University of Blida 1

Faculty of Technology
Department of Automation and Electrical Engineering

Doctoral Thesis

in Automation and Industrial Computing

Contribution to the intelligent control of robot manipulators

Defended by
Abdelhadi AOUAICHIA

Before the jury composed of:

A. FERDJOUNI	Pr, U. of Blida 1	Chairman
M. L. FAS	MCA, U. of Blida 1	Examiner
A. MADDI	MCA, U. of Blida 1	Examiner
K. BOUDJIT	MCA, USTHB	Examiner
K. KARA	Pr, U. of Blida 1	Supervisor

U. of Blida 1, 01/07/2024

To my parents, my pillars of strength and my anchors in life; to my supportive brothers and sister, who have always been by my side; to my extended family.

To my colleagues, who have inspired me, and to my friends, who have shared my joys and sorrows.

Each one of you has left indelible marks on this work and my life. This achievement is as much yours as it is mine. Thank you.

Acknowledgment

In the name of Allah, the Most Gracious and the Most Merciful, praise be to Allah for providing the strength and patience to navigate through the challenges.

This doctoral thesis, conducted at the Electrical Systems and Remote Control Laboratory (LabSET), Department of Automation and Electrical Engineering, Faculty of Technology, University of Blida 1, was made possible by the guidance of Mr. **Kamel KARA**, Professor at the University of Blida 1, whose scholarly insights and unwavering support have been instrumental in shaping this work.

I extend my sincere appreciation to Mr. **Abdellaziz FERDJOUNI**, Professor at the University Blida 1, for his esteemed presence as the chair of the jury. His academic stature has added immense value to this thesis.

My gratitude goes to Mr. **Mohamed Lamine FAS**, Associate Professor at the University of Blida 1, **Abdelkader MADDI**, Associate Professor at the University Blida 1, and Mr. **Kamel BOUDJIT**, Associate Professor at the University of Science and Technology Houari Boumediene, for their willingness to attend as examiners.

I am grateful to Mr. **Mohamed BENRABAH**, Associate Professor at the University of Science and Technology Houari Boumediene, for his readiness to assist and answer my queries throughout this thesis. His contributions have been invaluable.

Lastly, I extend my warmest thanks to all those who contributed to this humble work. Their support has been instrumental in the completion of this thesis. In the pursuit of knowledge, this thesis stands as a testament to the collective efforts of many.

الملخَّص

تُقدِّم أُطروحة الدُّكتوراه هذه دراسة شاملة حول التَّحكُّم الذَّكي في الرُّوبوتات المُتلاعِبة. حيث تمَّ استخدام تِقنيات مُتقدِّمة مثل الشَّبكات العصبيَّة، وخوارزميَّات التَّحسين، والمَنطق الضَّبابي لتطوير استراتيجيَّات تَحكُّم ذكيّة. تَقترِح هذه الأُطروحة عدَّة تِقنيات تَحكُّم وتُقارنها بطرق التَّحكُّم الموجودة حاليًّا، كما تمَّ تقييم أداءها من حيث دقَّة تتبُّع المسارات المرجعيَّة ورفض الاضطرابات من خلال المُحاكاة والتَّحقُّق التَّجريبي باستعمال نماذج مختلفة من الرُّوبوتات المُتلاعِبة.

تَتمثَّل المساهمة الأولى لهذا البحث في تطوير خوارزميَّة تَحكُّم تنبُّؤي بنموذج الشَّبكة العصبيَّة اعتمادًا على خوارزميَّة أرخميدس للتَّحسين AOA. تعتمد خوارزميَّة التَّحكُّم المُقترَحة على شبكة عصبيَّة متعدِّدة الطَّبقات لتوقُّع قيم مُخرَجات النِّظام المستقبليَّة بدقَّة وتستخدم خوارزميَّة أرخميدس للتَّحسين لحساب معلَّمات التَّحكُّم المُثلَى. كما تمَّ دراسة أداء خوارزميَّة التَّحكُّم المُقترَحة في تتبُّع المسار ورفض الاضطرابات من خلال المحاكاة على روبوت مُتلاعِب ذو درجتين من الحريَّة. حيث يَتم مقارنة خوارزميَّة التَّحكُّم المُقترَحة مع خوارزميَّة التَّحكُّم PID، وخوارزميَّة التَّحكُّم في عزم الدَّوران المَحسوب، وخوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة اعتمادًا على خوارزميَّة التَّحسين القائمة على التَّعلم TLBO، وخوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة اعتمادًا على خوارزميَّة سِرب الجُسيمات للتَّحسين PSO. بالإضافة إلى ذلك، تمَّ برمجة خوارزميَّة التَّحكُّم المُقترَحة باستعمال المُعالِج DSP وذلك من أجل التَّحكُّم في الرُّوبوت المُتلاعِب SCARA ذو ثلاث درجات من الحريَّة، مع إجراء مقارنة مع خوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة اعتمادًا على خوارزميَّة TLBO وخوارزمية التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة اعتمادًا على خوارزميَّة PSO.

تُساعد المساهمة الثَّانية في تعزيز ثبات خوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة ضدَّ الاضطرابات الخارجيَّة والدِّيناميكيَّات غير المُنمذجَة ومصادر عدم اليقين من خلال إدراج التَّحكُّم الفعَّال في رفض الاضطرابات. تَتضمَّن خوارزميَّة التَّحكُّم التَّنبُّؤي شرط التَّكلفة النِّهائيَّة لضمان ثبات التَّحكُّم، بينما تستخدم وحدة التَّحكُّم الفعَّال في رفض الاضطرابات مراقب الحالة المُوسَّعة لتقدير ومُوازنة الاضطرابات الكليَّة. تمَّ إثبات كفاءة طريقة التَّحكُّم المُقترَحة من خلال التَّحقُّق التَّجريبي للتَّحكُّم في الرُّوبوت المُتلاعِب MICO ذو أربع درجات من الحريَّة.

أمَّا في المساهمة الثَّالثة، فإنَّنا نجمع بين التَّحكُّم ذو الأداء المحدَّد وبين خوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة للحفاظ على دِقَّة التَّتبُّع ضمن قيم محدَّدة مُسبقا، ممَّا يُحسِّن بشكل كبير من الاستجابة الانتقاليَّة للنِّظام. كما تمَّ مُقارنة أداء خوارزميَّة التَّحكُّم المُقترَحة مع خوارزميَّة التَّحكُّم التَّنبُّؤي بنموذج الشَّبكة العصبيَّة عن طريق المحاكاة للتَّحكُّم في الرُّوبوت المُتلاعِب MICO ذو أربع درجات من الحريَّة.

تُركِّز المساهمة الرَّابعة في تعزيز خوارزميَّة التَّحكُّم في عزم الدَّوران المحسوب عن طريق إضافة وِحدة تَحكُّم ضَبابي كعنصر غير خطِّي وتستخدم خوارزميَّة أرخميدس للتَّحسين لحساب معلَّمات التَّحكُّم. كما تمَّ تقييم أداء خوارزميَّة التَّحكُّم الضَّبابي في عزم الدَّوران المحسوب المُقترَحة عن طريق المحاكاة للتَّحكُّم في الرُّوبوت المُتلاعِب PUMA 560 ذو سِتِّ درجات من الحريَّة وتمَّ مقارنتُها بخوارزميَّة التَّحكُّم PID وخوارزميَّة التَّحكُّم في عزم الدَّوران المحسوب.

Abstract

This doctoral thesis presents a comprehensive study of the intelligent control of robot manipulators. It employs advanced techniques such as neural networks, optimization algorithms, and fuzzy logic to develop intelligent control strategies. This thesis proposes several control techniques, compares them to existing methods, and evaluates their performances in terms of tracking accuracy of reference trajectories and disturbance rejection through simulations and experimental validations on different models of robot manipulators.

The first contribution of this work is the development of a neural network model predictive controller based on Archimedes Optimization Algorithm (AOA). This proposed controller relies on a feed-forward multi-layer neural network to accurately predict the system's future outputs and employs the Archimedes optimization algorithm to compute optimal control actions. The proposed controller's trajectory tracking and disturbance rejection performances are investigated through simulation on a two degrees of freedom robot manipulator. A comparative study between the proposed control technique, the PID controller, the computed torque controller, the neural network model predictive controller based on the Teaching-Learning-Based Optimization (TLBO), and the neural network model predictive controller based on the Particle Swarm Optimization (PSO) is carried out. Additionally, the proposed control algorithm is implemented on a DSP board to control a three degrees of freedom SCARA robot manipulator and compared to the neural network model predictive control based on the TLBO algorithm and the neural network model predictive control based on the PSO algorithm.

The second contribution enhances the neural network model predictive controller's robustness against external disturbances, unmodeled dynamics, and uncertainties by integrating active disturbance rejection control. The predictive controller incorporates a terminal cost constraint to ensure stability, while the active disturbance rejection controller uses an extended state observer to estimate and compensate for the total disturbances. The efficiency of the suggested control approach is demonstrated through experimental validation to control a four degrees of freedom MICO robot manipulator.

In the third contribution, we combine the prescribed performance control with the neural network model predictive controller to maintain tracking errors within predefined bounds, which significantly improves the system's transient response. The performances of the proposed controller are compared against the neural net-

work model predictive controller in simulation to control a four degrees of freedom MICO robot manipulator.

The fourth contribution focuses on enhancing the computed torque controller by adding a fuzzy controller as a nonlinear element and employs the Archimedes optimization algorithm for the controller's parameters optimization. The performances of the proposed fuzzy computed torque controller are evaluated in simulation, considering the control of a six degrees of freedom PUMA 560 robot manipulator and comparing to the PID and the computed torque controllers.

Résumé

Cette thèse de doctorat présente une étude compréhensive de la commande intelligente des robots manipulateurs. Elle utilise des techniques avancées telles que les réseaux de neurones, les algorithmes d'optimisation et la logique floue pour développer des stratégies de commande intelligentes. Cette thèse propose plusieurs techniques de commande, les compare aux méthodes existantes, et évalue leurs performances en termes de précision de suivi des trajectoires de références et de rejet des perturbations à travers des simulations et des validations expérimentales sur différents modèles de robots manipulateurs.

La première contribution de ce travail est le développement d'une commande prédictive à modèle neuronal basée sur l'algorithme d'optimisation d'Archimède (AOA). Cette commande proposée s'appuie sur un réseau de neurones multicouches pour prédire avec précision les sorties futures du système et utilise l'algorithme d'optimisation d'Archimède pour calculer les actions de commande optimales. Les performances de l'algorithme de commande développé en termes de suivi de trajectoire et de rejet des perturbations sont étudiées en simulation sur un robot manipulateur à deux degrés de liberté. Une étude comparative entre la technique de commande proposée, la commande PID, la commande de couple calculée, la commande prédictive à modèle neuronal en utilisant l'optimisation basée sur l'apprentissage (TLBO) et la commande prédictive à modèle neuronal en utilisant l'optimisation par essaims de particules (PSO) est réalisée. En outre, l'algorithme de commande proposé est implémenté sur une carte DSP pour commander un robot manipulateur SCARA à trois degrés de liberté et comparé à la commande prédictive à modèle neuronal en utilisant l'algorithme TLBO et à la commande prédictive à modèle neuronal en utilisant l'algorithme PSO.

La deuxième contribution permet d'améliorer la robustesse de la commande prédictive modèle neuronal vis-à-vis les perturbations externes, les dynamiques non modélisées et les incertitudes en intégrant une commande active de rejet de perturbation. La commande prédictive utilise une contrainte de coût terminal pour assurer la stabilité, tandis que la commande active de rejet de perturbation utilise un observateur d'état étendu pour estimer et compenser les perturbations totales. L'efficacité de l'approche de commande proposée est démontrée par une validation expérimentale pour commander un robot manipulateur MICO à quatre degrés de liberté.

Dans la troisième contribution, nous combinons la commande de performance prescrite avec la commande prédictive à modèle neuronal pour maintenir les erreurs

de poursuite dans des limites prédéfinies, ce qui améliore considérablement la réponse transitoire du système. Les performances de la commande proposée sont comparées à celles de la commande prédictive à modèle neuronal en simulation pour commander un robot manipulateur MICO à quatre degrés de liberté.

La quatrième contribution concerne l'amélioration de la commande de couple calculée en ajoutant une commande floue comme élément non linéaire et utilisant l'algorithme d'optimisation d'Archimède pour l'optimisation des paramètres de commande. Les performances de la commande de couple calculée floue proposée sont évaluées par simulation en considérant la commande d'un robot manipulateur PUMA 560 à six degrés de liberté et en faisant une comparaison avec la commande PID et la commande de couple calculée.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

FKM	Forward Kinematic Model
DH	Denavit-Hartenberg
IKM	Inverse Kinematic Model
CTC	Computed Torque Control
SMC	Sliding Mode Control
ANN	Artificial Neural Network
NN	Neural Network
MPC	Model Predictive Control
NMPC	nonlinear MPC
NNMPC	Neural Network MPC
AOA	Archimedes Optimization Algorithm
PSO	Particle Swarm Optimization
TLBO	Teaching-Learning-Based Optimization
ADRC	Active Disturbance Rejection Control
ESO	Extended State Observer
PPC	Prescribed Performance Control
PPF	Prescribed Performance Function
DoF	Degrees of Freedom
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
NNMPC-AOA	NNMPC based on AOA
NNMPC-PSO	NNMPC based on PSO
NNMPC-TLBO	NNMPC based on TLBO
PP-NNMPC	Prescribed Performance NNMPC
FCTC	Fuzzy CTC

For models formulations

n	system order
m	dimentional space of the robot

X	end-effector Cartesian position
x,y,z	position coordinates
q, q_i	joint angles, ith joint angle
$\dot{q}$, $\ddot{q}$	dot notation indicates first and second order derivatives
J_m	Jacobian matrix
w_x, w_x, w_x	angular velocities
P	constant parameters
α_{i-1}, a_{i-1}, d_i	DH parameters
$\left[{}_{i}^{i-1}T\right]$	homogeneous transformation matrix between frames i and $i-1$
$M(q)$, M_{ij}	mass and inertia matrix, element of line i and column j of $M(q)$
$V(q)$	centrifugal and Coriolis matrix
$B(q)$	Coriolis matrix
$D(q)$	centrifugal matrix
$\dot{q}\dot{q}$	vector of velocity products
$G(q)$, G_i	gravity vector, element i of $G(q)$
$F(q)$, F_i	friction vector, element i of $F(q)$
τ_d	external disturbances
Δh	system uncertainties
τ, τ_i	input vector of torques, element i of τ
g	gravitational acceleration
u	control signals
y, y_i	system outputs, ith system output
$f_{NN}(.)$	neural network function
χ	input vector of neural network
n_u	number of past control inputs
n_y	number of past outputs
σ_1, σ_2	hidden layer activation function, output layer activation function
W_1, W_2	weight matrices
b_1, b_2	bias vectors
Ω	friction plus external disturbances plus system uncertainties
x_i	state variable
b	coefficient of control signal
$f(.)$	time-varying function
$f_d(.)$	discrete function of f
T_s	sampling time
t	continuous time
k	discrete time index
ψ, φ, R, r	measurments related to the robot
l_i	length of link i of the robot

m_i	mass of link i of the robot
m_L	load mass
r_i	length to center of mass of link i
t_0, t_f	initial time, final time
$q_{i,0}$, $q_{i,f}$	ith angle initial value, ith angle final value
$v_{i,0}$, $v_{i,f}$	ith angle initial velocity, ith angle final velocity
$\alpha_{i,0}$, $\alpha_{i,f}$	ith angle initial acceleration, ith angle final acceleration
I_i	inertia of link i
μ_1, μ_2	friction coefficients

For controllers formulations

N_1	prediction horizon
N_2	prediction horizon
N_u	control horizon
N_s	constraint horizon
κ, κ^*	optimal control sequence, sub-optimal control
$J(.)$	cost function
J^*, J_ε	sub-optimal cost function, prescribed performance cost function
$\min_\kappa\{.\}$	minimization of a given function for κ
Q, R	positive semi-definite matrix, positive definite matrix
Γ_y, $\Gamma_{y_i}(0)$	output-dependent weight matrix, constant minimum value of Γ_y
ref	reference traectories
$\hat{y}(k+i \mid k)$	predicted outputs $k+i$ at time k
Δu	control increment
$\Delta u_{\max}$, Δu^*	limit of the control increment, sub-optimal control increment
$e(.)$	output error
$\underline{y}$, $\bar{y}$	system output lower limitation, system output upper limitation
$\underline{u}$, $\bar{u}$	control lower limitation, control upper limitation
ϵ_i	penalization factor of the slack variables
h	step of prediction
k_p, k_i, k_d	PID parameters
K_p, K_v	CTC parameters
σ^2	variance
λ	weighting factor
x_{n+1}	augmented system state
Θ	derivative of the extened state
$\tilde{x}_i$	estimation of x_i
$\tilde{e}_i$	ith estimation error
b_0	input gain

$g_i(.)$	ith function of extended state observer
u_c	active disturbance rejection control law
u_0	output signal of a feedback controller
μ_e	constant bounding the prediction error
μ_Θ	positive constant
δ_i	gain of the linear extended state observer
w_0	observer bandwidth
$\lambda_0(s)$	characteristic polynomial
ν	factor for the equality constraint
$\eta(.)$	prescribed performance function
η_0, η_∞	PPF initial, PPF final value
γ	PPF minimal convergence speed
ε_i	ith transformed error
$S(\varepsilon)$	error-transformed function
Δe	change rate of the error
R^2	coefficient of determination

For optimization algorithms

P_s	population size, number of objects
D	The dimension of the optimization
s, $s_{\max}$	iteration of optimization, maximum number of iterations
C_1, C_2, C_3, C_4	AOA constants
p_j, p_{best}	position of object j, best object's position
ρ_j, ρ_{best}	density of object j, best object's density
υ_j, υ_{best}	volume of object j, best object's volume
α_j, α_{best}	acceleration of object j, best object's acceleration
α_{j-norm}^{s+1}	normalized acceleration of object j at optimization iteration $s+1$
$rand$	random vector between 0 and 1
p_{rand}	random position
mr	random object
F	flag
TF	transfer operator
d^{s+1}	density factor
ub, lb	normalization limits
c_1, c_2, w, w_d	PSO parameters

Introduction

Control theory, a fundamental discipline in modern engineering, provides designs and analyses of controllers that regulate processes or devices. As a multidisciplinary field, it has applications in various areas, from electrical engineering to economic systems. In robotics, control theory takes on a critical role, ensuring that robotic systems perform their tasks accurately and efficiently. The increasing complexity of modern robotic systems necessitates more sophisticated control strategies, leading to the emergence of intelligent control. This new approach integrates artificial intelligence techniques with traditional control theory, opening up new possibilities for enhancing the performance and capabilities of robotic systems.

Robots have become increasingly important and valuable across various fields of science, engineering, and society. They are often employed to perform tasks that are difficult, dangerous, or tedious for humans. For instance, robots have been utilized in space exploration, underwater surveys, industrial manufacturing, medical surgeries, and military operations [1–5]. Moreover, robots have the potential to enhance human capabilities in areas such as rehabilitation, education, and social interaction [6–8]. As such, they have the potential to improve the quality of life, productivity, and efficiency of many activities and processes.

Technological advancements and the emergence of new needs and challenges have driven the evolution of robotics. The history of robotics can be traced back to ancient times when humans invented mechanical devices to mimic living creatures or perform simple tasks. The modern era of robotics began in the 20th century with the development of the first programmable and electrically powered robots. The industrial revolution of the early 1960s saw the introduction of industrial robots into factories, freeing human operators from risky and harmful tasks [5]. As industrial robots were later incorporated into other types of production processes, new requirements emerged that called for greater flexibility and intelligence in these machines. Since then, robotics has progressed rapidly with the development of sensors, actuators, computers, communication technologies, and artificial intelligence. These innovations have enabled robots to become more flexible, intelligent, and adaptable, capable of performing complex and diverse tasks in various environments. Current trends and requirements in robotics include the development of robots that

can cooperate and collaborate with humans and other robots, learn from data and experience, and exhibit ethical and social behaviors.

Robot manipulators have attracted significant interest due to their wide range of applications, such as manufacturing, healthcare, and space applications. Their adaptability and versatility make them a subject of intense study, and with their ability to perform complex tasks with high precision, they have become indispensable. However, controlling robot manipulators poses certain challenges, such as complex nonlinear dynamics that can be difficult to model or predict accurately, constraints that arise from joint limits or the robot's geometry, external disturbances, and the need for high precision. These challenges necessitate the implementation of innovative advanced control strategies.

Intelligent control represents a significant part of modern control theory, leveraging artificial intelligence techniques to enhance system performance. It combines traditional control theory's strengths with artificial intelligence's flexibility and adaptability. This thesis aims to explore the application of intelligent control techniques to robot manipulators, with a focus on improving their precision and robustness.

- Motivations and control objectives:

The motivation for this work is driven by the need for control strategies that can effectively address the challenges in robot manipulator control, such as non-linearity, uncertainty, and disturbances. Therefore, the objective of this doctoral thesis is to develop control algorithms for robot manipulators using artificial intelligence tools, such as neural networks and meta-heuristic optimization algorithms. Furthermore, the experimental validation of the developed control techniques is necessary to demonstrate their effectiveness.

- Contributions:

The contribution of this thesis is the development of several intelligent control techniques. Firstly, Neural Network Model Predictive Control based on the Archimedes Optimization Algorithm (NNMPC-AOA) introduces a novel approach by using the Archimedes optimization algorithm to solve the neural network model predictive control optimization problem. Secondly, NNMPC with Active Disturbance Rejection Control (NNMPC-ADRC) combines neural network model predictive control with active disturbance rejection control to improve robustness against disturbances and uncertainties. Thirdly, Prescribed Performance NNMPC (PP-NNMPC) ensures predefined performance bounds to enhance tracking accuracy and response time. Finally, Fuzzy Computed Torque Control (FCTC) enhances control precision by combining fuzzy logic control with computed torque control.

The thesis is structured into four chapters. Chapter 1 provides a comprehensive overview of robot manipulators, including their definitions, classifications, modeling, and control techniques. It emphasizes the development of control techniques and the importance of novel approaches to intelligent control that can handle robot systems' complexity, uncertainty, and nonlinearity. Chapter 2 introduces NNMPC based on the Archimedes optimization algorithm with simulation on a two Degrees of Freedom (DoF) robot manipulator and experimental validation on a three DoF SCARA robot. Chapter 3 explores the combination of NNMPC with active disturbance rejection control, offering a stability analysis and experimental validation of the proposed controller on a four DoF MICO robot. Chapter 4 introduces two control techniques: the prescribed performance NNMPC with simulation on a four DoF MICO robot and the FCTC with simulation on a six DoF PUMA manipulator.

Chapter 1

State of the Art in Robot Manipulator Control

1.1 Introduction

This chapter provides a comprehensive overview of robot manipulators, including definitions, classifications, modeling, and control techniques. It focuses on the development of control techniques for robot manipulators, showcasing the limitations and the proposed solutions to overcome the challenges in robotic control. The chapter emphasizes the importance of novel and innovative approaches to intelligent control of robot manipulators. Additionally, it serves as a thorough literature review, setting in perspective developed work in the following chapters.

1.2 Generalities on Robots

1.2.1 Definition

Robots are machines that can perform tasks autonomously or semi-autonomously, often by manipulating objects or interacting with their environment. The design and structure of these machines can vary significantly, reflecting their intended application and functionality. According to the Robot Institute of America (RIA), a robot is:

> *"A reprogrammable, multi-functional manipulator designed to move material, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks."*

1.2.2 Classifications of robots

Depending on the criteria used, robots can be classified into different types. Some common criteria are the domain of operation, the degree of autonomy, the kinematic structure, the intended application area, and the method of control [9]. In this subsection, we will focus on three main types of robots: robot manipulators, mobile robots, and biologically inspired robots. Figure 1.1 depicts a general classification of robots.

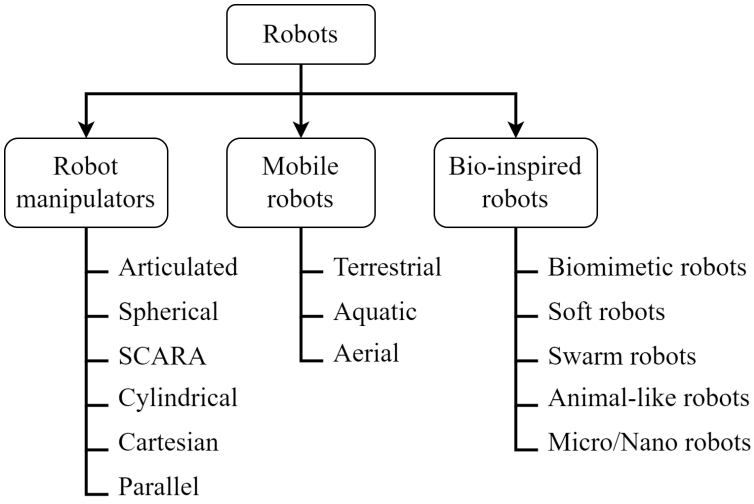


Figure 1.1: Classification of robots

A) Robot manipulators:

Robot manipulators, also known as robot arms, emulate the functions of a human arm, typically consisting of a series of links connected by rotary or linear joints. They can perform tasks such as welding, painting, assembly, and material handling. The evolution of the technical necessities of society and the technological advances achieved have helped the strong growth of new applications in recent years, such as surgery assistance, rehabilitation, and automatic refueling.

Robot manipulators are often fixed to a base or a mobile platform and can have different geometries, such as spherical, cylindrical, Cartesian, articulated, SCARA, or parallel. Some of these types are shown in Figure 1.2.

B) Mobile robots:

Mobile robots are robots that can move in their environment using locomotive elements such as wheels, tracks, legs, propellers, or screws. They can be categorized as terrestrial, aquatic, or aerial robots.

Mobile robots can be classified by their degree of autonomy, such as autonomous mobile robots, automated guided vehicles, or teleoperated robots. They can also

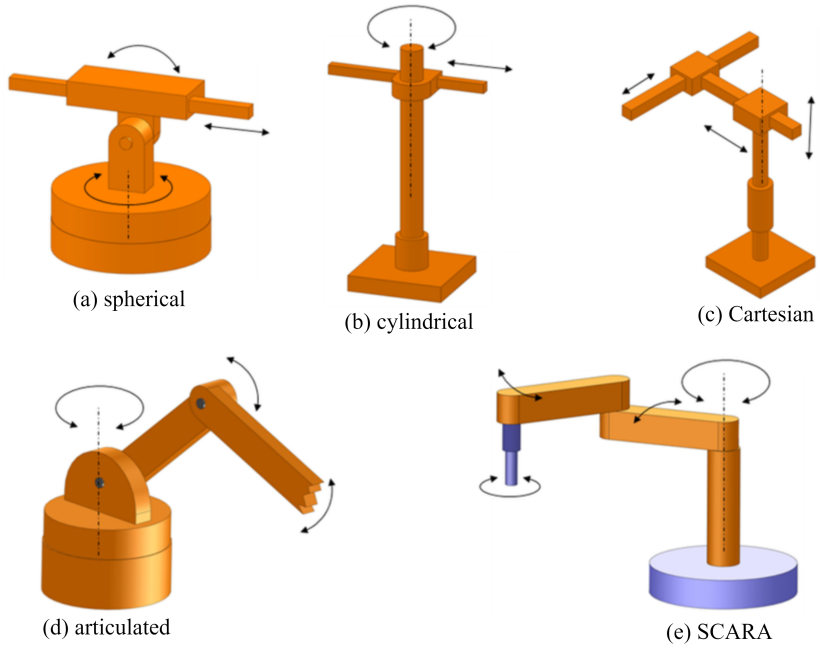


Figure 1.2: Robot manipulators

be classified by their application area, such as service robots, exploration robots, or military robots.

C) Bio-inspired robots:

Biologically inspired robots mimic the structures, behaviors, or functions of living organisms, such as animals, plants, or humans. They can provide novel solutions to complex problems and enhance the performance and adaptability of robotic systems.

Bio-inspired robots can be classified by their biological models, such as insect robots, snake robots, fish robots, bird robots, or humanoid robots. They can also be classified by their research goals, such as understanding biological systems, creating artificial life, or solving engineering challenges.

Robot manipulators are the most common and versatile among these three types of robots, as they can perform a wide range of tasks with high precision and speed. Robot manipulators are also the focus of this thesis, as we will discuss their modeling and control in the following chapters.

1.3 Modeling of Robots

Robot modeling is a crucial process that involves creating a mathematical representation of a robot's structure, kinematics, and dynamics. This process is fundamental for designing, simulating, analyzing, and controlling robotic systems [9]. A well-constructed model can capture the primary features and behaviors of the robot

while abstracting away unnecessary details and complexities. Moreover, modeling can aid in understanding the physical principles and limitations of the robot, thereby optimizing its performance and efficiency.

One of the foundational concepts in robot modeling is the configuration space, which encompasses all possible configurations of the robot. A configuration is a comprehensive specification of the location and orientation of every point on the robot. For instance, a configuration for a robot arm with revolute joints can be described by the joint angles. The configuration space can be represented as a high-dimensional space, where each dimension corresponds to a degree of freedom of the robot.

A related concept is the workspace, which comprises all reachable configurations by the robot's end-effector. The end-effector is the part of the robot that interacts with the environment; it could be a gripper, a tool, or a sensor. The workspace is a subset of the configuration space and depends on the robot's geometry, kinematics, and constraints. It is constrained by the geometry of the manipulator and mechanical constraints on the joints. The workspace can be used to evaluate the robot's reachability, dexterity, and manipulability.

Another significant concept is the task space, which includes all possible positions and orientations of the robot's end-effector relevant to a specific task. For example, for a pick-and-place task, the task space may be defined by the locations of the objects to be picked and placed. The task space, usually a lower-dimensional space than the configuration space, can be mapped to the configuration space using inverse kinematics.

Based on the concepts mentioned above, the following models represent the robot's kinematics and dynamics:

A) Forward Kinematic Model:

The Forward Kinematic Model (FKM) involves using the kinematic equations of a robot to compute the end-effector's position from specified values of the joint angles [10]. It's a crucial aspect of robot motion planning and control.

One of the most widely used methods for kinematic modeling is the Denavit-Hartenberg (DH) method. This method is a systematic way to define reference frames and homogeneous transformation matrices for each link and joint of the robot. The DH method simplifies the computation of the forward kinematic model, which describes the relationship between the joint variables and the end-effector pose.

B) Inverse Kinematic Model:

The inverse Kinematic Model (IKM) is the mathematical process of calculating

the joint angles needed to place the end-effector at a specific position and orientation [11]. This process is essential for programming a robot to perform tasks.

The inverse kinematic model maps the end-effector pose to the joint variables and is usually more challenging to obtain than the forward kinematic model. This difficulty is especially pronounced for complex and redundant robots. Different methods, including analytical, numerical, and algebraic methods, can be used to solve the inverse kinematics problem.

C) Dynamic Model:

A dynamic model of a robot provides a compact representation of the physical features that influence its dynamics. When modeling a robot as a rigid-body system, its dynamic model is comprised of components that separately describe link connectivity, connecting joints, and link masses and inertias. The dynamic model of the robot represents the robot's behavior in response to external forces, torques, and accelerations [12].

There are various methods of modeling of robots, depending on the required level of detail and accuracy [13]. Some of these methods include:

- Euler-Bernoulli Modeling [14]: it is used in soft robotics, and it helps in characterizing nonlinear deformations.

- Cosserat Modeling [15]: this method is useful for modeling self-controllable variable curvature soft continuum robots.

- Jacobian Modeling [15]: it is essential in controlling the velocity of the end effector of a robotic arm.

- Gibbs-Appel Modeling [16]: it is applied in the dynamic modeling of snake-like robots.

Other models can be derived from the principles of Newtonian mechanics, Lagrangian mechanics, or Hamiltonian mechanics. The common forms of the dynamic model are [17]:

- Newton-Euler equations: These equations are based on Newton's second law of motion and Euler's equation of motion. They are applied to each link of the robot and then solved recursively from the base to the end-effector. The Newton-Euler equations are simple and easy to implement but may be difficult for multi-objective optimization and constraint handling.

• Lagrange equations: These equations are based on the principle of least action and the Lagrangian function, which is the difference between the kinetic and potential energies of the robot. They are applied to the whole robot system and then solved simultaneously for all the joint variables. The Lagrange equations are constraint-free and suitable for multi-objective optimization, but they may be complex and computationally intensive.

With the emergence of artificial intelligence methods, new techniques have been used to model robot dynamics. Artificial neural networks rely on a learning-based approach that is used to approximate the robot's kinematics or dynamics [18–21]. Fuzzy logic has also been used to develop flexible models [22]. However, they may require a large amount of data and training and may lack comprehension and generalization. A detailed discussion of kinematic and dynamic modeling fundamentals can be found in the literature [23, 24].

1.4 Control of Robots Manipulators

One of the primary challenges in robotics is designing effective controllers that can achieve the desired performance and robustness in the presence of uncertainties, disturbances, nonlinearities, and coupling effects. Robot manipulators are complex, multi-input multi-output, and highly nonlinear systems that necessitate sophisticated control techniques to accomplish various tasks.

Depending on the control objective, different schemes can be adopted, such as position control, force control, hybrid position/force control, and impedance control [25]. Position control, the most frequently used control scheme in robotics, aims to track a motion trajectory as closely as possible. While this scheme generally works well, it may encounter difficulties when the robot interacts with the environment. Force control, on the other hand, aims to regulate the contact force between the robot and the environment but may compromise position accuracy. Hybrid position/force control combines the benefits of both position control and force control but requires precise knowledge of the environment geometry and the switching conditions. Impedance control is a more general and flexible scheme that modulates the dynamic relationship between the motion and the force of the robot, allowing it to adapt to various environments and tasks.

The position control usually generates reference trajectories in task space, and then, using the IKM, it is transformed into the joint space. Figure 1.3 shows the general control diagram for robots.

The design of controllers for robot manipulators can be based on linear or nonlinear models of the system [13]. Linear controllers are simpler and easier to

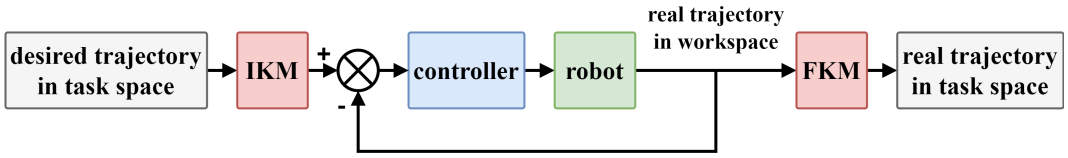


Figure 1.3: Robot general control diagram

implement, but they may not capture the full dynamics of the robot and may suffer from model uncertainties and parameter variations. Nonlinear controllers can account for the nonlinearities and coupling effects of the robot, achieving better performance and robustness, but they may require more computational resources and more accurate models.

The following subsections will review some of the classical and advanced control techniques for robot manipulators, as well as some intelligent control methods that use machine learning and optimization tools to enhance the control capabilities.

1.4.1 Classical control techniques

Some of the most common classical control techniques for robot manipulators are:

- Proportional-Integral-Derivative:

Proportional-Integral-Derivative (PID) and Proportional-Derivative (PD) controllers are the simplest and most widely used control schemes in many applications, including in robotics [26–28]. They adjust the control input based on the proportional, derivative, and integral terms of the error signal. These controllers can achieve satisfactory performance for linear and low-order systems, but they may suffer from steady-state errors, parameter tuning difficulties, and poor robustness to uncertainties and disturbances. Moreover, these controllers have a linear form, which may not be suitable for the nonlinear and coupled dynamics of robot manipulators [29, 30]. Despite these limitations, the PID controller is broadly implemented and still in use in industrial robots due to its simple structure, which is easy to implement. With fine-tuning, it can provide acceptable performances. Different techniques to tune the parameters of the PID controller can be found in [27, 31].

- Computed Torque Control:

Computed Torque Control (CTC) is a control technique that decouples the robot dynamics by using the dynamical models to stabilize the system [28]. CTC can achieve asymptotic trajectory tracking and robustness to bounded disturbances, but it requires the exact knowledge of the robot parameters and the desired trajectory. This controller has been widely used and was applied for the PUMA robot in [32, 33]. Furthermore, a combination of PID and CTC was used [34]. Nonetheless, the

CTC still has some limitations since it applies linearization to compute the control input, and it may not be suitable for types of robots with complex and unmodeled dynamics.

- Sliding Mode Control:

Sliding Mode Control (SMC) is a robust control technique that forces the system states to converge to a predefined sliding surface and stay on it, regardless of the uncertainties and disturbances. This control technique has been explored in various studies [35]. SMC can achieve finite-time convergence and high accuracy. Still, it may generate chattering phenomena, which are high-frequency oscillations in the control input that may damage the actuators and sensors [36]. To reduce the chattering, various modifications of SMC have been proposed, such as boundary layer SMC, higher-order SMC, and terminal SMC [37–43].

1.4.2 Advanced control techniques

Besides the classical control techniques, such as PID and CTC, there are some advanced control techniques that can deal with the nonlinearities, uncertainties, and disturbances in robotic systems more effectively. These techniques are based on fuzzy logic, neural networks, adaptive control, robust control, model predictive control, optimization techniques, active disturbance rejection control, and prescribed performance control. This subsection provides a brief introduction to these techniques and their applications to robot control.

- Fuzzy Logic Control:

Fuzzy logic control is a type of approach that seeks to mimic the way humans think and make decisions by creating a set of rules that are utilized by the controller to analyze the input and determine the appropriate output. It can handle the imprecision of the system parameters and the environment and does not require a precise mathematical model of the system [44].

The primary steps of designing a fuzzy logic controller are as follows: First, the inputs and outputs of the controller are defined. While there are no universal rules for selecting the controller inputs, it's common to use the states of the system, the errors, and the variation of errors. Second, the fuzzification step is initiated. This involves the use of rules expressed in linguistic terms to map the precise values of inputs to appropriate linguistic values. These rules are then evaluated to obtain fuzzy control action. Third, a defuzzification step is necessary to derive a crisp value for the control action. This step ensures that the output of the controller is a definite value that can be applied to the system. Figure 1.4 shows the main steps of the design of fuzzy logic controllers.

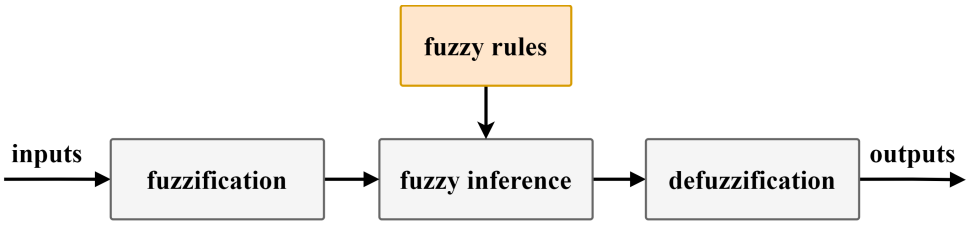


Figure 1.4: Standard fuzzy logic controller structure

Fuzzy control has been combined with other control strategies to enhance the performance and robustness of the controller [45–49]. For example, a supervisory fuzzy controller was employed to find the parameters of the PID controller for a two link planar robot [50]. Moreover, in [51], the CTC was combined with Fuzzy Inference Systems to compensate for the system's nonlinearities. Furthermore, fuzzy SMC can reduce the chattering phenomenon and improve the tracking accuracy [44, 52].

- Neural Network Control:

Neural network control is a technique that uses Artificial Neural Networks (ANNs) to model, identify, and control the nonlinear and uncertain systems [53–55]. ANNs are composed of interconnected processing units that can learn from data and approximate any nonlinear function. Neural network control can be classified into different types according to the structure (such as feedforward and recurrent) and the learning algorithm of the network (such as supervised, unsupervised, and reinforcement). Neural network control can also be combined with other control techniques to improve the performance and robustness of the controller. For example, neural PID control can use Neural Networks (NN) to estimate the system dynamics and compensate for the uncertainties and disturbances [56]. Neural SMC can use NN to approximate the unknown nonlinear functions and reduce the chattering and the control effort [57, 58]. Furthermore, neural fuzzy control can use NN to learn the fuzzy rules and membership functions and enhance the adaptability and accuracy of the controller [59, 60].

- Adaptive Control:

Adaptive control is an approach that can adjust the controller parameters according to the variations of the system dynamics and the environment. Adaptive control can cope with the uncertainties and disturbances of the system and achieve the desired performance and stability [61]. Adaptive control can be classified into different types according to the adaptation mechanism and the control objective. Adaptive control can also be integrated with other control techniques to enhance the performance and robustness of the controller. For example, improvements have been made to the traditional controllers in order to overcome their limitations, such

in [62–64], where the PID and CTC were used with adaptive strategies. Adaptive NN control can use NN to approximate the unknown system dynamics and update the weights online [65]. Adaptive fuzzy control can use fuzzy logic to represent the nonlinear functions and update the fuzzy rules and membership functions online [66].

- Robust Control:

Robust control is a control technique that can guarantee the performance and stability of the system in the presence of uncertainties and disturbances. Robust control can cope with the bounded variations of the system parameters and the external perturbations and does not require precise knowledge of the system dynamics [67]. Robust control can also be combined with other control techniques to enhance the performance and robustness of the controller, such as in [68], where a robust SMC is proposed. Robust NN control can use NN to estimate the uncertainties and disturbances and compensate for them [69, 70]. Additionally, robust fuzzy control can use fuzzy logic to represent the nonlinear functions and attenuate the uncertainties and disturbances [71, 72].

- Model Predictive Control:

Model Predictive Control (MPC) is considered one of the most advanced and successful control strategies that have been developed both within the research control community and in the industry. This success is attributed to the fact that MPC can control a wide range of relatively simple to complex multi-variable processes, such as micro-grids [73], power management of electric vehicles [74], quadrupeds [75], agriculture [76], and HVAC systems [77]. This control strategy solves an optimization problem at each time step to compute the optimal control input. The foundational framework of MPC, also known as Receding Horizon Control (RHC) [78, 79], was laid in the 1970s by [80, 81] with the introduction of Model Predictive Heuristic Control (MPHC) and Dynamic Matrix Control (DMC). It is characterized by the explicit use of a model to predict the process's future outputs over a prediction horizon and then calculate the control sequence that minimizes an objective function. This function takes the form of a constrained nonlinear quadratic function of the error between the model output and the reference trajectory and, in most cases, includes the control effort [82,83].

MPC can handle the system's constraints, nonlinearities, and uncertainties and achieve the desired performance and stability. MPC can be classified into different types according to the model and the optimization method [84]. It provides better performance and robustness compared to traditional controllers. The Generalized Predictive Control (GPC), built on a linearized state-space model, was used for trajectory tracking of a parallel robot in [85]. Other linearization techniques were

explored in [86,87]. However, the linearized model does not fully represent the real system dynamics. As a result, Nonlinear MPC (NMPC) techniques were adopted in robotics [88,89], but its complex model increases the computational cost of MPC. Neural network models, which are effective at capturing the system's nonlinear dynamics, have been used in neural network MPC [90, 91]. The second part of MPC, following the prediction model, is optimization. This research area has seen the development of various techniques. For instance, explicit MPC that employs a pre-computed optimal solution to address the problem was presented in [92]. Dynamic programming has been suggested in [93]. Numerous numerical optimization methods have been utilized to solve the optimization problem of MPC. These include the Particle Swarm Optimization (PSO) and the Teaching-Learning-Based Optimization (TLBO), among others [94–98]. Further improvements of MPC have seen its integration with other control strategies. For example, MPC was combined with H-infinity control in [99], and in [100], the authors proposed MPC with integral compensation (MPC-I) to offset matched uncertainties from the robot's unmodeled dynamics. Moreover, an integral sliding mode compensator was used in [101–103] for robot manipulator control to achieve optimal tracking and robustness.

Various methods have been proposed to prove the stability of model predictive control. The Lyapunov method that uses a proposed Lyapunov function has been employed in several works to demonstrate the stability of MPC [104, 105]. In addition, terminal constraints have been introduced to the optimization problem of MPC in [102,106]. This approach guarantees the convergence of error and thereby proves the stability of MPC. The input-to-state stability method has also been used to ensure the stability of MPC [107–110]. Furthermore, a Lyapunov-based MPC (LMPC) was proposed in other studies [108, 111] where the Lyapunov function is incorporated as a constraint in the objective function.

- Optimization techniques in control:

Optimization is a powerful approach that can find the optimal solution to a problem by minimizing or maximizing a specific objective function subject to some constraints. It can be used to design the optimal controller parameters, trajectory, state feedback, and observer. For instance, PID control can use optimization methods to tune the gains and achieve optimal performance. Boundjou et al. used an adaptive particle swarm optimizer with PID to control a two degrees of freedom arm manipulator [112]. Furthermore, the PID controller parameters were optimized using the genetic algorithm and the PSO to control the PUMA 560 manipulator [113,114]. Similarly, optimization algorithms have been used to design the SMC sliding surface and the reaching law [115,116]. In the case of MPC, meta-heuristic optimization has been used to solve the problem and achieve optimal performance. Although these

methods cannot guarantee global optimality, good solutions can be obtained in a reasonable computational time. Several meta-heuristic algorithms such as Genetic Algorithms (GA) [117, 118] and Evolutionary Algorithms (EA) [119] have been used to solve the optimization problem of MPC. Furthermore, Artificial Bee Colony (ABC) has been used to solve the MPC optimization problem [120, 121].

- Active Disturbance Rejection Control:

Active Disturbance Rejection Control (ADRC), initially proposed by Han [122, 123], emerged as a new robust control method that has gained significant attention due to its innovative concepts, simplicity in implementation, and remarkable performance [124–126]. The strength of ADRC comes from its ability to handle a broad range of uncertainties and disturbances, making it a powerful tool applicable to various practical systems, including robotics [127, 128]. Further works that addressed the application of ADRC in robotics could be found in [129]. The key concept of ADRC resides in approaching all external disturbances along with internal uncertainties as a generalized disturbance that is estimated through an Extended State Observer (ESO) and subsequently compensated for in the control input in real-time. This approach enables ADRC to deal with external disturbances, uncertainties arising from unmodeled dynamics, and parameter uncertainties, enhancing the robustness of the controller.

However, ADRC faces challenges such as the limitation of the ESO bandwidth [130]. This limitation can affect the accuracy of the disturbance estimation and, consequently, the performance of the controller. To address this issue, advanced controllers have been proposed to replace the feedback controller in ADRC, such as the combination of ADRC with joint torque control [131], SMC [132], and fractional order controller [133]. More applications of ADRC in conjunction with predictive control could be found in [134–137]. However, only a few studies have explored the integration of ADRC and MPC within robotic systems, and these have been limited to simulations. For example, in [138], ADRC is paired with MPC for controlling the motion of wheeled mobile robots. The combination of ADRC and MPC has been utilized to control quadrotor systems, as documented in [139, 140]. Moreover, [141, 142] have discussed its application for underwater vehicles.

- Prescribed Performance Control:

Prescribed Performance Control (PPC) has been proven to be a powerful tool that ensures control system outputs/errors with desired transient performance as well as steady-state performance. It was first proposed by Bechlioulis in 2008 [143, 144]. PPC denotes that the tracking error should converge to an arbitrarily predefined small value.

PPC has been integrated with different control schemes, such as adaptive sliding mode prescribed performance control that was introduced in [131]. Various propositions of the recently developed Funnel Model Predictive Control (FMPC) that ensure tracking with a predetermined tracking error performance have been discussed in [145]. In [146], an Adaptive Prescribed Performance Model Predictive Control (APPMPC) approach was presented. Furthermore, another work proposed a noncomplex model predictive control optimization by converting constrained systems into unconstrained ones using the Prescribed Performance Function (PPF) [147]. A prescribed performance-based MPC has been proposed in [148] with an application to a 4 DoF robot manipulator. More applications of PPF in robotics can be found in [149–151].

1.5 Studied Cases

This thesis is dedicated to the exploration of intelligent control for robot manipulators. Various controllers have been developed and implemented on a range of robots. The specific robots considered in this work are the following:

- Two DoF planar robot:

  This is a simple robot with two revolute joints and a prismatic end-effector. It can be used for tasks such as drawing, picking and placing, and welding. The control challenges for this robot include achieving accurate and fast tracking, dealing with nonlinearities and uncertainties, and avoiding model singularities.

- Three DoF SCARA robot:

  This is a widely used industrial robot with three revolute joints and a vertical prismatic joint. It can perform tasks such as assembly, inspection, and packaging. The control challenges for this robot include achieving high precision and repeatability and handling payload variations.

- Four DoF Kinova MICO robot:

  This lightweight and modular robot has four revolute joints and a two-fingered gripper. It can be used for manipulation, rehabilitation, and human-robot interaction. The control challenges for this robot include ensuring stability, achieving high precision, and handling disturbances.

- Six DoF PUMA robot:

  This is a classic robot with six revolute joints and a spherical wrist. It can perform tasks such as machining, painting, and medical surgery. The control challenges for this robot include achieving high precision and flexibility,

coping with complex dynamics and kinematics, and avoiding joint limits and collisions.

1.6 Conclusion

This chapter provided a comprehensive overview of robot manipulator control, including definitions, classifications, modeling, and control techniques. It also showcased the various types of robot manipulators that have been considered in this work. The chapter underscored the challenges and limitations of classical and advanced control methods, thereby highlighting the need for intelligent control techniques capable of handling the complexity, uncertainty, and nonlinearities inherent in robot systems. The chapter emphasizes the importance of novel and innovative approaches to intelligent control of robot manipulators. Additionally, it served as a thorough literature review, offering readers a broad understanding of the field's current state.

Chapter 2

Constrained Neural Network Model Predictive Control Based on Archimedes Optimization Algorithm

2.1 Introduction

In this chapter, a novel neural network model predictive controller based on Archimedes optimization algorithm is introduced. This controller integrates neural networks with model predictive control to accurately predict the system's future outputs; then, it uses the Archimedes optimization algorithm to solve the optimization problem and compute the optimal control action. The performances of the suggested control algorithm are investigated by simulating a two degrees of freedom robot manipulator. The obtained results are compared with those of various techniques, namely the PID controller, the computed torque controller, and the neural network model predictive control using the teaching-learning-based optimization and the particle swarm optimization. To complete the study, the developed controller is implemented on a DSP board and used to control a three degrees of freedom robot manipulator; its results are compared to those of the neural network model predictive control using the teaching-learning-based optimization and the neural network model predictive control based on the particle swarm optimization.

2.2 System Description

Robot kinematics studies the relation between the robot's rigid bodies. The forward kinematics allows us to calculate the position of the end of each joint and ultimately find the position of the end effector. In contrast, inverse kinematics are used to obtain the joints' angles for a specific end-effector position. This conversion can be

solved by obtaining the transformations connecting Cartesian and angular positions.

a) Forward kinematics: Consider an n DoF robot manipulator. The end-effector position in Cartesian coordinates can be denoted by the vector $X = [x, y, z]^T \in \Re^{3\times1}$ and the joint angles are $q = [q_1, \ldots, q_n]^T \in \Re^{n\times1}$. The Jacobian $J_m \in \Re^{m\times n}$ matrix is a mathematical representation that transforms joint velocities into end-effector velocities. This relation is written as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ w_x \\ w_y \\ w_z \end{bmatrix} = J_m(q, P)\dot{q} \tag{2.1}$$

where w_x, w_y, and w_z are the angular velocities, P represents constant parameters, and m equals 3 in two dimensional space and 6 in three dimensional space. The configuration of the robot is assumed to be far from singularities.

To obtain the Jacobian matrix, rotation matrices between the frame and the homogeneous transformation matrices are utilized. The latter defines the relationship between each frame of reference in the robotic arm.

First, the Denavit-Hartenberg table is obtained by defining these parameters for each joint:

- q_i the joint angle between x_{i-1} and x_i, rotating around z_i.
- α_i the link twist between z_i and z_{i+1}, rotating around x_i.
- a_i the link length between z_i and z_{i+1}, measured along x_i.
- d_i the link offset between x_{i-1} and x_i, taken along z_i.

Then, the homogeneous transformation matrix $\left[{}^{i-1}_{i}T\right]$ for a particular link, using the rotation matrices, can be given as:

$$\left[{}^{i-1}_{i}T \right] = \begin{bmatrix} c_i & -s_i & 0 & \alpha_{i-1} \\ s_i c_{\alpha_{i-1}} & c_i c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -d_i s_{\alpha_{i-1}} \\ s_i s_{\alpha_{i-1}} & c_i s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & d_i c_{\alpha_{i-1}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where c_i and s_i is a shorthand for $\cos(q_i)$ and $\sin(q_i)$, respectively.

The complete 4×4 transformation from the base frame to the end-effector frame is obtained by multiplying the individual transformation matrices:

$$\begin{aligned} \left[{}^{0}_{n}T\right] &= \left[{}^{0}_{1}T\right]\left[{}^{1}_{2}T\right]\ldots\left[{}^{n-1}_{n}T\right] \\ &= \begin{bmatrix} a_x & b_x & o_x & x \\ a_y & b_y & o_y & y \\ a_z & b_z & o_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.3}$$

This matrix transforms the angular position at each joint (q_1, q_2, q_3, q_4, q_5, and q_6) and gives the position (x, y, and z) and the orientation matrix of the end effector.

b) Inverse kinematics: The inverse kinematics is solved by multiplying the inverse of each transformation on both sides of Equation (2.3):

$$\left[{}^{4}_{5}T\right]^{-1}\left[{}^{3}_{4}T\right]^{-1}\left[{}^{2}_{3}T\right]^{-1}\left[{}^{1}_{2}T\right]^{-1}\left[{}^{0}_{1}T\right]^{-1}\left[{}^{0}_{6}T\right] = \left[{}^{5}_{6}T\right] \tag{2.4}$$

By solving Equation (2.4), all the unknown variables could be obtained [152].

c) Dynamic model:

The dynamic model, a nonlinear differential equation derived using the Lagrangian approach, describes the motion of the joints in relation to their velocities, accelerations, and the different forces. The standard equation of motion for an n DoF robot manipulator is given by the following equation [9, 32, 153]:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + \tau_d + \Delta h = \tau \tag{2.5}$$

where $M(q) \in \Re^{n\times n}$ denotes the mass and inertia matrix, $V(q,\dot{q}) \in \Re^{n\times n}$ represents the centrifugal and Coriolis matrix, $G(q) \in \Re^{n\times 1}$ denotes the gravity vector, $F(\dot{q}) \in \Re^{n\times 1}$ is the friction vector, $\tau_d \in \Re^{n\times 1}$ are the external disturbances, $\Delta h \in \Re^{n\times 1}$ are system uncertainties, and $\tau \in \Re^{n\times 1}$ represents the input vector of torques at each joint. All with regard to joint angles $q \in \Re^{n\times 1}$ and their acceleration and velocity components $\ddot{q}$ and $\dot{q}$, respectively.

At a given instant k, where $t = kT_s$ with $k \in \aleph$ and T_s is the sampling time, the output of the system denoted as $y(k) = q(k)$, and the control input is $u(k) = \tau(k)$.

2.3 Neural Network Model Predictive Control

2.3.1 Predictive control formulation

Predictive control is a closed-loop control strategy that forecasts the future outputs of the system based on a prediction model over a receding horizon, denoted as $[N_1\ N_2]$, also known as the prediction horizon. At each instant k, an optimal control sequence referred to as $\kappa(k) = [u(k), \ldots, u\,(k + N_u - 1)]$ is obtained by solving an optimization problem online, with the control horizon $N_u < N_2$.

This optimization problem aims to minimize a quadratic cost function J to keep the system as close as possible to a given reference trajectory. The cost function J is usually represented as the squared tracking error between the reference trajectories and the predicted outputs, subject to a set of constraints. In most cases, the control effort is incorporated in J. It can be written as follows:

$$\begin{aligned} J(k) = &\sum_{i=N_1}^{N_2} \left[(ref(k+i) - \hat{y}(k+i \mid k))^T Q (ref(k+i) - \hat{y}(k+i \mid k))\right] \\ &+ \sum_{i=1}^{N_u} \left[(\Delta u(k+i-1))^T R (\Delta u(k+i-1))\right] \end{aligned} \tag{2.6}$$

with $ref(k+i) \in \Re^{n\times 1}$ representing the reference trajectory, $\hat{y}(k+i \mid k)$ is the predicted output at time $k+i$ using the information up to time k, and $\Delta u(k+i) = u(k+i) - u(k+i-1)$ is the control increment. The matrices Q and R are positive semi-definite and positive definite, respectively.

The predictive control can be described in the following steps:

1. First, the system output is measured at instant k.
2. The prediction model of the system is then used to predict the future values of the system outputs over the prediction horizon $[N_1, N_2]$.
3. The optimal control sequence $\kappa(k)$, solution of the following minimization problem, is computed:

$$\begin{aligned} \kappa(k) &= \arg\min_{\kappa} J \\ &= \arg\min_{\kappa} \sum_{i=N_1}^{N_2} \left[(e(k+i \mid k))^T Q (e(k+i \mid k))\right] \\ &\quad + \sum_{i=1}^{N_u} \left[(\Delta u(k+i-1))^T R (\Delta u(k+i-1))\right] \end{aligned} \tag{2.7}$$

Subject to

$$\begin{aligned} \underline{y} \leq \hat{y}(k+i) \leq \bar{y}, \quad & \forall\, i = N_1, \ldots, N_2 \\ \underline{u} \leq u(k+i) \leq \bar{u}, \quad & \forall\, i = 0, 1, \ldots, N_u - 1 \\ |\Delta u(k+i)| \leq \Delta u_{max}, \quad & \forall\, i = 0, 1, \ldots, N_u - 1 \\ \Delta u(k+i) = 0, \quad & \forall\, i > N_u - 1 \end{aligned}$$

where $e(k+i \mid k) = ref(k+i) - \hat{y}(k+i \mid k)$, $\underline{y}$ and $\bar{y}$ are the lower and upper limits of the output, and $\underline{u}$ and $\bar{u}$ are the lower and upper limits of the control input u.

4. Apply only the first element $u(k)$ of $\kappa(k)$ and repeat the procedure at the next sampling time.

This approach, known as receding horizon optimization, may cause an increase in computation time, especially for nonlinear optimization problems with constraints. Therefore, it is essential to reduce computation time per iteration by using a simple model and fast optimization techniques.

2.3.2 Constrained neural network model predictive control

Neural networks have demonstrated their effectiveness in modeling nonlinear dynamic processes, providing a powerful tool for system representation and control [20, 53]. Specifically, feed-forward neural networks with just a single hidden layer have been shown to approximate any continuous function with arbitrary precision, given an adequate number of neurons [154].

In the context of model predictive control, neural networks have found significant application [155, 156]. The ability to accurately model complex systems allows for more precise control and prediction and also reduces the computation time associated with complex derivative-based models.

Consider the vector:

$$\chi(k) = [u(k), \ldots, u\,(k - n_u)\,, y(k), \ldots, y\,(k - n_y)]^T \tag{2.8}$$

where n_u and n_y denote the number of past control inputs and outputs.

The prediction model of a given system can be written as:

$$\hat{y}(k+1) = f_{NN}(\chi(k)) \tag{2.9}$$

where f_{NN} is the neural network mathematical function.

A simple and effective solution is to use a feed-forward neural network with a

single hidden layer that can be represented by:

$$f_{NN}(\chi(k)) = \sigma_2\left(W_2\sigma_1\left(W_1\chi(k) + b_1\right) + b_2\right) \tag{2.10}$$

where σ_1 and σ_2 represent the hidden and the output layers' activation functions, W_1 and W_2 denote the weight matrices, and b_1 and b_2 refer to the bias vectors.

Having established the prediction model of MPC, it's important to discuss how MPC handles the various constraints of the system. Depending on their nature, constraints can be mainly grouped into three categories: input constraints caused by the actuators' capabilities, state constraints due to the sensors' limitations or safety reasons, and output constraints to keep the system in the acceptable operating range. Several methods were proposed to handle the constraints [157–159]. One particular technique is to soften the constraints using slack variables [160]; it is used to treat output constraints by adding a penalty to the optimization problem in Equation (2.7) as follows [161]:

$$\begin{aligned}\kappa(k) &= \arg\min_{\kappa} J \\ &= \arg\min_{\kappa} \sum_{i=N_1}^{N_2} \left[(e(k+i \mid k))^T \Gamma_y (e(k+i \mid k))\right] \\ &\quad + \sum_{i=1}^{N_u} \left[(\Delta u(k+i-1))^T R (\Delta u(k+i-1))\right]\end{aligned} \tag{2.11}$$

Subject to

$$\begin{aligned}\underline{u} \le u(k+i) \le \bar{u}, \quad \forall\, i = 0, 1, \ldots, N_u - 1 \\ |\Delta u(k+i)| \le \Delta u_{max}, \quad \forall\, i = 0, 1, \ldots, N_u - 1 \\ \Delta u(k+i) = 0, \quad \forall\, i > N_u - 1\end{aligned}$$

The output-dependent weight Γ_y is given as:

$$\Gamma_y(y) = \begin{bmatrix} \Gamma_{y_1}(y_1) & 0 & \cdots & 0 \\ 0 & \Gamma_{y_2}(y_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \Gamma_{y_n}(y_n) \end{bmatrix} \tag{2.12}$$

And:

$$\Gamma_{\hat{y}_i}(\hat{y}_i) = \begin{cases} \Gamma_{\hat{y}_i}(0)\left[1 + \epsilon_i\left(\hat{y}_i - \underline{y}\right)^2\right] & \text{if } \hat{y}_i < \underline{y} \\ \Gamma_{\hat{y}_i}(0) & \text{if } \underline{y} \le \hat{y}_i \le \bar{y} \\ \Gamma_{\hat{y}_i}(0)\left[1 + \epsilon_i\left(\hat{y}_i - \bar{y}\right)^2\right] & \text{if } \hat{y}_i < \bar{y} \end{cases} \tag{2.13}$$

where $i = 1, \ldots, n$, n is the number of outputs in the system, ϵ_i is a constant used to

change the value of penalization. $\Gamma_{\hat{y}_i}(0)$ is a constant minimum value of the weight function at which the output is not penalized.

Input constraints can be handled by limiting the search space of the optimization problem to the given lower and upper limits $\underline{u} \leq u(k) \leq \bar{u}$.

2.3.3 Archimedes optimization algorithm

The Archimedes Optimization Algorithm (AOA) is a meta-heuristic optimization method, developed by Hashim et al. [162], that draws its inspiration from the principle of buoyancy, a concept attributed to Archimedes. This principle states that an object, whether partially or entirely immersed in a fluid, experiences an upward force equal to the weight of the fluid displaced by the object.

In the context of AOA, objects of varied volumes and densities submerged in a fluid are represented as the population. These objects, when immersed in the same fluid, will accelerate differently in order to reach equilibrium. This phenomenon forms the basis of the population-based AOA.

The position of each object of the AOA population is randomly initialized, and the corresponding objective function J is evaluated. Subsequently, the densities, accelerations, and volumes of these objects are updated in each iteration to determine new positions until the criteria for termination are satisfied. The steps of AOA are described as follows:

a) Initialization

Prior to running the algorithm, the afterward parameters are initialized:

- The population size P_s (number of objects).
- The dimension of the optimization D (number of variables associated with every object).
- The search space limits $\bar{u}$ and $\underline{u}$.
- The maximum number of iterations $s_{\max}$.
- The AOA constants C_1, C_2, C_3, and C_4.

The initial objects' attributes positions p, densities ρ, volumes υ, and accelerations α for each object $j = 1, 2, \ldots, P_s$ are randomly set using the equations:

$$\begin{cases} p_j = \underline{u} + rand \times (\bar{u} - \underline{u}) \\ \rho_j = rand \\ \upsilon_j = rand \\ \alpha_j = \underline{u} + rand \times (\bar{u} - \underline{u}) \end{cases} \tag{2.14}$$

where $rand$ is a random vector between 0 and 1 of dimension D.

The objective function is evaluated, and the object having the best cost value is chosen with the associated best position p_{best}, density ρ_{best}, volume υ_{best}, and acceleration α_{best}.

b) Update densities and volumes

Each object's density and volume are adjusted at every iteration s with:

$$\begin{cases} \rho_j^{s+1} = \rho_j^s + rand \times \left(\rho_{best} - \rho_j^s\right) \\ \upsilon_j^{s+1} = \upsilon_j^s + rand \times \left(\upsilon_{best} - \upsilon_j^s\right) \end{cases} \tag{2.15}$$

where ρ_j^s and υ_j^s are the density and the volume of the j th object at iteration s.

c) Transfer operator and density factor

The transfer operator TF progressively increases using the following equation to shift the search from a phase of exploration to exploitation.

$$TF = \exp\left(\frac{s - s_{\max}}{s_{\max}}\right) \tag{2.16}$$

The density factor switches the search from global to local; it decreases gradually as follows:

$$d^{s+1} = \exp\left(\frac{s_{\max} - s}{s_{\max}}\right) - \left(\frac{s}{s_{\max}}\right) \tag{2.17}$$

i. Exploration phase (collision between objects)

In this phase, $TF \leq 0.5$ and the objects are in collision. The acceleration is modified utilizing a random object mr:

$$\alpha_j^{s+1} = \frac{\rho_{mr} + \upsilon_{mr} \times \alpha_{mr}}{\rho_j^{s+1} \times \upsilon_j^{s+1}} \tag{2.18}$$

ii. Exploitation phase (no collision between objects)

No collision takes place between objects when $TF > 0.5$. The object's acceleration can be expressed as:

$$\alpha_j^{s+1} = \frac{\rho_{best} + \upsilon_{best} \times \alpha_{best}}{\rho_j^{s+1} \times \upsilon_j^{s+1}} \tag{2.19}$$

The exploration-exploitation ratio could be changed by choosing a different value than 0.5.

iii. Normalize acceleration

It is essential to normalize the acceleration to determine the step percentage that each object will change; this allows the solutions to approach the global best and also avoid trapping in the local minima. The acceleration is normalized between ub and lb as:

$$\alpha_{j-norm}^{s+1} = ub \times \frac{\alpha_j^{s+1} - \min(\alpha)}{\max(\alpha) - \min(\alpha)} + lb \tag{2.20}$$

d) Update the position

For $TF \leq 0.5$ (phase of exploitation), the position of the object is updated as follows:

$$p_j^{s+1} = p_j^s + C_1 \times rand \times \alpha_{j-norm}^{s+1} \times d^{s+1} \times \left(p_{rand} - p_j^s\right) \tag{2.21}$$

where p_{rand} is a chosen random position.

For $TF > 0.5$ (phase of exploitation), the position of the object is adjusted using:

$$\begin{aligned} p_j^{s+1} = p_{best}^s + F \times C_2 \times rand \times \alpha_{j-norm}^{s+1} \\ \times d^{s+1} \times \left(C_3 \times TF \times p_{best} - p_j^s\right) \end{aligned} \tag{2.22}$$

where F is a flag that allows to change the direction of motion of the object:

$$F = \begin{cases} +1 & \text{if } 2 \times r - C_4 < 0.5 \\ -1 & \text{if } 2 \times r - C_4 > 0.5 \end{cases} \tag{2.23}$$

e) Evaluation

Finally, the objective function of each object is evaluated, and the optimal solution is modified (i.e., p_{best}, ρ_{best}, υ_{best}, and α_{best}).

The flowchart of the AOA is given by Figure 2.1.

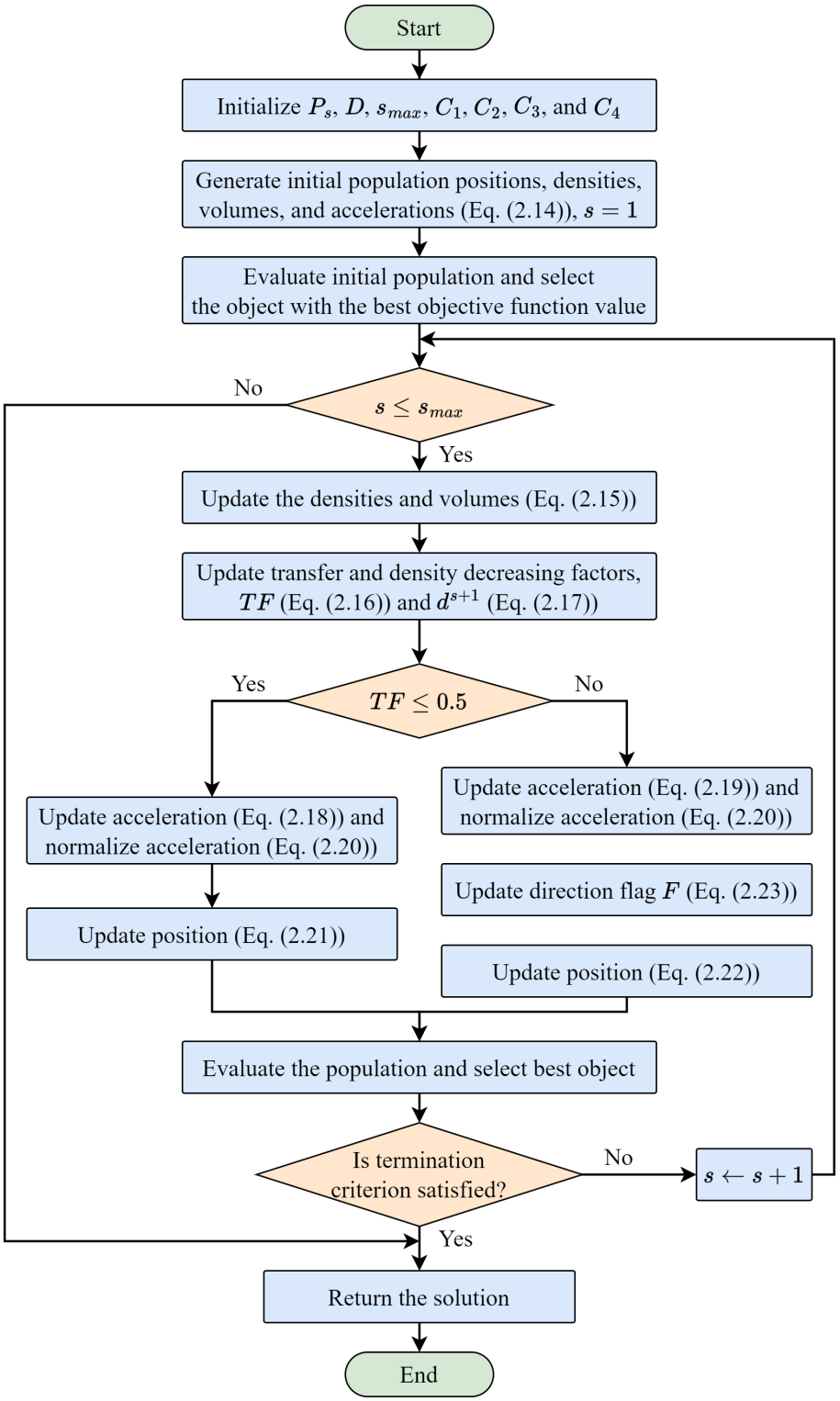


Figure 2.1: Flowchart of AOA

2.3.4 NNMPC-AOA control algorithm

Based on the general formulation of the MPC and the AOA, the proposed neural network model predictive control based on Archimedes optimization algorithm can be described by the following steps of Algorithm 1.

Algorithm 1 NNMPC-AOA algorithm

1: Define the MPC parameters N_1, N_2, N_u, and ϵ_i.
2: Define the AOA parameters P_s, D, $s_{\max}$, $\underline{u}$, $\bar{u}$, C_1, C_2, C_3, C_4, lb, and ub.
3: At each sampling time k, let the initial step of the optimization be $s = 1$.
4: **for** $j = 1$ to P_s **do**
5: Generate initial solution positions, densities, volumes, and accelerations using Equation (2.14).
6: **end for**
7: Select reference trajectories between $k + N_1$ and $k + N_2$.
8: **for** $j = 1$ to P_s **do**
9: Calculate the system's predicted outputs using the neural network.
10: Evaluate the objective function J.
11: **end for**
12: Select the best object (p_{best}, ρ_{best} , υ_{best}, and α_{best}) with the best objective function J_{best}.
13: Calculate TF and d^{s+1} using Equations (2.16) and (2.17).
14: **for** $j = 1$ to P_s **do**
15: Use Equation (2.15) to update densities and volumes.
16: **if** $TF \leq 0.5$ **then**
17: Update accelerations using Equation (2.18).
18: **else**
19: Update accelerations using Equation (2.19).
20: **end if**
21: **end for**
22: Normalize the acceleration using Equation (2.20).
23: **for** $j = 1$ to P_s **do**
24: Use Equation (2.15) to update densities and volumes.
25: **if** $TF \leq 0.5$ **then**
26: Find the new positions using Equation (2.21).
27: **else**
28: Update direction flag F by Equation (2.23).
29: Find the new positions using Equation (2.22).
30: **end if**
31: **end for**
32: **for** $j = 1$ to P_s **do**
33: Calculate the system's predicted outputs using the neural network.
34: Evaluate the objective function J.
35: **end for**
36: Select the best object (p_{best}, ρ_{best}, υ_{best}, and α_{best}) with the best objective function J_{best}.
37: **if** $s \geq s_{\max}$ **then**
38: Go to 40.
39: **end if**
40: Set $s \leftarrow s + 1$ and repeat 12.
41: Apply the obtained control action (the first element of p_{best}).
42: Wait for the next sampling time $k \leftarrow k + 1$ then go to 3.

2.4 Control of Two DoF Robot Manipulator

2.4.1 Two DoF robot description

The planar robot shown in Figure 2.2 has the following dynamic model based on Equation (2.5) [163].

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \tag{2.24}$$

where:

$$\begin{aligned} M_{11} =& I_1 + I_2 + m_1 r_1^2 + (m_2 + m_L)\, l_1^2 + m_2 r_2^2 \\ &+ (2 m_2 l_1 r_2 + 2 m_L l_1 l_2) \cos(q_2) + m_L l_2^2 \\ M_{12} =& I_2 + m_2 r_2^2 + (m_2 l_1 r_2 + m_L l_1 l_2) \cos(q_2) + m_L l_2^2 \\ M_{21} =& M_{12} \\ M_{22} =& I_2 + m_2 r_2^2 + m_L l_2^2 \\ V_1 =& -(m_2 l_1 r_2 + m_L l_1 l_2)\,(2\dot{q}_1 + \dot{q}_2)\,\dot{q}_2 \sin(q_2) \\ V_2 =& (m_2 l_1 r_2 + m_L l_1 l_2)\,\dot{q}_1^2 \sin(q_2) \\ F_1 =& \mu_1 \dot{q}_1 \\ F_2 =& \mu_2 \dot{q}_2 \\ G_1 =& (m_1 r_1 + m_2 l_1 + m_L l_1)\, g \cos(q_1) \\ &+ (m_2 r_2 + m_L l_2)\, g \cos(q_1 + q_2) \\ G_2 =& (m_2 + m_L)\, g\, r_2 \cos(q_1 + q_2) \end{aligned}$$

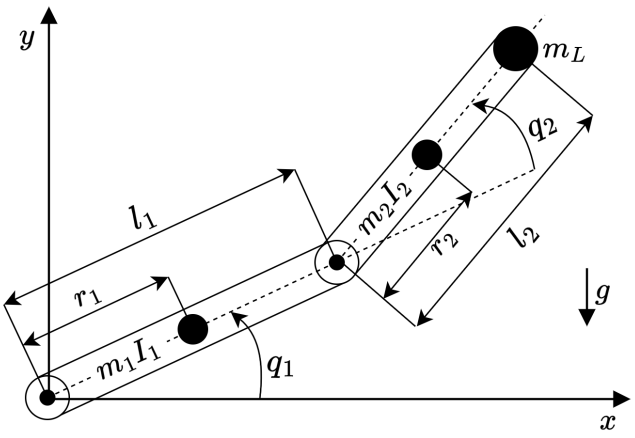


Figure 2.2: Two degrees of freedom robot manipulator

The input variables are the torques at each joint τ_1 and τ_2, and the outputs are the angular positions q_1 and q_2. Table 2.1 gives the system's physical parameters [163].

Table 2.1: Parameters values of the robot manipulator

Parameter	Value	Parameter	Value
m_1	0.3929243 kg	m_L	0.2 kg
m_2	0.0944039 kg	I_1	0.00114111 m^2 kg
l_1	0.2032 m	I_2	0.00202470 m^2 kg
l_2	0.1524 m	μ_1	0.141231 N m
r_1	0.104648 m	μ_2	0.353078 N m
r_2	0.081788 m	g	9.81 m/s^2

2.4.2 Neural network identification of two DoF robot

Since the robot manipulator has two outputs, two separate neural networks are used, one to predict each output with a sampling time $T_s = 10$ ms. This approach yielded better results than using only one network with two outputs.

Both neural networks have a similar structure: an input layer containing 8 neurons, one hidden layer containing 20 neurons with a sigmoid activation function, and an output layer with one neuron and having a linear activation function. The input vector of each neural network is given by:

$$\chi(k) = [\tau_1(k), \tau_1(k-1), \tau_2(k), \tau_2(k-1), q_1(k), q_1(k-1), q_2(k), q_2(k-1)]^T \quad (2.25)$$

The output of each neural network is the predicted angular position at each joint $\hat{q}_1(k+1)$ and $\hat{q}_2(k+1)$, respectively. Training and test datasets are generated by solving Equation (2.24) for random inputs within the operating range of the system. Levenberg-Marquardt algorithm is used to train each neural network.

Figures 2.3 and 2.4 give the test results for each neural network, and they show good accuracy of the obtained models.

The performances of the trained networks are evaluated based on the coefficient of determination R^2 and the Root Mean Squared Error (RMSE) of the one-step-ahead predictions. Table 2.2 gives the RMSE and R^2 values of the h-step-ahead predictions of the obtained models. The predictions are computed by considering the model to be in a closed loop, where the predicted output at step $h > 2$ is computed using the output of step $h-1$ as input.

2.4.3 Controller implementation

In this section, the control performances of the proposed NNMPC-AOA are evaluated using a robot manipulator with two degrees of freedom. The proposed controller is compared to the Neural Network Model Predictive Control based on TLBO

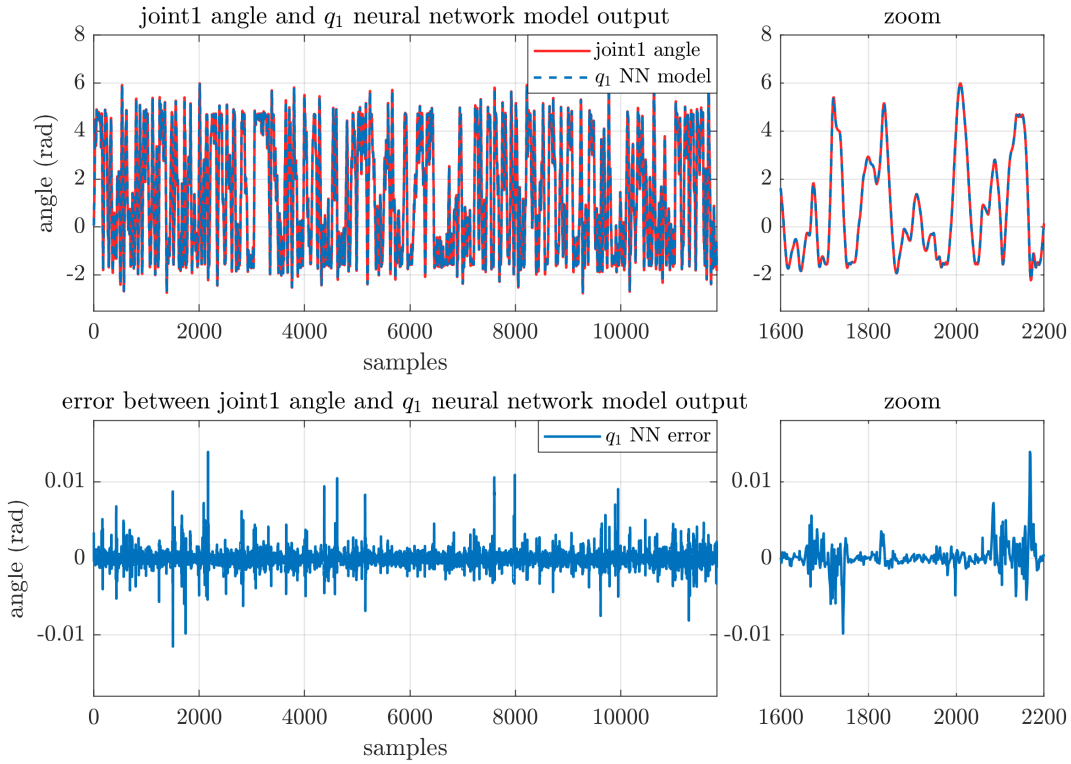


Figure 2.3: Response of the neural network model of the first angle for the test data

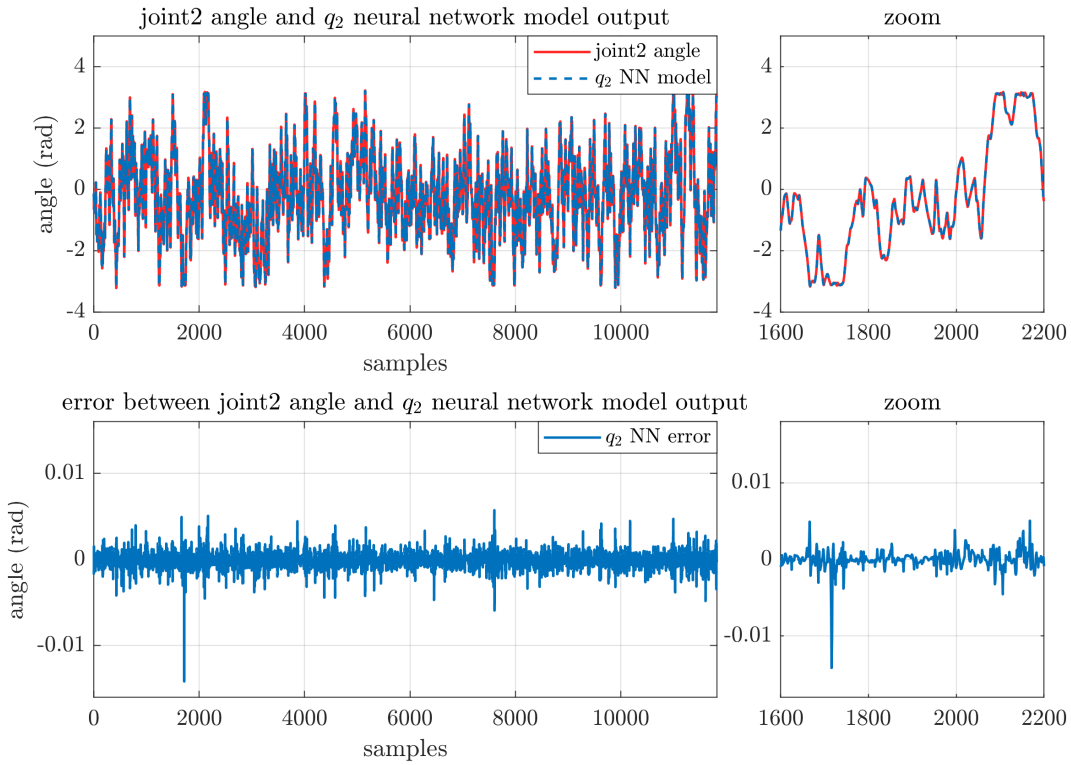


Figure 2.4: Response of the neural network model of the second angle for the test data

Table 2.2: RMSE and R^2 values of the multi-step-ahead predictions of q_1 and q_2 using the obtained models

h	q_1 neural network		q_2 neural network	
	RMSE	R^2	RMSE	R^2
1	0.0009728	0.9999998	0.0007442	0.9999997
2	0.0025035	0.9999989	0.0016849	0.9999984
3	0.0043639	0.9999966	0.0028137	0.9999955
4	0.0063865	0.9999926	0.0041306	0.9999904
6	0.0106713	0.9999794	0.0071915	0.9999708
8	0.0151694	0.9999584	0.0106399	0.9999361
10	0.0198856	0.9999286	0.0144571	0.9998821
15	0.0324679	0.9998098	0.0239938	0.9996751

(NNMPC-TLBO), the Neural Network Model Predictive Control based on PSO (NNMPC-PSO), the PID, and the computed torque controller. Each simulation is run in MATLAB using an Intel Core (TM) i7 3.60 GHz machine.

The NNMPC-TLBO and the NNMPC-PSO are designed in a similar way to the proposed controller, with the only difference being the use of a different optimization algorithm, namely the TLBO and PSO algorithms. For all controllers, $\tau_1 \in [-15, 15]$, $\tau_2 \in [-7, 7]$, and the sampling time is $T_s = 0.01$ second.

The predictive controllers have the control block diagram shown in Figure 2.5 with the following parameters: $N_1 = 1$, $N_2 = 4$, $N_u = 1$, $R = 4 \times 10^{-4}$, $s_{\max} = 10$, $P_s = 8$, and $D = N_u$.

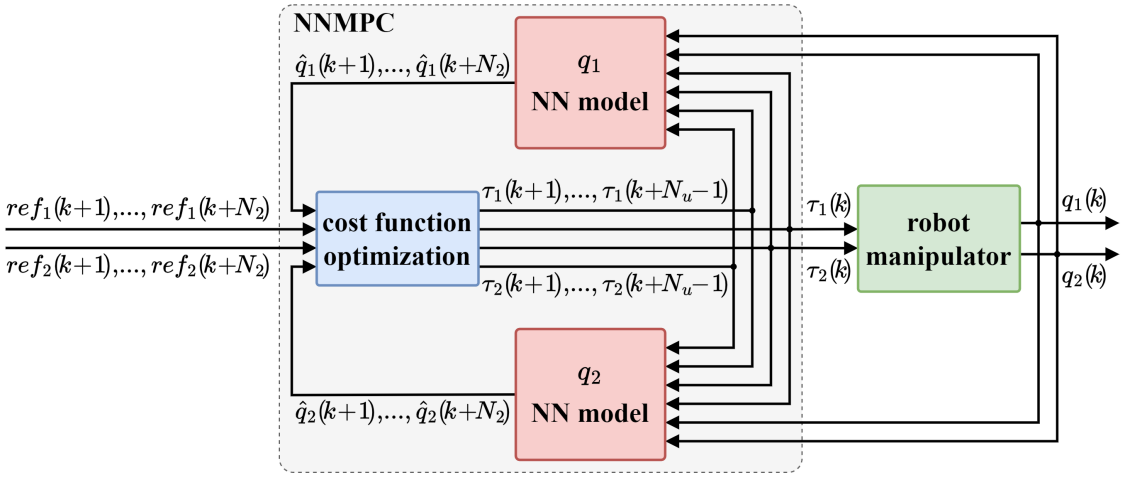


Figure 2.5: Robot manipulator control block diagram

The parameters of the PSO are given in Table 2.3. The values of AOA parameters that are provided in Table 2.4 are those suggested by [162] for engineering problems.

Table 2.3: PSO parameters

Parameter	Value	Parameter	Value
c_1	2	w	1
c_2	2	w_d	0.99

Table 2.4: AOA parameters

Parameter	Value	Parameter	Value
C_1	2	C_4	0.5
C_2	6	lb	0.1
C_3	2	ub	0.9

The PID parameters are optimized using the PSO algorithm, where the RMSE is used as a cost function, and the parameters of the controller are the attributes of each particle. The optimized PID parameters are given in Table 2.5.

Table 2.5: PID controller parameters

Parameter	Value	Parameter	Value
k_{p1}	40	k_{p2}	50
k_{i1}	12	k_{i2}	20
k_{d1}	0.03	k_{d2}	0.01

The CTC uses the control law given by the following equation:

$$\begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \left(\begin{bmatrix} \ddot{ref}_1 \\ \ddot{ref}_2 \end{bmatrix} + K_v \begin{bmatrix} \dot{ref}_1 - \dot{q}_1 \\ \dot{ref}_2 - \dot{q}_2 \end{bmatrix} + K_p \begin{bmatrix} ref_1 - q_1 \\ ref_2 - q_2 \end{bmatrix} \right) + \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \tag{2.26}$$

where ref_1 and ref_2 are the reference trajectories, $K_v = 59.7$, and $K_p = 651.4$ are obtained using the PSO algorithm.

With no output constraints, Figures 2.6 and 2.7 respectively show the results for the multi-step and the sinusoidal trajectories.

The average Mean Absolute Error (MAE), Mean Squared Error (MSE), and RMSE over a thousand runs in both cases are listed in Table 2.6. The average computing time is presented in Table 2.7.

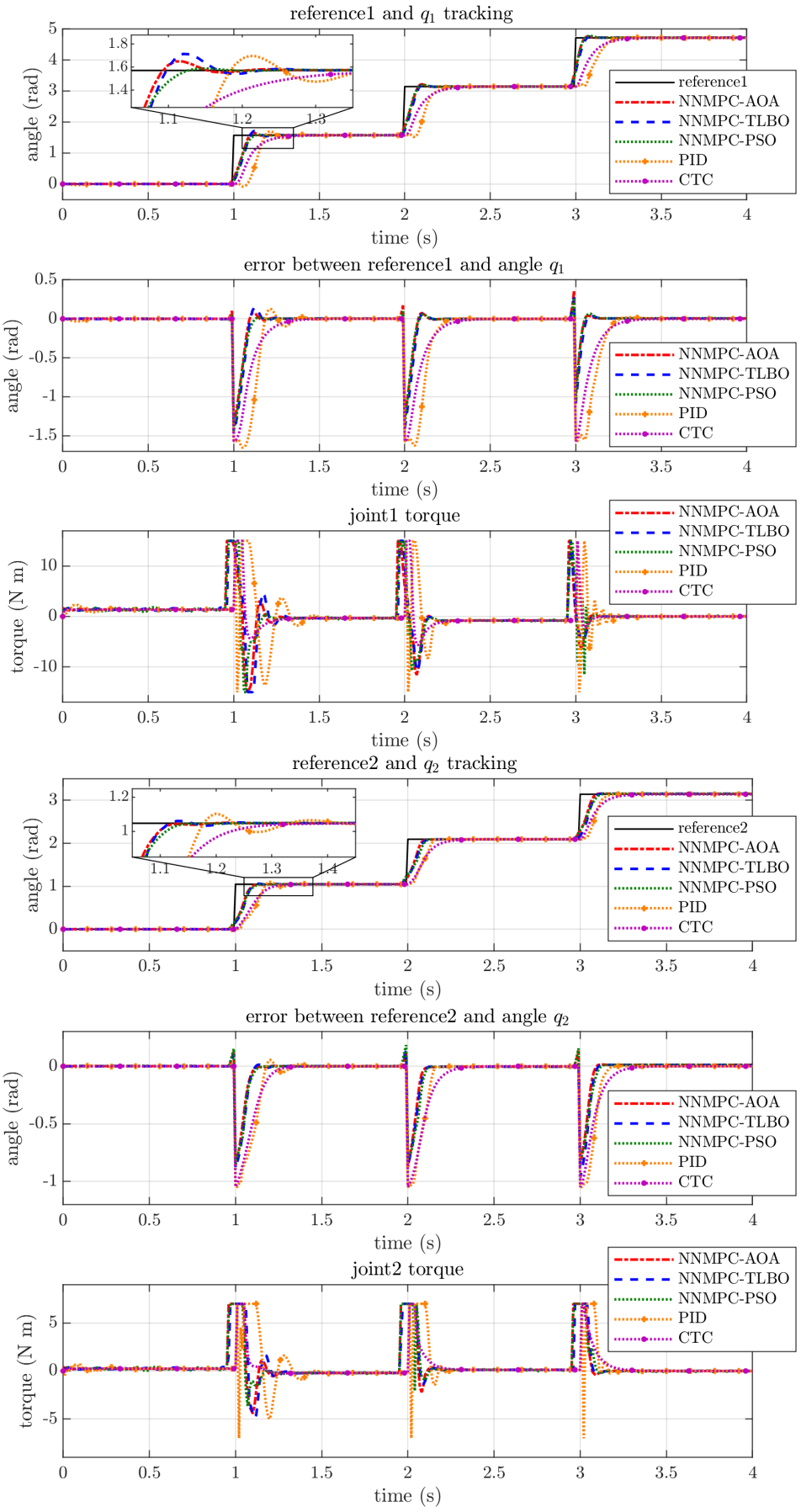


Figure 2.6: Control performance of the robot with multi-step trajectories, without constraints

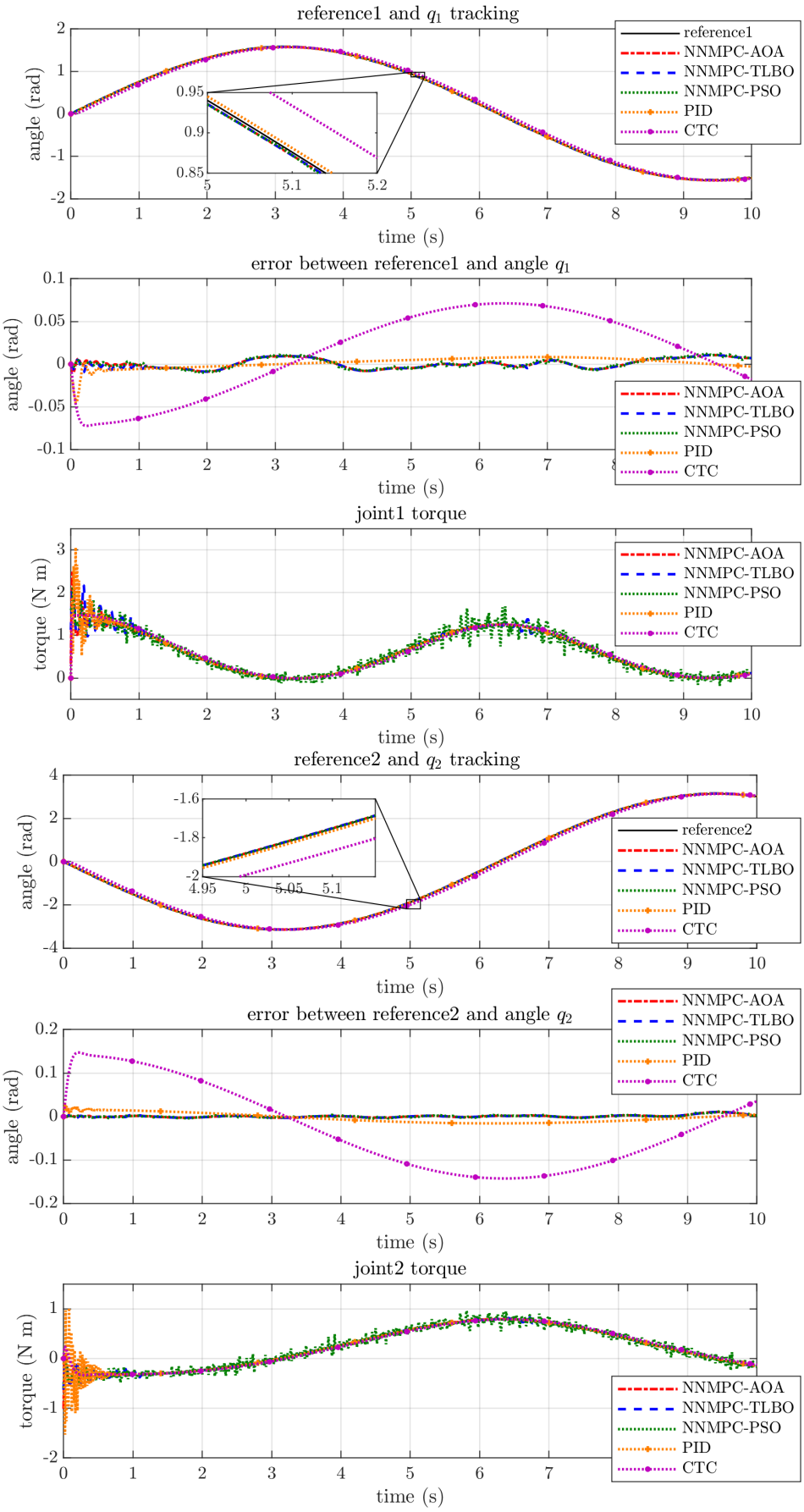


Figure 2.7: Control performance of the robot with sinusoidal trajectories, without constraints

Table 2.6: MAE, MSE, and RMSE values without constraints

	Controller	MAE	MSE	RMSE
Multi-step trajectory	NNMPC-AOA	**8.505e-2**	**5.647e-2**	**3.330e-1**
	NNMPC-TLBO	9.640e-2	7.163e-2	3.748e-1
	NNMPC-PSO	1.004e-1	6.461e-2	3.554e-1
	PID	2.415e-1	2.698e-1	7.066e-1
	CTC	2.418e-1	1.749e-1	5.808e-1
Sinusoidal trajectory	NNMPC-AOA	**6.680e-3**	**4.037e-5**	**8.588e-3**
	NNMPC-TLBO	6.781e-3	4.041e-5	8.591e-3
	NNMPC-PSO	9.496e-3	4.237e-5	8.837e-3
	PID	1.436e-2	1.597e-4	1.731e-2
	CTC	8.638e-2	1.196e-2	1.467e-1

Table 2.7: NNMPC-AOA, NNMPC-TLBO, and NNMPC-PSO computing time without constraints

Controller	Computing time (ms)	
	Multi-step trajectory	Sinusoidal trajectory
NNMPC-AOA	2.068	2.073
NNMPC-TLBO	4.038	4.043
NNMPC-PSO	**1.925**	**1.926**

It is clear from Figure 2.6 that the NNMPC-AOA, NNMPC-TLBO, and NNMPC-PSO have faster settling times than the PID and CTC techniques since the predictive controllers anticipate the changes in reference and proactively change the output.

Tables 2.6 and 2.7 show that the proposed NNMPC-AOA has a better tracking accuracy than the other controllers and maintains a reasonable computing time, compared to the NNMPC-PSO that is the fastest in execution, and it remains within the limit of the sampling time (10 ms).

The NNMPC-AOA is remarkably faster than the NNMPC-TLBO and slightly slower than the NNMPC-PSO, which was expected since AOA uses more equations to update the solution compared to only two equations in the case of PSO.

In another simulation, the overshoot is limited to 1% in the case of multi-step trajectory for the predictive controllers. Equation (2.11) is used with $\epsilon_1 = \epsilon_2 = 100$. Figure 2.8 shows the simulation results, and Table 2.8 gives the obtained average values of MAE, MSE, RMSE, and computing time over a thousand runs.

It can be noticed that the NNMPC-PSO exhibits a noticeable degradation in performance. At the same time, the NNMPC-AOA outperforms the NNMPC-TLBO

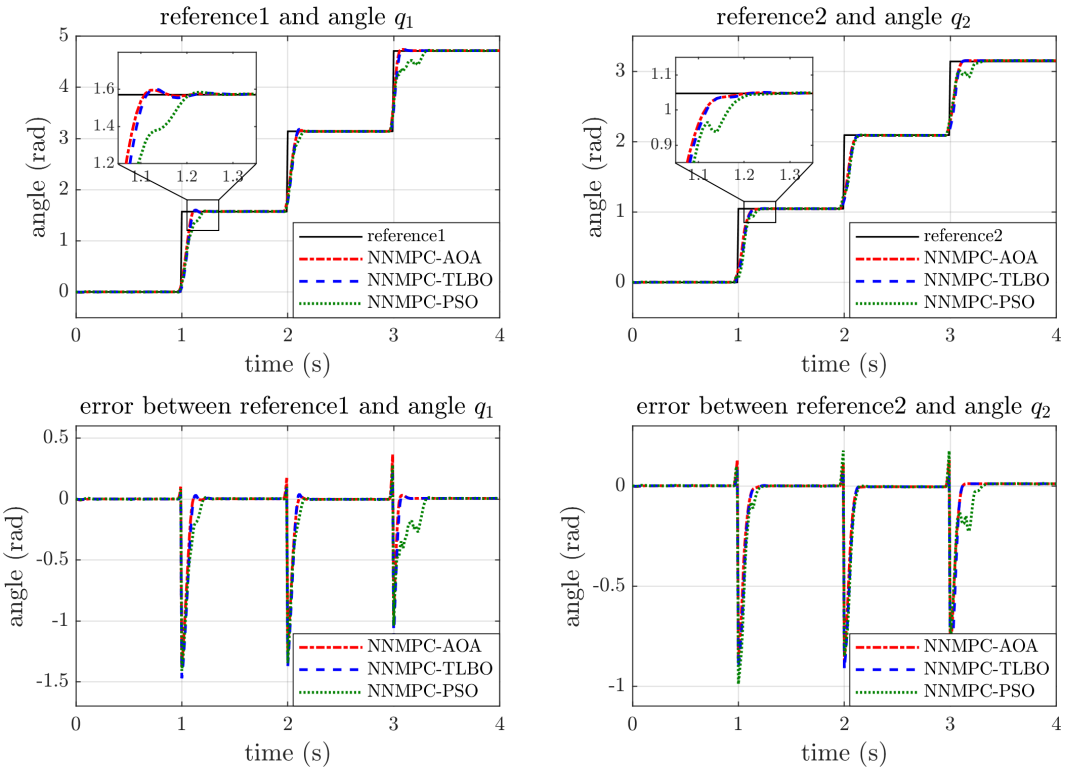


Figure 2.8: Control performance of the robot with constraints

Table 2.8: MAE, MSE, RMSE values, and computing time with constraints

Controller	NNMPC-AOA	NNMPC-TLBO	NNMPC-PSO
MAE	**8.902e-2**	9.629e-2	1.399e-1
MSE	**5.781e-2**	7.179e-2	7.387e-2
RMSE	**3.372e-1**	3.754e-1	3.787e-1
Computing time (ms)	2.078	4.070	**1.974**

in terms of tracking accuracy with a shorter computing time that is still within the limit of the sampling time.

2.4.4 Performance analysis of the controller

To evaluate the effectiveness of the proposed controller, various scenarios are simulated to analyze its ability to reject external disturbances and its sensitivity to measurement noise.

In the first scenario, various loads, which are different from those of the nominal state, are considered to be a disturbance to the system. The respective metrics across thousand iterations in the case of sinusoidal and multi-step trajectories are given in Table 2.9.

It could be noticed that the values of errors remain within acceptable limits, as

Table 2.9: MAE, MSE, and RMSE values for different loads

	Weight (kg)	MAE	MSE	RMSE
Multi-step trajectory	m_L= 0.000	8.596e-2	3.064e-2	2.392e-1
	m_L= 0.150	7.731e-2	4.321e-2	2.896e-1
	m_L= 0.300	9.382e-2	6.267e-2	3.511e-1
	m_L= 0.430	1.215e-1	8.299e-2	4.051e-1
	m_L= 0.567	1.461e-1	9.993e-2	4.455e-1
Sinusoidal trajectory	m_L= 0.000	1.316e-2	1.434e-4	1.514e-2
	m_L= 0.150	7.814e-3	5.260e-5	9.624e-3
	m_L= 0.300	7.528e-3	4.802e-5	9.279e-3
	m_L= 0.430	1.315e-2	1.369e-4	1.490e-2
	m_L= 0.567	1.836e-2	2.820e-4	2.092e-2

they do not significantly exceed the values presented in Tables 2.6 and 2.8. The controller is capable of rejecting disturbances caused by weight-lifting variations within the considered range in the simulation.

Figures 2.9 and 2.10 shows the system's response when the end effector lifts a mass of $m_L = 0.567$ kg, where the perturbation is more significant.

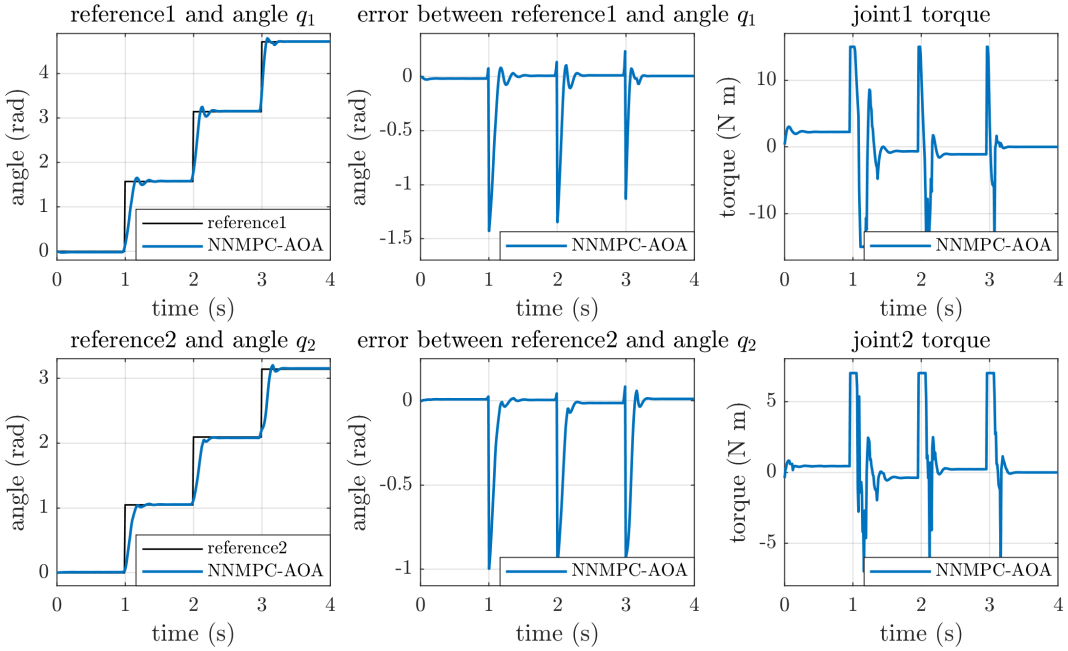


Figure 2.9: Control performance of the robot with a load $m_L = 0.567\ kg$ in the case of multi-step trajectories

In the second simulation, various additive input and output perturbations are considered. The amplitude of the input signal is limited by adding -20% and -50% of the maximum value of the control signal within the intervals of [1.5 2.5] seconds

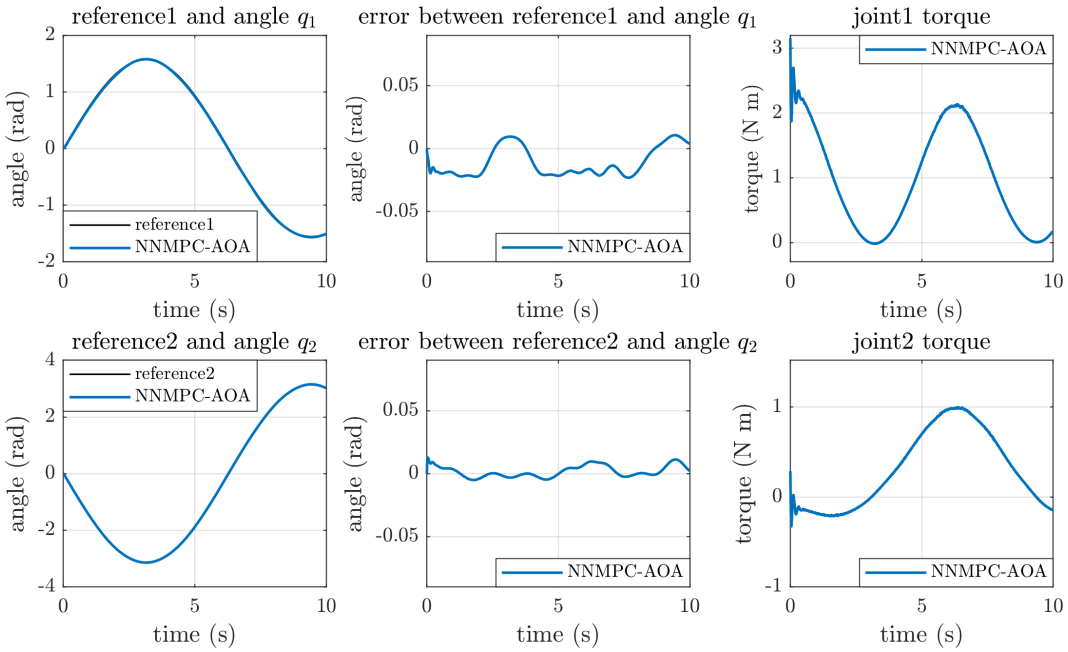


Figure 2.10: Control performance of the robot with a load $m_L = 0.567$ kg in the case of sinusoidal trajectories

for the multi-step trajectories and [3.5 6.5] seconds for the sinusoidal trajectories.

Additionally, the output signal is disturbed by adding an increment of 50% between [1.5 2.5] seconds for the multi-step trajectories and [3.5 6.5] seconds for the sinusoidal trajectories.

In the last simulation, a measurement noise is added to the outputs throughout the entire duration of the simulation. Specifically, a white noise with an amplitude of 5 degrees (equivalent to 0.0873 rad), a mean value of 8.93e-4 rad, and a variance $\sigma^2 = 0.0026$ rad^2.

Table 2.10 gives the values of the different metrics over 1000 runs for both trajectories in each scenario.

Table 2.10: MAE, MSE, and RMSE values for different perturbations

	simulation	MAE	MSE	RMSE
Multi-step trajectory	-20% input disturbance	9.598e-2	6.318e-2	3.529e-1
	-50% input disturbance	1.165e-1	7.776e-2	3.925e-1
	+50% output disturbance	4.244e-1	9.097e-2	4.244e-1
	Measurement noise	1.704e-1	7.247e-2	3.747e-1
Sinusoidal trajectory	-20% input disturbance	9.126e-3	6.238e-5	1.096e-2
	-50% input disturbance	1.796e-2	3.350e-4	2.587e-2
	+50% output disturbance	2.648e-2	1.151e-2	1.433e-1
	Measurement noise	6.184e-2	2.888e-3	7.583e-2

Figures 2.11 and 2.12 shows the system's response with measurement noise.

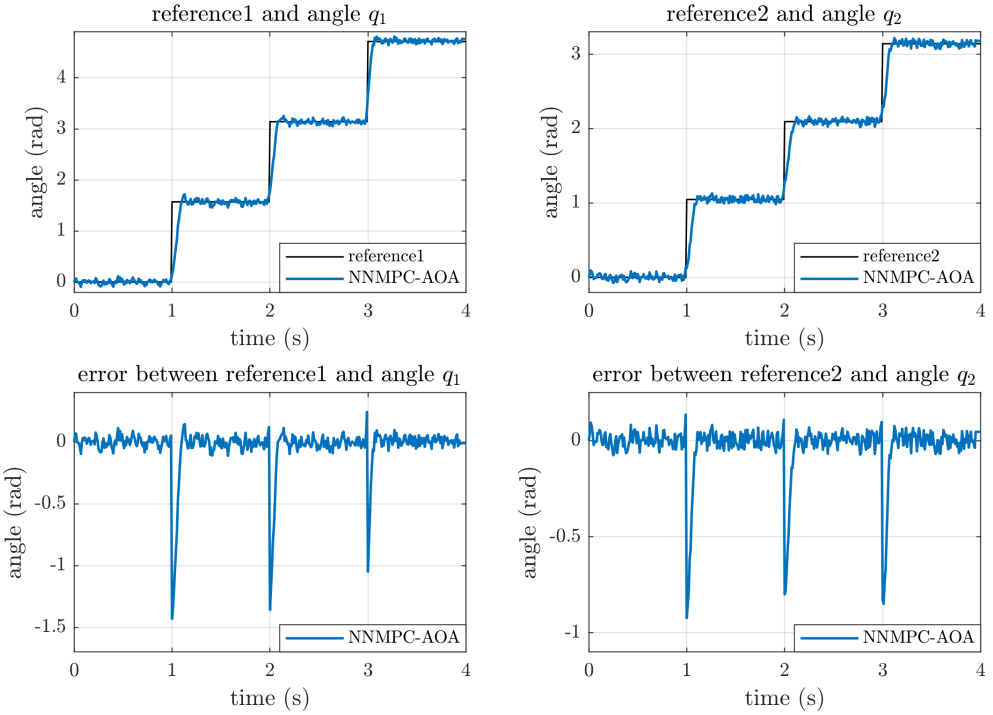


Figure 2.11: Control performance of the robot with measurement noise in the case of multi-step trajectories

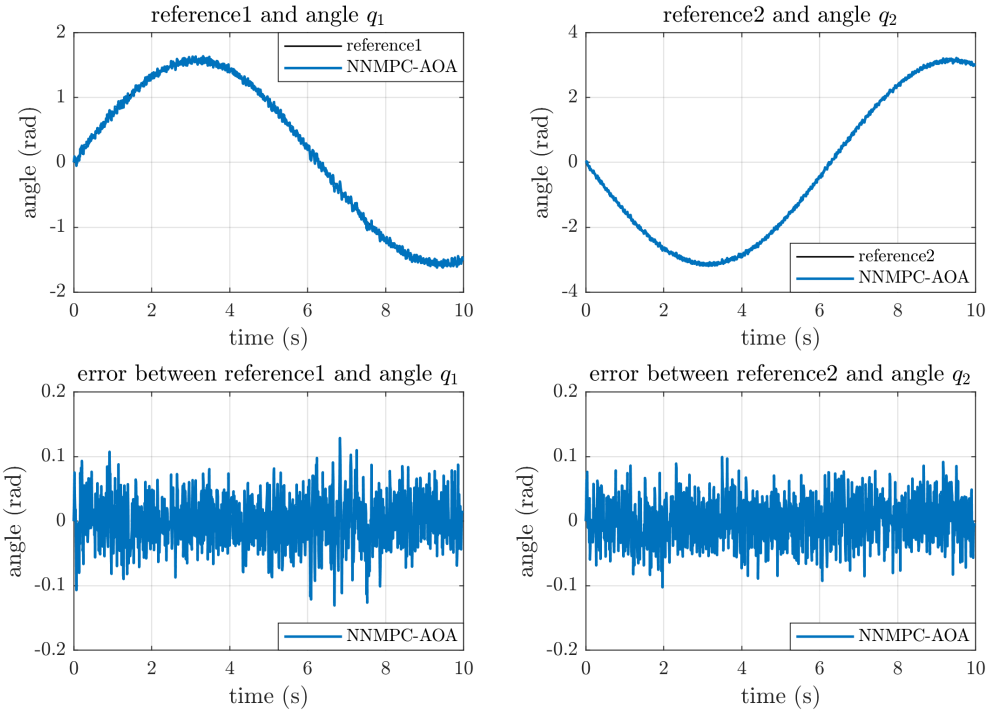


Figure 2.12: Control performance of the robot with measurement noise in the case of sinusoidal trajectories

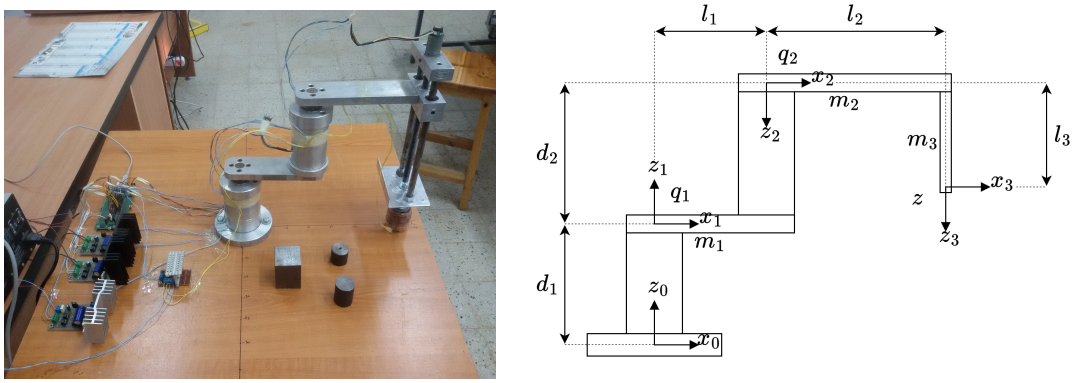
Despite the different types of perturbations and noise introduced to the system, the proposed controller demonstrated acceptable robustness against external

disturbances and measurement noise.

2.5 Experimental Study on Three DoF Robot

2.5.1 Three DoF robot description

The three degrees of freedom arm manipulator shown in the experimental setup in Figure 2.13 is used to demonstrate the control performances of the proposed NNMPC-AOA controller. To control the position of the electromagnet end effector, two 24V 12RPM DC motors provide the rotational movements in joints 1 and 2, and a 12V 170RPM DC motor controls the movement along the vertical z-axis. Each motor is driven using an H-bridge DC motor driver [164]. The arm manipulator has the structure shown in the diagram in Figure 2.13, and the numerical values of its parameters are given in Table 2.11.



(a) The experimental setup **(b)** Robotic arm diagram

Figure 2.13: Three degrees of freedom arm manipulator

Table 2.11: Arm manipulator parameter values

Parameter	Value	Parameter	Value
m_1	2.30 kg	l_2	0.1965 m
m_2	0.60 kg	l_3	0.3400 m
m_3	2.36 kg	d_1	0.1750 m
l_1	0.1380 m	d_2	0.1650 m

The control algorithm is implemented using the TMS320F28335 DSP board, where q_1, q_2, and z are measured, and the PWM signals necessary for the motors' drivers are generated. The control block diagram is illustrated in Figure 2.14.

Three separate neural networks are used to model the behavior of the arm manipulator, one for each output q_1, q_2, and z. The motors are driven using the

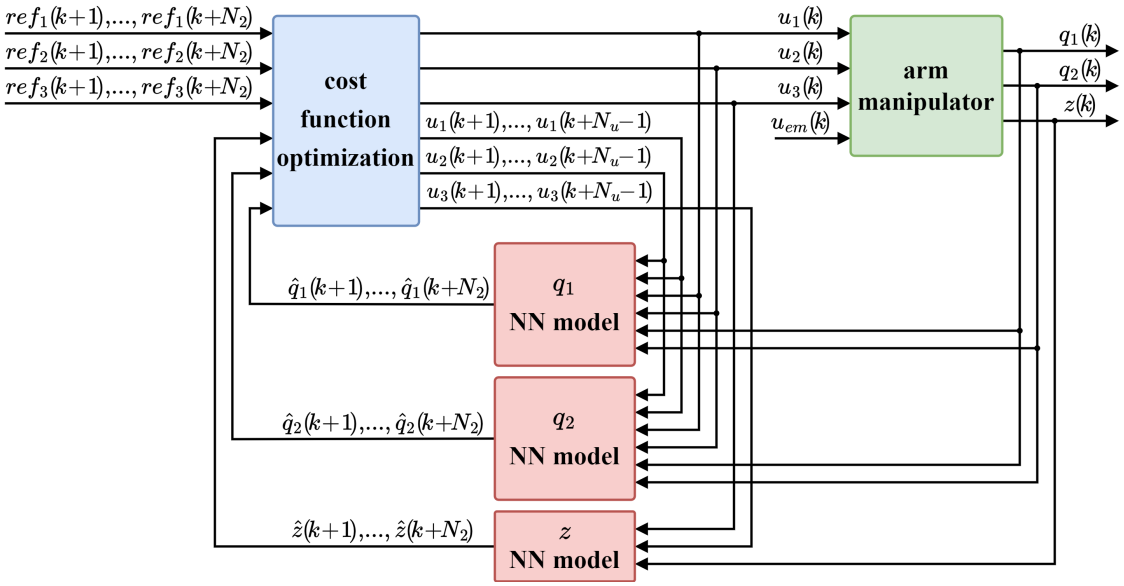


Figure 2.14: Control block diagram of the arm manipulator

inputs u_1, u_2, and u_3, where the electromagnet is turned on and off using u_{em}.

The following inverse kinematics model is used to transform the Cartesian coordinates of the end effector to angular positions.

$$\begin{aligned} q_1 &= \text{atan2}(y, x) - \text{atan2}\left(l_2 \sin(q_2), l_1 + l_2 \cos(q_2)\right) \\ q_2 &= \text{acos}\left(\frac{(x^2 + y^2) - (l_1^2 + l_2^2)}{2 l_1 l_2}\right) \end{aligned} \tag{2.27}$$

2.5.2 Neural network identification of the three DoF robot

Having a separate neural network for each output simplifies the model's structure and avoids the complexity of training one network with multiple outputs. To predict the future angles of joints 1 and 2 ($\hat{q}_1(k+1)$ and $\hat{q}_2(k+1)$), the neural networks have an input layer containing the previous and the current angular positions ($q_1(k-1)$, $q_1(k)$, $q_2(k-1)$, and $q_2(k)$) and inputs ($u_1(k-1)$, $u_1(k)$, $u_2(k-1)$, and $u_2(k)$). Each NN has a hidden layer with 7 neurons and a sigmoid activation function. The NN used to predict the future vertical position ($\hat{z}(k+1)$) has an input layer containing the previous and the current vertical position ($z(k-1)$ and $z(k)$) and inputs ($u_3(k-1)$ and $u_3(k)$), it includes one hidden layer of 4 neurons with a sigmoid activation function.

These neural networks are trained and evaluated offline using training and test datasets of 37153 and 9288 samples, respectively. The datasets are generated by applying random inputs to the arm manipulator with a sample time $T_s = 0.02$ second. Figure 2.15 gives the validation results of each neural network, and

Table 2.12 shows the associated RMSE and R^2 of the one-step-ahead predictions.

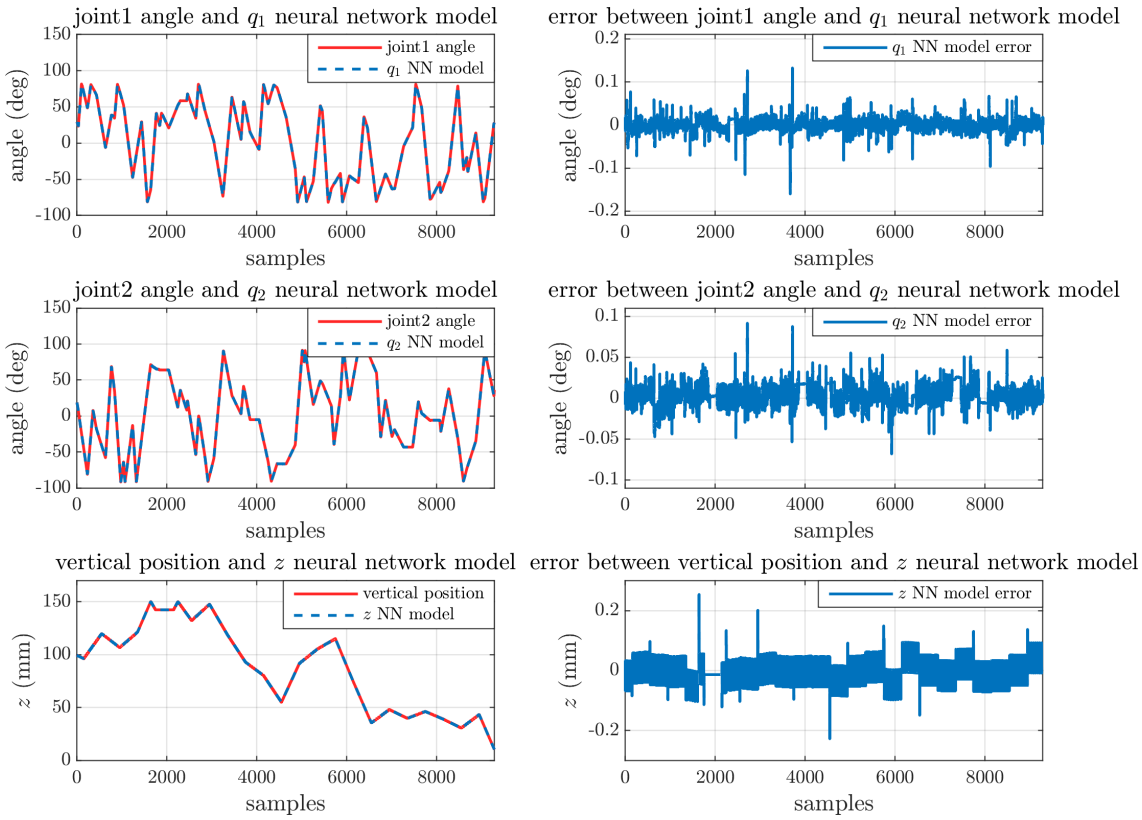


Figure 2.15: Responses of the trained neural network models for the test data

Table 2.12: RMSE and R^2 values for the trained models

Neural network	RMSE	R^2
q_1 NN model	1.4366e-2	0.99999991106
q_2 NN model	1.2291e-2	0.99999993585
z NN model	3.6994e-2	0.99999909945

Table 2.13 gives the RMSE and R^2 of the h-step-ahead predictions of the neural network models.

2.5.3 Controller implementation

The proposed NNMPC-AOA controller is implemented in the DSP board, together with the NNMPC-TLBO and the NNMPC-PSO, with a sample time of 0.02 second. A comparative study is carried out using the parameters given in Table 2.14. The AOA parameters are shown in Table 2.4, and the PSO parameters are gathered in Table 2.3.

Table 2.13: RMSE and R^2 values of the multi-step-ahead predictions of q_1 and q_2 neural networks

h	q_1 NN		q_2 NN		z NN	
	RMSE	R^2	RMSE	R^2	RMSE	R^2
1	0.0144	0.9999	0.0123	0.9999	0.0370	0.9999
2	0.0276	0.9999	0.0234	0.9999	0.0354	0.9999
3	0.0439	0.9999	0.0369	0.9999	0.0386	0.9999
4	0.0676	0.9999	0.0557	0.9999	0.0431	0.9999
6	0.2006	0.9999	0.1538	0.9999	0.0535	0.9999
8	0.7863	0.9997	0.5613	0.9999	0.0624	0.9999
10	3.3354	0.9952	2.2398	0.9979	0.0721	0.9999

Table 2.14: Controller parameters

Parameter	Value	Parameter	Value
N_1	1	R	0
N_2	3	$s_{\max}$	5
N_u	1	P_s	4

2.5.4 Sinusoidal trajectory tracking

In the first experiment, each output of the arm manipulator, free of load, is considered to follow a sinusoidal trajectory. A disturbance is applied in the time interval [9 29] seconds with an amplitude of $-30°$ in joints 1 and 2 and +30 mm in the vertical position. The obtained results are shown in Figure 2.16, and the corresponding MAE, MSE, RMSE, and the computing time are given in Table 2.15.

Table 2.15: MAE, MSE, RMSE, and computing time values (experimental study)

Controller	NNMPC-AOA	NNMPC-TLBO	NNMPC-PSO
MAE	**5.333**	5.540	5.639
MSE	**106.728**	109.328	108.728
RMSE	**17.412**	17.643	17.546
Computing time (ms)	7.527	13.314	**6.549**

All controllers have good tracking performances and can reject output disturbances. The smallest tracking error is obtained in the case of NNMPC-AOA, which shows the accuracy of the proposed controller.

The NNMPC-PSO is the fastest controller as it utilizes only two equations to update the position of the particles. In contrast, the NNMPC-AOA uses several

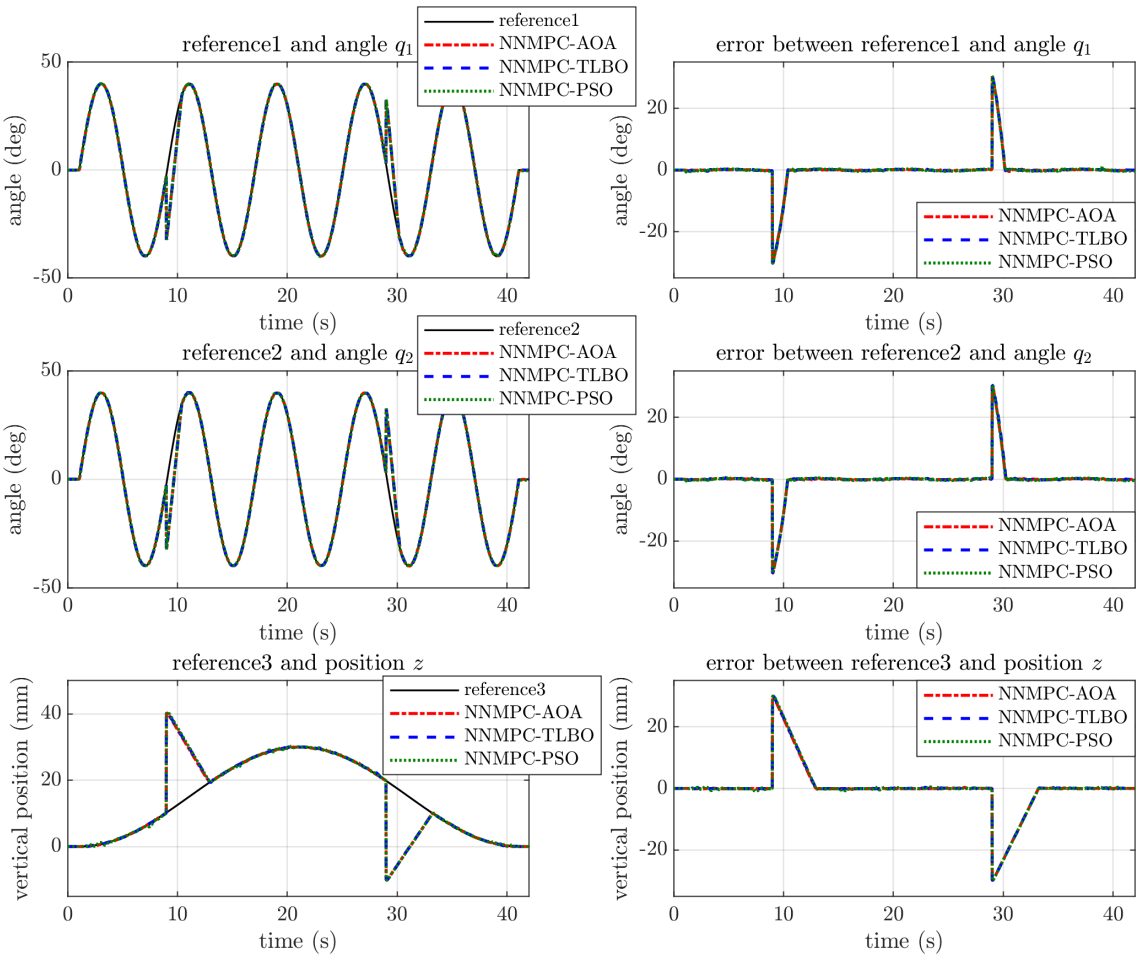


Figure 2.16: Tracking of sinusoidal trajectories

equations to update the positions of the objects, resulting in a difference of less than 1 ms in computing time. Nevertheless, the proposed controller offers good tracking accuracy, making it a valuable trade-off.

2.5.5 Multi-step trajectory tracking

In a second experiment, different weights are picked by the arm manipulator from an initial position and dropped at a final position. These weights are considered as disturbances since they are not part of the robot's model. The system's outputs are subjected to constraints limiting the overshoot to 5%, which are implemented via Equation (2.11) with $\epsilon_1 = \epsilon_2 = \epsilon_3 = 100$.

The positions with the corresponding weights are given in Table 2.16. Equation (2.27) is used to calculate the associated references.

The results of this experiment are shown in Figure 2.17, and Table 2.17 gives the MAE, MSE, RMSE, and the computing time.

From the obtained results, it can be concluded that the proposed NNMPC-AOA controller has the smallest tracking error compared to the other controllers, can

Table 2.16: Positions and weights of the used loads

	Weight (kg)	Initial position $(x,\ y)$ (m)	Final position $(x,\ y)$ (m)
Load 1	0.3	$(0.2, -0.25)$	$(0.2, 0.25)$
Load 2	0.4	$(0.2, -0.15)$	$(0.2, -0.15)$
Load 3	1.4	$(0.25, -0.1)$	$(0.25, 0.1)$

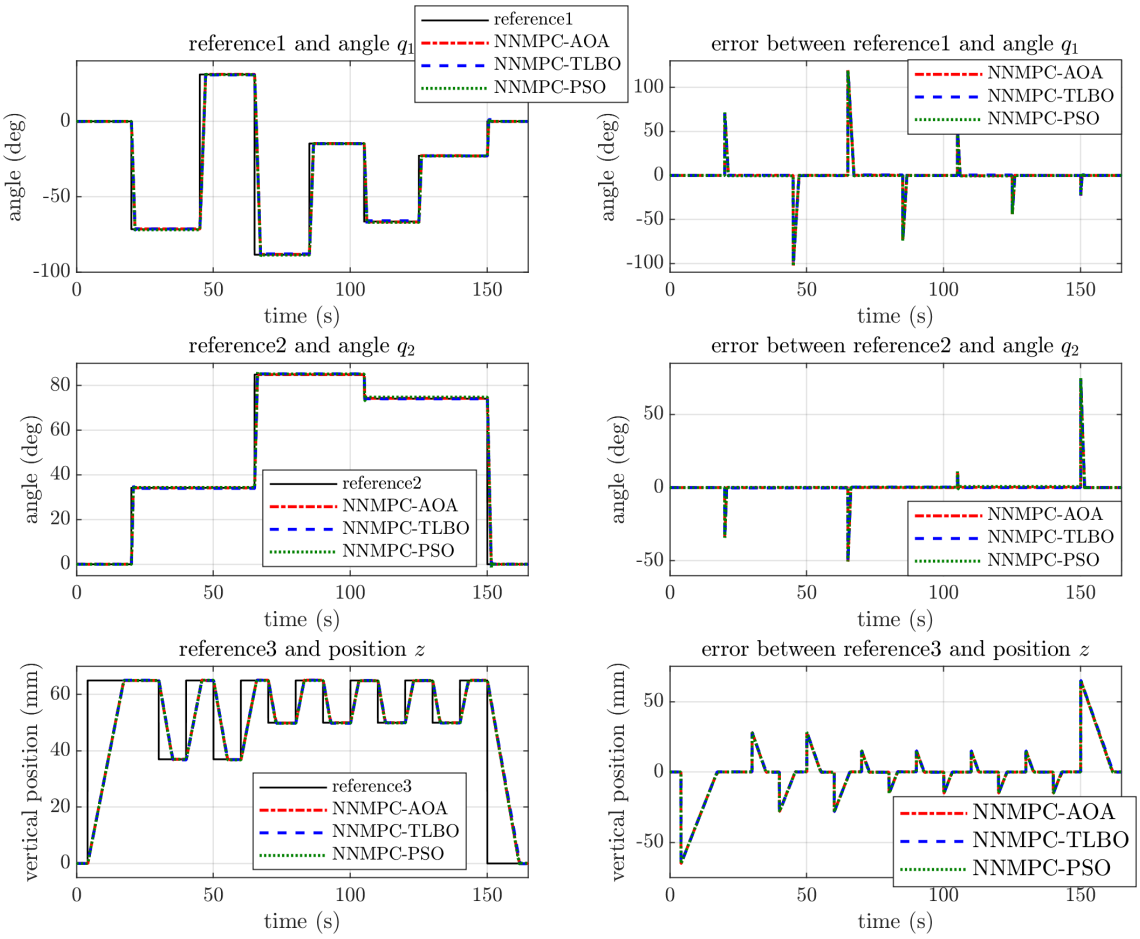


Figure 2.17: Trajectories of the arm manipulator for picking up loads

Table 2.17: MAE, MSE, RMSE, and computing time values for the case of picking up loads

Metric	NNMPC-AOA	NNMPC-TLBO	NNMPC-PSO
MAE	**11.117**	11.442	11.462
MSE	**439.834**	445.256	441.119
RMSE	**33.527**	33.744	33.612
Computing time (ms)	7.432	13.172	**6.455**

reject the considered external disturbances effect, and has a reasonable execution time.

2.6 Conclusion

A novel predictive control strategy called neural network model predictive control based on AOA was proposed in this chapter. The model of the MPC was replaced with a neural network to predict the system's future behavior; this approach provided a nonlinear model that can approximate complex systems with a simple structure. The control actions were computed using the Archimedes optimization algorithm, an efficient new algorithm with acceptable computing time. Constraints were handled during the design process and incorporated into the controller by means of the objective function and the optimization search space.

It has been shown, both in simulation and experimental studies, that good results were achieved in terms of robustness and accuracy over conventional controllers and other optimization-based NNMPC. The controller anticipates the changes in the reference, and with the help of AOA, it accordingly calculates the appropriate action while respecting the imposed limits. The proposed NNMPC-AOA controller can be used to control constrained multi-variable nonlinear systems with fast dynamics.

Chapter 3

Combination of Neural Network Model Predictive Controller with Active Disturbance Rejection Control

3.1 Introduction

This chapter presents a combination of neural network model predictive controller with active disturbance rejection control for robot trajectory tracking. This hybrid strategy showcases the controller's robustness against disturbances and its ability to reduce tracking errors. A terminal cost stabilizing constraint is introduced to the optimization problem, which is used to ensure the stability of the controlled system. The proposed controller enhances the efficiency and robustness of the model predictive controller against disturbances. The efficiency of the suggested control approach is demonstrated through experimental validation on a 4 DoF MICO robot manipulator.

3.2 MICO Robot Description

The MICO arm robot model, a lightweight serial link robotic manipulator, is taken into consideration in this study. Four rotary joints make up the main structure of the manipulator robot, which is generally controlled in three dimensions. The concepts related to Denavit-Hartenberg factors, inverse and forward kinematics, motion control, and trajectory coordination are illustrated in this section. The construction of the 4 DoF robot is shown in Figure 3.1.

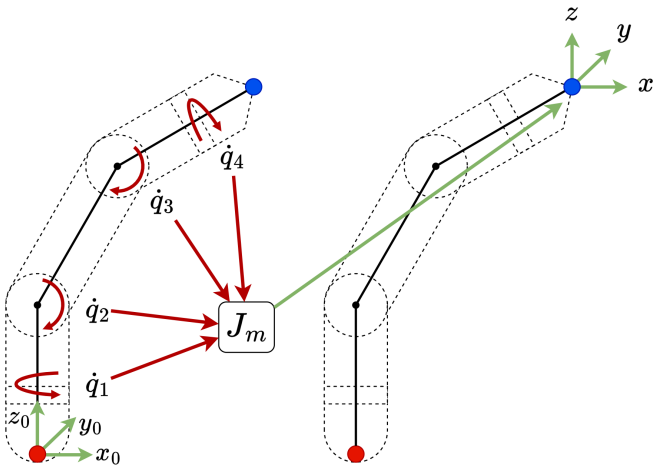


Figure 3.1: Schematic of the 4 DOF robot manipulator

The robot dynamics from Equation (2.5) can be represented as follows:

$$\ddot{q} = -M^{-1}(q)\left(V(q,\dot{q})\dot{q} + G(q) + \Omega\right) + M^{-1}(q)\tau \tag{3.1}$$

where $\Omega = F(\dot{q}) + \tau_d + \Delta h$ are the friction and the external disturbances in addition to uncertainties of the system.

The state-space model of the robot is easily derived as follows using the state variables $x_1 = q$ and $x_2 = \dot{q}$:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = f\left(x_1, x_2, \Omega\right) + bu \\ y = x_1 \end{cases} \tag{3.2}$$

And $f = -M^{-1}(q)(C(q,\dot{q})\dot{q} + G(q) + \Omega)$ is a time-varying function, with $b = M^{-1}(q)$ and $u = \tau$.

The model in Equation (3.2) is discretized with the sampling time T_s, where $t = kT_s$ with $k \in \aleph$, and the following model is obtained:

$$\begin{cases} x_1(k+1) = x_2(k) \\ x_2(k+1) = f_d\left(x_1(k), x_2(k), \Omega\right) + bu(k) \\ y(k) = x_1(k) \end{cases} \tag{3.3}$$

where f_d is the discrete function of f. The obtained form is used to design the controllers.

The FKM is used to calculate the position of the robot's end-effector $X = [x, y, z]^T$, with respect to the base coordination. This calculation requires knowledge of the robot's homogeneous transformation matrix. The FKM is derived as follows [165]:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (-c_1c_2s_3 - c_1c_3s_2)(l_3 + l_4) + c_1c_2l_2 \\ (-c_2s_1s_3 - c_3s_1s_2)(l_3 + l_4) + s_1c_2l_2 \\ (-c_2c_3 + s_2s_3)(l_3 + l_4) - s_2l_2 + l_1 \end{bmatrix} \tag{3.4}$$

where s_i is $\sin(q_i)$ and c_i is $\cos(q_i)$.

The IKM, on the other hand, is used to determine the joint angles given the position of the end effector. It essentially solves the inverse problem of the FKM. The IKM uses the end-effector position X derived from the FKM and geometric equations.

A potential solution for the inverse kinematic model is given by:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{y}{x}\right) \\ -(\psi + \varphi) \\ \sin^{-1}\left(\frac{l_2^2+(l_3+l_4)^2-(x^2+y^2+(z-l_1)^2)}{2l_2(l_3+l_4)}\right) \end{bmatrix} \tag{3.5}$$

where ψ and φ are given by:

$$\begin{cases} \psi = \tan^{-1}\left(\frac{z-l_1}{\sqrt{x^2+y^2}}\right) \\ \varphi = \cos^{-1}\left(\frac{R^2+l_2^2-(l_3+l_4)^2}{2l_2R}\right) \end{cases} \tag{3.6}$$

Here, $R = \sqrt{r^2 + (z - l_1)^2}$, and $r = l_2\cos(q_2) - (l_3 + l_4)\sin(q_2 + q_3)$. Note that the position of the end effector is not impacted by the joint angle q_4, it only affects the orientation.

3.3 Trajectory Generation

3.3.1 Task space and joint space

• Task space: The task space is the physical environment where the robot. It is typically defined in Cartesian coordinates and includes the position and orientation of the robot's end effector. Trajectories in the task space describe the desired path for completing specific tasks.

• Joint space: The joint space refers to the configuration of the robot's joints. Trajectories in the joint space describe the movement of each joint over time to achieve the end effector's path in the task space.

3.3.2 Types of trajectories

a) Trapezoidal trajectory: A trapezoidal trajectory is characterized by three velocity phases: acceleration, constant velocity, and deceleration. The velocity profile resembles a trapezoid, starting with acceleration, followed by a period of constant velocity, and concluding with deceleration.

b) Polynomial trajectories: Polynomial trajectories offer smooth transitions by ensuring continuous velocity and acceleration.

The objective is to find a trajectory that connects an initial to a final configuration while satisfying other specified constraints at the endpoints (velocity and/or acceleration constraints).

Suppose that at time t_0 the joint variable satisfies:

$$\begin{cases} q_i(t_0) = q_{i,0} \\ \dot{q}_i(t_0) = v_{i,0} \end{cases} \tag{3.7}$$

and the final values at t_f are:

$$\begin{cases} q_i(t_f) = q_{i,f} \\ \dot{q}_i(t_f) = v_{i,f} \end{cases} \tag{3.8}$$

In addition, constraints on initial and final accelerations may be specified by:

$$\begin{cases} \ddot{q}_i(t_0) = \alpha_{i,0} \\ \ddot{q}_i(t_f) = \alpha_{i,f} \end{cases} \tag{3.9}$$

- Cubic trajectory: Defined by a third degree polynomial, cubic trajectory ensures that initial and final positions and velocities are met. Thus, a trajectory of a cubic form is given by:

$$q_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.10}$$

Then, the desired velocity is given as:

$$\dot{q}_i(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{3.11}$$

Combining Equations (3.10) and (3.11) with the constraints of Equations (3.7)

and (3.8) yields four equations with four unknowns:

$$\begin{cases} q_{i,0} = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 \\ \dot{q}_{i,0} = a_1 + 2a_2 t_0 + 3a_3 t_0^2 \\ q_{i,f} = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ \dot{q}_{i,f} = a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{cases} \tag{3.12}$$

This equation always has a unique solution, provided a nonzero time interval is allowed for the execution of the trajectory.

- Quintic trajectory: It is defined by a fifth degree polynomial that also accounts for initial and final accelerations, providing even smoother motion. Thus, a trajectory of a quintic form is given by:

$$q_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{3.13}$$

Combining Equation (3.13) with the constraints of Equations (3.7) to (3.9) gives:

$$\begin{cases} q_{i,0} = a_0 + a_1 t_0 + a_2 t_0^2 + a_3 t_0^3 + a_4 t_0^4 + a_5 t_0^5 \\ \dot{q}_{i,0} = a_1 + 2a_2 t_0 + 3a_3 t_0^2 + 4a_4 t_0^3 + 5a_5 t_0^4 \\ \alpha_{i,0} = 2a_2 + 6a_3 t_0 + 12a_4 t_0^2 + 20a_5 t_0^3 \\ q_{i,f} = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ \dot{q}_{i,f} = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ \alpha_{i,f} = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{cases} \tag{3.14}$$

This equation also has a unique solution.

3.4 Neural Network Model Predictive Control with Active Disturbance Rejection Control

3.4.1 NNMPC formulation

In this chapter, the objective function is enhanced by incorporating terminal constraints. Based on the cost function described in Equation (2.6) and the prediction model outlined in Equation (2.9), the NNMPC minimization problem is thus formulated as:

$$\begin{aligned} \kappa(k) &= \arg\min_{\kappa} J \\ &= \arg\min_{\kappa} \left[\sum_{i=N_1}^{N_2} [e(k+i \mid k)]^2 + \lambda \sum_{i=0}^{N_u} [\Delta u(k+i-1)]^2 \right] \end{aligned} \tag{3.15a}$$

Subject to:

$$e\left(k+N_2+i\right)=0 \quad \forall\, i \in [1, N_s] \tag{3.15b}$$
$$\underline{u} \leq u(k+i) \leq \bar{u} \quad \forall\, i \in [0, N_u-1] \tag{3.15c}$$
$$\underline{y} \leq \hat{y}(k+i \mid k) \leq \bar{y} \quad \forall\, i \in [N_1, N_2] \tag{3.15d}$$
$$\Delta u\left(k+N_u+i\right)=0 \quad \forall\, i \geq 0 \tag{3.15e}$$

where N_s is the constraint horizon, the upper and lower control input limitations are $\bar{u}$ and $\underline{u}$, the upper and lower system output limitations are $\bar{y}$ and $\underline{y}$, respectively.

3.4.2 ADRC principle

Robotic systems often face disturbances, such as changes in load, which can affect their performance. The ADRC strategy primarily focuses on real-time estimation and compensation of all external and internal disturbances.

There are three primary components of ADRC. The first component is the Tracking Differentiator (TD), which constructs the reference input and effectively manages the transient process. The second component is the ESO, which is utilized to estimate the system uncertainties and the total disturbances. Lastly, a feedback control (typically, a PD controller) is employed to track the references and compensate for the total disturbances. Figure 3.2 illustrates the components of ADRC.

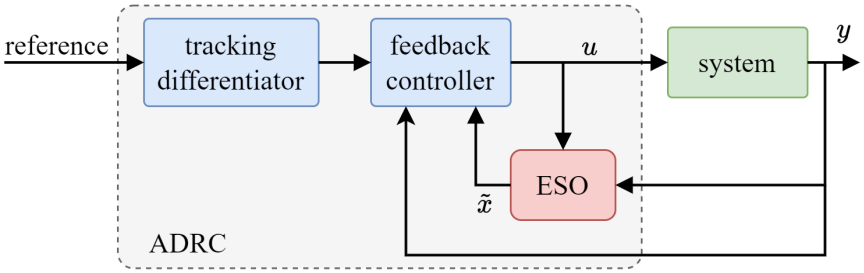


Figure 3.2: Active disturbance rejection control topology

Given the state-space model in Equation (3.2), ADRC estimates and compensates for the total disturbance, denoted as f. An extra state $x_3 = f$ is included in the state-space model to obtain a new augmented model:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 + bu \\ \dot{x}_3 = \dot{f}\left(t, x_1, x_2, \Omega\right) = \Theta \end{cases} \tag{3.16}$$

Considering $[\tilde{x}_1, \tilde{x}_2, \tilde{x}_3]^T$ as the estimation of $[x_1, x_2, x_3]^T$ and considering the first estimation error $\tilde{e}_1 = x_1 - \tilde{x}_1$, and assuming that the system is observable, the ESO

is given as:

$$\begin{cases} \dot{\tilde{x}}_1 = \tilde{x}_2 + g_1\left(\tilde{e}_1\right) \\ \dot{\tilde{x}}_2 = \tilde{x}_3 + g_2\left(\tilde{e}_1\right) + b_0 u \\ \dot{\tilde{x}}_3 = g_3\left(\tilde{e}_1\right) \end{cases} \tag{3.17}$$

where g_1, g_2, and g_3 are chosen to approximate the states of the system and its external disturbances and b_0 is the input gain.

3.4.3 NNMPC with ADRC formulation

The control block diagram of the proposed NNMPC-ADRC is shown in Figure 3.3.

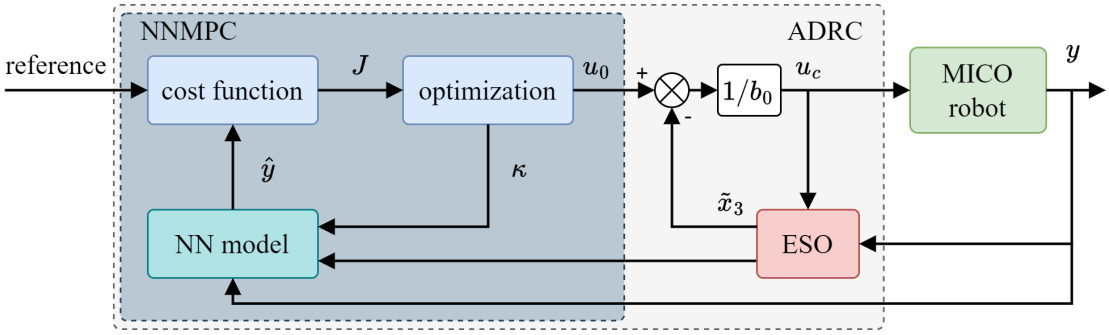


Figure 3.3: NNMPC-ADRC block diagram

After the observer is properly configured and optimized, its outputs will correspondingly approximate $[x_1, x_2, x_3]^T$ of Equation (3.16). The ADRC utilizes $\tilde{x}_3$ to actively cancel the effect of f in real time. The control law of ADRC is defined according to [123]:

$$u_c = \frac{u_0 - \tilde{x}_3}{b_0} \tag{3.18}$$

where u_0 denotes the output signal of a feedback controller.

In our case, $u_0 = u(k)$ is the optimal control computed by the NNMPC at time k. The combination of NNMPC with ADRC ensures that the robot manipulator effectively manages the disturbances.

Algorithm 2 summarizes the implementation process for the proposed NNMPC-ADRC.

Algorithm 2 NNMPC-ADRC algorithm.

1: Set the prediction parameters N_1, N_2, N_u, and N_s.
2: Set the reference trajectory along the interval $[k+N_1 \quad k+N_2]$.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Measure the current outputs of the system $y(k)$ and predict the future outputs $\hat{y}(k)$.
5: Estimate the total disturbance $\tilde{x}_3$ using the ESO.
6: Solve the optimization problem Equation (3.15a) with AOA to determine $\kappa(k)$.
7: Apply $u(k)$, the first control element of the control sequence $\kappa(k)$, as the input of the MICO robot.
8: Wait for the following sampling instant $k+1$.
9: **end for**

3.5 Stability analysis of NNMPC-ADRC

Control theory places a great deal of emphasis on ensuring system stability. Several methods exist on the stability of MPC for both linear and nonlinear dynamic systems [79,101,110,166,167]. This chapter employs a similar methodology to [168] to ensure the system's stability.

Assumption 1. *It is assumed that there exists a set of feasible solutions for the optimization problem.*

Remark 1. *While this work does not deal with the feasibility of the MPC problem, a numerical algorithm for solving the MPC problem has been presented in Section 2.3.3.*

Assumption 2. *Let a constant $\mu_e > 0$. Given that the training data is representative of the system behavior, it is assumed that the prediction error is bounded $|y - \hat{y}| \leq \mu_e$.*

Lemma 1. *Let $\kappa(k)$ be the optimal control at time k computed by solving the problem in Equation (3.15a), and let Assumptions 1 and 2 hold. The system in Equation (3.3) achieves asymptotic stability under the following conditions:*

(a) $N_s = \max(n_y + 1, n_u + 1 + N_u - N_2)$

(b) $\lambda > 0$

Proof. Considering the cost function at time k in Equation (2.6). The sub-optimal control $\kappa^*(k+1)$ at time $k+1$ is introduced as:

$$\kappa^*(k+1) = [u(k+1), \ldots, u(k+N_u)] \tag{3.19}$$

The associated cost function is defined as:

$$J^*(k+1) = \sum_{i=N_1+1}^{N_2+1} [e(k+i \mid k)]^2 + \lambda \sum_{i=1}^{N_u} [\Delta u(k+i)]^2 \tag{3.20}$$

The difference between the cost functions $J^*(k+1)$ and $J(k)$ can be written as:

$$\begin{aligned} J^*(k+1) - J(k) = &\left[e\left(k+N_2+1\right)\right]^2 + \lambda \left[\Delta u\left(k+N_u\right)\right]^2 \\ &- \left[e\left(k+N_1\right)\right]^2 - \lambda [\Delta u(k)]^2 \end{aligned} \tag{3.21}$$

Taking into account Equations (3.15b) and (3.15e) , we get $e\left(k+N_2+1\right) = 0$ and $\Delta u\left(k+N_u\right) = 0$. Equation (3.21) becomes:

$$J^*(k+1) - J(k) = -\left[e\left(k+N_1\right)\right]^2 - \lambda [\Delta u(k)]^2 < 0 \tag{3.22}$$

The cost function J is decreasing for $\lambda > 0$ and $i \leq N_2$.

Now, the function J is examined beyond the prediction horizon. For $i \geq N_2 + j$, with $j \geq 1$, the prediction equation is:

$$\begin{aligned} \hat{y}\left(k+1+N_2+j\right) &= f_{NN}\left(\chi\left(k+N_2+j\right)\right) \\ &= f_{NN}\Big(\left[u\left(k+N_2+j\right), \ldots, u\left(k+N_2+j-n_u\right),\right. \\ &\qquad \left. y\left(k+N_2+j\right), \ldots, y\left(k+N_2+j-n_y\right)\right]^T\Big) \end{aligned} \tag{3.23}$$

and:

$$\begin{aligned} \hat{y}\left(k+1+N_2+N_s\right) &= f_{NN}\left(\chi\left(k+N_2+N_s\right)\right) \\ &= f_{NN}\Big(\left[u\left(k+N_2+N_s\right), \ldots, u\left(k+N_2+N_s-n_u\right),\right. \\ &\qquad \left. y\left(k+N_2+N_s\right), \ldots, y\left(k+N_2+N_s-n_y\right)\right]^T\Big) \end{aligned} \tag{3.24}$$

Considering the constraint in Equation (3.15b), the prediction error becomes constant beyond the prediction horizon, which implies that the index of the further past system output of Equation (3.24) is beyond the prediction horizon, i.e.:

$$k + N_2 + N_s - n_y \geq k + N_2 + 1 \tag{3.25}$$

Which gives:

$$N_s \geq n_y + 1 \tag{3.26}$$

Similarly, the index of the further past system input of Equation (3.24) is beyond

the control horizon, i.e.:

$$k + N_2 + N_s - n_u \geq k + N_u + 1 \tag{3.27}$$

Which gives:

$$N_s \geq n_u + 1 + N_u - N_2 \tag{3.28}$$

Since N_s must verify either one of Equation (3.27) or Equation (3.28), we chose the biggest value between the two:

$$N_s = \max\left(n_y + 1, n_u + 1 + N_u - N_2\right) \tag{3.29}$$

The cost function J becomes monotone for all $i > N_2$. This ensures that the tracking error equality constraints are still true for all $i \geq 1$.

Equations (3.23) and (3.24) verify the constraint of Equation (3.15d). In addition, Equation (3.19) verify the constraint of Equation (3.15c), and by consequence $\Delta u^*(k+1)$ verify Equation (3.15e).

Finally, for the optimal control $\kappa(k+1)$ at time $k+1$ we have $J(k+1) \leq J^*(k+1)$:

$$J(k+1) - J(k) \leq -\left[e\left(k + N_1\right)\right]^2 - \lambda[\Delta u(k)]^2 < 0 \tag{3.30}$$

The cost function monotonically decreases if $\lambda > 0$; therefore, the control system is stable. □

Assumption 3. *It is assumed that Θ is bounded ($\Theta < \mu_\Theta$) and μ_Θ is a positive constant.*

Remark 2. *Assumption 3 is not very restrictive as, for most physical systems, the rate of change is physically limited.*

Lemma 2. *Considering the system in Equation (3.16), the ESO in Equation (3.17) is asymptotically stable if the eigenvalues of the ESO error have negative real parts.*

Proof. Let the dynamic of the ESO error be:

$$\begin{cases} \dot{\tilde{e}}_1 = \tilde{e}_2 - g_1\left(\tilde{e}_1\right) \\ \dot{\tilde{e}}_2 = \tilde{e}_3 - g_2\left(\tilde{e}_1\right) + (b - b_0)u \\ \dot{\tilde{e}}_3 = \Theta - g_3\left(\tilde{e}_1\right) \end{cases} \tag{3.31}$$

Take the Luenberger observer with $g_i(\tilde{e}_1) = \delta_i \tilde{e}_1,\ i = 1, 2, 3$, the system in

Equation (3.31) could be written on the canonical form:

$$\begin{cases} \dot{\tilde{e}}_1 = \tilde{e}_2 - \delta_1 \tilde{e}_1 \\ \dot{\tilde{e}}_2 = \tilde{e}_3 - \delta_2 \tilde{e}_1 + (b - b_0)u \\ \dot{\tilde{e}}_3 = \Theta - \delta_3 \tilde{e}_1 \end{cases} \tag{3.32}$$

To make the characteristic polynomial Hurwitz, the poles of the observer are placed at $-w_0$:

$$\lambda_0(s) = s^3 + \delta_1 s^2 + \delta_2 s + \delta_3 = (s + w_0)^3 \tag{3.33}$$

where w_0 is the observer bandwidth and $[\delta_1, \delta_2, \delta_3]^T = [3w_0, 3w_0^2, w_0^3]^T$.

Given that Assumption 3 is verified, and by choosing $w_0 > 0$, the estimation error of the ESO in Equation (3.32) is asymptotically stable. □

3.6 Experimental Study on MICO Robot

3.6.1 Experimental setup description

The experimental validation of the NNMPC and NNMPC-ADRC on the 4 DoF robot manipulator, shown in Figure 3.4, is conducted using a MATLAB/SIMULINK environment with QUARC open-source for real-time implementation. The RD 422/RS485 data acquisition card (DAQ) enables the connection between the robot and the control software.

The manipulator's dynamic model is provided in Equation (3.1), and its state-space model is represented in Equation (3.2). The goal is to track the specified joint-space trajectories of the manipulator.

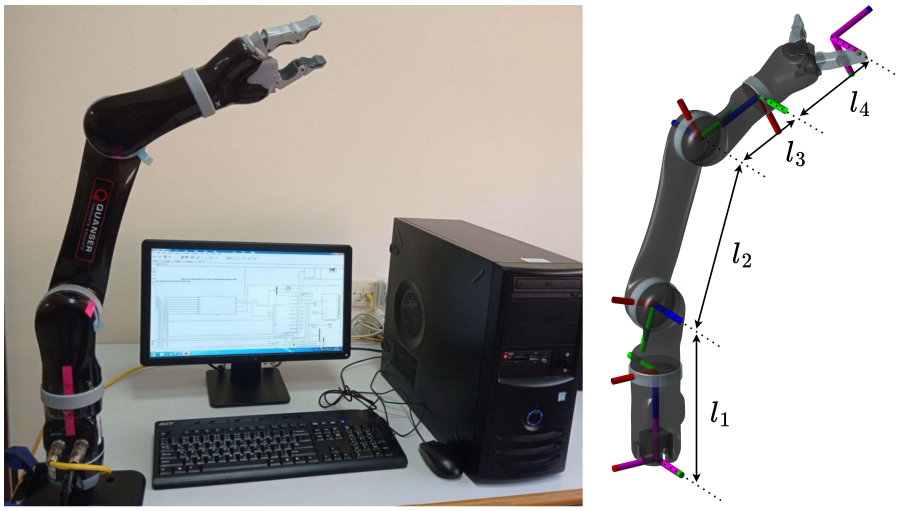


Figure 3.4: The experimental setup

The IKM is calculated using the DH parameters listed in Table 3.1.

Table 3.1: MICO robot DH parameters

Joint i	α_{i-1} (rad)	a_{i-1} (m)	d_i (m)	q_i (rad)
1	0	0	l_1	q_1
2	$-\pi/2$	0	0	q_2
3	0	l_2	0	q_3
4	$-\pi/2$	0	$l_3 + l_4$	q_4

where a_{i-1} is the distance measured along x_{i-1} from z_{i-1} to z_i, while α_{i-1} denotes the angular twist around x_{i-1} between z_{i-1} and z_i. The term d_i refers to the offset measured along z_i from x_{i-1} to x_i, q_i is the rotational angle around z_i from x_{i-1} to x_i, and l_i is the length of link i.

The physical specifications of the MICO robot are detailed in Table 3.2.

Table 3.2: MICO robot parameters

Parameter	Value (kg)	Parameter	Value (m)
m_1	0.182	l_1	0.2755
m_2	0.424	l_2	0.2900
m_3	0.211	l_3	0.1233
m_4	0.016	l_4	0.0160

3.6.2 NNMPC-ADRC experimental implementation

The NNMPC utilizes parameters in Table 3.3, with torque limits referenced from [169].

Table 3.3: NNMPC parameters

Parameter	Value	Parameter	Value
N_1	1	N_u	1
N_2	5	λ	0.0018

The neural network model used in this section incorporates a single hidden layer of five neurons. The activation functions used in the hidden and the output layers are sigmoid functions and a linear function, denoted by σ_1 and σ_2, respectively. The input layer has $n_y = 1$ and $n_u = 1$, and the output layer represents the predicted joint angles. The sampling time is $T_s = 0.02$ second.

The NN model is first trained offline with simulation data using the Levenberg–Marquardt algorithm, and then its accuracy is refined in the experiment. The data collected from several tests are used as validation data for the obtained model. The h-step-ahead prediction performance of the NN model is presented in Table 3.4

with the RMSE and the coefficient of determination R^2. The accuracy of the NN model for $h = 15$ can be seen in Figure 3.5.

Table 3.4: Neural network model multi-step-ahead prediction

Metrics	Joints	Multi-step prediction					
		1	2	4	7	10	15
R^2	q_1	0.999998	0.999983	0.999861	0.999167	0.997368	0.990817
	q_2	0.999995	0.999959	0.999652	0.998028	0.994064	0.979339
	q_3	0.999996	0.999966	0.999705	0.998376	0.995398	0.985874
RMSE	q_1	0.000632	0.001747	0.004956	0.012057	0.021482	0.040430
	q_2	0.000520	0.001437	0.004148	0.009651	0.016378	0.029326
	q_3	0.000620	0.001744	0.005122	0.011968	0.020231	0.036056

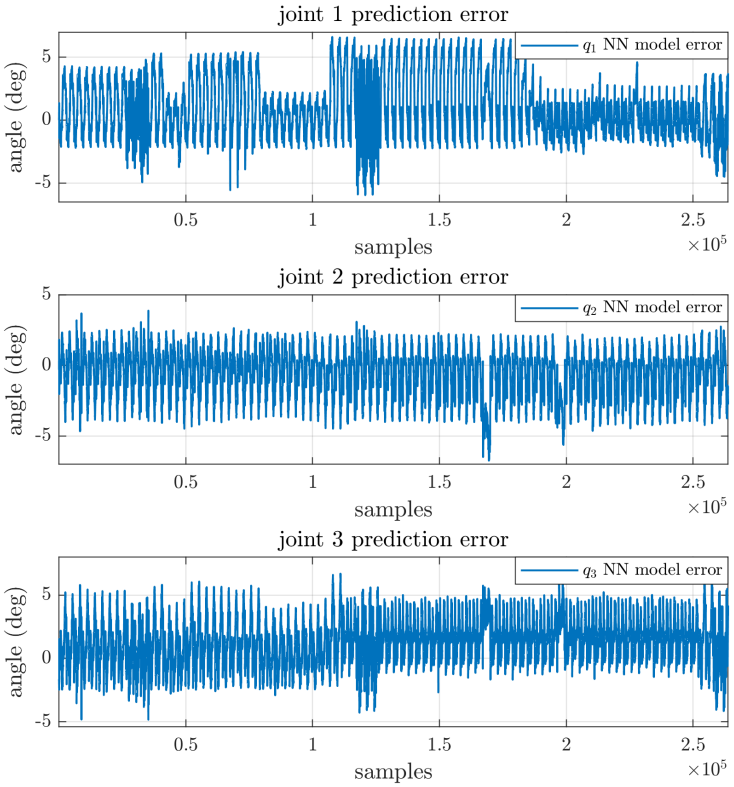


Figure 3.5: Neural network model prediction for $h = 15$

It could be noticed that even for $h = 15$, the prediction error remains small, which indicates that the neural network model has good precision. It is worth mentioning that joint 4 has been excluded from consideration as the experimental data was obtained with $q_4 = 0$.

One common strategy for addressing constraints involves reformulating the initial problem into an unconstrained variant by incorporating a penalty function. The new cost function is modified as follows:

$$\bar{J}(k) = J(k) + \nu \sum_{i=1}^{N_s} [e(k + N_2 + i)]^2 \tag{3.34}$$

with $\nu = 1$ being a positive constant and $N_s = \max(1 + 1, 1 + 1 + 1 - 5) = 2$.

The optimization algorithm accounts for all equality and inequality constraints. The goal is now to solve the unconstrained optimization problem:

$$\kappa(k) = \arg\min_{\kappa} \bar{J} \tag{3.35}$$

The parameters of the AOA are given in Table 3.5:

Table 3.5: AOA parameters

Parameter	Value	Parameter	Value
C_1	2	lb	0.1
C_2	6	ub	0.9
C_3	2	s_{max}	15
C_4	0.5	P_s	40

3.6.3 Circular trajectory tracking with no disturbances

The experiment results include tracking the joint space trajectories and corresponding errors using both NNMPC and NNMPC-ADRC. The observer gains of the system are set with $w_0 = [30,\ 30,\ 50,\ 50]^T$ and $b_0 = [0.4,\ 0.00002,\ 0.0001,\ 0.001]^T$.

The desired circular trajectory is generated in the workspace using the cubic polynomial trajectories (Section 3.3.2) to achieve smooth positions and velocities profiles. Then, the joint space desired trajectories are subsequently obtained using the IKM given in Equation (3.5). Figure 3.6 displays the joint space tracking and errors for both the NNMPC and NNMPC-ADRC with no disturbances. Table 3.6 provides performance metrics, which are: MAE, MSE, and RMSE.

Table 3.6: Joint tracking error with no disturbances

Joint	Controller	MAE	MSE	RMSE
q_1 error	NNMPC-ADRC	**4.1680e-03**	**2.6423e-05**	**5.1403e-03**
	NNMPC	4.3348e-03	2.8495e-05	5.3381e-03
q_2 error	NNMPC-ADRC	**4.2667e-03**	**2.7684e-05**	**5.2616e-03**
	NNMPC	4.6112e-03	3.3445e-05	5.7832e-03
q_3 error	NNMPC-ADRC	**4.7174e-03**	**3.8350e-05**	**6.1927e-03**
	NNMPC	4.9855e-03	4.4012e-05	6.6342e-03

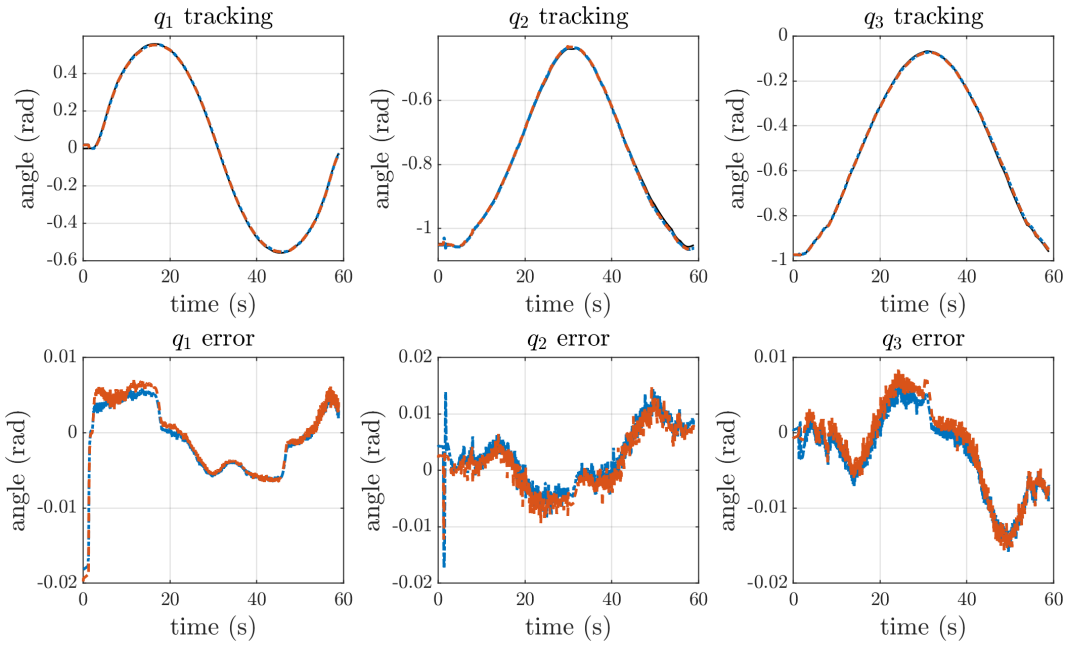


Figure 3.6: Trajectories tracking and tracking errors at each joint without disturbances (reference —, NNMPC-ADRC -·-·-, NNMPC - - -)

Both controllers offer good tracking of the desired trajectories. However, the NNMPC-ADRC is slightly better than the standard NNMPC. Initial spikes in tracking errors for joints 1, 2, and 3 are due to the end effector's initial position. In the absence of disturbances, the NNMPC-ADRC still compensates for the uncertainties of parameters and unmodeled dynamics.

The required torques by both controllers are compared in Figure 3.7.

Moreover, to assess and demonstrate the ESO's effectiveness, the robot's estimated states and the estimation errors are depicted in Figure 3.8. The metrics of the estimation error of the ESO are detailed in Table 3.7.

Table 3.7: ESO estimation error with no disturbances

Estimation Error	MAE	MSE	RMSE
q_1 estimation error	2.0598e-04	7.3877e-07	8.5952e-04
q_2 estimation error	4.0538e-04	3.0392e-07	5.5129e-04
q_3 estimation error	3.3506e-04	2.2472e-07	4.7405e-04

Figure 3.8 and Table 3.7 show that the ESO has an excellent estimation with errors not exceeding 0.01 rad for joints 2 and 3. Furthermore, the estimation errors maintain lower values afterward.

The tracking of the circular trajectory in task space is illustrated in Figure 3.9. This figure highlights the capability of the proposed NNMPC-ADRC to maintain accurate tracking.

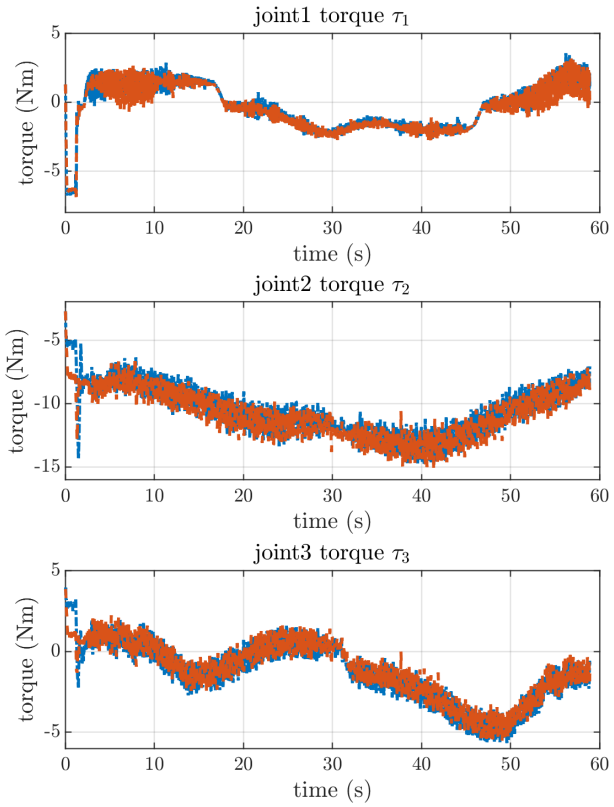


Figure 3.7: Torque signals at each joint without disturbances (NNMPC-ADRC -·-·-, NNMPC - - -)

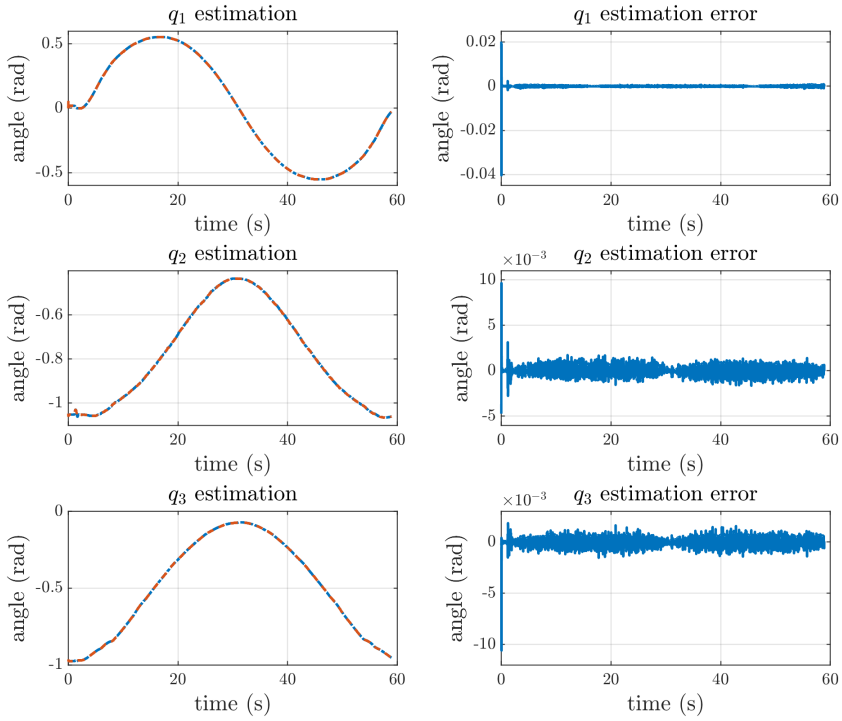


Figure 3.8: ESO estimations and errors at each joint without disturbances (NNMPC-ADRC -·-·-, ESO - - -, ESO error ——)

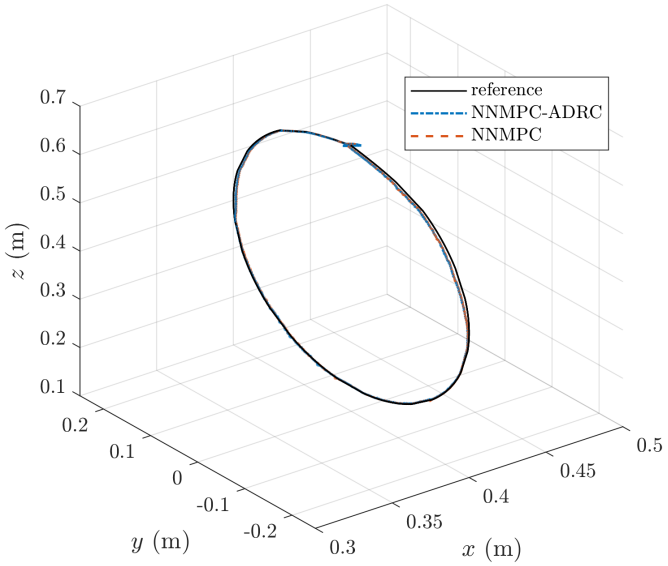


Figure 3.9: Trajectory tracking in task space without disturbances (reference ——, NNMPC-ADRC -·-·-, NNMPC - - -)

3.6.4 Case of step disturbances

An input step disturbance τ_d with an amplitude of 2 N.m is added to the control input at each joint during the interval [15 45] second. Figure 3.10 depicts the joint space tracking and errors with the step disturbances, and Table 3.8 provides the metric values for both controllers.

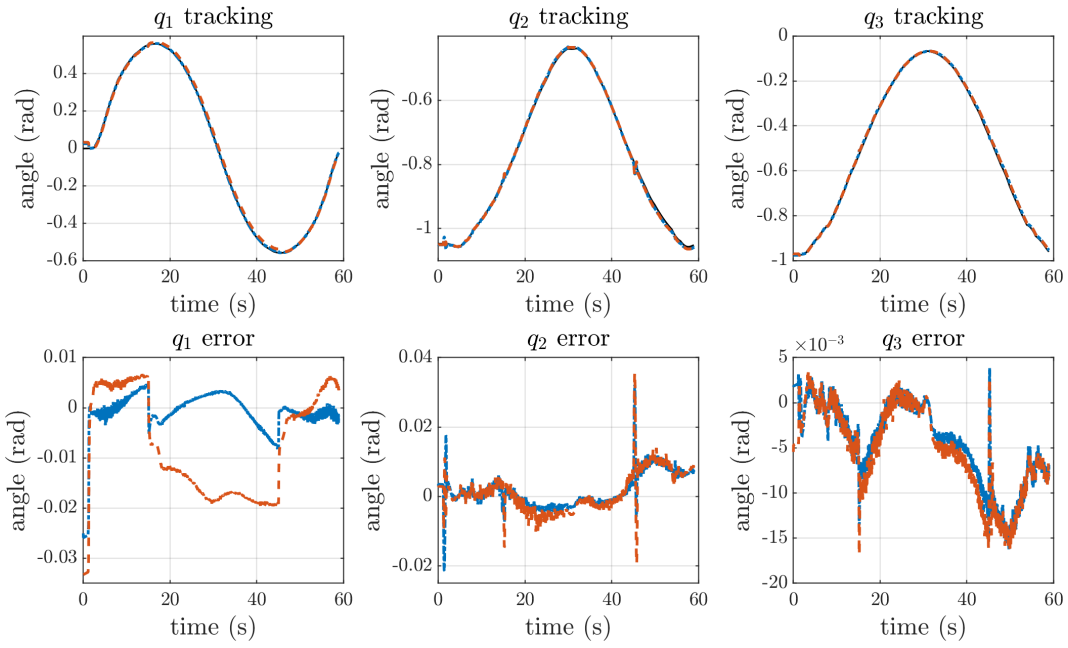


Figure 3.10: Trajectories tracking and tracking errors at each joint with step disturbances (reference ——, NNMPC-ADRC -·-·-, NNMPC - - -)

The NNMPC-ADRC outperforms the NNMPC, offering more robust disturbance rejection.

Table 3.8: Joint tracking error with step disturbances

Joint	Controller	MAE	MSE	RMSE
q_1 error	NNMPC-ADRC	**2.5836e-03**	**2.0424e-05**	**4.5193e-03**
	NNMPC	1.0601e-02	1.6466e-04	1.2832e-02
q_2 error	NNMPC-ADRC	**4.3816e-03**	**3.2909e-05**	**5.7367e-03**
	NNMPC	4.4396e-03	3.4248e-05	5.8522e-03
q_3 error	NNMPC-ADRC	**5.2646e-03**	**4.6643e-05**	**6.8296e-03**
	NNMPC	5.4468e-03	4.9235e-05	7.0168e-03

Figure 3.11 compares the required torques for disturbance rejection, indicating that NNMPC-ADRC demands less energy than NNMPC. Table 3.9 presents the ESO estimation error metrics, showing excellent estimation even in the presence of disturbances.

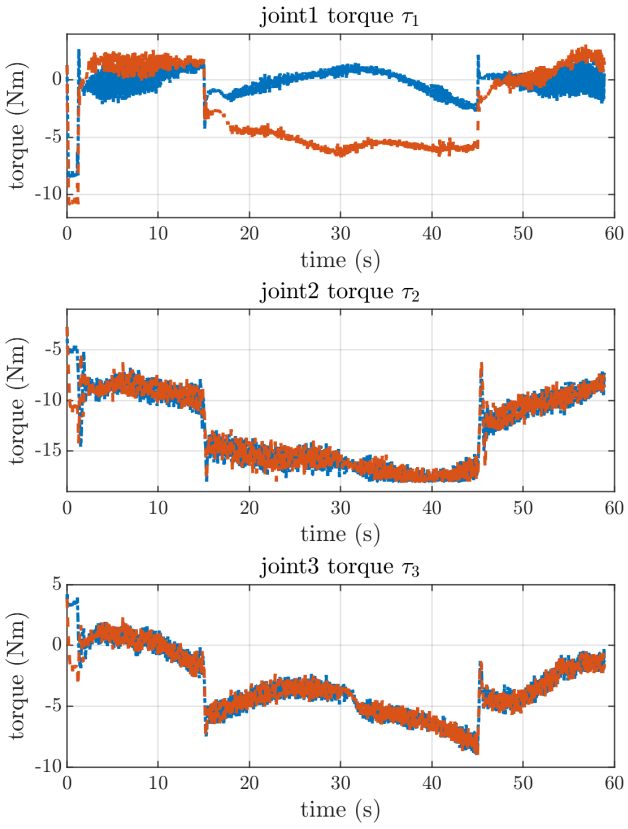


Figure 3.11: Torque signals at each joint with step disturbances (NNMPC-ADRC -·-·-, NNMPC - - -)

The task space tracking in the presence of a step disturbance is shown in Figure 3.12, demonstrating the NNMPC-ADRC's robustness against step disturbances.

The experimental findings showcase the success of the suggested NNMPC-ADRC in the tracking of desired trajectories and disturbance rejection, outperforming the standard NNMPC and demonstrating its potential for practical applications in robotic control systems.

Table 3.9: ESO estimation error with step disturbances

Estimation Error	MAE	MSE	RMSE
q_1 estimation error	2.8320e-04	3.6136e-07	6.0114e-04
q_2 estimation error	8.8671e-04	1.3803e-06	1.1749e-03
q_3 estimation error	3.4019e-04	2.5079e-07	5.0079e-04

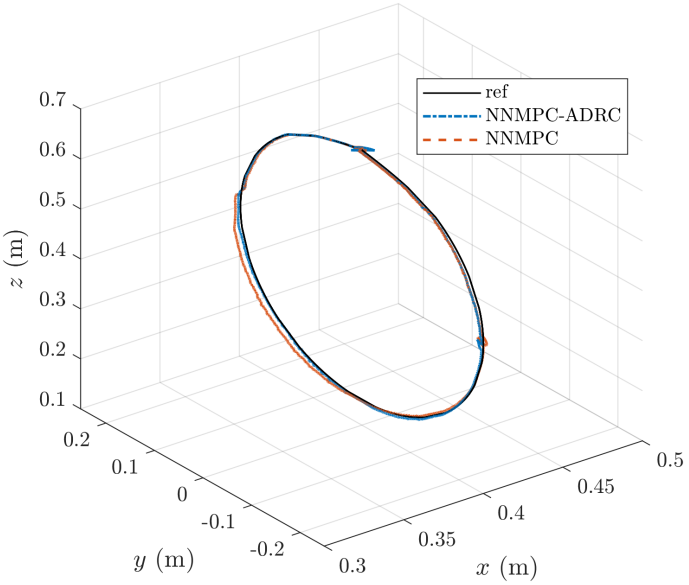


Figure 3.12: Trajectory tracking in task space with step disturbances (reference ——, NNMPC-ADRC -·-·-, NNMPC - - -)

3.6.5 Case of sinusoidal disturbances

The experimental setup also tests the controllers' ability to reject sinusoidal disturbances. The disturbance τ_d is modeled as a sinusoidal function of 2 N.m magnitude and a frequency of 1 rad/s, and it is added to the control input throughout the entire duration of the experiment. Figure 3.13 presents the joint space tracking and errors for NNMPC and NNMPC-ADRC under these sinusoidal disturbances, while Table 3.10 provides the comparative metrics for their performance.

Table 3.10: Joint tracking error with sinusoidal disturbances

Joint	Controller	MAE	MSE	RMSE
q_1 error	NNMPC-ADRC	**5.8066e-03**	**5.7841e-05**	**7.6053e-03**
	NNMPC	8.3067e-03	9.3175e-05	9.6527e-03
q_2 error	NNMPC-ADRC	**4.3444e-03**	**4.7505e-05**	**6.8924e-03**
	NNMPC	4.7240e-03	5.8027e-05	7.6176e-03
q_3 error	NNMPC-ADRC	**6.0805e-03**	**5.8027e-05**	**7.6176e-03**
	NNMPC	6.1981e-03	6.0598e-05	7.7845e-03

The obtained results, as depicted in Figure 3.13 and quantified in Table 3.10,

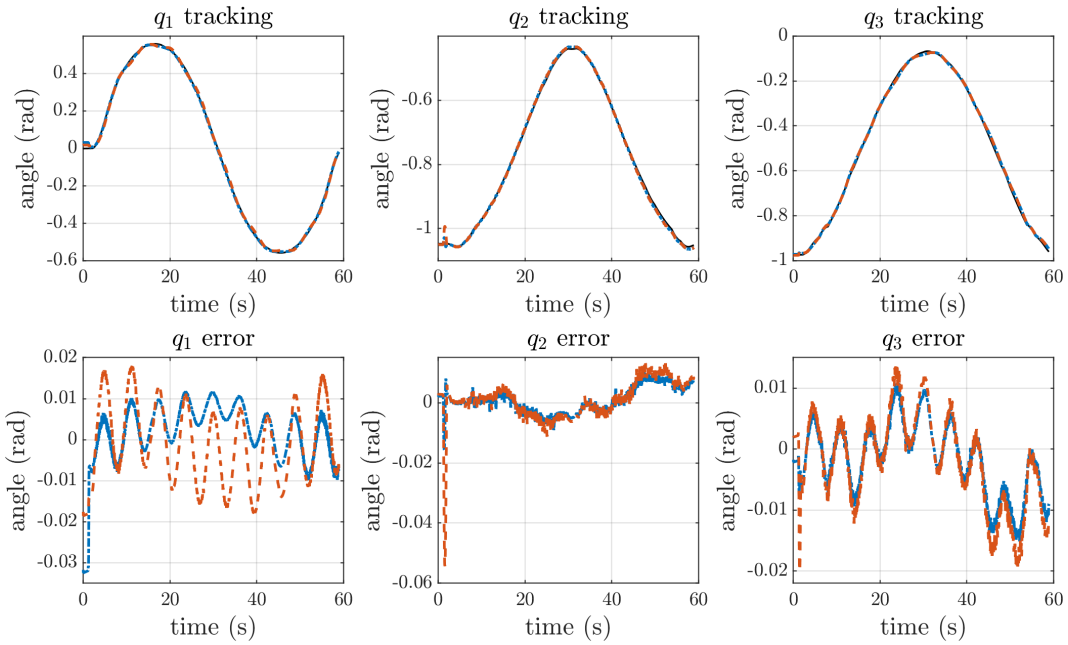


Figure 3.13: Trajectories tracking and tracking errors at each joint with sinusoidal disturbances (reference ——, NNMPC-ADRC -·-·-, NNMPC - - -)

demonstrate that NNMPC-ADRC significantly outperforms NNMPC in terms of disturbance rejection. Figure 3.14 compares the torques that both controllers require, illustrating that NNMPC-ADRC operates with greater energy efficiency.

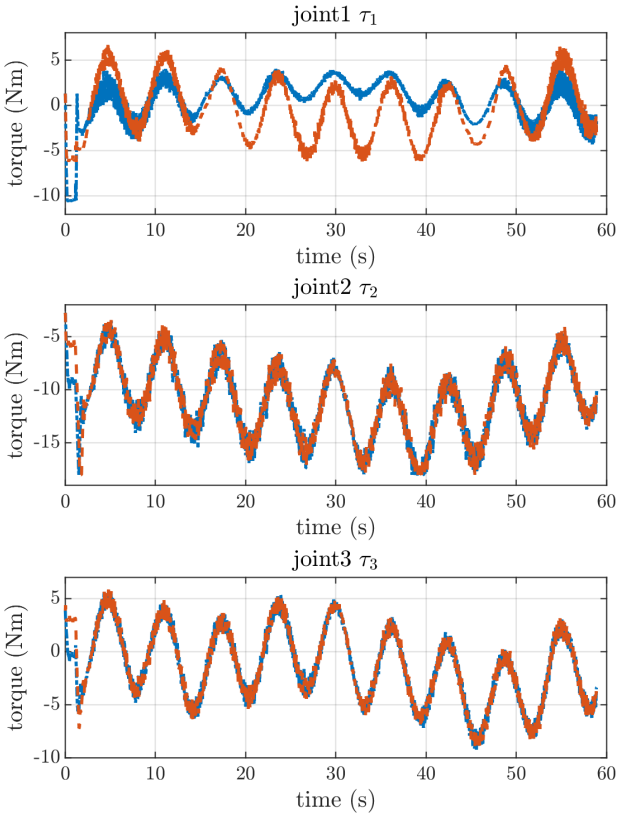


Figure 3.14: Torque signals at each joint with sinusoidal disturbances (NNMPC-ADRC -·-·-, NNMPC - - -)

Table 3.11 details the metrics of the estimation error of the ESO, confirming that despite the presence of external sinusoidal disturbances, the estimation remains precise.

Table 3.11: ESO estimation error with sinusoidal disturbances

Estimation Error	MAE	MSE	RMSE
q_1 estimation error	3.1710e-04	4.9185e-07	7.0132e-04
q_2 estimation error	9.2547e-04	1.3921e-06	1.1799e-03
q_3 estimation error	3.4193e-04	2.2862e-07	4.7814e-04

Task space tracking of the circular trajectory under the sinusoidal disturbances is shown in Figure 3.15. The experimental outcomes validate that NNMPC-ADRC maintains good tracking performance.

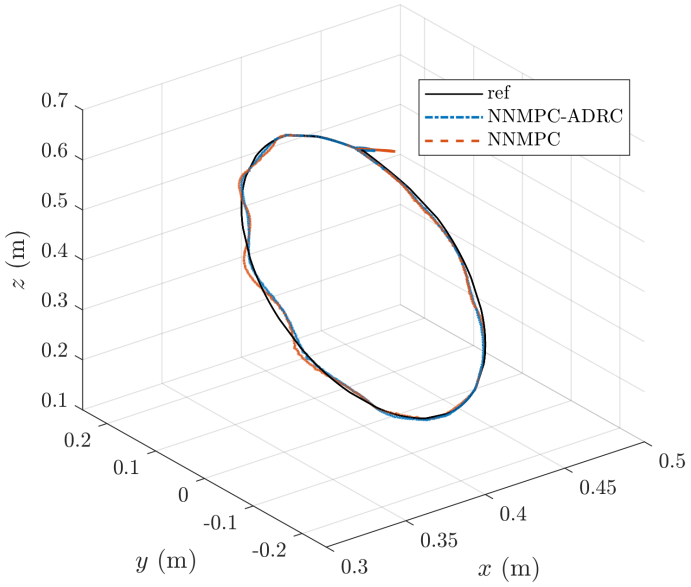


Figure 3.15: Trajectory tracking in task space with sinusoidal disturbances (reference ——, NNMPC-ADRC -·-·-, NNMPC - - -)

3.7 Conclusion

This study has demonstrated the capabilities of neural network model predictive control with active disturbance rejection control in trajectory tracking of a four DoF robot manipulator. The stability of the proposed NNMPC-ADRC strategy is guaranteed by integrating a terminal cost stabilizing constraint to the optimization problem. Compared to NNMPC, the proposed NNMPC-ADRC has shown significant improvements in tracking performance and disturbance rejection. The deployment of a neural network as the prediction model of the robot dynamics has reduced the computation time and improved the model accuracy. Additionally,

the ADRC's extended state observer has effectively estimated and compensated for total disturbances, enhancing the system's robustness against external disturbances. Experimental validation on the 4 DoF MICO robot manipulator has confirmed the practical applicability of the proposed control strategy.

Chapter 4

Advanced Control Techniques: Prescribed Performance NNMPC and Fuzzy CTC

4.1 Introduction

This chapter discusses two advanced control techniques for robot manipulators: prescribed performance NNMPC and fuzzy CTC. The first technique incorporates the prescribed performance function into the control law by integrating the transformed error in the optimization problem of the NNMPC. This combination maintains the tracking error within predefined limits, enhancing the system's transient response. Simulation on a four DoF MICO robot emphasizes the superior efficiency of the proposed controller, demonstrating reduced overshoot, small settling time, and an improved tracking of desired set-points. The second technique improves the computed torque controller through the addition of a nonlinear element, that is, the fuzzy controller, which compensates for the limitation of the CTC control action. The Archimedes optimization algorithm is utilized to optimize the FCTC parameters. The performances of the proposed controller are compared to those of the optimized CTC and PID controllers for different trajectories of a six DoF PUMA robot.

4.2 Prescribed Performance Neural Network Model Predictive Control

The first part of this chapter presents the simulation of the prescribed performance neural network model predictive control to control the 4 DoF MICO robot presented in Section 3.2 that has the model in Equation (3.1).

4.2.1 Prescribed performance function

In the literature, the theory of prescribed performance control has been utilized extensively [170] and is notably recognized for its efficacy in ensuring a system's steady-state and transient performance. The prescribed performance function should be a decreasing positive function [151], and it is usually an exponential function that has the following expression:

$$\eta(t) = (\eta_0 - \eta_\infty)e^{-\gamma t} + \eta_\infty \tag{4.1}$$

The difference vector between the desired and actual values of the robot positions at time t is called the error $e(t)$. The minimal speed of convergence is defined by $\gamma > 0$, the initial value of the PPF for $t = 0$ is η_0, and the maximum permitted steady-state error, which cannot be zero $\eta_0 > \eta_\infty > 0$, is the maximum value of PPF defined by η_∞ for $t \to \infty$. Figure 4.1 shows the recommended exponential performance function graphically. where the error's absolute value is $\mid e(t) \mid$, and the error's initial value is $e(0)$.

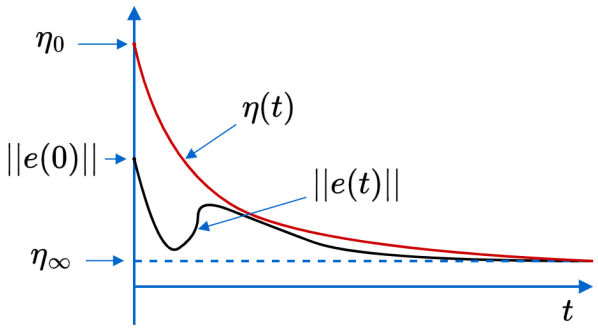


Figure 4.1: Graphical presentation of the prescribed performance function $\eta(t)$

The goal of the control law is to keep the error $e(t)$ in the limits of the PPF, so the following condition needs to be satisfied:

$$-\eta(t) < e(t) < \eta(t) \tag{4.2}$$

Since $\eta(t)$ is all the time greater than zero, Equation (4.2) can be divided by $\eta(t)$ and give the following expression:

$$-1 < \frac{e(t)}{\eta(t)} < 1 \tag{4.3}$$

Two techniques can be used to apply the PPF in the control law and to modify the constrained control law in Equation (4.2) to the unconstrained one. These techniques are the Barrier Lyapunov function and the transformed error. In this

chapter, the transformed error $\varepsilon(t)$ is adopted with $\varepsilon(t) \in \Re$, and it is defined as:

$$\varepsilon(t) = S^{-1}\left(\frac{e(t)}{\eta(t)}\right) \tag{4.4}$$

While the error-transformed function $S(\varepsilon)$ has been chosen as follows:

$$S(\varepsilon) = \frac{e(t)}{\eta(t)} \tag{4.5}$$

Equation (4.2) should be satisfied by $S(\varepsilon)$, meaning that: $-1 < S(\varepsilon) < 1$.

An alternative function satisfying the characteristics of $S(\varepsilon)$ is chosen as [143, 144]:

$$S(\varepsilon) = \frac{e^{\varepsilon} - e^{-\varepsilon}}{e^{\varepsilon} + e^{-\varepsilon}} \tag{4.6}$$

Upon computing the function $S(\varepsilon)$ inverse, the transformed error is expressed as follows:

$$\varepsilon(t) = \frac{1}{2}\ln\left(\frac{\frac{e(t)}{\eta(t)} + 1}{1 - \frac{e(t)}{\eta(t)}}\right) \tag{4.7}$$

To ensure the tracking control with the PPF, the transformed error dynamic must be stabilized.

4.2.2 PP-NNMPC formulation

Using the mentioned transformed error in Equation (4.7) to update the minimization problem of NNMPC in Equation (3.15a), we get:

$$\begin{aligned}\kappa(k) &= \arg\min_{\kappa} J_{\varepsilon} \\ &= \arg\min_{\kappa}\left[\sum_{i=N_1}^{N_2}[\varepsilon(k+i \mid k)]^2 + \lambda\sum_{i=0}^{N_u}[\Delta u(k+i-1)]^2\right]\end{aligned} \tag{4.8}$$

Subject to:

$$\begin{aligned}\varepsilon\left(k+N_2+i\right) &= 0 \quad \forall\, i \in [1, N_s] \\ \underline{u} \leq u(k+i) &\leq \bar{u} \quad \forall\, i \in [0, N_u - 1] \\ \underline{y} \leq \hat{y}(k+i \mid k) &\leq \bar{y} \quad \forall\, i \in [N_1, N_2] \\ \Delta u\left(k+N_u+i\right) &= 0 \quad \forall\, i \geq 0\end{aligned}$$

Figure 4.2 represents the control block diagram of the proposed PP-NNMPC.

4.2.3 Simulation on MICO robot

The design of NNMPC for the 4 DoF robot relies on two essential components: the first one is the neural network employed as a predictive model, and the second

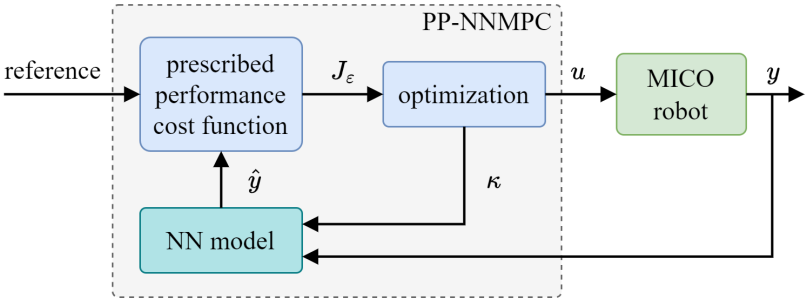


Figure 4.2: PP-NNMPC block diagram

one is the optimization algorithm that computes the control signal. The neural network architecture comprises one hidden layer of five neurons and outputs the robot's predicted joint angles. Table 4.1 gives the coefficient of determination R^2 and the root mean squared error of the multi-step prediction of the trained model. The sampling time of the system is set to $T_s = 0.02$ second. Moreover, The NNMPC and AOA parameters are given in Tables 3.3 and 3.5.

Table 4.1: Evaluation of the neural network prediction model

Metrics	Joints	Multi-step prediction				
		1	2	3	5	10
R^2	q_1	0.999992	0.999942	0.999808	0.999155	0.994567
	q_2	0.999970	0.999815	0.999453	0.997912	0.989117
	q_3	0.999968	0.999805	0.999432	0.997865	0.989202
	q_4	0.999990	0.999930	0.999761	0.998873	0.991643
RMSE	q_1	0.003969	0.010584	0.019198	0.040291	0.102209
	q_2	0.007263	0.018086	0.031088	0.060710	0.138269
	q_3	0.007529	0.018503	0.031590	0.061246	0.137568
	q_4	0.005794	0.015580	0.028721	0.062343	0.168985

The NNMPC defined in Equation (3.15a) and the PP-NNMPC defined in Equation (4.8) are designed using the same parameters and $\lambda = 2\,e-6$. Both controllers use the same neural network as a prediction model and solve the optimization problem using the Archimedes optimization algorithm. The transformed error in Equation (4.7) used for the PP-NNMPC for each joint is:

$$\varepsilon_i(k) = \frac{1}{2} \ln \left(\frac{\frac{e_i(k)}{\eta_i(k)} + 1}{1 - \frac{e_i(k)}{\eta_i(k)}} \right) \quad \forall i = 1, \ldots, n \tag{4.9}$$

where:

$$\begin{cases} \eta_1(k) = \exp(-kT_s/5) + 0.1 \\ \eta_2(k) = \exp(-kT_s/4) + 0.15 \\ \eta_3(k) = 1.2\ \exp(-kT_s/5) + 0.2 \\ \eta_4(k) = \exp(-kT_s/6) + 0.15 \end{cases} \tag{4.10}$$

The trajectory tracking performances of both controllers are evaluated through simulation on the four DoF robot. Figure 4.3 presents the tracking in joint space with the initial joints position [-0.53 -0.46 -0.26 -0.5]^T(deg), and the applied torques are shown in Figure 4.4. Both controllers provide good tracking performances. However, the proposed PP-NNMPC showcases a faster and better transient response than the NNMPC.

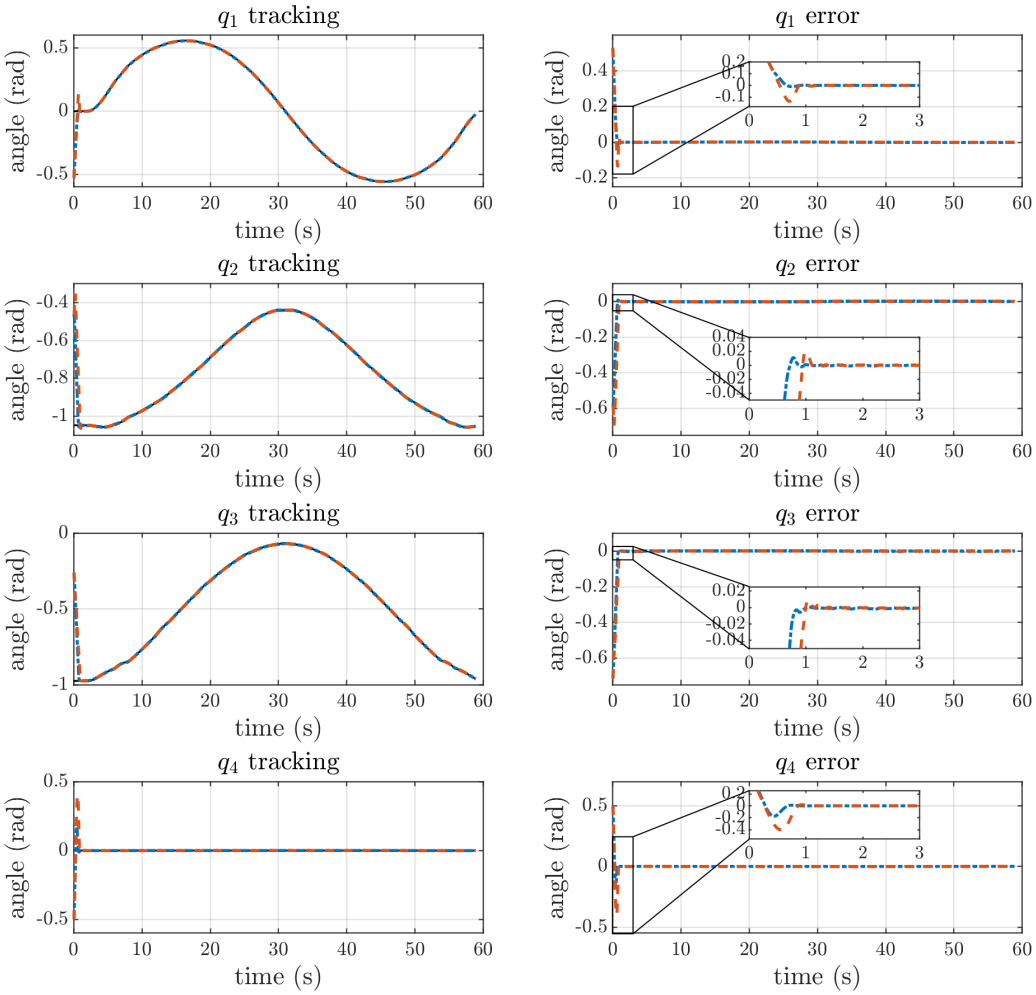


Figure 4.3: Tracking performances in joint space (reference ——, PP-NNMPC -·-·-, NNMPC - - -)

Table 4.2 gives the values of overshoot, the 2% settling time, the mean absolute error for each joint, and the mean square error. From Figure 4.3 and Table 4.2, it is clear that the PP-NNMPC assures a faster response than NNMPC with minimal overshoot and error.

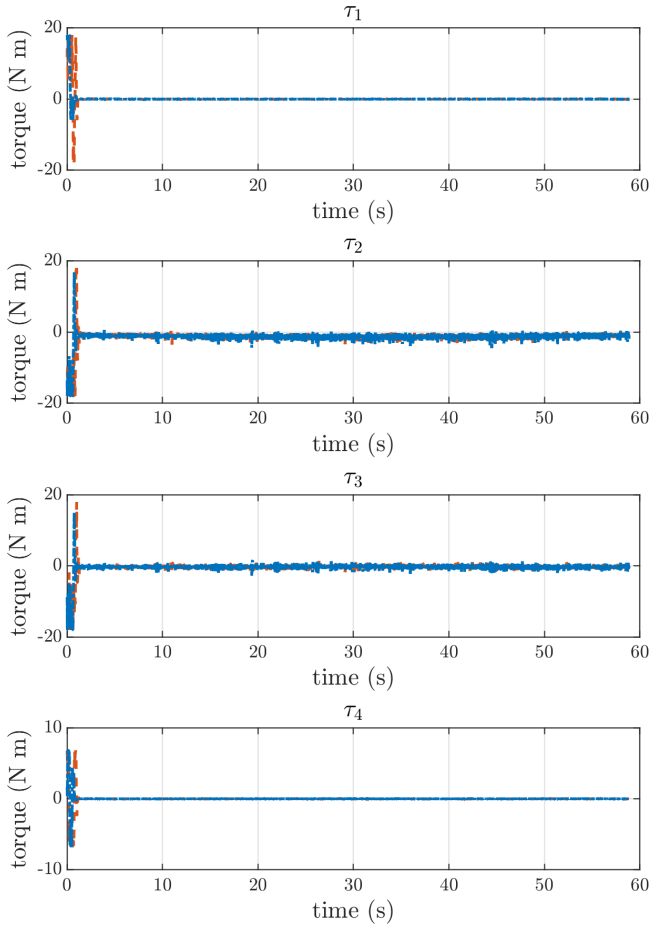


Figure 4.4: Torque signals at each joint (PP-NNMPC ·-·-·, NNMPC - - -)

Table 4.2: Performance metrics of the PP-NNMPC and NNMPC

Performance	Joint	PP-NNMPC	NNMPC
Overshoot (%)	1	01.66%	25.56%
	2	01.93%	03.08%
	3	00.10%	00.96%
	4	34.49%	80.36%
2% settling time (s)	1	0.64	1.02
	2	0.70	1.06
	3	0.78	0.98
	4	0.80	1.00
MSE	1	9.7724e-04	1.0164e-03
	2	1.6587e-03	3.5506e-03
	3	2.6061e-03	3.0035e-03
	4	6.5515e-04	1.2608e-03
MAE	1	3.7700e-03	4.0825e-03
	2	4.7369e-03	7.6247e-03
	3	5.6273e-03	6.7197e-03
	4	2.6421e-03	4.3635e-03

Figure 4.5 shows the tracking of a circular reference trajectory in the workspace for both controllers.

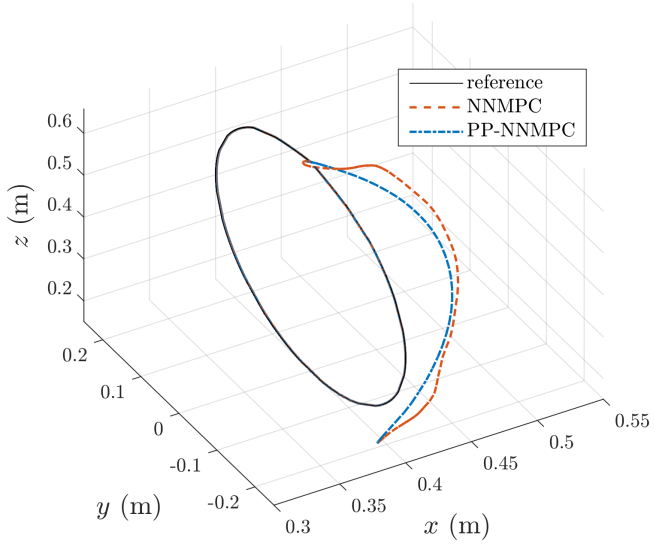


Figure 4.5: 3D representation of the tracking performances

Figure 4.6 shows the trajectories tracking of both controllers in the workspace. It is clear that the convergence of the proposed PP-NNMPC is faster with minimal overshoot, which demonstrates that it outperforms the NNMPC.

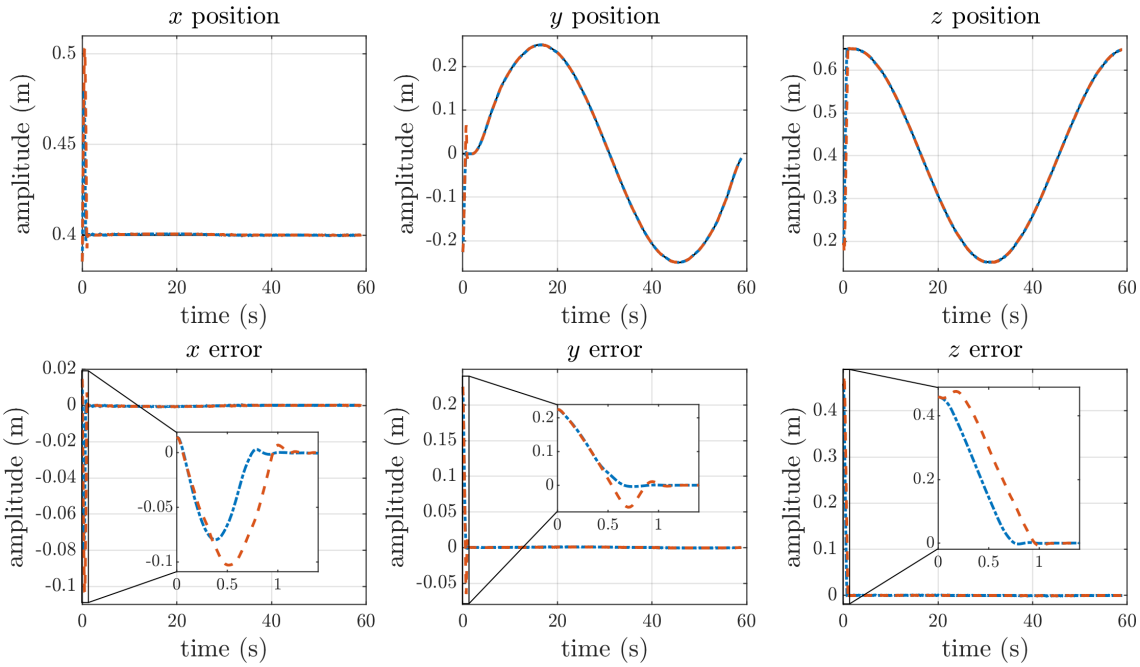


Figure 4.6: Tracking performances in workspace (reference ——, PP-NNMPC -·-·-, NNMPC - - -)

4.3 Fuzzy Computed Torque Control

The second part of this chapter presents the simulation of the fuzzy computed torque control on the 6 DoF PUMA robot.

4.3.1 PUMA robot description

The considered PUMA 560 robot has the configuration shown in Figure 4.7.

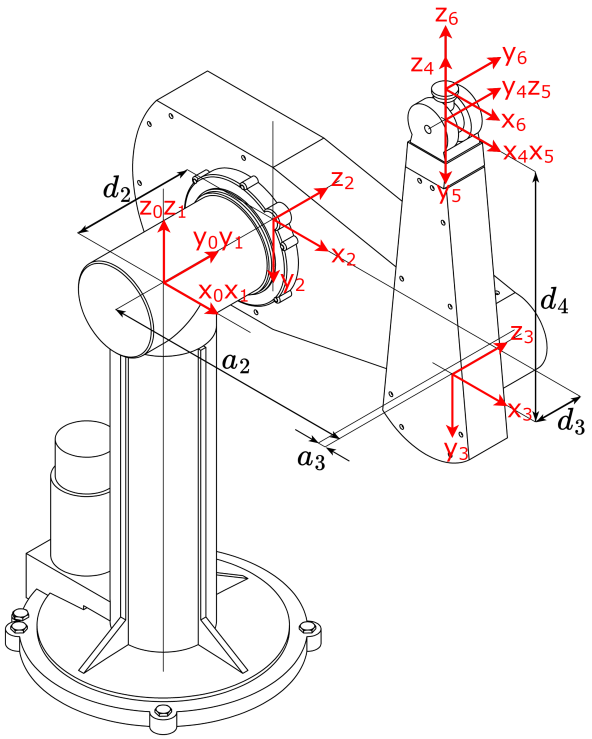


Figure 4.7: PUMA 560 with the attached coordinate frames

The Denavit-Hartenberg parameters of the PUMA manipulator shown in Table 4.3 are given in [171].

Table 4.3: DH values for PUMA 560

Link	q_i (rad)	α_i (rad)	a_i (m)	d_i (m)
1	q_1	$-\pi/2$	0	0
2	q_2	0	0.4318	0.2435
3	q_3	$\pi/2$	-0.0203	-0.0934
4	q_4	$-\pi/2$	0	0.4331
5	q_5	$\pi/2$	0	0
6	q_6	0	0	0

The following dynamic model of the six degrees of freedom PUMA robot can be found in [171]:

$$M(q)\begin{bmatrix}\ddot{q}_1\\ \ddot{q}_2\\ \ddot{q}_3\\ \ddot{q}_4\\ \ddot{q}_5\\ \ddot{q}_6\end{bmatrix} + B(q)[\dot{q}\dot{q}] + D(q)\begin{bmatrix}\dot{q}_1^2\\ \dot{q}_2^2\\ \dot{q}_3^2\\ \dot{q}_4^2\\ \dot{q}_5^2\\ \dot{q}_6^2\end{bmatrix} + G(q) = \begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\\ \tau_5\\ \tau_6\end{bmatrix} \tag{4.11}$$

The vector of velocity products is:

$$[\dot{q}\dot{q}] = [\dot{q}_1\dot{q}_2, \dot{q}_1\dot{q}_3, \ldots, \dot{q}_1\dot{q}_6, \dot{q}_2\dot{q}_3, \dot{q}_2\dot{q}_4, \ldots, \dot{q}_4\dot{q}_6, \dot{q}_5\dot{q}_6]^T \tag{4.12}$$

where $B(q)$ is the Coriolis matrix of 6×15 dimension and $D(q)$ is for the centrifugal forces of 6×6 dimension.

The input variables of the dynamic model are the torques applied at each joint (τ_1, τ_2, τ_3, τ_4, τ_5, and τ_6, respectively). This section addresses the control of the end effector position; thus, only the first three joints are considered, and the orientation of the end effector is fixed.

4.3.2 Computed torque controller

The CTC is an efficient controller that has been applied in robotics. Although it requires a knowledge of the system parameters, many researchers have proven that the parameters of robots can be measured and calculated, where a model with satisfactory fidelity could be obtained [171–173]. The control law of the CTC is given by the following equation:

$$\tau = M(q)\left(\ddot{ref} + K_v\dot{e} + K_p e\right) + B(q)[\dot{q}\dot{q}] + D(q)\left[\dot{q}^2\right] + G(q) \tag{4.13}$$

The parameters of CTC are optimized using the AOA.

4.3.3 Fuzzy CTC formulation

The proposed fuzzy computed torque controller uses the same law as the CTC and adds a fuzzy controller to the output of the CTC. The motivation for this modification is to enhance the control performance of the CTC.

The fuzzy controller is designed according to the standard design of fuzzy controllers, which consists of fuzzification, fuzzy inference, and defuzzification. The inputs of this controller are the error e and the change rate of the error Δe. The

fuzzy system output is then added to the CTC to form the control action u of FCTC. Figure 4.8 gives the control diagram of the proposed controller.

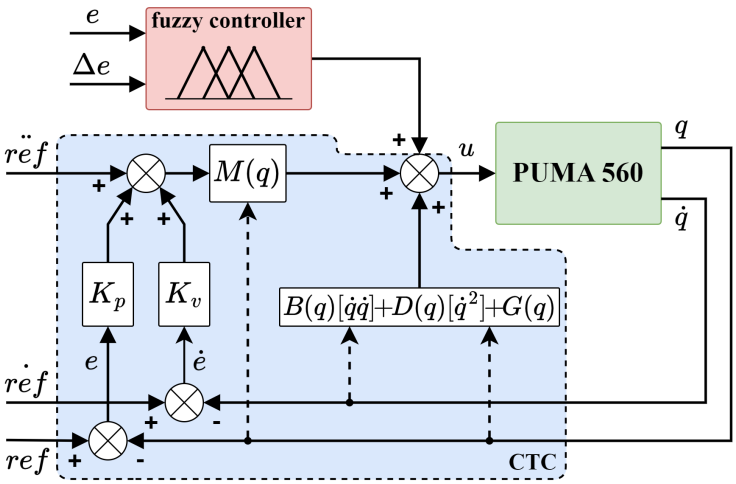


Figure 4.8: PUMA 560 manipulator control block diagram using the FCTC

a) Fuzzification:

The linguistic variables (the inputs e and Δe) can get values as Negative (N), Zero (Z), or Positive (P). The membership functions of each input are shown in Figure 4.9.

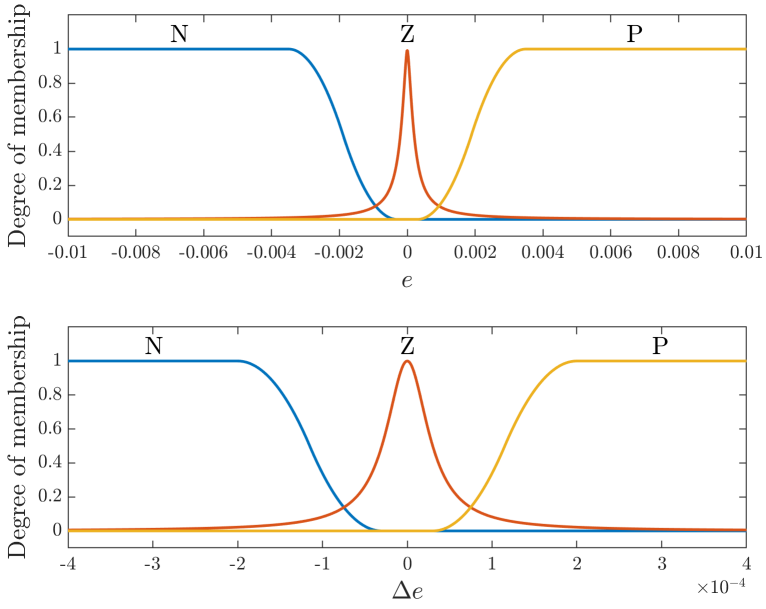


Figure 4.9: Membership functions of the inputs

b) Fuzzy inference:

The fuzzy rules used for this controller are shown in Table 4.4. The output linguistic values are: negative big, negative small, zero, positive small, and positive big (NB, NS, Z, PS, and PB, respectively).

Table 4.4: Fuzzy rules

e \ Δe	N	Z	P
N	NS	NS	NB
Z	Z	Z	Z
P	PB	PS	PS

c) Defuzzification:

This fuzzy controller uses the center of gravity defuzzification method to compute its output. The membership functions of the output are shown in Figure 4.10.

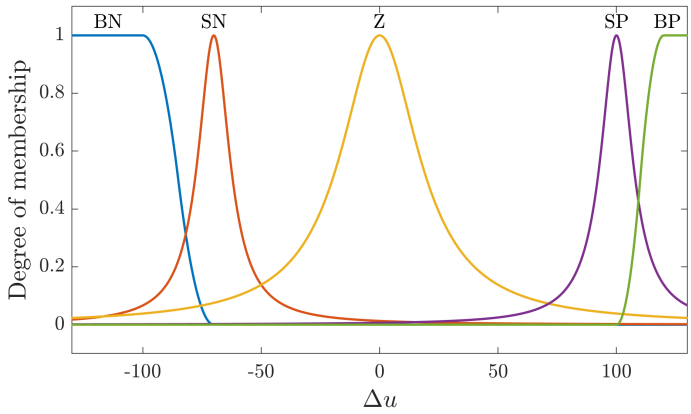


Figure 4.10: Membership functions of the output

4.3.4 Simulation on PUMA robot

The fuzzy CTC is evaluated in this section through simulation using the dynamic model of the robot manipulator PUMA 560. A spiral and square trajectories are considered for the end effector, where the desired trajectories for each joint are obtained using the inverse kinematic model. The performances of the proposed controller are evaluated against the optimized CTC and PID controllers.

The Archimedes optimization algorithm is used to tune the parameters of each controller by minimizing the RMSE between the desired reference trajectory and the obtained angle at each joint. The parameters of AOA used to tune all the controllers are summarized in Table 4.5. The optimized parameters of the controllers are: PID: $k_p = 1000$, $k_i = 1000$ and $k_d = 37.5$; CTC: $K_v = 112.5$ and $K_p = 1000$; FCTC: $K_v = 107$ and $K_p = 1000$.

In the first simulation, a spiral trajectory that ends with a circle is considered. Figure 4.11 shows the obtained joint angles' results for the PID, the CTC, and

Table 4.5: RMSE values for a spiral trajectory

parameter	value	parameter	value
C_1	2	C_4	0.5
C_2	6	P_s	20
C_3	2	s_{max}	15

the FCTC. The end effector's trajectory in the Cartesian coordinates is shown in Figure 4.12.

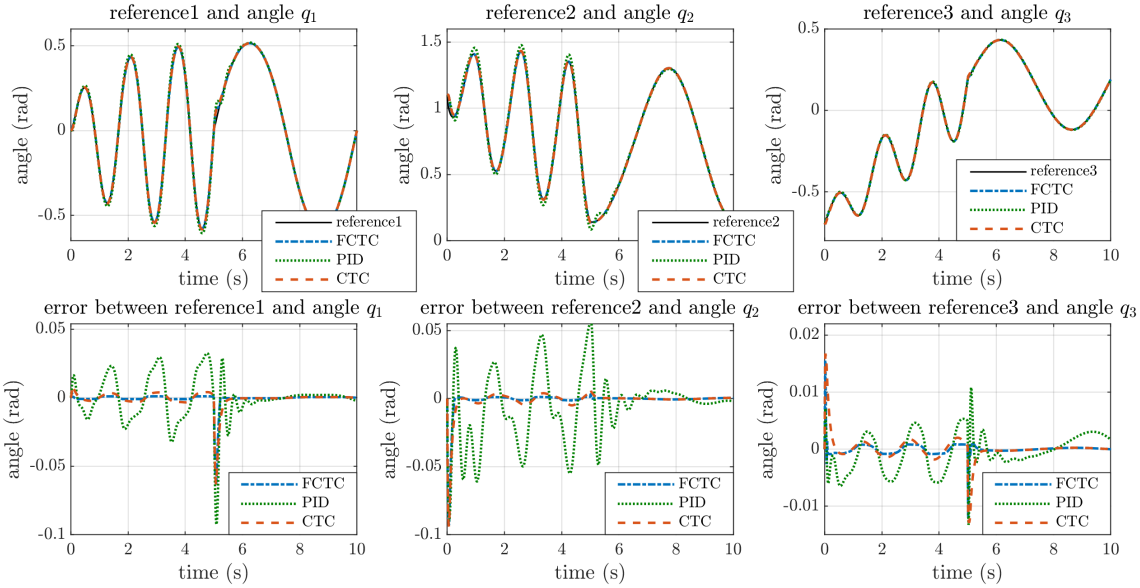


Figure 4.11: Robot angles in the case of the spiral trajectory

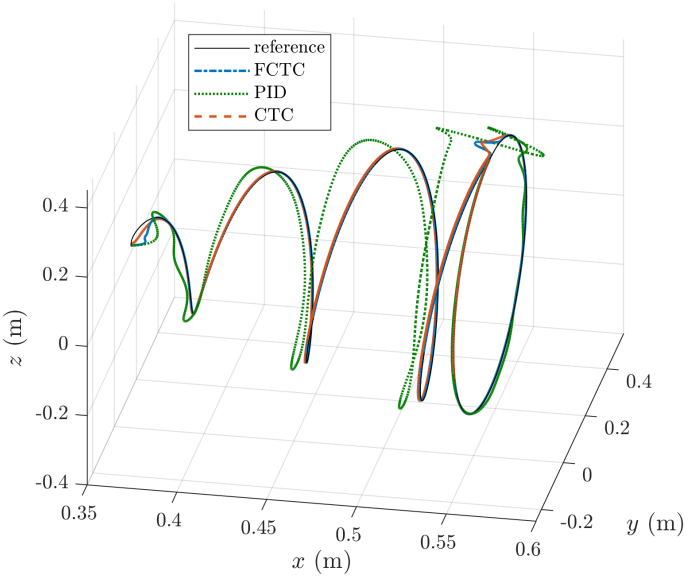


Figure 4.12: Tracking of the spiral trajectory

Table 4.6 gives the mean values calculated over 100 runs of the RMSE between the reference and the obtained angles and between the reference and the obtained space position.

Table 4.6: RMSE values for the spiral trajectory

Metrics	PID	CTC	FCTC
RMSE of angles	3.0669e-2	1.1734e-2	**1.0473e-2**
RMSE of positions	1.6341e-2	4.9660e-3	**4.5806e-3**

In a second simulation, a square trajectory is used. The obtained joint angles are shown in Figure 4.13.

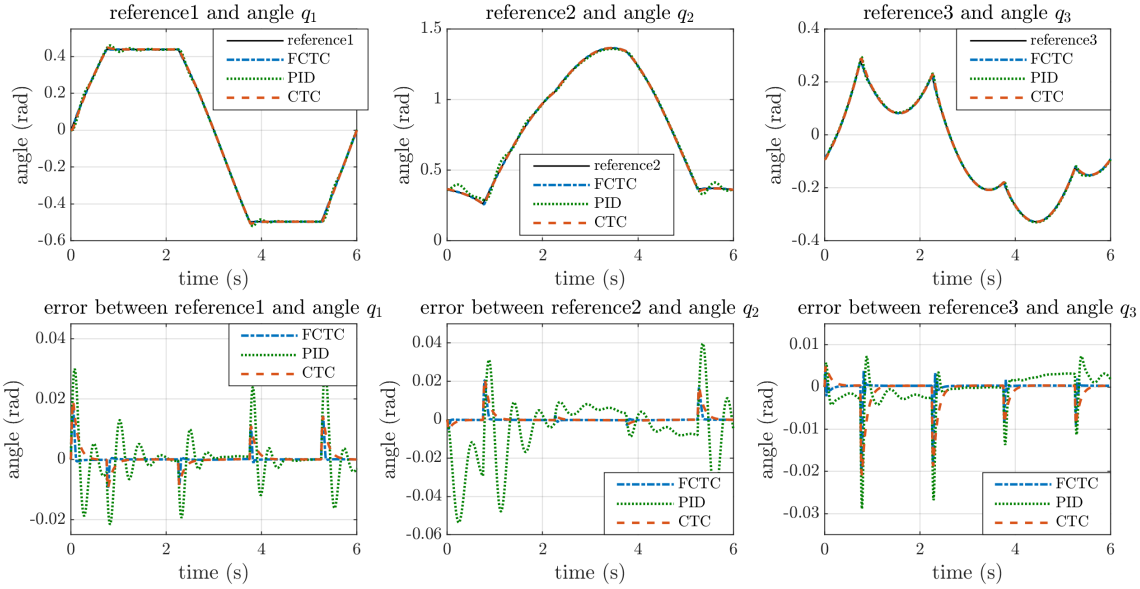


Figure 4.13: Robot angles in the case of the square trajectory

Figure 4.14 shows the corresponding tracking error in Cartesian space. For each controller, the RMSE values between the desired and obtained angles and between the desired and obtained positions in the operational space for 100 runs of the control algorithm are given in Table 4.7.

Table 4.7: RMSE values for the square trajectory

Metrics	PID	CTC	FCTC
RMSE of angles	1.9824e-2	5.5832e-3	**3.8070e-3**
RMSE of positions	1.1762e-2	2.3542e-3	**1.8035e-3**

It could be noticed from Figures 4.11 and 4.13 that the tracking error of the PID is significant. In contrast, the error of the proposed FCTC is smaller than the CTC

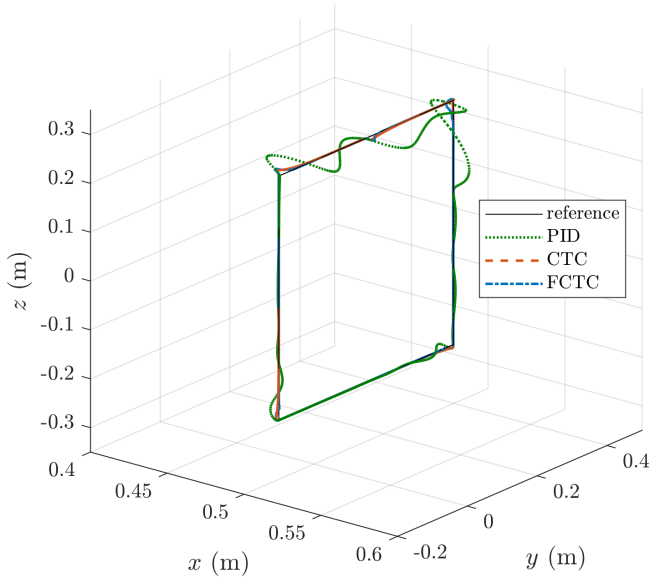


Figure 4.14: Tracking of the square trajectory

and PID controllers. Both CTC and FCTC maintain a near-zero error. Figures 4.12 and 4.14 show the tracking performances in Cartesian space; it could be noticed that the PID controller is less accurate than the CTC and FCTC in the case of a spiral trajectory. The FCTC has a minor tracking error, whereas the CTC maintains good performances in both cases. Tables 4.6 and 4.7 show that the proposed controller offers better tracking over the PID controller and the CTC for both trajectories.

4.4 Conclusion

In this chapter, the prescribed performance neural network model predictive control was designed for the four DoF robot manipulator control. The proposed approach ensures that the error stays within predefined limits, which enhances the transient response of the system. Moreover, with the use of neural networks and optimization techniques in the MPC formulation, the optimization problem is simplified, and the computational burden is reduced. The control performance of the proposed PP-NNMPC was compared to that of NNMPC. Simulation results have shown that the proposed controller ensures faster transient response with a small overshoot and error, both in joint space and workspace. Implementing the PP-NNMPC controller through experimentation on a real four DoF arm robot will be the primary goal of future research.

A fuzzy computed torque controller was also introduced in this chapter to control the robot manipulator PUMA 560, where a fuzzy controller was used in parallel to the computed torque controller. The Mamdani-type fuzzy inference system with two inputs and one output was used to enhance the performance of the CTC.

The parameters of the suggested controller were optimized using the Archimedes optimization algorithm; this optimizer was chosen since it has been proven that it provides superior results over multiple other optimization algorithms. The FCTC was then used to control the position of the end effector of the PUMA 560 robot manipulator. Comparisons between the optimized CTC, the optimized PID controller, and the FCTC were conducted for different trajectories. The obtained results have shown that the proposed controller provided better tracking performances than the other controllers.

Conclusion and Future Research Directions

The main objective of this thesis was to develop and implement intelligent control techniques for robot manipulators, using advanced control methods such as neural network model predictive control, active disturbance rejection control, prescribed performance control, and fuzzy logic control. The thesis presented four novel approaches to enhance the performance, robustness, and adaptability of robot manipulators control, and validated them through simulation and experimental studies.

The first controller integrated the Archimedes optimization algorithm with a neural network model predictive controller, forming a novel method called NNMPC-AOA. This method leveraged the meta-heuristic capabilities of AOA to optimize the control actions and improve the precision and efficiency of robot control. The method was simulated on a two DoF robot manipulator and implemented on a three DoF SCARA robot. The obtained results demonstrated the superiority of NNMPC-AOA over traditional methods, such as PID controller and CTC, and over similar approaches such as NNMPC-PSO and NNMPC-TLBO, in terms of tracking accuracy, computational time, and constraint handling.

The second control technique combined a neural network model predictive controller with active disturbance rejection control, forming a hybrid strategy called NNMPC-ADRC. This strategy showcased the controller's robustness against disturbances and its ability to maintain precise trajectory tracking. Furthermore, the stability of the proposed controller was ensured using terminal constraints. The strategy was applied on a four DoF MICO robot and compared with NNMPC. The obtained results showed that NNMPC-ADRC outperforms the other controllers in terms of disturbance rejection and tracking error.

The third approach applied prescribed performance control to the neural network model predictive controller, forming a method called PP-NNMPC. This method ensured predefined performance bounds, contributing to the controller's reliability and accuracy. The method was applied in simulation on a four DoF MICO robot and compared with NNMPC. The obtained results indicated that PP-NNMPC achieved better tracking performance and smaller tracking errors than NNMPC

while satisfying the prescribed performance function.

The fourth technique combined fuzzy logic with a computed torque controller, forming a method called FCTC. This method integrated fuzzy logic to adaptively adjust the control parameters, resulting in enhanced control precision. The method was applied in simulation on a six DoF PUMA robot and compared with PID and CTC controllers. The obtained results revealed that FCTC achieved better tracking performance and smaller tracking errors than the other controllers while handling the nonlinearities and uncertainties of the robot dynamics.

The main directions for future works are:

- Development of new techniques to apply MPC in experimental setups with reduced computation time and improved robustness: This involves exploring more efficient algorithms and computational methods to speed up the solution of the MPC problem, as well as implementing robust control into the MPC formulation.

- Integration of ADRC with other control techniques such as adaptive control and fractional order-based controllers: This involves developing hybrid control strategies that combine the robustness and disturbance rejection capabilities of ADRC with the adaptability and precision of other advanced control methods.

- Exploitation of nonlinear ADRC techniques: This involves extending the ADRC framework to nonlinear formulation with nonlinear ESO, and investigating the performance and robustness of the resulting nonlinear ADRC methods.

- Investigation of machine learning techniques for intelligent control: This involves exploring the use of machine learning methods, such as reinforcement learning and deep learning, to enhance the learning and adaptation capabilities of the intelligent control techniques. This could potentially lead to more autonomous and intelligent robot manipulators that can learn from and adapt to their environment and tasks.

Bibliography

[1] Y. Gao and S. Chien, "Review on space robotics: Toward top-level science through space exploration," *Science Robotics*, vol. 2, no. 7, 2017. (Cited on page 1)

[2] J. Yuh, "Design and control of autonomous underwater robots: A survey," *Autonomous Robots*, vol. 8, no. 1, pp. 7–24, 2000. (Cited on page 1)

[3] L. D. Evjemo, T. Gjerstad, E. I. Grøtli, and G. Sziebig, "Trends in Smart Manufacturing: Role of Humans and Industrial Robots in Smart Factories," *Current Robotics Reports*, vol. 1, no. 2, pp. 35–41, 2020. (Cited on page 1)

[4] R. H. Taylor, A. Menciassi, G. Fichtinger, P. Fiorini, and P. Dario, "Medical Robotics and Computer-Integrated Surgery," *Springer Handbooks*, pp. 1657–1684, 2016. (Cited on page 1)

[5] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 90–103, 2007. (Cited on page 1)

[6] Z. Qian and Z. Bi, "Recent Development of Rehabilitation Robots," *Advances in Mechanical Engineering*, vol. 7, no. 2, 2015. (Cited on page 1)

[7] O. Mubin, C. J. Stevens, S. Shahid, A. A. Mahmud, and J.-J. Dong, "a Review of the Applicability of Robots in Education," *Technology for Education and Learning*, vol. 1, no. 1, 2013. (Cited on page 1)

[8] R. Gockley, A. Bruce, J. Forlizzi, M. Michalowski, A. Mundell, S. Rosenthal, B. Sellner, R. Simmons, K. Snipes, A. C. Schultz, and J. Wang, "Designing robots for long-term social interaction," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005, pp. 1338–1343. (Cited on page 1)

[9] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020. (Cited on pages 5, 6, and 20)

[10] Singh, Tarun Pratap and Suresh, P and Chandan, Swet, "Forward and inverse kinematic analysis of robotic manipulators," *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, pp. 1459–1468, 2017. (Cited on page 7)

[11] R.P. Dayal, BBVL Deepak, Devedutta Nayak, and Dkk., "Forward and Kinematics Models for an Articulated Robotic Manipulator," *International Journal of Artificial Intelligence and Computational Research*, vol. 4, no. 2, pp. 1–7, 2012. (Cited on page 8)

[12] R. Johansson, A. Robertsson, K. Nilsson, and M. Verhaegen, "State-space system identification of robot manipulator dynamics," *Mechatronics*, vol. 10, no. 3, pp. 403–418, 2000. (Cited on page 8)

[13] Z. Liu, K. Peng, L. Han, and S. Guan, "Modeling and Control of Robotic Manipulators Based on Artificial Neural Networks: A Review," *Iranian Journal of Science and Technology - Transactions of Mechanical Engineering*, vol. 47, no. 4, pp. 1307–1347, 2023. (Cited on pages 8 and 9)

[14] L. U. Odhner and A. M. Dollar, "The smooth curvature model: An efficient representation of Euler-Bernoulli flexures as robot joints," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 761–772, 2012. (Cited on page 8)

[15] S. M. Sadati, S. E. Naghibi, A. Shiva, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Mechanics of continuum manipulators, a comparative study of five methods with experiments," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10454 LNAI, 2017, pp. 686–702. (Cited on page 8)

[16] A. M. Shafei and H. Mirzaeinejad, "A novel recursive formulation for dynamic modeling and trajectory tracking control of multi-rigid-link robotic manipulators mounted on a mobile platform," in *Proceedings of the Institution of Mechanical Engineers. Part I: Journal of Systems and Control Engineering*, vol. 235, no. 7, 2021, pp. 1204–1217. (Cited on page 8)

[17] V. Satya Durga Manohar Sahu, P. Samal, and C. Kumar Panigrahi, "Modelling, and control techniques of robotic manipulators: A review," *Materials Today: Proceedings*, vol. 56, pp. 2758–2766, 2022. (Cited on page 8)

[18] A. Ghoul, K. Kara, S. Djeffal, M. Benrabah, and M. L. Hadjili, "Inverse Kinematic Model of Continuum Robots Using Artificial Neural Network," in *2022 19th IEEE International Multi-Conference on Systems, Signals and Devices, SSD 2022*, 2022, pp. 1893–1898. (Cited on page 9)

[19] A. Ghoul, K. Kara, S. Djeffal, M. Benrabah, and M. L. Hadjili, "Artificial neural network for solving the inverse kinematic model of a spatial and planar variable curvature continuum robot," *Archive of Mechanical Engineering*, vol. 69, no. 4, pp. 595–613, 2022. (Cited on page 9)

[20] E. Jiménez-López, D. Servín De La Mora-Pulido, L. A. Reyes-Ávila, R. Servín De La Mora-Pulido, J. Melendez-Campos, and A. A. López-Martínez, "Modeling of Inverse Kinematic of 3-DoF Robot, Using Unit Quaternions and Artificial Neural Network," *Robotica*, vol. 39, no. 7, pp. 1230–1250, 2021. (Cited on pages 9 and 22)

[21] S. Baressi Šegota, N. Anđelić, M. Šercer, and H. Meštrić, "Dynamics Modeling of Industrial Robotic Manipulators: A Machine Learning Approach Based on Synthetic Data," *Mathematics*, vol. 10, no. 7, 2022. (Cited on page 9)

[22] M. Zeinali and L. Notash, "Fuzzy logic-based inverse dynamic modelling of robot manipulators," *Transactions of the Canadian Society for Mechanical Engineering*, vol. 34, no. 1, pp. 137–150, 2010. (Cited on page 9)

[23] J. J. Craig, *Introduction to robotics*. Pearson Educacion, 2006. (Cited on page 9)

[24] T. S. Lee and E. A. Alandoli, "A critical review of modelling methods for flexible and rigid link manipulators," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 10, 2020. (Cited on page 9)

[25] P. Song, Y. Yu, and X. Zhang, "A Tutorial Survey and Comparison of Impedance Control on Robotic Manipulation," *Robotica*, vol. 37, no. 5, pp. 801–836, 2019. (Cited on page 9)

[26] D. Sun, S. Hu, X. Shao, and C. Liu, "Global stability of a saturated nonlinear PID controller for robot manipulators," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 4, pp. 892–899, 2009. (Cited on page 10)

[27] S. Mahdi Swadi, A. Ibrahim Majeed, M. Abdulhussain shuriji, and A. A. Ugla, "Design and Simulation of Robotic Arm PD Controller Based on PSO," *University of Thi-Qar Journal for Engineering Sciences*, vol. 10, no. 1, pp. 18–24, 2019. (Cited on page 10)

[28] I. Cervantes and J. Alvarez-Ramirez, "On the PID tracking control of robot manipulators," *Systems and Control Letters*, vol. 42, no. 1, pp. 37–46, 2001. (Cited on page 10)

[29] Y. Choi and Wan Kyun Chung, "On the optimality and performance of PID controller for robotic manipulators," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2, 2001, pp. 1142–1148. (Cited on page 10)

[30] Y. Choi and W. K. Chung, "PID performance tuning methods for a robotic manipulator based on ISS," *Asian Journal of Control*, vol. 5, no. 2, pp. 206–216, 2003. (Cited on page 10)

[31] H. Rahimi Nohooji, A. Zaraki, and H. Voos, "Actor–critic learning based PID control for robotic manipulators," *Applied Soft Computing*, vol. 151, 2024. (Cited on page 10)

[32] O. K. Bruno Siciliano, *Springer Handbook of Robotics*, O. K. Bruno Siciliano, Ed. Springer Berlin, Heidelberg, 2008. (Cited on pages 10 and 20)

[33] D. Nguyeii-Tuoiig, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *Proceedings of the American Control Conference*, 2008, pp. 212–217. (Cited on page 10)

[34] F. Piltan and M. Yarmahmoudi, "PUMA-560 robot manipulator position computed torque control methods using MATLAB/SIMULINK and their integration into graduate nonlinear control and MATLAB courses," *International Journal of Robotics and Automation*, vol. 3, no. 3, pp. 167–191, 2012. (Cited on page 10)

[35] T. P. Leung, "A Sliding Mode Controller with Bound Estimation for Robot Manipulators," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 2, pp. 208–214, 1993. (Cited on page 11)

[36] K. D. Young, V. I. Utkin, and Ü. Özgüner, "A control engineer's guide to sliding mode control," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 3, pp. 328–342, 1999. (Cited on page 11)

[37] M. Hadi Barhaghtalab, V. Meigoli, M. R. Golbahar Haghighi, S. A. Nayeri, and A. Ebrahimi, "Dynamic analysis, simulation, and control of a 6-DOF IRB-120 robot manipulator using sliding mode control and boundary layer method," *Journal of Central South University*, vol. 25, no. 9, pp. 2219–2244, 2018. (Cited on page 11)

[38] S. Tayebi-Haghighi, F. Piltan, and J. M. Kim, "Robust composite high-order super-twisting sliding mode control of robot manipulators," *Robotics*, vol. 7, no. 1, 2018. (Cited on page 11)

[39] B. Brahmi, M. Driscoll, M. H. Laraki, and A. Brahmi, "Adaptive high-order sliding mode control based on quasi-time delay estimation for uncertain robot manipulator," *Control Theory and Technology*, vol. 18, no. 3, pp. 279–292, 2020. (Cited on page 11)

[40] Q. V. Doan, A. T. Vo, T. D. Le, H. J. Kang, and N. H. A. Nguyen, "A novel fast terminal sliding mode tracking control methodology for robot manipulators," *Applied Sciences (Switzerland)*, vol. 10, no. 9, 2020. (Cited on page 11)

[41] S. Hao, L. Hu, and P. X. Liu, "Second-order adaptive integral terminal sliding mode approach to tracking control of robotic manipulators," *IET Control Theory and Applications*, vol. 15, no. 17, pp. 2145–2157, 2021. (Cited on page 11)

[42] A. T. Vo and H. J. Kang, "An Adaptive Terminal Sliding Mode Control for Robot Manipulators with Non-Singular Terminal Sliding Surface Variables," *IEEE Access*, vol. 7, pp. 8701–8712, 2019. (Cited on page 11)

[43] L. Zhang, Y. Su, Z. Wang, and H. Wang, "Fixed-time terminal sliding mode control for uncertain robot manipulators," *ISA Transactions*, vol. 144, pp. 364–373, 2024. (Cited on page 11)

[44] H. Bezine, N. Derbel, and A. M. Alimi, "Fuzzy control of robot manipulators: Some issues on design and rule base size reduction," *Engineering Applications of Artificial Intelligence*, vol. 15, no. 5, pp. 401–416, 2002. (Cited on pages 11 and 12)

[45] S. A. Mazhari and S. Kumar, "Heuristic search algorithms for tuning PUMA fuzzy PID controller," *International Journal of Computer Science*, vol. 3, no. 4, pp. 887–896, 2008. (Cited on page 12)

[46] M. A. Llama, R. Kelly, and V. Santibañez, "Stable computed-torque control of robot manipulators via fuzzy self-tuning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 30, no. 1, pp. 143–150, 2000. (Cited on page 12)

[47] T. D. C. Thanh and K. K. Ahn, "Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network," *Mechatronics*, vol. 16, no. 9, pp. 577–587, 2006. (Cited on page 12)

[48] J. L. Meza, V. Santibáñez, R. Soto, and M. A. Llama, "Fuzzy self-tuning PID semiglobal regulator for robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 6, pp. 2709–2717, 2012. (Cited on page 12)

[49] Y. Chen, G. Ma, S. Lin, and J. Gao, "Adaptive fuzzy computed-torque control for robot manipulator with uncertain dynamics," *International Journal of Advanced Robotic Systems*, vol. 9, 2012. (Cited on page 12)

[50] H. B. Kazemian, "Intelligent Fuzzy PID Controller," *Foundations of Generic Optimization: Volume 2: Applications of Fuzzy Control, Genetic Algorithms and Neural Networks*, pp. 241–260, 2008. (Cited on page 12)

[51] A. Aouaichia, K. Kara, and A. Ghoul, "An optimized fuzzy computed torque control for the robot manipulator PUMA 560," in *2023 International Conference on Advances in Electronics, Control and Communication Systems, ICAECCS 2023*, Blida 2023. (Cited on page 12)

[52] M. S. Ahmed, A. H. Mary, and H. H. Jasim, "Model and chattering free adaptive fuzzy smc for robotic manipulator systems," *Texas Journal of Engineering and Technology*, vol. 8, pp. 33–43, 2022. (Cited on page 12)

[53] L. Jin, S. Li, J. Yu, and J. He, "Robot manipulator control using neural networks: A survey," *Neurocomputing*, vol. 285, pp. 23–34, 2018. (Cited on pages 12 and 22)

[54] Z. H. Jiang, T. Ishida, and M. Sunawada, "Neural network aided dynamic parameter identification of robot manipulators," in *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, 2006, pp. 3298–3303. (Cited on page 12)

[55] W. He, A. O. David, Z. Yin, and C. Sun, "Neural Network Control of a Robotic Manipulator With Input Deadzone and Output Constraint," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 6, pp. 759–770, 2016. (Cited on page 12)

[56] W. Yu and J. Rosen, "Neural PID control of robot manipulators with application to an upper limb exoskeleton," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 673–684, 2013. (Cited on page 12)

[57] L. Wang, T. Chai, and L. Zhai, "Neural-network-based terminal sliding-mode control of robotic manipulators including actuator dynamics," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 9, pp. 3296–3304, 2009. (Cited on page 12)

[58] N. Kapoor and J. Ohri, "Sliding Mode Control (SMC) of Robot Manipulator via Intelligent Controllers," *Journal of The Institution of Engineers (India): Series B*, vol. 98, no. 1, pp. 83–98, 2017. (Cited on page 12)

[59] L. Peng and P. Y. Woo, "Neural-Fuzzy Control System for Robotic Manipulators," *IEEE Control Systems*, vol. 22, no. 1, pp. 53–63, 2002. (Cited on page 12)

[60] C. Sun, H. Gao, W. He, and Y. Yu, "Fuzzy neural network control of a flexible robotic manipulator using assumed mode method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5214–5227, 2018. (Cited on page 12)

[61] B. Wei, "Adaptive control design and stability analysis of robotic manipulators," *Actuators*, vol. 7, no. 4, 2018. (Cited on page 12)

[62] J. Baek, M. Jin, and S. Han, "A New Adaptive Sliding-Mode Control Scheme for Application to Robot Manipulators," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 6, pp. 3628–3637, 2016. (Cited on page 13)

[63] T. N. Truong, H. J. Kang, and T. D. Le, "Adaptive Neural Sliding Mode Control for 3-DOF Planar Parallel Manipulators," in *ACM International Conference Proceeding Series*, 2019. (Cited on page 13)

[64] J. Lee, P. H. Chang, B. Yu, and M. Jin, "An adaptive PID control for robot manipulators under substantial payload variations," *IEEE Access*, vol. 8, pp. 162 261–162 270, 2020. (Cited on page 13)

[65] N. N. Son, H. P. H. Anh, and T. D. Chau, "Adaptive neural model optimized by modified differential evolution for identifying 5-DOF robot manipulator dynamic system," *Soft Computing*, vol. 22, no. 3, pp. 979–988, 2018. (Cited on page 13)

[66] S. Fateh and M. M. Fateh, "Adaptive Fuzzy Control of Robot Manipulators with Asymptotic Tracking Performance," *Journal of Control, Automation and Electrical Systems*, vol. 31, no. 1, pp. 52–61, 2020. (Cited on page 13)

[67] J. P. Kolhe, M. Shaheed, T. S. Chandar, and S. E. Talole, "Robust control of robot manipulators based on uncertainty and disturbance estimation," *International Journal of Robust and Nonlinear Control*, vol. 23, no. 1, pp. 104–122, 2013. (Cited on page 13)

[68] 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, 2011. (Cited on page 13)

[69] S. E. Shafiei and M. R. Soltanpour, "Robust neural network control of electrically driven robot manipulator using backstepping approach,"

International Journal of Advanced Robotic Systems, vol. 6, no. 4, pp. 285–292, 2009. (Cited on page 13)

[70] R. J. Wai and P. C. Chen, "Robust neural-fuzzy-network control for robot manipulator including actuator dynamics," *IEEE Transactions on Industrial Electronics*, vol. 53, no. 4, pp. 1328–1349, 2006. (Cited on page 13)

[71] H. F. Ho, Y. K. Wong, and A. B. Rad, "Robust fuzzy tracking control for robotic manipulators," *Simulation Modelling Practice and Theory*, vol. 15, no. 7, pp. 801–816, 2007. (Cited on page 13)

[72] R. Burkan and A. Mutlu, "Robust control of robot manipulators with an adaptive fuzzy unmodelled parameter estimation law," *Robotica*, vol. 40, no. 7, pp. 2365–2380, 2022. (Cited on page 13)

[73] G. Bruni, S. Cordiner, V. Mulone, V. Rocco, and F. Spagnolo, "A study on the energy management in domestic micro-grids based on model predictive control strategies q," *Energy Conversion and Management*, vol. 102, pp. 50–58, 2015. (Cited on page 13)

[74] Y. Huang, H. Wang, A. Khajepour, H. He, and J. Ji, "Model predictive control power management strategies for HEVs: A review," *Journal of Power Sources*, vol. 341, pp. 91–106, 2017. (Cited on page 13)

[75] Y. Ding, A. Pandala, C. Li, Y. H. Shin, and H. W. Park, "Representation-Free Model Predictive Control for Dynamic Motions in Quadrupeds," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1154–1171, 2021. (Cited on page 13)

[76] Y. Ding, L. Wang, Y. Li, and D. Li, "Model predictive control and its application in agriculture: A review," *Computers and Electronics in Agriculture*, vol. 151, pp. 104–117, 2018. (Cited on page 13)

[77] A. Afram, F. Janabi-Sharifi, A. S. Fung, and K. Raahemifar, "Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system," *Energy and Buildings*, vol. 141, pp. 96–113, 2017. (Cited on page 13)

[78] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000. (Cited on page 13)

[79] H. Li, Y. Shi, and W. Yan, "Distributed receding horizon control of constrained nonlinear vehicle formations with guaranteed γ-gain stability," *Automatica*, vol. 68, pp. 148–154, 2016. (Cited on pages 13 and 55)

[80] J. Richalet, A. Rault, J. L. Testud, and J. Papon, "Model predictive heuristic control. Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978. (Cited on page 13)

[81] B. L. Cutler, Charles R and Ramaker, "Dynamic matrix control?? A computer control algorithm," in *joint automatic control conference*, 1980, p. 72. (Cited on page 13)

[82] E. C. Kerrigan and J. M. Maciejowski, "Designing model predictive controllers with prioritised constraints and objectives," in *2002 IEEE International Symposium on Computer Aided Control System Design, CACSD 2002 - Proceedings*, 2002, pp. 33–38. (Cited on page 13)

[83] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. Springer International Publishing, 2017, pp. 45–69. (Cited on page 13)

[84] E. Camacho and C. Bordons, *Model Predictive Control - Second edition*. Springer, 2007. (Cited on page 13)

[85] K. Belda, J. Böhm, and M. Valášek, "State-Space Generalized Predictive Control for Redundant Parallel Robots," *Mechanics Based Design of Structures and Machines*, vol. 31, no. 3, pp. 413–432, 2003. (Cited on page 13)

[86] J. S. Terry, L. Rupert, and M. D. Killpack, "Comparison of linearized dynamic robot manipulator models for model predictive control," in *IEEE-RAS International Conference on Humanoid Robots*, 2017, pp. 205–212. (Cited on page 14)

[87] F. A. Lara-Molina, J. M. Rosário, D. Dumur, and P. Wenger, "Robust generalized predictive control of the Orthoglide robot," *Industrial Robot*, vol. 41, no. 3, pp. 275–285, 2014. (Cited on page 14)

[88] R. Grandia, A. J. Taylor, A. Singletary, M. Hutter, and A. D. Ames, "Nonlinear Model Predictive Control of Robotic Systems with Control Lyapunov Functions," *Robotics: Science and Systems*, 2020. (Cited on page 14)

[89] S. Hu, E. Babaians, M. Karimi, and E. Steinbach, "NMPC-MP: Real-time Nonlinear Model Predictive Control for Safe Motion Planning in Manipulator Teleoperation," in *IEEE International Conference on Intelligent Robots and Systems*, 2021, pp. 8309–8316. (Cited on page 14)

[90] E. Kang, H. Qiao, J. Gao, and W. Yang, "Neural network-based model predictive tracking control of an uncertain robotic manipulator with input constraints," *ISA Transactions*, vol. 109, pp. 89–101, 2021. (Cited on page 14)

[91] C. Stiti, K. Kara, M. Benrabah, and A. Aouaichia, "Neural Network Model Predictive Control Based on PSO Approach: Applied to DC Motor," in *2023 2nd International Conference on Electronics, Energy and Measurement (IC2EM)*, vol. 1, 2023, pp. 1–6. (Cited on page 14)

[92] T. A. Johansen and A. Grancharova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Transactions on Automatic Control*, vol. 48, no. 5, pp. 810–815, 2003. (Cited on page 14)

[93] P. Hyatt and M. D. Killpack, "Real-Time Nonlinear Model Predictive Control of Robots Using a Graphics Processing Unit," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1468–1475, 2020. (Cited on page 14)

[94] J. P. Coelho, P. B. De Moura Oliveira, and J. B. Cunha, "Greenhouse air temperature predictive control using the particle swarm optimisation algorithm," *Computers and Electronics in Agriculture*, vol. 49, no. 3, pp. 330–344, 2005. (Cited on page 14)

[95] H. Zhixiang, C. Hui, and L. Heqing, "Neural networks predictive control using AEPSO," in *Proceedings of the 27th Chinese Control Conference, CCC*, 2008, pp. 180–183. (Cited on page 14)

[96] J. Mercieca and S. G. Fabri, "Particle swarm optimization for nonlinear model predictive control," *Proc. ADVCOMP*, pp. 88–93, 2011. (Cited on page 14)

[97] M. L. Hadjili, K. Kara, O. A. Sahed, and J. Bouyanzar, "Fuzzy predictive control using particle swarm optimization: Application to SCARA robot," *Applied Mechanics and Materials*, vol. 527, pp. 230–236, 2014. (Cited on page 14)

[98] M. Benrabah, K. Kara, O. AitSahed, and M. L. Hadjili, "Constrained Nonlinear Predictive Control Using Neural Networks and Teaching–Learning-Based Optimization," *Journal of Control, Automation and Electrical Systems*, vol. 32, no. 5, pp. 1228–1243, 2021. (Cited on page 14)

[99] M. I. Ullah, S. A. Ajwad, M. Irfan, and J. Iqbal, "MPC and H-Infinity Based Feedback Control of Non-Linear Robotic Manipulator," in *Proceedings - 14th International Conference on Frontiers of Information Technology, FIT 2016*, 2017, pp. 136–141. (Cited on page 14)

[100] Y. Chen, X. Luo, B. Han, Q. Luo, and L. Qiao, "Model Predictive Control with Integral Compensation for Motion Control of Robot Manipulator in Joint and Task Spaces," *IEEE Access*, vol. 8, pp. 107 063–107 075, 2020. (Cited on page 14)

[101] M. Rubagotti, D. M. Raimondo, A. Ferrara, and L. Magni, "Robust model predictive control with integral sliding mode in continuous-time sampled-data nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 56, no. 3, pp. 556–570, 2011. (Cited on pages 14 and 55)

[102] G. P. Incremona, A. Ferrara, and L. Magni, "MPC for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, 2017. (Cited on page 14)

[103] R. Galvan-Guerra, G. P. Incremona, L. Fridman, and A. Ferrara, "Robust Multi-Model Predictive Control via Integral Sliding Modes," *IEEE Control Systems Letters*, vol. 6, pp. 2623–2628, 2022. (Cited on page 14)

[104] A. M. Jasour and M. Farrokhi, "Adaptive neuro-predictive control for redundant robot manipulators in presence of static and dynamic obstacles: A Lyapunov-based approach," *International Journal of Adaptive Control and Signal Processing*, vol. 28, no. 3-5, pp. 386–411, 2014. (Cited on page 14)

[105] E. Kang, H. Qiao, Z. Chen, and J. Gao, "Tracking of Uncertain Robotic Manipulators Using Event-Triggered Model Predictive Control With Learning Terminal Cost," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2801–2815, 2022. (Cited on page 14)

[106] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020. (Cited on page 14)

[107] L. Dai, Y. Yu, D. H. Zhai, T. Huang, and Y. Xia, "Robust Model Predictive Tracking Control for Robot Manipulators with Disturbances," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 5, pp. 4288–4297, 2021. (Cited on page 14)

[108] V. C. Nguyen, H. L. Thi, and T. L. Nguyen, "A Lyapunov-based model predictive control strategy with a disturbances compensation mechanism for dual-arm manipulators," *European Journal of Control*, vol. 75, 2024. (Cited on page 14)

[109] Y. Wang, M. Leibold, J. Lee, W. Ye, J. Xie, and M. Buss, "Incremental Model Predictive Control Exploiting Time-Delay Estimation for a Robot Manipulator," *IEEE Transactions on Control Systems Technology*, vol. 30, no. 6, pp. 2285–2300, 2022. (Cited on page 14)

[110] K. Seel, E. I. Grotli, S. Moe, J. T. Gravdahl, and K. Y. Pettersen, "Neural Network-Based Model Predictive Control with Input-to-State Stability," in *Proceedings of the American Control Conference*, vol. 2021-May, 2021, pp. 3556–3563. (Cited on pages 14 and 55)

[111] M. V. Minniti, R. Grandia, F. Farshidian, and M. Hutter, "Adaptive CLF-MPC with Application to Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 565–572, 2022. (Cited on page 14)

[112] O. Djaneye-Boundjou, X. Xu, and R. Ordonez, "Automated particle swarm optimization based PID tuning for control of robotic arm," in *Proceedings of the IEEE National Aerospace Electronics Conference, NAECON*, vol. 0, 2016, pp. 164–169. (Cited on page 14)

[113] S. A. Mazhari and S. Kumar, "PUMA 560 Optimal Trajectory Control using Genetic Algorithm , Simulated Annealing and Generalized Pattern Search Techniques," *International Journal of Electrical, Computer, and Systems Engineering*, vol. 2, no. 1, pp. 71–80, 2008. (Cited on page 14)

[114] H. A. Akkar and S. Q. G. Haddad, "Design Stable Controller for PUMA 560 Robot with PID and Sliding Mode Controller Based on PSO Algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 6, pp. 487–499, 2020. (Cited on page 14)

[115] P. Ji, C. Li, and F. Ma, "Sliding Mode Control of Manipulator Based on Improved Reaching Law and Sliding Surface," *Mathematics*, vol. 10, no. 11, 2022. (Cited on page 14)

[116] V. Mehra and D. Shah, "Parameter Optimization of Reaching Law Based Sliding Mode Control by Computational Intelligence Techniques," *Communications in Computer and Information Science*, vol. 1374, pp. 88–100, 2021. (Cited on page 14)

[117] H. Sarimveis and G. Bafas, "Fuzzy model predictive control of non-linear processes using genetic algorithms," *Fuzzy Sets and Systems*, vol. 139, no. 1, pp. 59–80, 2003. (Cited on page 15)

[118] Y. Li, J. Shen, K. Y. Lee, and X. Liu, "Offset-free fuzzy model predictive control of a boiler-turbine system based on genetic algorithm," *Simulation Modelling Practice and Theory*, vol. 26, pp. 77–95, 2012. (Cited on page 15)

[119] A. Zimmer, A. Schmidt, A. Ostfeld, and B. Minsker, "Evolutionary algorithm enhancement for model predictive control and real-time decision support," *Environmental Modelling and Software*, vol. 69, pp. 330–341, 2015. (Cited on page 15)

[120] O. A. Sahed, K. Kara, and A. Benyoucef, "Artificial bee colony-based predictive control for non-linear systems," *Transactions of the Institute of Measurement and Control*, vol. 37, no. 6, pp. 780–792, 2015. (Cited on page 15)

[121] O. A. Sahed, K. Kara, A. Benyoucef, and M. L. Hadjili, "An efficient artificial bee colony algorithm with application to nonlinear predictive control," *International Journal of General Systems*, vol. 45, no. 4, pp. 393–417, 2016. (Cited on page 15)

[122] J.-Q. Han, "Nonlinear design methods for control systems," *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 1531–1536, 1999. (Cited on page 15)

[123] J. Han, "From PID to active disturbance rejection control," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 3, pp. 900–906, 2009. (Cited on pages 15 and 54)

[124] Q. Zheng, Z. Chen, and Z. Gao, "A practical approach to disturbance decoupling control," *Control Engineering Practice*, vol. 17, no. 9, pp. 1016–1025, 2009. (Cited on page 15)

[125] Y. Huang and W. Xue, "Active disturbance rejection control: Methodology and theoretical analysis," *ISA Transactions*, vol. 53, no. 4, pp. 963–976, 2014. (Cited on page 15)

[126] Q. Zheng and Z. Gao, "Active disturbance rejection control: some recent experimental and industrial case studies," *Control Theory and Technology*, vol. 16, no. 4, pp. 301–313, 2018. (Cited on page 15)

[127] M. Ali and C. K. Alexander, "Trajectory tracking control for a robotic manipulator using nonlinear active disturbance rejection control," in *ASME 2017 Dynamic Systems and Control Conference, DSCC 2017*, vol. 2, 2017. (Cited on page 15)

[128] R. Fareh, M. Al-Shabi, M. Bettayeb, and J. Ghommam, "Robust Active Disturbance Rejection Control for Flexible Link Manipulator," *Robotica*, vol. 38, no. 1, pp. 118–135, 2020. (Cited on page 15)

[129] R. Fareh, S. Khadraoui, M. Y. Abdallah, M. Baziyad, and M. Bettayeb, "Active disturbance rejection control for robotic systems: A review," *Mechatronics*, vol. 80, 2021. (Cited on page 15)

[130] Y. Cheng, L. Dai, A. Li, Y. Yuan, and Z. Chen, "Active Disturbance Rejection Generalized Predictive Control of a Quadrotor UAV via Quantitative Feedback Theory," *IEEE Access*, vol. 10, pp. 37 912–37 923, 2022. (Cited on page 15)

[131] C. Jing, H. Xu, and X. Niu, "Adaptive sliding mode disturbance rejection control with prescribed performance for robotic manipulators," *ISA Transactions*, vol. 91, pp. 41–51, 2019. (Cited on pages 15 and 16)

[132] T. Ren, Y. Dong, D. Wu, G. Wang, and K. Chen, "Joint Torque Control of a Collaborative Robot Based on Active Disturbance Rejection With the Consideration of Actuator Delay," in *International Mechanical Engineering Congress and Exposition*, 2017. (Cited on page 15)

[133] M. Abdallah and R. Fareh, "Fractional order active disturbance rejection control for trajectory tracking for 4-DOF serial link manipulator," *International Journal of Modelling, Identification and Control*, vol. 36, no. 1, pp. 57–65, 2020. (Cited on page 15)

[134] A. Aboelhassan, A. M. Diab, M. Galea, and S. Bozhko, "Investigating Electrical Drive Performance Employing Model Predictive Control and Active Disturbance Rejection Control Algorithms," in *23rd International Conference on Electrical Machines and Systems, ICEMS 2020*, 2020, pp. 1379–1384. (Cited on page 15)

[135] S. A. Suhail, M. A. Bazaz, and S. Hussain, "MPC based active disturbance rejection control for automated steering control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 235, no. 12, pp. 3199–3206, 2021. (Cited on page 15)

[136] J. Yang, H. Peng, W. Zhou, J. Zhang, and Z. Wu, "A modular approach for dynamic modeling of multisegment continuum robots," *Mechanism and Machine Theory*, vol. 165, p. 104429, 2021. (Cited on page 15)

[137] B. Zhan, L. Zhang, Y. Liu, and J. Gao, "Model predictive and compensated ADRC for permanent magnet synchronous linear motors," *ISA Transactions*, 2022. (Cited on page 15)

[138] H. Yang, M. Guo, Y. Xia, and Z. Sun, "Dual closed-loop tracking control for wheeled mobile robots via active disturbance rejection control and model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 1, pp. 80–99, 2020. (Cited on page 15)

[139] D. Ma, Y. Xia, T. Li, and K. Chang, "Active disturbance rejection and predictive control strategy for a quadrotor helicopter," *IET Control Theory and Applications*, vol. 10, no. 17, pp. 2213–2222, 2016. (Cited on page 15)

[140] Y. Cheng, Z. Chen, M. Sun, and Q. Sun, "Active disturbance rejection generalized predictive control for a high purity distillation column process with time delay," *Canadian Journal of Chemical Engineering*, vol. 97, no. 11, pp. 2941–2951, 2019. (Cited on page 15)

[141] J. Gao, X. Liang, Y. Chen, L. Zhang, and S. Jia, "Hierarchical image-based visual serving of underwater vehicle manipulator systems based on model predictive control and active disturbance rejection control," *Ocean Engineering*, vol. 229, 2021. (Cited on page 15)

[142] J. Arcos-Legarda and Á. Gutiérrez, "Robust Model Predictive Control Based on Active Disturbance Rejection Control for a Robotic Autonomous Underwater Vehicle," *Journal of Marine Science and Engineering*, vol. 11, no. 5, 2023. (Cited on page 15)

[143] C. P. Bechlioulis and G. A. Rovithakis, "Prescribed performance adaptive control of SISO feedback linearizable systems with disturbances," in *2008 Mediterranean Conference on Control and Automation - Conference Proceedings, MED'08*, 2008, pp. 1035–1040. (Cited on pages 15 and 72)

[144] C. P. Bechlioulis and G. A. Rovithakis, "Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance," *IEEE Transactions on Automatic Control*, vol. 53, no. 9, pp. 2090–2099, 2008. (Cited on pages 15 and 72)

[145] T. Berger, A. Ilchmann, and E. P. Ryan, "Funnel control–a survey," *arXiv preprint arXiv:2310.03449*, 2023. (Cited on page 16)

[146] J. Wang, X. Zhu, Y. Pei, D. Wang, Q. Shen, and Y. Niu, "Adaptive prescribed performance tuning for model predictive control," in *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2021, pp. 1523–1529. (Cited on page 16)

[147] P. Marantos, A. Eqtami, C. P. Bechlioulis, and K. J. Kyriakopoulos, "A prescribed performance robust nonlinear model predictive control framework," in *2014 European Control Conference (ECC)*. IEEE, 2014, pp. 2182–2187. (Cited on page 16)

[148] S. Stihi, A. Aouaichia, O. Gad, R. Fareh, S. Khadraoui, M. Bettayeb, and K. Kara, "Funnel Neural Network Model Predictive Control for a 4-DoF Robot Manipulator," in *The International Conference on Mechatronics, Industry 4.0, IoT and their Applications - 2024 Advances in Science and Engineering Technology (ASET) International Conferences*, 2024. (Cited on page 16)

[149] S. Stihi, M. Baziyad, O. Gad, R. Fareh, S. Khadraoui, and M. Bettayeb, "Tracking funnel control applied on a 4-dof robot manipulator: A comparative study," in *2023 Advances in Science and Engineering Technology International Conferences (ASET)*. IEEE, 2023, pp. 1–6. (Cited on page 16)

[150] A. T. Vo, T. N. Truong, and H. J. Kang, "An Adaptive Prescribed Performance Tracking Motion Control Methodology for Robotic Manipulators with Global Finite-Time Stability," *Sensors*, vol. 22, no. 20, 2022. (Cited on page 16)

[151] A. K. Kostarigka, Z. Doulgeri, and G. A. Rovithakis, "Prescribed performance tracking for flexible joint robots with unknown dynamics and variable elasticity," *Automatica*, vol. 49, no. 5, pp. 1137–1147, 2013. (Cited on pages 16 and 71)

[152] T. Mei, Y. Yang, J. Chen, G. Zhang, Z. Jiao, L. Gao, X. Ren, and Q. Li, "Simulation Research on Motion Trajectory of PUMA 560 Manipulator Based on MATLAB," in *Proceedings of the 31st Chinese Control and Decision Conference, CCDC 2019*, 2019, pp. 4857–4862. (Cited on page 20)

[153] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012. (Cited on page 20)

[154] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. (Cited on page 22)

[155] A. Afram, F. Janabi-Sharifi, A. S. Fung, and K. Raahemifar, "Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system," *Energy and Buildings*, vol. 141, pp. 96–113, 2017. (Cited on page 22)

[156] J. Li, J. Wang, S. Wang, W. Qi, L. Zhang, Y. Hu, and H. Su, "Neural Approximation-based Model Predictive Tracking Control of Non-holonomic Wheel-legged Robots," *International Journal of Control, Automation and Systems*, vol. 19, no. 1, pp. 372–381, 2021. (Cited on page 22)

[157] Z. Michalewicz, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996. (Cited on page 23)

[158] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, no. 11-12, pp. 1245–1287, 2002. (Cited on page 23)

[159] C. A. C. Coello, "Constraint-handling techniques used with evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. New York, NY, USA: ACM, jul 2022, pp. 1310–1333. (Cited on page 23)

[160] J. A. Rossiter, *Model-based predictive control : a practical approach*. CRC Press, 2017. (Cited on page 23)

[161] B. Kouvaritakis and M. Cannon, *Non-linear Predictive Control: theory and practice*. Iet, 2001. (Cited on page 23)

[162] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems," *Applied Intelligence*, vol. 51, no. 3, pp. 1531–1551, 2021. (Cited on pages 24 and 32)

[163] F. Lin, *Robust control design: an optimal control approach*. John Wiley & Sons, 2007. (Cited on page 29)

[164] H. OUNNAS and S. SEGHIR ABDELLAH, "Conception, réalisation et commande optimisée d'un bras manipulateur scara," Ph.D. dissertation, Université de Blida 1, 2019. (Cited on page 41)

[165] S. Stihi, M. Baziyad, O. Gad, R. Fareh, S. Khadraoui, and M. Bettayeb, "Tracking Funnel Control Applied on a 4-DOF Robot Manipulator: A Comparative Study," in *2023 Advances in Science and Engineering Technology International Conferences, ASET 2023*, 2023. (Cited on page 49)

[166] J. Bongard, J. Berberich, J. Koehler, and F. Allgower, "Robust stability analysis of a simple data-driven model predictive control approach," *IEEE Transactions on Automatic Control*, 2022. (Cited on page 55)

[167] J. Luo, Y. Li, P. Liu, S. Ye, R. Feng, and J. Wang, "Lyapunov Based Nonlinear Model Predictive Control of Wind Power Generation System With External Disturbances," *IEEE Access*, vol. 12, pp. 5103–5116, 2024. (Cited on page 55)

[168] K. Patan, "Neural Network-Based Model Predictive Control: Fault Tolerance and Stability," *IEEE Transactions on Control Systems Technology*, vol. 23, no. 3, pp. 1147–1155, 2015. (Cited on page 55)

[169] A. Campeau-Lecours, H. Lamontagne, S. Latour, P. Fauteux, V. Maheu, F. Boucher, C. Deguire, and L. J. C. L'Ecuyer, "Kinova Modular Robot Arms for Service Robotics Applications," *Rapid Automation: Concepts, Methodologies, Tools, and Applications*, pp. 693–719, 2019. (Cited on page 59)

[170] X. Bu, "Prescribed performance control approaches, applications and challenges: A comprehensive survey," *Asian Journal of Control*, vol. 25, no. 1, pp. 241–261, 2023. (Cited on page 71)

[171] B. Armstrong, O. Khatib, and J. Burdick, "Explicit Dynamic Model and Inertial Parameters of the Puma 560 Arm." in *IEEE International Conference on Robotics and Automation (ICRA)*, 1986, pp. 510–518. (Cited on pages 77 and 78)

[172] P. I. Corke and B. Armstrong-Helouvry, "Search for consensus among model parameters reported for the PUMA 560 robot," in *Proceedings - IEEE International Conference on Robotics and Automation*, no. pt 2, 1994, pp. 1608–1613. (Cited on page 78)

[173] B. Armstrong-Hélouvry, "A Meta-Study of PUMA 560 Dynamics: A Critical Appraisal of Literature Data," *Robotica*, vol. 13, no. 3, pp. 253–258, 1995. (Cited on page 78)