft computing techniques are becoming increasingly important in industrial robotics. These techniques enable robots to make intelligent decisions, handle uncertainties, and select optimal solutions from a vast number of possibilities using advanced algorithms. Throughout this chapter, we explored the application of fuzzy logic, artificial neural networks, and evolutionary algorithms in various robotic tasks, such as assembly, welding, and pick-and-place operations. These methods have proven effective in enhancing adaptability, optimizing performance, and improving efficiency in industrial settings. Furthermore, soft computing techniques are already widely integrated into real-world industrial applications, and their adoption is expected to grow significantly over the next decade.

# Chapter 3

# Serial Robot Manipulator: Overview and Fundamentals

Abstract

This chapter provides an overview of serial robots, beginning with a brief introduction to serial manipulators. The kinematics of serial manipulators, including forward, inverse, and differential kinematics, is discussed in Section 3.2. Section 3.3 explores the dynamics of robots, presenting the equations of motion that describe their behavior. A concise review of various robot control strategies is provided in Section 3.4. The chapter concludes with a case study of the Fanuc 710ic/70 industrial robot, highlighting its modeling and control design.

## 3.1 Introduction

Serial robot manipulators are a fundamental class of industrial robots, widely utilized across industries for tasks such as assembly, welding, material handling and precision operations. Their design, consisting of a series of rigid links connected by joints, allows for flexible and precise movements in a structured environment. The performance and efficiency of these robots hinge on a deep understanding of their kinematics, dynamics and control mechanisms.

A serial robot manipulator consists of $n$ interconnected links joined by movable joints, forming a chain that extends from the robot's base to its end-effector, as illustrated in Figure 3.1. Each link $i$ (where $k = 1, ..., n$) is connected to a joint, whose position is represented by $q_i$. In serial robots, the number of links and joints typically corresponds to the robot's degrees of freedom (DOF). Each joint experiences a force or torque $\tau_i$, which is the result of external forces, interactions between links and the joint's actuator. The complete set of joint positions forms the joint position vector $q$, which defines the robot's posture, while the set of joint forces or torques constitutes the joint force vector $\tau$.

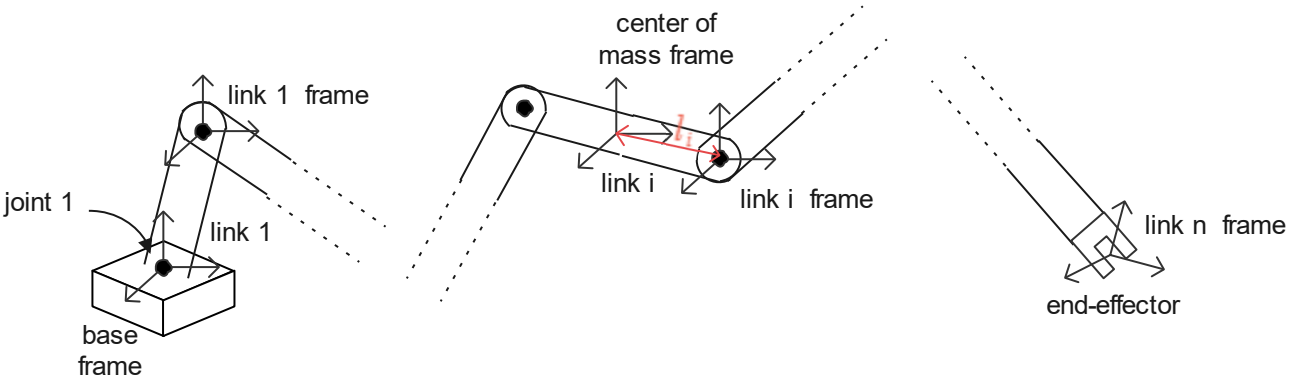



Figure 3.1: Links, joints and frames of a serial manipulator.

## 3.2 Kinematics

The kinematics of robots is cricial for analyzing their behavior,encompassing two main aspects: forward kinematics (FK) and inverse kinematics (IK). In this section, we will explore both topics. Forward kinematics involves determining the pose (position and orientation) of the robot's end-effector based on its joint variables, $q_i$. various methids can be used to solve this problem, such as the Denavit-Hartenberg (D-H) convention and the screw axis representation. In this study, we will focus exclusively on the D-H convention. Inverse kinematics, on the other hand, addresses the problem of calculating the joint variables required to achieve a desired end-effector pose. This is a fundamental task in robot trajectory planning and control, ensuring the robot operates effectively within its workspace.

### 3.2.1 Homogeneous Transformations

Referring to Figure 3.2, consider an arbitrary pose $P$ in space. Denote the vector of coordinates of $P$ relative to the reference frame $o_0 - x_0y_0z_0$ as $p^0$. Now, introduce another frame in space, $o_1 - x_1y_1z_1$. Let $o_1^0$ represent the vector describing the origin of frame 1 with respect to Frame 0, and $R_1^0$ denote the rotation matrix of frame 1 relative to frame 0. Furthermore, let $p^1$ be the vector of coordinates of $P$ relative to frame 1. Using basic geometric relationships, the position of point $p^0$ with respect to the reference frame can be expressed as follows

$$p^0 = o_1^0 + R_1^0 p^0 \tag{3.1}$$

Thus, (3.1) represents the coordinate transformation, comprising both translation and rotation, of a bound vector between two reference frames.

To obtain a concise representation of the relationship between the coordinates of a point in two different frames, the homogeneous representation of a generic vector $\tilde{p}$ can be introduced. This involves augmenting the vector $p$ with a fourth component set to one, resulting in the vector $p$, defined as

$$\tilde{p} = \begin{bmatrix} p \\ 1 \end{bmatrix} \tag{3.2}$$

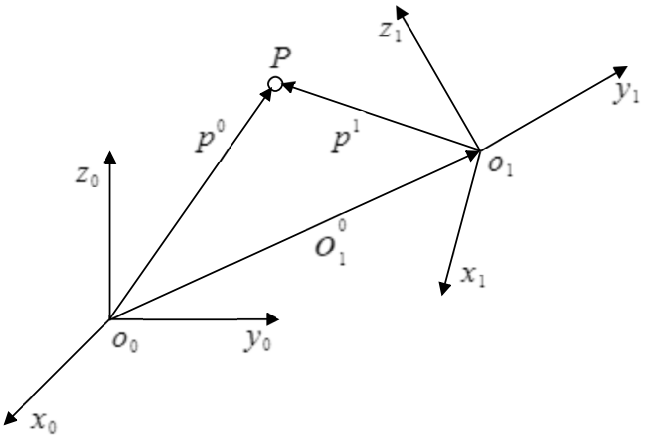



Figure 3.2: Representation of a pose $P$ in different coordinate frames.

Using this representation for the vectors $p^0$ and $p^1$ in (3.1), the coordinate transformation can be expressed in terms of a $(4 \times 4)$ matrix

$$A_1^0 = \begin{bmatrix} R_1^0 & o_1^0 \\ 0^T & 1 \end{bmatrix} \tag{3.3}$$

which, according to (3.2), is termed *homogeneous transformation matrix*. Consequently, the coordinate transformation in (3.1) can be compactly expressed as

$$\tilde{p}^0 = A_1^0 \tilde{p}^1 \tag{3.4}$$

### 3.2.2 Forward Kinematics

The process of computing the forward kinematics of serial robots naturally follows the structure of the open kinematic chain. Since each joint connects two consecutive links, it is logical to first describe the kinematic relationship between adjacent links. This can then be extended recursively to derive the overall kinematic description of the manipulator. To achieve this, a coordinate frame is assigned to each link, from Link 0 to Link 0. The transformation that defines the position and orientation of Frame $n$ relative to Frame 0 (as shown in Figure 3.3 is computed by sequentially combining the transformations between consecutive links and is expressed as

$$A_n^0(q) = A_1^0(q_1) A_2^1(q_2) \dots A_n^{n-1}(q_n) \tag{3.5}$$

#### DenavitHartenberg Convention

The Denavit-Hartenberg (D-H) convention [189] is a widely used methodology for systematically representing the kinematic structure of a robot manipulator. It provides a standardized way to describe the spatial relationship between adjacent links and joints in a robotic arm, simplifying the mathematical modeling of forward and inverse kinematics. Figure 3.4 shows

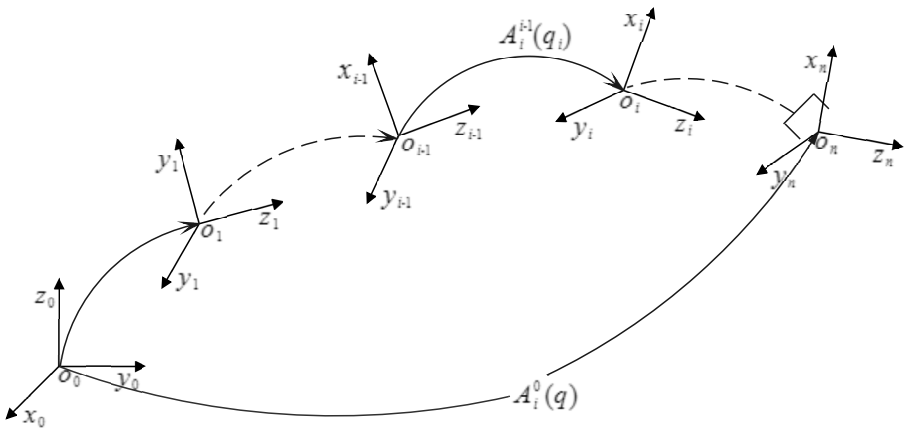



Figure 3.3: Coordinate transformations in an open kinematic chain.

the assignment and parameters of the axes according to the DH convention for two adjacent links, and their relative transformations are described using four parameters

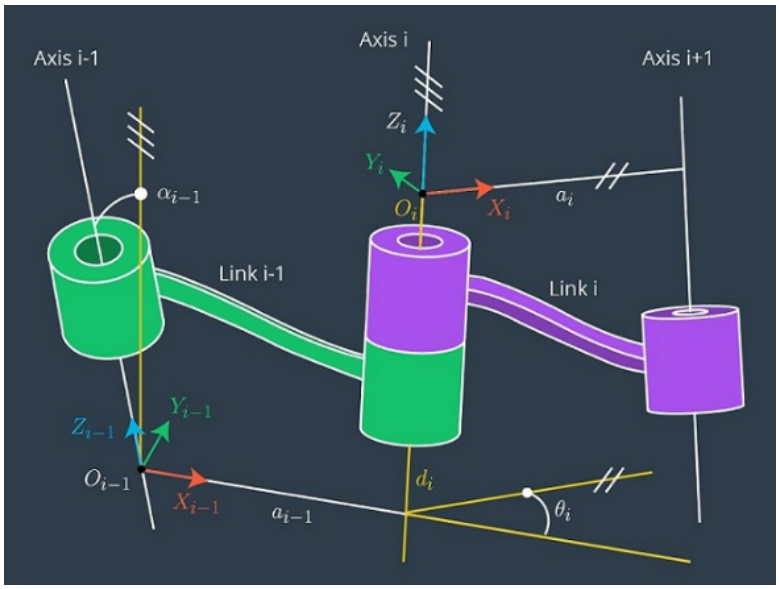



Figure 3.4: DenavitHartenberg kinematic parameters.

- $d_i$: Link offset, distance between $X_{i-1}$ and $X_i$, measured along $Z_{i-1}$.
- $\alpha_{i-1}$: Angle between $Z_{i-1}$ and $Z_i$, measured along $X_i$.
- $a_{i-1}$: Link length, distance between $Z_{i-1}$ and $Z_i$, measured along $X_i$.
- $\theta_i$: Joint angle, Angle between $X_{i-1}$ and $X_i$, measured along $Z_i$.

According to this convention, the total transformation between links $i-1$ and $i$ can be described step by step. First, there is a rotation by $\alpha_{i-1}$ around the $X_{i-1}$ axis. This is followed by a translation by $a_{i-1}$ along the $X_{i-1}$ axis. Next, there is a rotation by $\theta_i$ around the $Z_i$ axis, and finally, a translation by $d_i$ along the $Z_i$ axis. That is,

$$A_i^{i-1} = R(X_{i-1}, \alpha_{i-1})T(X_{i-1}, a_{i-1})R(Z_i, \theta_i)T(Z_i, d_i) \tag{3.6}$$

When expanded analytically, it becomes

$$A_i^{i-1} = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i) & 0 & a_{i-1} \\ sin(\theta_i)cos(\alpha_{i-1}) & cos(\theta_i)cos(\alpha_{i-1}) & -sin(\alpha_{i-1}) & -d_i sin(\alpha_{i-1}) \\ sin(\theta_i)sin(\alpha_{i-1}) & cos(\theta_i)sin(\alpha_{i-1}) & cos(\alpha_{i-1}) & d_i cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

### 3.2.3 Inverse Kinematics

Inverse kinematics is essential for controlling robotic manipulators and has been a subject of study for several decades. Solving the inverse kinematics problem is computationally intensive and can be time-consuming, particularly in real-time control scenarios. Manipulator tasks are performed in Cartesian space, which is represented using a homogeneous transformation matrix that encodes the position and orientation of the end-effector. Meanwhile, actuators operate in joint space, defined by joint angles. Transforming from Cartesian space to joint space is known as the inverse kinematics problem. The commonly employed methods for solving the inverse kinematics problem can be classified into four categories, as illustrated in Figure 3.5.

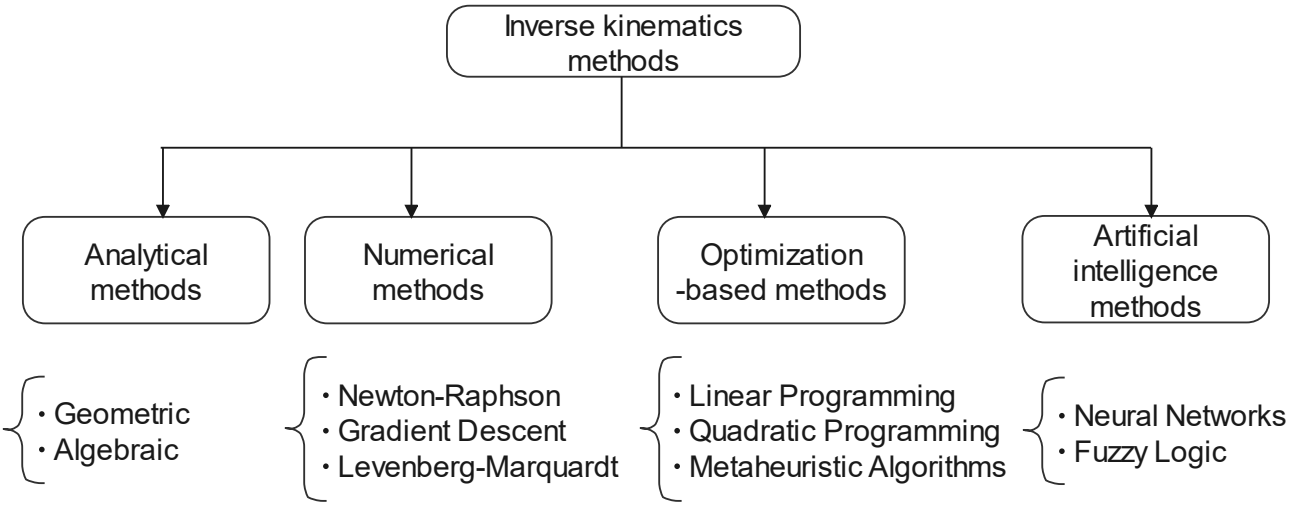



Figure 3.5: Inverse kinematics solving approaches.

#### Analytical Methods

The analytical method is a mathematical approach to solving the inverse kinematics problem by deriving exact solutions through closed-form equations. It relies on the geometry and structure of the robot and involves techniques like algebra, trigonometry, and matrix manipulation.

#### Numerical Methods

Numerical methods iteratively solve the inverse kinematics problem, providing approximate solutions instead of exact ones. These approaches are highly versatile, capable of handling

robots with complex or redundant kinematics, but they demand significant computational resources compared to analytical methods. Among these methods are Jacobian matrix-based techniques, which use the robot's kinematic properties to iteratively refine joint variables, and the Levenberg-Marquardt algorithm, which combines gradient descent with least-squares optimization to enhance robustness and convergence, making it ideal for non-linear, constrained problems.

#### Optimization-Based Methods

Optimization algorithms solve inverse kinematics by framing it as an optimization problem, where the goal is to minimize a cost function quantifying the error between the robot's current and target end-effector pose. These methods are particularly effective for handling non-linearities, redundancy, and constraints. Common techniques include Genetic Algorithms (GA), which use evolutionary principles to explore solutions, and Particle Swarm Optimization (PSO), which leverages swarm intelligence to find optimal configurations. These approaches are robust and versatile, making them ideal for complex robotic systems.

#### Artificial Intellegence Methods

Artificial Intelligence (AI) techniques such as neural networks and fuzzy logic provide innovative solutions to the inverse kinematics problem, especially for robots with complex or highly non-linear kinematics where conventional methods struggle. Neural networks learn the mapping from Cartesian space to joint space through training on data, using architectures like multi-layer perceptrons (MLPs), convolutional, or recurrent networks. Fuzzy logic systems, on the other hand, approximate inverse kinematics by applying rules based on linguistic variables, combining human-like reasoning with mathematical precision to address uncertainty and imprecision effectively.

### 3.2.4 Differential Kinematics

The concept of differential kinematics relates joint velocities to the linear and angular velocity of a robot's end-effector. This relationship is captured by the Jacobian matrix, which depends on the manipulator's configuration. The Jacobian is a fundamental tool in robotics, with applications in identifying singularities, analyzing redundancy, implementing inverse kinematics algorithms, mapping forces and torques between the end-effector and joints, deriving dynamic motion equations, and designing operational space control strategies. Its versatility makes it essential for manipulator analysis and control.

**Jacobian Computation**

Let us consider the forward kinematics equation of an $N$-DoF manipulator as

$$A_e = \begin{bmatrix} R_e(q) & p_e(q) \\ 0^T & 1 \end{bmatrix} \tag{3.8}$$

where $q = [q_1 \cdots q_N]^T$ is the vector of joint variables, $R_e(q)$ and $p_e(q)$ are orientation and position of end-effector respectively. The objective of differential kinematics is to establish the relationship between the joint velocities $\dot{q}$ and the linear velocity $\dot{p}_e$ and angular velocity $\omega_e$ of the end-effector, expressed as

$$\begin{bmatrix} \dot{p}_e \\ \omega_e \end{bmatrix} = J(q)\dot{q} \tag{3.9}$$

where the $(6 \times N)$ matrix $J$ represents manipulator *Jacobian* and

$$J = \begin{bmatrix} J_{p_1} \cdots J_{p_N} \\ J_{\omega_1} \cdots J_{\omega_N} \end{bmatrix} \tag{3.10}$$

where $J_{p_i}$ and $J_{\omega_i}$ discribe the effect of $i$-th joint on translational and rotational velocity, respectively. They can be evaluated based on the type of joint as follows

$$J_{p_i} = \begin{cases} z_{i-1} \times (O_N - O_{i-1}) & \text{for revolut joint } i \\ z_{i-1} & \text{for prismatic joint } i \end{cases} \tag{3.11}$$

$$J_{\omega_i} = \begin{cases} z_{i-1} & \text{for revolut joint } i \\ 0 & \text{for prismatic joint } i \end{cases} \tag{3.12}$$

The Jacobian matrix can be formulated by merging its upper and lower components as $J = [J_1 \cdots J_N]$, the column $J_i$ is defined as follows

$$Ji = \begin{bmatrix} z_{i-1} \times (O_N - O_{i-1}) \\ z_{i-1} \end{bmatrix} \tag{3.13}$$

if joint $i$ is revolute and

$$Ji = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} \tag{3.14}$$

if joint $i$ is prismatic.

## 3.3 Dynamics

The dynamic of an $N$-DoF manipulatro is a system of $N$ second-order differential equations describes the relationship between the first and second derivatives of the $N$ joint coordinates

with respect to time and the $N$ joint torques or forces. This modeling approach is essential for deriving the control laws required for precise position control of the robot.
There are two primary methods to derive these equations:

1. **Euler-Lagrange Method:** This approach is based on the principles of analytical mechanics, where the dynamic behavior of the system is derived from the Lagrangian, defined as the difference between kinetic and potential energies. It is particularly useful for systems with a clear energy representation.

2. **Newton-Euler Method:** This method relies on the direct application of Newton's second law of motion and rotational dynamics. It involves iterative calculations from the base to the end-effector (or vice versa) to compute the forces and torques acting on each link of the robot.

Both methods ultimately provide the equations of motion required to understand and control the dynamics of robotic manipulators.

### 3.3.1 Equation of Motion

In this section, we focus on application and specialization of the Euler-Lagrange equations to robotic systems. Let the Euler-Lagrange equation be

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{3.15}$$

where $L$ represents Lagrangian of the system that expressed by

$$L = K - V \tag{3.16}$$

where $K$ and $V$ are kinetic and potential energy of the system, and $\tau_i$ is the torque of joint $i$

**Kinetic energy**

the overall kinetic energy of the manipulator can be computed as

$$K = \sum_{i=1}^{n} \left( \frac{1}{2} m_i v_i^T v_i + \frac{1}{2} \omega_i^T I_i \omega_i \right) \tag{3.17}$$

where $m_i$ is the masse of link $i$, $I_i$ is the inertia tensor relative to the centre of mass of Link $i$ which given by following equation

$$I_i = \begin{bmatrix} \int_s (y^2 + z^2) dm_i & -\int_s xy dm_i & -\int_s xz dm_i \\ -\int_s xy dm_i & \int_s (y^2 + z^2) dm_i & -\int_s yz dm_i \\ -\int_s xz dm_i & -\int_s yz dm_i & \int_s (y^2 + z^2) dm_i \end{bmatrix} \tag{3.18}$$

where $dm_i$ is an infinitesimal mass element of a body S with dimensions $x, y$ and$z$.
As discussed in section 3.2.4 the linear and angular velocities of any point on a link can be represented using the Jacobian matrix and the time derivatives of the joint variables, we have that

$$v_i = J_{v_i}(q)\dot{q}, \quad \omega_i = J_{\omega_i}(q)\dot{q} \tag{3.19}$$

Thus, based on (3.17), the total kinetic energy of the manipulator can be expressed as

$$K = \frac{1}{2}\dot{q}^T \sum_{i=1}^{n} \left( m_i J_{v_i}(q)^T J_{v_i}(q) + J_{\omega_i}(q)^T R_i(q) I_i R_i(q)^T J_{\omega_i}(q) \right) \dot{q} \tag{3.20}$$

and can be rewritten as

$$K = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{3.21}$$

where $M(q)$ is an (n×$n$) symmetric positive-definite matrix, referred to as the inertia matrix of the manipulator.

### Potential energy

In a robot, gravity is the primary source of potential energy. This energy can be calculated by assuming that the entire mass of each link is concentrated at its center of mass. thus, the potential energy of manipulator is expressed as

$$V = -g_0^T \sum_{i=1}^{n} o_{c_i} m_i \tag{3.22}$$

where $g_0$ is the gravitational acceleration vector relative to the base frame and $o_{c_i}$ represents the coordinates of the center of mass of link $i$.
Having computed the kinetic and potential energy of the manipulator,as given in (3.21) and (3.22), the Lagrangian of the manipulator is expressed as

$$L = K - V = \frac{1}{2}\dot{q}^T M(q)\dot{q} + V = \tag{3.23}$$

$K$ can be expressed in the form

$$K = \frac{1}{2}\dot{q}^T M(q)\dot{q} = \frac{1}{2} \sum_{i=1,j=1}^{n} d_{ij}(q)\dot{q}_i\dot{q}_j \tag{3.24}$$

with $d_{ij}$ being the elements of the matrix $M(q)$. $V$ depends only on the configuration $q$: $V = V(q)$. Thus, the Lagrangian $L$ is written as

$$L = K - V = \frac{1}{2} \sum_{i=1,j=1}^{n} d_{ij}(q)\dot{q}_i\dot{q}_j - V(q) \tag{3.25}$$

we have that

$$\frac{\partial L}{\partial \dot{q}_k} = \sum_j d_{kj}(q)\dot{q}_j \tag{3.26}$$

and

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{q}_k} &= \sum_j d_{kj}(q)\ddot{q}_j + \sum_j \frac{d}{dt} d_{kj}(q)\dot{q}_j \\ &= \sum_j d_{kj}(q)\ddot{q}_j + \sum_{i,j} \frac{\partial d_{kj}(q)}{\partial d_i}\dot{q}_i\dot{q}_j \end{aligned} \tag{3.27}$$

also

$$\frac{\partial L}{\partial q_k} = \frac{1}{2}\sum_{i,j} \frac{\partial d_{ij}(q)}{\partial d_k}\dot{q}_i\dot{q}_j - \frac{\partial V(q)}{\partial d_k} \tag{3.28}$$

the Euler-Lagrange equation becomes

$$\sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\left(\frac{\partial d_{kj}}{\partial d_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial d_k}\right)\dot{q}_i\dot{q}_j + \frac{\partial V(q)}{\partial d_k} = \tau_k, \quad k = 1, \cdots, n \tag{3.29}$$

The equivalent matrix form of (3.29) is

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{3.30}$$

where $M$ is $(n \times n)$ matrix of termes $d_{ij}$, $C$ is called centrifugal and coriolis matrix of $(n \times n)$ termes $C_{kj}$ which is given as

$$\begin{aligned} C_{kj} &= \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i \\ &= \sum_{i=1}^{n} \frac{1}{2}\left(\frac{\partial d_{kj}}{\partial q_j} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right)\dot{q}_i \end{aligned} \tag{3.31}$$

and $G = \left[\frac{\partial V}{\partial q_1} \cdots \frac{\partial V}{\partial q_n}\right]^T$ is a gravitational forces matrix and $\tau = [\tau_1 \cdots \tau_n]^T$ is torque vector.

## 3.4 Control Strategies

The goal of motion control in industrial robots is to enable the position, velocity, and other state variables to follow a desired trajectory by applying the appropriate driving torque to each joint. To achieve accurate tracking of the reference trajectory, precise control of the robot is essential. Due to the flexibility, load variations, and unpredictable interferences inherent in industrial robots, it is crucial to design a control strategy that is robust, adaptable, and structurally simple [190]. In the control of industrial robots, various approaches for manipulators have been explored, ranging from traditional techniques to advanced methods.

PID control is one of the most commonly employed methods for controlling manipulators due to its simplicity. The PID control scheme operates by correcting the error between the actual and desired trajectories using three components: proportional, integral, and derivative actions.

The proportional action addresses the present error, the integral action accounts for the accumulation of past errors to eliminate steady-state offset, and the derivative action predicts and mitigates future errors by responding to the rate of error change [191].
The Computed Torque Control (CTC) is a widely used nonlinear control strategy tailored for the precise control of highly nonlinear and coupled robot manipulator, such as industrial robots. This control approach is rooted in the Feedback Linearization (FL) technique, a mathematical method used to simplify the complexity of nonlinear systems. The fundamental idea behind FL is to transform a nonlinear system into an equivalent linear system through a change of variables and input transformation. By leveraging the system's dynamic equations, CTC compensates for the inherent nonlinearities and couplings in real time. This process involves determining the robots dynamic model, including its inertia, Coriolis, and gravity effects, and then designing control inputs that cancel these nonlinear components. [192]–[194]
Sliding Mode Control (SMC) has gained significant attention in robust control of robotic manipulators. It offers a versatile approach for managing uncertainties, external disturbances, and the nonlinear dynamics commonly encountered in robotic systems [195]–[199]. The essence of Sliding Mode Control (SMC) lies in its capability to drive the system state toward a predefined sliding surface and ensure that it remains on this surface during subsequent motion. This sliding surface is designed to represent the desired system behavior, such as stability or trajectory tracking. Once the system state reaches the surface, SMC operates in a switching mode, rapidly alternating between control actions to maintain the system's trajectory on the sliding surface. This switching mechanism, which distinguishes SMC from other control methods, enables it to handle uncertainties and disturbances effectively, resulting in a robust and resilient response.
In recent years, intelligent techniques for robotic control have garnered increasing attention for their capacity to mimic human intelligence. These methods, which include artificial neural networks, fuzzy logic control, expert systems, metaheuristic algorithms, and machine learning, exhibit advanced decision-making abilities. They provide a deeper insight into the inputoutput dynamics of systems and improve conventional controllers by fine-tuning their parameters. Additionally, they excel at identifying and compensating for nonlinear uncertainties, thereby enhancing overall system performance [200].

## 3.5 Case Study

In this case study, we will explore the modeling and control of a specific type of industrial robot, namely the Fanuc M-710iC. This investigation will encompass the mathematical models describing the robot's kinematics and dynamics, as well as the design and implementation of effective control strategy. A thorough analysis of the Fanuc M-710iC will serve as a solid foundation for other applications will discusse in the subsequent chapters.

### 3.5.1 The Robot Modeling

The Fanuc M-710iC/70 is a multi-application industrial robot designed for handling payloads of up to 70 kg. This innovative series of lightweight robots features a slim wrist, a rigid arm, and a compact design, making it ideal for operations in space-constrained environments. With best-in-class load capacity and inertia handling, the six-axis M-710iC/70 is well-suited for a wide range of industrial applications. It combines a high payload capacity of 70 kg with exceptionally fast axis speeds, ensuring efficiency, precision, and optimal performance in demanding automation tasks [201].

#### Forward kinematics

To calculate the kinematics of the Fanuc 710i/70 robot, the robot's kinematic structure must first be validated to determine the Denavit-Hartenberg (D-H) parameters. Once the D-H parameters are established, the robot's kinematics can be derived. Figure 3.6 shows FANUC 710iC/70 robot dimensions and its workspace. figure 2 robot architucture and frames, table 1 shows D-H parameters.

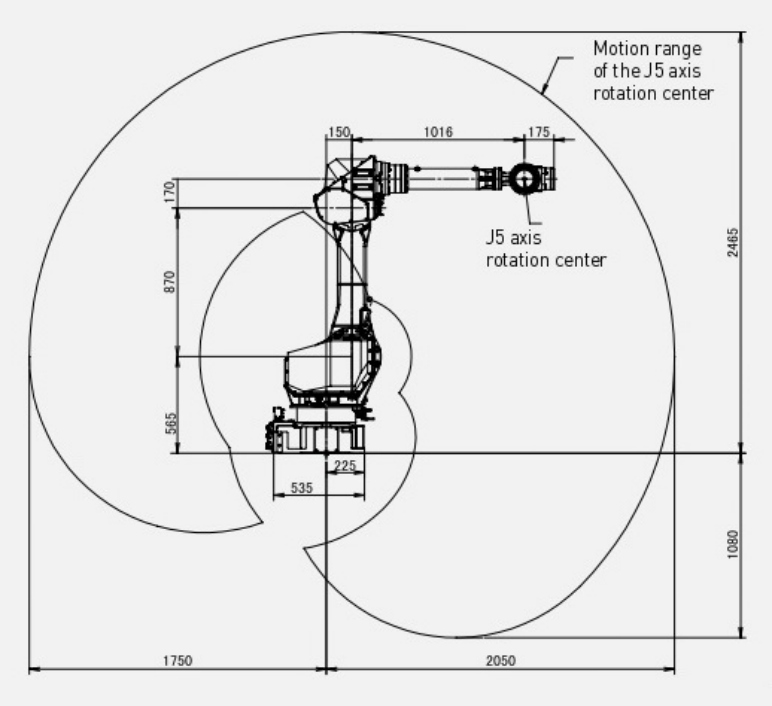



Figure 3.6: Robot dimensions.

From a methodological persepective, we first align the $z_i$ axes with the joint axes, foloowed by the placement of the $x_i$ axes. this process determines the robot's geometric parameters. the arrangement of these frames is illustrated in Figure 3.7.
The robot's coordinate axes make it possible to derive the D-H parameters listed in Table 3.1 below. Using D-H parameters, the homogeneous transformation matrices are defined as

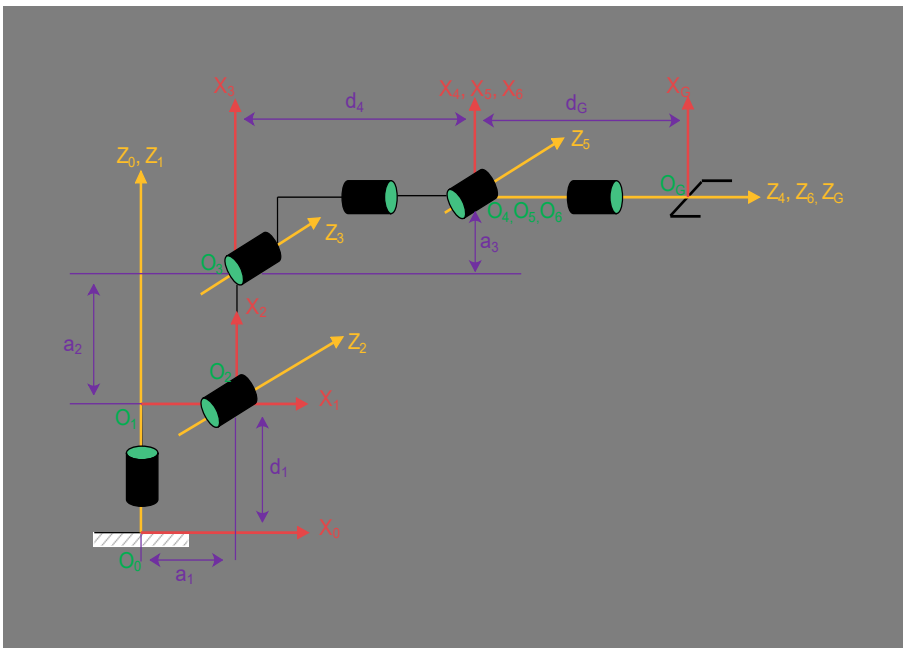



Figure 3.7: Kinemtic diagramme of Fanuc 710iC/70.

Table 3.1: D-H parameters of the robot.

| Joint $i$ | $\alpha_{i-1}(^\circ)$ | $a_{i-1}(mm)$ | $d_i(mm)$ | $\theta_i(^\circ)$ |
|---|---|---|---|---|
| 1 | 0 | 0 | $d_1 = 565$ | $q_1$ |
| 2 | -90 | $a_1 = 150$ | 0 | $q_1$-90 |
| 3 | 0 | $a_2 = 870$ | 0 | $q_3$ |
| 4 | -90 | $a_3 = 170$ | $d_4 = 1016$ | $q_4$ |
| 5 | 90 | 0 | 0 | $q_5$ |
| 6 | -90 | 0 | 0 | $q_6$ |
| 7(Gripper) | 0 | 0 | $d_G = 175$ | 0 |

follows

$$A_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_2^1 = \begin{bmatrix} s_2 & c_2 & 0 & a_1 \\ 0 & 0 & 1 & 0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_3^2 = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$A_4^3 = \begin{bmatrix} c_4 & -s_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_5^4 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, A_6^5 = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.32)$$

with $c_i = cos(q_i)$ and $s_i = sin(q_i)$. The forward kinematics of the end-effector relative to the base frame is determined by the successive multiplication of all $A_i^{i-1}$ matrices, it can be written as

$$A_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5 \quad (3.33)$$

Let $G$ be the transformation matrix that defines the orientation and position of the end-effector (gripper) relative to sixth link frame.

$$G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & d_G \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

where a $d_G$ value relative to the added tool length. Thus, the the forward kinematics of the attached tool relative to the base frame becomes

$$A_E^0 = A_6^0 G = \begin{bmatrix} s_x & n_x & a_x & P_x \\ s_y & n_y & a_y & P_y \\ s_z & n_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

with

$$\begin{cases} P_x = a_1c_1 + d_4c_{(2+3)}c_1 + a_3s_{(2+3)}c_1 + a_2c_1s_2 + d_Gc_{(2+3)}c_1c_5 \\ \qquad -d_Gs_1s_4s_5 - d_Gc_1c_2c_4s_3s_5 - d_Gc_1c_3c_4s_2s_5 \\ P_y = a_1s_1 + d_4c_{(2+3)}s_1 + a_3s_{(2+3)}s_1 + a_2s_1s_2 + d_Gc_{(2+3)}c_5s_1 \\ \qquad +d_Gc_1s_4s_5 - d_Gc_2c_4s_1s_3s_5 - d_Gc_3c_4s_1s_2s_5 \\ P_z = d_1 + a_3c_{(2+3)} - d_4s_{(2+3)} + a_2c_2 - \frac{1}{2}d_Gc_{(2+3)}s_{(4+5)} \\ \qquad -d_Gs_{(2+3)}c_5 + \frac{1}{2}d_Gs_{(2-3)}c_{(2+3)} \end{cases} \tag{3.36}$$

such that $c_{(i\pm j)} = cos(q_i \pm q_j)$ and $s_{(i\pm j)} = sin(q_i \pm q_j)$.

### Inverse kinematics

The inverse kinematics problem of a robot involves determining the joint coordinates that correspond to a specified position and orientation of the end-effector. When a solution exists, the representation that provides all possible solutions is referred to as the Inverse Kinematic Model (IKM).

The Fanuc M710iC/70 robot features an anthropomorphic structure, meaning the last three joints ($q_4, q_5$ and $q_6$) do not influence the position of the wrist center $P_w$ ($O_5$). This characteristic is particularly advantageous as it allows for decoupling the inverse kinematics problem into two separate tasks: position and orientation. Initially, the position of the wrist center $P_w$ can be computed to determine the first three joint angles ($q_1, q_2$ and $q_3$). Subsequently, the orientation of the wrist is calculated to obtain the remaining joint angles ($q_4, q_5$ and $q_6$).

Given the end-effector position and orientation, the wrist center position is computed as follows

$$P_w = P_e - d_G R_e \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.37}$$

where $P_e$ and $R_e$ the target position and orientation of the end-effector.
Consider the elbow manipulator illustrated in Figure 3.8. The wrist center position $P_w$ is defined by its components: $P_{w_x}$, $P_{w_y}$ and $P_{w_z}$. To analyze its position, we project $P_w$ onto the $x_0 - y_0$ plane. We see from this projection that

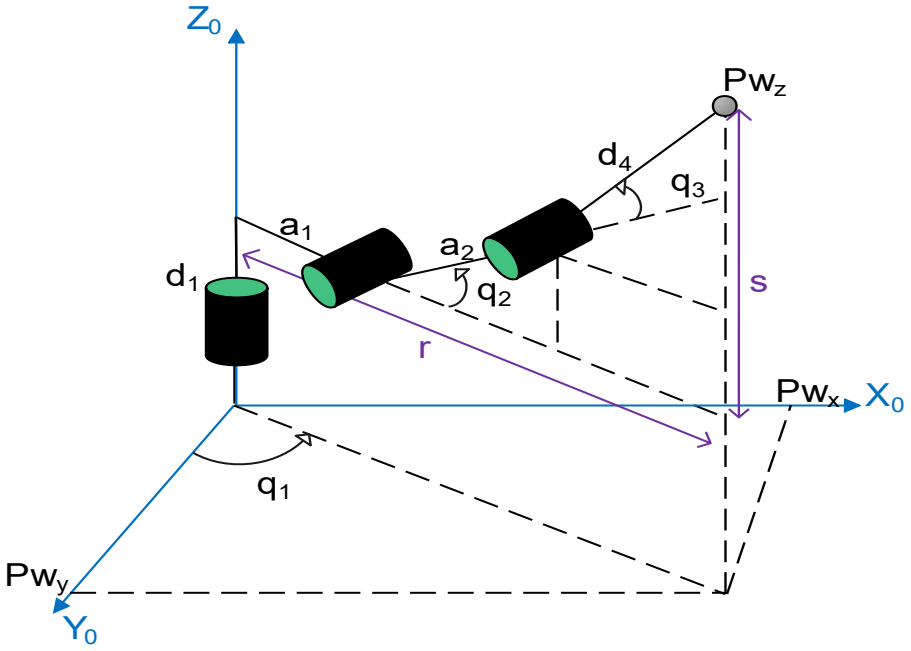



Figure 3.8: Elbow manipulator.

$$q_1 = arctan2(P_{w_x}, P_{w_y}) \tag{3.38}$$

From the Figure 3.8

$$\begin{cases} s = P_{w_z} - d_1 = a_2 sin(q_2) + d_4 sin(q_2 + q_3) \\ r - a_1 = \sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1 = a_2 cos(q_2) + d_4 cos(q_2 + q_3) \end{cases} \tag{3.39}$$

moreover

$$\begin{cases} (P_{w_z} - d_1)^2 = a_2^2 sin^2(q_2) + d_4^2 sin^2(q_2 + q_3) + 2a_2 d_4 sin(q_2) sin(q_2 + q_3) \\ (\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)^2 = a_2^2 cos^2(q_2) + d_4^2 cos^2(q_2 + q_3) + 2a_2 d_4 cos(q_2) cos(q_2 + q_3) \end{cases} \tag{3.40}$$

we can apply the law of cosines to obtain

$$(P_{w_z} - d_1)^2 + (\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)^2 = a_2^2 + d_4^2 + 2a_2 d_4 cos(q_3) \tag{3.41}$$

$$cos(q_3) = \frac{(P_{w_z} - d_1)^2 + (\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)^2 - a_2^2 - d_4^2}{2a_2 d_4} = Q \tag{3.42}$$

Hence, $q_3$ is given by

$$q_3 = \pm arcos(Q) \tag{3.43}$$

To obtain $q_2$, through trigonometry formulas (3.39) becomes

$$\begin{cases} P_{w_z} - d_1 = k_1 sin(q_2) + k_2 cos(q_2) \\ \sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1 = k_1 cos(q_2) - k_2 sin(q_2) \end{cases} \tag{3.44}$$

with

$$\begin{cases} k_1 = a_2 + d_4 cos(q_3) \\ k_2 = d_4 sin(q_3) \end{cases} \tag{3.45}$$

By solving the system (3.44), we obtain

$$\begin{cases} cos(q_2) = \frac{k_2(P_{w_z} - d_1) + k_1(\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)}{k_1^2 + k_2^2} \\ sin(q_2) = \frac{k_1(P_{w_z} - d_1) - k_2(\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)}{k_1^2 + k_2^2} \end{cases} \tag{3.46}$$

thus

$$q_2 = arctan2(k_1(P_{w_z} - d_1) - k_2(\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1), k_2(P_{w_z} - d_1) + k_1(\sqrt{P_{w_x}^2 + P_{w_y}^2} - a_1)) \tag{3.47}$$

Now, we need to calculate $q_4$, $q_5$ and $q_6$. To do this, we first compute the rotation matrix $R_E^3$. Since $q_1$, $q_2$ and $q_3$ are already determined, $R_0^3$ is known. The desired rotation $R_E^0$ is also given. Therefore, $R_E^R$ can be calculated as

$$R_E^3 = (R_3^0)^{-1} R_E^0 \tag{3.48}$$

Let:

$$R_E^3 = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.49}$$

which corresponds to

$$R_E^3 = \begin{bmatrix} c_4 c_5 c_6 - s_4 s_6 & -c_6 s_4 - c_4 c_5 s_6 & -c_4 s_5 \\ c_6 s_5 & -s_5 s_6 & c_5 \\ -c_4 s_6 - c_5 c_6 s_4 & c_5 s_4 s_6 - c_4 c_6 & s_4 s_5 \end{bmatrix} \tag{3.50}$$

By matching (3.49) and (3.50):
If $q_5 \neq 0[\pi]$

$$\begin{cases} q_5 = \pm arccos(r_{23}) \\ q_4 = -arctang(r_{33}, r_{13}) \\ q_6 = -arctang(r_{22}, r_{21}) \end{cases} \tag{3.51}$$

If $q_5 = 0[\pi]$

$$\begin{cases} q_4 = 0 \\ q_6 = arctang(r_{12}, r_{32}) \end{cases} \tag{3.52}$$

### Dynamics

Due to the complex structure of robots, establishing an analytical dynamic model is challenging. In this section, we leverage the Simscape Multibody/Matlab tool to construct the robot's dynamic model. Simscape Multibody (formerly SimMechanics) provides a robust simulation environment for 3D mechanical systems, including robots, vehicle suspensions, construction equipment and aircraft landing gear. In this tool, multibody systems are modeled using blocks that represent bodies, joints, constraints, force elements, and sensors. Simscape Multibody not only formulates and solves the equations of motion for the entire mechanical system but also enables the import of complete CAD assemblies. Additionally, its automatically generated 3D animations allow for visualization of system dynamics, providing valuable insights. These capabilities help in developing control systems and testing system-level performance effectively [202].

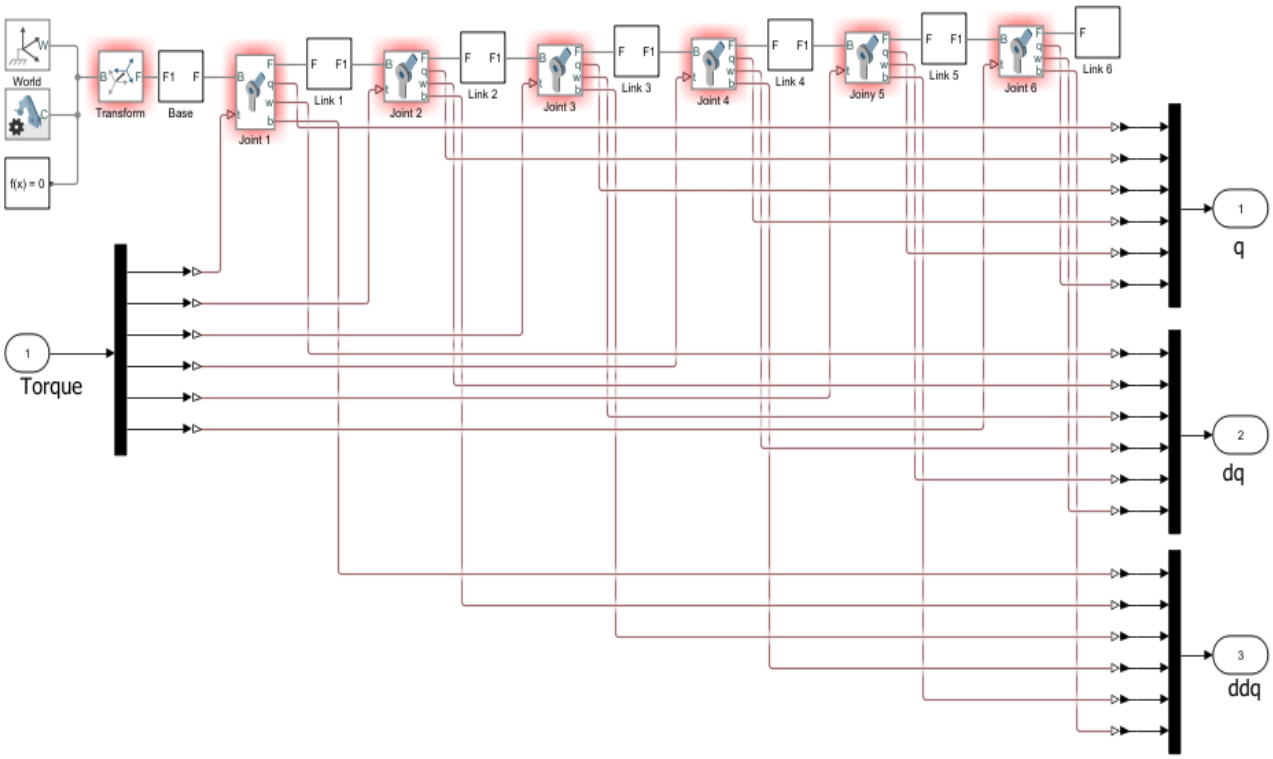



Figure 3.9: Combined SimulinkSimscape diagram (Fanuc M710iC/70).

The process of creating the dynamic model of the Fanuc M-710iC/70 robot in MATLAB-Simulink Simscape Multibody began with the 3D model of the robot, which was made available through the RoboDK software tool. This model provided a detailed and accurate representation of the robot, serving as the foundation for further development. The next step involved opening and editing the 3D model in Solidworks, a CAD program selected due to its compatibility with

the Simscape environment, facilitated by a translator from CAD to Simscape. After finalizing the robotic assembly in Solidworks, the model was exported to Simscape. The export process generated an XML file. This XML file was subsequently imported into MATLAB-Simulink, resulting in the creation of two key files: a Simulink model file (.slx) and an associated data file (.m). The Simulink-Simscape simulation diagram generated from the import process (the .slx file) is presented in Figure 3.9. This diagram forms the core of the simulation environment. Additionally, this diagram illustrates the subsequent integration of the imposed trajectory, sourced from a CAM (Computer-Aided Manufacturing) program. This trajectory was processed by the Simscape model, converting it into motion data, including joint angles, velocities and accelerations. The simulation outputs the torques required at each joint to execute the specified trajectory accurately. This diagram represents the dynamic behavior of the robot, which is actuated by input torques and generates the corresponding position, velocity, and acceleration of its articulations. This simulation environment provides a powerful tool for analyzing and testing system-level performance, enabling the evaluation of the robot's response to various control inputs and operating conditions effectively (see Figure 3.10).

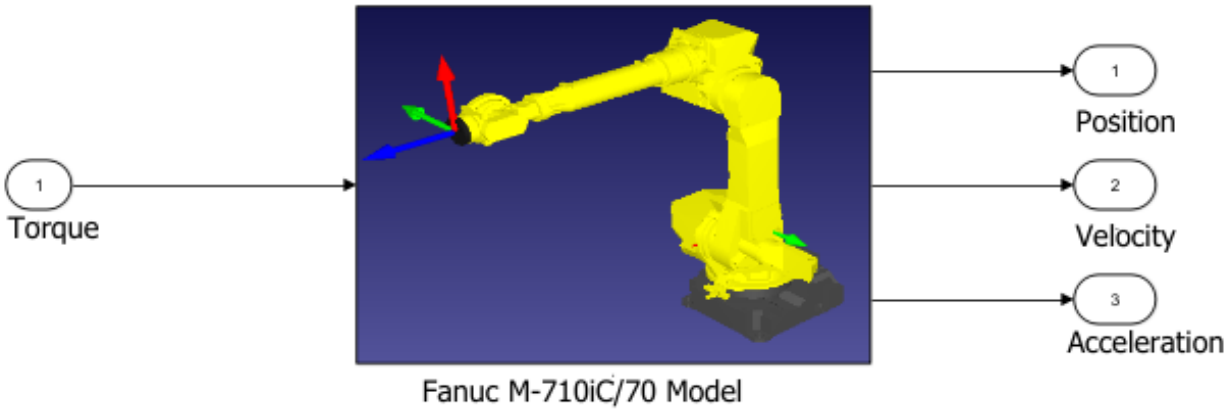



Figure 3.10: Fanuc M-710iC/70 industrial robot model in Simulink.

### 3.5.2 The Robot Control Design

Computed torque control (CTC) stands as an effective control method for achieving precise trajectory tracking in robotic manipulators. Nevertheless, computed torque control necessitates accurate dynamic models of robotic manipulators and is notably susceptible to the adverse impact of uncertain dynamics. This section presents a comprehensive approach to trajectory tracking control for robot manipulators that excels in handling disturbances encountered during operation. The combination of velocity observer-based computed torque control techniques results in a versatile and robust control framework that can significantly enhance the reliability and performance of robot manipulators in practical applications.

**Computed Torque Control**

This approach based on the robots dynamic model. The dynamic equation of an n-joint serial robot can be described in the joint-space coordinate frame as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_d \tag{3.53}$$

and can be written as follows:

$$\ddot{q} = M^{-1}(q)(\tau + \tau_d - C(q,\dot{q})\dot{q} - G(q)) \tag{3.54}$$

where, $q, \dot{q}$ and $\ddot{q} \in \mathbb{R}^{n\times 1}$ are the vectors of the joint position, velocity and acceleration, respectively.$M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix ,$C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal forces matrix, $G(q) \in \mathbb{R}^{n\times 1}$ is the gravity vector, $\tau$ is the torque and $\tau_d \in \mathbb{R}^{n\times 1}$ represents uncertainties and the external disturbance vector.
In trajectory tracking of a robot manipulator, to ensure that the joint variable $q$ follows the desired trajectory $q_d$, the tracking error is defined as follows

$$e = q_d - q \tag{3.55}$$

Thus

$$\ddot{e} = \ddot{q_d} - \ddot{q} \tag{3.56}$$

By substituting (3.54) into (3.56):

$$\ddot{e} = \ddot{q_d} - M^{-1}(q)(\tau + \tau_d - C(q,\dot{q})\dot{q} - G(q)) \tag{3.57}$$

The (3.57) can be written in the form of the error state space as

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{3.58}$$

where $u$ is a virtual input equal

$$u = \ddot{q_d} - M^{-1}(q)(\tau + \tau_d - C(q,\dot{q})\dot{q} - G(q)) \tag{3.59}$$

From (3.59), the computed joint torque $\tau$ is given by

$$\tau = M(q)(\ddot{q_d} - u) - \tau_d - C(q,\dot{q})\dot{q} - G(q) \tag{3.60}$$

Select the control signal $u$ as the proportional-Derivative (PD) Feedback

$$u = -k_p e - k_d \dot{e} \tag{3.61}$$

By substituting (3.61) into (3.60), the computed joint torque which is the robot manipulator input becomes

$$\tau = M(q)(\ddot{q}_d + k_p e + k_d \dot{e}) - \tau_d - C(q,\dot{q})\dot{q} - G(q) \tag{3.62}$$

The equation for the entire system can be derived from (3.53) and (3.62) by

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = M(q)(\ddot{q}_d + k_p e + k_d \dot{e}) - C(q,\dot{q})\dot{q} - G(q) \tag{3.63}$$

Therefore, we can get the closed control system as

$$\ddot{e} + k_d \dot{e} + k_p e = 0 \tag{3.64}$$

we design $k_p$ and $k_d$ so that all the roots of the polynomial $s^2 + k_d s + k_p = 0$ are in the left part of the complex plane. Then, we have $t \to \infty$, $e(t) \to 0$, and $\dot{e}(t) \to 0$.

### Velocity Observer

From (3.62), we know if the function $\tau_d$ is unknown, the *Computed Torque Control* will not be realized. In this part, we design the velocity observer to estimate $\tau_d$. This approach is based on employing a reduced-order observer for dynamically estimating the joint velocity $\dot{q}$. The observer is then integrated into a disturbance estimation framework to account for the unknown external joint torque $\tau_d$. Unlike conventional observers that operate on the full state space of dimension $2n$ (as required for a mechanical system with $n$ generalized coordinates), this reduced observer only considers an $n$ dimensional state. As a result, it responds more quickly to variations in the external joint torque, behaving as a first-order system.
The actual acceleration $\ddot{q}$ is

$$\ddot{q} = M^{-1}(q)(\tau + \tau_d - C(q,\dot{q})\dot{q} - G(q)) \tag{3.65}$$

The observer dynamic is then defined as

$$\begin{cases} \hat{\ddot{q}} = \hat{M}^{-1}(q)(\tau - \hat{C}(q,\dot{q})\dot{q} - \hat{G}(q) + r) \\ \dot{r} = L(\ddot{q} - \hat{\ddot{q}}) \end{cases} \tag{3.66}$$

where $r = \hat{\tau}_d$, and $r \in \mathbb{R}^{n\times 1}$ is residual vector, which represent disturbance torque estimation, and $L = diag\{l_i\} > 0$ is a gain matrix of observer. 3.11 shows the block diagram of the control strategy. The observer output can defined as

$$r(t) = L(\dot{q} - \int_0^t \hat{M}^{-1}(q)(\tau - \hat{C}(q,\dot{q})\dot{q} - \hat{G}(q) + r)ds - \dot{q}(0)) \tag{3.67}$$

We applied this control strategy to the Fanuc M-710iC/70 robot. To evaluate its performance, a disturbance torque was applied exclusively to the first joint of the robot as a random function. The following results were obtained, demonstrating the effectiveness of the proposed approach

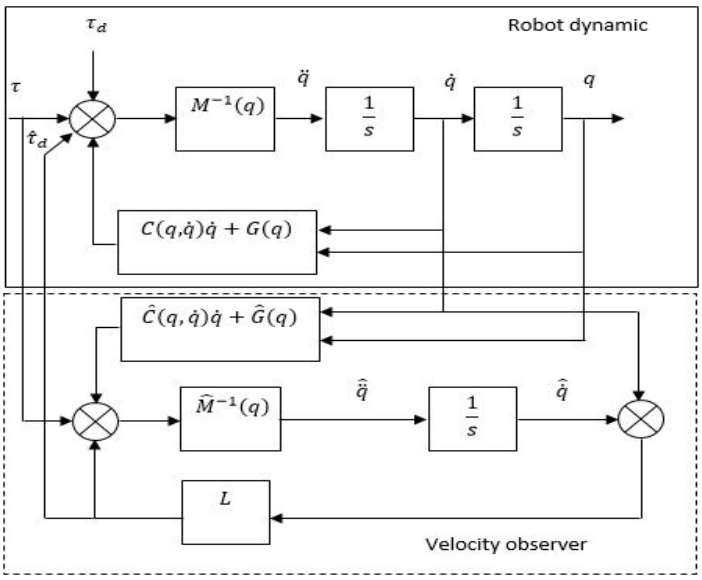
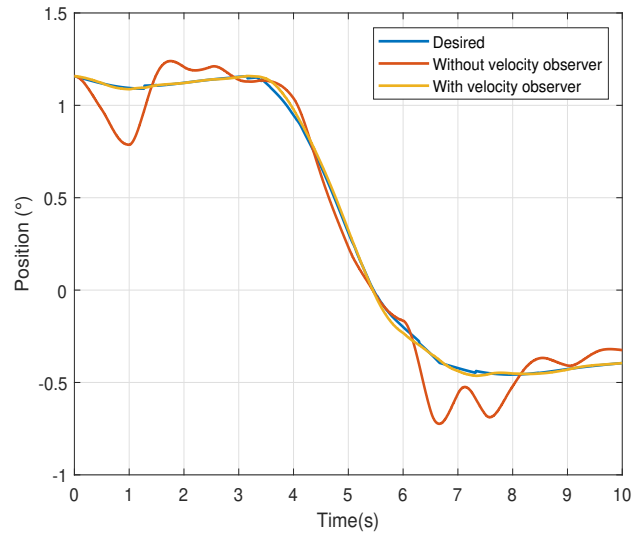


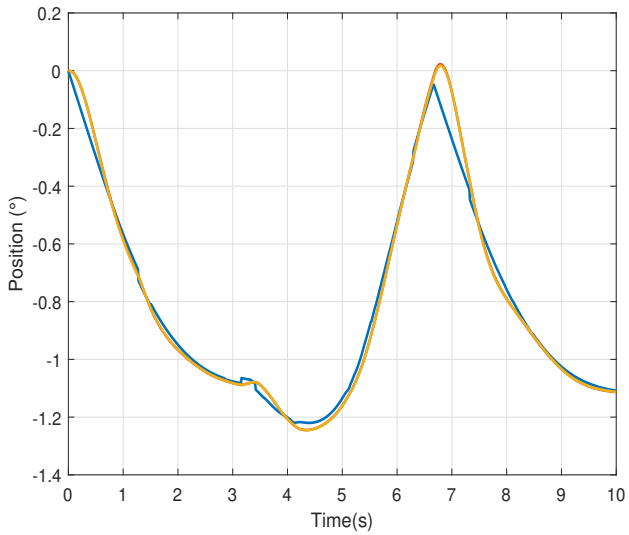

Figure 3.11: Control scheme based on velocity observer.

in handling disturbances and maintaining stability. Figure 3.12 illustrates the position of each joint of the robot. In the absence of the velocity observer, when the disturbance torque is applied to the first joint, the positions of the first, third, fourth, and sixth joints deviate from their desired positions. This indicates that these joints are dynamically coupled with the applied torque of the first joint in the robots dynamic equations. After integrating the observer, it successfully compensates for the disturbances, ensuring that the actual position closely tracks the desired position. The compensation is achieved by estimating the disturbance values and injecting them into the system input. Figure 3.13 illustrates the accuracy of the disturbance estimation by comparing the estimated disturbance with the actual disturbance applied to the system. The close alignment between these values indicates the effectiveness of the estimation method. Meanwhile, Figure 3.14 presents the error between the true disturbances and their estimated values. The error remains close to zero for most of the operation, except for transient phases where peak deviations occur. These peaks are likely caused by sudden changes in system dynamics or unmodeled effects. Addressing these transient errors through advanced filtering techniques or adaptive estimation methods could be a focus of future research..

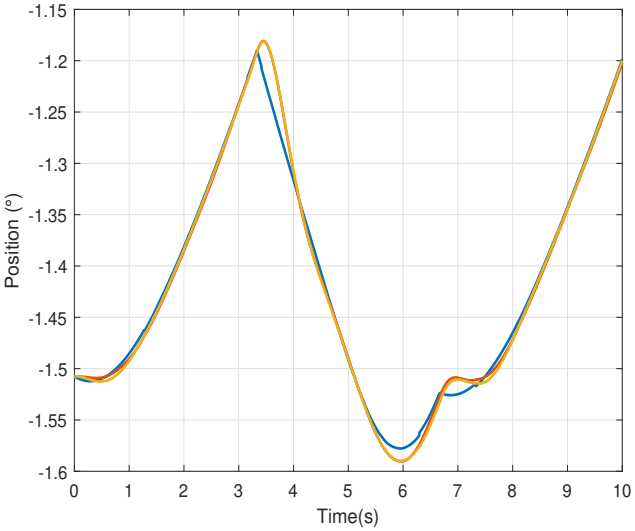
Figure 3.15 illustrates the trajectory of the end-effector in Cartesian space, highlighting the impact of the velocity observer on system performance. The trajectory closely follows the desired path, demonstrating the observers effectiveness in enhancing trajectory tracking accuracy. Additionally, the figure provides clear evidence of improved disturbance rejection, as deviations caused by external perturbations are minimized, allowing the system to maintain stability and precision in motion execution.



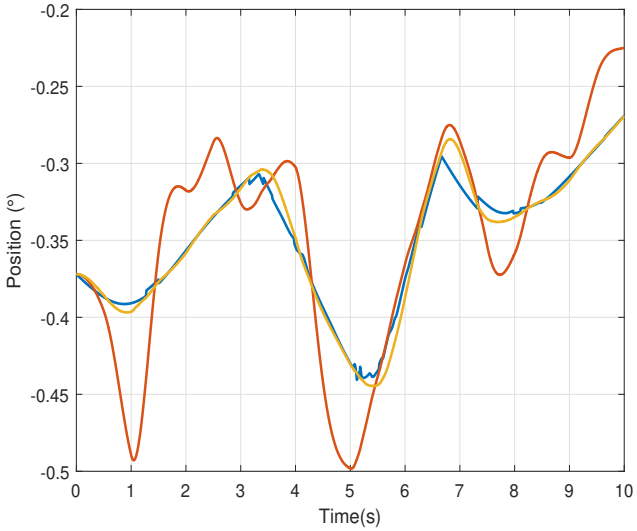

(a) Joint 1



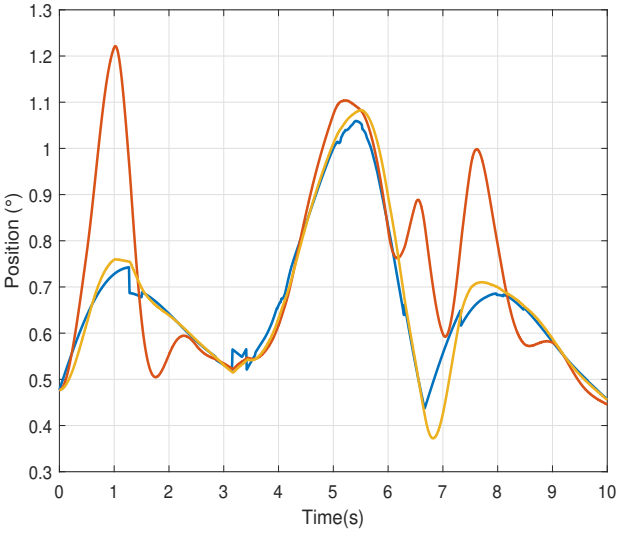

(b) Joint 2



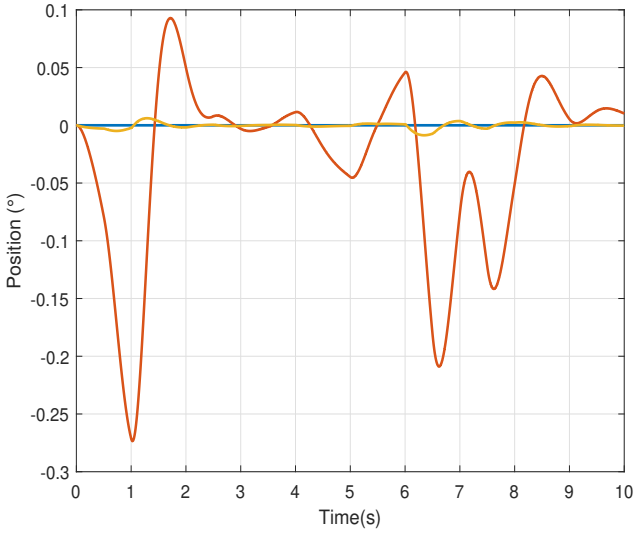

(c) Joint 3




(d) Joint 4




(e) Joint 5




(f) Joint 6

Figure 3.12: Position of the sixth joints.

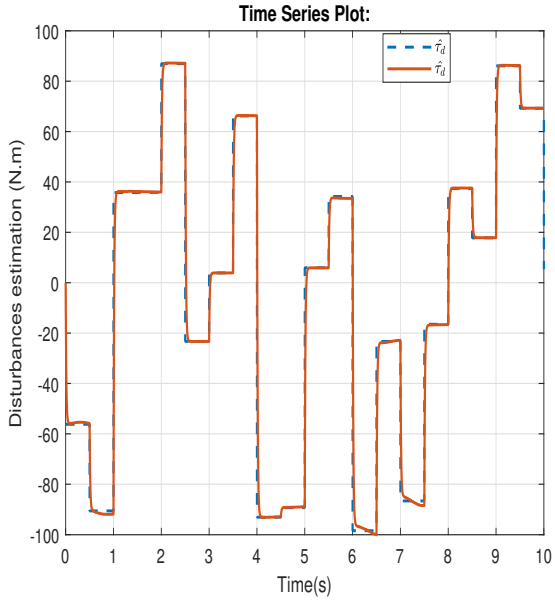



Figure 3.13: Estimation of disturbance.

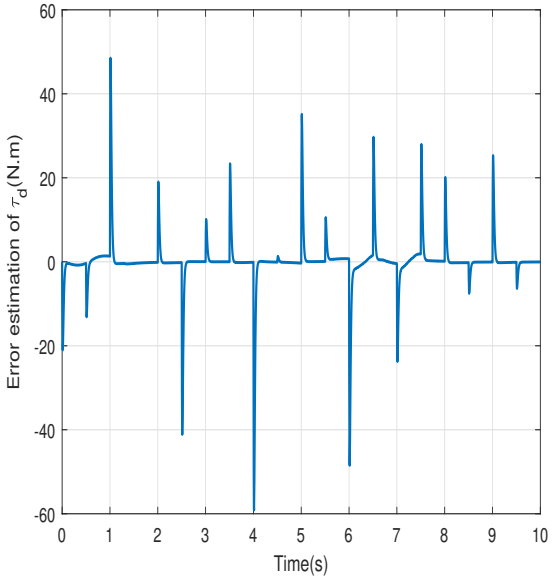



Figure 3.14: Estimation error.

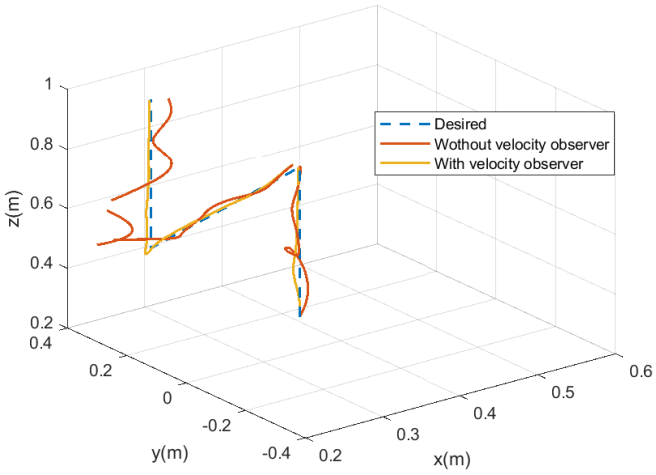



Figure 3.15: Trajectory in 3D space.

## 3.6 Conclusion

In this chapter, we presented the fundamental concepts of serial robot manipulators, covering different modeling approaches, including forward and inverse kinematics, as well as the dynamic model. To deepen our understanding of industrial robots, we conducted a case study on the Fanuc M-710iC/70 robot. The kinematic modelsboth forward and inverseof the Fanuc M-710iC/70 were analyzed to describe the relationship between joint space and Cartesian space. Furthermore, its dynamic model was developed using Simscape/Matlab, allowing for a more accurate simulation of its physical behavior. To enhance trajectory tracking performance, we implemented a control strategy based on computed torque control (CTC) integrated with a velocity observer. The velocity observer played a crucial role in estimating unmeasured velocities and compensating for disturbances, thereby improving the overall stability and accuracy of

the control system. The simulation results demonstrated the effectiveness of this approach in maintaining precise trajectory tracking under dynamic conditions. This study provides a solid foundation for understanding the modeling and control of industrial robots, which will serve as a basis for further research and improvements in advanced control strategies, disturbance rejection techniques and real-time implementation.

# Chapter 4

# Collision Detection for Industrial Robots Using Soft Sensors

Abstract

This chapter addresses the problem of collision detection for robot manipulators, presenting an improved model-based method that utilizes a momentum observer enhanced with fuzzy logic techniques. Following an introduction that reviews literature on model-based collision detection methods, Section 4.2 outlines the problem statement and describes the use of the generalized momentum observer in collision detection strategies. Section 4.3 details the design of the proposed improved observer, incorporating fuzzy logic to enhance its performance. Section 4.4 explains the methodology for collision monitoring, while Section 4.5 demonstrates the effectiveness of the proposed approach through simulation tests. Finally, Section 4.6 summarizes the chapter's findings and contributions.

## 4.1 Introduction

In recent years, the field of Physical Human-Robot Interaction (pHRI) has gained significant attention due to the increasing integration of robots in environments where they work closely alongside humans. This interaction is especially important in settings such as industrial manufacturing, healthcare, and service robotics, where robots collaborate with human workers to perform tasks more efficiently. However, one of the key challenges in pHRI is ensuring the safety of human workers during these interactions, particularly when unintended collisions between humans and robots occur. These collisions can lead to serious injuries, making it crucial to develop systems that can prevent such accidents and enable safe human-robot collaboration. Unintended collisions between humans and robots are an inevitable challenge in pHRI due to the complexity and unpredictability of human movement. While robots are programmed to follow precise movements, humans often act in ways that are unpredictable, which can lead to scenarios where a robot unintentionally makes contact with a human. Such collisions can cause a range of injuries, from minor bruises to more severe harm, depending on the type of robot

and its operation. The challenge, therefore, is to minimize the likelihood of these collisions and mitigate their potential harm.

Collision detection methods can be categorized into two types: model-based and model-independent methods. The model-independent category can further be divided into two subclasses: methods utilizing additional sensors and those that operate without them. The use of additional sensors, such as vision systems [203] and skin sensors [204], [205], allows robots to detect and estimate external forces caused by collisions. While effective, this approach increases system costs and adds complexity to the manufacturing process of robotic manipulators. To address these limitations, alternative methods based on learning techniques have been developed that do not require extra sensors. Examples include neural networks [206], [207], deep learning [18], support vector machines (SVM) [208]) and fuzzy systems [209], [210]. These techniques train robots to identify unintended collisions using training data, with proprioceptive sensor signals serving as inputs. The output is then analyzed to determine whether a collision has occurred. However, these methods have certain drawbacks. For instance, collecting training data in real-time can be challenging, and processing large datasets for collision detection can be time-consuming, limiting the practical application of such approaches.

On the other hand, model-based methods have emerged as a promising trend in robot collision detection due to their practical feasibility. Traditional model-based approaches [211], [212] rely on inverse dynamic models (IDM) to detect collisions by monitoring residual signals, which are the differences between measured and estimated joint torques. However, these methods require acceleration signal computations, leading to increased measurement noise and degraded performance. To address this limitation, a velocity observer was proposed [213] to eliminate the need for acceleration signals. This observer reduces the system's state dimension from $2n$ to $n$ (where $n$ represents the number of generalized coordinates), enabling faster response times. Nevertheless, this approach demands real-time computation of the inverse of the inertia matrix, which significantly increases computational costs. Further advancements include the development of a disturbance Kalman filter (DKF) [214] and an adaptive observer [215] designed to estimate actuator fault vectors while accommodating external disturbances within Takagi-Sugeno fuzzy systems. While these methods offer disturbance immunity, they suffer from reduced sensitivity due to inaccuracies in the system model.

To address the limitations of the previously discussed observers, a novel observer based on generalized momentum dynamics was introduced by [216]. Known as the Generalized Momentum Observer (GMO), this approach acts as a first-order filter, taking collision forces as input and producing a momentum residual as output, as explained [217]. Its straightforward design and reliable performance have made it a popular foundation for further advancements in collision detection for pHRI safety. Building on this framework, the work [218]) proposed a finite-time observer by integrating the sliding mode technique with the momentum observer. This approach ensures that the estimated external force converges to the actual external force within a finite time, enhancing the observer's responsiveness and accuracy. to reduce the chattering phenomenon caused by the observer mentioned [218], the authors in [219] introduced

a new sliding mode momentum observer (NSOMO). This improved observer employs a novel reaching law to effectively eliminate the chattering issue. To enhance the sensitivity of Generalized Momentum Observers (GMOs), [220] and [221] proposed an Extended State Observer (ESO) for the fast and robust detection of external forces. This observer operates as a second-order filter, offering improved performance. The ESO has also been applied in control systems [222], where its effectiveness has been demonstrated. However, a limitation of this observer is its use of a constant filter bandwidth, which leads to the peaking-value phenomenon during the initial phase of collision detection. A key issue with this type of observer is its use of a constant filter bandwidth, which leads to the peaking-value phenomenon during initial collision detection. Several attempts have been made to address the challenges associated with extended state observers. For instance,[223] introduced a generalized or higher-order ESO to handle sinusoidal disturbances in DC servo motors. Similarly, [224] proposed an Extended State Observer with a generalized integrator (GI-ESO) to monitor low-frequency or direct current (DC) disturbances in grid-connected converters. However, a limitation of these approaches is that the observer bandwidth is selected based on the disturbance frequency (whether larger or smaller), which poses challenges in practical applications, especially in robotic manipulators, where such frequency-related information is not readily available. In [225], a new concept is added to the extended state observer, is super-twisting algorithm it's allowing for precise state estimation and effective control even in the presence of uncertainties. It is true that this provides a good estimation, but the the peak value problem remains. Faced with this problem of an unavoidable trade-off between the collision sensitivity and the reduction of the peaking value, the authors proposed in [226]) a compromise solution by using nonlinear functions in the state observer design, this nonlinear generalized momentum observer (NGMO) works by providing appropriate bandwidth during collision detection observation. Although it can offer better performance, improving an extended momentum observer by applying nonlinear functions has some drawbacks, such as the selection of nonlinear functions is basically empirical. Additionally, there is difficulty in proving the stability of the observer due to the complexity of the nonlinear functions [227].

## 4.2 Problem statement

### 4.2.1 Preliminaries

The dynamic model of a robot manipulator is represented by dynamic equations that describe the relationship between the position, velocity and acceleration, and torque of the robot joints. Under Lagrange-Euler method, dynamic of $n$ link rigid robot manipulator can be established as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau + \tau_{ext} \tag{4.1}$$

where $q, \dot{q}$ and $\ddot{q} \in \mathbb{R}^{n\times 1}$ are the vectors of the joint position, velocity and acceleration, respectively. $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix , $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is the Coriolis and centrifugal forces

matrix, $G(q) \in \mathbb{R}^{n\times 1}$ is the gravity vector, $\tau \in \mathbb{R}^{n\times 1}$ is the torque vector applied to joints of manipulator and $\tau_{ext} \in \mathbb{R}^{n\times 1}$ represent external torque vector caused by the collision.

**Lemma 1.** *[228] The matrix $M(q) - 2C(q, \dot{q})$ is skew-symmetry, therefor it has the property*

$$\dot{M}(q) = C(q, \dot{q}) + C^T(q, \dot{q}) \tag{4.2}$$

where $C^T$ represent matrix transpose of $C$.

**Assumption 1.** *$\tau_{ext}$ is bounded, and thus $|\tau_{ext}| < \Upsilon$ , with $\Upsilon$ is unknown bounded value.*

### 4.2.2 Classical Generalized Momentum Observer

In the study by [217], a Generalized Momentum Observer (GMO) is used for robot collision detection. This approach is based on the generalized momentum equation and its time derivative, which are expressed as

$$\begin{cases} p = M(q)\dot{q} \\ \dot{p} = \dot{M}(q)\dot{q} + M(q)\ddot{q} \end{cases} \tag{4.3}$$

By combining (4.1), (4.2) and (4.3), a first-order dynamic equation for pp can be derived as

$$\dot{p} = \tau + \tau_{ext} + C^T(q, \dot{q})\dot{q} - G(q) \tag{4.4}$$

According to (4.4), the dynamics of the momentum observer is defined as

$$\begin{cases} \dot{\hat{p}} = \tau + C^T(q, \dot{q})\dot{q} - G(q) + r \\ \dot{r} = K(\dot{p} - \dot{\hat{p}}) \end{cases} \tag{4.5}$$

where $r \in \mathbb{R}^{n\times 1}$ represent the residual vector, $\hat{p} \in \mathbb{R}^{n\times 1}$ is the estimate value of momentum $p$ and $K = diag(k_i) > 0$ is the gain of observer.This observer works as a low pass filter, when a collision occurs, there is a deviation of the momentum, this deviation represents the residual value $r$. Increasing the values of $k_i$ leads to greater accuracy in the estimation of $\tau_{ext}$. But the other hand is limited by the system noise.

### 4.2.3 The Extended State Momentum Observer

To improve the overall performance of the GM observer, an extended state momentum observer was proposed in 2018), designed as follows:
The state space form of (4.4) is written as

$$\begin{cases} x_1 = p \\ \dot{x}_1 = \tau_p(\tau, q, \dot{q}) + \tau_{ext} \\ y = x_1 \end{cases} \tag{4.6}$$

with

$$\tau_p(\tau, q, \dot{q}) = \tau + C^T(q, \dot{q})\dot{q} - G(q) \tag{4.7}$$

When $\tau_{ext}$ is treated as an extended state, the state-space representation can be reformulated as

$$\begin{cases} x_1 = p \\ \dot{x}_1 = \tau_p(\tau, q, \dot{q}) + x_2 \\ \dot{x}_2 = \varphi(t) \\ y = x_1 \end{cases} \tag{4.8}$$

where $x_2$ is the extended state.

**Assumption 2.** *$\varphi(t)$ is an unknown bounded function $|\varphi(t)| \leq \Delta$,for a constant $\Delta$.*

An extended state momentum observer is designed to estimate $p$ and $\tau_{ext}$, and it is expressed as follows

$$\begin{cases} \dot{\hat{x}}_1 = \hat{x_2} - \beta_1 e + \tau_p \\ \dot{\hat{x}}_2 = -\beta_2 e \\ e = \hat{x}_1 - y \end{cases} \tag{4.9}$$

and

$$\begin{cases} \hat{x}_1 = \hat{p} \\ \hat{x}_2 = \hat{\tau}_{ext} \end{cases} \tag{4.10}$$

where $\beta_1$ and $\beta_2$ are a positive-definite diagonal gain matrices of observer, which can be obtained by the pole placement method as

$$(s + \omega_0)^2 = s^2 + \beta_1 s + \beta_2 \tag{4.11}$$

Here, $\omega_0$ represents the observer's bandwidth. This observer functions as a second-order filter. By selecting an appropriate value for $\omega_0$, the estimation error of the external torque can converge to zero in a shorter time compared to previous observers. However, this advantage comes with a drawback: the observer may produce a significant overshoot, which could affect the system's stability or precision if not properly addressed.

### 4.2.4 The Nonlinear Momentum Observer

To address the unavoidable trade-off between overshoot and steady-state error in the system, a compromise solution was proposed by Li et al. [221]. This approach involves incorporating nonlinear functions into the design of the momentum observer. By leveraging nonlinear dynamics, the observer achieves improved performance by reducing overshoot while maintaining a minimal steady-state error, offering a balanced and effective solution for precise collision detection and system stability.

The nonlinear momentum observer is defined as

$$\begin{cases} \dot{\hat{x}}_1 = x_2 - \beta_1 \psi_1(\mu, \delta, e) + \tau_p \\ \dot{\hat{x}}_2 = -\beta_2 \psi_2(\mu, \delta, e) \\ e = \hat{x}_1 - y \end{cases} \tag{4.12}$$

where $\psi_i(\mu, \delta, e)$ is a nonlinear function which can be takes a several forms, $\mu$ and $\delta$ are the shaping parameters of a non-linear function. Although the nonlinear observer has a better performance compared to previous observers. The problem with the design of this observer is the selection of the form of the nonlinear function, which is basically empirical, and the difficulty in proving the stability of the observer due to the complexity of the nonlinear functions.

## 4.3 Fuzzy Generalized Momentum Observer Design

To develop an efficient solution that enhances the performance of the Generalized Momentum Observer (GMO) while avoiding the complexity associated with nonlinear functions and intricate theoretical analysis, we propose a novel Fuzzy Generalized Momentum Observer (FGMO). This observer integrates an Extended State Observer (ESO) with a fuzzy system, where the fuzzy system intelligently adjusts the observer's bandwidth during the observation process.

This section describes the design of the fuzzy generalized momentum observer (FGMO), this type of observer incorporates the extended state observer (ESO) and fuzzy logic. The ESO observer is expressed as follows

$$\begin{cases} \dot{\hat{x}}_1 = \hat{x}_2 - \beta_1 e + \tau_p \\ \dot{\hat{x}}_2 = -\beta_2 e \\ e = \hat{x}_1 - y \end{cases} \tag{4.13}$$

From (4.13), the transfer function between the estimated external torque and input signal can be obtained as

$$\begin{cases} \hat{x}_1 = \frac{\hat{x}_2 + \tau_p - \beta_1 p}{s + \beta_1} \\ \hat{x}_2 = \frac{\beta_2 sp - \beta_2 \tau_p}{s^2 + \beta_1 s + \beta_2} \end{cases} \tag{4.14}$$

From (4.14),it can be see that the transfer function represent a critically damping second-order system. The system's damping ratio and bandwidth can be expressed as

$$\begin{cases} \omega_0 = \sqrt{\beta_2} \\ \zeta = \frac{\beta_1}{2\sqrt{\beta_2}} \end{cases} \tag{4.15}$$

where $\beta_1$ and $\beta_2$ are time varying gains of observer tuned by fuzzy system

$$\beta_1 = \begin{pmatrix} 2\zeta\omega_{0,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 2\zeta\omega_{0,n} \end{pmatrix}, \beta_2 = \begin{pmatrix} \omega_{0,1}^2 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \omega_{0,n}^2 \end{pmatrix} \tag{4.16}$$

From (4.15), it can be observed that there is a trade-off between the damping ratio and the observer's bandwidth. To enhance tracking accuracy, an appropriate value of $\omega_0$ must be selected. However, when the observer operates with a critical damping ratio, the collision signal it detects may produce overshoots, leading to false alarms. To address this, the damping ratio can be increased during the initial moments of operation to eliminate overshoot. This trade-off highlights the conflicting requirements between overshoot elimination and precise tracking, necessitating a balanced solution. To achieve this, we propose time-varying observer parameters controlled dynamically during the manipulator robot's operation, implemented through a fuzzy system.

A single-input, single-output fuzzy system is applied to reduce the computational time, where the external torque estimation error represents the input of this system, and the observer bandwidth is applied as the output of the fuzzy system. This fuzzy system is constructed according to the following steps:

*Step 1*: We define fuzzy sets for input fuzzy system estimation error of external torque $e$ and output fuzzy system $\omega_{0,i}^*$ , for the input, a three membership functions are designed, negative large (NL),small (S) and positive large (PL), $e^i \in \{ NL^i, S^i, PL^i\}, (i = 1, 2 \dots, n)$, and two membership functions are designed for the output, small (S) and large (L), $\omega_{0,i} \in \{ S^i, L^i\}$.

*Step 2*: Develop a set of if-then rules that specify how the inputs influence the outputs. The rules should be based on the knowledge of the system and its behavior and should consider the relationships between the input and output variables. In this context, to provide proper performance of the observer FGMO, the IF-THEN fuzzy group is constructed according to the following rules:

IF $e$ is NL, THEN $\omega_0$ is L
IF $e$ is S, THEN $\omega_0$ is S
IF $e$ is PL, THEN $\omega_0$ is L

*Step 3:* Because the input and output of this system are crisp values, an interface between the fuzzy values and the real values is needed. These interfaces are *fuzzifier* and *defuzzifier*. The fuzzifier is the tool that transforms a real input and output value into a corresponding fuzzy value. Here, we propose a Gaussian fuzzifier, which has the following form

$$\mu_X(e) = exp^{-\frac{(c-e)^2}{2\sigma^2}} \tag{4.17}$$

where $c$ and $\sigma$ are the center and width of the fuzzy set $X$. Whereas the defuzzifier is defined as a mapping from fuzzy value to crisp value. The center of gravity defuzzifier is applied through following equation

$$\omega_0^* = \frac{\int \omega_0 \mu(\omega_0) d\omega_0}{\int \mu(\omega_0) d\omega_0} \tag{4.18}$$

### 4.3.1 Dynamic Error of The Observer

Based on the (4.8) and (4.13), the dynamic error of observer can be written as

$$\begin{cases} \dot{e}_1 = e_2 - \beta_1 e_1 \\ \dot{e}_2 = -\beta_2 e_1 - \varphi(t) \end{cases} \tag{4.19}$$

The convergence analysis of observer is mainly inspired by [227]. The dynamic error of (4.19) can be rewritten as

$$\dot{E} = AE + B(-\varphi(t)) \tag{4.20}$$

where

$$A = \begin{pmatrix} -\beta_1 & 1 \\ -\beta_2 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.21}$$

**Assumption 3.** *The solution of systems states $x_i$ and the input $\tau_p$ are bounded.*

**Theorem 2.** *Consider that the above assumptions are met,then, the estimation error (E) ultimately converges into the following bounded ball*

$$\mathcal{B} = \left\{ E : \|E\| \leq \frac{2\Delta\lambda_{max}(P)}{\lambda} \right\} \tag{4.22}$$

*Proof.* Let a Lyapunov function $V$ for the dynamic error (4.20) is defined as

$$V = E^T P E \tag{4.23}$$

where $P$ is a positive definite matrix, which satisfies

$$A^T P + PA = -\lambda I \tag{4.24}$$

The time derivative of $V$ is expressed as

$$\dot{V} = E^T (A^T P + PA) E + 2E^T P B(-\varphi(t)) \tag{4.25}$$

Using (4.24) and Assumption (2),$\dot{V}$ becomes

$$\begin{aligned} \dot{V} &\leq -\lambda \|E\|^2 + 2\Delta\lambda_{max}(P)\|E\| \\ &\leq -\|E\|(\lambda\|E\| - 2\Delta\lambda_{max}(P)) \end{aligned} \tag{4.26}$$

where $\lambda_{max}$ are maximum eigenvalue of $P$, and $\lambda$ is positive, then $E$ ultimately converges into

the $\|E\| \leq \frac{2\Delta\lambda_{max}(P)}{\lambda}$ bounded ball. □

## 4.4 Collision Monitoring Method

To enhance the sensitivity of collision detection, it is essential to define an appropriate threshold that effectively differentiates between external collision moments and modeling errors. This threshold is represented by an envelope region, bounded by upper and lower values, which is designed to encompass all non-collisional disturbance torques arising from system model inaccuracies, friction between joint actuators, and other factors.
To determine the collision threshold, the robot performs a trajectory execution in free space, without any external impacts (see Figure 4.1). During this collision-free motion, the residuals produced by the Fuzzy Generalized Momentum Observer (FGMO) reflect the disturbance torque signals $\tau_{dis}$ at each robot joint. The maximum observed disturbance torque at each joint is then used to define the collision threshold, which can be expressed as follows

$$|\rho_i| \leq max|\tau_{dis,i}| \tag{4.27}$$

In the absence of collision, the residual $r$ generated by the FGMO contains only the perturbing torque, that is

$$|r_i| \leq |\rho_i| \tag{4.28}$$

After that, the robot can be run in workspace. When a collision occurs, the residual value also includes the external torque, causing the residual value to increase sharply and exceed the threshold and can be expressed as

$$|r_i| > |\rho_i| \tag{4.29}$$

To distinguish the external collision from the non-collisional disturbance torque, a collision detection index is defined as follows

$$\epsilon_i = \begin{cases} 1 & \text{if } |r_i| > |\rho_i| \\ 0 & \text{if } |r_i| \leq |\rho_i| \end{cases} \tag{4.30}$$

## 4.5 Simulations Results and Discussion

In this section, we present a simulation to evaluate the performance and validate the effectiveness of the proposed collision detection algorithm. The simulation is conducted using the industrial robot model Fanuc M710iC/70, which was thoroughly analyzed in the previous chapter.

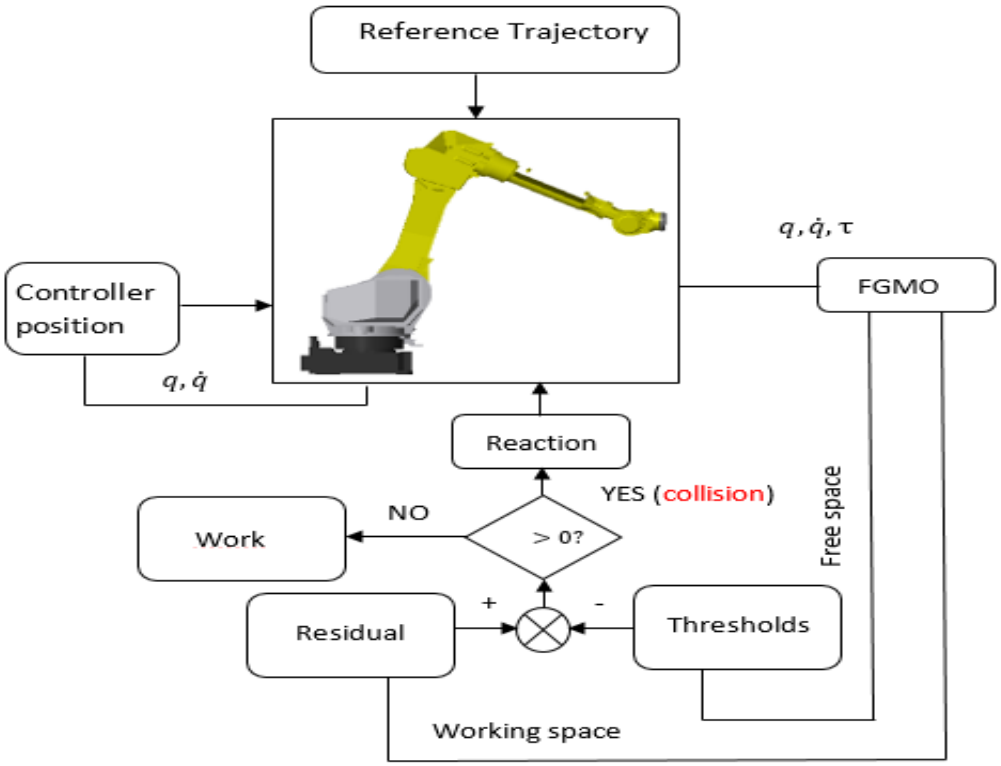



Figure 4.1: Collision detection workflow.

### 4.5.1 Collision Description

Unwanted collisions can be characterized as abrupt disturbances, such as vibrations in the actuators, which lead to torque deviations. To simulate a collision between the robot and a human (or the environment), a collision force is applied to the third joint of the robot, as illustrated in Figure 4.2. Specifically, a collision force Fc is introduced at 3 seconds into the motion trajectory and remains active for a duration of 1 second.

A computed torque control strategy is used to control the robot, which meets the requirements of controlling the trajectory of the manipulator robot. To test the performance of the observers presented above, it is necessary to know how to adjust the parameters of the observer. For the GMO, it has a $K$ gain, which is the only parameter to adjust, To obtain the highest possible estimation accuracy, $K$ must be increased . However, increasing the gain $K$ leads to overshoot peaks. For ESO and NGMO, the gains $\beta_1$ and $\beta_2$ are the parameters that must be adjusted. Since Li et al. [221], it is known that $\beta_2$ determines the speed of evolution of the estimates, while the increasing of $\beta_1$, cause overshoot peaks and a low enough value will reduce convergence, while the nonlinear function $\psi_i(\mu, \delta, e)$ is defined as

$$\psi_i(\mu, \delta, e) = \begin{cases} |e|^{\mu} sign(e), & |e| > \delta \\ \dfrac{e}{\delta^{1-\mu}}, & |e| \leq \delta \end{cases} \tag{4.31}$$

The parameter of three observers can be obtained based on the parameter analysis of observers. (as shown in Table 4.1). Based on the estimated external torque results, the peak value differs

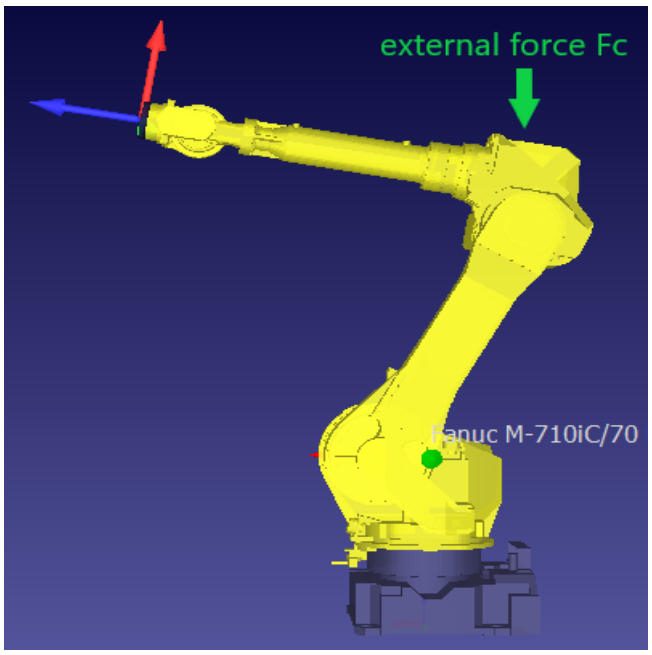



Figure 4.2: Collision modelling.

Table 4.1: Observers parameters

| Observer | Parameters |
|---|---|
| GMO | $K_i = 20$ |
| EGMO | $\beta_{1,i} = 20, \beta_{2,i} = 100$ |
| NGMO | $\beta_{1,i} = 20, \beta_{2,i} = 100,$ $\mu_1 = 1, \mu_2 = 0.5, \delta_1 = \delta_2 = 0.1$ |

from joint to joint, and it comes back to the mass of the joint. In the first case, the estimation using GMO, as shown in Figures 4.3a-4.3c with the red graph, the result of the peak value of the first joint is -0.28 while for the second and third joints, the peak values are 11.41 and 5.97 respectively. This comes back to the joints mass of the robot, which weight of first, second and third joint of robot are 170.31 kg ,63.73 kg and 98.97 kg respectively. We can conclude that the mass of the joint affects the dynamics of the observer on its gain, which is responsible for the peak phenomenon.

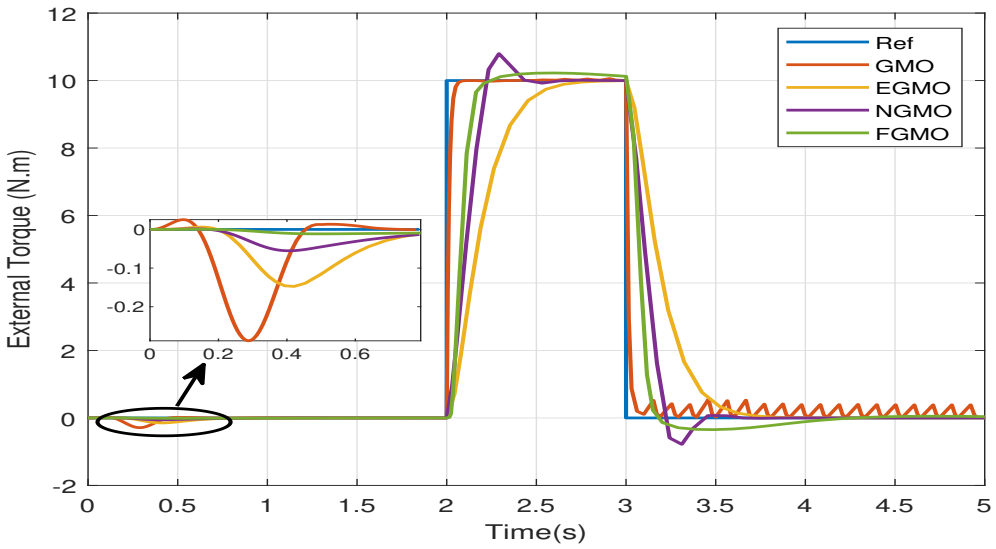
As mentioned in the discussion in the introduction, the objective of this work is to reduce the peak value and maintain the sensitivity of the collision estimation, as shown in the figures. Regarding the peak phenomenon in the initial time. In Figure 4.3a, all observers show an obvious peak, except for the FGMO observer which almost completely eliminates the peak. For the second joint, as shown in Figure 4.3b, a significant peak is obtained, especially in the case of GMO, which reaches 11.41, and which FGMO was able to reduce to 1.31, a reduction of 88%. In Figure 4.3a, for the third joint, the FGMO observer generates a lower peak value than the other observers, which has an 86% reduction in the GMO peak value.In the second phase, after the initial time phase, the observer must vary the bandwidth so as to be able to react quickly to changes in the system's state (appearance of a collision). And a wider bandwidth

allows it to track changes more sensitively and accurately. However, increasing the bandwidth of an observer also increases the systems overshoot as shown in Figures 4.3a, 4.3b and 4.3c at time 2.3 s, it can be found the nonlinear observer NGMO has an obvious overshoot, while the FGMO follows the state change smoothly. Therefore, it is important to balance the benefits of a higher bandwidth with the risks of overshoot, and design the observer carefully to ensure optimal performance.

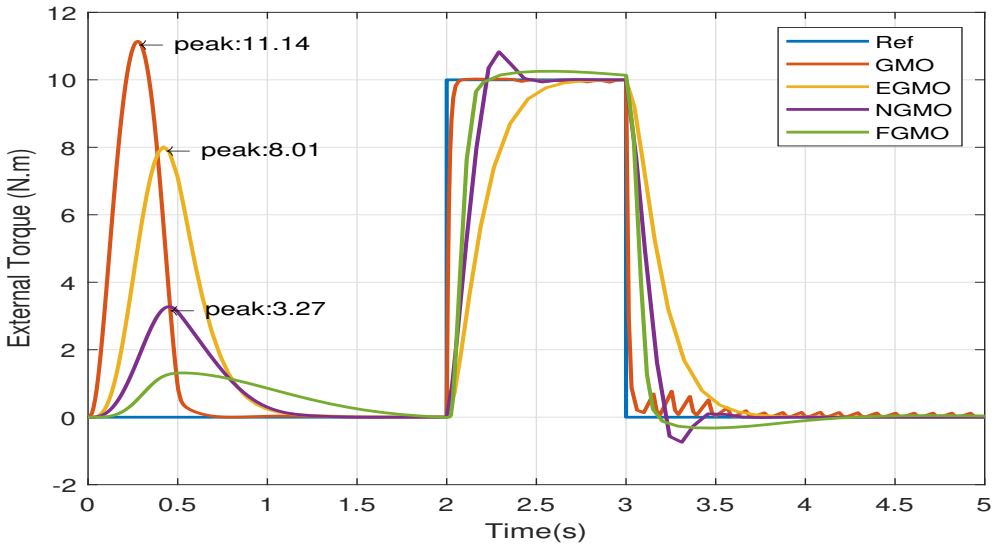
From the Figures 4.4a, 4.4b and 4.4c, whenever the torque estimation error is nonzero, it attempts to converge to zero as quickly as possible. To obtain better transient performances, it is necessary to define an adequate $\omega_0$. However, according of the (4.15), the observer may detect the external force with a critical damping ratio that will lead to an undesirable overshoot phenomenon. It can be seen that system overshoot and response time have opposite requirements for the damping ratio. For GMO, EGMO and NGMO observers, when bandwidth is constant, it is difficult to find a compromise between overshoot and response time. Unlike the observers mentioned above, the FGMO observer can provide time-varying bandwidth, as shown in Figure 4.5, which present the observer bandwidth $\omega_0$ for the three first joints. Comparing the figures for the estimation error and the bandwidth evolution, it can see that when the estimation error is far from zero, the bandwidth is as wide as possible so that the estimation error converges more quickly to zero and, consequently, to obtain the best sensitivity to collisions, then decreases rapidly to avoid overshoot and also to obtain the shortest transit time. Table 4.2 summarizes the characteristics of these moment observers, and we classify the overall performance of each momentum observer in terms of complexity, sensitivity, initial peak, need for joint acceleration, need for inverse of inertia matrix or not (* = " low " to **** = " high " It can be seen that the designed FGMO observer shows high performance in terms of reduced complexity, high sensitivity and low initial peak value compared to the other momentum observers used.

Table 4.2: Collision monitoring methods using momentum observers

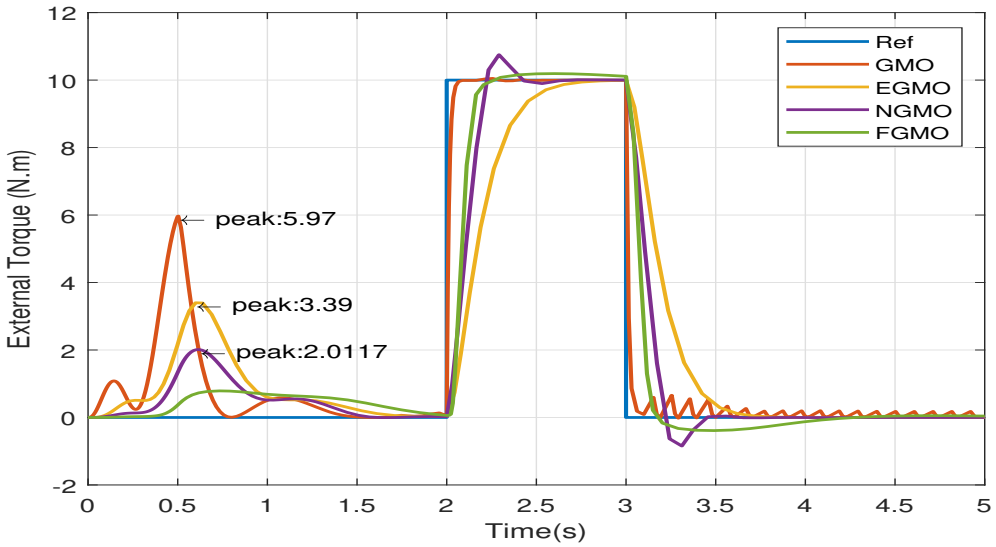
| Observer | Characteristics of the momentum observers |
|---|---|
| GMO | Complexity *<br>Sensitivity ***<br>Initial peaking value ****<br>$\ddot{q}$ not needed Only $M(q)$, no inverse |
| EGMO | Complexity **<br>Sensitivity *<br>Initial peaking value ***<br>$\ddot{q}$ not needed Only $M(q)$, no inverse |
| NGMO | Complexity ****<br>Sensitivity **<br>Initial peaking value ***<br>$\ddot{q}$ not needed Only $M(q)$, no inverse |
| FGMO | Complexity **<br>Sensitivity ***<br>Initial peaking value *<br>$\ddot{q}$ not needed Only $M(q)$, no inverse |




(a)

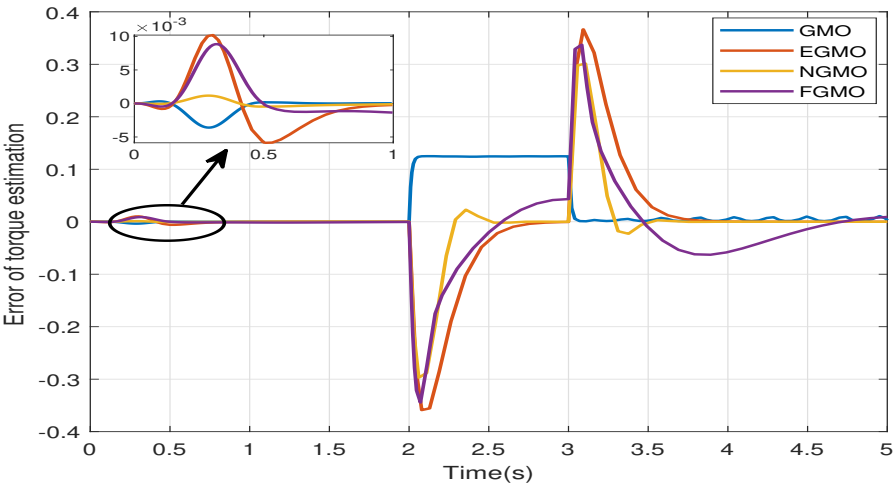



(b)

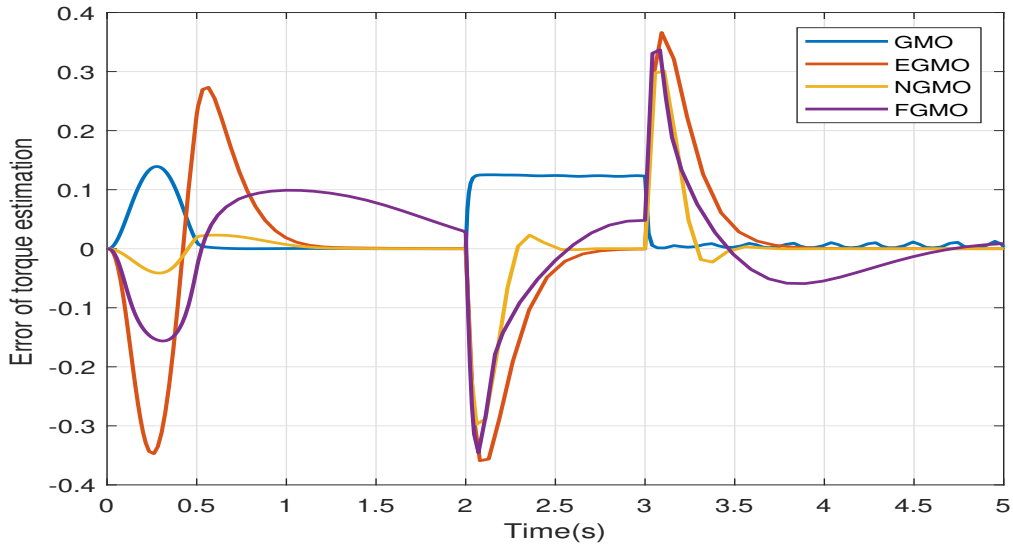



(c)

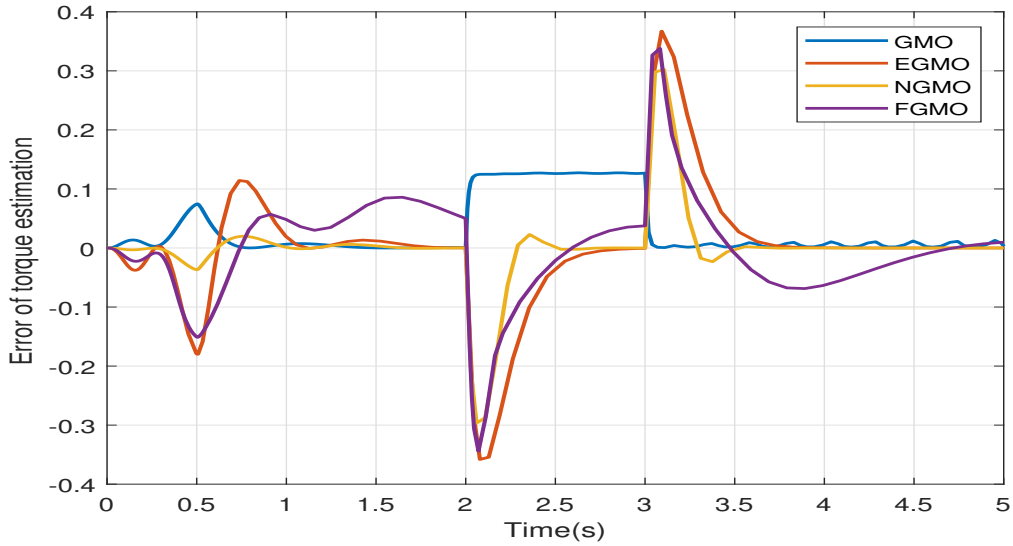
Figure 4.3: Residual of the first three joints. (a) for first link. (b) for second link. (c) for third link.




(a)




(b)




(c)

Figure 4.4: Estimation error of the first three joints. (a) for first link. (b) for second link. (c) for third link.

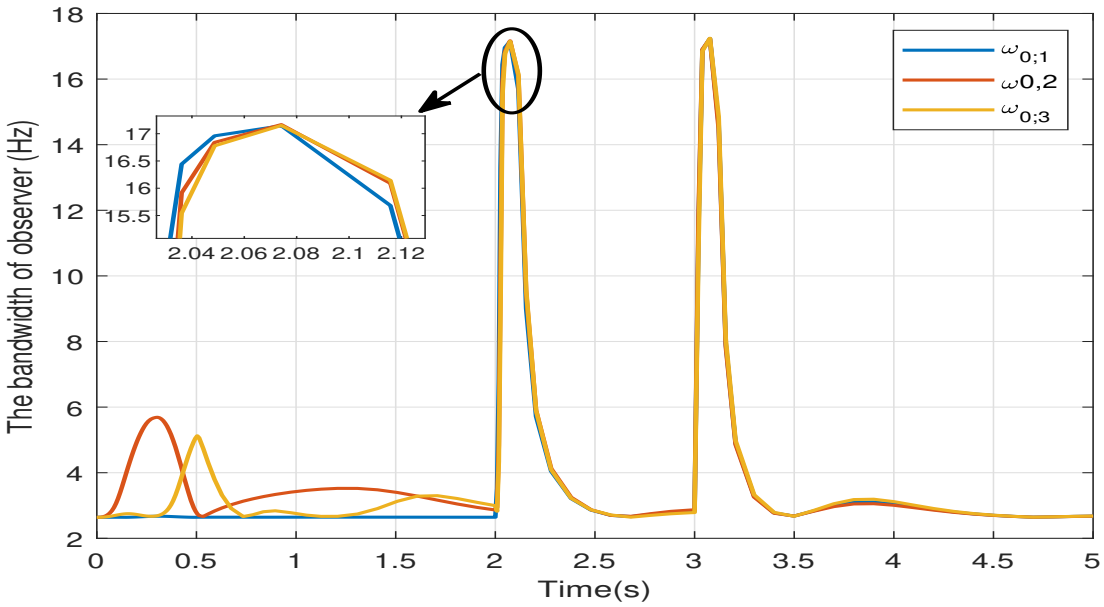



Figure 4.5: Evolution of observer's bandwidth

### 4.5.2 Detection and Localization of Collision

The detection step must decide whether the robot manipulator is in a normal operating state or not. For this purpose, we adopt a threshold to generate a collision detection indicator from the monitoring signal based on (4.30). While collision localization consists of identifying the robot links that are affected by a collision. When a collision force is applied on $i^{th}(1 \leq i \leq n)$ link of the serial robot manipulator leads to [219].

$$\begin{cases} r_1, \ldots, r_i \neq 0 \\ r_{i+1}, \ldots, r_n = 0 \end{cases} , i = 1, 2, \ldots, n \tag{4.32}$$

we consider for example that the robot manipulator affected by a collision on third link (see figure 4.2), therefore the collision force is distributed to second and first link of the robot. To simulate this scenario, we apply an external torque to the first three joints of the manipulator robot at different times while it executes the desired trajectory.Simulation results are depicted in Figures 4.6 and 4.7, the designed collision threshold is large enough to account for disturbances and modeling errors. Therefore, the collision threshold can distinguish real collisions from non-collisional disturbance signals.

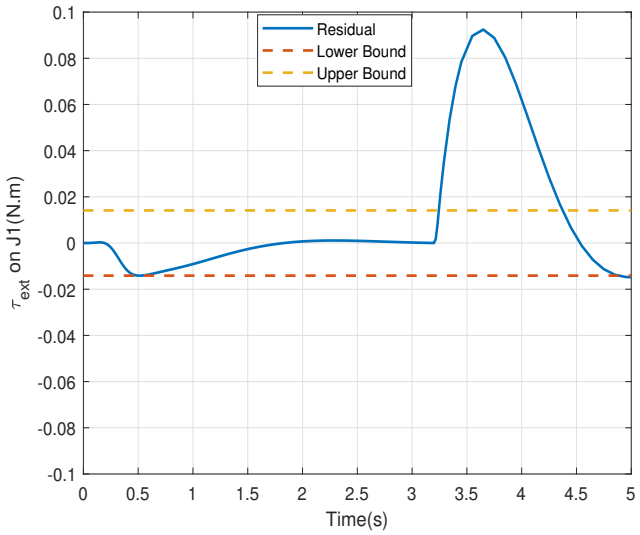
In the start-up phase, despite the appearance of a peak in the initial time, which does not exceeding the collision threshold, it is therefore represented as a non-collisional disturbance signal as long as it remains within the envelope region of collision threshold. However, collision detection can be done by comparing the residuals to the detection threshold $\rho$.

From the figure 4.6, the residuals stay inside threshold region until the moment of 3 s which the external torque estimated by the observer changes abruptly and exceeds the collision threshold for the first joint, and at the moments 3.1 s and 3.2 s for the second and third joints respec-

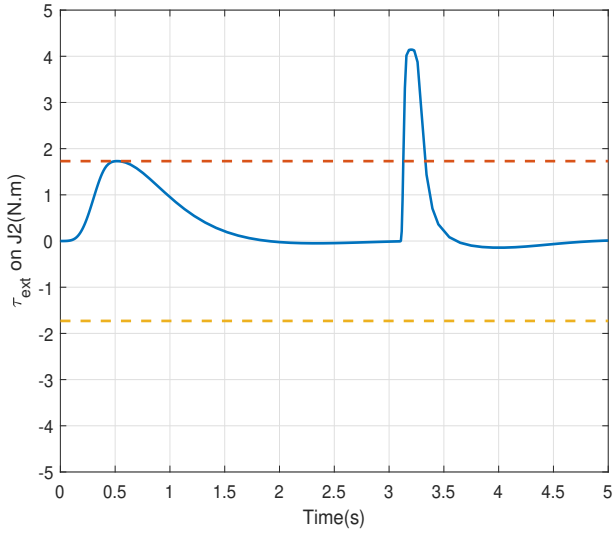
tively. Therefor there is a collision at link three according to (4.32). While the residual signal of joints 4, 5 and 6 remains at zero, which means that there is no collision at these locations of the robot.
when the collision occurs, the torque estimation error for the first three joints is around -0.5 and quickly returns to zero, demonstrating the sensitivity of the FGMO observer in estimating external torque (see Figures 4.7a-4.7c). While the error remains zero for the three second joints of the robot as shown in the Figures 4.7d-4.7f, which can be translated by these joints not affected by the collision.
Once the collision has been detected, it must be localized. This localization is carried out from the signature table which based on the (4.32). The table of signatures is given in the table 4.3. In this example, when the collision has occurred at the third link, according to this table, the residual vector $r$ associated with the collision is $r = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T$.
Figure 4.8 summarizes the key steps involved in collision detection and localization. The Fuzzy-Gain Momentum Observer (FGMO) generates residual signals that are highly sensitive to various collision events. By comparing these residual signals with predefined detection thresholds, a detection index is obtained, indicating the presence of a collision. Furthermore, by matching this detection index against a signature table, the system can accurately localize the point of impact. This approach ensures reliable collision detection while enabling precise identification of the affected joint or region of the robot.

Table 4.3: Signature table

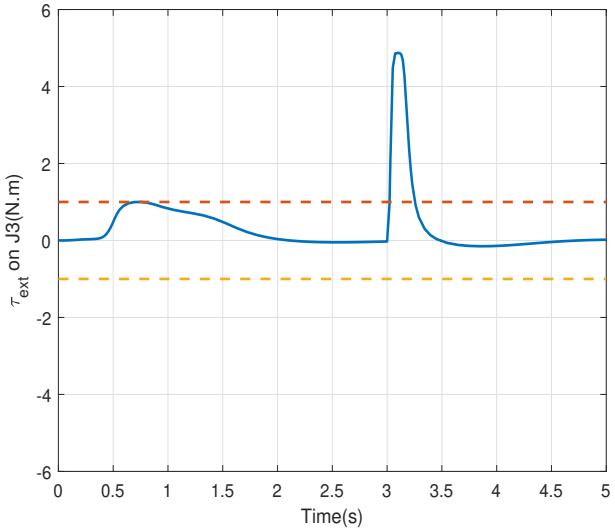
| Link | 1st | 2nd | 3th | 4th | 5th | 6th |
|---|---|---|---|---|---|---|
| $r_1$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $r_2$ | 0 | 1 | 1 | 1 | 1 | 1 |
| $r_3$ | 0 | 0 | 1 | 1 | 1 | 1 |
| $r_4$ | 0 | 0 | 0 | 1 | 1 | 1 |
| $r_5$ | 0 | 0 | 0 | 0 | 1 | 1 |
| $r_6$ | 0 | 0 | 0 | 0 | 0 | 1 |



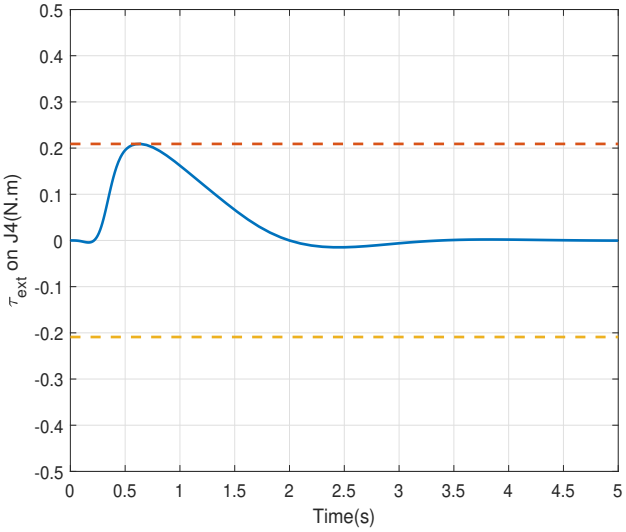

(a) Joint 1



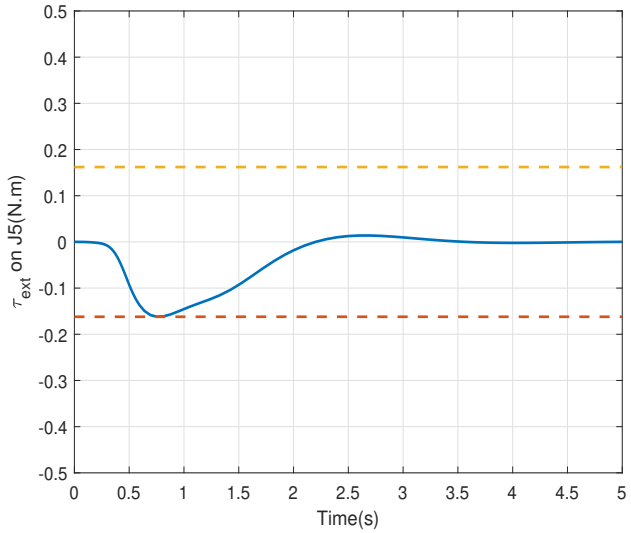

(b) Joint 2



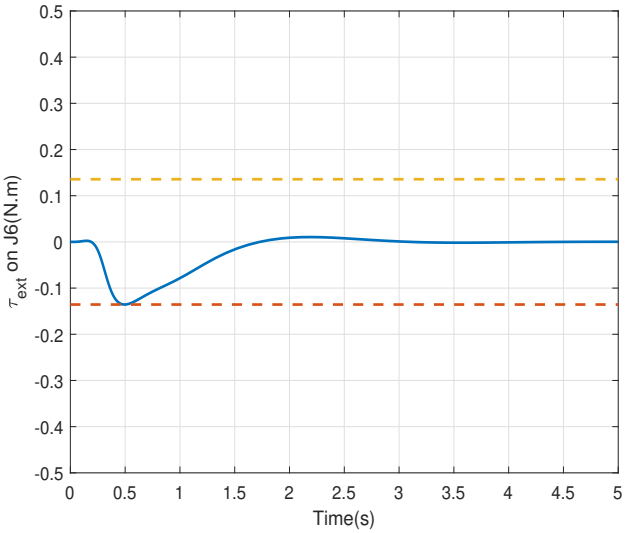

(c) Joint 3




(d) Joint 4

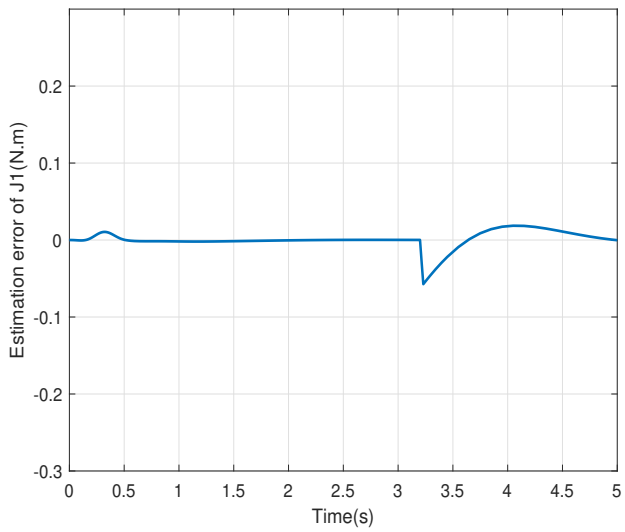



(e) Joint 5

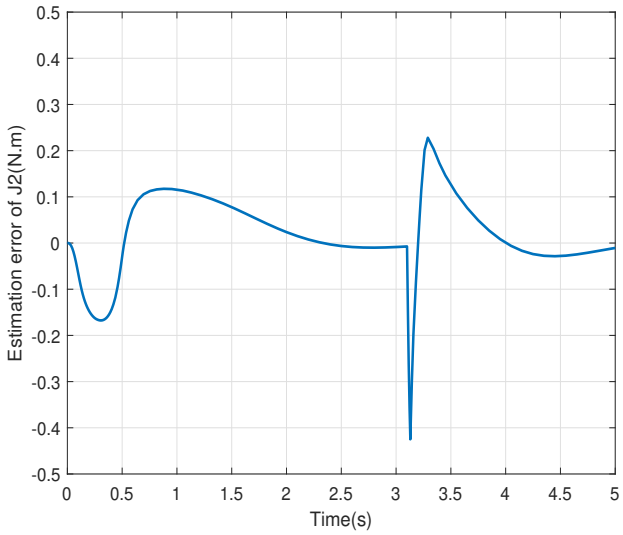



(f) Joint 6

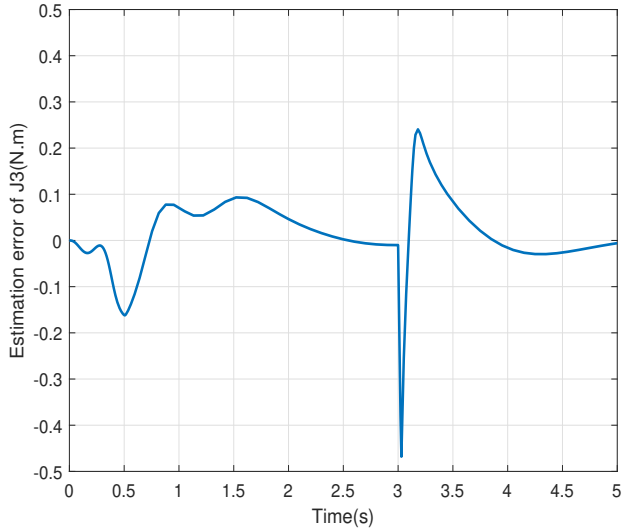
Figure 4.6: Residual of the first three joints. (a) for first link. (b) for second link. (c) for third link. And Estimation error of the first three joints. (d) for first link. (e) for second link. (f) for third link.



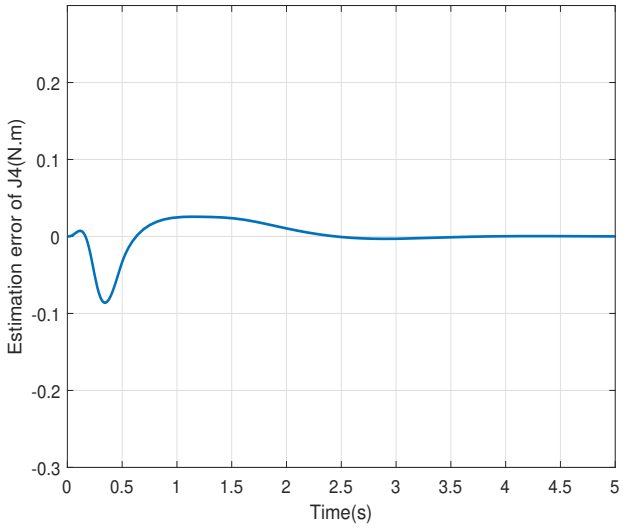

(a)



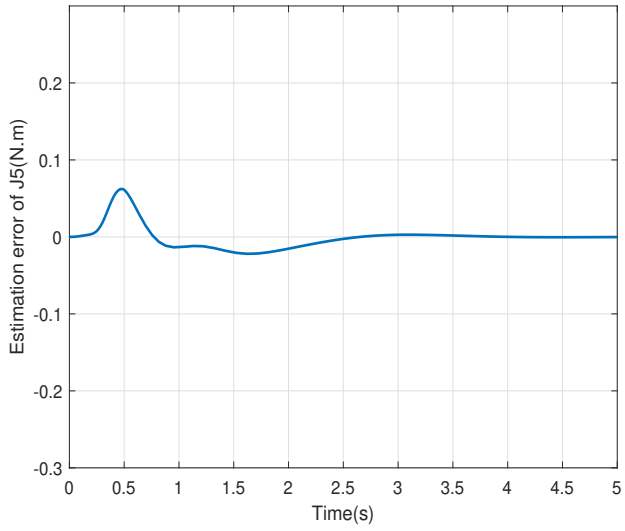

(b)



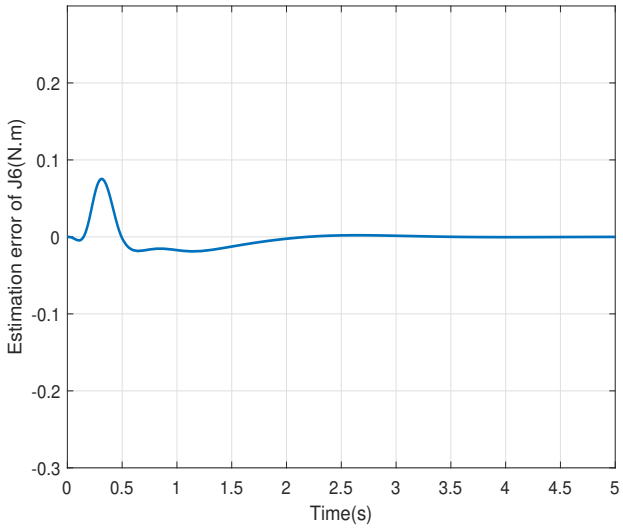

(c)




(d)




(e)




(f)

Figure 4.7: Estimation error. (a) for first link. (b) for second link. (c) for third link. (d) for fourth link. (e) for fifth link and (f) for sixth link.

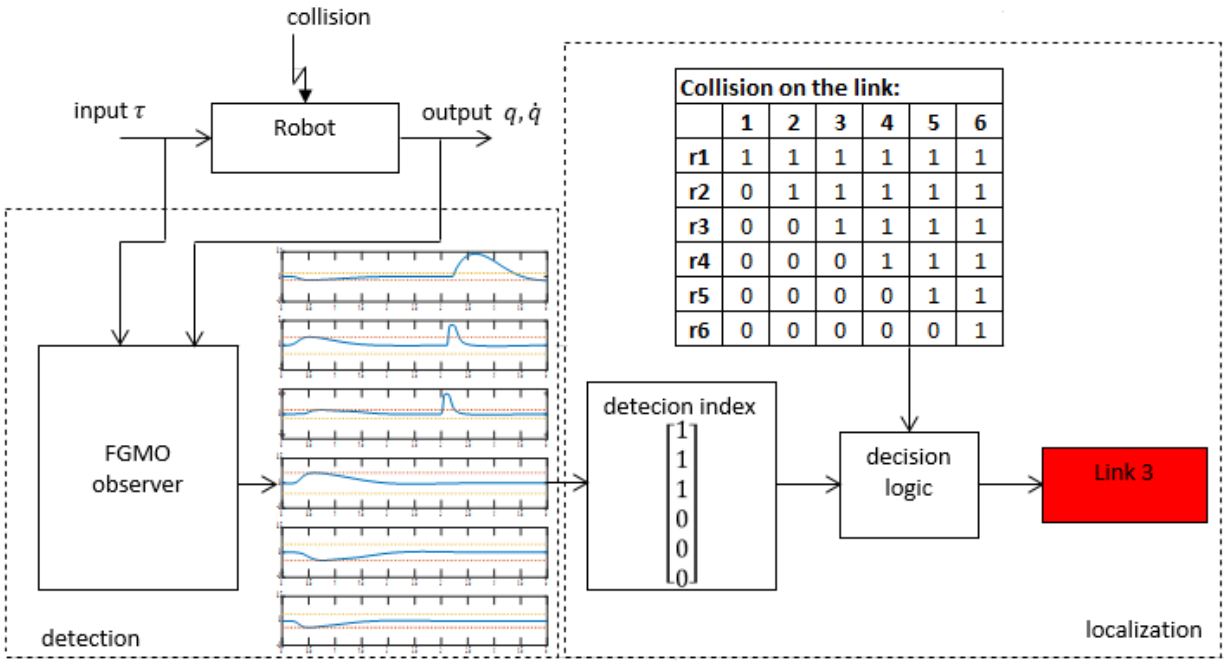



Figure 4.8: Collision detection and localization.

## 4.6 Conclusion

This chapter presented an alternative collision detection method for manipulator robots operating in sensorless conditions, utilizing a Fuzzy Generalized Momentum Observer (FGMO). Building upon traditional generalized momentum observers, the FGMO incorporates a fuzzy system to intelligently adjust the observer's bandwidth. This innovative approach effectively addresses the trade-off between collision sensitivity and peak value reduction, which is a limitation in conventional methods.

The integration of the FGMO with a collision threshold enables the distinction between external forces and internal torque disturbances. Furthermore, the collision detection algorithm can accurately identify the specific robot link where the collision occurs. The results demonstrate that this solution is robust and suitable for detecting both dynamic and quasi-static collisions in robot manipulators.

For future work, the proposed algorithm could be enhanced to extract additional information from the residual vector, such as the magnitude of the collision force, allowing the robot to respond appropriately. Further development will focus on adapting the algorithm to incorporate dynamic thresholds and integrating it into real-world applications with robotic manipulator arms.

# Chapter 5

# Soft Computing Approaches for Optimal Industrial Robot Trajectory Planning

Abstract

The present chapter addresses the problem of optimal trajectory planning for industrial robots (IRs), with a particular focus on minimizing energy consumption. An effective approach is presented to obtain optimal trajectories in terms of time, jerk, and energy consumption. Following an introduction that reviews recent literature on the topic, Section 5.2 defines the problem statement. Section 5.3 introduces the artificial LSTM model used for energy consumption modeling. Section 5.4 outlines the optimization strategy for achieving optimal trajectories. Section 5.5 validates the proposed approach through simulation experiments, while Section 5.6 highlights the chapter's key findings and contributions.

## 5.1 Introduction

The integration of robot manipulators into assembly and manufacturing lines is steadily increasing due to their efficiency, flexibility, and safety [229]. However, there is a growing emphasis on enhancing the motion performance of industrial robots. Key objectives in trajectory planning include minimizing execution time, reducing jerk and optimizing energy consumption. Trajectory planning is a critical operation for industrial robots before they are deployed, as it plays an important role in enhancing productivity. An optimal trajectory minimizes unnecessary movements and cycle times. Additionally, well-planned trajectories result in smoother motion profiles, reducing jerky movements and vibrations that can cause wear and tear on mechanical components. Optimized trajectories also contribute to energy efficiency by minimizing unnecessary accelerations and decelerations, leading to lower energy consumption and operating costs. Therefore, one of the primary concerns in the use of robot manipulators is the optimization of trajectory planning, with a focus on the key criteria of time, jerk and energy

consumption.

To achieve higher production rates in robotic cells, time-optimal trajectory planning is one of the primary requirements. This involves determining the shortest possible time to execute a predefined task path while adhering to the robot's physical constraints and maintaining task quality without compromise. In recent years, many researches have addressed this problem, in [230], the trajectory is represented by splines interpolation, then optimized using dynamic programming (DP) algorithm to generate time-optimal trajectory and smoothly. In [231], a novel Dynamic Programming (DP) algorithm is proposed for obtaining a time-optimal trajectory while considering torque and jerk constraints. The performance of the algorithm is compared with the Sequential Convex Programming (SCP) method. In the work of Serdar [232], a numerical method is developed to generate collision-free trajectories while avoiding robot singularities. The trajectory is constructed using cubic splines and subsequently optimized using the Particle Swarm Optimization (PSO) algorithm to achieve optimal execution time. Further research on the time-optimal trajectory planning problem has been conducted using numerical methods, as presented in [233]–[237]. These approaches focus on determining appropriate time scaling to identify the switching points between maximum acceleration and maximum deceleration, thereby reducing the trajectory time. However, considering only the time factor in robot trajectory planning is insufficient, as it often results in higher energy consumption, particularly given the rising cost of energy resources.

Minimizing the trajectory time of a robot can significantly impact jerk, defined as the rate of change of acceleration. Jerk plays a crucial role in determining the smoothness and efficiency of a robot's motion between points and is generally undesirable due to its negative effects on motion stability and mechanical wear. The problem of simultaneously minimizing time and jerk in trajectory planning has been explored in several studies. Notably, Gasparetto and Zanotto [238] proposed the use of third- and fifth-order B-spline interpolations for trajectory planning. Their approach optimizes a single objective function that combines execution time and the integral of squared jerk. However, this method is limited in its applicability to scenarios where the solution set is non-convex, making it less effective in complex trajectory optimization problems. In [239], the motion profile of a pick-and-place parallel robot is constructed using quintic B-spline curves to achieve C4C4 continuity. The approach considers two key factors: acceleration and jerk, ensuring smoother and more efficient motion trajectories. Junsen et al. [240] applied fifth-order B-spline interpolation to construct the trajectory in joint space. The trajectory was then optimized using the elitist Non-Dominated Sorting Genetic Algorithm (NSGA-II), with time and jerk considered as optimization objectives. In [241], the trajectory is interpolated using fifth-degree polynomials in Cartesian space. The resulting trajectory is then optimized using the Sequential Quadratic Programming (SQP) algorithm, with the objective of minimizing joint jerks to achieve smoother robot motion. Zhang et al. [242] also applied SQP algorithm to obtain time-jerk optimal trajectory for a robotic excavator. To address the trade-off between jerk and time, hybrid meta-heuristic algorithms have been employed to optimize the objective function. In [243], Particle Swarm Optimization (PSO) and the Improved Whale Optimization

Algorithm (IWOA) are combined to enhance the convergence toward the optimal solution. As is well known, the time required to complete tasks and the jerk profile of a robot's movement can significantly impact its energy consumption. Optimizing motion profiles, minimizing task execution delays, and ensuring the efficient operation of components are essential strategies for reducing energy consumption in robotic systems [244].

Given the rising cost of energy resources, minimizing the energy consumption of robot manipulators has become a critical requirement for reducing the operating costs of robotic applications [245]. Additionally, energy-efficient operations contribute to reducing the environmental impact, aligning with sustainable development goals [246], [247]. This has motivated academic research groups to focus on research into energy consumption recently. In [248], a method for minimum-energy trajectory planning in industrial robotic systems is proposed. The approach is based on modeling an electromechanical system with one degree of freedom (1-DOF) to analytically compute energy consumption. While this method is effective for simple 1-DOF systems, it is less suitable for high-degree-of-freedom (high-DOF) systems, such as those with six degrees of freedom, due to increased complexity and computational challenges. In [249], an energy analysis based on dynamic and electromechanical models of a 3-DOF SCARA robot is conducted to demonstrate the correlation between the inertia ellipsoid index and the robot's effective energy consumption. However, the study does not account for the power consumption of the electronic components within the system cabinet, potentially limiting the comprehensiveness of the energy evaluation. In [250], the authors developed a software tool designed to interact with offline programming simulators for industrial robots. The tool enables the computation and optimization of motion parameters with a focus on energy efficiency, providing valuable support for energy-conscious trajectory planning in industrial applications. A new investigation into energy consumption adopts a data-driven approach, where the robot's energy profile is modeled using an artificial neural network for prediction [251], [252]. Li et al. [253] optimized energy consumption based on the robots dynamic model, considering joint torque as a key factor influencing energy usage. The optimization was carried out using the Sequential Quadratic Programming (SQP) algorithm. However, most research on robot energy consumption overlooks crucial factors such as time and jerk, despite their significant impact on motion smoothness, efficiency, and overall system performance.

From the above discussion, research studies that simultaneously consider execution time, jerk, and energy consumption remain limited, with only a few papers addressing all three factors comprehensively. As presented in [254], the trajectory of a serial manipulator is planned using the Non-Uniform Rational B-Splines (NURBS) method, which simultaneously considers time, energy and jerk. In [255], the robot's trajectory is interpolated using a quintic B-spline and optimized with a Quantum-behaved Particle Swarm Optimization (QPSO) algorithm to achieve a time-jerk-energy optimal trajectory for a parallel robot. However, these studies consider joint accelerations as a measure of energy consumption, which does not accurately reflect the actual energy usage of the robot. The true energy consumption is primarily determined by the torque exerted by the actuators, but accurately estimating this is challenging due to the unavailability

of an exact dynamic model of the robot.

## 5.2 Problem statements

### 5.2.1 Time-energy-jerk optimization problem formulation

To execute a manipulator robots motion path, the robot must pass through a sequence of waypoints (or via points) in Cartesian space. As noted by Gasparetto in [256], robot trajectories are typically planned in joint space. In this work, the trajectory planning is also carried out in joint space, where consecutive waypoints in Cartesian space are transformed into joint space using the inverse kinematic model. Based on these joint-space waypoints, a trajectory is generated and subsequently optimized to minimize specific objectives. The trajectory must be sufficiently smooth to prevent excessive mechanical vibrations while ensuring minimal travel time and reduced energy consumption. Additionally, kinematic constraints such as velocity, acceleration, and jerk must be taken into account to achieve efficient and reliable motion planning.

Various performance indices are used to assess the effectiveness of the planned trajectory. In [257], [258], the objective function incorporates the sum of execution time and the integral of squared jerk. In [259], the maximum jerk value is considered as a key performance criterion. In this paper, the performance indices for execution time, jerk and energy consumption are defined in (5.1)(5.3).

The robot trajectory planning translate into multi-objective optimization problem and can defined mathematically as follows:

Minimize:

$$f_{\text{time}} = \sum_{k=0}^{n-1} T_k = \sum_{k=0}^{n-1} (t_{k+1} - t_k) \tag{5.1}$$

$$f_{\text{jerk}} = \sum_{i=1}^{N} \sqrt{\frac{1}{T} \int_0^T (\dddot{s}_i(t))^2 dt} \tag{5.2}$$

$$f_{\text{energy}} = max(|EC(t)|) \tag{5.3}$$

Subject to:

$$\begin{cases} 0 < T \leq T_{\max} \\ |\dot{s}_i(t)| \leq \dot{s}_{\max}, \quad i = 1, 2, ..., N. \\ |\ddot{s}_i(t)| \leq \ddot{s}_{\max} \\ |\dddot{s}_i(t)| \leq \dddot{s}_{\max} \end{cases} \tag{5.4}$$

The symbols mentioned above are explained in Table 5.1

Table 5.1: Explanation of symbols in the optimization problem formulation.

| Symbol | Definition |
|---|---|
| $f_{\text{time}}$ | Objective function of execution time |
| $f_{\text{jerk}}$ | Objective function of jerk |
| $f_{\text{energy}}$ | Objective function of energy consuming |
| $EC$ | Total energy consuming of the robot |
| $s_i$ | Position of $i$th robot joint |
| $\dot{s}_i$ | Velocity of $i$th robot joint |
| $\ddot{s}_i$ | Acceleration of $i$th robot joint |
| $\dddot{s}_i$ | Jerk of the $i$th joint |
| $\dot{s}_{\max}$ | Velocity constraint value |
| $\ddot{s}_{\max}$ | Acceleration constraint value |
| $\dddot{s}_{\max}$ | Jerk constraint value |
| $T_k$ | The time between $k$th and $(k+1)$th waypoints |
| $T$ | Total travel time of the complete trajectory |
| $T_{\max}$ | Total travel time constraint |
| $t_k$ | Time instant of $k$th node |
| $n+1$ | Number of the waypoints |
| $N$ | Number of robot joints |

### 5.2.2 Trajectory planning by 5th-order B-spline in joint space

In robotic programming applications, ensuring smooth and continuous trajectories up to the third derivative (jerk) is essential [260]. To achieve this, the B-spline method is highly recommended due to its flexibility in shaping the trajectory. Moreover, it can be optimized to obtain an optimal trajectory in terms of execution time, jerk, and energy efficiency. This is mathematically expressed by the following equation

$$s(u) = \sum_{j=0}^{m} d_j B_j^p(u) \tag{5.5}$$

where $d_j$ represents the control points of curve, $B_j^p(u)$ denotes spline functions of degree $p$, defined according to knot vector $\mathbf{u} = [u_0, ..., u_{n_{knot}}]$ and $s(u)$ represents the joint poistion at the instant $u$. The $j$-th B-spline basis function of degree $p$ is recursively defined using the Cox-de Boor recursion formula [261].

$$\begin{cases} B_j^0(u) = \begin{cases} 1 & \text{if } \ u_j \leq u < u_{j+1} \\ 0 & \text{otherwise} \end{cases} \\ B_j^p(u) = \frac{(u-u_j)}{(u_{j+p}-u_j)} B_j^{p-1}(u) + \\ \frac{(u_{j+p+1}-u)}{(u_{j+p+1}-u_{j+1})} B_{j+1}^{p-1}(u) \end{cases} \tag{5.6}$$

The continuity of the jerk requires adopting a fifth-degree B-spline ($p = 5$).The trajectory in the cartesian space is discretized into number of waypoints and two virtual ($vp = 2$) points are added at the second and second-last position of the waypoints sequence to obtain a smooth

trajectory (no jerk) at either extremities [257]. These points are then mapped into joint space through the robots inverse kinematic model to find the viapoints $q_k(k = 0, ..., n)$, which are interpolated at times $t_k$, The goal is to find the control points $d_j, j = 0, ..., m$ that ensure

$$s(t_k) = q_k \tag{5.7}$$

It is first necessary to define the knot vector $\mathbf{u}$, composed of sequence of $t_k$ corresponding to the waypoints.

$$\mathbf{u} = \left[\underbrace{t_0 \quad \cdots \quad t_0}_{p+1} \quad t_{vp,1} \quad t_1 \quad \cdots \quad t_{n-1} \quad t_{vp,2} \quad \underbrace{t_n \quad \cdots \quad t_n}_{p+1}\right] \tag{5.8}$$

where $t_{vp,1}$ and $t_{vp,2}$ are time instants of the virtual points.
The length of knot vector is $n + 2p + vp + 1 = n_{knot} + 1$, in B-spline function calculations, the relation between $n_{knot}$, $m$ and $p$ is $n_{knot} - p - 1 = m$, thus the control points number is $n + vp + p = m + 1$.
To determine the control point vector $\mathbf{d}$, a linear system can be constructed by combining of $n + 1$ equations that obtain through interpolation conditions imposed for each viapoint $q_k$ at instant $t_k$:

$$\begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_{n-1} \\ q_n \end{bmatrix} = \begin{bmatrix} B_0^p(t_0) & B_1^p(t_0) & \cdots & B_{m-1}^p(t_0) & B_m^p(t_0) \\ B_0^p(t_1) & B_1^p(t_1) & \cdots & B_{m-1}^p(t_1) & B_m^p(t_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_0^p(t_{n-1}) & B_1^p(t_{n-1}) & \cdots & B_{m-1}^p(t_{n-1}) & B_m^p(t_{n-1}) \\ B_0^p(t_n) & B_1^p(t_n) & \cdots & B_{m-1}^p(t_n) & B_m^p(t_n) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix} \tag{5.9}$$

From (5.9), $n + vp + p$ equations are required to build a square system of $m + 1$ equations and $m + 1$ unknown control points, the extra $p + 1$ can be provide by imposing initial and final conditions of velocity, acceleration and jerk of trajectory. The formulas of velocity, acceleration and jerk can be obtained by differentiating (5.5) up to the third order

$$\dot{s}(u) = \sum_{j=0}^{m-1} c_j B_j^{p-1}(u) \tag{5.10}$$

$$\ddot{s}(u) = \sum_{j=0}^{m-1} l_j B_j^{p-2}(u) \tag{5.11}$$

$$\dddot{s}(u) = \sum_{j=0}^{m-1} w_j B_j^{p-3}(u) \tag{5.12}$$

where $c_j$ ,$l_j$ and $w_j$ are, respectively, the velocity curve, acceleration, and jerk control points and are computed as

$$\begin{cases} c_j = p\frac{d_{j+1}-d_j}{u_{j+p+1}-u_{j+1}} \\ l_j = (p-1)\frac{c_{j+1}-c_j}{u_{j+p+1}-u_{j+1}} \\ w_j = (p-2)\frac{l_{j+1}-l_j}{u_{j+p+1}-u_{j+1}} \end{cases} \tag{5.13}$$

The velocity , acceleration and jerk at waypoints can be described as following

$$\dot{q}_k = \begin{bmatrix} B_0^{p(1)}(t_k) & B_1^{p(1)}(t_k) & \cdots & B_{m-1}^{p\ (1)}(t_k) & B_m^{p\ (1)}(t_k) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix} \tag{5.14}$$

$$\ddot{q}_k = \begin{bmatrix} B_0^{p(2)}(t_k) & B_1^{p(2)}(t_k) & \cdots & B_{m-1}^{p\ (2)}(t_k) & B_m^{p\ (2)}(t_k) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix} \tag{5.15}$$

$$\dddot{q}_k = \begin{bmatrix} B_0^{p(3)}(t_k) & B_1^{p(3)}(t_k) & \cdots & B_{m-1}^{p\ (3)}(t_k) & B_m^{p\ (3)}(t_k) \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{m-1} \\ d_m \end{bmatrix} \tag{5.16}$$

where $B_j^{p(i)}(t_k)$ represent $i$-th derivative of $B_j^p(t)$ at $t_k$
The initial and final values of velocity, acceleration and jerk can be expressed as follows

$$\begin{cases} \dot{s}(t_0) = \dot{q}_0 \\ \dot{s}(t_n) = \dot{q}_n \\ \ddot{s}(t_0) = \ddot{q}_0 \\ \ddot{s}(t_n) = \ddot{q}_n \\ \dddot{s}(t_0) = \dddot{q}_0 \\ \dddot{s}(t_n) = \dddot{q}_n \end{cases} \tag{5.17}$$

this leads (5.9) to be square linear system with same number of equations and unknowns of $m+1$

$$A\mathbf{d} = c \tag{5.18}$$

where

$$\mathbf{d} = \begin{bmatrix} d_0 & d_1 & \cdots & d_{m-1} & d_m \end{bmatrix}^T \tag{5.19}$$

$$c = \begin{bmatrix} q_0 & \dot{q_0} & \dddot{q_0} & q_1 & \cdots & q_{n-1} & \dddot{q_n} & \ddot{q_n} & \dot{q_n} & q_n \end{bmatrix}^T \tag{5.20}$$

$$A = \begin{bmatrix} B_0^p(t_0) & B_1^p(t_0) & \cdots & B_{m-1}^p(t_0) & B_m^p(t_0) \\ B_0^{p(1)}(t_0) & B_1^{p(1)}(t_0) & \cdots & B_{m-1}^{p\ (1)}(t_0) & B_m^{p\ (1)}(t_0) \\ B_0^{p(2)}(t_0) & B_1^{p(2)}(t_0) & \cdots & B_{m-1}^{p\ (2)}(t_0) & B_m^{p\ (2)}(t_0) \\ B_0^{p(3)}(t_0) & B_1^{p(3)}(t_0) & \cdots & B_{m-1}^{p\ (3)}(t_0) & B_m^{p\ (3)}(t_0) \\ B_0^p(t_1) & B_1^p(t_1) & \cdots & B_{m-1}^p(t_1) & B_m^p(t_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ B_0^p(t_{n-1}) & B_1^p(t_{n-1}) & \cdots & B_{m-1}^p(t_{n-1}) & B_m^p(t_{n-1}) \\ B_0^{p(3)}(t_n) & B_1^{p(3)}(t_n) & \cdots & B_{m-1}^{p\ (3)}(t_n) & B_m^{p\ (3)}(t_n) \\ B_0^{p(2)}(t_n) & B_1^{p(2)}(t_n) & \cdots & B_{m-1}^{p\ (2)}(t_n) & B_m^{p\ (2)}(t_n) \\ B_0^{p(1)}(t_n) & B_1^{p(1)}(t_n) & \cdots & B_{m-1}^{p\ (1)}(t_n) & B_m^{p\ (1)}(t_n) \\ B_0^p(t_n) & B_1^p(t_n) & \cdots & B_{m-1}^p(t_n) & B_m^p(t_n) \end{bmatrix} \tag{5.21}$$

Thus, the control points can be obtained by following equation (5.22)

$$\mathbf{d} = A^{-1}c \tag{5.22}$$

Considering that the control points vector relies on the times intervals between waypoints, alterations to the trajectory's velocity, acceleration and jerk may arise.

## 5.3 Prediction model of energy consumption using LSTM

As stated in the problem formulation, an energy consumption model is required to construct an optimization objective function. To determine the energy consumed by the manipulator robot, one must either analyze and extract the mathematical relationship between the current, voltage, and time of each servo motor of the robot to calculate power, which is difficult to resolve, or describe it by torque through robot dynamic model which is difficult to obtained in practice.

Methods based on deep learning have recently gained popularity for creating models to predict the robot's energy consumption. This is due to their simple modeling approach, which avoids complex dynamic modeling processes [262]–[264]. LSTM is an improved version of the theory of recurrent neural network (RNN) regression, due to its ability to identify non-linear and complex relationships [265]. In this study LSTM is applied to establish the relationship between the executed trajectory and its corresponding consumed energy.

The components of the proposed LSTM network are: data preprocessing module, input layer, three LSTM layers, dropout layer, fully connected layer and regression layer. The data preprocessing module normalizes the training data and removes features with constant values that

may have a negative impact on training. The LSTM layers can learn from sequences data and retain information over time steps and exploits deep relationships between energy consumption and input variables. To prevent LSTM overfitting risk a dropout layer is adopted. The fully connected layer is added for learning nonlinear combinations of data features. The regression layer calculates the error loss in output layer for the regression task. Figure 5.1 illustrates the architecture of our proposed LSTM network for regression. The input sequence of our designing LSTM network can be represented as:

$$x_t = \begin{bmatrix} s_1 & \cdots & s_6 & \dot{s}_1 & \cdots & \dot{s}_6 & \ddot{s}_1 & \cdots & \ddot{s}_6 \end{bmatrix}^T \tag{5.23}$$

and the output sequence, representing the energy consumption indicated by the absolute value of the torque at each robot joint, can be expressed as

$$\sum_{i=1}^{N} \int_0^T |\tau_i(t)| dt \tag{5.24}$$

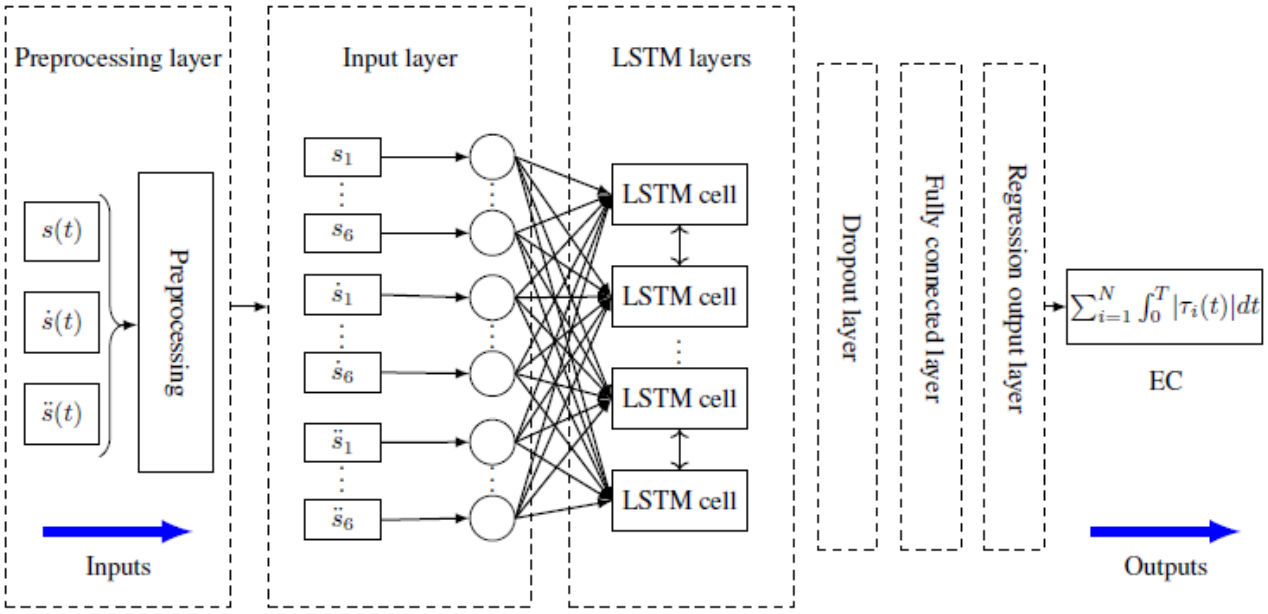



Figure 5.1: Schematic of the proposed LSTM network.

### 5.3.1 Structure of LSTM cell

The LSTM cell is the alternative element of the neuron in RNNs. Capable of modeling long-term dependencies through the use of memory cells to store information and gates to regulate its flow between them (see Figure 5.2).

The flow of information is regulated by the forget gate $G_t$, input gate $I_t$, output gate $O_t$ and also cell candidate $\tilde{C}t$ via following formulas

$$G_t = sig(W_G[y_{t-1}\ x_t]^T + b_G) \tag{5.25}$$

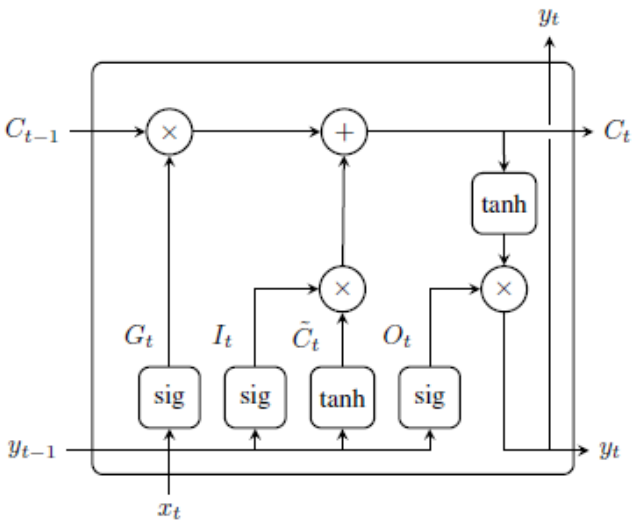



Figure 5.2: LSTM cell architecture.

$$I_t = sig(W_I[y_{t-1}\ x_t]^T + b_I) \tag{5.26}$$

$$\tilde{C}_t = tanh(W_C[y_{t-1}\ x_t]^T + b_C) \tag{5.27}$$

$$O_t = sig(W_O[y_{t-1}\ x_t]^T + b_O) \tag{5.28}$$

where $W_{(.)}$ are the weight vectors of gates, $b_{(.)}$ are bias vectors, $sig$ is sigmoid activation function and $tanh$ is hyperbolic tangent activation function. They are defined as follows:

$$sig(v) = \frac{1}{1+e^{-v}} \tag{5.29}$$

$$tanh(v) = \frac{e^{v} - e^{-v}}{e^{v} + e^{-v}} \tag{5.30}$$

while cell output $y_t$ and cell state $C_t$ are given by

$$C_t = G_t \otimes C_{t-1} + I_t \otimes \tilde{C}_t \tag{5.31}$$

$$y_t = O_t \otimes tanh(C_t) \tag{5.32}$$

where $\otimes$ denotes element-wise multiplication of vectors.

## 5.4 Time-jerk-energy optimization using NSGA-II

The NSGA-II is multi-optimization method that developed by Deb et al.[266] has proven its effectiveness in various fields where optimization problems involve multiple conflicting objectives. Aims to efficiently explore the solution space, maintain diversity among solutions, and identify solutions for a conflicting objectives called Pareto-ptimal solutions.
In the NSGA-II process, the offspring population $Q_t$, is initially formed by utilizing the parental population $P_{it}$, along with the standard genetic operators. Subsequently, $P_{it}$ and $Q_{it}$ are merged to form a new population $R_{it}$, of size $2P$. Then, the $R_{it}$ population is classified based on

non-domination sorting method. In new generation, the parent population $P_{it+1}$, is filled by non-dominated fronts increasingly until the population size achieves $P$. Solutions are sorted based on crowding distance in descending order. This scenario is illustrated in Algorithm 1.

**Algorithm 1** The NSGA-II procedure

```
Initialize random population P_it of size N_p;
Generate offspring population Q_t;
while it ≤ number of generation do                ▷ it is iteration number
    R_it = P_it ∪ Q_it;
    F_i = non-dominated sorting(R_it);            ▷ F_i are non-dominated fronts of R_it
    P_it+1 = ∅ and i = 1;
    while |P_it+1| + |F_i| ≤ N_p do
        classify by crowding distance(F_i);
        P_it+1 = P_it+1 ∪ F_i;
        i = i + 1;
    end while
    Q_it+1 = generate new pop(P_it+1);
    it = it + 1;
end while
```

According to the section 5.2.2 analysis, solving the problem of time-jerk-energy optimal trajectory planning consists of finding the control points of the B-spline curve, which relies on the time instants of the waypoints. In this study, the optimizer works to find appropriate time instants for the waypoints of the trajectory, minimizing the objectives described in Eqs.(5.1-5.3) under the kinematic constraints described in Eq.(5.4).

In the optimization procedure, the optimizer defines the instants of time $t_k$ corresponding to the predefined waypoints of the reference trajectory. The generated trajectory is tested against the constraints, then evaluated through the objective functions,and optimizes the sequence of $t_k$, repeating the process until the objective is achieved. Figure 5.3 illustrates the optimization flowchart.

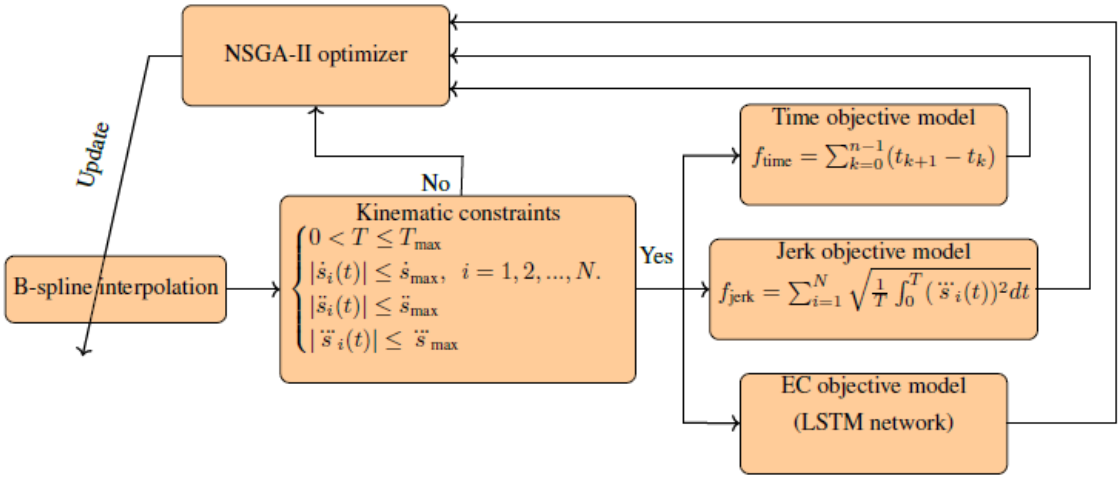



Figure 5.3: Flowchart of optimization.

## 5.5 Results and discussion

To demonstrate the feasibility of the proposed approach, simulation experiments are carried out on an industrial robot model the of 6-DOF Fanuc M710iC70 (see figure 5.4). The proposed approach is implemented in the Matlab/Simulink software environment. The simulation experiments focus on three main aspects:

Robot energy consumption model: This involves designing an energy consumption profile model for the robot, which based on data collection, training and validation using a test sample.

Process optimization: This aspect involves running the process optimization to obtain optimal time parameters, which results in an optimal trajectory in terms of time, jerk, and energy.

Method comparison: This provides a comparison between the proposed approach and one of the classical methods in trajectory planning of robots to evaluate the effectiveness of our approach.

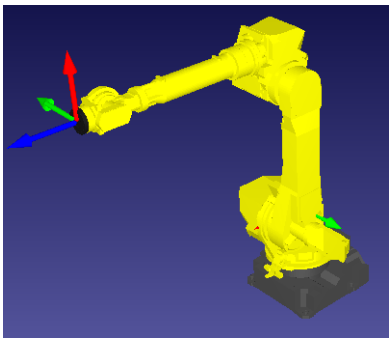


Figure 5.4: 3D robot model of Fanuc M710iC70.

### 5.5.1 Robot energy consumption model

For training the energy model, input/output data were extracted from the experimental simulations. The position, velocity and acceleration are considered as input data, and the integral of the absolute value of the corresponding torque, representing energy consumption, is taken as output data. In this study, the input data consist of the position, velocity and acceleration of six robot joints, as the movement of a single joint can affect the robot's overall energy consumption, while in the output data, we preferred to take the sum of energy consumption of all joints to avoid complexity in calculation, as well as the total energy being what matters us for a IR system.

The trajectories were generated using the B-spline method, with random waypoints selected using MATLAB's random function.The waypoints were constrained within the robot's joints movement limits of [-90ř, 90ř]. This approach ensures diverse trajectory generation, enabling the robot to learn effectively across a wide range of motions.In future work, this method can be extended to comprehensively cover the entire robot workspace. The measurements of position, velocity, acceleration and corresponding energy consumption values are collected into a training set. The other LSTM network parameters are listed in Table 5.2.

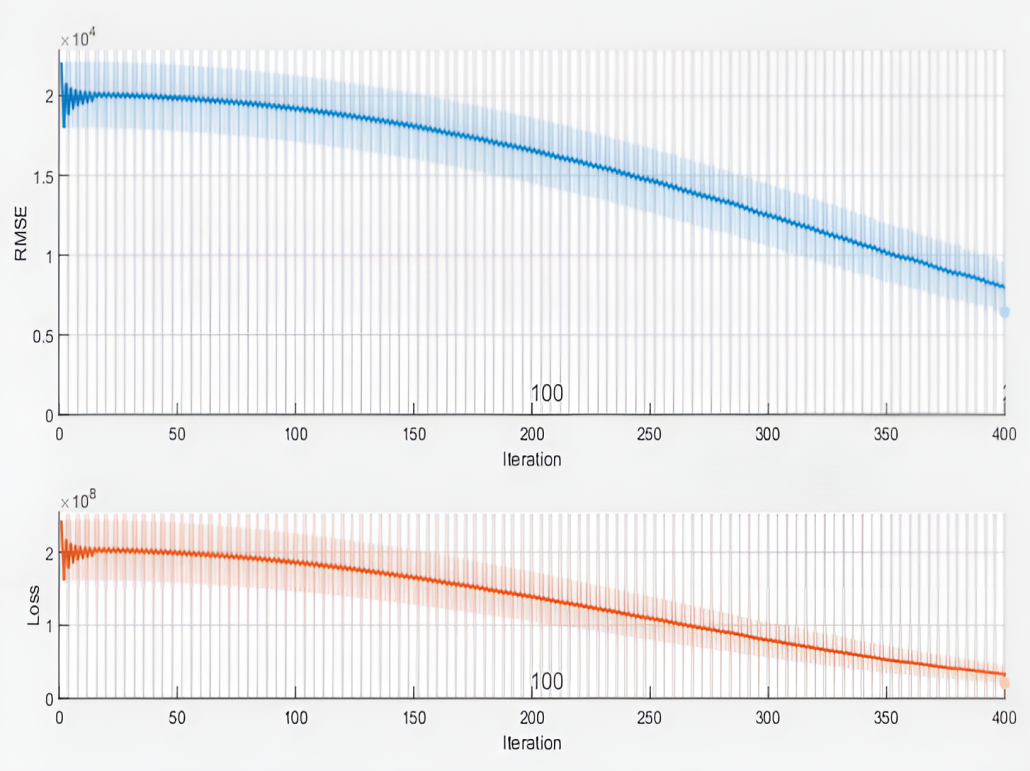



Figure 5.5: Training model performance with 50 trajectories.

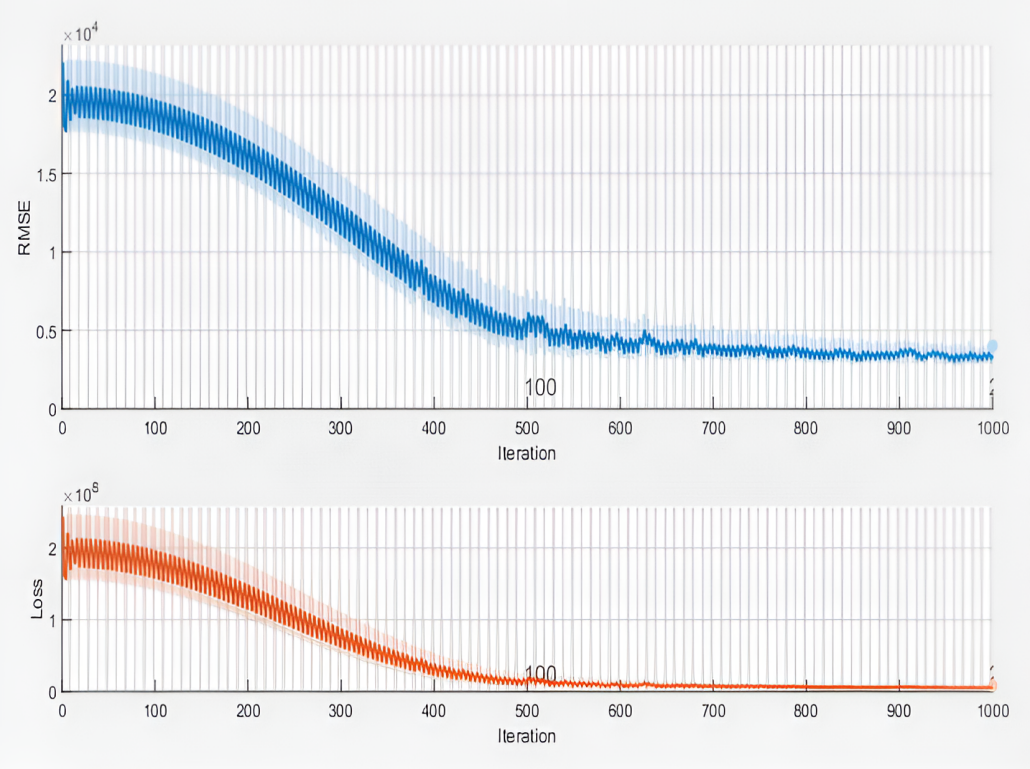



Figure 5.6: Training model performance with 100 trajectories.

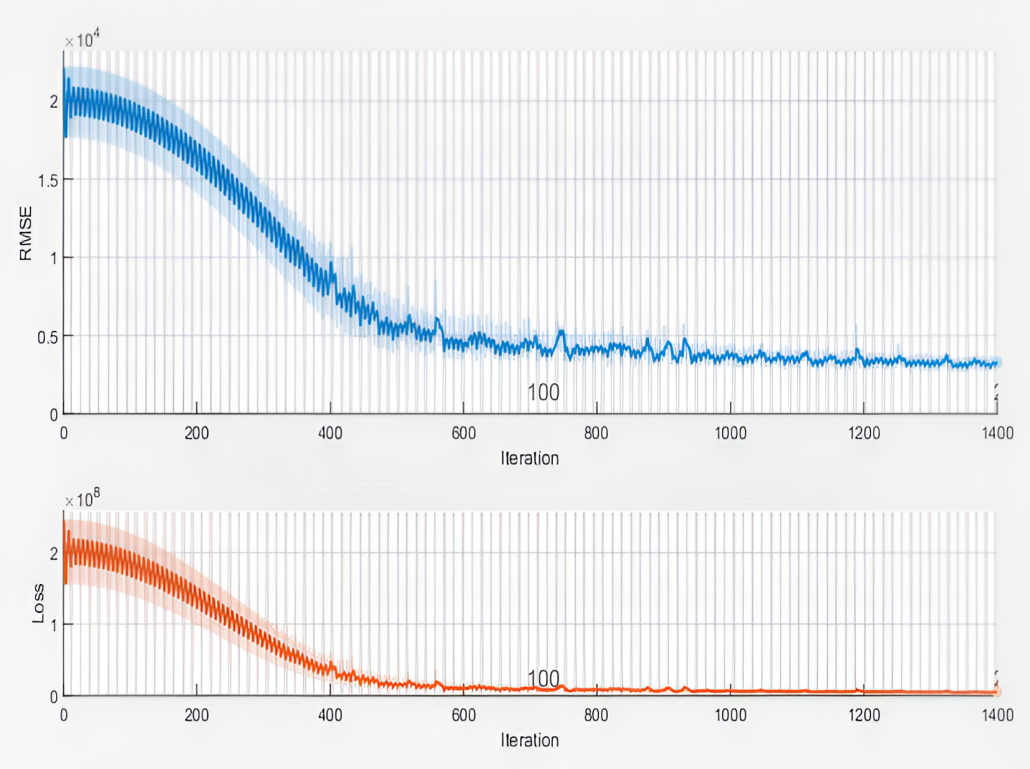



Figure 5.7: Training model performance with 150 trajectories.

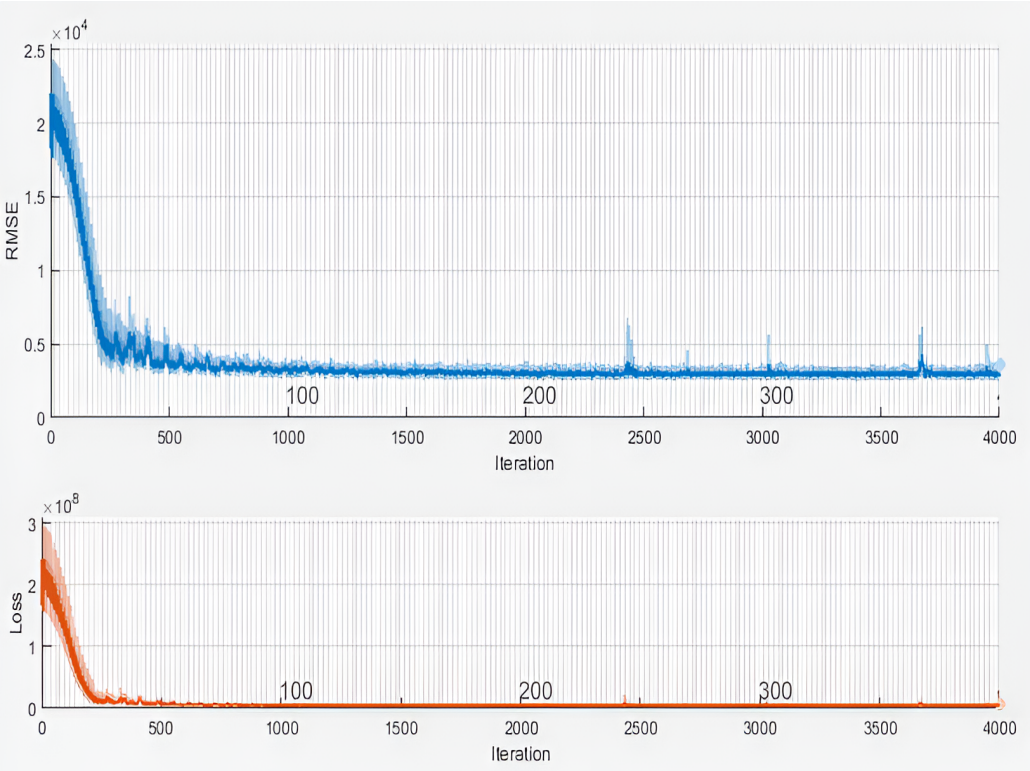



Figure 5.8: Training model performance with 200 trajectories.

Table 5.2: Parameters of proposed LSTM.

| Parameters | Values |
|---|---|
| Number of input Features | 18 |
| Number of layers | 3 |
| Number of hidden units of each layer | 100 |
| Number of Responses | 1 |
| Epochs | 400 |
| Min BatchSize | 20 |
| Learning Rate | 0.01 |
| Solver | adam |

The LSTM model was trained using different numbers of trajectories (50, 100, 150 and 200) to evaluate its performance in terms of Root Mean Squared Error (RMSE) and achieve maximum accuracy. As shown in Figures 5.5, 5.6 and 5.7, the RMSE decreases as the number of trajectories increases, indicating improved model performance with larger training sets. At 200 trajectories, the RMSE stabilizes at iteration 2000, suggesting that further increasing the number of trajectories may not significantly enhance accuracy beyond this point as depicted in Figure 5.8.

After training the proposed energy consumption (EC) model, the results demonstrated a significant improvement in convergence speed, indicating that the model quickly reached a satisfactory level of accuracy in predicting the robot's power consumption. This faster convergence suggests that the model effectively captured the underlying energy consumption patterns, reducing the need for extensive retraining. During the training process, the loss function exhibited smooth stabilization, implying that the model's performance had plateaued. This stabilization suggests that additional training iterations would likely yield minimal improvements, confirming that the model had attained an optimal balance between accuracy and generalization without overfitting to the training data. Figure 5.9 illustrates the evolution of energy consumption along a randomly selected trajectory, showing that the trained model accurately predicts the robots energy consumption. The strong consistency between the models predictions and the experimental data further validates its reliability and effectiveness in estimating real-world energy usage.

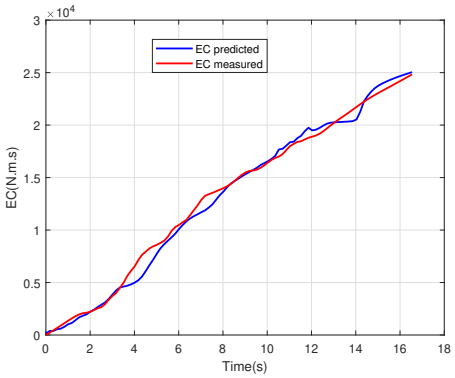



Figure 5.9: Test samples of EC predicted vs EC measured.

### 5.5.2 Running the optimization process

After modeling the energy profile of the robot, this section focuses on the optimization process. The time-jerk-energy optimization is addressed using NSGA-II to determine the optimal time instants. The proposed method is examined on the trajectory of waypoints in the joint space presented in Table 5.3, subject to kinematic constraints, which are presented in Table 5.4. The initial and final conditions of velocity, acceleration and jerk of the trajectory are initialized to zero. The NSGA-II algorithm is used for optimizing the objective functions with these settings: number of generation = 50, size of population = 100 and the crossover probability is randomly selected at each generation.

Table 5.3: Kinematic constraints of the robot joints.

| Number of joints | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Velocity $\dot{s}_{\max}$ (deg/s) | 10 | 15 | 10 | 10 | 15 | 10 |
| Acceleration $\ddot{s}_{\max}$ (deg/s$^2$ ) | 15 | 20 | 20 | 10 | 15 | 10 |
| Jerk $\dddot{s}_{\max}$ (deg/s$^3$) | 18 | 25 | 20 | 15 | 20 | 12 |

Table 5.4: Waypoints of trajectory in joint space (°).

| Waypoints | J1 | J2 | J3 | J4 | J5 | J6 |
|---|---|---|---|---|---|---|
| 1 | 19 | 70 | 76 | -2 | -17 | 18 |
| 2 | Virtual | | | | | |
| 3 | 23 | 89 | 66 | -6 | -16 | 13 |
| 4 | 18 | 81 | 75 | -10 | -7 | 21 |
| 5 | 21 | 81 | 71 | -8 | -5 | 23 |
| 6 | 26 | 86 | 73 | -5 | 0 | 24 |
| 7 | 29 | 88 | 77 | -4 | 4 | 27 |
| 8 | Virtual | | | | | |
| 9 | 26 | 90 | 75 | -3 | 1 | 27 |

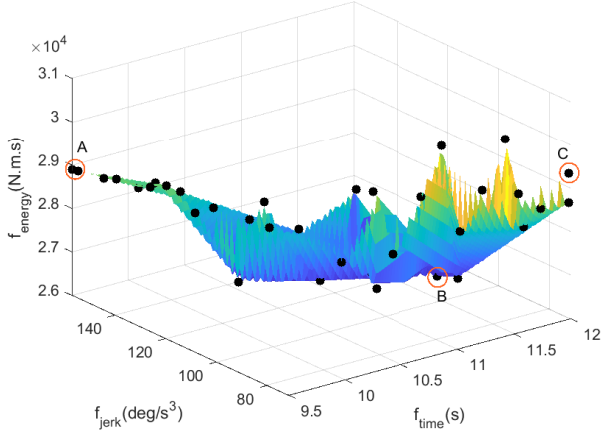
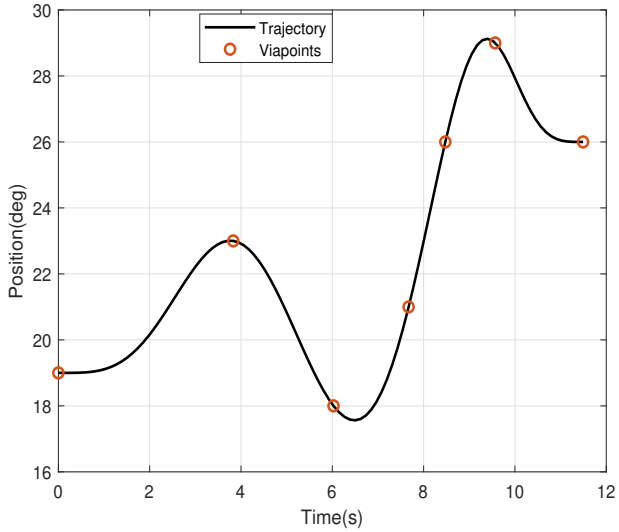
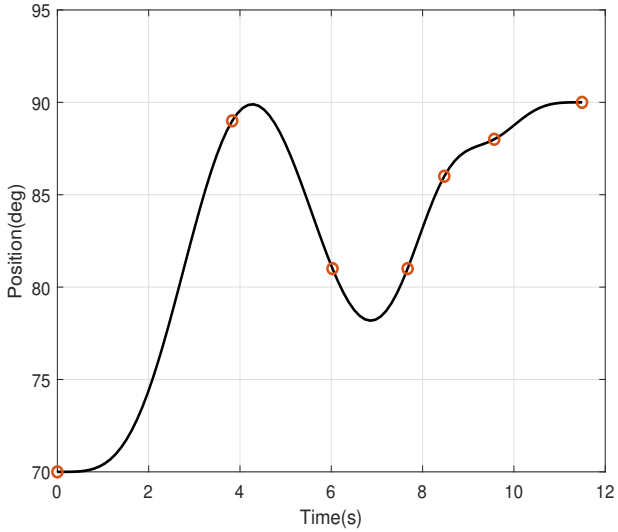



Figure 5.10: Pareto front of time-jerk-energy optimization.

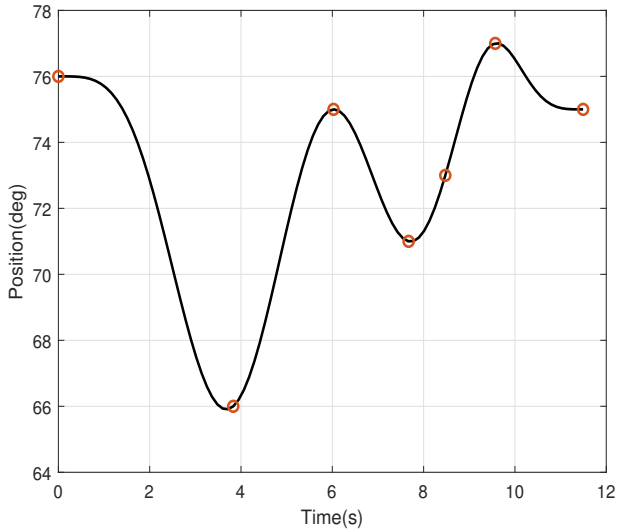
As illustrated in Figure 5.10, optimal Pareto front of the time-jerk-energy optimization obtained



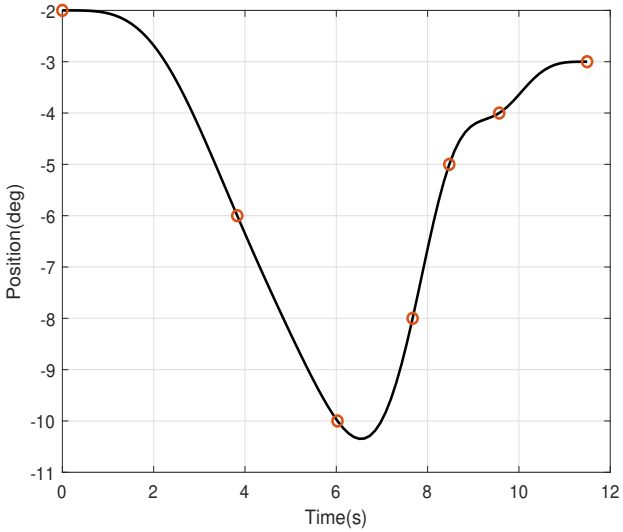

(a) Joint 1



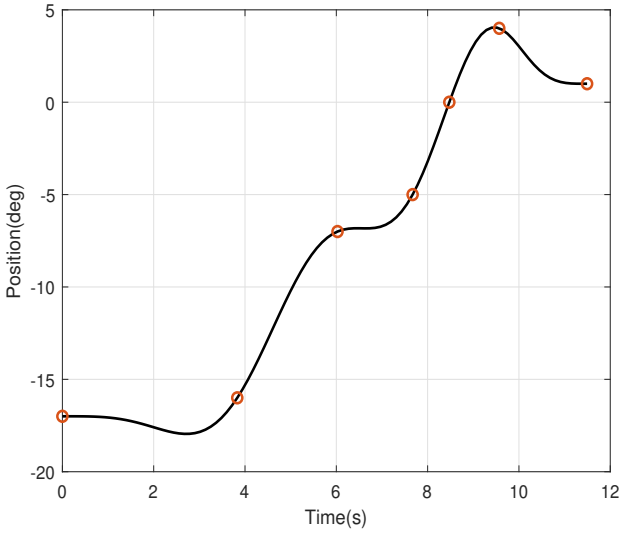

(b) Joint 2



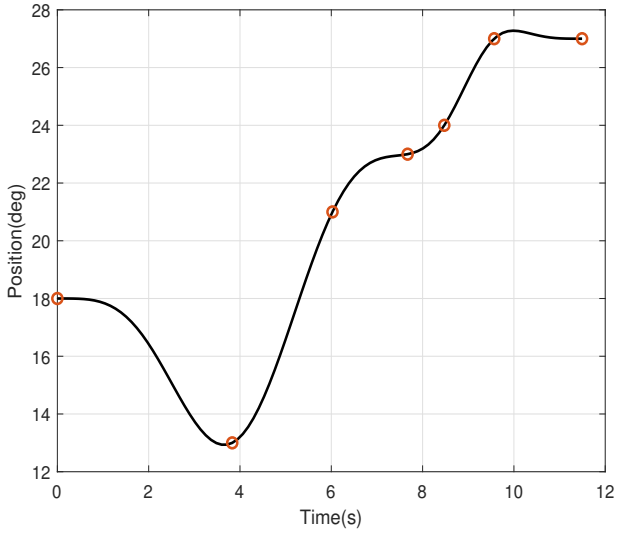

(c) Joint 3




(d) Joint 4

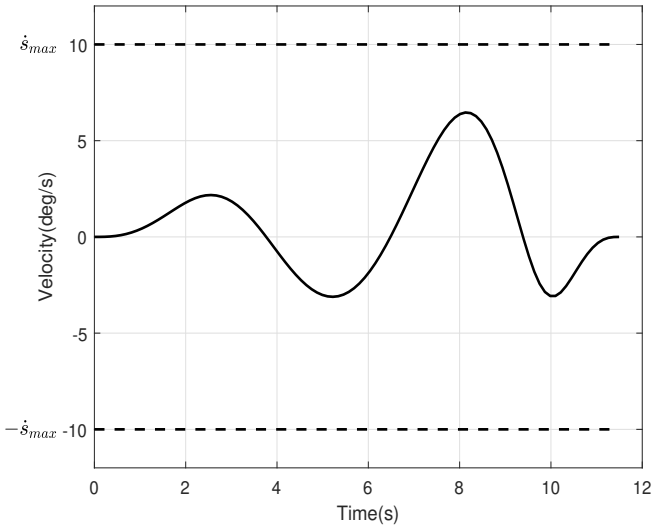



(e) Joint 5

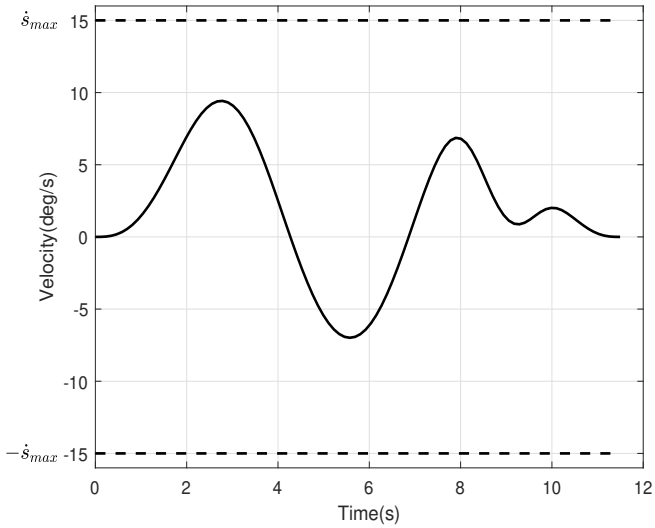



(f) Joint 6

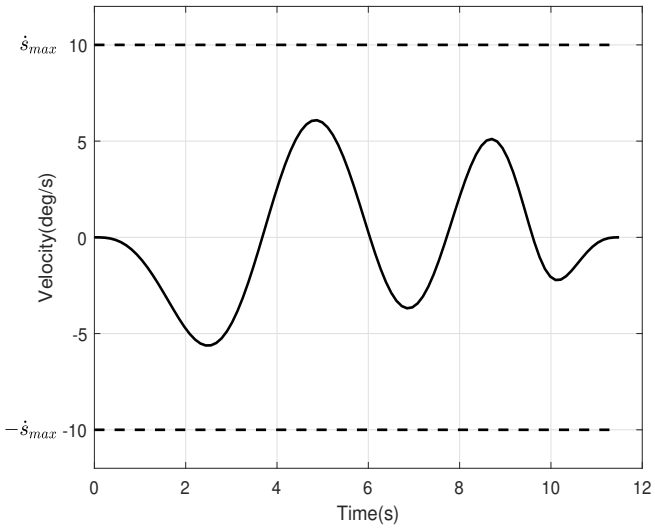
Figure 5.11: The position of joints of robot trajectory.



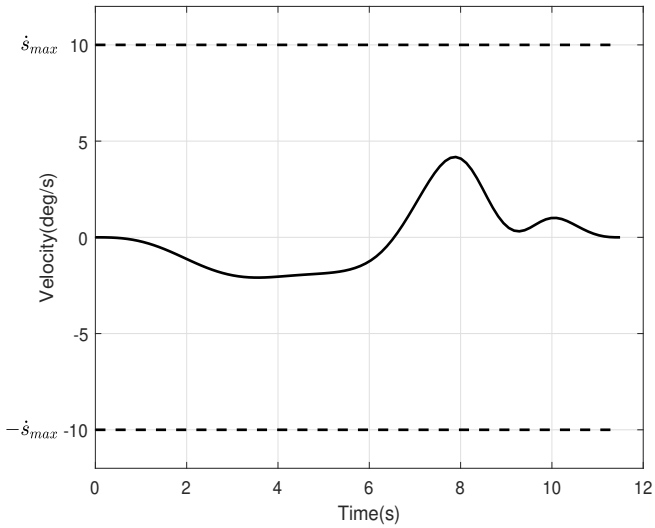

(a) Joint 1



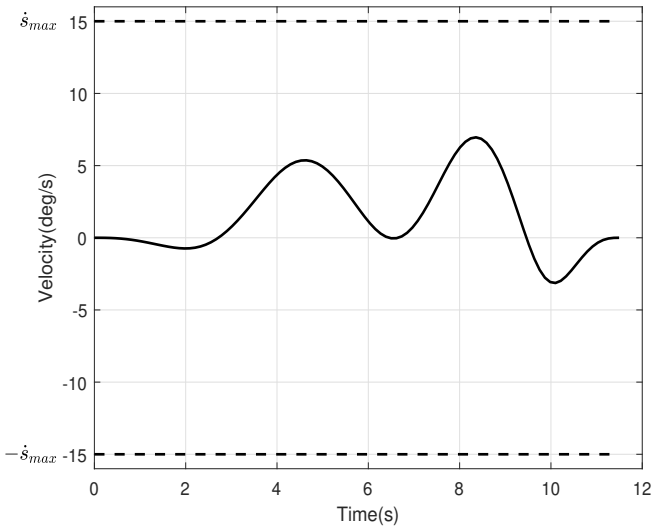

(b) Joint 2



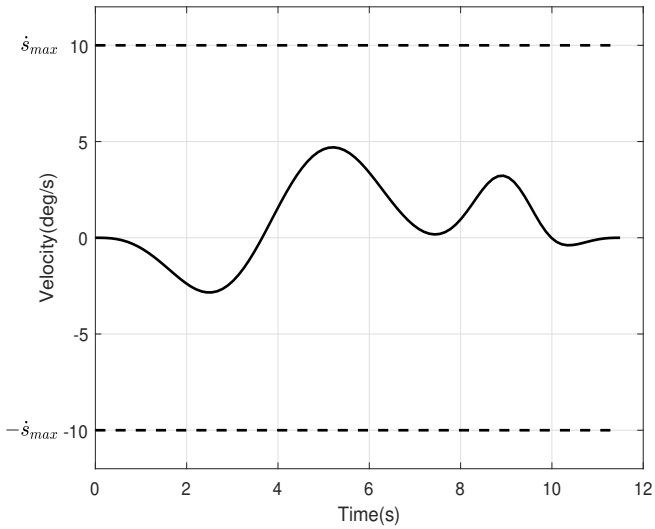

(c) Joint 3




(d) Joint 4

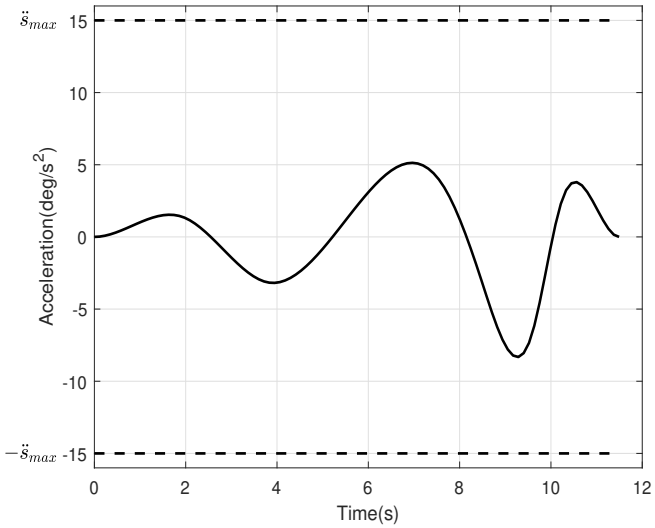



(e) Joint 5

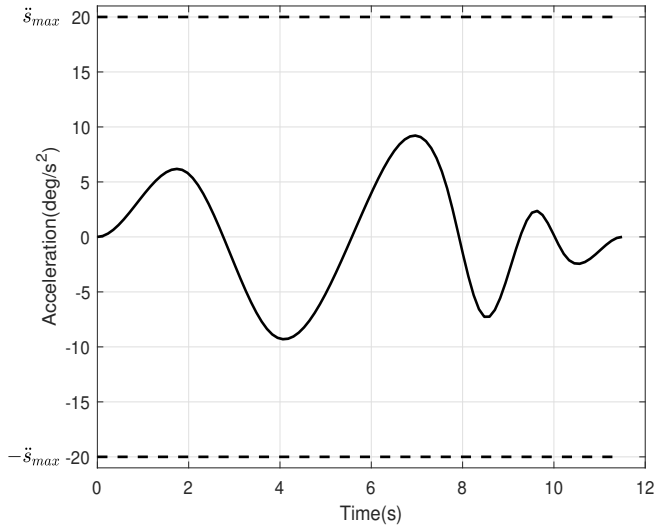



(f) Joint 6

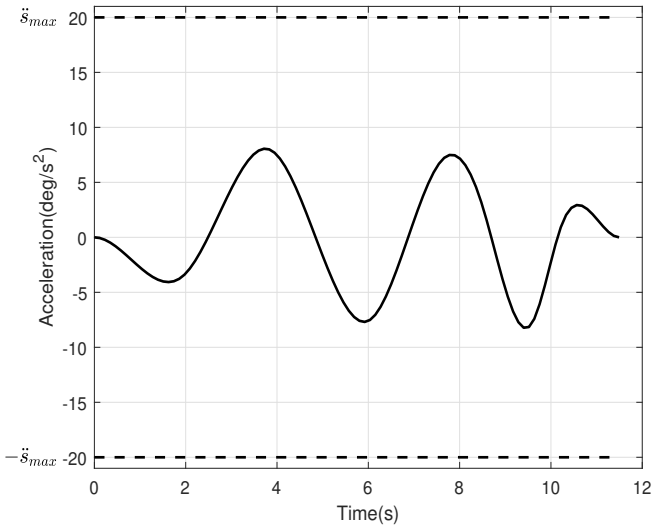
Figure 5.12: The velocity of joints of robot trajectory.



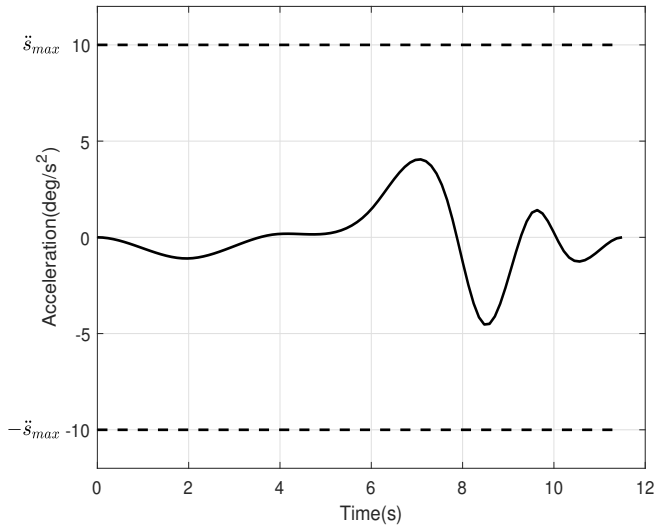

(a) Joint 1



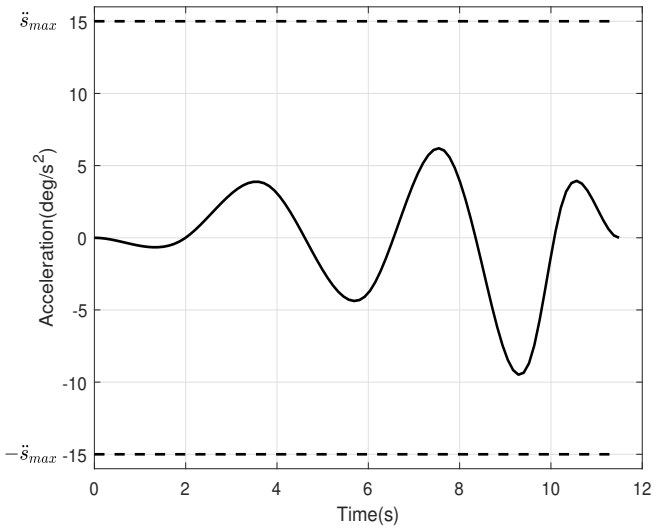

(b) Joint 2



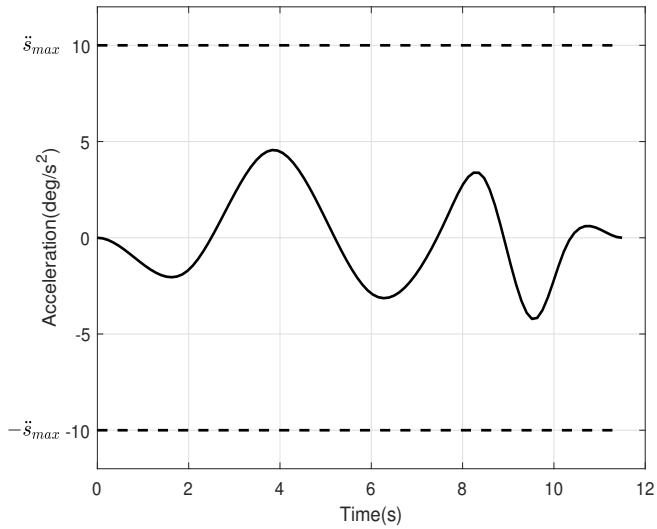

(c) Joint 3




(d) Joint 4

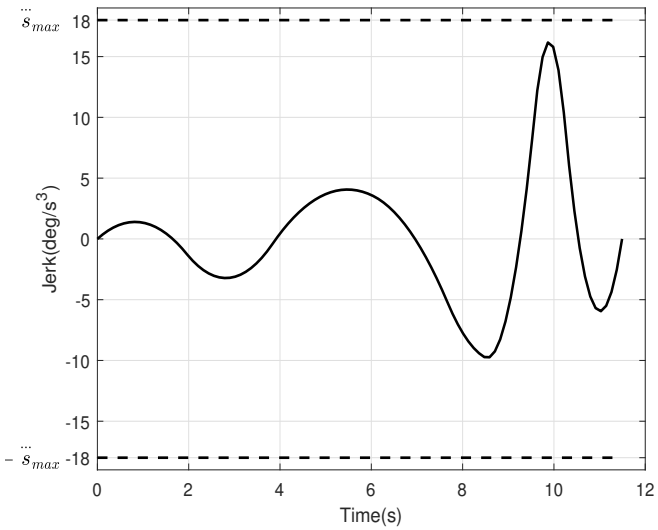



(e) Joint 5

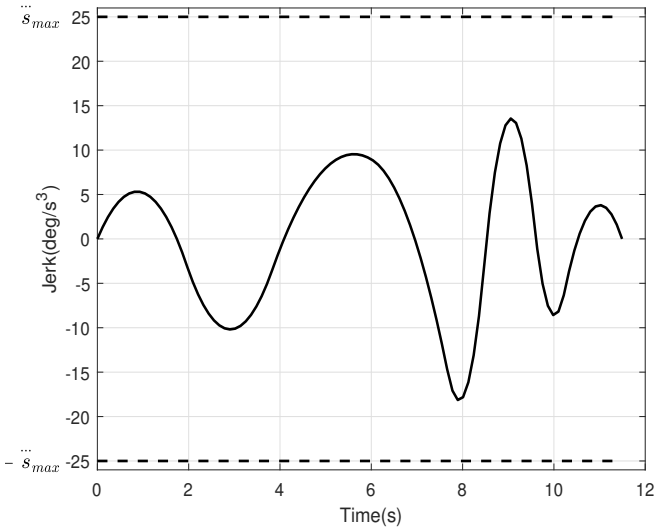



(f) Joint 6

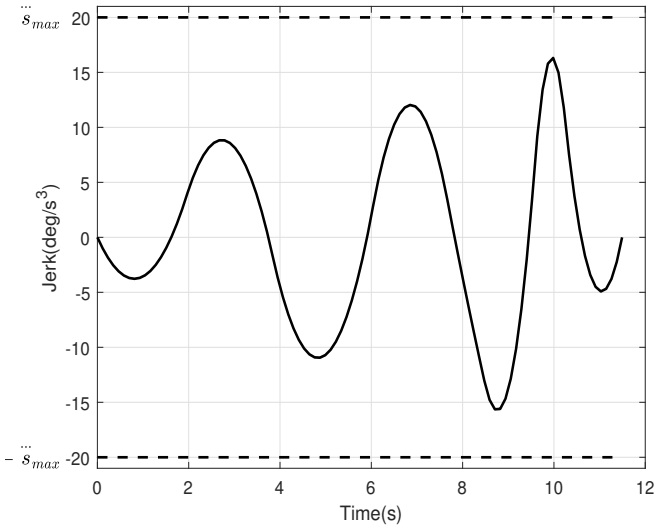
Figure 5.13: The acceleration of joints of robot trajectory.



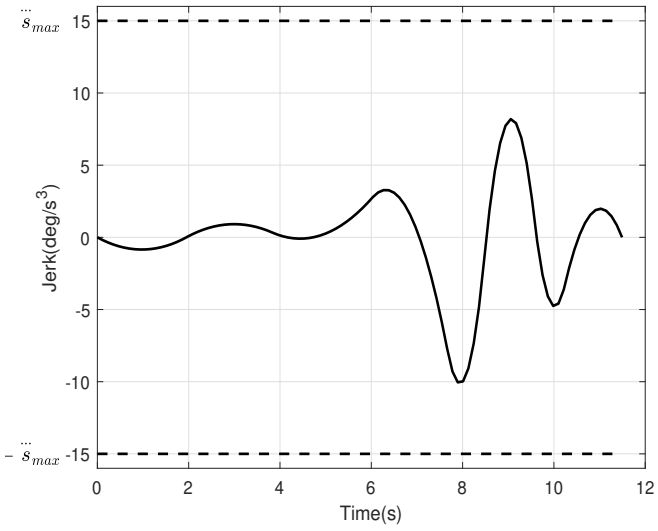

(a) Joint 1



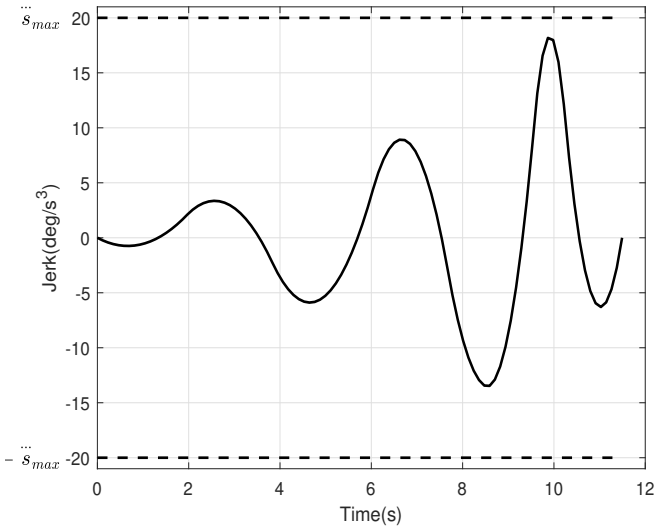

(b) Joint 2



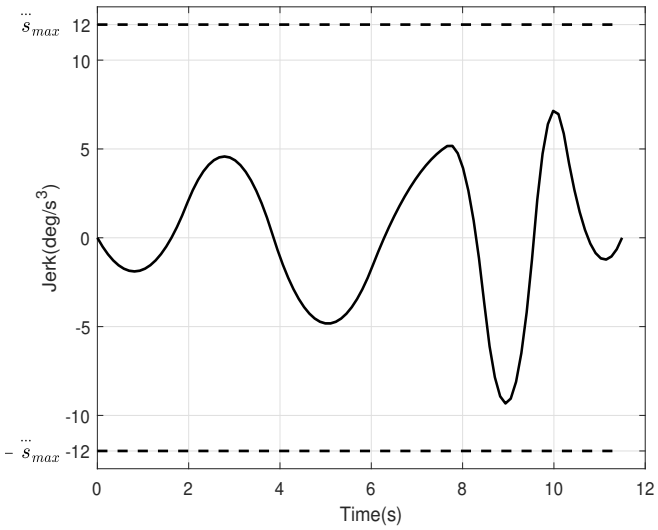

(c) Joint 3




(d) Joint 4




(e) Joint 5




(f) Joint 6

Figure 5.14: The jerk of joints of robot trajectory.

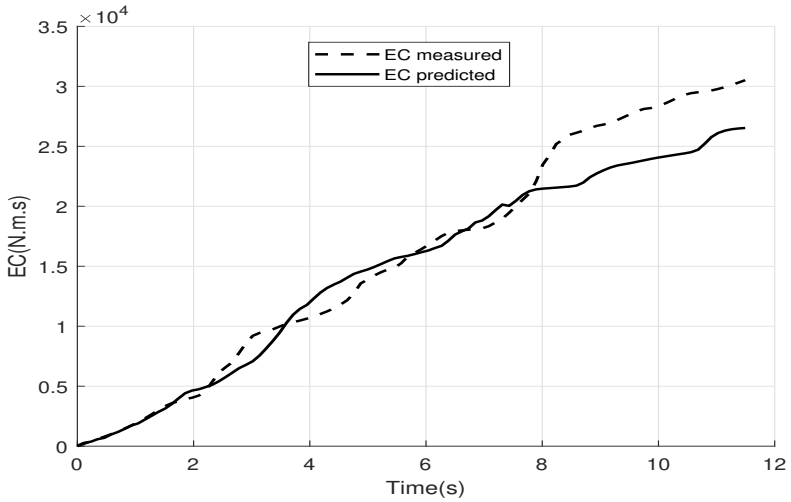



Figure 5.15: Energy consuming for the trajectory with an execution time of 11.49s .

by NSGA-II. In the pareto front solutions, the trajectory time varies between 9.5s and 11.99s, the jerk index ranges from 71.73 deg/s$^3$ to 156.91 deg/s$^3$ while the energy consumption index from 26534 N.m.s to 30238 N.m.s. The shortest execution time with the greatest value of jerk for solution A, while the energy value lies intermediate between their limiting values, meanwhile the solution C requires a longer traveling time but exhibits the least jerk and highly energy consuming. In terms of execution time, option B is better than solution C. It also performs better than solution A when it comes to jerk, while surpassing both solutions A and C in terms of energy consumption. Therefore, it can be considered a compromise solution that provide a good balance between the competing objectives. The final selection of solutions from the Pareto front depends on the specific problem and the preferences of the decision-maker. This process may involve choosing one or more solutions and considering factors such as computational cost. Given that the optimal solutions on the Pareto front comply with the kinematic constraints and that the difference between the traveling time values corresponding to these solutions is not significant, the criterion for choosing the optimal solution is therefore the energy consuming, which in this case is solution B with a pridected energy consuming of 26534 N.m.s and with a traveling time of 11.49s, the corresponding time intervals of this solution is shown in Table 5.5. The position, velocity, acceleration and jerk of trajectory with a 11.49s execution time are

Table 5.5: Time intervals of solution 11.49s.

| Interval | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_5$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|
| Interval of time | 1.952s | 1.877s | 2.195s | 1.645s | 0.803s | 1.093s | 0.674s | 1.249s |

shown in Figures 5.11-5.14. From the resulting trajectory, the first important observation is that the trajectory passes through the predefined waypoints, demonstrating the flexibility of the B-spline method. Additionally, all kinematic limits are respected, proving that NSGA-II selects solutions that best meet specific needs and constraints. In Figure 5.14 the jerk curve is smooth and continuous, which increases the accuracy of the trajectory movement, avoids

actuator vibrations and reduces energy consumption. Figure 5.15 shows the measured and predicted energy consumption during the trajectory, both of which exhibit nearly the same growth. This demonstrates that our LSMT model maintained its effectiveness during the optimization process.

### 5.5.3 Compared the suggested approach with the classical method

This section compares the proposed approach with the classical method for trajectory planning in terms of energy and jerk. In trajectory planning using the B-spline method, determination the time instants corresponding to the interpolated waypoints is needed. This section presents a comparison in obtaining the time vector between the suggested approach and one of the most widespread methods found in the literature [267], known as the chord length distribution.

**Chord length distribution method:**

In the chord length distribution method, which is referred to by the expression "the classic method" in this article, the time parameter vector $\bar{u}$ covering the interval [0, 1] is obtained based on computed the distances between successive waypoints.
The waypoints of the robot's trajectory are labeled as $D_0, D_1, D_2, , D_n$. The length between two consecutive waypoints $D_k$ and $D_{k+1}$ is written as

$$L_k = |D_{k+1} - D_k| \tag{5.33}$$

and the total length of the trajectory is the sum of the lengths of these segments is

$$L = \sum_{k=0}^{n-1} |D_{k+1} - D_k| \tag{5.34}$$

therefore, $\bar{u}$ can be determined by equation (5.35)

$$\begin{cases} \bar{u}_0 = 0 \\ \bar{u}_k = \bar{u}_{k-1} + \frac{|D_{k+1} - D_k|}{\sum_{k=0}^{n-1} |D_{k+1} - D_k|} \quad k = 1, 2, \cdots, n-1 \\ \bar{u}_n = 1 \end{cases} \tag{5.35}$$

it is required to set the total execution time of the robot trajectory a priori to determine the time instants of waypoints. For a fair comparison, the execution time must be the same for both methods. Therefore, an execution time of 11.49s is considered for calculating the time instants based on equation (5.33). The resulting time intervals are presented in Table 5.6.

Table 5.6: Time intervals from the classic method.

| Interval | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_5$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|---|
| Interval of time | 0.905s | 0.908s | 2.233s | 2.582s | 1.43s | 1.499s | 0.969s | 0.963s |

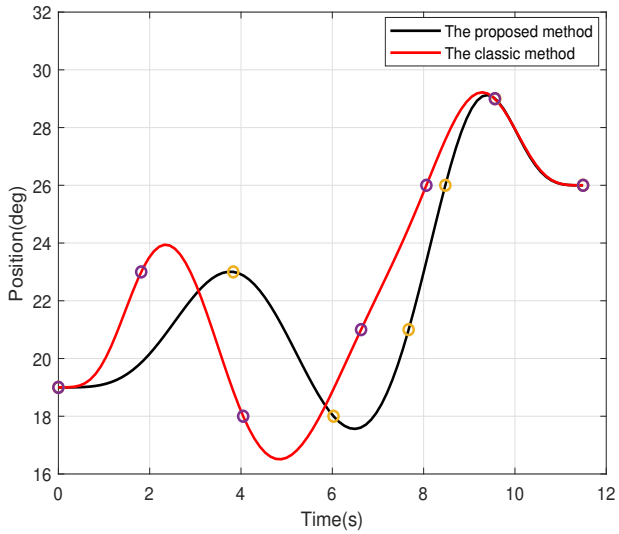
Table 5.7: The joints kinematic indices for both methods.

| | | Number of joint | | | | | |
|---|---|---|---|---|---|---|---|
| | | J1 | J2 | J3 | J4 | J5 | J6 |
| Max. absolute velocity (deg/s) | proposed method | 6.46 | 9.42 | 6.08 | 4.17 | 6.95 | 4.69 |
| | classic method | 5.76 | 24.4 | 12.63 | 5.5 | 6.05 | 8.68 |
| Max. absolute acceleration (deg/s$^2$) | proposed method | 8.32 | 9.3 | 8.22 | 4.52 | 9.48 | 4.55 |
| | classic method | 10.08 | 37.71 | 23.04 | 7.09 | 9.5 | 13.44 |
| Max. absolute jerk (deg/s$^3$) | proposed method | 16.15 | 18.13 | 16.31 | 10.04 | 18.17 | 9.32 |
| | classic method | 24.48 | 97.94 | 57.86 | 15.25 | 21.21 | 31.47 |

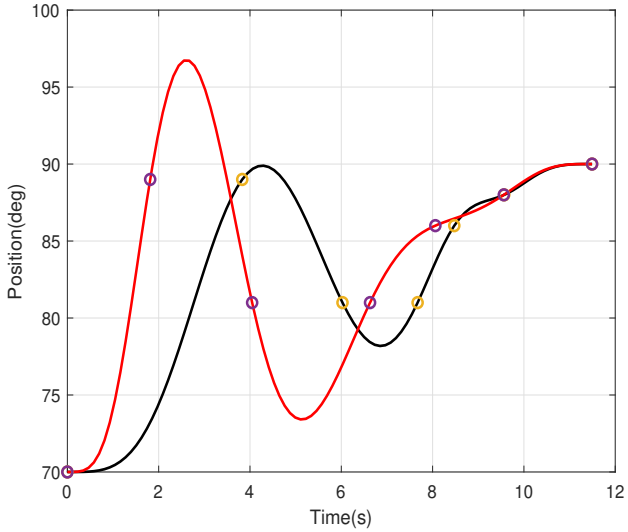
Table 5.8: The indices to optimal trajectory for the two methods.

| | the proposed method | the classical method |
|---|---|---|
| The execution time (s) | 11.49 | 11.49 |
| The energy consuming (N.m.s) | 30513 | 60875 |
| The average absolute jerk (deg/s$^3$) | 14.69 | 41.37 |

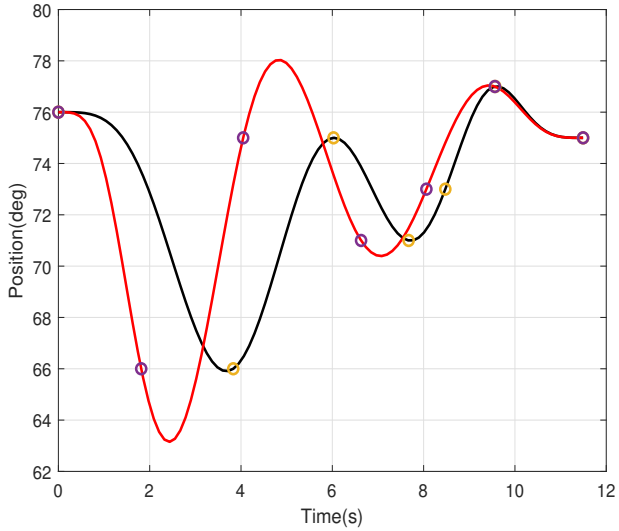
The comparison of the trajectory is shown in Figures 5.16-5.19 in regards position, velocity, acceleration and jerk. Figure 5.20 shows energy consuming comparison.
Figure 5.16 shows the position of the six joints using the proposed approach and the conventional method. The execution time is the same for both methods, and the passage of the trajectory of the six joints over the waypoints is respected, but at different time instants. The movement of the joint departs progressive from the initial point compared to the classical method, and, as is known, the startup phase is important in terms of energy consumption. The velocity is shown in Figure 5.17, which indicates that the variation in velocity resulting from the proposed approach is smoother than of the classical method. It also shows that the maximum velocity of each joint remains within the kinematic constraints for the proposed approach, whereas in the classical method, the velocity exceeds the kinematic limits in the second and third joints. For acceleration, as illustrated in Figure 5.18, in classical method the maximum acceleration value of the first, fourth and fifth joints is within the kinematic constraint limits, while the values for the second, third and sixth joints exceed the constraint limits. So far, we can say that this method does not meet our needs. In contrast, the maximum values of acceleration of the six joints are within the acceleration limits of the kinematic constraints in our proposed method. As is known, jerk is an important criterion in trajectory planning. In Figure 5.19, the jerk result from the classical method have peak values that also exceed the kinematic constraints, which confirms weak of the classical method. In contrast, the jerk results of the proposed method respect the jerk constraints of the robot.
Table 5.7 presents the joints' kinematic indices and the energy consumption for the two methods. The maximum absolute kinematic indices for the proposed method are well within their constraint limits, thus ensuring the safety of the robot during operation. Comparing the energy consumption of the two methods shows that the proposed method consumes less energy than the classical method (see Figure 5.20), where there is reduction of 49.87% energy consuming. Table 5.8 shows the time, jerk and energy indices of the trajectory for the two methods. These tra-



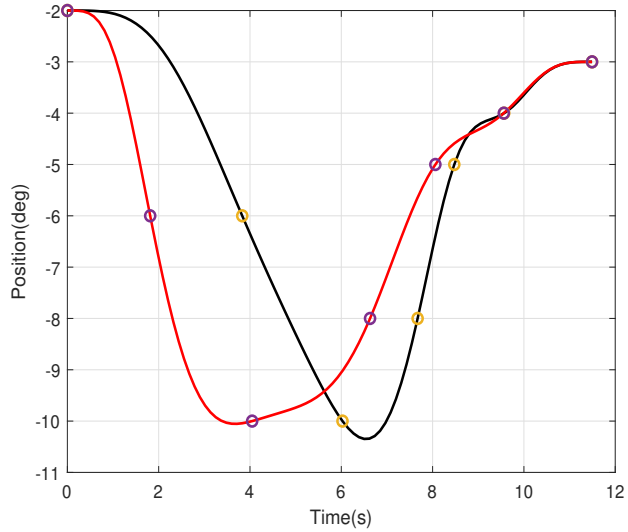

(a) Joint 1



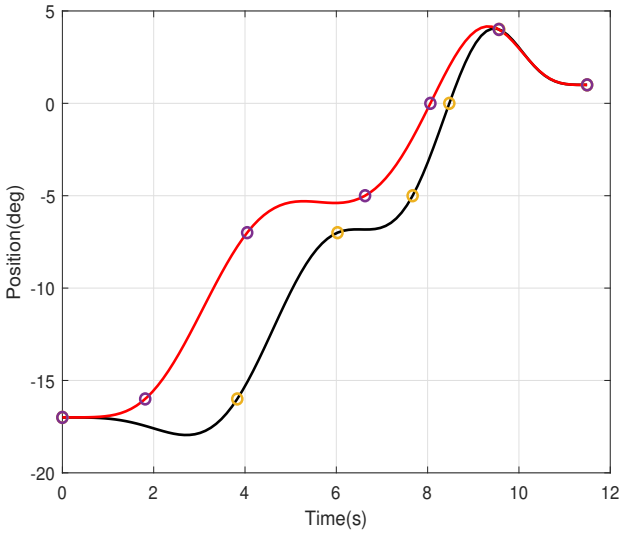

(b) Joint 2



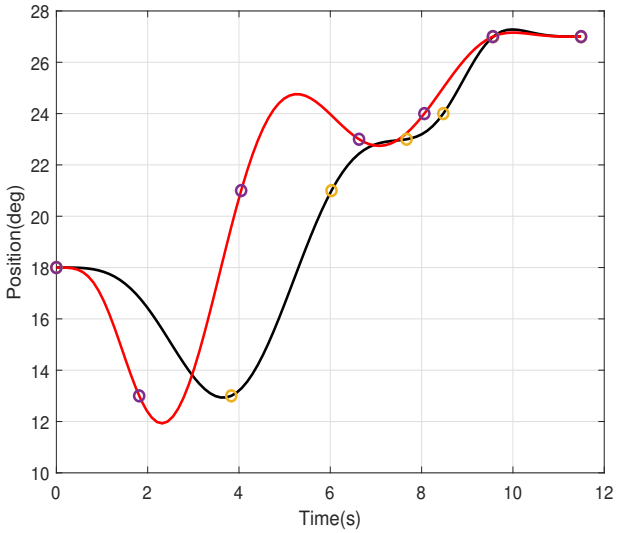

(c) Joint 3




(d) Joint 4

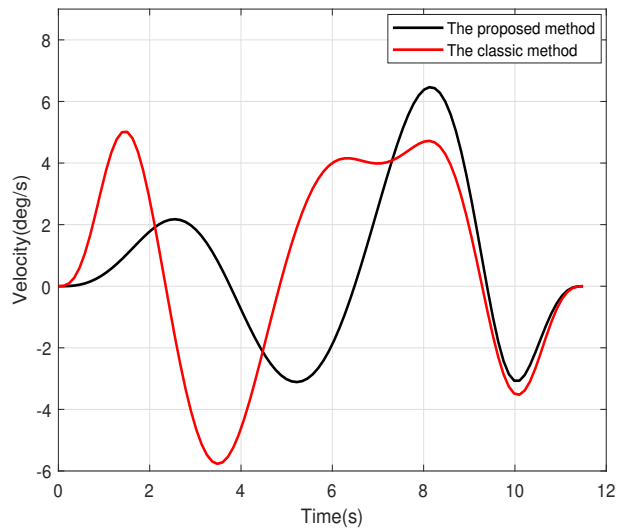



(e) Joint 5

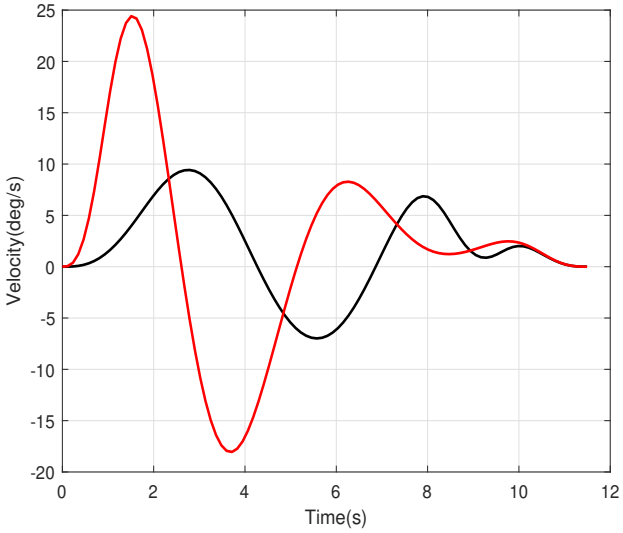



(f) Joint 6

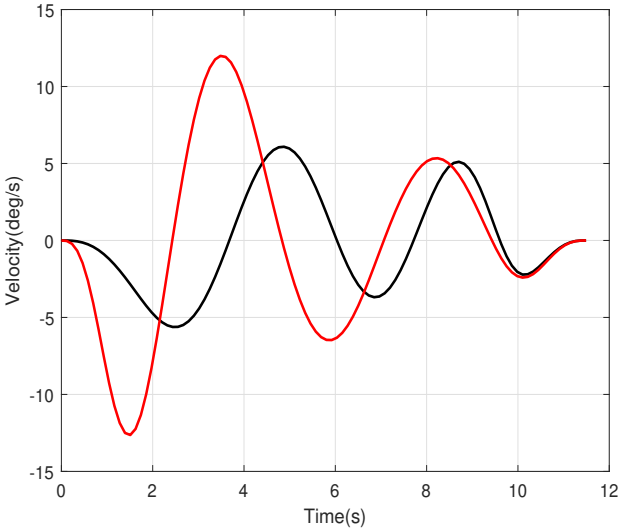
Figure 5.16: Comparison of proposed method and chord length distribution for the position of the six joints of trajectory with an execution time of 11.49s.



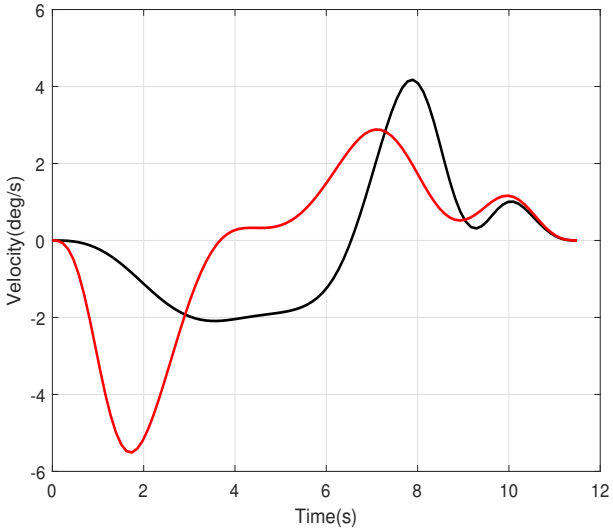

(a) Joint 1



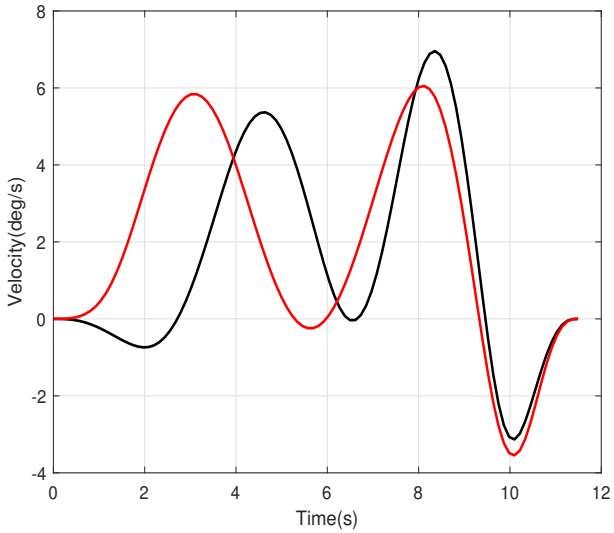

(b) Joint 2



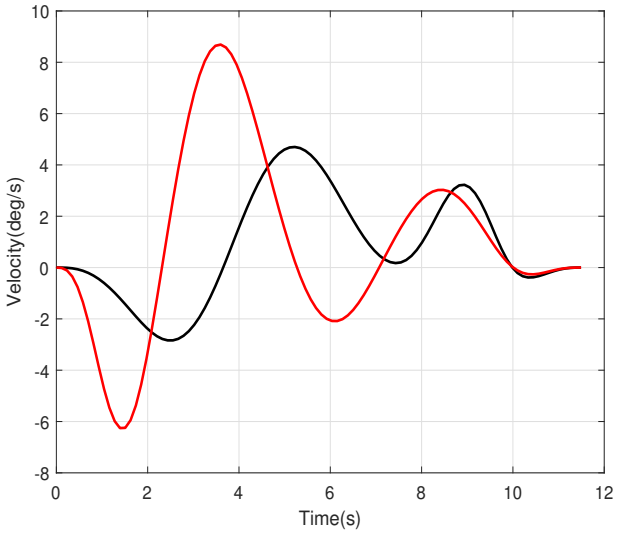

(c) Joint 3




(d) Joint 4

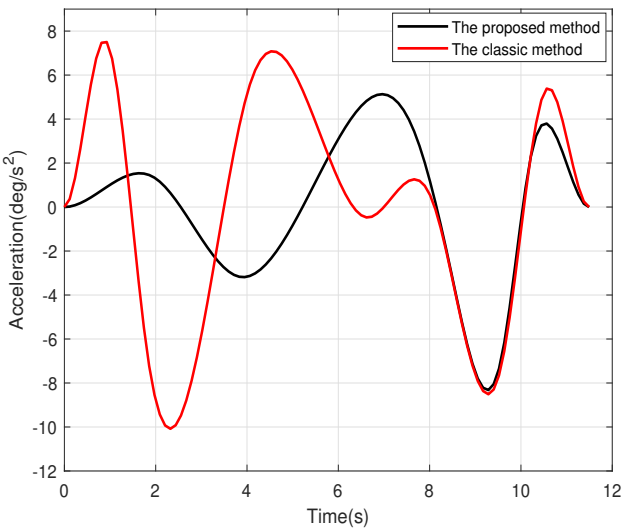



(e) Joint 5

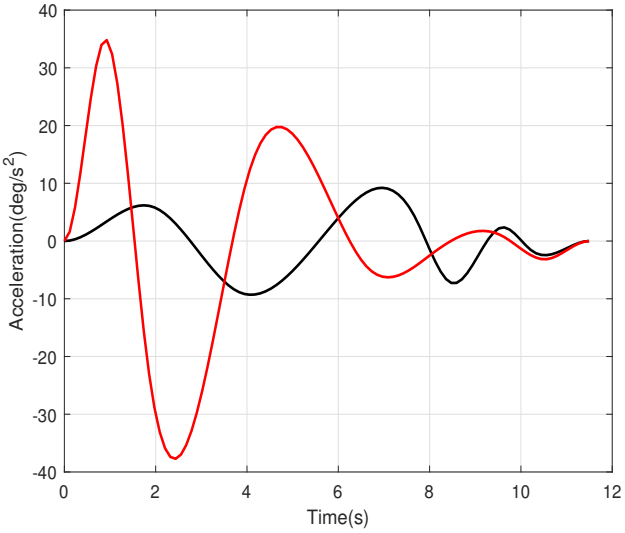



(f) Joint 6

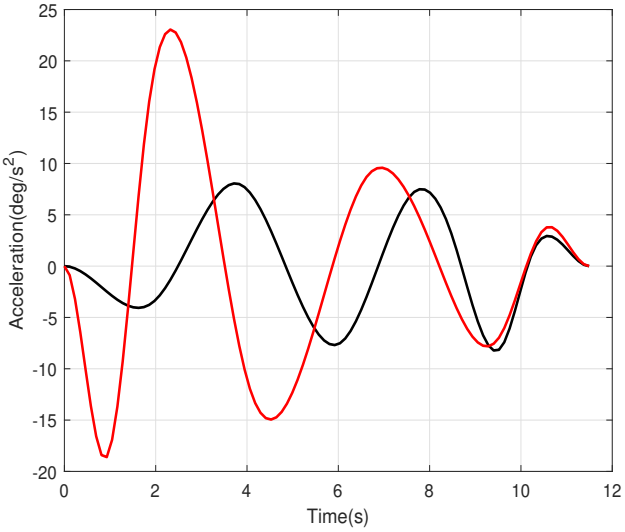
Figure 5.17: Comparison of proposed method and chord length distribution for the velocity of the six joints of trajectory with an execution time of 11.49s.



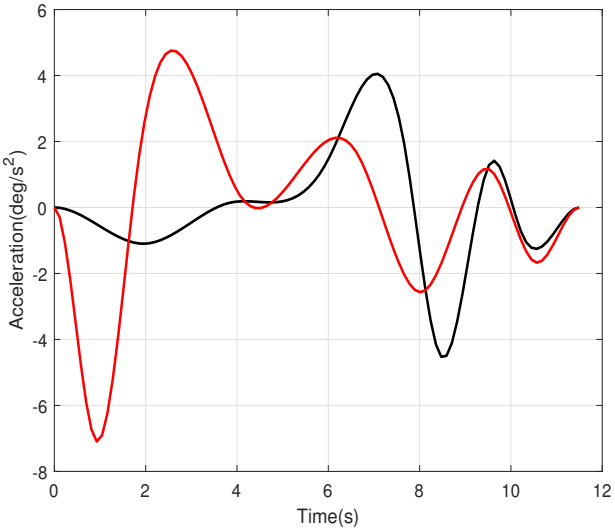

(a) Joint 1



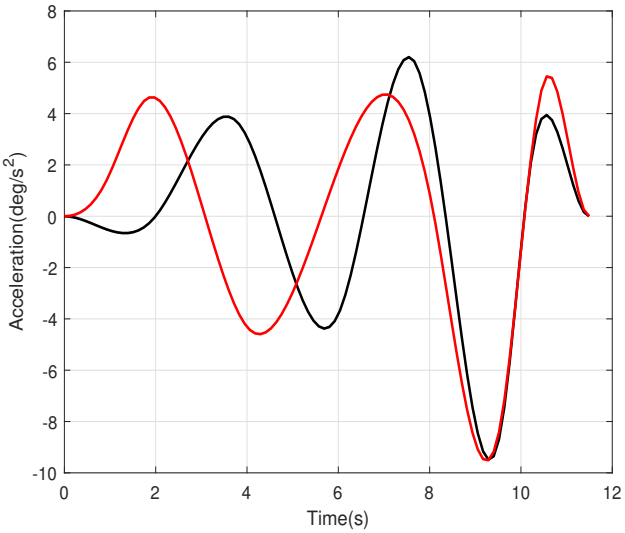

(b) Joint 2



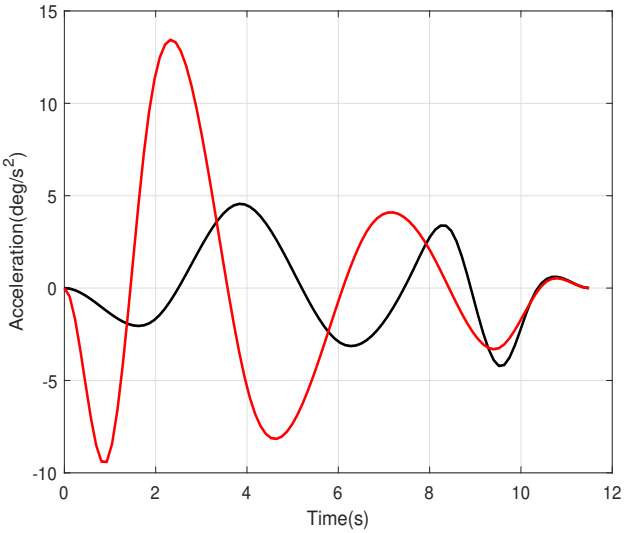

(c) Joint 3




(d) Joint 4

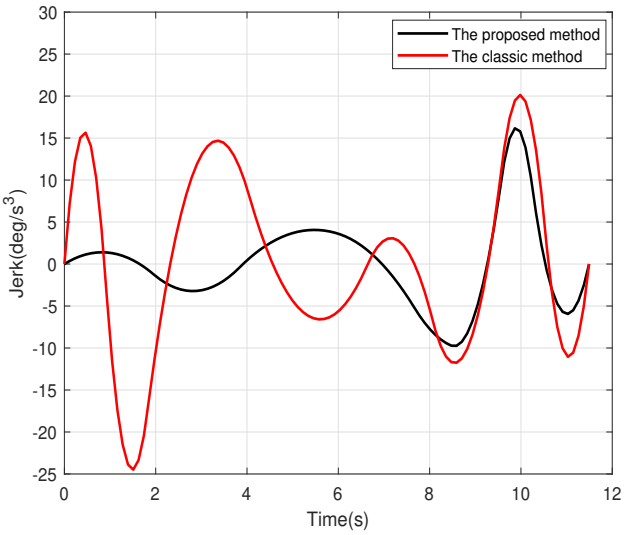



(e) Joint 5

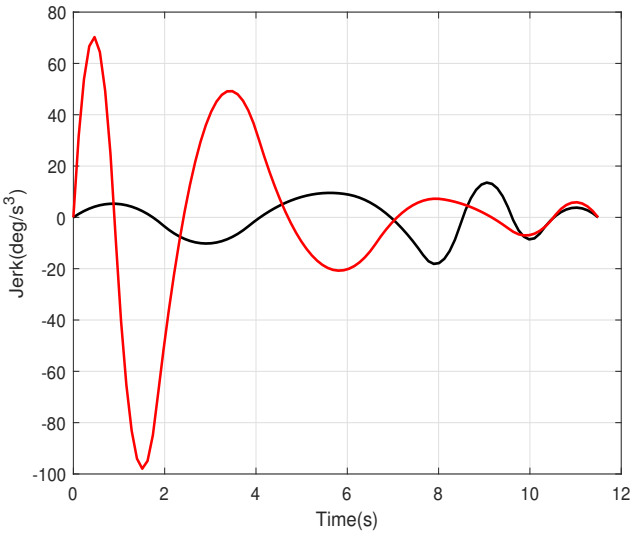



(f) Joint 6

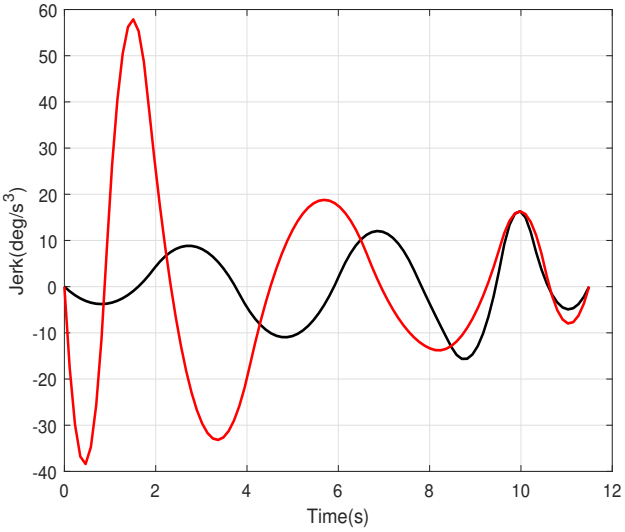
Figure 5.18: Comparison of proposed method and chord length distribution for the acceleration of the six joints of trajectory with an execution time of 11.49s.



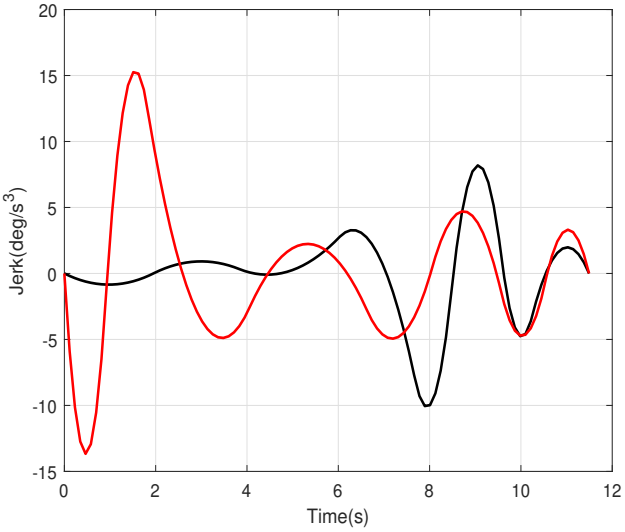

(a) Joint 1



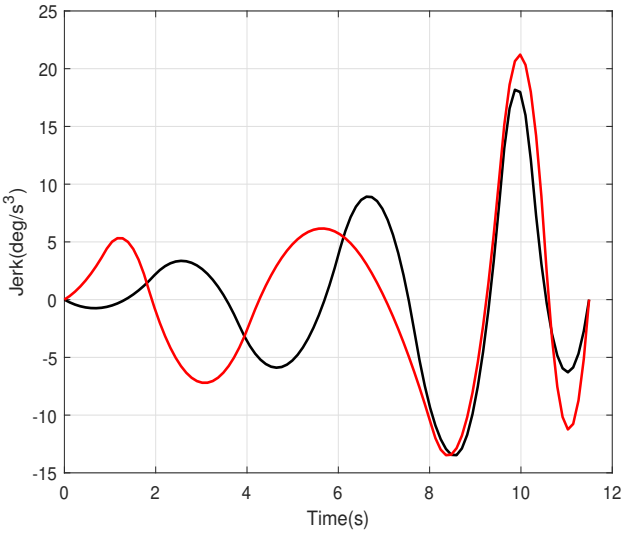

(b) Joint 2



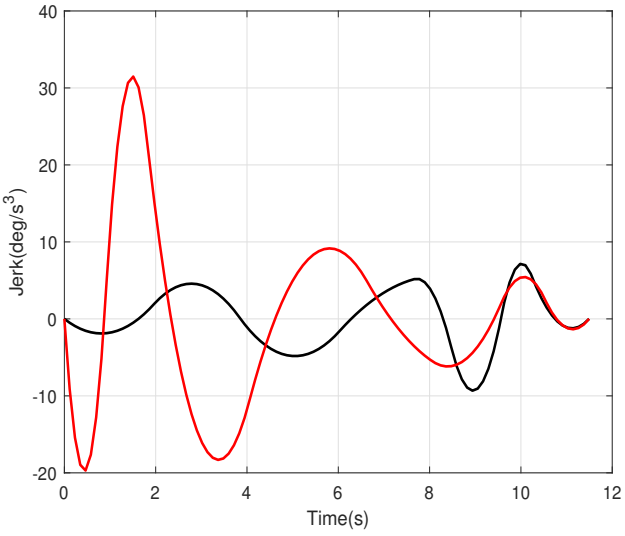

(c) Joint 3




(d) Joint 4




(e) Joint 5




(f) Joint 6

Figure 5.19: Comparison of proposed method and chord length distribution for the jerk of the six joints of trajectory with an execution time of 11.49s.

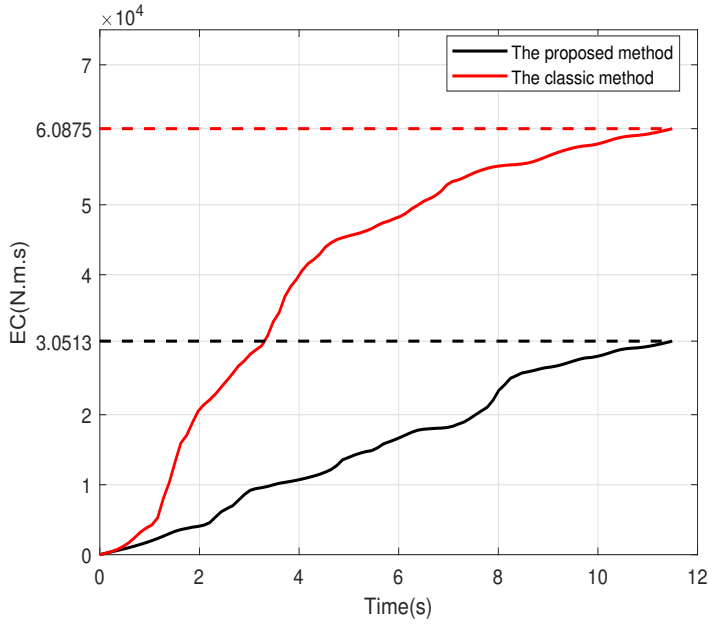



Figure 5.20: Comparison of energy consuming between proposed method and classical method with an execution time of 11.49s .

jectory indices play a significant role in ensuring productivity, flexibility and cost-effectiveness of energy resources during the operation of the manipulator.

## 5.6 Conclusion

This chapter provides an in-depth analysis of various approaches and practical methods for trajectory planning of industrial robots before their real-time implementation. The optimization process considers execution time, jerk, and energy consumption to determine an optimal trajectory that maximizes performance while minimizing vibrations and energy usage, ultimately enhancing the robots efficiency and longevity.

A key challenge addressed in this study is the trade-off between time, jerk, and energy in the absence of an explicit energy expression. To tackle this, the trajectory planning problem is formulated as a multi-objective optimization problem, incorporating execution time, jerk, and energy consumption as objective functions while respecting the robots kinematic constraints. To model energy consumption, a deep learning approach based on Long Short-Term Memory (LSTM) networks is employed. The optimization problem is then solved using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). Finally, the proposed method is compared with the classical chord length distribution technique under the same execution time constraints, demonstrating its superiority in achieving a more efficient trajectory.

# Chapter 6

# Conclusion

This thesis explores the broad research field of industrial robotics, with a particular focus on expanding the workspace of robotic manipulators in industrial settings through the use of soft computing techniques. While promising results have been reported in the literature, the application of soft computing in robotic manipulation remains an evolving research area. It is still in its early stages of development, characterized by a lack of maturity and several open research challenges that need to be addressed.

This manuscript, in particular, presents contributions in three key areas of industrial robotics. First, it addresses the designing of control strategies for accurate trajectory tracking in the presence of disturbances. Second, it explores the problem of collision detection between robots, their environment and human operators using fuzzy logic-based techniques. Finally, it tackles the challenge of optimal trajectory planning, considering execution time, jerk, and energy consumption using deep learning techniques.

After establishing a general introduction and defining the research context in **chapter 1**, **chapter 2** presents a comprehensive literature review and state-of-the-art analysis on soft computing techniques applied to industrial robotics. This chapter explores recent studies in the field, providing an in-depth understanding of the research landscape. By examining existing approaches and advancements, it identifies key challenges and open research questions that need to be addressed, serving as a foundation for the subsequent contributions of this work.

In **chapter 3**, we introduced the fundamental concepts of serial robot manipulators, exploring various modeling approaches, including forward and inverse kinematics, as well as the dynamic model. To gain deeper insights into industrial robots, we conducted a case study on the Fanuc M-710iC/70 robot. The kinematic modelsboth direct and inversewere analyzed to establish the relationship between joint space and Cartesian space. Additionally, the robots dynamic model was developed using Simscape/Matlab, enabling a more precise simulation of its physical behavior. To improve trajectory tracking performance, we implemented a control strategy based on computed torque control (CTC) combined with a velocity observer. The velocity observer played a crucial role in estimating unmeasured velocities and compensating for disturbances, thereby enhancing the stability and accuracy of the control system. Simulation results confirmed the effectiveness of this approach in maintaining precise trajectory tracking under

dynamic conditions.

**Chapter 4** introduced an alternative sensorless collision detection method for manipulator robots based on a Fuzzy Generalized Momentum Observer (FGMO). This approach extends the traditional generalized momentum observer by incorporating a fuzzy system that dynamically adjusts the observers bandwidth. By doing so, it effectively balances collision sensitivity and peak value reduction, overcoming a key limitation of conventional methods. The FGMO is integrated with a collision threshold, allowing for precise differentiation between external forces and internal torque disturbances. Additionally, the proposed algorithm can accurately pinpoint the specific robot link where a collision occurs. The results confirm the robustness of this approach, demonstrating its effectiveness in detecting both dynamic and quasi-static collisions in robotic manipulators.

**Chapter 5** presents a comprehensive study of various strategies and practical methodologies for trajectory planning in industrial robots prior to their real-time deployment. The optimization framework considers execution time, jerk, and energy consumption to generate an optimal trajectory that enhances performance while minimizing vibrations and energy usage, ultimately improving the robots operational efficiency and lifespan. One of the key challenges tackled in this study is balancing the trade-off between time, jerk, and energy, particularly in the absence of a direct energy expression. To address this, the trajectory planning problem is formulated as a multi-objective optimization task, incorporating execution time, jerk, and energy consumption as objective functions while adhering to the robots kinematic constraints. A deep learning-based Long Short-Term Memory (LSTM) network is utilized to model energy consumption, and the optimization is performed using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). Finally, the proposed approach is benchmarked against the classical chord length distribution method under identical execution time constraints, demonstrating its effectiveness in achieving a more efficient and optimized trajectory.

The current study focuses on a trajectory planning approach where the robot controller utilizes a B-spline method to generate smooth and efficient motion paths. Additionally, the trajectory planner has direct access to the controller, allowing for the adjustment of optimal parameters to enhance performance. However, in real-world industrial applications, most robot manufacturers do not provide direct access to their proprietary control software, limiting the ability to modify or optimize built-in trajectory planning algorithms.

In perspective of this work, our future research will to investigate and identify the underlying trajectory planning models used in commercially available industrial robots. By analyzing these models, we aim to develop an external optimization framework that can work independently of proprietary software while still improving trajectory efficiency. This approach will involve reverse engineering existing trajectory planning mechanisms and exploring data-driven techniques to approximate their behavior. Ultimately, the objective is to integrate advanced optimization strategies into practical industrial settings, ensuring improved execution time, jerk minimization, and energy efficiency without requiring direct modification of the robots internal control system.

# Bibliography


[1] J. Iqbal, R. U. Islam, S. Z. Abbas, A. A. Khan, and S. A. Ajwad, "Automating industrial tasks through mechatronic systems-a review of robotics in industrial perspective.," *Tehnicki vjesnik/Technical Gazette*, vol. 23, no. 3, 2016.

[2] C. R. Asfahl, *Robots and manufacturing automation*. John Wiley & Sons, Inc., 1985.

[3] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st. Springer Publishing Company, Incorporated, 2009.

[4] J. Baek, S. Cho, and S. Han, "Practical time-delay control with adaptive gains for trajectory tracking of robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5682–5692, 2017.

[5] Y. T. Oh and S. I. Han, "Finite-time sliding mode joint positioning error constraint control for robot manipulator in the presence of unknown deadzone," *Journal of Mechanical Science and Technology*, vol. 32, pp. 875–884, 2018.

[6] N. Alibeji and N. Sharma, "A pid-type robust input delay compensation method for uncertain euler–lagrange systems," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2235–2242, 2017.

[7] L. Dulger, M. T. Das, R. Halicioglu, S. Kapucu, and M. Topalbekiroglu, "Robotics and servo press control applications: Experimental implementations," in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, 2016, pp. 102–107.

[8] Y. Song and J. Guo, "Neuro-adaptive fault-tolerant tracking control of lagrange systems pursuing targets with unknown trajectory," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, pp. 3913–3920, 2016.

[9] S. Tong, T. Wang, and Y. Li, "Fuzzy adaptive actuator failure compensation control of uncertain stochastic nonlinear systems with unmodeled dynamics," *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 3, pp. 563–574, 2013.

[10] H. Yildiz, N. Korkmaz Can, O. C. Ozguney, and N. Yagiz, "Sliding mode control of a line following robot," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 11, p. 561, 2020.

[11] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of robotic systems*, vol. 17, no. 5, pp. 233–249, 2000.

[12] P. Tangpattanakul and P. Artrit, "Minimum-time trajectory of robot manipulator using harmony search algorithm," in *2009 6th international conference on electrical engineering/electronics, computer, telecommunications and information technology*, IEEE, vol. 1, 2009, pp. 354–357.

[13] B. J. Martin and J. E. Bobrow, "Minimum-effort motions for open-chain manipulators with task-dependent end-effector constraints," *The international journal of robotics research*, vol. 18, no. 2, pp. 213–224, 1999.

[14] A. Abe, "Minimum energy trajectory planning method for robot manipulator mounted on flexible base," in *2013 9th Asian control conference (ASCC)*, IEEE, 2013, pp. 1–7.

[15] Y. Zhao, Y. Wang, S. A. Bortoff, and D. Nikovski, "Energy-efficient collision-free trajectory planning using alternating quadratic programming," in *2014 American control conference*, IEEE, 2014, pp. 1249–1254.

[16] A. Piazzi and A. Visioli, "Global minimum-jerk trajectory planning of robot manipulators," *IEEE transactions on industrial electronics*, vol. 47, no. 1, pp. 140–149, 2000.

[17] H.-I. Lin, "A fast and unified method to find a minimum-jerk robot joint trajectory using particle swarm optimization," *Journal of Intelligent & Robotic Systems*, vol. 75, pp. 379–392, 2014.

[18] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[19] H. Jang and E. Topal, "A review of soft computing technology applications in several mining problems," *Applied Soft Computing*, vol. 22, pp. 638–651, 2014.

[20] F. Aminzadeh, "Applications of ai and soft computing for challenging problems in the oil industry," *Journal of Petroleum Science and Engineering*, vol. 47, no. 1-2, pp. 5–14, 2005.

[21] S. Datta and P. Chattopadhyay, "Soft computing techniques in advancement of structural metals," *International Materials Reviews*, vol. 58, no. 8, pp. 475–504, 2013.

[22] K. M. Saridakis and A. J. Dentsoras, "Soft computing in engineering design–a review," *Advanced Engineering Informatics*, vol. 22, no. 2, pp. 202–221, 2008.

[23] C. Urrea, J. Kern, and J. Alvarado, "Design and evaluation of a new fuzzy control algorithm applied to a manipulator robot," *Applied Sciences*, vol. 10, no. 21, p. 7482, 2020.

[24] S.-W. Kim, J.-J. Lee, and M. Sugisaka, "Inverse kinematics solution based on fuzzy logic for redundant manipulators," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'93)*, IEEE, vol. 2, 1993, pp. 904–910.

[25] M. Ramos and A. J. Koivo, "Fuzzy logic-based optimization for redundant manipulators," *IEEE transactions on fuzzy systems*, vol. 10, no. 4, pp. 498–509, 2002.

[26] I. Eski, S. Erkaya, S. Savas, and S. Yildirim, "Fault detection on robot manipulators using artificial neural networks," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 115–123, 2011.

[27] N. Kumar, V. Panwar, N. Sukavanam, S. P. Sharma, and J.-H. Borm, "Neural network based hybrid force/position control for robot manipulators," *International Journal of Precision Engineering and Manufacturing*, vol. 12, pp. 419–426, 2011.

[28] H. Su, W. Qi, C. Yang, J. Sandoval, G. Ferrigno, and E. De Momi, "Deep neural network approach in robot tool dynamics identification for bilateral teleoperation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2943–2949, 2020.

[29] Y. Lin, A. S. Wang, E. Undersander, and A. Rai, "Efficient and interpretable robot manipulation with graph neural networks," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2740–2747, 2022. DOI: 10.1109/LRA.2022.3143518.

[30] A. S. Polydoros, L. Nalpantidis, and V. Krüger, "Real-time deep learning of robotic manipulator inverse dynamics," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 3442–3448.

[31] B. Sangiovanni, A. Rendiniello, G. P. Incremona, A. Ferrara, and M. Piastra, "Deep reinforcement learning for collision avoidance of robotic manipulators," in *2018 European Control Conference (ECC)*, IEEE, 2018, pp. 2063–2068.

[32] Y. Xue and J.-Q. Sun, "Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm," *Applied Sciences*, vol. 8, no. 9, p. 1425, 2018.

[33] S. Baressi egota, N. Aneli, I. Lorencin, M. Saga, and Z. Car, "Path planning optimization of six-degree-of-freedom robotic manipulators using evolutionary algorithms," *International journal of advanced robotic systems*, vol. 17, no. 2, p. 1 729 881 420 908 076, 2020.

[34] S. Zaer, N. M. Mirza, S. M. Mirza, and M. Arif, "Cartesian path generation of robot manipulators using continuous genetic algorithms," *Robotics and autonomous systems*, vol. 41, no. 4, pp. 179–223, 2002.

[35] D. A. Souza, J. G. Batista, L. L. dos Reis, and A. B. Júnior, "Pid controller with novel pso applied to a joint of a robotic manipulator," *Journal of the Brazilian society of mechanical sciences and engineering*, vol. 43, no. 8, p. 377, 2021.

[36] P. Duan, Z. Yu, K. Gao, L. Meng, Y. Han, and F. Ye, "Solving the multi-objective path planning problem for mobile robot using an improved nsga-ii algorithm," *Swarm and Evolutionary Computation*, vol. 87, p. 101 576, 2024.

[37] M. N. O. Sadiku, C. Uwakwe C, A. M. Abayomi, and M. Sarhan M, "Soft computing in robotics," *International Journal of Trend in Scientific Research and Development (ijtsrd)*, vol. 06, pp. 864–869, 2022.

[38] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, World Scientific, 1996, pp. 775–782.

[39] L. A. Zadeh, "Making computers think like people [fuzzy set theory]," *IEEE spectrum*, vol. 21, no. 8, pp. 26–32, 1984.

[40] D. Rawat, M. K. Gupta, and A. Sharma, "Intelligent control of robotic manipulators: A comprehensive review," *Spatial Information Research*, vol. 31, no. 3, pp. 345–357, 2023.

[41] K. K. Kumbla and M. Jamshidi, "Control of robotic manipulator using fuzzy logic," in *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, IEEE, 1994, pp. 518–523.

[42] M. M. Abdelhameed, "Enhancement of sliding mode controller by fuzzy logic with application to robotic manipulators," *Mechatronics*, vol. 15, no. 4, pp. 439–458, 2005.

[43] A. Kumar, R. Raj, A. Kumar, and B. Verma, "Design of a novel mixed interval type-2 fuzzy logic controller for 2-dof robot manipulator with payload," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106 329, 2023.

[44] Z. Wang, L. Zou, X. Su, G. Luo, R. Li, and Y. Huang, "Hybrid force/position control in workspace of robotic manipulator in uncertain environments based on adaptive fuzzy control," *Robotics and Autonomous Systems*, vol. 145, p. 103 870, 2021.

[45] M. Van, Y. Sun, S. Mcllvanna, M.-N. Nguyen, M. O. Khyam, and D. Ceglarek, "Adaptive fuzzy fault tolerant control for robot manipulators with fixed-time convergence," *IEEE Transactions on Fuzzy Systems*, vol. 31, no. 9, pp. 3210–3219, 2023.

[46] A. Hentout, A. Maoudj, and M. Aouache, "A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots," *Artificial Intelligence Review*, vol. 56, no. 4, pp. 3369–3444, 2023.

[47] R.-J. Lian, "Grey-prediction self-organizing fuzzy controller for robotic motion control," *Information Sciences*, vol. 202, pp. 73–89, 2012.

[48] D. Szabó and E. G. Szádeczky-Kardoss, "Robotic manipulator path-planning: Cost-function approximation with fuzzy inference system," in *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*, IEEE, 2019, pp. 259–264.

[49] M. Beheshti and A. Tehrani, "Obstacle avoidance for kinematically redundant robots using an adaptive fuzzy logic algorithm," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*, IEEE, vol. 2, 1999, pp. 1371–1375.

[50] F. Dimeas, L. D. Avendaño-Valencia, and N. Aspragathos, "Human-robot collision detection and identification based on fuzzy and time series modelling," *Robotica*, vol. 33, no. 9, pp. 1886–1898, 2015.

[51] M. Crengani, M. Tera, C. Biri, and C. Grjob, "Dynamic analysis of a 7 dof robot using fuzzy logic for inverse kinematics problem," *Procedia computer science*, vol. 162, pp. 298–306, 2019.

[52] J. Dembys, Y. Gao, and G. N. DeSouza, "A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network," in *2019 IEEE international conference on fuzzy systems (FUZZ-IEEE)*, IEEE, 2019, pp. 1–6.

[53] A. H. Mary, T. Kara, and A. H. Miry, "Inverse kinematics solution for robotic manipulators based on fuzzy logic and pd control," in *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, IEEE, 2016, pp. 1–6.

[54] M. Zhao and Y. Dai, "Application of fuzzy ant colony algorithm to robotics arm inverse kinematics problem," *ICIC Express Letters*, vol. 10, no. 1, pp. 43–49, 2016.

[55] K. Shihabudheen and G. N. Pillai, "Evolutionary fuzzy extreme learning machine for inverse kinematic modeling of robotic arms," in *2015 39th National Systems Conference (NSC)*, IEEE, 2015, pp. 1–6.

[56] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan, New york, 1961.

[57] 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.

[58] R. Kang, H. Chanal, T. Bonnemains, S. Pateloup, D. T. Branson, and P. Ray, "Learning the forward kinematics behavior of a hybrid robot employing artificial neural networks," *Robotica*, vol. 30, no. 5, pp. 847–855, 2012.

[59] A. R. Almusawi, L. C. Dülger, and S. Kapucu, "A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242)," *Computational intelligence and neuroscience*, vol. 2016, no. 1, p. 5 720 163, 2016.

[60] X. Sun, "Kinematics model identification and motion control of robot based on fast learning neural network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 12, pp. 6145–6154, 2020.

[61] A. Zhao, A. Toudeshki, R. Ehsani, and J.-Q. Sun, "Data-driven inverse kinematics approximation of a delta robot with stepper motors," *Robotics*, vol. 12, no. 5, p. 135, 2023.

[62] T. H. Lee and C. J. Harris, *Adaptive neural network control of robotic manipulators*. World Scientific, 1998, vol. 19.

[63] C. Liu, Z. Zhao, and G. Wen, "Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators," *Neurocomputing*, vol. 350, pp. 136–145, 2019.

[64] W. He, Y. Dong, and C. Sun, "Adaptive neural impedance control of a robotic manipulator with input saturation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 3, pp. 334–344, 2015.

[65] L. A. Soriano, E. Zamora, J. Vazquez-Nicolas, G. Hernández, J. A. Barraza Madrigal, and D. Balderas, "Pd control compensation based on a cascade neural network applied to a robot manipulator," *Frontiers in Neurorobotics*, vol. 14, p. 577 749, 2020.

[66] H.-T. Nguyen and C. C. Cheah, "Analytic deep neural network-based robot control," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 4, pp. 2176–2184, 2022.

[67] X. Li, H. Gao, L. Xiong, H. Zhang, and B. Li, "A novel adaptive sliding mode control of robot manipulator based on rbf neural network and exponential convergence observer," *Neural Processing Letters*, vol. 55, no. 7, pp. 10 037–10 052, 2023.

[68] L. Bo, T. Wei, C. Zhang, H. Fangfang, C. Guangyu, and L. Yufei, "Positioning error compensation of an industrial robot using neural networks and experimental study," *Chinese Journal of Aeronautics*, vol. 35, no. 2, pp. 346–360, 2022.

[69] Y. Jiang, C. Yang, J. Na, G. Li, Y. Li, and J. Zhong, "A brief review of neural networks based learning and control and their applications for robots," *Complexity*, vol. 2017, no. 1, p. 1 895 897, 2017.

[70] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural Networks*, vol. 13, no. 2, pp. 143–148, 2000, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(99)00103-3. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608099001033.

[71] A. Abe, "Trajectory planning for flexible cartesian robot manipulator by using artificial neural network: Numerical simulation and experimental verification," *Robotica*, vol. 29, no. 5, pp. 797–804, 2011.

[72] R. S. Nair and P. Supriya, "Robotic path planning using recurrent neural networks," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, 2020, pp. 1–5.

[73] J. Wang, J. Liu, W. Chen, W. Chi, and M. Q.-H. Meng, "Robot path planning via neural-network-driven prediction," *IEEE transactions on artificial intelligence*, vol. 3, no. 3, pp. 451–460, 2021.

[74] K.-C. Ying, P. Pourhejazy, C.-Y. Cheng, and Z.-Y. Cai, "Deep learning-based optimization for motion planning of dual-arm assembly robots," *Computers & Industrial Engineering*, vol. 160, p. 107 603, 2021.

[75] R. Matousek, L. Dobrovsky, and J. Kudela, "How to start a heuristic? utilizing lower bounds for solving the quadratic assignment problem," *International Journal of Industrial Engineering Computations*, vol. 13, no. 2, pp. 151–164, 2022.

[76] A. G. Gad, "Particle swarm optimization algorithm and its applications: A systematic review," *Archives of computational methods in engineering*, vol. 29, no. 5, pp. 2531–2561, 2022.

[77] Z. Jiang and Q. Zhang, "Time optimal trajectory planning of five degrees of freedom manipulator based on pso algorithm," in *2022 4th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP)*, IEEE, 2022, pp. 1059–1062.

[78] B. Shi and H. Zeng, "Time-optimal trajectory planning for industrial robot based on improved hybrid-pso," in *2021 40th Chinese Control Conference (CCC)*, IEEE, 2021, pp. 3888–3893.

[79] Y. Du and Y. Chen, "Time optimal trajectory planning algorithm for robotic manipulator based on locally chaotic particle swarm optimization," *Chinese Journal of Electronics*, vol. 31, no. 5, pp. 906–914, 2022.

[80] S. Han, X. Shan, J. Fu, W. Xu, and H. Mi, "Industrial robot trajectory planning based on improved pso algorithm," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1820, 2021, p. 012 185.

[81] M. Juíek, R. Parák, and J. Kdela, "Evolutionary computation techniques for path planning problems in industrial robotics: A state-of-the-art review," *Computation*, vol. 11, no. 12, p. 245, 2023.

[82] L. Yiyang, J. Xi, B. Hongfei, W. Zhining, and S. Liangliang, "A general robot inverse kinematics solution method based on improved pso algorithm," *Ieee Access*, vol. 9, pp. 32 341–32 350, 2021.

[83] D. O. Santos, L. Molina, J. G. Carvalho, E. A. Carvalho, and E. O. Freire, "Modifications of fully resampled pso in the inverse kinematics of robot manipulators," *IEEE Robotics and Automation Letters*, 2024.

[84] Y. Liu, D. Jiang, J. Yun, *et al.*, "Self-tuning control of manipulator positioning based on fuzzy pid and pso algorithm," *Frontiers in Bioengineering and Biotechnology*, vol. 9, p. 817 723, 2022.

[85] H. Tiaiba, M. E. H. Daachi, and T. Madani, "Real-time adaptive super twisting algorithm based on pso algorithm: Application for an exoskeleton robot," *Robotica*, pp. 1–26, 2024.

[86] P. Yue, B. Xu, and M. Zhang, "An improve nonlinear robust control approach for robotic manipulators with pso-based global optimization strategy," *Scientific Reports*, vol. 14, no. 1, p. 21 447, 2024.

[87] Ö. Ekrem and B. Aksoy, "Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106 099, 2023.

[88] A. Vysock, R. Papok, J. afaík, *et al.*, "Reduction in robotic arm energy consumption by particle swarm optimization," *Applied Sciences*, vol. 10, no. 22, p. 8241, 2020.

[89] K. Nonoyama, Z. Liu, T. Fujiwara, M. M. Alam, and T. Nishi, "Energy-efficient robot configuration and motion planning using genetic algorithm and particle swarm optimization," *Energies*, vol. 15, no. 6, p. 2074, 2022.

[90] H. Deng and C. Xie, "An improved particle swarm optimization algorithm for inverse kinematics solution of multi-dof serial robotic manipulators," *Soft Computing*, vol. 25, no. 21, pp. 13 695–13 708, 2021.

[91] F. Liu, H. Huang, B. Li, and F. Xi, "A parallel learning particle swarm optimizer for inverse kinematics of robotic manipulator," *International Journal of Intelligent Systems*, vol. 36, no. 10, pp. 6101–6132, 2021.

[92] S. Dereli and R. Köker, "A meta-heuristic proposal for inverse kinematics solution of 7-dof serial robotic manipulator: Quantum behaved particle swarm algorithm," *Artificial Intelligence Review*, vol. 53, pp. 949–964, 2020.

[93] A. K. Kashyap and D. R. Parhi, "Particle swarm optimization aided pid gait controller design for a humanoid robot," *ISA transactions*, vol. 114, pp. 306–330, 2021.

[94] N. Rokbani, B. Neji, M. Slim, S. Mirjalili, and R. Ghandour, "A multi-objective modified pso for inverse kinematics of a 5-dof robotic arm," *Applied Sciences*, vol. 12, no. 14, p. 7091, 2022.

[95] J. Xiao, S. Liu, H. Liu, M. Wang, G. Li, and Y. Wang, "A jerk-limited heuristic feedrate scheduling method based on particle swarm optimization for a 5-dof hybrid robot," *Robotics and Computer-Integrated Manufacturing*, vol. 78, p. 102 396, 2022.

[96] P. Sutyasadi and M. B. Wicaksono, "Joint control of a robotic arm using particle swarm optimization based h2/h robust control on arduino," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 2, pp. 1021–1029, 2020.

[97] Z. Iklima, A. Adriansyah, and S. Hitimana, "Self-collision avoidance of arm robot using generative adversarial network and particles swarm optimization (gan-pso)," *Sinergi*, vol. 25, no. 2, pp. 141–152, 2021.

[98] X. Li, J. Gu, X. Sun, J. Li, and S. Tang, "Parameter identification of robot manipulators with unknown payloads using an improved chaotic sparrow search algorithm," *Applied Intelligence*, pp. 1–11, 2022.

[99] O. Karahan and H. Karci, "Swarm intelligence based nonlinear friction and dynamic parameters identification for a 6-dof robotic manipulator," *Journal of Intelligent & Robotic Systems*, vol. 108, no. 2, p. 19, 2023.

[100] S. tevo, I. Sekaj, and M. Dekan, "Optimization of robotic arm trajectory using genetic algorithm," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 1748–1753, 2014.

[101] D. P. Garg and M. Kumar, "Optimization techniques applied to multiple manipulators for path planning and torque minimization," *Engineering applications of artificial intelligence*, vol. 15, no. 3-4, pp. 241–252, 2002.

[102] F. J. Abu-Dakka, I. F. Assad, R. M. Alkhdour, and M. Abderahim, "Statistical evaluation of an evolutionary algorithm for minimum time trajectory planning problem for industrial robots," *The international journal of advanced manufacturing technology*, vol. 89, pp. 389–406, 2017.

[103] A. M. Zanchettin, C. Messeri, D. Cristantielli, and P. Rocco, "Trajectory optimisation in collaborative robotics based on simulations and genetic algorithms," *International Journal of Intelligent Robotics and Applications*, vol. 6, no. 4, pp. 707–723, 2022.

[104] F. Wang, Z. Wu, and T. Bao, "Time-jerk optimal trajectory planning of industrial robots based on a hybrid woa-ga algorithm," *Processes*, vol. 10, no. 5, p. 1014, 2022.

[105] A. Rout, D. BBVL, B. B. Biswal, and G. B. Mahanta, "Optimal trajectory planning of industrial robot for improving positional accuracy," *Industrial Robot: the international journal of robotics research and application*, vol. 48, no. 1, pp. 71–83, 2021.

[106] F. Fei, H. Hongjie, and G. Zhongtong, "Application of genetic algorithm pso in parameter identification of scara robot," in *2017 Chinese Automation Congress (CAC)*, IEEE, 2017, pp. 923–927.

[107] Z. Lu, C. Wei, D. Ni, J. Bi, Q. Wang, and Y. Li, "Dynamic parameter identification of modular robot manipulators based on hybrid optimization strategy: Genetic algorithm and least squares method," *Soft Computing*, vol. 28, no. 17, pp. 9991–10 005, 2024.

[108] C. Urrea and J. Pascal, "Design, simulation, comparison and evaluation of parameter identification methods for an industrial robot," *Computers & electrical engineering*, vol. 67, pp. 791–806, 2018.

[109] K. Liu, J. Xia, F. Zhong, and L. Zhang, "Structural parameters identification for industrial robot using a hybrid algorithm," *International Journal of Advanced Robotic Systems*, vol. 19, no. 2, p. 17 298 806 221 082 398, 2022.

[110] J. Zhao, L. Han, L. Wang, and Z. Yu, "The fuzzy pid control optimized by genetic algorithm for trajectory tracking of robot arm," in *2016 12th world congress on intelligent control and automation (WCICA)*, IEEE, 2016, pp. 556–559.

[111] M. Sangdani, A. R. Tavakolpour-Saleh, and A. Lotfavar, "Genetic algorithm-based optimal computed torque control of a vision-based tracker robot: Simulation and experiment," *Engineering Applications of Artificial Intelligence*, vol. 67, pp. 24–38, 2018.

[112] S. Refoufi and K. Benmahammed, "Control of a manipulator robot by neuro-fuzzy subsets form approach control optimized by the genetic algorithms," *ISA transactions*, vol. 77, pp. 133–145, 2018.

[113] K. Saidi, A. Bournediene, and D. Boubekeur, "Genetic algorithm optimization of sliding mode controller parameters for robot manipulator," *International Journal on Emerging Technologies*, vol. 12, no. 2, pp. 119–127, 2021.

[114] W. Boukadida, A. Benamor, and H. Messaoud, "Multi-objective design of optimal sliding mode control for trajectory tracking of scara robot based on genetic algorithm," *Journal of Dynamic Systems, Measurement, and Control*, vol. 141, no. 3, p. 031 015, 2019.

[115] L. Sheng and W. Li, "Optimization design by genetic algorithm controller for trajectory control of a 3-rrr parallel robot," *Algorithms*, vol. 11, no. 1, p. 7, 2018.

[116] A. Eltayeb, G. Ahmed, I. H. Imran, N. M. Alyazidi, and A. Abubaker, "Comparative analysis: Fractional pid vs. pid controllers for robotic arm using genetic algorithm optimization," *Automation*, vol. 5, no. 3, pp. 230–245, 2024.

[117] A. Kukker and R. Sharma, "Stochastic genetic algorithm-assisted fuzzy q-learning for robotic manipulators," *Arabian Journal for Science and Engineering*, vol. 46, no. 10, pp. 9527–9539, 2021.

[118] A. Shrivastava and V. K. Dalla, "Failure control and energy optimization of multi-axes space manipulator through genetic algorithm approach," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 43, no. 10, p. 445, 2021.

[119] R.-A. Sánchez-Sosa and E. Chavero-Navarrete, "Robotic cell layout optimization using a genetic algorithm," *Applied Sciences*, vol. 14, no. 19, p. 8605, 2024.

[120] M. Wang, J. Luo, L. Zheng, J. Yuan, and U. Walter, "Generate optimal grasping trajectories to the end-effector using an improved genetic algorithm," *Advances in Space Research*, vol. 66, no. 7, pp. 1803–1817, 2020.

[121] M. Ölgün and U. Tilki, "Neural network based sliding mode controller with genetic algorithm for two link robot manipulator," *Avrupa Bilim ve Teknoloji Dergisi*, pp. 120–129, 2020.

[122] J. Wang, M. Yang, F. Liang, K. Feng, K. Zhang, and Q. Wang, "An algorithm for painting large objects based on a nine-axis ur5 robotic manipulator," *Applied Sciences*, vol. 12, no. 14, p. 7219, 2022.

[123] T. Qiao, D. Yang, W. Hao, J. Yan, and R. Wang, "Trajectory planning of manipulator based on improved genetic algorithm," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 1576, 2020, p. 012 035.

[124] Z. Wang, Y. Li, K. Shuai, W. Zhu, B. Chen, and K. Chen, "Multi-objective trajectory planning method based on the improved elitist non-dominated sorting genetic algorithm," *Chinese Journal of Mechanical Engineering*, vol. 35, no. 1, p. 7, 2022.

[125] A. Saxena, J. Kumar, and V. K. Deolia, "Optimization of npic controller using genetic algorithm," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 1104, 2021, p. 012 001.

[126] S. lgen, A. Durdu, E. Gülbahçe, A. Çakan, and M. Kalyoncu, "The bees algorithm approach to determining smc controller parameters for the position control of a scara robot manipulator," *Avrupa Bilim ve Teknoloji Dergisi*, no. 33, pp. 267–273, 2022.

[127] S. B. Liu, A. Giusti, and M. Althoff, "Velocity estimation of robot manipulators: An experimental comparison," *IEEE Open Journal of Control Systems*, vol. 2, pp. 1–11, 2022.

[128] Y. Yokose, "Energy-saving trajectory planning for robots using the genetic algorithm with assistant chromosomes," *Artificial Life and Robotics*, vol. 25, no. 1, pp. 89–93, 2020.

[129] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[130] S. Dereli, "A new modified grey wolf optimization algorithm proposal for a fundamental engineering problem in robotics," *Neural Computing and Applications*, vol. 33, no. 21, pp. 14 119–14 131, 2021.

[131] E. Galan-Uribe and L. Morales-Velazquez, "Kinematic optimization of 6dof serial robot arms by bio-inspired algorithms," *IEEE Access*, vol. 10, pp. 110 485–110 496, 2022.

[132] M. Zavar, N. Manouchehri, and A. Safa, "Forward and inverse kinematics of 4-dof scara: Using optimization algorithms," *Journal of Applied Dynamic Systems and Control*, vol. 6, no. 3, pp. 25–34, 2023.

[133] J. Cui, T. Liu, H. Sai, Y. Li, M. Zhu, and Z. Xu, "Randomness-enhanced grey wolf optimizer for inverse kinematics solution of redundant robotic manipulators," in *2023 6th International Conference on Intelligent Robotics and Control Engineering (IRCE)*, IEEE, 2023, pp. 41–48.

[134] M. H. Zafar, S. K. R. Moosavi, and F. Sanfilippo, "Inverse kinematic modelling of a 3-dof robotic manipulator using hybrid deep learning models," *Procedia CIRP*, vol. 120, pp. 213–218, 2023.

[135] A. Shrivastava and V. K. Dalla, "Jerk optimized motion planning of redundant space robot based on grey-wolf optimization approach," *Arabian Journal for Science and Engineering*, vol. 48, no. 3, pp. 2687–2699, 2023.

[136] C. Choubey and J. Ohri, "Optimal trajectory generation for a 6-dof parallel manipulator using grey wolf optimization algorithm," *Robotica*, vol. 39, no. 3, pp. 411–427, 2021.

[137] M. Feng, J. Dai, W. Zhou, H. Xu, and Z. Wang, "Kinematics analysis and trajectory planning of 6-dof hydraulic robotic arm in driving side pile," *Machines*, vol. 12, no. 3, p. 191, 2024.

[138] A. Rezoug, J. Iqbal, and M. Tadjine, "Extended grey wolf optimization–based adaptive fast nonsingular terminal sliding mode control of a robotic manipulator," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 236, no. 9, pp. 1738–1754, 2022.

[139] M. Rahmani, H. Komijani, and M. H. Rahman, "New sliding mode control of 2-dof robot manipulator based on extended grey wolf optimizer," *International Journal of Control, Automation and Systems*, vol. 18, pp. 1572–1580, 2020.

[140] H. Komijani, M. Masoumnezhad, M. M. Zanjireh, and M. Mir, "Robust hybrid fractional order proportional derivative sliding mode controller for robot manipulator based on extended grey wolf optimizer," *Robotica*, vol. 38, no. 4, pp. 605–616, 2020.

[141] Z. Zhou, C. Wang, Z. Zhu, Y. Wang, and D. Yang, "Sliding mode control based on a hybrid grey-wolf-optimized extreme learning machine for robot manipulators," *Optik*, vol. 185, pp. 364–380, 2019.

[142] S. C. Kalshetti and S. Dixit, "Self-adaptive grey wolf optimization based adaptive fuzzy aided sliding mode control for robotic manipulator," *Control and Cybernetics*, vol. 47, no. 4, 2018.

[143] R. J. Salman, H. M. Alwan, M. A. Yousif, and A. M. Abdullah, "Computed torque-nn-gwo dynamic hybrid control of manipulator robotic arm,"

[144] M. A. en and M. Kalyoncu, "Grey wolf optimizer based tuning of a hybrid lqr-pid controller for foot trajectory control of a quadruped robot," *Gazi University Journal of Science*, vol. 32, no. 2, pp. 674–684, 2019.

[145] X. Zhang and Z. Ming, "Trajectory planning and optimization for a par4 parallel robot based on energy consumption," *Applied Sciences*, vol. 9, no. 13, p. 2770, 2019.

[146] M. Y. Silaa, O. Barambones, and A. Bencherif, "Robust adaptive sliding mode control using stochastic gradient descent for robot arm manipulator trajectory tracking," *Electronics*, vol. 13, no. 19, p. 3903, 2024.

[147] B. E. Nyong-Bassey and A. M. Epemu, "Inverse kinematics analysis of novel 6-dof robotic arm manipulator for oil and gas welding using meta-heuristic algorithms," *International Journal on Robotics, Automation and Sciences*, vol. 4, pp. 13–22, 2022.

[148] F. Loucif and S. Kechida, "Optimization of sliding mode control with pid surface for robot manipulator by evolutionary algorithms," *Open Computer Science*, vol. 10, no. 1, pp. 396–407, 2020.

[149] A. Shaar and J. A. Ghaeb, "Optimizing robot motion: A practical control for accurate and low energy-consumption industrial manipulator," in *2024 22nd International Conference on Research and Education in Mechatronics (REM)*, IEEE, 2024, pp. 119–125.

[150] S. Ziamanesh, H. S. Salimi, A. Tavana, and A. A. Ghavifekr, "Parameter tuning of discontinues lyapunov based controller based on the gray wolf optimization algorithm applied to a robotic manipulator," in *2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*, IEEE, 2022, pp. 1–6.

[151] M. H. Zafar, H. B. Younus, S. K. R. Moosavi, M. Mansoor, and F. Sanfilippo, "Online pid tuning of a 3-dof robotic arm using a metaheuristic optimisation algorithm: A comparative analysis," in *International Conference on Information and Software Technologies*, Springer, 2023, pp. 25–37.

[152] M. Bayati, "Using cuckoo optimization algorithm and imperialist competitive algorithm to solve inverse kinematics problem for numerical control of robotic manipulators," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 229, no. 5, pp. 375–387, 2015.

[153] L. Zhang, Y. Wang, X. Zhao, P. Zhao, and L. He, "Time-optimal trajectory planning of serial manipulator based on adaptive cuckoo search algorithm," *Journal of Mechanical Science and Technology*, vol. 35, no. 7, pp. 3171–3181, 2021.

[154] R. Liu and F. Pan, "A multi-objective trajectory planning method of the dual-arm robot for cabin docking based on the modified cuckoo search algorithm," *Machines*, vol. 12, no. 1, p. 64, 2024.

[155] H. Tlijani, A. Jouila, and K. Nouri, "Optimized sliding mode control based on cuckoo search algorithm: Application for 2df robot manipulator," *Cybernetics and Systems*, pp. 1–17, 2023.

[156] Y. Cheng, C. Li, S. Li, and Z. Li, "Motion planning of redundant manipulator with variable joint velocity limit based on beetle antennae search algorithm," *IEEE Access*, vol. 8, pp. 138 788–138 799, 2020.

[157] A. T. Khan, X. Cao, Z. Li, and S. Li, "Evolutionary computation based real-time robot arm path-planning using beetle antennae search," *EAI Endorsed Transactions on AI and Robotics*, vol. 1, e3–e3, 2022.

[158] Z. Li, S. Li, and X. Luo, "Using quadratic interpolated beetle antennae search to enhance robot arm calibration accuracy," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 046–12 053, 2022.

[159] A. H. Khan, S. Li, and X. Luo, "Obstacle avoidance and tracking control of redundant robotic manipulator: An rnn-based metaheuristic approach," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4670–4680, 2020. DOI: 10.1109/TII.2019.2941916.

[160] B. Kou, S. Guo, and D. Ren, "A new method for identifying kinetic parameters of industrial robots," in *Actuators*, MDPI, vol. 11, 2021, p. 2.

[161] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[162] F. Loucif, S. Kechida, and A. Sebbagh, "Whale optimizer algorithm to tune pid controller for the trajectory tracking control of robot manipulator," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 1, p. 1, 2020.

[163] T. Wang, Z. Xin, H. Miao, H. Zhang, Z. Chen, and Y. Du, "Optimal trajectory planning of grinding robot based on improved whale optimization algorithm," *Mathematical Problems in Engineering*, vol. 2020, no. 1, p. 3 424 313, 2020.

[164] Y. Zhou, Z. Li, A. Feng, X. Zhang, and M. Zhu, "A novel hyper-redundant manipulator dynamic identification method based on whale optimization and nonlinear friction model," in *International Conference on Intelligent Robotics and Applications*, Springer, 2023, pp. 80–91.

[165] D. Karaboga *et al.*, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer, Tech. Rep., 2005.

[166] Z. Zhou, J. Zhao, Z. Zhang, and X. Li, "Motion planning of dual-chain manipulator based on artificial bee colony algorithm," in *2023 9th International Conference on Control, Automation and Robotics (ICCAR)*, IEEE, 2023, pp. 55–60.

[167] S. Dereli and R. Köker, "Simulation based calculation of the inverse kinematics solution of 7-dof robot manipulator using artificial bee colony algorithm," *SN Applied Sciences*, vol. 2, no. 1, p. 27, 2020.

[168] A. Jamali, I. Mat Darus, M. Talib, H. Yatim, M. Hadi, and M. Tokhi, "Intelligent tuning of pid controller for double-link flexible robotic arm manipulator by artificial bee colony algorithm," in *Sensor Networks and Signal Processing: Proceedings of the 2nd Sensor Networks and Signal Processing (SNSP 2019), 19-22 November 2019, Hualien, Taiwan*, Springer, 2021, pp. 533–547.

[169] Y. Cui, W. Hu, and A. Rahmani, "A reinforcement learning based artificial bee colony algorithm with application in robot path planning," *Expert Systems with Applications*, vol. 203, p. 117 389, 2022.

[170] J. Muñoz, B. López, F. Quevedo, R. Barber, S. Garrido, and L. Moreno, "Geometrically constrained path planning for robotic grasping with differential evolution and fast marching square," *Robotica*, vol. 41, no. 2, pp. 414–432, 2023.

[171] J. Hernandez-Barragan, J. Plascencia-Lopez, M. Lopez-Franco, N. Arana-Daniel, and C. Lopez-Franco, "Inverse kinematics of robotic manipulators based on hybrid differential evolution and jacobian pseudoinverse approach," *Algorithms*, vol. 17, no. 10, p. 454, 2024.

[172] S. D. Das, V. Bain, and P. Rakshit, "Energy optimized robot arm path planning using differential evolution in dynamic environment," in *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, IEEE, 2018, pp. 1267–1272.

[173] B. Kong, X. Ni, T. Wang, X. Xu, and X. Li, "Simulation of manipulator control model based on improved differential evolution algorithm," in *2023 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, IEEE, 2023, pp. 499–503.

[174] A. T. Sadiq, F. A. Raheem, and N. Abbas, "Ant colony algorithm improvement for robot arm path planning optimization based on d* strategy," *International Journal of Mechanical & Mechatronics Engineering*, vol. 21, no. 1, pp. 96–111, 2021.

[175] Z. Huadong, L. Chaofan, and J. Nan, "A path planning method of robot arm obstacle avoidance based on dynamic recursive ant colony algorithm," in *2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*, IEEE, 2019, pp. 549–552.

[176] S. A. Dahmane, A. Azzedine, and A. Megueni, "Ant colony optimization algorithm based on optimal pid parameters for a robotic arm," *International Journal of Control Systems and Robotics*, vol. 5, 2020.

[177] R. Bansal, M. A. Khanesar, and D. Branson, "Ant colony optimization algorithm for industrial robot programming in a digital twin," in *2019 25th International Conference on Automation and Computing (ICAC)*, IEEE, 2019, pp. 1–5.

[178] P. Wu, Z. Wang, H. Jing, and P. Zhao, "Optimal time–jerk trajectory planning for delta parallel robot based on improved butterfly optimization algorithm," *Applied Sciences*, vol. 12, no. 16, p. 8145, 2022.

[179] H. Q. Cao, H. X. Nguyen, T. N.-C. Tran, H. N. Tran, and J. W. Jeon, "A robot calibration method using a neural network based on a butterfly and flower pollination algorithm," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 4, pp. 3865–3875, 2021.

[180] M. Elsisi, H. G. Zaini, K. Mahmoud, S. Bergies, and S. S. Ghoneim, "Improvement of trajectory tracking by robot manipulator based on a new co-operative optimization algorithm," *Mathematics*, vol. 9, no. 24, p. 3231, 2021.

[181] O. Karahan, H. Karci, and A. Tangel, "Optimal trajectory generation in joint space for 6r industrial serial robots using cuckoo search algorithm," *Intelligent Service Robotics*, vol. 15, no. 5, pp. 627–648, 2022.

[182] H. Zhao, B. Zhang, L. Yang, J. Sun, and Z. Gao, "Obstacle avoidance and near time-optimal trajectory planning of a robotic manipulator based on an improved whale optimisation algorithm," *Arabian Journal for Science and Engineering*, vol. 47, no. 12, pp. 16 421–16 438, 2022.

[183] B. Zhou, Y. Wang, B. Zi, and W. Zhu, "Fuzzy adaptive whale optimization control algorithm for trajectory tracking of a cable-driven parallel robot," *IEEE Transactions on Automation Science and Engineering*, 2023.

[184] H.-C. Huang and C.-C. Chuang, "Artificial bee colony optimization algorithm incorporated with fuzzy theory for real-time machine learning control of articulated robotic manipulators," *IEEE Access*, vol. 8, pp. 192 481–192 492, 2020.

[185] R. D. Al-Dabbagh, A. Kinsheel, S. Mekhilef, M. S. Baba, and S. Shamshirband, "System identification and control of robot manipulator based on fuzzy adaptive differential evolution algorithm," *Advances in Engineering Software*, vol. 78, pp. 60–66, 2014.

[186] M. F. Parra-Ocampo, O. Serrano-Pérez, A. Rodríguez-Molina, *et al.*, "Enhancing the performance in the offline controller tuning of robotic manipulators with chaos: A comparative study with differential evolution," *International Journal of Dynamics and Control*, pp. 1–38, 2024.

[187] X. Meng and X. Zhu, "Autonomous obstacle avoidance path planning for grasping manipulator based on elite smoothing ant colony algorithm," *Symmetry*, vol. 14, no. 9, p. 1843, 2022.

[188] W. Xi, L. Ding, and R. Ma, "System identification for rigid-flexible coupled dynamics model of a cable-driven aerial manipulator," *International Journal of Aerospace Engineering*, vol. 2024, no. 1, p. 2 136 038, 2024.

[189] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[190] Y. Xu, R. Liu, J. Liu, and J. Zhang, "A novel constraint tracking control with sliding mode control for industrial robots," *International Journal of Advanced Robotic Systems*, vol. 18, no. 4, p. 17 298 814 211 029 778, 2021.

[191] P. Rocco, "Stability of pid control for industrial robot arms," *IEEE transactions on robotics and automation*, vol. 12, no. 4, pp. 606–614, 1996.

[192] Z. Song, J. Yi, D. Zhao, and X. Li, "A computed torque controller for uncertain robotic manipulator systems: Fuzzy approach," *Fuzzy sets and systems*, vol. 154, no. 2, pp. 208–226, 2005.

[193] J. Shah, S. Rattan, and B. Nakra, "Dynamic analysis of two link robot manipulator for control design using computed torque control," *International Journal of research in computer applications and Robotics*, vol. 3, no. 1, pp. 52–59, 2015.

[194] J. Viola and L. Angel, "Tracking control for robotic manipulators using fractional order controllers with computed torque control," *IEEE Latin America Transactions*, vol. 16, no. 7, pp. 1884–1891, 2018.

[195] L. A. Soriano, J. d. J. Rubio, E. Orozco, *et al.*, "Optimization of sliding mode control to save energy in a scara robot," *Mathematics*, vol. 9, no. 24, p. 3160, 2021.

[196] S. Islam and X. P. Liu, "Robust sliding mode control for robot manipulators," *IEEE Transactions on industrial electronics*, vol. 58, no. 6, pp. 2444–2453, 2010.

[197] J.-J. E. Slotine, "Sliding controller design for non-linear systems," *International Journal of control*, vol. 40, no. 2, pp. 421–434, 1984.

[198] F. Piltan and N. B. Sulaiman, "Review of sliding mode control of robotic manipulator," *World Applied Sciences Journal*, vol. 18, no. 12, pp. 1855–1869, 2012.

[199] M. Zeinali and L. Notash, "Adaptive sliding mode control with uncertainty estimator for robot manipulators," *Mechanism and Machine Theory*, vol. 45, no. 1, pp. 80–90, 2010.

[200] D. Rawat, M. K. Gupta, and A. Sharma, "Intelligent control of robotic manipulators: A comprehensive review," *Spatial Information Research*, vol. 31, no. 3, pp. 345–357, 2023.

[201] (), [Online]. Available: https://www.fanuc.eu/.

[202] (), [Online]. Available: https://www.mathworks.com/products/simscape-multibody.html.

[203] F. Flacco, T. Kröger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," pp. 338–345, 2012.

[204] J. Ulmen and M. Cutkosky, "A robust, low-cost and low-noise artificial skin for human-friendly robots," pp. 4836–4841, 2010.

[205] A. Cirillo, F. Ficuciello, C. Natale, S. Pirozzi, and L. Villani, "A conformable force/tactile skin for physical human–robot interaction," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 41–48, 2015.

[206] A.-N. Sharkawy, P. N. Koustoumpardis, and N. A. Aspragathos, "Manipulator collision detection and collided link identification based on neural networks," pp. 3–12, 2019.

[207] F. Min, G. Wang, and N. Liu, "Collision detection and identification on robot manipulators based on vibration analysis," *Sensors*, vol. 19, no. 5, p. 1080, 2019.

[208] K. Narukawa, T. Yoshiike, K. Tanaka, and M. Kuroda, "Real-time collision detection based on one class svm for safe movement of humanoid robot," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, IEEE, 2017, pp. 791–796.

[209] F. Dimeas, L. D. Avendaño-Valencia, and N. Aspragathos, *Robotica*, vol. 33, no. 9, pp. 1886–1898, 2015.

[210] X. Jing, W. Guangxin, L. Chongyang, and L. Hong, "Real-time collision detection for manipulators based on fuzzy synthetic evaluation," pp. 777–782, 2016.

[211] F. Caccavale and I. D. Walker, *Observer-based fault detection for robot manipulators*. 1997, vol. 4, pp. 2881–2887.

[212] S. Morinaga and K. Kosuge, "Collision detection system for manipulator based on adaptive impedance control law," vol. 1, pp. 1080–1085, 2003.

[213] S. Haddadin, *Towards safe robots: approaching Asimovs 1st law*. Springer, 2013, vol. 90.

[214] J. Hu and R. Xiong, "Contact force estimation for robot manipulator using semiparametric model and disturbance kalman filter," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 4, pp. 3365–3375, 2017.

[215] S. Makni, M. Bouattour, A. El Hajjaji, and M. Chaabane, "Robust fault tolerant control based on adaptive observer for takagi-sugeno fuzzy systems with sensor and actuator faults: Application to single-link manipulator," *Transactions of the Institute of Measurement and Control*, vol. 42, no. 12, pp. 2308–2323, 2020.

[216] A. De Luca and R. Mattone, "Actuator failure detection and isolation using generalized momenta," vol. 1, pp. 634–639, 2003.

[217] S. Haddadin, A. De Luca, and A. Albu-Schäffer, "Robot collisions: A survey on detection, isolation, and identification," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.

[218] G. Garofalo, N. Mansfeld, J. Jankowski, and C. Ott, "Sliding mode momentum observers for estimation of external torques and joint acceleration," pp. 6117–6123, 2019.

[219] S. Long, X. Dang, S. Sun, Y. Wang, and M. Gui, "A novel sliding mode momentum observer for collaborative robot collision detection," *Machines*, vol. 10, no. 9, p. 818, 2022.

[220] T. Ren, Y. Dong, D. Wu, and K. Chen, "Collision detection and identification for robot manipulators based on extended state observer," *Control Engineering Practice*, vol. 79, pp. 144–153, 2018.

[221] S. A. B. Birjandi, J. Kühn, and S. Haddadin, "Observer-extended direct method for collision monitoring in robot manipulators using proprioception and imu sensing," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 954–961, 2020.

[222] H. Li, R. Li, and J. Zhang, "On the design of extended state observer-based robust finite controller: For underactuated robotic system with multiple sources of uncertainties," *Transactions of the Institute of Measurement and Control*, vol. 43, no. 2, pp. 473–483, 2021.

[223] A. A. Godbole, J. P. Kolhe, and S. E. Talole, "Performance analysis of generalized extended state observer in tackling sinusoidal disturbances," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2212–2223, 2012.

[224] B. Guo, S. Bacha, M. Alamir, A. Hably, and C. Boudinet, "Generalized integrator-extended state observer with applications to grid-connected converters in the presence of disturbances," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 2, pp. 744–755, 2020.

[225] X. Zhou and X. Li, "Trajectory tracking control for electro-optical tracking system based on fractional-order sliding mode controller with super-twisting extended state observer," *ISA transactions*, vol. 117, pp. 85–95, 2021.

[226] Y. Li, Y. Li, M. Zhu, Z. Xu, and D. Mu, "A nonlinear momentum observer for sensorless robot collision detection under model uncertainties," *Mechatronics*, vol. 78, p. 102 603, 2021.

[227] M. Naghdi and M. A. Sadrnia, "A novel fuzzy extended state observer," *ISA transactions*, vol. 102, pp. 1–11, 2020.

[228] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[229] B. Rezali, B. Ibari, M. Hebali, M. Berka, M. Bennaoum, and H. A. Azzedine, "Sensorless robot collision detection based on fuzzy momentum observer," *Transactions of the Institute of Measurement and Control*, p. 01 423 312 241 262 538, 2024.

[230] M. Oberherber, H. Gattringer, and A. Müller, "Successive dynamic programming and subsequent spline optimization for smooth time optimal robot path tracking," *Mechanical Sciences*, vol. 6, no. 2, pp. 245–254, 2015. DOI: \url{10.5194/ms-6-245-2015}.

[231] D. Kaserer, H. Gattringer, and A. Müller, "Nearly optimal path following with jerk and torque rate limits using dynamic programming," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 521–528, 2018. DOI: 10.1109/TRO.2018.2880120.

[232] S. Kucuk, "Maximal dexterous trajectory generation and cubic spline optimization for fully planar parallel manipulators," *Computers & Electrical Engineering*, vol. 56, pp. 634–647, 2016, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2016.07.012.

[233] T. Kunz and M. Stilman, "Time-optimal trajectory generation for path following with bounded acceleration and velocity," *Robotics: Science and Systems VIII*, pp. 1–8, 2012. DOI: https://doi.org/10.7551/mitpress/9816.003.0032.

[234] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1533–1540, 2014. DOI: 10.1109/TRO.2014.2351113.

[235] H. Pham and P. Quang-Cuong, "On the structure of the time-optimal path parameterization problem with third-order constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 679–686. DOI: 10.1109/ICRA.2017.7989084.

[236] H. Pham and P. Quang-Cuong, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 645–659, 2018. DOI: 10.1109/TRO.2018.2819195.

[237] R. Wang, Y. Xie, X. Chen, and Y. Li, "Path-constrained time-optimal motion planning for robot manipulators with third-order constraints," *IEEE/ASME Transactions on Mechatronics*, 2023. DOI: 10.1109/TMECH.2023.3234584.

[238] A. Gasparetto and V. Zanotto, "Optimal trajectory planning for industrial robots," *Advances in Engineering Software*, vol. 41, no. 4, pp. 548–556, 2010. DOI: https://doi.org/10.1016/j.advengsoft.2009.11.001.

[239] Y. Li, T. Huang, and D. G. Chetwynd, "An approach for smooth trajectory planning of high-speed pick-and-place parallel robots using quintic b-splines," *Mechanism and Machine Theory*, vol. 126, pp. 479–490, 2018. DOI: https://doi.org/10.1016/j.mechmachtheory.2018.04.026.

[240] J. Huang, P. Hu, K. Wu, and M. Zeng, "Optimal time-jerk trajectory planning for industrial robots," *Mechanism and Machine Theory*, vol. 121, pp. 530–544, 2018. DOI: https://doi.org/10.1016/j.mechmachtheory.2017.11.006.

[241] S. Lu, B. Ding, and Y. Li, "Minimum-jerk trajectory planning pertaining to a translational 3-degree-of-freedom parallel manipulator through piecewise quintic polynomials interpolation," *Advances in Mechanical Engineering*, vol. 12, no. 3, p. 1 687 814 020 913 667, 2020. DOI: 10.1177/1687814020913667.

[242] Y. Zhang, Z. Sun, Q. Sun, Y. Wang, X. Li, and J. Yang, "Time-jerk optimal trajectory planning of hydraulic robotic excavator," *Advances in Mechanical Engineering*, vol. 13, no. 7, p. 16 878 140 211 034 611, 2021. DOI: 10.1177/16878140211034611.

[243] J. Zhao, X. Zhu, and T. Song, "Serial manipulator time-jerk optimal trajectory planning based on hybrid iwoa-pso algorithm," *IEEE Access*, vol. 10, pp. 6592–6604, 2022. DOI: 10.1109/ACCESS.2022.3141448.

[244] H. Gultekin, S. Gürel, and R. Taspinar, "Bicriteria scheduling of a material handling robot in an m-machine cell to minimize the energy consumption of the robot and the cycle time," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102 207, 2021. DOI: https://doi.org/10.1016/j.rcim.2021.102207.

[245] M. Soori, B. Arezoo, and R. Dastres, "Optimization of energy consumption in industrial robots, a review," *Cognitive Robotics*, 2023. DOI: https://doi.org/10.1016/j.cogr.2023.05.003.

[246] A. D. Rocha, N. Freitas, D. Alemão, M. Guedes, R. Martins, and J. Barata, "Event-driven interoperable manufacturing ecosystem for energy consumption monitoring," *Energies*, vol. 14, no. 12, p. 3620, 2021. DOI: /10.3390/en14123620.

[247] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Substantial capabilities of robotics in enhancing industry 4.0 implementation," *Cognitive Robotics*, vol. 1, pp. 58–75, 2021. DOI: https://doi.org/10.1016/j.cogr.2021.06.001.

[248] G. Carabin and L. Scalera, "On the trajectory planning for energy efficiency in industrial robotic systems," *Robotics*, vol. 9, no. 4, p. 89, 2020.

[249] F. Vidussi, P. Boscariol, L. Scalera, and A. Gasparetto, "Local and trajectory-based indexes for task-related energetic performance optimization of robotic manipulators," *Journal of Mechanisms and Robotics*, vol. 13, no. 2, p. 021 018, 2021.

[250] M. Gadaleta, M. Pellicciari, and G. Berselli, "Optimization of the energy consumption of industrial robots for automatic code generation," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 452–464, 2019. DOI: https://doi.org/10.1016/j.rcim.2018.12.020.

[251] M. Zhang and J. Yan, "A data-driven method for optimizing the energy consumption of industrial robots," *Journal of Cleaner Production*, vol. 285, p. 124 862, 2021. DOI: https://doi.org/10.1016/j.jclepro.2020.124862.

[252] P. Jiang, Z. Wang, X. Li, X. V. Wang, B. Yang, and J. Zheng, "Energy consumption prediction and optimization of industrial robots based on lstm," *Journal of Manufacturing Systems*, vol. 70, pp. 137–148, 2023. DOI: https://doi.org/10.1016/j.jmsy.2023.07.009.

[253] Y. Li, Z. Wang, H. Yang, H. Zhang, and Y. Wei, "Energy-optimal planning of robot trajectory based on dynamics," *Arabian Journal for Science and Engineering*, vol. 48, no. 3, pp. 3523–3536, 2023. DOI: https://doi.org/10.1007/s13369-022-07185-7.

[254] G. Wu, W. Zhao, and X. Zhang, "Optimum time-energy-jerk trajectory planning for serial robotic manipulators by reparameterized quintic nurbs curves," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 235, no. 19, pp. 4382–4393, 2021. DOI: 10.1177/0954406220969734.

[255] W. Chen, H. Wang, Z. Liu, and K. Jiang, "Time-energy-jerk optimal trajectory planning for high-speed parallel manipulator based on quantum-behaved particle swarm optimization algorithm and quintic b-spline," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107 223, 2023. DOI: https://doi.org/10.1016/j.engappai.2023.107223.

[256] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*, pp. 3–27, 2015. DOI: https://doi.org/10.1007/978-3-319-14705-5_1.

[257] A. Gasparetto and V. Zanotto, "A technique for time-jerk optimal planning of robot trajectories," *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 3, pp. 415–426, 2008. DOI: https://doi.org/10.1016/j.rcim.2007.04.001.

[258] A. Gasparetto and V. Zanotto, "A new method for smooth trajectory planning of robot manipulators," *Mechanism and machine theory*, vol. 42, no. 4, pp. 455–471, 2007. DOI: https://doi.org/10.1016/j.mechmachtheory.2006.04.002.

[259] K. J. Kyriakopoulos and G. N. Saridis, "Minimum jerk path generation," in *Proceedings. 1988 IEEE international conference on robotics and automation*, IEEE, 1988, pp. 364–369. DOI: 10.1109/ROBOT.1988.12075.

[260] D. Simon, "Data smoothing and interpolation using eighth-order algebraic splines," *IEEE transactions on signal processing*, vol. 52, no. 4, pp. 1136–1144, 2004. DOI: 10.1109/TSP.2004.823489.

[261] C. De Boor, "On calculating with b-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972, ISSN: 0021-9045. DOI: https://doi.org/10.1016/0021-9045(72)90080-9.

[262] J. Yan and M. Zhang, "A transfer-learning based energy consumption modeling method for industrial robots," *Journal of Cleaner Production*, vol. 325, p. 129 299, 2021. DOI: https://doi.org/10.1016/j.jclepro.2021.129299.

[263] H.-I. Lin, R. Mandal, and F. S. Wibowo, "Bn-lstm-based energy consumption modeling approach for an industrial robot manipulator," *Robotics and Computer-Integrated Manufacturing*, vol. 85, p. 102 629, 2024. DOI: https://doi.org/10.1016/j.rcim.2023.102629.

[264] Y. He, P. Wu, Y. Li, Y. Wang, F. Tao, and Y. Wang, "A generic energy prediction model of machine tools using deep learning algorithms," *Applied Energy*, vol. 275, p. 115 402, 2020. DOI: https://doi.org/10.1016/j.apenergy.2020.115402.

[265] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling with Recurrent Neural Networks*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45. DOI: https://doi.org/10.1007/978-3-642-24797-2_4.

[266] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002. DOI: 10.1109/4235.996017.

[267] L. Piegl and W. Tiller, *The NURBS Book*, second. New York, NY, USA: Springer-Verlag, 1995. DOI: http://dx.doi.org/10.1007/978-3-642-97385-7.