Neural Network Based Sliding Mode Control for Robotic Manipulator

Ingredy Gabriela Gomes Carmo

A Thesis Submitted to the Faculty of Graduate Studies In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Graduate Program in Earth and Space Science

York University

Toronto, Ontario

May 2024

Abstract

Robotic manipulators are used in many applications. However, robotic arms are complex systems due to external disturbances, perturbations, and their coupled non-linear dynamics. This thesis aims to propose a robust control strategy for autonomous robotic manipulation. First, the trajectory tracking problem was introduced and an approach to overcome this issue using a sliding mode controller combined with a neural network is proposed. Then, the proposed approach is compared to classical and modern control methods including controllers from the literature to demonstrate the performance of the proposed controller. The proposed controller was then integrated with a grasp detection algorithm for an autonomous manipulation application. Simulations and hardware experiments were conducted to validate the performance of the proposed method.

In memory of my godfather.

Acknowledgments

First, I would like to acknowledge my supervisor Professor Jinjun Shan whose wisdom, guidance, and support were of great importance to my growth. I am thankful for being part of the SDCNLab and all the opportunities I was able to experience throughout my Master's.

I would like to thank my current and former lab mates for the insightful discussions and projects shared, Dr. Hassan Alkomy, Dr. Xiaoyu Zhu, Dr. Junjie Kang, Dr. Marc Savoie, Dr. Huarong Zhao, Hunter Schofield, Hao Wang, Mingfeng Yuan, Yida Zang, Amal, and Yibo Liu. My special thanks go to Dr. Hassan Alkomy who is not only a close friend but was also my mentor and supporter throughout these years, you have my gratitude.

I would also like to thank Quanser and Mitacs for supporting my internship. I had a great time working with the engineers on site. This opportunity enhanced my skills and knowledge in research and engineering and for that I am grateful.

Last but foremost, I would like to express my forever gratitude to my family who are my greatest supporters. To my mother Wilma without whom I would not be able to be here today. I would also like to express my gratitude to my grandmother Francelina, my godmother Marilsa, my aunt Marli, my sister Julia, my father Julio, and my cousins, aunts, and uncles for their support towards this degree. Thank you for your love and patience.

Table of Contents

List of Tables

List of Figures

List of Acronyms

DOF	Degree of freedom
PID	Proportional-Integral-Derivative
SDCNLab	Spacecraft Dynamics, Control and Navigation Laboratory
RBF	Radial basis funcion
NN	Neural Network
RBFNN	Radial basis funcion neural network
GGCNN	Generative grasping convolutional neural network
USB	Universal serial bus
SMC	Sliding mode control
RGBD	Red, green, blue, depth
RL	Reinforcement learning
VRL	Variable learning rate

1 Introduction

1.1 Overview and Motivation

Robotic manipulators have been extensively used in a variety of applications throughout the years, especially due to human limitations to work in some kinds of environments. Investigating their applications and developing methods to make them work safely and autonomously in remote and controlled sites is an important work to be done.

The applications of autonomous robots, such as manipulators, are not limited to performing repetitive tasks. They can collaborate with other robots for efficient warehouse operations [4]. They can also work in environments that are considered hazardous to humans, like space applications such as on-orbit systems (OOS), Canadarm2 [5], and space debris [6].

Among many applications of robotic arms, payload manipulation [7] is a common and extensively studied one. When it comes to performing an autonomous manipulation task, the trajectory tracking problem is often addressed since we have to consider a variety of constraints of the robot such as uncertainties, environmental noises and disturbances, unmodeled dynamics, and robot behavior while manipulating different kinds of objects [8, 9]. Considering these applications, accurate control is crucial for robots to complete their mission successfully.

Considering the task of autonomously and accurately manipulating a given object in the workspace, many procedures need to be fulfilled to achieve a successful performance like object detection and

recognition, pose assessment, or estimation of the object of interest relative to the robot's coordinate system, path planning and ultimately the control scheme design. This thesis focuses on the trajectory tracking control problem and implements the proposed controller, with a grasping detection technique, for autonomous manipulation tasks.

An autonomous robotic manipulation can be broken down into two main tasks robotic control and sensing. Considering the control problem for trajectory tracking of a manipulator with unknown dynamics [10], the main problem with current control methods is that they mostly rely on prior complete or partial knowledge of the robot's model [11, 12]. It is highly difficult to demonstrate an accurate dynamics model for robotic arms such as their time-varying dynamic parameters, nonlinearities, uncertainties, and coupled dynamics. Moreover, these manipulators are subjected to tricky working conditions, external disturbances like noises and vibrations, and payload variation [8, 9]. On the whole, the control design for such equipment is a very challenging task. Therefore, accurately estimating these parameters is of utmost importance for good controller performance.

Given that in recent years, neural networks and their applications in computer vision and intelligent control methods of robots have been deeply studied, this work aims to contribute to the existing literature on robust control applied to robot manipulators by proposing a trajectory-tracking controller based on a radial basis function neural network (RBFNN) with online parameters update for an n-DOF (Degrees-of-Freedom) robot manipulator to be used for payload manipulation tasks.

1.2 Objectives and Contributions

The goal of this thesis is to design a learning-based controller for payload manipulation and implement it in autonomous manipulation tasks. This thesis aims to fulfill the following research objectives:

(a) Design and implement a radial basis function neural network with a variable learning rate to estimate the system's unknown dynamics.

(b) Design and implement a sliding mode controller (SMC) that is robust against disturbances for tracking a desired trajectory.

(c) Integrate the RBFNN with the sliding mode controller as one pipeline to realize trajectory tracking of a 4-DOF manipulator with unknown dynamics.

(d) Validate the proposed controller experimentally and compare it to other control alternatives including the literature.

(e) Implement the proposed controller with a grasping detection technique for autonomous manipulation.

The main contributions of this thesis are as follows:

(i) Design and implementation of SMC with variable learning rate RBFNN for a robot manipulator with unknown dynamics that is robust to payload variation. The control performance is enhanced compared to other controller alternatives including the literature.

(ii) The proposed controller can be smoothly integrated with grasping detection techniques to complete autonomous tasks.

1.3 Experimental Setup

The experiments presented in this thesis were conducted at the Spacecraft Dynamics, Control, and Navigation Laboratory (SDCNLab). They were evaluated on either one or both: simulation environment and hardware. The robotic manipulator used to conduct the experiments was the QArm, shown in Fig. 1.2a, which is a 4-DOF robotic manipulator manufactured by Quanser, equipped with a tendon-based two-stage gripper and a Red Green Blue-Depth (RGBD) camera.

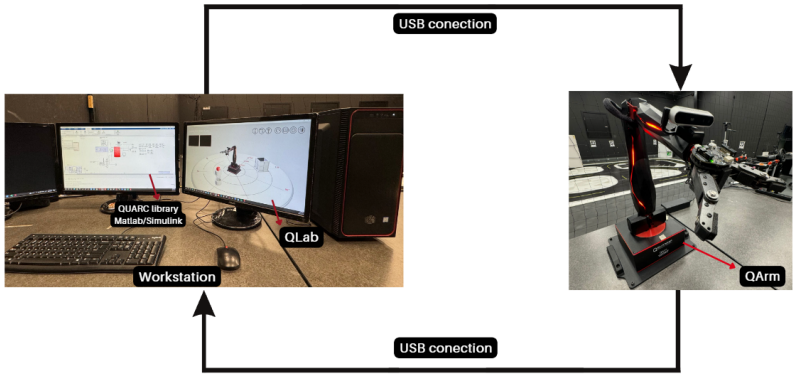
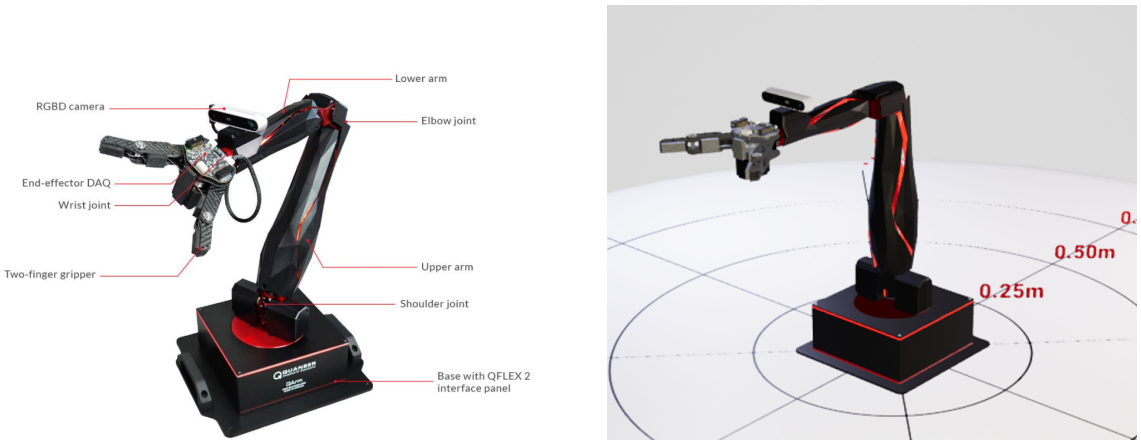


Figure 1.1: QArm manipulator [2]

For the hardware setup, the robotic arm is configured with a QFLEX 2 USB interface panel which allows it to be connected to a workstation supplied with QUARC library, MatLab® and Simulink® via USB. The control commands and the measurement feedback were sent and received via USB cables. The joints use servo motors with magnetic encoders, velocity measurement, current sensors, and temperature sensors. The base, shoulder, and elbow use the XM540-W270-R Dynamixel servomotor [13], and the wrist uses the XM430-W350-R Dynamixel servomotor [14]. The schematic for the experimental setup can be seen in Fig. 1.1. To acquire images from the robot workspace, the following device located just above the QArm gripper was used:

- Intel® RealSense™ R415 RGBD camera that provides up to 1920×1080 RGB resolution and a depth sensing range from 0.3 to 4.0 m.

It is also good to mention that for safety reasons, some of the trajectories were first implemented on a QArm-simulated environment, the Interactive QLabs provided by Quanser. The virtual duplicate of the QArm system that can render the actual hardware behavior can be seen in Fig. 1.2b.



(a) QArm manipulator. [2]

(b) QArm virtual environment. [15]

Figure 1.2: Actual and virtual QArm.

1.4 Thesis Outline

In addition to this Introduction, the contents of this thesis are arranged as follows:

Chapter 2: System Modeling - The kinematics and dynamics models of the 4-DOF robotic manipulator used in this thesis are derived.

Chapter 3: Control Design - A variable learning rate RBFNN-based sliding mode control is designed and the stability is proved. The proposed controller is compared to different control methods to demonstrate its performance. Then, the proposed controller is seamlessly integrated with a grasping detection technique to complete an autonomous manipulation task.

Chapter 4: Conclusions and Future Work - In this chapter, the work proposed in this thesis is summarized and a discussion about extending it for future developments is presented.

2 System Modeling

This chapter describes the fundamental concepts to understand how robotic manipulators are modeled and the specifications of the hardware used. First, the model parameters are shown. Then, the kinematics and dynamics model equations are derived according to [1, 16, 17].

2.1 Robot Specifications

In this section, the specifications of the hardware used for the experiments are shown. An image of the Quanser QArm can be seen in Fig. 1.2a. In Fig. 2.2 the reachable workspace of the robot is represented according to [1].

Table 2.1 shows the robot's position, speed, acceleration, and torque ranges for its four joints [1].

Table 2.1: Joint specifications.

	Base (θ_1)	Shoulder (θ_2)	Elbow (θ_3)	Wrist (θ_4)	units
Joint Range	$\pm 17\pi/18$	$\pm 17\pi/36$	$-19\pi/36$ to $+5\pi/12$	$\pm 8\pi/9$	rads
Speed	$\pm\pi/2$	$\pm\pi/2$	$\pm\pi/2$	$\pm\pi/2$	rad/s
Acceleration	$\pm\pi/3$	$\pm\pi/3$	$\pm\pi/3$	$\pm\pi/3$	rad/s^2
Torque	2.65	5.3	2.65	1.0	Nm

According to the frame assignments shown in Fig. 2.4 the physical parameters of this manipulator are:

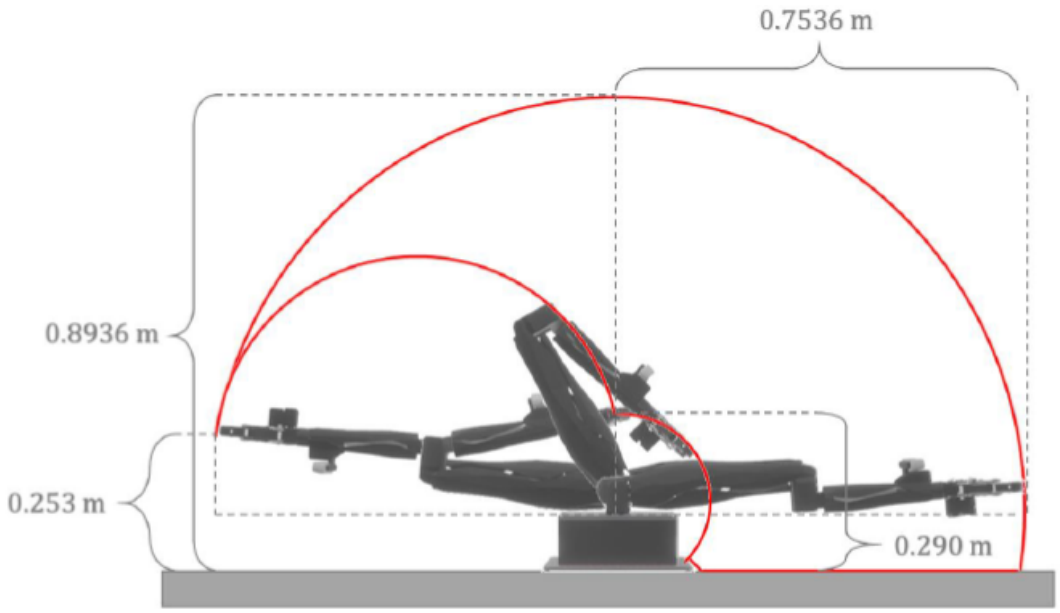


Figure 2.1: QArm reachable planar workspace side view [1]

Table 2.2: QArm physical parameters [1]

Physical and Dynamic Parameters			
L_1	Length of link 1	0.14	m
L_2	Length of link 2	0.35	m
L_3	Length of link 3	0.05	m
L_4	Length of link 4	0.25	m
m_1	Mass of link 1	0.7906	kg
m_2	Mass of link 2	0.4591	kg
m_3	Mass of link 3	0.269	kg
m_4	Mass of link 4	0.257	kg
G	Gravitational acceleration	9.81	kg m^2

2.2 Kinematics Modeling

Kinematics modeling describes the robot's motion by drawing a relationship between the joints and the end-effector positions and orientations with respect to a fixed Cartesian reference frame as shown in Fig. 2.3 where θ are the joint angles and p is the position and orientation of the end-

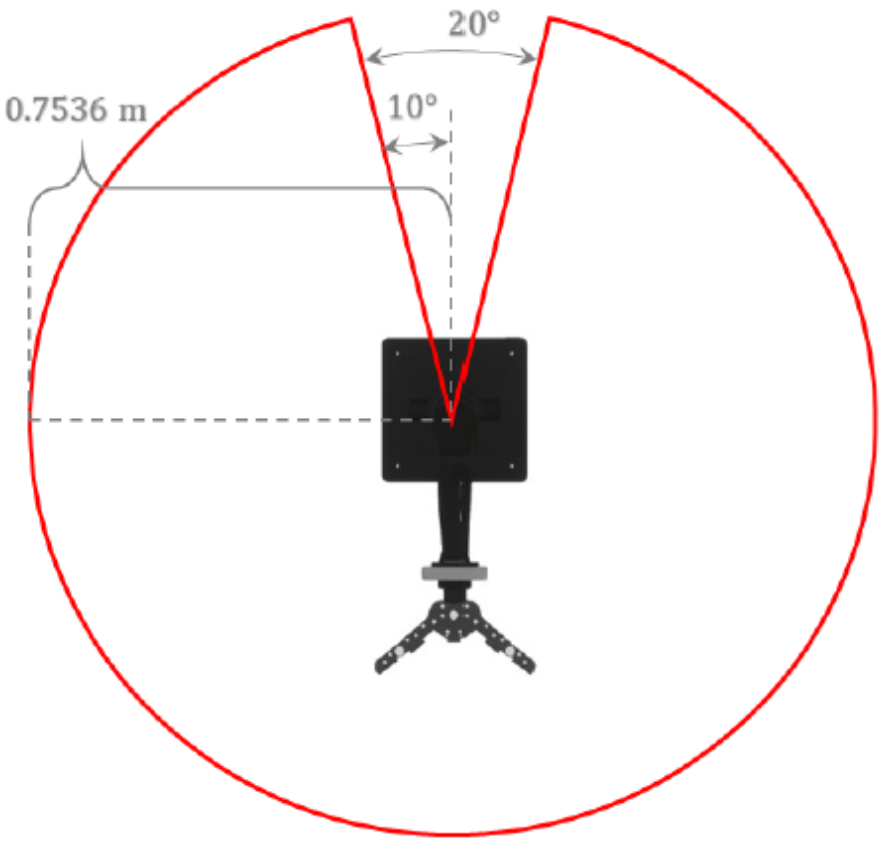


Figure 2.2: QArm reachable workspace top view [1]

effector. Contrary to dynamics modeling, kinematics does not consider the forces and moments of motion. It can be broken into forward kinematics and inverse kinematics.

The equations derived for the system modeling in this thesis will consider the frame assignment given by the manufacturer, see Fig. 2.4. Since the manipulator is in the home position as shown in Fig. 2.4, the joint space vector is expressed as $\boldsymbol{\theta}_{\text{home}} = \left[0, (\beta - \frac{\pi}{2}), -\beta, 0\right]^T$. To get rid of the offset in the joint space expression, a mapping to another space, $\boldsymbol{\phi}$ space, is introduced as follows:

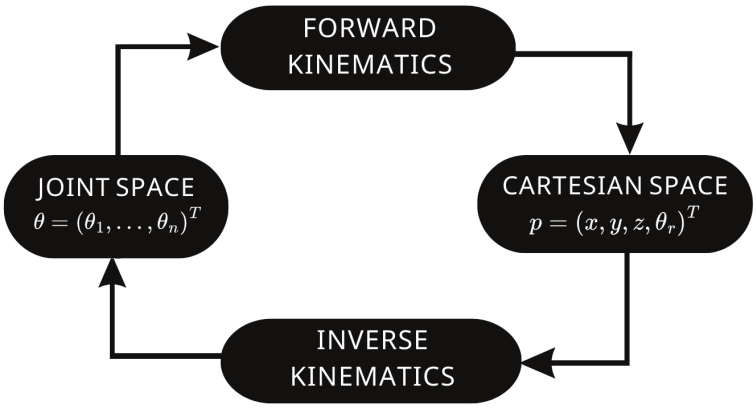


Figure 2.3: Forward and inverse kinematics flowchart

$$\begin{aligned} \phi_1 &= \theta_1 \\ \phi_2 &= \theta_2 + \frac{\pi}{2} - \beta \\ \phi_3 &= \theta_3 + \beta \\ \phi_4 &= \theta_4 \end{aligned} \tag{2.1}$$

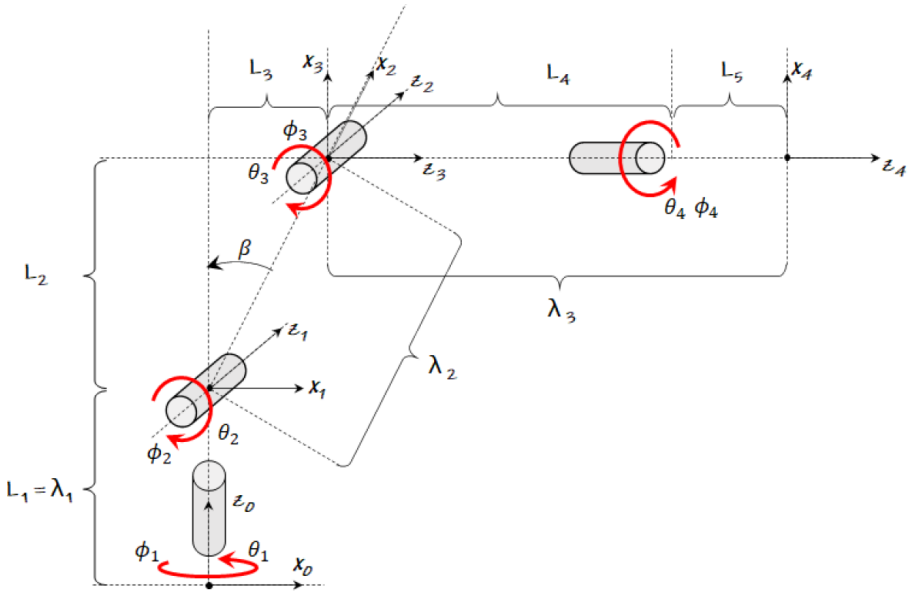


Figure 2.4: Frame assignments for Quanser QArm in home position [1]

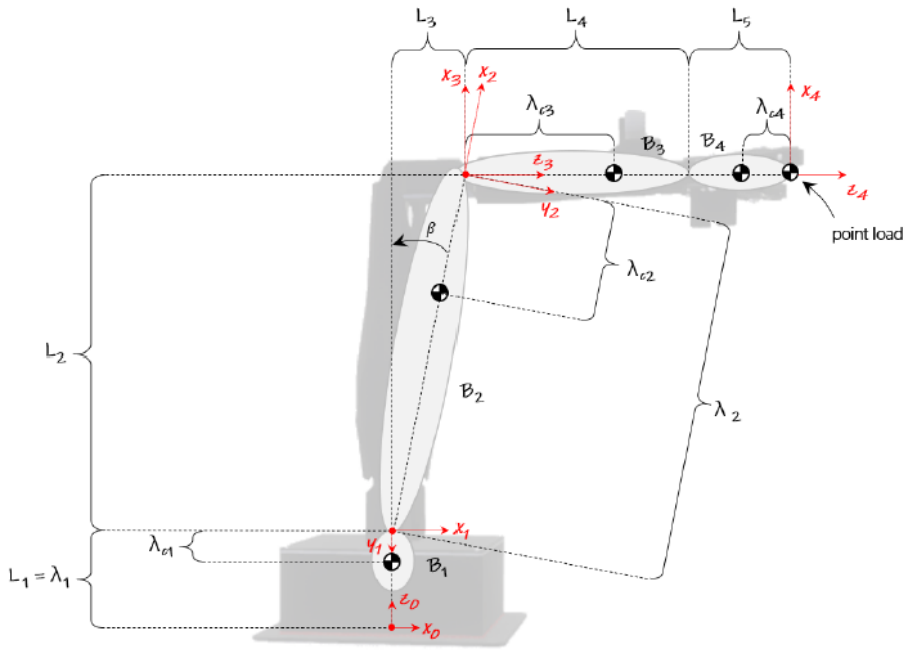


Figure 2.5: Frame assignments for Quanser QArm in home position [1]

2.2.1 Forward Kinematics

The forward kinematics concerns the description of the end-effector position and orientation as a function of the joint angles. With $\boldsymbol{\theta}$ being the joint angles, the end effector position in the Cartesian space with respect to a reference frame attached to the base of the robot can be expressed as follows:

$$\mathbf{p} = f(\boldsymbol{\theta}) \tag{2.2}$$

where $\boldsymbol{\theta} = \begin{bmatrix}\theta_1, \theta_2, \theta_3, \theta_4\end{bmatrix}^T$ is the representation of the joint states and $\mathbf{p} = \begin{bmatrix}p_x, p_y, p_z\end{bmatrix}^T$ the position of the end-effector in xyz-coordinates.

For the convenience of the reader and to simplify the mathematical formulations the abbreviations and mappings in Table 2.3 are considered.

The Denavit–Hartenberg (DH) convention is a simplified way to formulate the kinematics

Table 2.3: Mathematical abbreviation and mapping

λ_1	L_1
λ_2	$\sqrt{L_2^2 + L_3^2}$
λ_3	$\mathrm{L}_4 + L_5$
β	$\tan^{-1}(\frac{L_3}{L_4})$
ϕ_1	θ_1
ϕ_2	$\theta_2 + \frac{\pi}{2} - \beta$
ϕ_3	$\theta_3 + \beta$
ϕ_4	θ_4
$\cos\theta_i$	c_i
$\sin\theta_i$	s_i
$\cos(\theta_i + \theta_j)$	c_{ij}
$\sin(\theta_i + \theta_j)$	s_{ij}

Table 2.4: Denavit–Hartenberg table for the QArm.

i	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	λ_1	θ_1
2	λ_2	0	0	θ_2
3	0	$-\frac{\pi}{2}$	0	θ_3
4	0	0	λ_3	θ_4

equations as a series of transformation matrices. It consists of four parameters:

- The joint angle θ_i, which is the angle measured about z_{i-1} from x_{i-1} to x_i.
- The link length a_i, which is the distance along x_i from z_{i-1} to z_i.
- The link twist α_i, which is the angle measured about x_i from z_{i-1} and z_i.
- The link offset d_i, which is the offset along z_{i-1} from x_{i-1} to x_i.

The DH table for QArm is presented in Table 2.4.

Let us consider the general form of the homogeneous transformation matrix:

$$ {}^{i-1}T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

Then, the transformation matrix for each link can be computed as Eq. (2.4).

$$ \begin{aligned} {}^0T_1 &= \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & \lambda_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^1T_2 &= \begin{bmatrix} c_2 & -s_2 & 0 & \lambda_2 c_2 \\ s_2 & c_2 & 0 & \lambda_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^2T_3 &= \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ {}^3T_4 &= \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & \lambda_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.4} $$

The forward kinematics formulation to determine ${}^0T_4$ can be found by multiplying the ${}^{i-1}T_i$ matrices, where $i = 1, 2, 3, 4$, and it is given as:

$$ {}^0T_4 = \left[\begin{array}{ccc|c} & {}^0\mathbf{R}_4 & & \mathbf{p} \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.5} $$

where ${}^0\mathbf{R}_4$ is the rotation matrix that gives the orientation of the end effector, shown in Eq. (2.6), and $\mathbf{p}$ represents its position of the end effector in the Cartesian space, shown in Eq. (2.7).

$$ {}^0R_4 = \begin{bmatrix} c_1c_{23}c_4 + s_1s_4 & -c_1c_{23}s_4 + s_1c_4 & -c_1c_{23} \\ s_1c_{23}c_4 - s_1c_4 & -s_1c_{23}s_4 - c_1c_4 & s_1s_{23} \\ -s_{23}c_4 & s_{23}s_4 & -c_{23} \end{bmatrix} \tag{2.6} $$

$$
\begin{aligned}
\mathbf{p} &= [p_x, p_y, p_z]^T \\
p_x &= \lambda_2 c_1 c_2 - \lambda_3 c_1 s_{23} = c_1(\lambda_2 c_2 - \lambda_3 s_{23}) \\
p_y &= \lambda_2 s_1 c_2 - \lambda_3 s_1 s_{23} = s_1(\lambda_2 c_2 - \lambda_3 s_{23}) \\
p_z &= \lambda_1 - \lambda_2 s_2 - \lambda_2 c_{23}
\end{aligned}
\tag{2.7}
$$

2.2.2 Inverse Kinematics

Opposite to forward kinematics, inverse kinematics describes the joint angles as a function of the end-effector position. It can result in multiple solutions or no solution at all given the mechanical and DOF constraints of the robot. The inverse kinematics problem can be defined as:

$$
\boldsymbol{\theta} = f(\mathbf{p}) \tag{2.8}
$$

By squaring the x and y components of Eq. (2.7) and adding them, it can be found that:

$$
\begin{aligned}
p_x^2 + p_y^2 &= c_1^2(\lambda_2 c_2 - \lambda_3 s_{23})^2 + s_1^2(\lambda_2 c_2 - \lambda_3 s_{23})^2 \\
&= (\lambda_2 c_2 - \lambda_3 s_{23})^2
\end{aligned}
\tag{2.9}
$$

This results in:

$$
\pm\sqrt{p_x^2 + p_y^2} = \lambda_2 c_2 - \lambda_3 s_{23} \tag{2.10}
$$

The z component of Eq. (2.7) can be rewritten as follows:

$$
\lambda_1 - p_z = \lambda_2 s_2 + \lambda_3 c_{23} \tag{2.11}
$$

Recall the following trigonometric identities:

$$
\begin{aligned}
A c_2 + B c_{23} + C s_{23} &= D \\
A s_2 + B s_{23} - C c_{23} &= H
\end{aligned}
\tag{2.12}
$$

where

$$
\begin{aligned}
A &= \lambda_2 \\
B &= 0 \\
C &= -\lambda_3 \\
D &= \pm\sqrt{p_x^2 + p_y^2} \\
H &= \lambda_1 - p_z \\
F &= \frac{D^2 + H^2 - A^2 - C^2}{2A}
\end{aligned}
\tag{2.13}
$$

Solving for θ_3 results in two possible solutions:

$$
\begin{aligned}
\theta_{3_1} &= 2 \text{ atan2}\left(C + \sqrt{C^2 + F^2}, F\right) \\
\theta_{3_2} &= 2 \text{ atan2}\left(C - \sqrt{C^2 + F^2}, F\right)
\end{aligned}
\tag{2.14}
$$

where atan2 is the arctan function that takes two arguments and results in an angle in the range of $[-\pi, \pi]$. Let us introduce the following mappings:

$$
\begin{aligned}
M &= A + Cs_3 \\
N &= -Cc_3 \\
c_2 &= \frac{DM + HN}{M^2 + N^2} \\
s_2 &= \frac{H - Nc_2}{M}
\end{aligned}
\tag{2.15}
$$

Solving for θ_2:

$$
\theta_2 = \text{atan2}\left(s_2, c_2\right) \tag{2.16}
$$

It is worth mentioning that the calculation of θ_2 from Eq. (2.16) depends on θ_3, which has two values as shown in Eq. (2.14). Additionally, c_2 depends on D, which has two values according to Eq. (2.13). Therefore, Eq. (2.16) will result in four possible solutions that can be selected based on the robot's task constraints and performance requirements.

With θ_2 and θ_3 known, θ_1 can be solved for from Eq. (2.7) as follows:

$$\theta_1 = \text{atan2}\left(\frac{p_y}{\lambda_2 c_2 - \lambda_3 s_{23}}, \frac{p_x}{\lambda_2 c_2 - \lambda_3 s_{23}}\right) \tag{2.17}$$

It is worth mentioning that Eq. (2.17) results in one angle θ_1 for each set of angles θ_2 and θ_3.

Finally, the wrist angle θ_4 is an independent angle and it can be set freely within the range of $\pm 160^\circ$ as reported in Table 2.1.

2.3 Dynamics Modeling

The Lagrangian $\mathcal{L}$ for an n-DOF manipulator is given by:

$$\mathcal{L} = \mathcal{T} - \mathcal{P} \tag{2.18}$$

where $\mathcal{T}$ is the kinetic energy and $\mathcal{P}$ is the potential energy. for an n-DOF manipulator, $\mathcal{T}$ and $\mathcal{P}$ can be expressed as:

$$\mathcal{T}_j = \frac{1}{2} m_j ({}^{\mathbf{i}}\mathbf{v_{cj}})^T ({}^{\mathbf{i}}\mathbf{v_{cj}}) + \frac{1}{2} ({}^i\boldsymbol{\omega}_j)^T ({}^{\mathbf{i}}\mathbf{I_{cj}}) ({}^i\boldsymbol{\omega}_j) \tag{2.19}$$

$$\mathcal{P}_j = -m_j ({}^{\mathbf{0}}\mathbf{g})^{\mathbf{T}} ({}^{\mathbf{0}}\mathbf{p_{cj}}) \tag{2.20}$$

where m_j is the mass of link j, ${}^{\mathbf{i}}\mathbf{v_{cj}}$ is the velocity of the center of mass of link j expressed in the ith frame, ${}^{\mathbf{i}}\mathbf{I_{cj}}$ is the second mass moment of inertia matrix of link j about its center of mass expressed in the ith frame, and ${}^i\boldsymbol{\omega}_j$ is the angular velocity of link j expressed in the corresponding ith frame, $i = 1, \cdots, n$ denotes the reference frame, and $j = 1, \cdots, n$ is the link number.

Therefore the Lagrangian can be rewritten as the sum of the kinetic energy of each link minus the sum of the potential energy of each link:

$$\mathcal{L} = \sum_{j=1}^{n} \mathcal{T}_j - \sum_{j=1}^{n} \mathcal{P}_j \tag{2.21}$$

where $j = 1, \cdots, n$ is the link number of the manipulator.

Then, the equation of motion is defined as:

$$\tau_j = \frac{d}{dt}\left(\frac{d\mathcal{L}}{\dot{\theta}_j}\right) - \frac{d\mathcal{L}}{\theta_j}, \qquad j = 1, \cdots, n \tag{2.22}$$

where τ_j corresponds to the torques at each joint of the manipulator.

From Eq. (2.22), the dynamics model can be expressed as follows:

$$\begin{bmatrix}\tau_1\\ \tau_2\\ \tau_3\\ \tau_4\end{bmatrix} = \begin{bmatrix}M_{11} & 0 & 0 & M_{14}\\ 0 & M_{22} & M_{23} & 0\\ 0 & M_{32} & M_{33} & 0\\ M_{41} & 0 & 0 & M_{44}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\\ \ddot{\theta}_4\end{bmatrix} + \begin{bmatrix}B_{11} & B_{12} & 0 & 0 & B_{15} & B_{16}\\ 0 & 0 & B_{23} & B_{24} & 0 & 0\\ 0 & 0 & B_{33} & 0 & 0 & 0\\ B_{41} & B_{42} & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}_1\dot{\theta}_2\\ \dot{\theta}_2\dot{\theta}_3\\ \dot{\theta}_1\dot{\theta}_4\\ \dot{\theta}_2\dot{\theta}_3\\ \dot{\theta}_2\dot{\theta}_4\\ \dot{\theta}_3\dot{\theta}_4\end{bmatrix}$$
$$+ \begin{bmatrix}0 & 0 & 0 & 0\\ C_{32} & 0 & 0 & 0\\ C_{32} & C_{32} & 0 & 0\\ 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}_1^2\\ \dot{\theta}_2^2\\ \dot{\theta}_3^2\\ \dot{\theta}_4^2\end{bmatrix} + g\begin{bmatrix}0\\ G_2\\ G_3\\ G_4\end{bmatrix} \tag{2.23}$$

where the inertia, Coriolis, centrifugal and gravitational parameters are:

$$\begin{aligned} M_{11} = {} & I_{1A} + I_{3A}\,{c_{23}}^2 + I_{4A}\,{c_{23}}^2 + I_{2L}\,{c_2}^2 + I_{2A}\,{s_2}^2 + I_{3L}\,{s_{23}}^2 + \mathrm{I_{4L}}\,{s_{23}}^2 + {c_2}^2 \\ & {\lambda_2}^2\,m_3 + {c_2}^2\,{\lambda_2}^2\,m_4 + {\lambda_{c_3}}^2\,m_3\,{s_{23}}^2 + {c_2}^2\,m_2\left(\lambda_2 - \lambda_{c_2}\right)^2 + m_4\,{s_{23}}^2\left(\lambda_3 - \lambda_{c_4}\right)^2 \\ & -2\,c_2\,\lambda_2\,\lambda_{c_3}\,m_3\,s_{23} + c_2\,\lambda_2\,\lambda_3\,\mathrm{m_l}\,s_{23} - 2\,c_2\,\lambda_2\,m_4\,s_{23}\,\left(\lambda_3 - \lambda_{c_4}\right) \end{aligned} \tag{2.24}$$

$$M_{14} = -\mathrm{I_{4A}}\,c_{23} \tag{2.25}$$

$$\begin{aligned} M_{22} = {} & \mathrm{I_{2L}} + \mathrm{I_{3L}} + \mathrm{I_{4L}} + m_4\left(\lambda_2 + \lambda_3 - \lambda_{c_4}\right)^2 + {\lambda_2}^2\,m_3 + {\lambda_{c_3}}^2\,m_3 \\ & + {\lambda_2}^2\,\mathrm{m_l} + {\lambda_3}^2\,\mathrm{m_l} + m_2\left(\lambda_2 - \lambda_{c_2}\right)^2 - 2\,\lambda_2\,\lambda_{c_3}\,m_3\,s_3 - 2\,\lambda_2\,\lambda_3\,\mathrm{m_l}\,s_3 \end{aligned} \tag{2.26}$$

$$\begin{aligned} M_{23} = {} & m_3\,{\lambda_{c_3}}^2 - \lambda_2\,m_3\,s_3\,\lambda_{c_3} + \mathrm{I_{3L}} - \mathrm{m_l}\left({\lambda_3}^2 - \lambda_2\,\lambda_3\,s_3\right) \\ & -m_4\left(\lambda_3 - \lambda_{c_4}\right)\left(\lambda_{c_4} - \lambda_3 + \lambda_2\,s_3\right) \end{aligned} \tag{2.27}$$

$$\begin{aligned} M_{32} = {} & m_3\,{\lambda_{c_3}}^2 - \lambda_2\,m_3\,s_3\,\lambda_{c_3} + \mathrm{I_{3L}} + \mathrm{I_{4L}} - \mathrm{m_l}\left({\lambda_3}^2 - \lambda_2\,\lambda_3\,s_3\right) \\ & -m_4\left(\lambda_3 - \lambda_{c_4}\right)\left(\lambda_{c_4} - \lambda_3 + \lambda_2\,s_3\right) \end{aligned} \tag{2.28}$$

$$M_{33} = \mathrm{I_{3L}} + \mathrm{I_{4L}} + {\lambda_{c_3}}^2\, m_3 - {\lambda_3}^2\, \mathrm{m_l} + m_4 \left(\lambda_3 - \lambda_{c_4}\right)^2 \tag{2.29}$$

$$M_{41} = -\mathrm{I_{4A}}\, c_{23} \tag{2.30}$$

$$M_{44} = I_{4A} \tag{2.31}$$

$$\begin{aligned} B_{11} = {} & 2\,\mathrm{I_{2A}}\, c_2\, s_2 - 2\,\mathrm{I_{2A}}\, c_{23}\, s_{23} - 2\,\mathrm{I_{3A}}\, c_{23}\, s_{23} - 2\,\mathrm{I_{2L}}\, c_2\, s_2 + 2\,\mathrm{I_{2L}}\, c_{23}\, s_{23} \\ & + 2\,\mathrm{I_{3L}}\, c_{23}\, s_{23} - 2\, c_2\, {\lambda_2}^2\, m_3\, s_2 - 2\, c_2\, {\lambda_2}^2\, m_4\, s_2 + 2\, c_{23}\, {\lambda_{c_3}}^2\, m_3\, s_{23} - 2\, c_2\, {\lambda_2}^2\, \mathrm{m_l}\, s_2 \\ & + 2\, c_{23}\, {\lambda_3}^2\, \mathrm{m_l}\, s_{23} - 2\, c_2\, m_2\, s_2 \left(\lambda_2 - \lambda_{c_2}\right)^2 + 2\, c_{23}\, m_4\, s_{23} \left(\lambda_3 - \lambda_{c_4}\right)^2 \\ & - 2\, c_2\, c_{23}\, \lambda_2\, \lambda_{c_3}\, m_3 - 2\, c_2\, c_{23}\, \lambda_2\, \lambda_3\, \mathrm{m_l} + 2\, \lambda_2\, \lambda_{c_3}\, m_3\, s_2\, s_{23} + 2\, \lambda_2\, \lambda_3\, \mathrm{m_l}\, s_2\, s_{23} \\ & - 2\, c_2\, c_{23}\, \lambda_2\, m_4 \left(\lambda_3 - \lambda_{c_4}\right) + 2\, \lambda_2\, m_4\, s_2\, s_{23} \left(\lambda_3 - \lambda_{c_4}\right) \end{aligned} \tag{2.32}$$

$$\begin{aligned} B_{12} = {} & 2\, c_{23}\, {\lambda_{c_3}}^2\, m_3\, s_{23} - 2\, c_2\, c_{23}\, \lambda_2\, \lambda_{c_3}\, m_3 + 2\,\mathrm{I_{3A}}\, c_{23}\, s_{23} + 2\,\mathrm{I_{3L}}\, c_{23}\, s_{23} \\ & + 2\, c_{23}\, m_4\, s_{23} \left(\lambda_3 - \lambda_{c_4}\right)^2 - 2\,\mathrm{I_{4L}}\, c_{23}\, s_{23} - 2\,\mathrm{I_{4A}}\, c_{23}\, s_{23} - 2\, c_2\, c_{23} \\ & \lambda_2\, m_4 \left(\lambda_3 - \lambda_{c_4}\right) + 2\, c_{23}\, {\lambda_3}^2\, \mathrm{m_l}\, s_{23} - 2\, c_2\, c_{23}\, \lambda_2\, \lambda_3\, \mathrm{m_l} \end{aligned} \tag{2.33}$$

$$B_{15} = \mathrm{I_{2A}}\, s_{23} \tag{2.34}$$

$$B_{16} = \mathrm{I_{4A}}\, s_{23} \tag{2.35}$$

$$B_{23} = c_3\, \lambda_2\, \lambda_3\, \mathrm{m_l} - c_3\, \lambda_2\, \lambda_{c_3}\, m_3 - c_3\, \lambda_2\, m_4 \left(\lambda_3 - \lambda_{c_4}\right) \tag{2.36}$$

$$B_{24} = -2\, c_3\, \lambda_2\, \lambda_{c_3}\, m_3 - 2\, c_3\, \lambda_2\, \lambda_3\, \mathrm{m_l} \tag{2.37}$$

$$B_{33} = -\mathrm{I_{4A}}\, s_{23} \tag{2.38}$$

$$B_{41} = \mathrm{I_{4A}}\, s_{23} \tag{2.39}$$

$$B_{42} = \mathrm{I_{4A}}\, s_{23} \tag{2.40}$$

$$
\begin{aligned}
C_{21} = {\lambda_2}^2\, m_4\, {s_2}^2 - \mathrm{I_{2A}}\, c_2\, s_2 + \mathrm{I_{3A}}\, c_{23}\, s_{23} + \mathrm{I_{4A}}\, c_{23}\, s_{23} + \mathrm{I_{2L}}\, c_2\, s_2 \\
-\mathrm{I_{3L}}\, c_{23}\, s_{23} - \mathrm{I_{4L}}\, c_{23}\, s_{23} + c_2\, {\lambda_2}^2\, m_3\, s_2 - c_{23}\, {\lambda_{c_3}}^2\, m_3\, s_{23} \\
+c_2\, {\lambda_2}^2\, \mathrm{m_l}\, s_2 - c_{23}\, {\lambda_3}^2\, \mathrm{m_l}\, s_{23} + c_2\, m_2\, s_2\, \left(\lambda_2 - \lambda_{c_2}\right)^2 \\
-c_{23}\, m_4\, s_{23}\, \left(\lambda_3 - \lambda_{c_4}\right)^2 + c_2\, c_{23}\, \lambda_2\, \lambda_{c_3}\, m_3 + c_2\, c_{23}\, \lambda_2\, \lambda_3\, \mathrm{m_l} \\
-\lambda_2\, \lambda_{c_3}\, m_3\, s_2\, s_{23} - \lambda_2\, \lambda_3\, \mathrm{m_l}\, s_2\, s_{23} + c_2\, c_{23}\, \lambda_2\, m_4\, \left(\lambda_3 - \lambda_{c_4}\right) \\
-\lambda_2\, m_4\, s_2\, s_{23}\, \left(\lambda_3 - \lambda_{c_4}\right)
\end{aligned}
\tag{2.41}
$$

$$
\begin{aligned}
C_{31} = \mathrm{I_{3A}}\, c_{23}\, s_{23} + \mathrm{I_{4A}}\, c_{23}\, s_{23} - \mathrm{I_{3L}}\, c_{23}\, s_{23} - \mathrm{I4L}\, c_{23}\, s_{23} - c_{23}\, {\lambda_{c_3}}^2\, m_3\, s_{23} \\
-c_{23}\, {\lambda_3}^2\, \mathrm{m_l}\, s_{23} - c_{23}\, m_4\, s_{23}\, \left(\lambda_3 - \lambda_{c_4}\right)^2 + c_2\, c_{23}\, \lambda_2\, \lambda_{c_3}\, m_3 \\
+c_2\, c_{23}\, \lambda_2\, \lambda_3\, \mathrm{m_l} + c_2\, c_{23}\, \lambda_2\, m_4\, \left(\lambda_3 - \lambda_{c_4}\right)
\end{aligned}
\tag{2.42}
$$

$$
C_{32} = c_3\, \lambda_2\, \lambda_{c_3}\, m_3 + c_3\, \lambda_2\, \lambda_3\, \mathrm{m_l} \tag{2.43}
$$

$$
\begin{aligned}
G_2 = m_3\, \left(c_2\, \lambda_2 - \lambda_{c_3}\, s_{23}\right) + \mathrm{m_l}\, \left(c_2\, \lambda_2 - \lambda_3\, s_{23}\right) \\
+m_4\, \left(c_2\, \lambda_2 - s_{23}\, \left(\lambda_3 - \lambda_{c_4}\right)\right) + c_2\, m_2\, \left(\lambda_2 - \lambda_{c_2}\right)
\end{aligned}
\tag{2.44}
$$

$$
G_3 = \lambda_{c_3}\, m_3\, s_{23} + \lambda_3\, \mathrm{m_l}\, s_{23} + m_4\, s_{23}\, \left(\lambda_3 - \lambda_{c_4}\right) \tag{2.45}
$$

$$
G_4 = 0 \tag{2.46}
$$

where m_l is the payload mass and the mass moment of inertia of the links are expressed as follows:

$$
{}^1I_{c_1} = \begin{bmatrix} I_{1L} & 0 & 0 \\ 0 & I_{1A} & 0 \\ 0 & 0 & I_{1L} \end{bmatrix} \tag{2.47}
$$

$$
{}^2I_{c_2} = \begin{bmatrix} I_{2A} & 0 & 0 \\ 0 & I_{2L} & 0 \\ 0 & 0 & I_{2L} \end{bmatrix} \tag{2.48}
$$

$$
{}^3I_{c_3} = \begin{bmatrix} I_{3L} & 0 & 0 \\ 0 & I_{3L} & 0 \\ 0 & 0 & I_{3A} \end{bmatrix} \tag{2.49}
$$

$$ {}^{4}I_{c_4} = \begin{bmatrix} I_{4L} & 0 & 0 \\ 0 & I_{4L} & 0 \\ 0 & 0 & I_{4A} \end{bmatrix} \tag{2.50} $$

The links are considered cylindrical rods, therefore the components of the mass moment of inertia matrices are the axial mass moment of inertia (about the major axis) I_{iA} and two symmetric lateral components about the other two axes I_{iL}.

Eq. (2.23) can be expressed in a compact form as:

$$ \tau = \mathbf{M}(\theta)\ddot{\theta} + \mathbf{B}(\theta)\dot{\theta}_i\dot{\theta}_j + \mathbf{C}(\theta)\dot{\theta}_k^2 + \mathbf{G}(\theta) \tag{2.51} $$

where $i, j \in \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$, $k \in \{1,2,3,4\}$, $\mathbf{M} \in \mathbb{R}^{4\times4}$ is the Inertia matrix, $\mathbf{B} \in \mathbb{R}^{4\times4}$ is the Coriolis matrix, $\mathbf{C} \in \mathbb{R}^{4\times4}$ is the Centrifugal matrix, $\mathbf{G} \in \mathbb{R}^{4\times1}$ is the Gravity matrix, and $\boldsymbol{\tau} \in \mathbb{R}^{4\times1}$ is the torque vector.

3 Control Design

As mentioned in Section 1.1, designing an efficient trajectory-tracking controller is essential to ensure that the end-effector follows the desired trajectory which is crucial for tasks that require precise positioning (e.g. grasping). This chapter introduces a trajectory-tracking control strategy for a 4-DOF robotic manipulator by designing a Radial Basis Function Neural Network (RBFNN) based sliding mode control. First, a literature review on control for robot manipulators is presented. Second, the sliding mode manifold is formulated by defining the system dynamics and the control law. Next, the system's stability is proved by the Lyapunov theory. Finally, the proposed controller performance is demonstrated and compared to other controllers. The proposed controller is also validated for an autonomous grasping task.

3.1 Literature Review

The accurate and precise control of robot manipulators has been a topic discussed for numerous years. Distinct control methods can be used to address different kinds of problems. Regarding payload manipulation, trajectory planning, and tracking are essential to determine the desired path to be traced and the motions necessary to follow it. In this work, we will focus on the trajectory tracking problem.

Proportional-integral-derivative (PID) controllers have been largely used in robot applications, especially in industry [18]. However, since robot manipulators are highly coupled nonlinear time-

varying systems, the main drawback of using PID controllers is that they may not be robust enough to external disturbances so in order to try to track a desired command the gains of the PID controller might need adjustments throughout a task execution to account for the changes in the dynamics behavior. For this reason, [19,20] designed adaptive laws that can update the PID controller parameters depending on the system's requirements. In [21], the authors used a filtered error function and two adaptive estimators to compensate for the time variation in the dynamics model and adaptively update the gains of the PID controller. In [22], the authors designed a novel optimization algorithm named whale optimizer algorithm which aimed to determine the optimal values of the PID controller gains for trajectory tracking.

Although many studies extensively discussed the trajectory tracking problem [23–26], the ability to adapt to diverse circumstances and perturbations in real-time is still a challenging topic in optimal and consistent robot performance. Several approaches to this problem have been proposed including sliding mode control (SMC) methods [27–32], adaptive control methods [25, 33, 34], and data-driven/neural network-based methods [11, 35–37]. SMC has been largely used due to its robustness, like in [38], where an SMC with improved reaching law to endure finite time convergence for a nonlinear system was designed. However, the chattering phenomenon caused by the switching terms in the SMC is presented as a challenge because it can make the system unstable [39]. This is why in [40] an event-trigger sliding mode control with a time-varying threshold was designed to reduce the control execution time and chattering. And in [41], the authors combined fuzzy logic with sliding mode control aiming to reduce the chattering phenomenon when tracking a trajectory.

Mathematical models of robots try to accurately approximate reality. However, there are unavoidable uncertainties and disturbances in addition to the highly coupled dynamics of robot manipulators. In [11, 12, 32, 34] as well as other controllers mentioned still depend on complete or partial knowledge of the parameters and the precise mathematical kinematics and dynamics model

of the system, which in real life is unrealistic. Therefore designing controllers that do not need to rely on the model description is an interesting area of research.

Several efforts to compensate for the system's unknown parameters, uncertainties, and external disturbances have been proposed. Neural networks (NNs) can be an effective tool when dealing with complex and unknown system models since they are able to learn parameters and approximate nonlinear functions [42]. As a consequence, a variety of nonlinear control strategies have been combined with neural networks to improve the control performance. In [43], a fixed-time neural network controller with prescribed transient performance was presented where the authors relaxed the upper bound assumption for the NN weights. In [44], the authors used two RBFNNs, one to estimate the system model offline and another one to compensate for the uncertainties of the nonlinear system. Authors in [45] used an RBFNN to approximate the uncertainties of a robot manipulator combined with a nonsingular terminal sliding mode control with a modified sliding variable to guarantee the system performance with parameter uncertainty. The work done in [46] proposed a disturbance observer combined with an RBF neural network to improve the system's robustness by estimating unknown parameters and approximating disturbances.

Neural network reinforcement learning methods have also been applied for robot manipulators' trajectory tracking. In [47], the authors proposed a deep reinforcement learning strategy to optimize a trajectory while training the model in the real robot. In [48], the Deep Deterministic Policy Gradient (DDPG) was combined with sliding mode control to approximate the system's uncertainties and to ensure finite-time convergence. Authors in [49] combined a fixed-time sliding mode control with an RL control algorithm based on RBFNN to guarantee fixed-time convergence and improve steady-state accuracy. Although RL methods can be precise, they are computationally expensive.

External disturbances and perturbations such as payload variations are factors that contribute to the controller performance, as a consequence designing controllers that can attenuate or com-

pensate for such factors is important. Several works investigated solutions to overcome or reject these perturbations such as disturbance observers [50–53] and disturbance rejection [54–56] controllers. In [57], an adaptive sliding mode control with a disturbance observer integrated with an RBFNN was designed to address parameter uncertainties and disturbances. In [58], the authors designed a backstepping sliding mode control strategy that combines a disturbance observer to compensate for the system's perturbations. Authors in [59] formulated a disturbance observer to estimate uncertain disturbances and then a disturbance rejection control scheme was designed to attenuate the disturbance. In [60], a continuous terminal sliding mode control was introduced to ensure finite-time convergence and a sliding mode differentiator estimated the disturbances and uncertainties of the system.

Based on the aforementioned considerations, this chapter introduces an adaptive RBFNN-based sliding mode control with a variable learning rate (VLR-RBFNN-SMC) for trajectory tracking of an n-DOF robot manipulator that assumes that the robot model is unknown. The neural network is used to estimate the system dynamics and the sliding mode control drives the model to follow the desired states to realize the trajectory tracking control.

3.2 RBFNN-based Sliding Mode Control Design

3.2.1 Preliminaries

Considering the model provided in section 2, the following properties, lemmas, and assumptions are considered [3, 61–63]:

Property 1. *[61] For the bounds of the inertia matrix,* $\mathbf{M}$ *is a symmetric positive definite matrix that satisfies the following:*

$$\sigma_1 \mathbf{I} \leq \mathbf{M}(\theta) \leq \sigma_2 \mathbf{I}, \qquad \forall\ \theta \in \mathbb{R}^n \times 1 \tag{3.1}$$

where σ_1 *and* σ_2 *are positive constants, and* $\mathbf{I}$ *is an identity matrix of* $n \times n$ *dimension.*

Property 2. *[61]* $\dot{\mathbf{M}}(\boldsymbol{\theta}) - 2\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ *is a skew-symmetric matrix that satisfies Eq.* (3.2) *for an arbitrary vector:*

$$\mathbf{x}^T \left[\dot{\mathbf{M}}(\boldsymbol{\theta}) - 2\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \right] \mathbf{x} = 0, \qquad \forall \ \mathbf{x} = [x_1, \cdots, x_n]^T \in \mathbb{R}^n \tag{3.2}$$

Assumption 1. *[3] The disturbance* $\mathbf{D}(\mathbf{t})$ *and its derivative are bounded and satisfy:*

$$\begin{aligned} ||\mathbf{D}(\mathbf{t})|| &\leq D_{max} \\ ||\dot{\mathbf{D}}(\mathbf{t})|| &\leq \dot{D}_{max} \end{aligned} \tag{3.3}$$

Assumption 2. *The upper bound acceleration of the desired position exists and is known.*

Assumption 3. *[3] The RBFNN ideal weight vector is bounded such that* $\mathbf{W}^* \leq \mathbf{W_{max}}^*$ *where* $\mathbf{W_{max}}^*$ *is a positive vector. And* $\mathbf{W}^*$ *is:*

$$\mathbf{W}^* = \arg \min_{\mathbf{W} \in \boldsymbol{\Omega}} [\sup |\hat{\mathbf{f}}(\mathbf{x}) - f(\mathbf{x})|] \tag{3.4}$$

Assumption 4. *[3] The RBFNN estimation error* ε *is bounded and sufficiently small such that:*

$$||\varepsilon|| \leq \varepsilon_{max} \tag{3.5}$$

Lemma 1. *[62] There exists a compact set* $\Omega_{\hat{\mathbf{W}}}$ *that satisfies:*

$$\Omega_{\hat{\mathbf{W}}} = \left\{ \hat{\mathbf{W}} \,\middle|\, ||\hat{\mathbf{W}}|| \leq \frac{H^*}{r_2} \right\} \tag{3.6}$$

where $\hat{\mathbf{W}}(\mathbf{t}) \in \Omega_{\hat{\mathbf{W}}}$, $\forall \ t \geq 0$, *given that* $\hat{\mathbf{W}}(0) \in \Omega_{\hat{\mathbf{W}}}$, r_2 *is the learning rate, and* $h_j \leq H^*$, *where* h_j *is the radial basis function.*

3.2.2 Radial Basis Function Neural Network Design

In this section, the RBF neural network structure design is presented, for more details the reader can refer to [3]. This neural network was designed to approximate the robotic arm system dynamics. The structure of this feedforward three-layer neural network consists of one input layer, a

hidden layer, and an output layer, which is shown in Fig. 3.1. The nodes in the hidden layer perform a nonlinear transformation by using a Gaussian Radial Basis Function expressed in Eq. (3.7) as a nonlinear activation function.

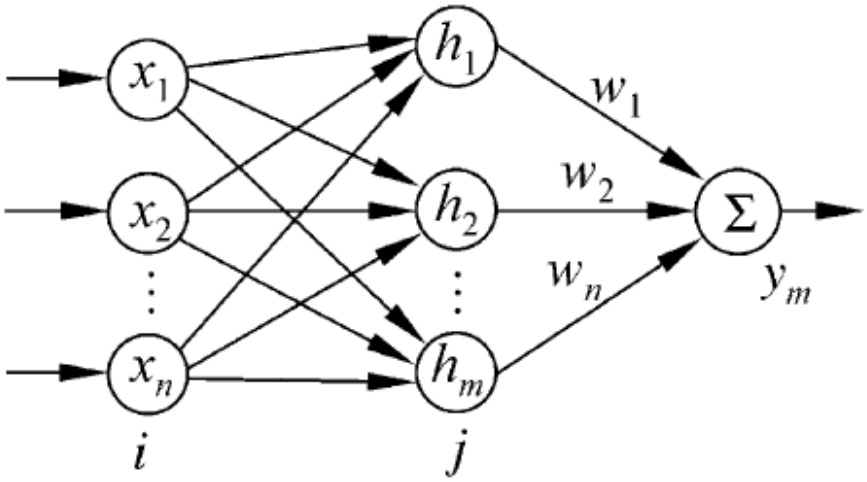


Figure 3.1: RBF neural network structure [3]

The input vector of the RBFNN is given by $\mathbf{x} = [x_i]^T$. For this implementation, the input vector is chosen to be $\mathbf{x} = [\mathbf{e}, \dot{\mathbf{e}}, \boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}_{\boldsymbol{d}}, \ddot{\boldsymbol{\theta}}_{\boldsymbol{d}}]^T$ where $\mathbf{e}, \dot{\mathbf{e}}, \boldsymbol{\theta}_{\boldsymbol{d}}, \dot{\boldsymbol{\theta}}_{\boldsymbol{d}}, \ddot{\boldsymbol{\theta}}_{\boldsymbol{d}} \in \mathbb{R}^{4\times 1}$. The radial basis activation function vector in the hidden layer is $\mathbf{h} = [h_j]^T$, where h_j is a Gaussian function that is defined as:

$$h_j = \exp\left(-\frac{||\mathbf{x} - \mathbf{c}_j||^2}{2\mathbf{b}_j^2}\right) \tag{3.7}$$

where $\mathbf{c} = [c_{ij}] = [c_{nm}]$ is the value of the center coordinate point of the Gaussian function Eq. (3.7), $\mathbf{b} = [b_1, \cdots, b_m]^T$ is its width value, j is the number of hidden layer nodes in the neural network and i is the input number of the network. Choosing the optimal values for the parameters c_{ij} and b_j is important because they will directly affect the Gaussian function mapping if they are not chosen well the RBF neural network will not converge to the area of interest.

The weight of the RBF is defined as $\mathbf{W} = [W_1, \cdots, W_m]^T$, and $\mathbf{f}(\mathbf{x})$ is the function representing the manipulator dynamics the RBF will approximate, so the output of the network is defined as the weighted sum of the output of the hidden layers as we can see in Eq. (3.8).

$$\hat{\mathbf{f}}(\mathbf{x}) = \hat{\mathbf{W}}^T \mathbf{h}(\mathbf{x}) \tag{3.8}$$

where $\hat{\mathbf{W}}$ is the estimation of the ideal weights of the neural network $\hat{\mathbf{W}}^*$ and $\tilde{\mathbf{W}} = \hat{\mathbf{W}}^* - \hat{\mathbf{W}}$.

The lack of adaptability and the chattering phenomenon are two drawbacks faced when dealing with SMC that may cause undesired behavior that can affect the tracking performance, such as vibrations. Therefore in this work, an adaptive mechanism to address this issue is designed. As shown in Eq. (3.9).

$$\begin{aligned} \dot{\tilde{\mathbf{W}}} &= -\gamma \mathbf{h}(\mathbf{x}) \mathbf{s}^T \\ \dot{\hat{\mathbf{W}}} &= \gamma \mathbf{h}(\mathbf{x}) \mathbf{s}^T \end{aligned} \tag{3.9}$$

where $\mathbf{s}$ is the sliding surface defined in Eq. (3.14) and γ is a variable learning rate defined by a Butterworth filter-type function [64]:

$$\gamma = \begin{cases} \dfrac{r_1}{1 + (||\mathrm{s}||/r_3)^{q+1}} + r_2, & \mathrm{s} \neq 0 \\ r_1 + r_2, & \mathrm{s} = 0 \end{cases} \tag{3.10}$$

where r_1, r_2, and r_3 are positive constants, $n \geq 0$, and $\gamma \leq r_1 + r_2$.

Remark 1. *In [62], the learning rate γ is a constant, and they have $\Omega_{\hat{\mathbf{W}}} = \left\{ \hat{\mathbf{W}} \middle| ||\hat{\mathbf{W}}|| \leq \dfrac{H^*}{\gamma} \right\}$. This work uses γ as an adaptive law that satisfies $\gamma \geq r_2$. The learning rate γ is at its minimum when $\gamma = r_2$, as a consequence, the right side of the inequality takes its maximum, so lemma 1 still holds.*

3.2.3 Sliding Mode Control Design

Sliding mode control is widely used to control robot manipulators due to their fast global convergence and robustness to external disturbances, modeling errors, and parameter variations [65].

Overall, two main steps are necessary for the formulation of this controller: the sliding mode which is the design of the sliding surface to reach the desired state, and the reaching mode which is the selection of the control law to reach the sliding surface.

In this section, the design of a sliding mode control based on RBFNN for unknown dynamics approximation is shown. The block diagram for the control schematic can be seen in Fig. 3.2. First, the RBF approximation is designed. Second, the dynamics model of the system is established. Third, the tracking errors are defined and the sliding surface is designed. Then, the control law is designed to ensure that the system's state reaches and stays on the sliding surface considering the desired system behavior and the control objectives. Finally, asymptotic stability is proved by Lyapunov theory, and the adaptation law and the variable learning rate are defined.

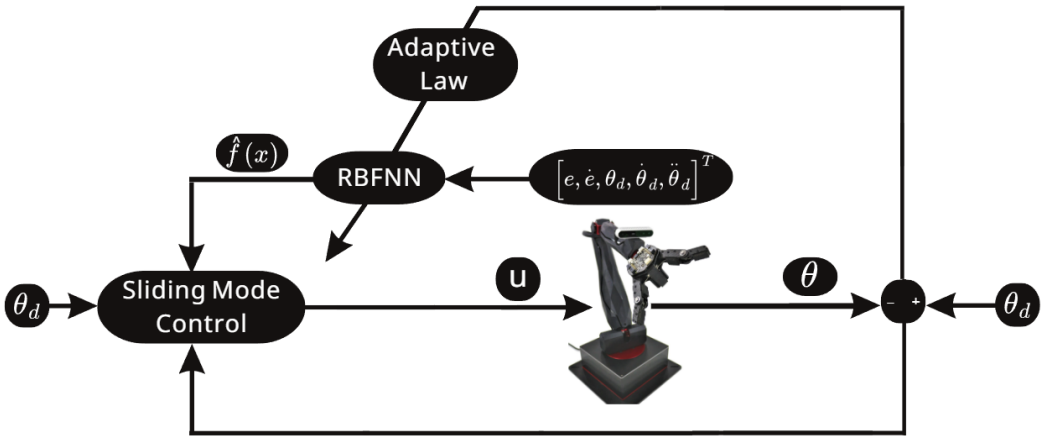


Figure 3.2: Control block diagram

To design the controller, consider the general Euler-Lagrange dynamics model of the system presented in Eq. (3.11), in addition to the disturbance vector D and with the generalized coordinate vector $\theta \in \mathbb{R}^n$ representing the system's states:

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) = \boldsymbol{\tau} + \mathbf{D} \tag{3.11}$$

Considering Eq. (3.11), and solving for $\ddot{\theta}$ yields,

$$\ddot{\boldsymbol{\theta}} = \mathbf{M}^{-1}(\boldsymbol{\theta})[-\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \mathbf{G}(\boldsymbol{\theta}) + \boldsymbol{\tau} + \mathbf{D}] \tag{3.12}$$

Let us define the position tracking error and its derivatives, velocity, and acceleration, as:

$$\begin{cases} \mathbf{e} = \boldsymbol{\theta}_d - \boldsymbol{\theta} \\ \dot{\mathbf{e}} = \dot{\boldsymbol{\theta}}_d - \dot{\boldsymbol{\theta}} \\ \ddot{\mathbf{e}} = \ddot{\boldsymbol{\theta}}_d - \ddot{\boldsymbol{\theta}} \end{cases} \tag{3.13}$$

Then the sliding mode surface and its derivative can be designed as:

$$\begin{cases} \mathbf{s} = \boldsymbol{\lambda}\mathbf{e} + \dot{\mathbf{e}} \\ \dot{\mathbf{s}} = \boldsymbol{\lambda}\dot{\mathbf{e}} + \ddot{\mathbf{e}} \end{cases} \tag{3.14}$$

where $\boldsymbol{\lambda} > \mathbf{0}$, where $\mathbf{0}$ is a null vector of the same dimension as $\boldsymbol{\lambda}$.

The RBFNN sliding mode control law with variable learning rate is given as:

$$\boldsymbol{\tau} = \hat{\mathbf{f}}(\mathbf{x}) + \mathbf{K}\mathbf{s} + (\boldsymbol{\varepsilon}_N + \boldsymbol{\varepsilon}_D)\text{sign}(\mathbf{s}) \tag{3.15}$$

where $\hat{\mathbf{f}}(\mathbf{x})$ is the output of the RBFNN, and is given by Eq. (3.8), $\mathbf{K}$ is a positive definite gain matrix, ε_N and ε_D are robust parameters to overcome the neural network approximation error ε and the disturbance $D(t)$ and satisfy $||\varepsilon|| \leq \varepsilon_N$, and $||D(t)|| \leq \varepsilon_D$, where $||\cdot||$ denotes the absolute value of its argument.

Remark 2. *The $sign(\mathbf{s})$ function in Eq.* (3.15) *is defined in Eq.* (3.16). *Its approximation is formulated as in [66] to attenuate the chattering phenomenon.*

$$\text{sign}\,(\mathbf{s}) = \frac{\mathbf{s}}{||\mathbf{s}|| + r} \tag{3.16}$$

where r can be a positive small function or a saturation function.

3.2.4 Lyapunov Stability Proof

Now, substituting Eq. (3.12) and Eq. (3.13) in Eq. (3.14), we have:

$$\dot{\mathbf{s}} = \ddot{\boldsymbol{\theta}}_{\mathbf{d}} - \mathbf{M^{-1}}(\boldsymbol{\theta})[-\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \mathbf{G}(\boldsymbol{\theta}) + \boldsymbol{\tau} + \mathbf{D}(\mathbf{t})] + \boldsymbol{\lambda}\dot{\mathbf{e}} \tag{3.17}$$

$$\mathbf{M}\dot{\mathbf{s}} = \mathbf{M}(\ddot{\boldsymbol{\theta}}_d + \boldsymbol{\lambda}\dot{\mathbf{e}}) - \mathbf{C}\mathbf{s} + \mathbf{C}(\dot{\boldsymbol{\theta}}_d + \boldsymbol{\lambda}\mathbf{e}) + \mathbf{G} + \mathbf{D} - \boldsymbol{\tau} \tag{3.18}$$

$\mathbf{f}(\mathbf{x})$ can be defined as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{M}(\ddot{\boldsymbol{\theta}}_d + \boldsymbol{\lambda}\dot{\mathbf{e}}) + \mathbf{C}(\dot{\boldsymbol{\theta}}_d + \boldsymbol{\lambda}\mathbf{e}) - \mathbf{G} \tag{3.19}$$

As a consequence we have:

$$\mathbf{M}\dot{\mathbf{s}} = \mathbf{f}(\mathbf{x}) - \mathbf{C}\mathbf{s} + \mathbf{D} - \boldsymbol{\tau} \tag{3.20}$$

From the RBFNN, we have $\tilde{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \widehat{\mathbf{f}}(\mathbf{x}) = \mathbf{W}^{*\mathbf{T}}\mathbf{h}(\mathbf{x}) + \epsilon - \hat{\mathbf{W}}^{\mathbf{T}}\mathbf{h}(\mathbf{x}) = \tilde{\mathbf{W}}^{\mathbf{T}}\mathbf{h}(\mathbf{x}) + \epsilon$ and $\tilde{\mathbf{W}} = \mathbf{W}^* - \widehat{\mathbf{W}}$.

Proof: Let us define the Lyapunov candidate as follows:

$$V = \frac{1}{2}\mathbf{s}^T\mathbf{M}\mathbf{s} + \frac{1}{2}tr(\tilde{\mathbf{W}}^T\gamma^{-1}\tilde{\mathbf{W}}) \tag{3.21}$$

Taking the time derivative of V, we have:

$$\dot{V} = \mathbf{s}^T\mathbf{M}\dot{\mathbf{s}} + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s} + \frac{1}{2}tr(\tilde{\mathbf{W}}^T\gamma^{-1}\dot{\tilde{\mathbf{W}}}) \tag{3.22}$$

Substituting (3.20) in (3.22):

$$\begin{aligned}\dot{V} = \mathbf{s}^T[\tilde{\mathbf{W}}^{\mathbf{T}}\mathbf{h}(\mathbf{x}) + \boldsymbol{\varepsilon} - \mathbf{C}\mathbf{s} + \mathbf{D} - \mathbf{K}\mathbf{s} \\ -(\boldsymbol{\varepsilon}_N + \boldsymbol{\varepsilon}_D)\text{sign}(\mathbf{s})] + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s} \\ + \frac{1}{2}tr(\tilde{\mathbf{W}}^T\gamma^{-1}\dot{\tilde{\mathbf{W}}})\end{aligned} \tag{3.23}$$

Let us consider $\sigma = \boldsymbol{\varepsilon} + \mathbf{D} - \mathbf{K}\mathbf{s} - (\boldsymbol{\varepsilon}_N + \boldsymbol{\varepsilon}_D)\text{sign}(\mathbf{s})$, then we have:

$$\begin{aligned}\dot{V} = \mathbf{s}^T\sigma - \mathbf{s}^T\mathbf{C}\mathbf{s} + \frac{1}{2}\mathbf{s}^T\dot{\mathbf{M}}\mathbf{s} \\ +\mathbf{s}^T\tilde{\mathbf{W}}^{\mathbf{T}}\mathbf{h}(\mathbf{x}) \\ +\frac{1}{2}tr(\tilde{\mathbf{W}}^T\gamma^{-1}\dot{\tilde{\mathbf{W}}})\end{aligned} \tag{3.24}$$

From property 2 we have that $\dot{\mathbf{M}}(\boldsymbol{\theta}) - 2\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = 0$. Then substituting the adaptive law from Eq (3.9) into Eq. (3.24), we find that:

$$\begin{aligned}\dot{V} = \mathbf{s}^T\sigma + \frac{1}{2}tr\tilde{\mathbf{W}}^{\mathbf{T}}(\gamma^{-1}(-\gamma\mathbf{h}(\mathbf{x})\mathbf{s}^T) + \mathbf{h}(\mathbf{x})\mathbf{s}^T) \\ \dot{V} = \mathbf{s}^T\sigma + \frac{1}{2}tr\tilde{\mathbf{W}}^{\mathbf{T}}(-\mathbf{h}(\mathbf{x})\mathbf{s}^T + \mathbf{h}(\mathbf{x})\mathbf{s}^T)\end{aligned} \tag{3.25}$$

We find that $\frac{1}{2}tr\tilde{\mathbf{W}}^{\mathbf{T}}(-\mathbf{h}(\mathbf{x})\mathbf{s}^T + \mathbf{h}(\mathbf{x})\mathbf{s}^T) = 0$ and we can see that the stability depends on the term σ.

$$\dot{V} = \mathbf{s}^T[\boldsymbol{\varepsilon} + \mathbf{D} - \mathbf{K}\mathbf{s} - (\boldsymbol{\varepsilon}_N + \boldsymbol{\varepsilon}_D)\text{sign}(\mathbf{s})] \tag{3.26}$$

So if $\mathbf{K} > 0$, and based on assumptions 1 and 4, we have $\boldsymbol{\varepsilon}_N > \boldsymbol{\varepsilon}_{max}$, and $\boldsymbol{\varepsilon}_D > \mathbf{D_{max}}$, then:

$$\dot{V} \leq 0 \tag{3.27}$$

The proof is complete.

3.3 Simulation Results

In order to avoid damage to the hardware platform, test the controller performance, and verify the impact of different learning rates on the system, a numerical simulation of the VLR-RBFNN-SMC controller was conducted.

Parameters Setting: The desired trajectory in the joint space is designed as:

$$\theta_d = \begin{bmatrix} 0.2\text{sin}(0.5t) \\ 0.2\text{sin}(0.6t) \\ 0.15\text{sin}(0.5t) \\ 0.2\text{sin}(0.4t) \end{bmatrix} \tag{3.28}$$

The manipulator dynamics were simulated given the mathematical equations from the system modeling discussed in Chapter 2. For the RBFNN, the neural network was designed with five nodes in the hidden layer, the variance is chosen as $\mathbf{b} = [0.5, 0.5, 0.5, 0.5, 0.5]$, and the center $\mathbf{c}$ is given as Eq. 3.29. The weight values are updated online by the adaptive law defined in Eq. (3.9) and their initial values are selected as $\mathbf{W} = [0.1, 0.1, 0.1, 0.1, 0.1]^T$. The variable learning rate parameters for the simulation and experimental results are set as $r_1 = 35$, $r_2 = 5$, $r_3 = 0.5$, and $q = 3$ throughout this thesis unless otherwise specified.

$$\mathbf{c} = \begin{bmatrix} -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -5 & -2.5 & 0 & 2.5 & 5 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \end{bmatrix} \tag{3.29}$$

The gains of the sliding mode control for the variable learning rate RBFNN in Eq. 3.15 are chosen as:

$$\varepsilon_N = \begin{bmatrix} 0.15 & 0.15 & 0.0015 & 0.0015 \end{bmatrix} \tag{3.30}$$

$$\varepsilon_D = \begin{bmatrix} 0.25 & 0.25 & 0.0025 & 0.0025 \end{bmatrix} \tag{3.31}$$

$$\mathbf{K} = \begin{bmatrix} 1.5 & 0 & 0 & 0 \\ 0 & 1.5 & 0 & 0 \\ 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 1.2 \end{bmatrix} \tag{3.32}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 8 \end{bmatrix} \tag{3.33}$$

The trajectory tracking in the joint space and the cartesian space are given in Fig. 3.3 to Fig. 3.9. The results show that the proposed controller can drive the manipulator to follow the desired trajectory.

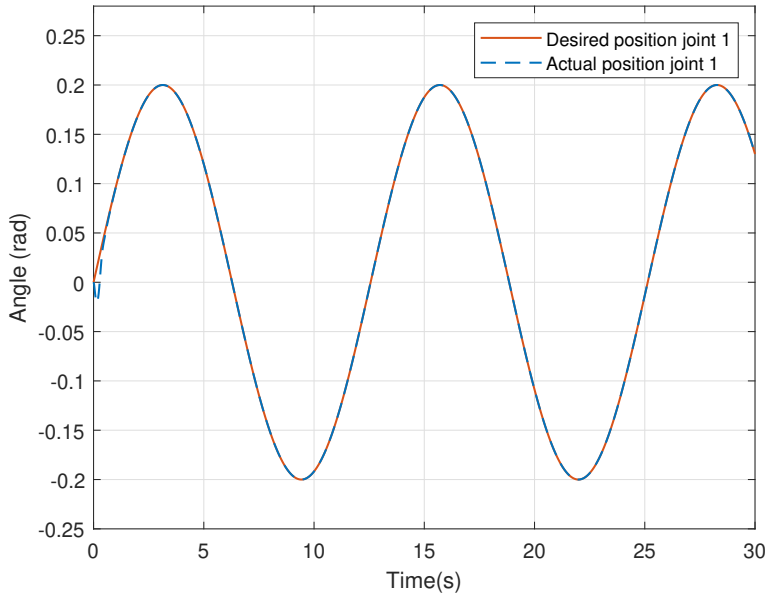


Figure 3.3: Simulation trajectory tracking for joint 1.

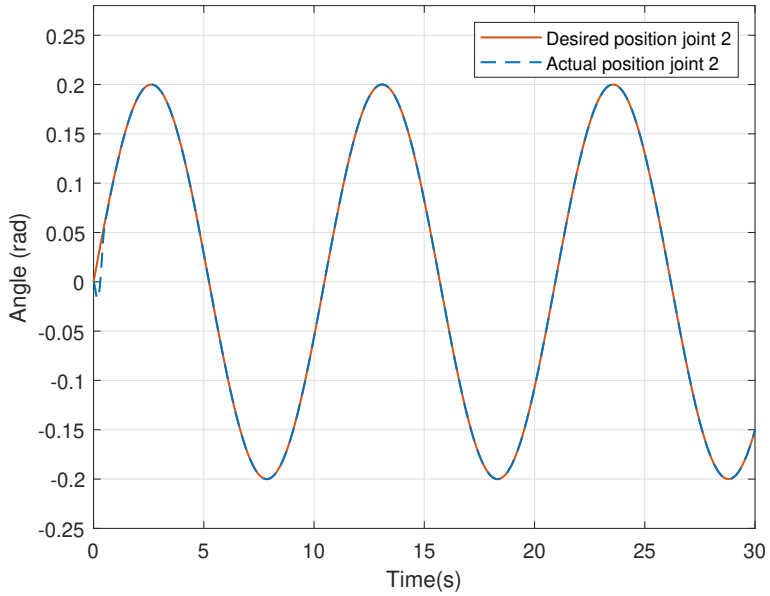


Figure 3.4: Simulation trajectory tracking for joint 2.

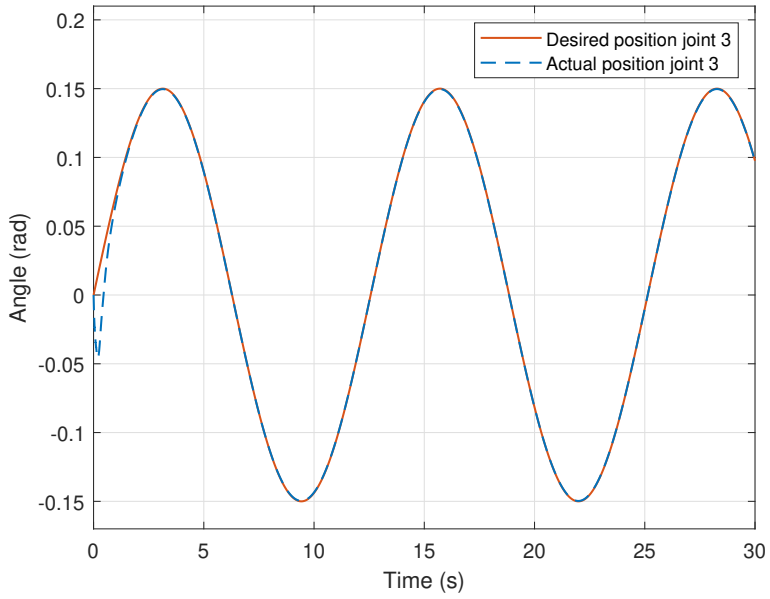


Figure 3.5: Simulation trajectory tracking for joint 3.

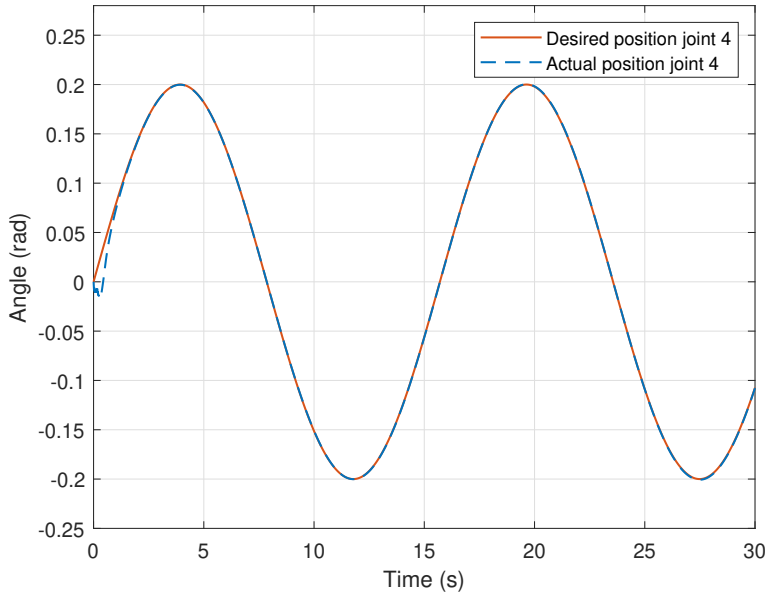


Figure 3.6: Simulation trajectory tracking for joint 4.

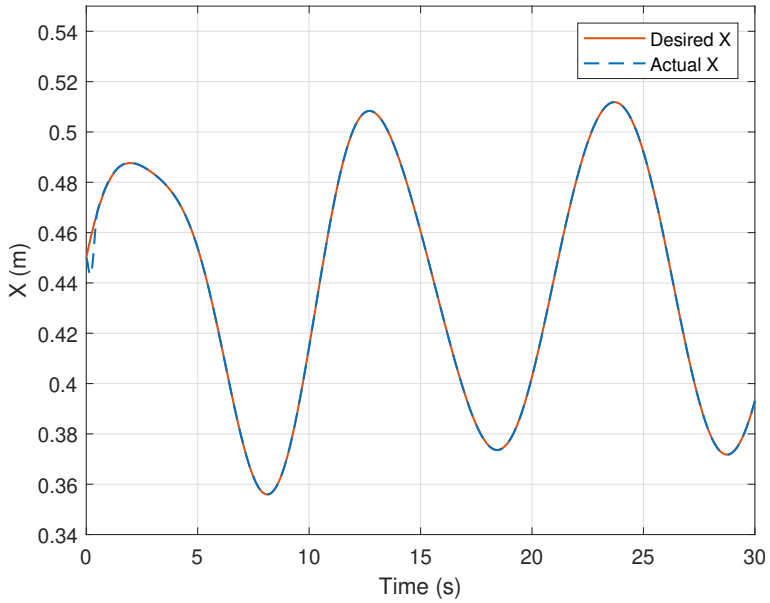


Figure 3.7: Simulation trajectory tracking in x-direction.

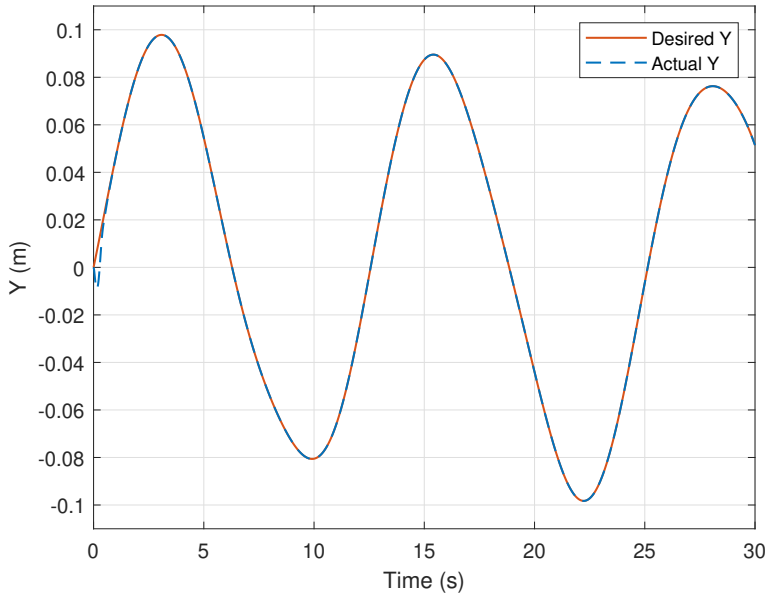


Figure 3.8: Simulation trajectory tracking in y-direction.

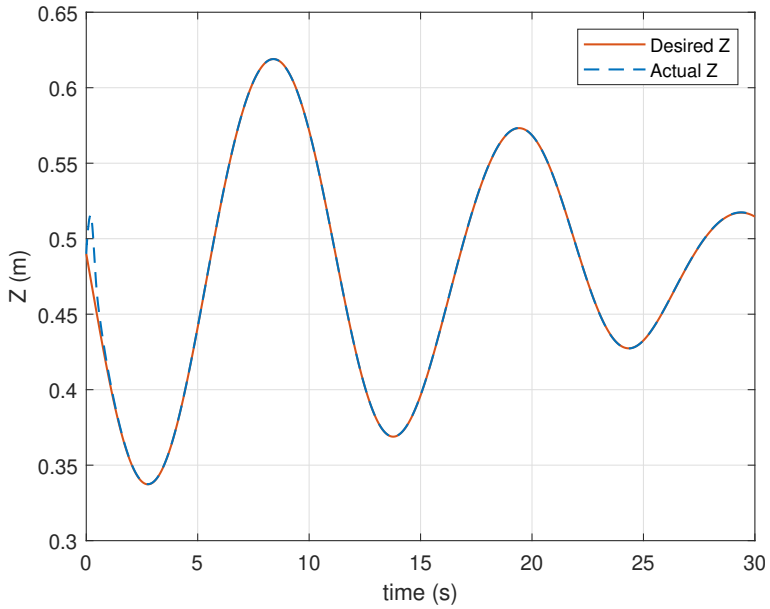


Figure 3.9: Simulation trajectory tracking in z-direction.

3.3.1 System Convergence for Different Learning Rates

Finding the best constant value for the learning rate, γ, is a trade-off between overshoot and steady-state error. Therefore, achieving a desired control performance requires carefully choosing and adjusting the values for the learning rate. The controller convergence will take longer if γ is too small due to the smaller weight changes. On the other hand, if γ is too large it may cause the system to jump the global minima and the optimal convergence value, resulting in rapid weight changes. For this reason, adopting a variable learning rate is a reasonable solution to avoid convergence problems such as settling time and oscillations.

This case study will show how the learning rate affects system convergence and stability. To do that, we implemented the controller with different learning rate values.

The absolute mean tracking errors for different learning rates are shown in Fig. 3.10 to Fig. 3.13. We can see that the tracking error is also larger if γ is too small or too large. However, we can see a range of values in which the tracking error decreases before increasing again. This would be the

ideal range of values to consider for the learning rate.

The overshoot for different learning rates is shown from Fig. 3.14 to Fig. 3.17. If γ is too small or too large, the overshoot is also larger. However, we can see that there is a range of values in which the overshoot decreases before increasing again. This would be the ideal range to consider for a constant learning rate.

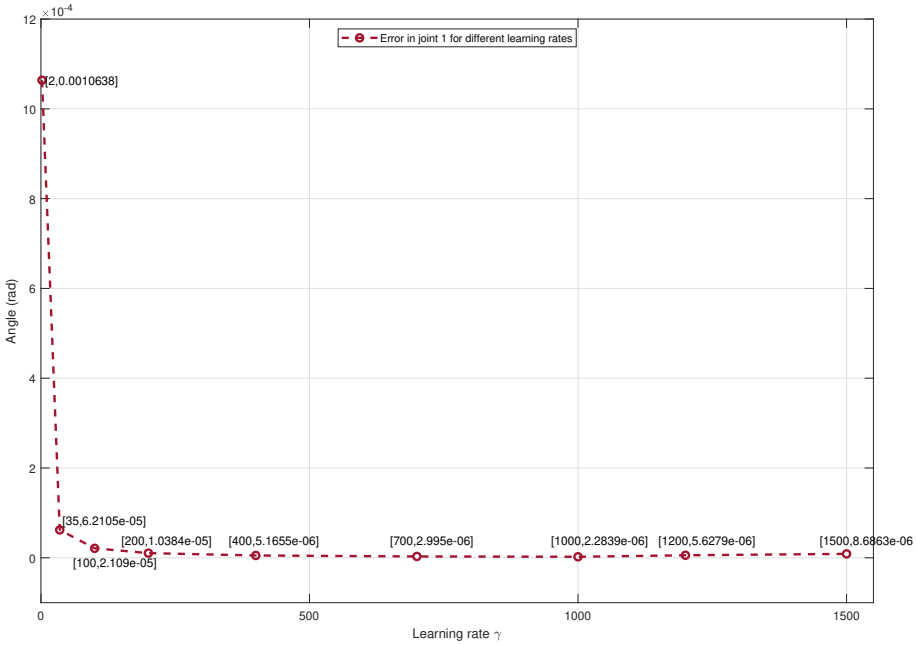


Figure 3.10: Tracking error of different learning rates for joint 1

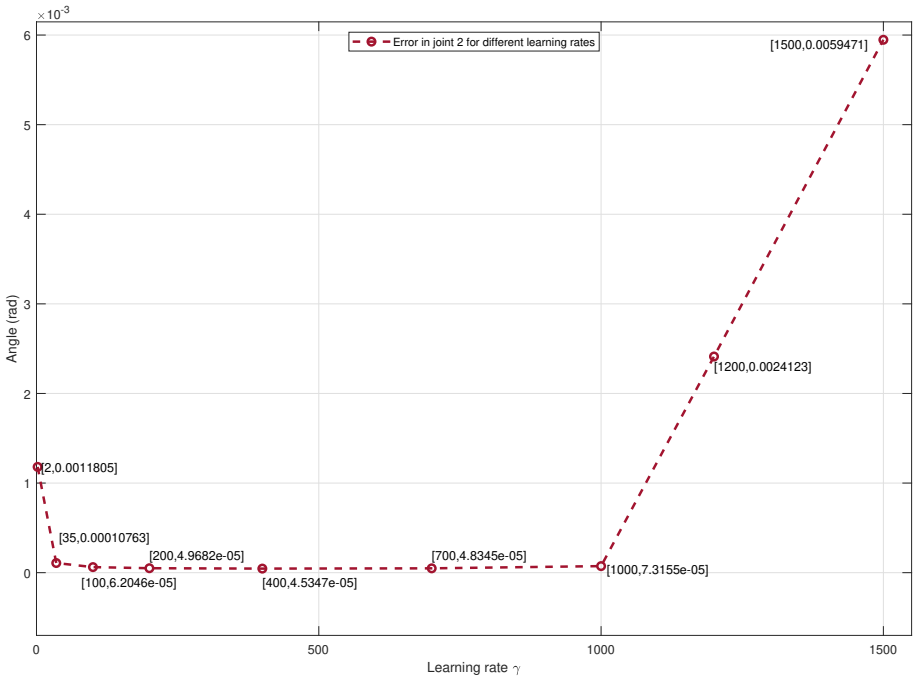


Figure 3.11: Tracking error of different learning rates for joint 2

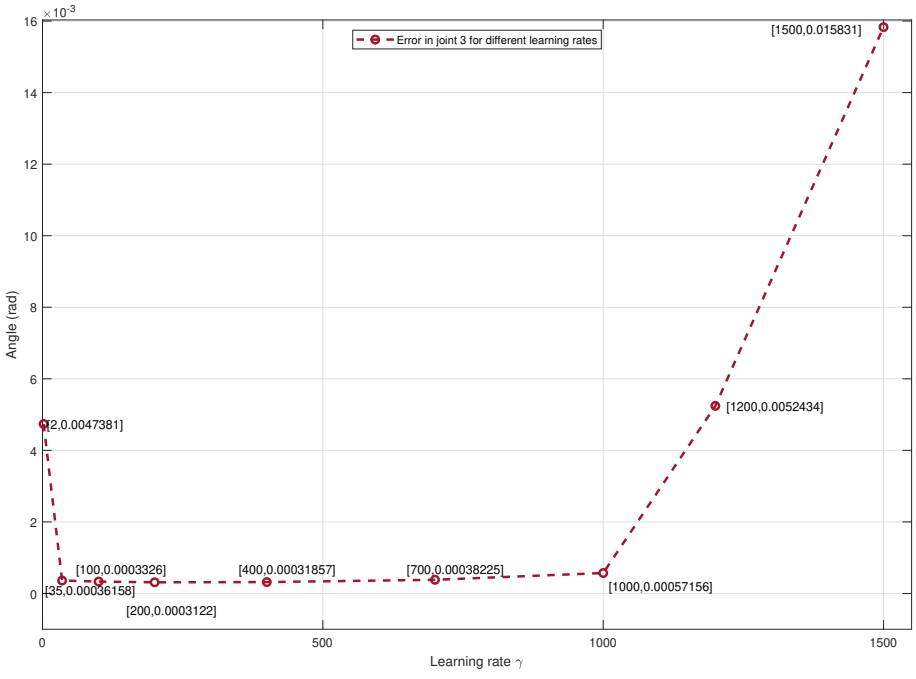


Figure 3.12: Tracking error of different learning rates for joint 3

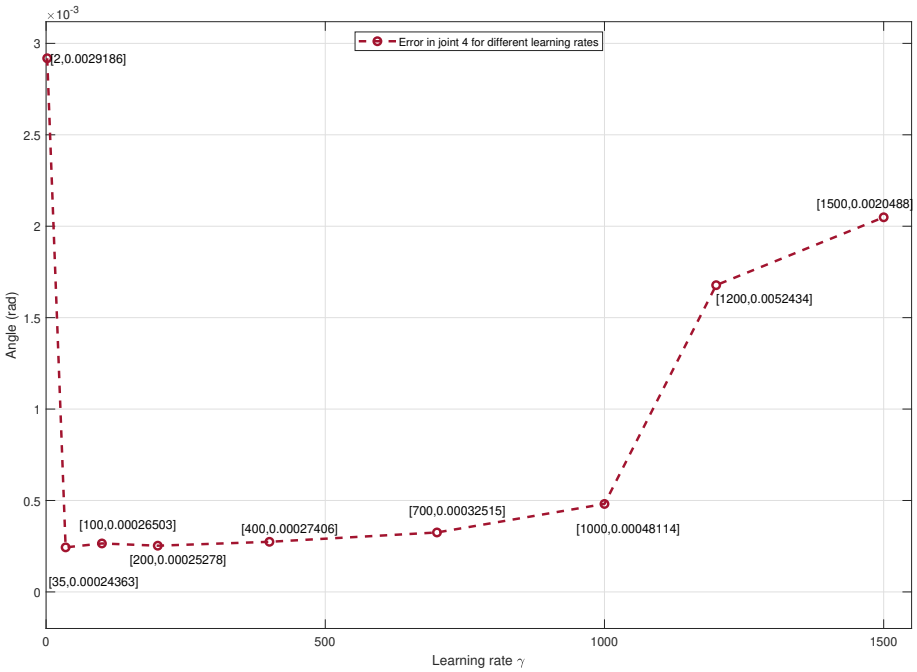


Figure 3.13: Tracking error of different learning rates for joint 4

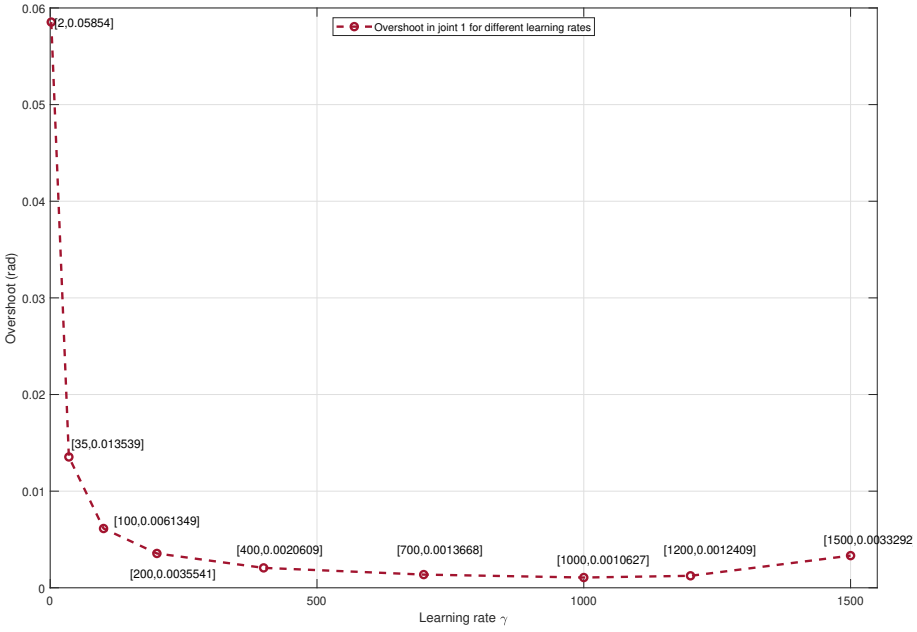


Figure 3.14: Overshoot of different learning rates for joint 1

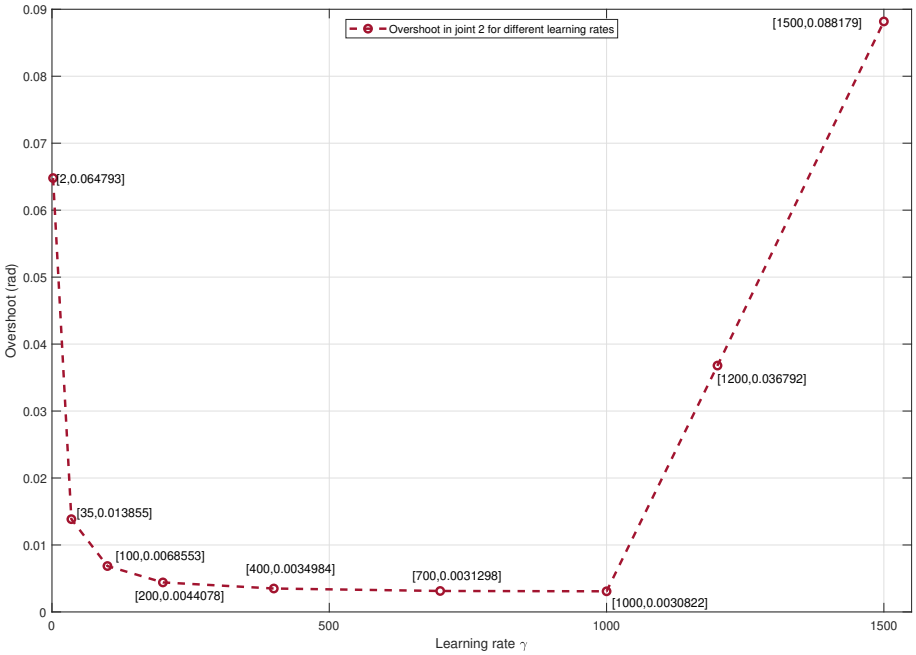


Figure 3.15: Overshoot of different learning rates for joint 2

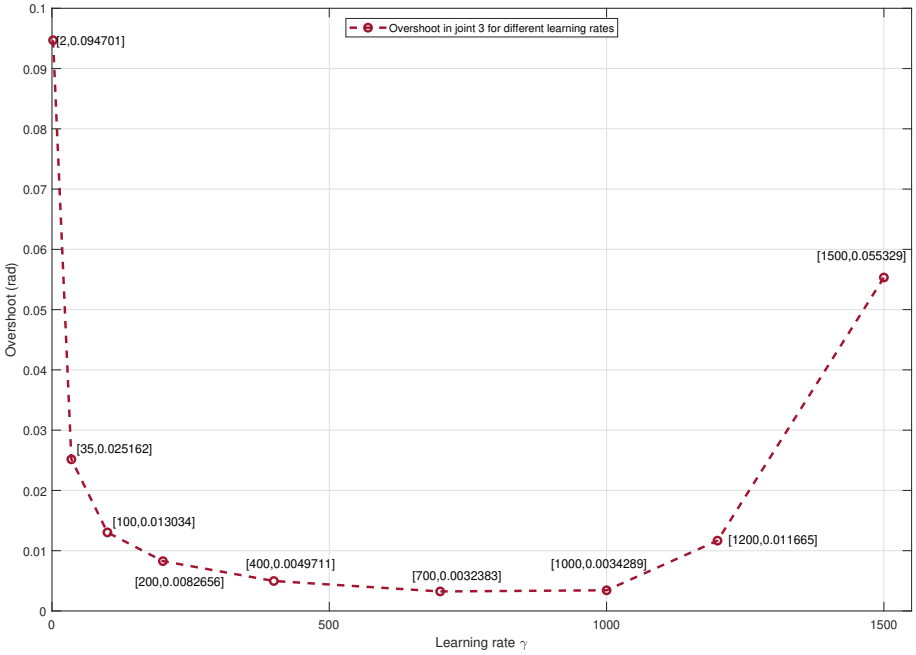


Figure 3.16: Overshoot of different learning rates for joint 3

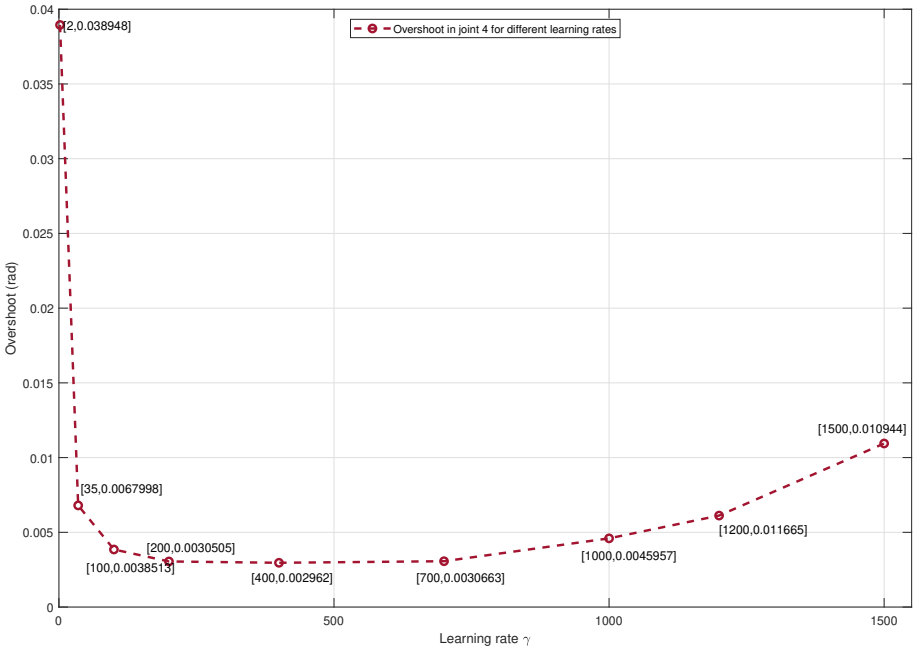


Figure 3.17: Overshoot of different learning rates for joint 4

3.3.2 Comparison with Classical Sliding Mode Control

To compare the robustness of the proposed control against disturbances, we compare the proposed controller with a classical sliding mode control which is given in Eq. (3.34). To do that, a disturbance of 0.2 rad was applied at $t = 4$ s and $t = 5$ s, by adding these signals to the mathematical dynamics model at the corresponding times, which can be seen in the zoomed-in sections of the controller's joints and cartesian tracking errors in Fig. 3.18 to Fig. 3.23. For a fair comparison, the gains of the sliding mode were manually tuned for better control performance.

$$\mathbf{u}_{smc} = \boldsymbol{\eta} sign(\mathbf{s}) + \mathbf{K}\mathbf{s} \tag{3.34}$$

The gains of the sliding mode control were chosen as follows unless otherwise stated:

$$\boldsymbol{\eta} = \begin{bmatrix} 1.2 & 1.25 & 0.005 & 0.002 \end{bmatrix} \tag{3.35}$$

$$\mathbf{K} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.36}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{3.37}$$

In the zoomed-in sections in Fig. 3.18 to Fig. 3.23 it can be observed that the proposed controller is more robust against disturbance in addition to having a smaller overshoot and faster setting time compared to classical sliding mode when subject to the same disturbance conditions.

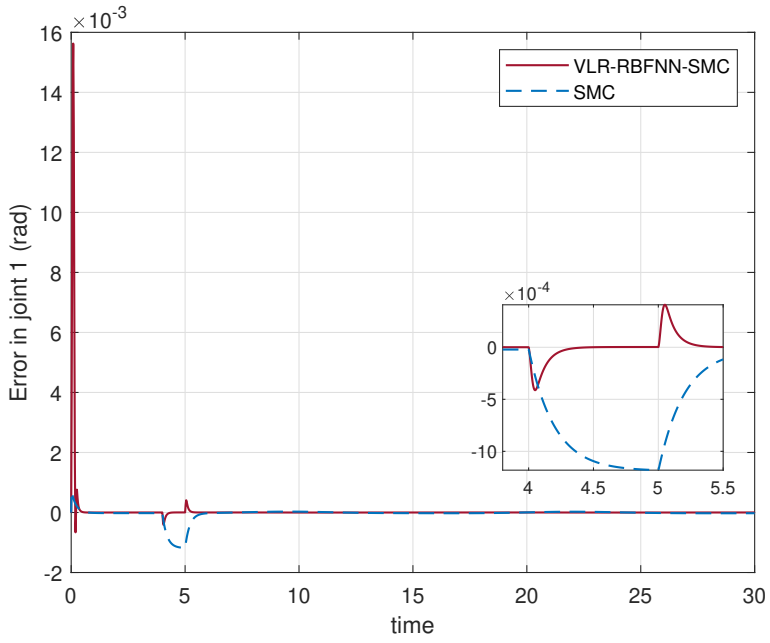


Figure 3.18: Simulation tracking error for joint 1

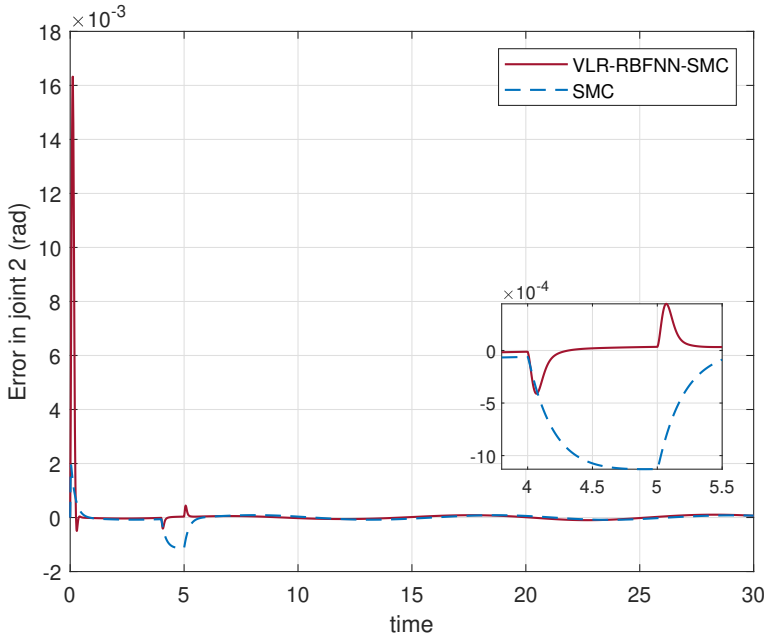


Figure 3.19: Simulation tracking error for joint 2

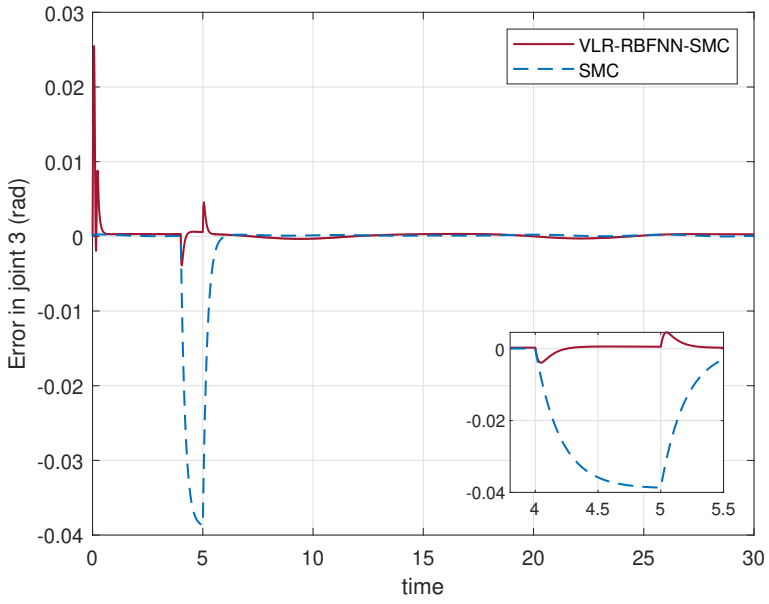


Figure 3.20: Simulation tracking error for joint 3

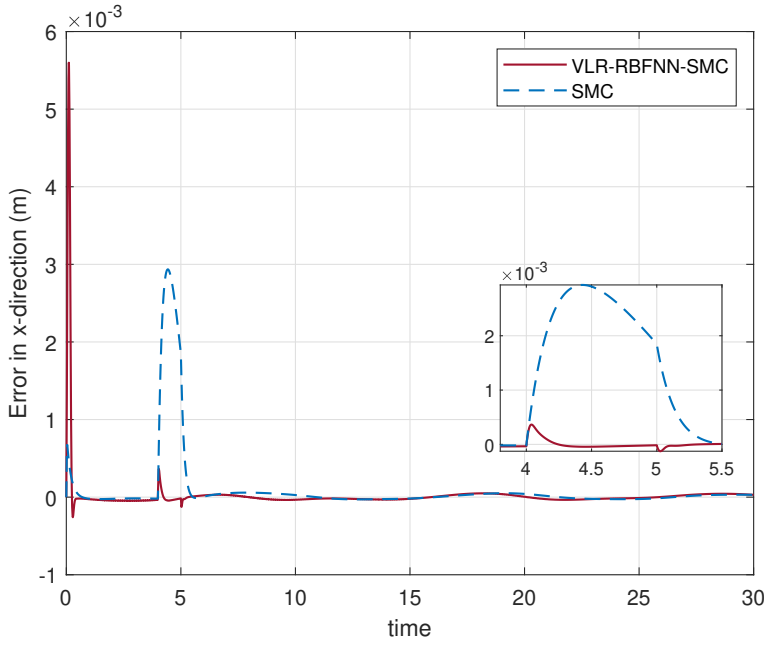


Figure 3.21: Simulation tracking error for x-direction

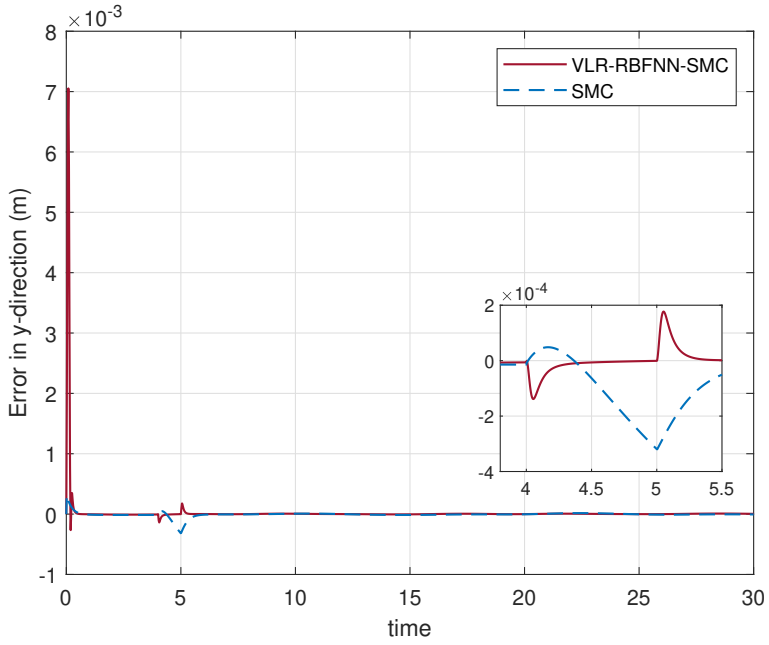


Figure 3.22: Simulation tracking error for y-direction

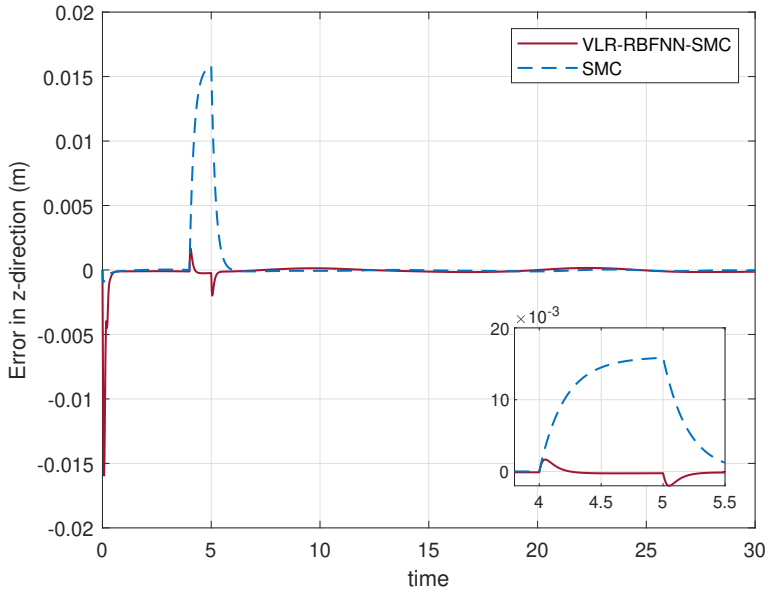


Figure 3.23: Simulation tracking error for z-direction

3.4 Experimental Results

Hardware experiments were conducted using Matlab/Simulink shown in Fig. 3.24 to validate the effectiveness and robustness of our proposed controller experimentally. Comparisons with classical proportional-integral-derivative controller and classical sliding mode control are presented. For these experiments, two different trajectories under three different scenarios were conducted one with no payload, one with a payload of 54.0 g, and the last one with a payload of 102.6 g. The payloads are shown in Fig. 3.25 and Fig. 3.26. Then, a comparison with a back-stepping controller with a high-order observer from the literature [67] is presented. Finally, the controller is integrated with a grasp detection algorithm to realize an autonomous task.

For the RBFNN, the neural network was designed with five nodes in the hidden layer, the variance was chosen as $\mathbf{b} = [0.5, 0.5, 0.5, 0.5, 0.5]$, and the center $\mathbf{c}$ is given as Eq. (3.29). The weight values are updated online by the adaptive law defined in Eq. (3.9) and their initial values

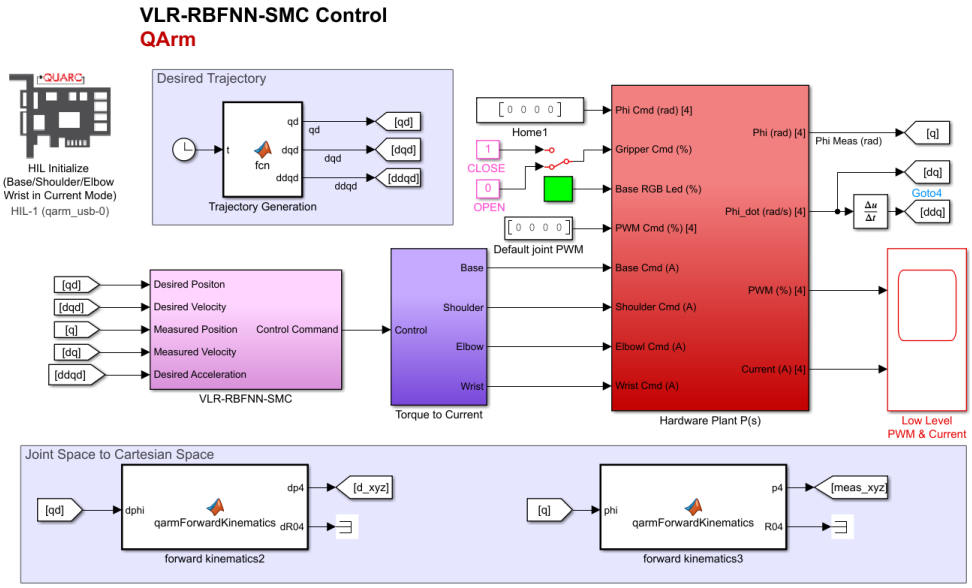


Figure 3.24: Matlab/Simulink model for the control implementation

are selected as $\mathbf{W} = [0.1 \;\; 0.1 \;\; 0.1 \;\; 0.1 \;\; 0.1]^T$.

$$\mathbf{c} = \begin{bmatrix} -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \\ -5 & -2.5 & 0 & 2.5 & 5 \\ -0.1 & -0.05 & 0 & 0.05 & 0.1 \end{bmatrix} \tag{3.38}$$

The gains of the VLR-RBFNN-SMC controller in Eq. 3.15 are chosen as unless otherwise stated:

$$\varepsilon_N = \begin{bmatrix} 0.15 & 0.15 & 0.15 & 0.15 \end{bmatrix} \tag{3.39}$$

$$\varepsilon_D = \begin{bmatrix} 0.35 & 0.35 & 0.35 & 0.35 \end{bmatrix} \tag{3.40}$$

$$\mathbf{K} = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix} \tag{3.41}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.42}$$

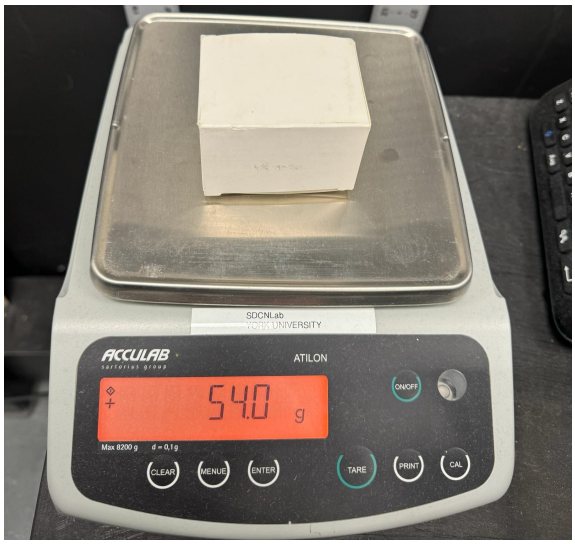


Figure 3.25: 54 g payload

Figure 3.26: 102.6 g payload

3.4.1 Trajectory Tracking 1

Since the ultimate goal is to follow a pick-and-place trajectory, two different trajectories representing such a path were designed to realize the experiments. The first trajectory is generated in the joint space so the joint commands change over time as shown in Eq. (3.44).

$$\theta_d = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^T \tag{3.43}$$

where

$$\theta_1 = 0.2\text{sin}(0.5t)$$

$$\theta_2 = \begin{cases} 0.05t, & 0 < t \leq 10s \\ 0.5, & 10 < t \leq 22s \\ 0.05(-t+32), & 22 < t \leq 32s \\ 0, & t > 32s \end{cases}$$

$$\theta_3 = \begin{cases} 0, & 0 < t \leq 10s \\ 0.05(t-10), & 10 < t \leq 22s \\ 0.6, & 22 < t \leq 32s \\ 0.05(-t+44), & 32 < t \leq 44s \\ 0, & t > 44s \end{cases} \tag{3.44}$$

$$\theta_4 = 0$$

The trajectory tracking for the desired trajectory in the joint space and the cartesian space are given from Fig. 3.27 to Fig. 3.32.

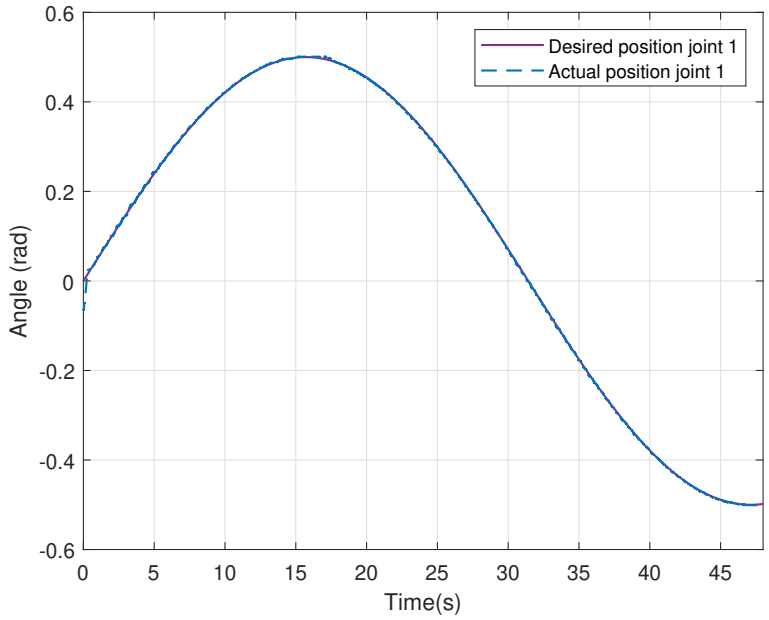


Figure 3.27: VLR-RBFNN-SMC trajectory 1 tracking for joint 1 with no payload

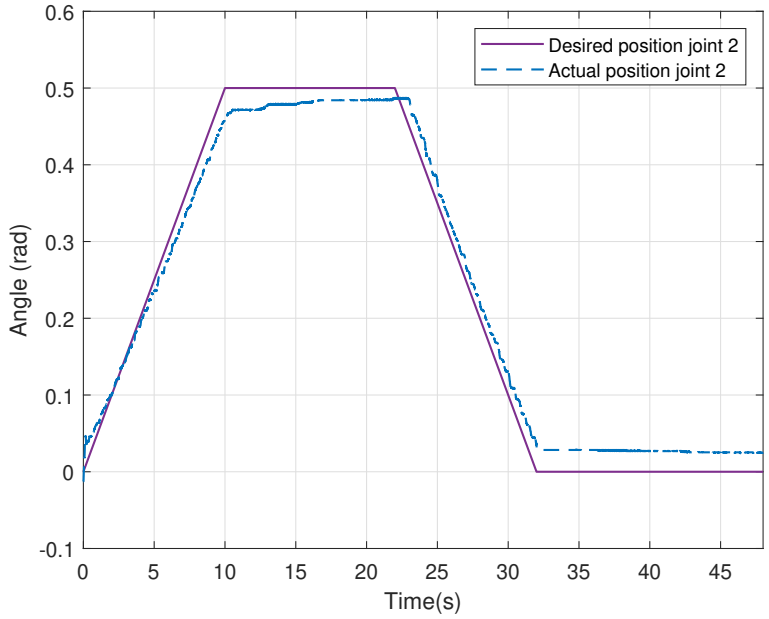


Figure 3.28: VLR-RBFNN-SMC trajectory 1 tracking for joint 2 with no payload

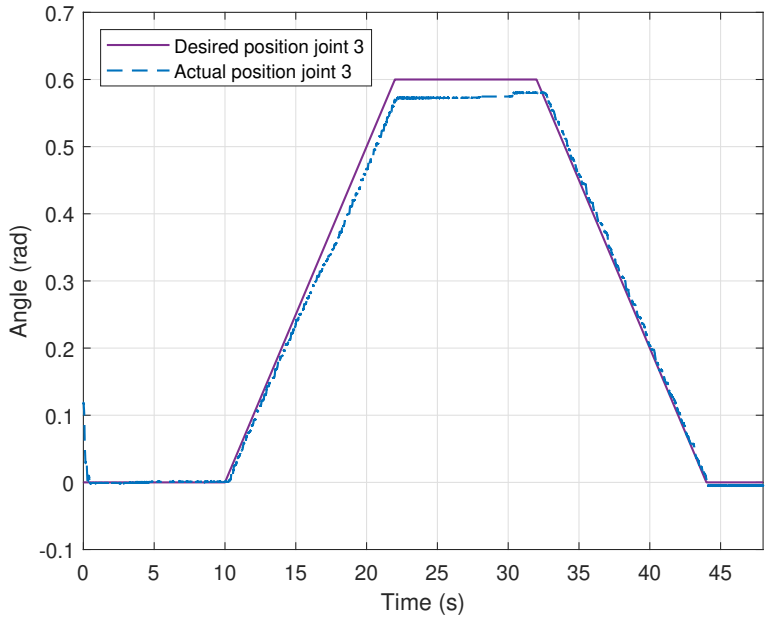


Figure 3.29: VLR-RBFNN-SMC trajectory 1 tracking for joint 3 with no payload

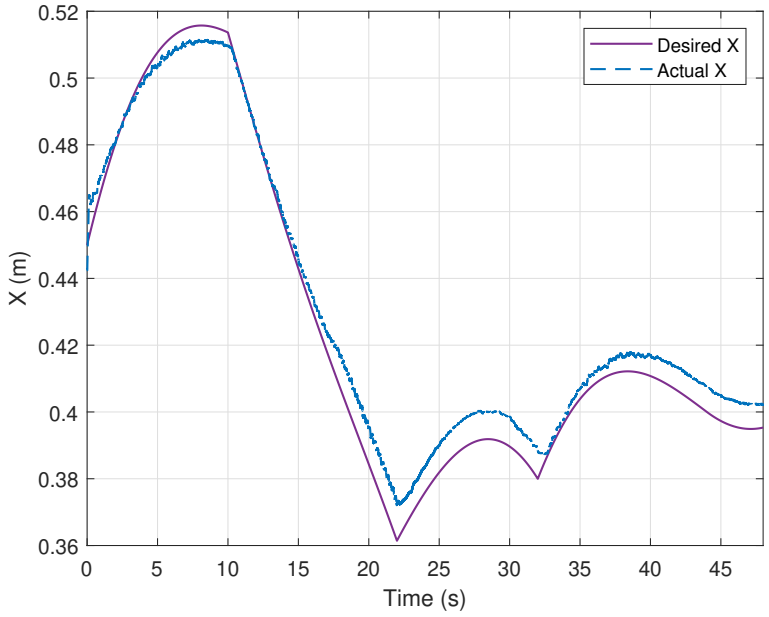


Figure 3.30: VLR-RBFNN-SMC trajectory 1 tracking in x-direction with no payload

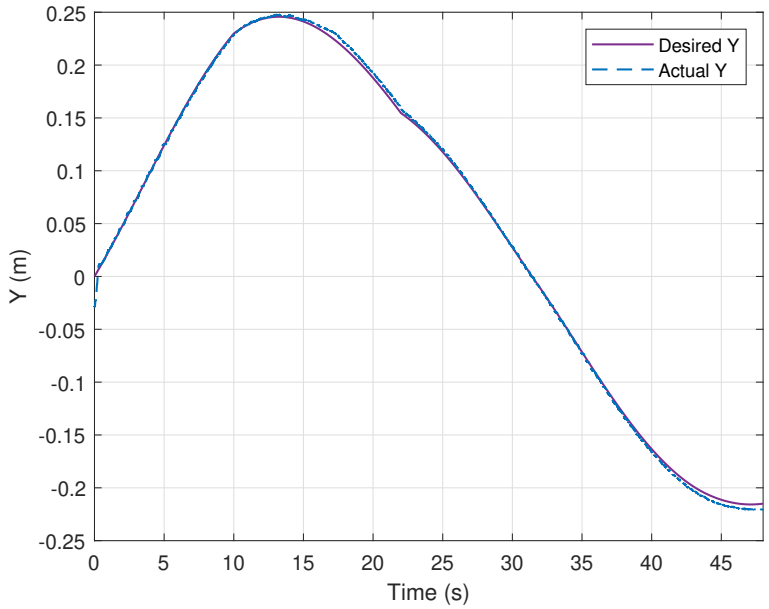


Figure 3.31: VLR-RBFNN-SMC trajectory 1 tracking in y-direction with no payload

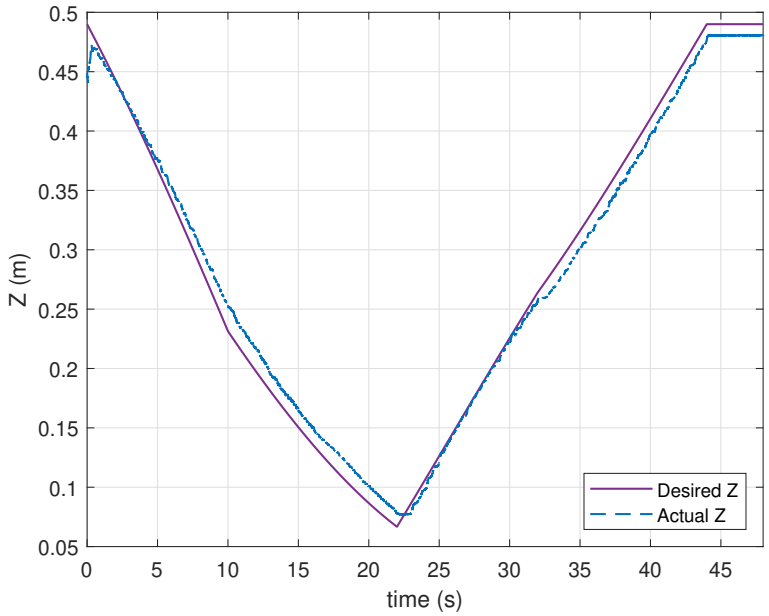


Figure 3.32: VLR-RBFNN-SMC trajectory 1 tracking in z-direction with no payload

3.4.2 Trajectory Tracking 2

Since the goal is to follow a pick-and-place trajectory, two different trajectories representing such a path were designed to realize the experiments. In the second trajectory, the waypoints are chosen in the cartesian space as can be seen in Eq. (3.45), and the trajectory is generated using a cubic polynomial Eq. (3.46) for a smoother trajectory generation and velocity controllability.

$$\text{waypoints} = \begin{bmatrix} 0.45 & 0 & 0.49 \end{bmatrix} \begin{bmatrix} 0 & -0.45 & 0.15 \end{bmatrix} \begin{bmatrix} 0 & 0.45 & 0.20 \end{bmatrix} \tag{3.45}$$

$$\theta_d = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.46}$$

where θ_d are the joints desired positions given as:

$$\theta_d = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{bmatrix}^T \tag{3.47}$$

The trajectory tracking for the desired trajectory in the joint space and the cartesian space are given from Fig. 3.33 to Fig. 3.38.

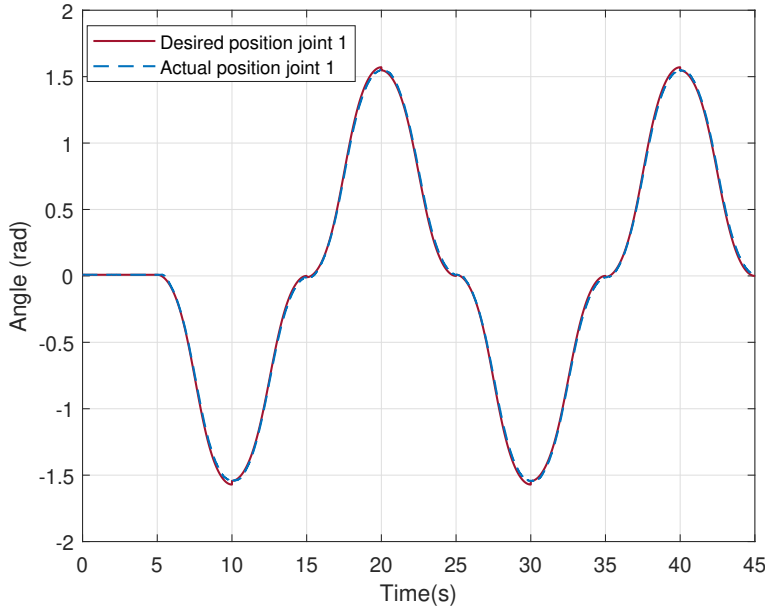


Figure 3.33: VLR-RBFNN-SMC trajectory 2 tracking for joint 1 with no payload

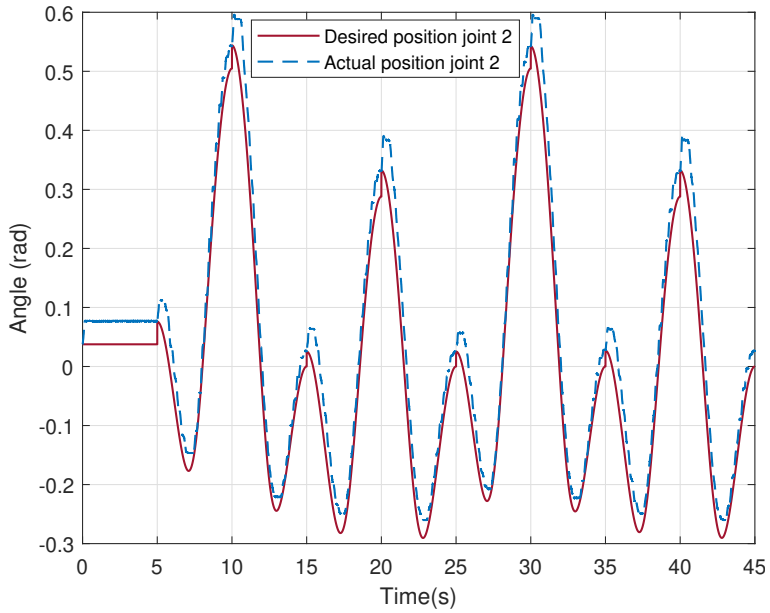


Figure 3.34: VLR-RBFNN-SMC trajectory 2 tracking for joint 2 with no payload

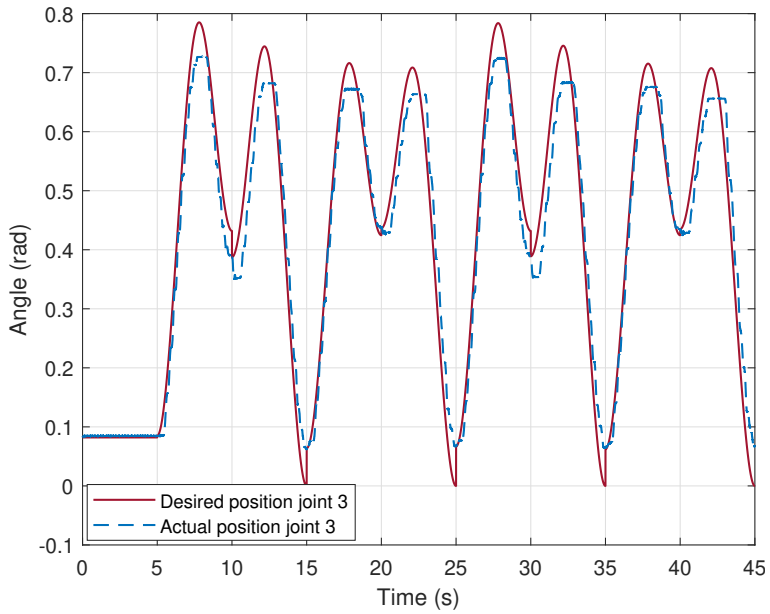


Figure 3.35: VLR-RBFNN-SMC trajectory 2 tracking for joint 3 with no payload

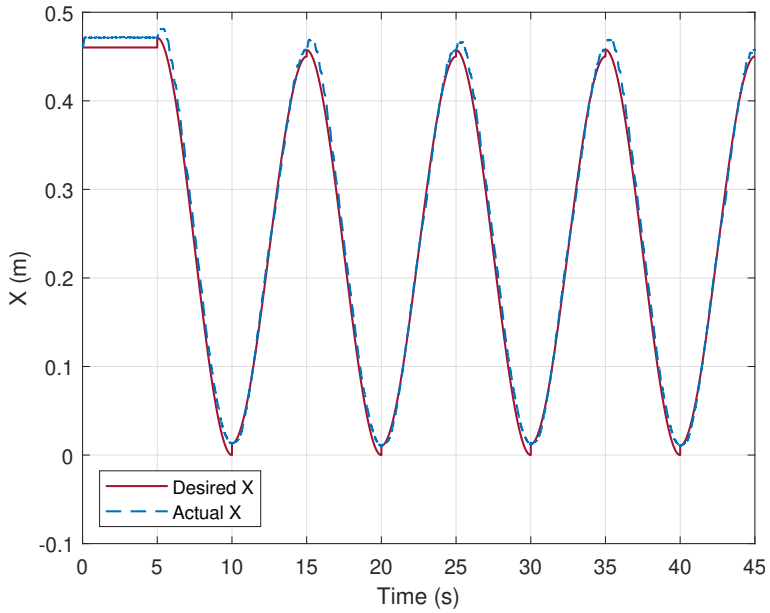


Figure 3.36: VLR-RBFNN-SMC trajectory 2 tracking in x-direction with no payload

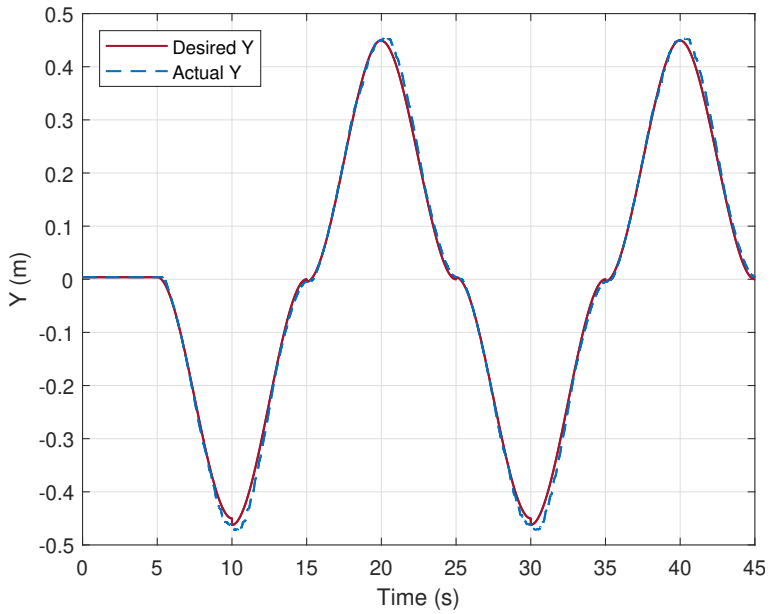


Figure 3.37: VLR-RBFNN-SMC trajectory 2 tracking in y-direction with no payload

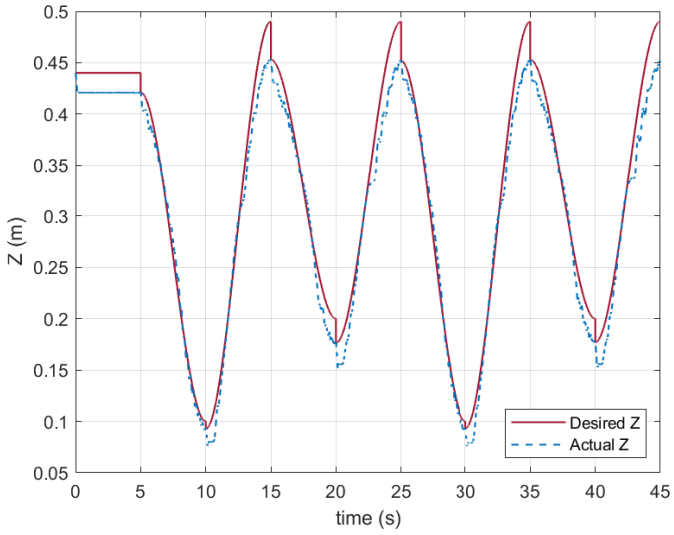


Figure 3.38: VLR-RBFNN-SMC trajectory 2 tracking in z-direction with no payload

3.4.3 Performance comparison of VLR-RBFNN-SCM and RBFNN-SMC for Trajectory 1

To demonstrate the learning behavior of the neural network, we compare the estimation performance of the proposed VLR-RBFNN-SCM with an RBFNN-SMC with a fixed learning rate for different payload variations. Considering the parameters chosen for the Butterworth filter function $r_1 = 35, r_2 = 5, r_3 = 0.5, q = 3$, the variable learning rate can be set as:

$$\gamma = \begin{cases} \dfrac{35}{1 + (||\mathrm{s}||/0.5)^4} + 5, & \mathrm{s} \neq 0 \\ 40, & \mathrm{s} = 0 \end{cases} \tag{3.48}$$

Therefore for a reliable experimental comparison, we set the fixed learning rate (γ_f) as $\gamma_f = max(\gamma) = 40$. Fig. 3.39, Fig. 3.40, Fig. 3.41 show the performance comparison for no payload. Fig. 3.42, Fig. 3.43, Fig. 3.44 show the performance comparison for 54g payload. Fig. 3.45, Fig. 3.46, Fig. 3.47 show the performance comparison for 102.6g payload. The results show the variable learning rate contribution to suppressing the overshoot when estimating the dynamics.

(a) No payload:

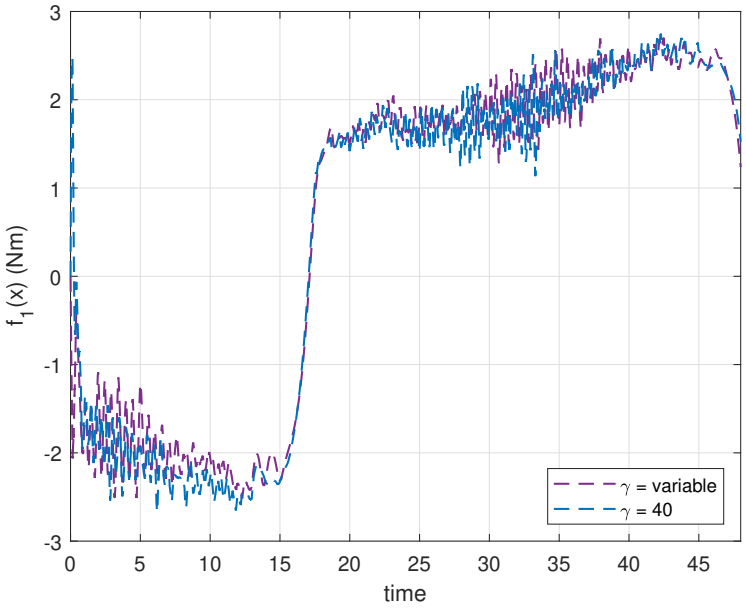


Figure 3.39: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with no payload in trajectory 1

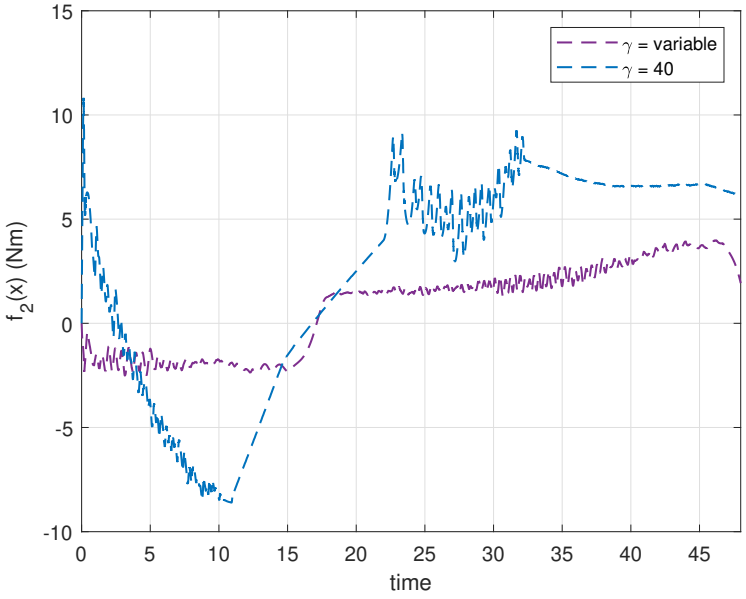


Figure 3.40: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with no payload in trajectory 1

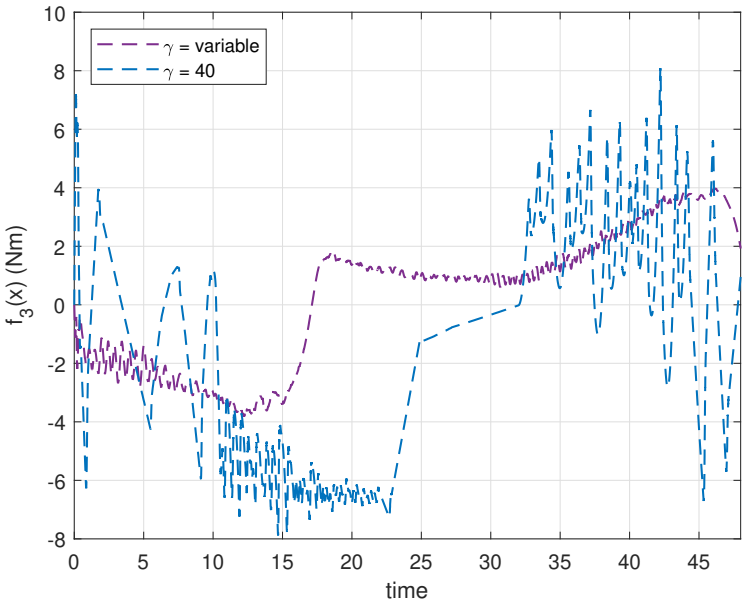


Figure 3.41: Estimation performance comparison between VLR and $\gamma = 40$ for joint 3 with no payload in trajectory 1

(b) 54 **g payload:**

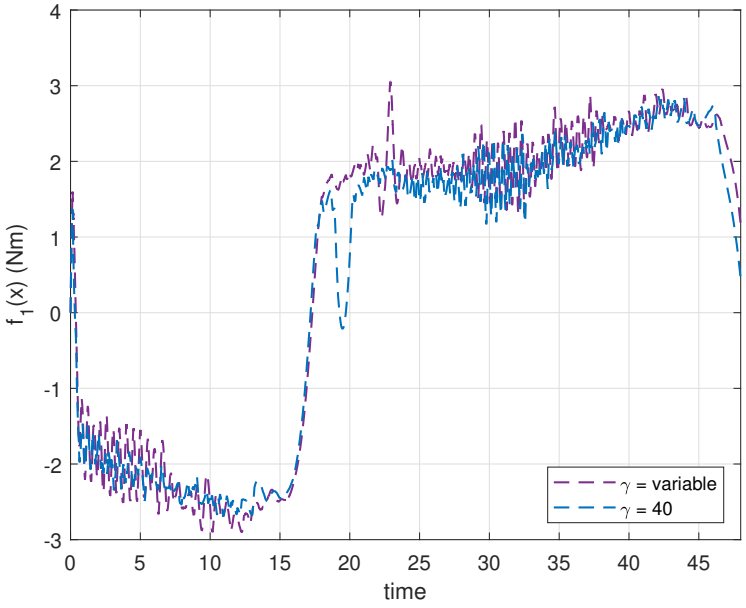


Figure 3.42: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with 54 g payload in trajectory 1

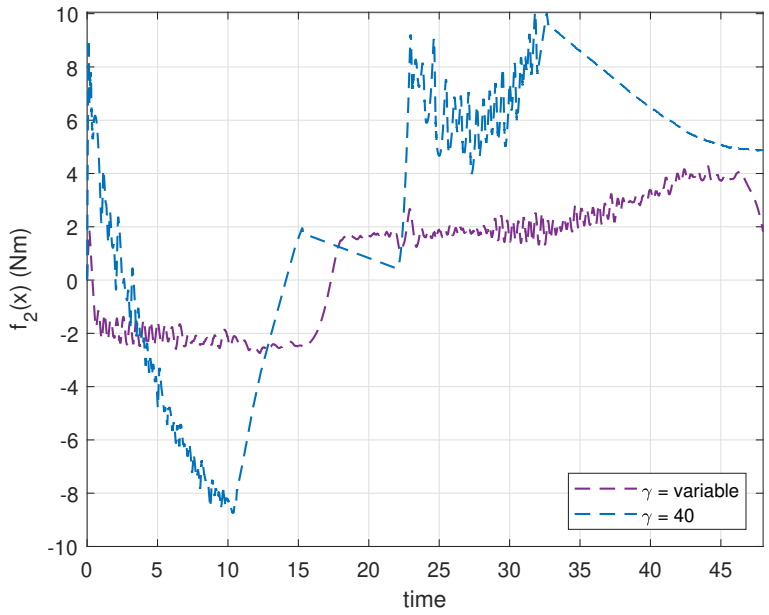


Figure 3.43: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with 54 g payload in trajectory 1

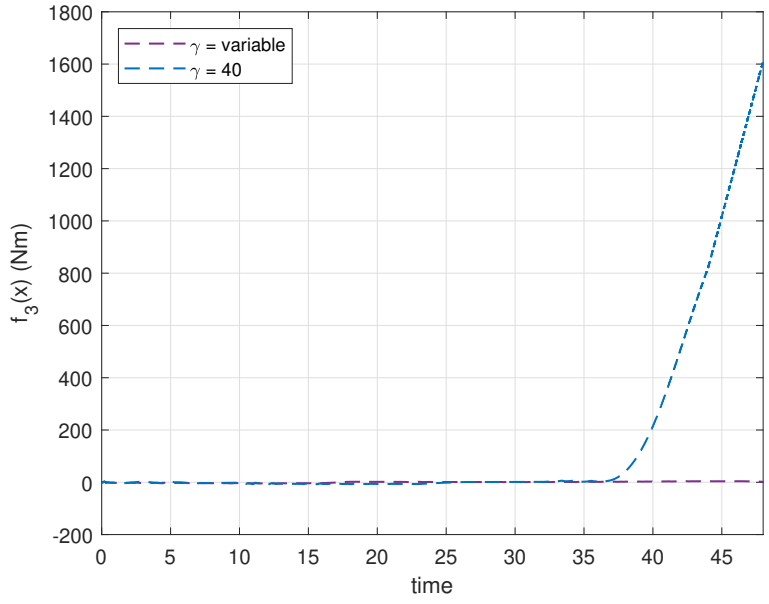


Figure 3.44: Estimation performance comparison between VLR and $\gamma = 40$ for joint 3 with 54 g payload in trajectory 1

(c) 102.6 **g payload:**

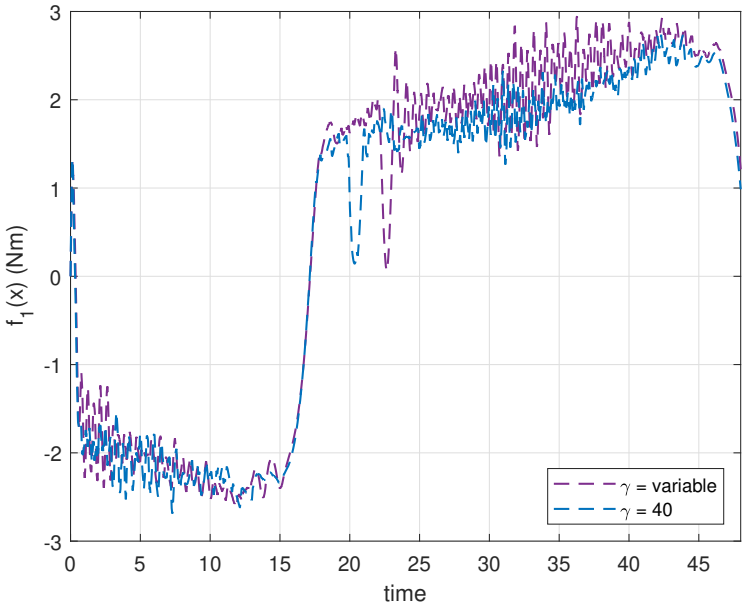


Figure 3.45: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with 102.6 g payload in trajectory 1

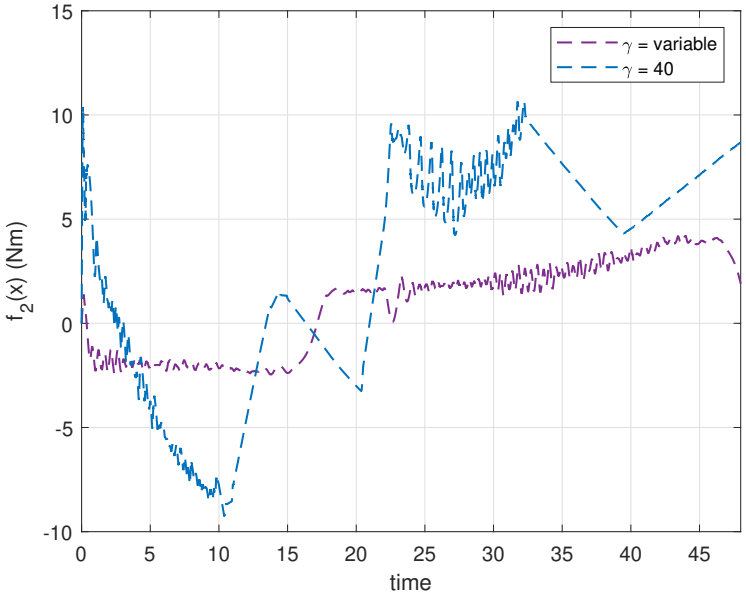


Figure 3.46: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with 102.6 g payload in trajectory 1

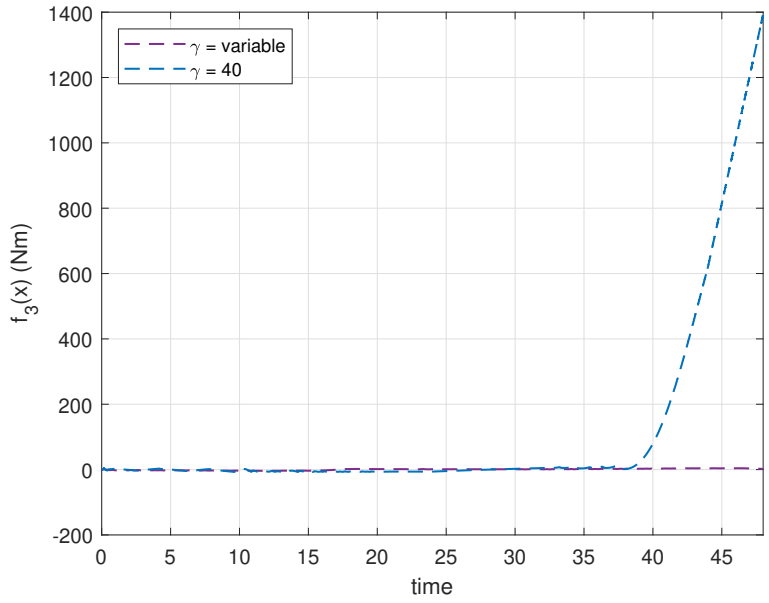


Figure 3.47: Estimation performance comparison between VLR and $\gamma = 40$ for joint 3 with 102.6 g payload in trajectory 1

3.4.4 Performance comparison of VLR-RBFNN-SCM and RBFNN-SMC for Trajectory 2

The experiments realized in subsection 3.4.3 were repeated here for the second trajectory. Fig. 3.48, Fig. 3.49, Fig. 3.50 show the performance comparison for no payload. Fig. 3.51, Fig. 3.52, Fig. 3.53 show the performance comparison for 54 g payload. Fig. 3.54, Fig. 3.55, Fig. 3.56 show the performance comparison for 102.6 g payload. The results show the variable learning rate contribution to suppressing the overshoot when estimating the dynamics.

(a) No payload:

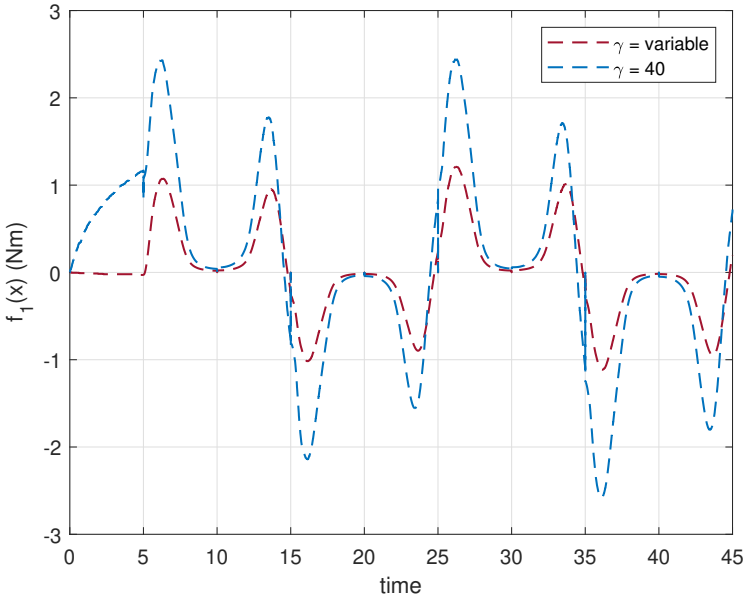


Figure 3.48: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with no payload in trajectory 2

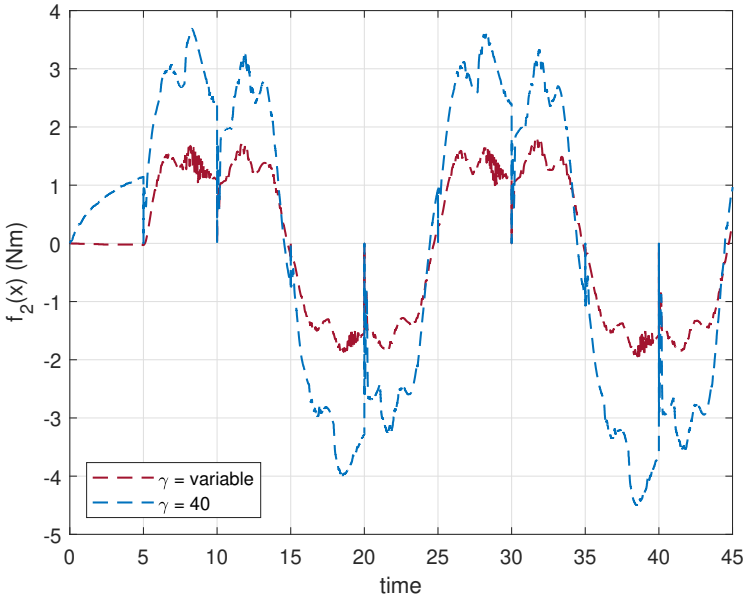


Figure 3.49: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with no payload in trajectory 2

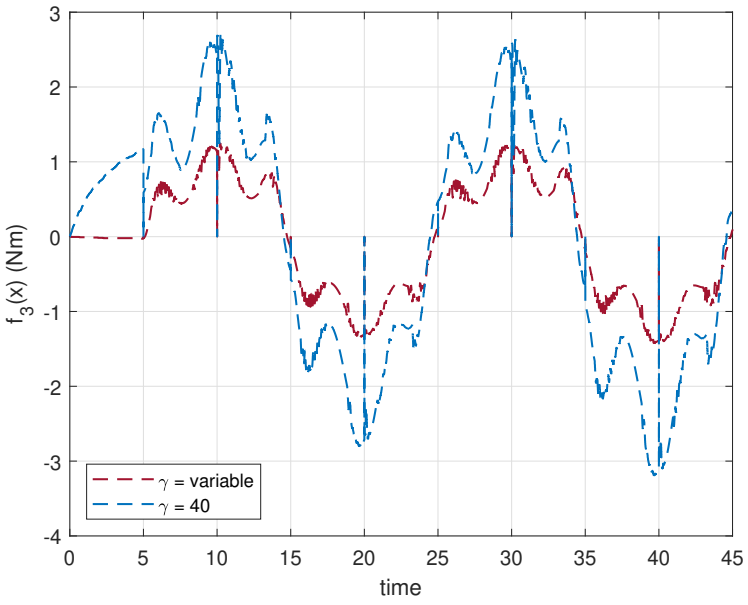


Figure 3.50: Estimation performance comparison between VLR and $\gamma = 40$ for joint 3 with no payload in trajectory 2

(b) 54 **g payload:**

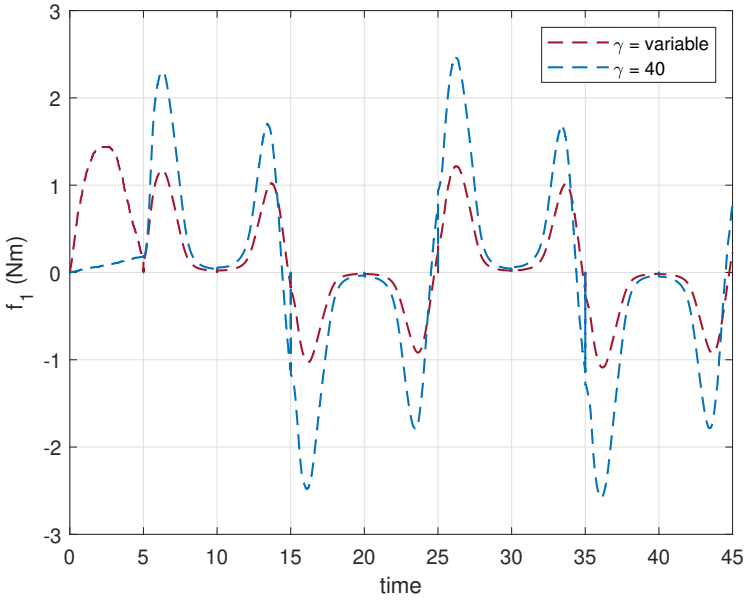


Figure 3.51: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with 54 g payload in trajectory 2

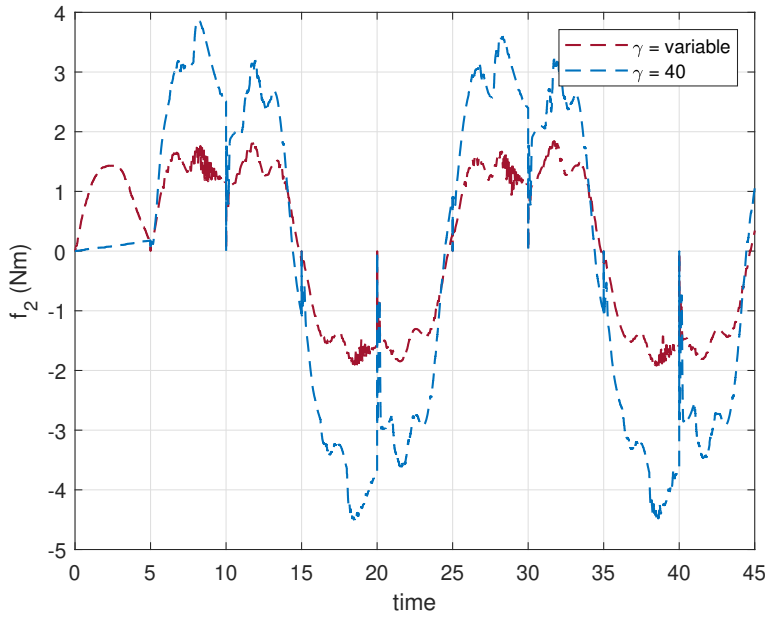


Figure 3.52: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with 54 g payload in trajectory 2

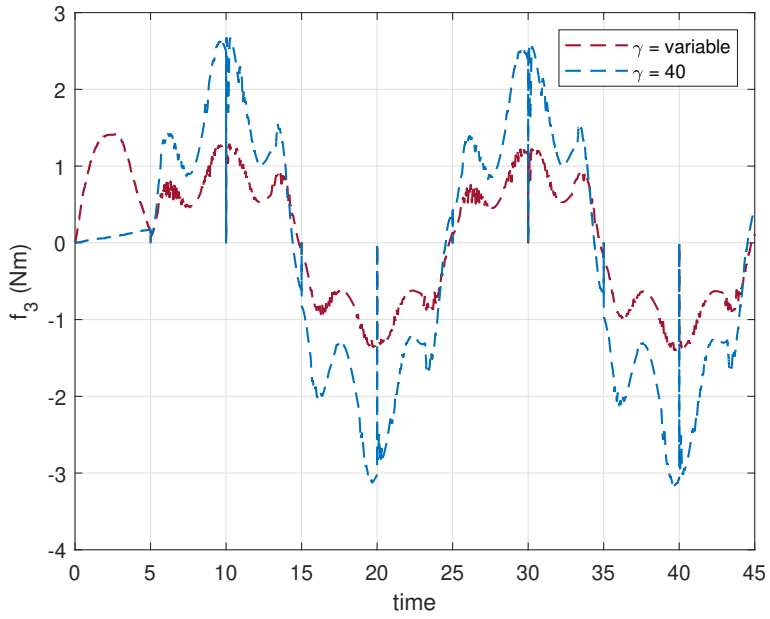


Figure 3.53: Estimation performance comparison between VLR and $\gamma = 40$ for joint 3 with 54 g payload in trajectory 2

(c) 102.6 **g payload:**

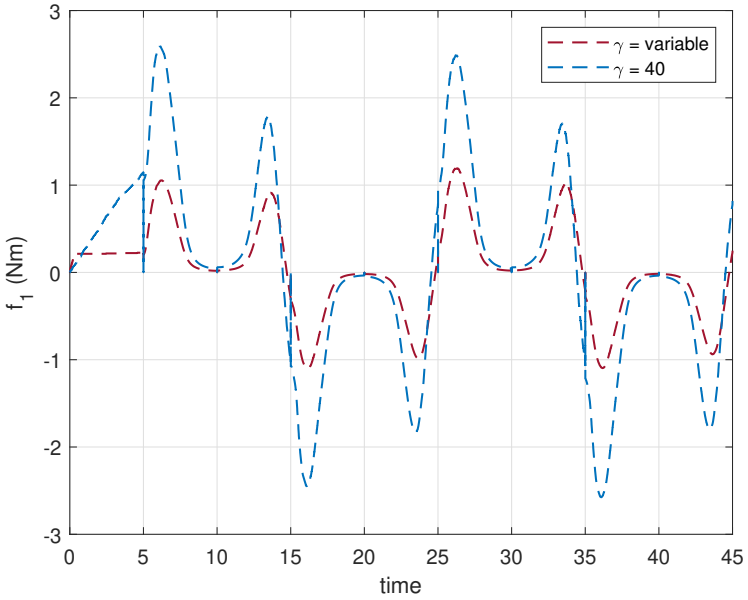


Figure 3.54: Estimation performance comparison between VLR and $\gamma = 40$ for joint 1 with 102.6 g payload in trajectory 2

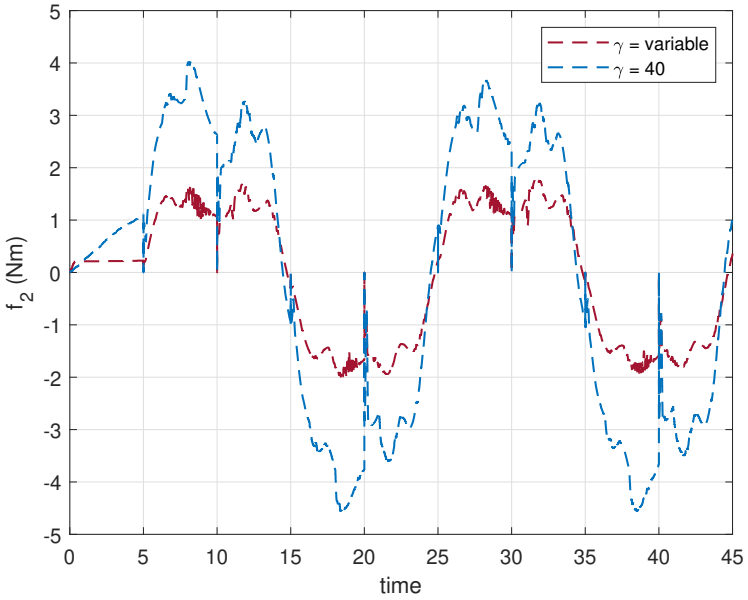


Figure 3.55: Estimation performance comparison between VLR and $\gamma = 40$ for joint 2 with 102.6 g payload in trajectory 2

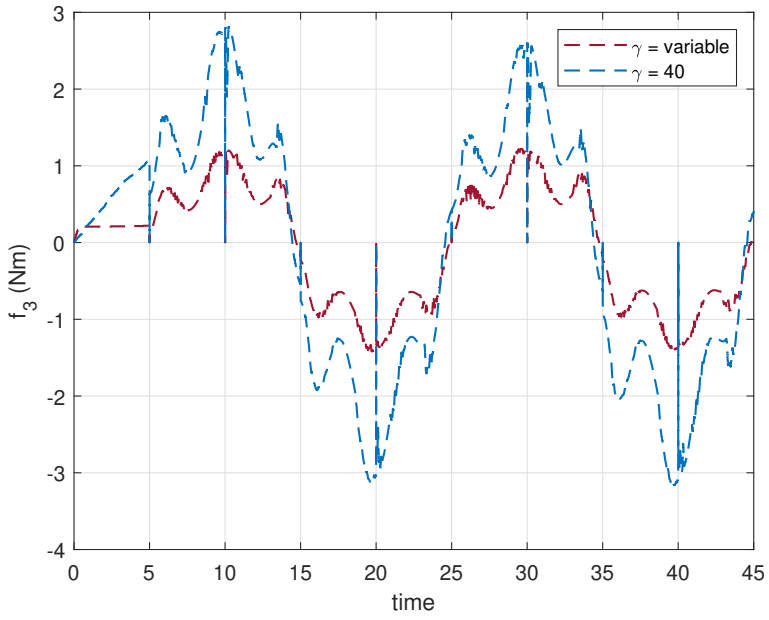


Figure 3.56: Estimation performance comparison between VLR and γ = 40 for joint 3 with 102.6 g payload in trajectory 2

3.4.5 Comparison with PID and SMC for Trajectory 1

To validate the robustness of our proposed control under different situations, we compare the proposed controller with classical PID control, given in Eq. (3.49).

$$\begin{aligned} \mathbf{u_{pid}} &= \mathbf{K_p e} + \mathbf{K_d}\dot{\mathbf{e}} + \mathbf{K_i}\int \mathbf{e(t)}\, dt \\ \mathbf{u_{pid}} &= \mathbf{K_p s} + \mathbf{K_i}\int \mathbf{e(t)}\, dt \end{aligned} \tag{3.49}$$

where s is defined in Eq. (3.14). Therefore,

$$\mathbf{K_p} = \begin{bmatrix} 20 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix} \tag{3.50}$$

$$\mathbf{K_i} = \begin{bmatrix} 0.003 & 0 & 0 & 0 \\ 0 & 0.003 & 0 & 0 \\ 0 & 0 & 0.003 & 0 \\ 0 & 0 & 0 & 0.003 \end{bmatrix} \tag{3.51}$$

$$\boldsymbol{\lambda}_{pid} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.52}$$

$$\mathbf{K_d} = \mathbf{K_p}\boldsymbol{\lambda}_{pid} \tag{3.53}$$

We also compare our proposed controller with classical sliding mode control. Its design is given as follows:

$$\mathbf{K_{smc}} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{3.54}$$

$$\boldsymbol{\eta}_{smc} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{3.55}$$

$$\boldsymbol{\lambda}_{smc} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.56}$$

Three experiments for each controller with payload variation were conducted, one with no payload, one with a 54g payload, and another with a 102.6g payload.

The controller's tracking errors for the VLR-RBFNN-SMC and the classical PID are plotted against one another. Fig. 3.57, Fig. 3.58, Fig. 3.59 show the tracking errors with no payload. Fig. 3.63, Fig. 3.64, Fig. 3.65 show the tracking errors for a 54 g payload. Fig. 3.69, Fig. 3.70, Fig. 3.71 show the tracking errors for a 102.6 g payload. For a fair comparison, the gains of the PID controller were manually tuned for a better control performance. It can be observed that the proposed controller is more robust against payload variation given that we have smaller tracking errors compared to classical PID controller when submitted to the same conditions.

The controller's tracking errors for the VLR-RBFNN-SMC and the classical SMC are plotted against one another. Fig. 3.18, Fig. 3.19, Fig. 3.20 show the tracking errors with no payload.

Fig. 3.66, Fig. 3.67, Fig. 3.68 show the tracking errors for a 54 g payload. Fig. 3.72, Fig. 3.73, Fig. 3.74 show the tracking errors for a 102.6 g payload. For a fair comparison, the gains of the sliding mode were manually tuned for a better control performance. It can be observed that the proposed controller is more robust against payload variation given that we have smaller tracking errors compared to classical sliding mode when submitted to the same conditions.

(a) No payload:

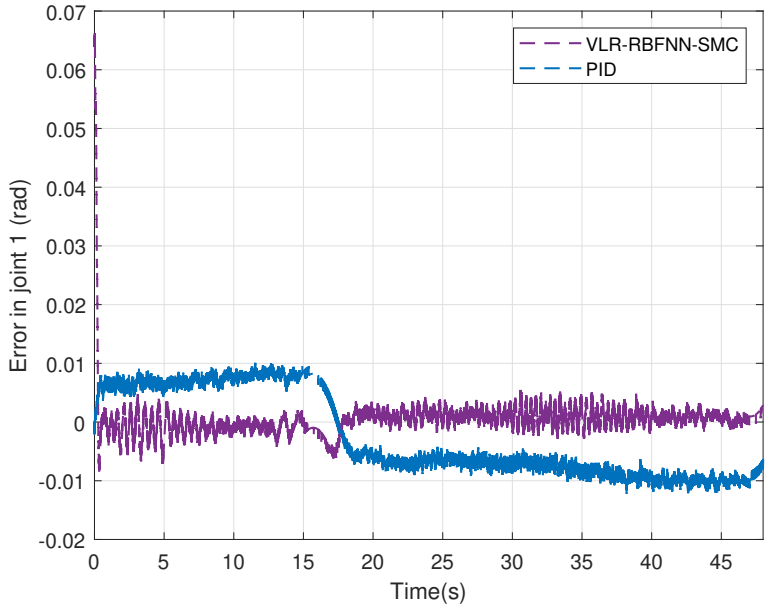


Figure 3.57: Trajectory 1 tracking error for joint 1 with no payload

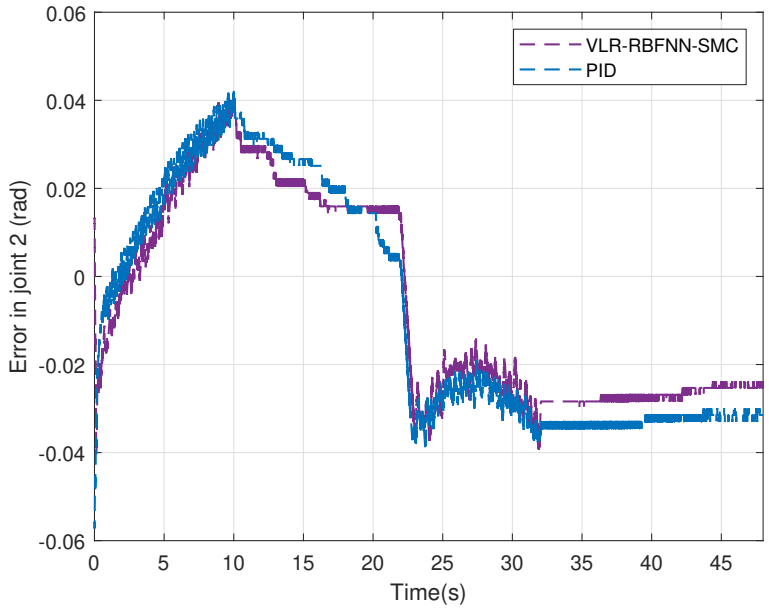


Figure 3.58: Trajectory 1 tracking error for joint 2 with no payload

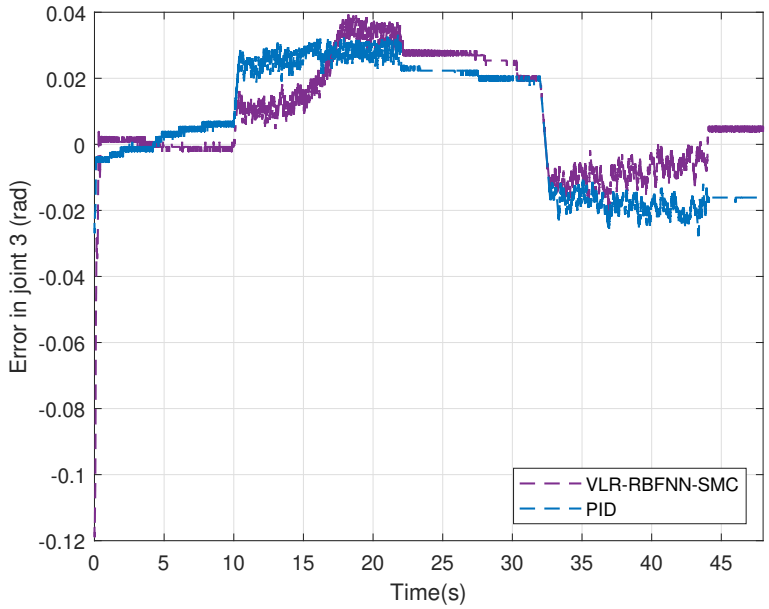


Figure 3.59: Trajectory 1 tracking error for joint 3 with no payload

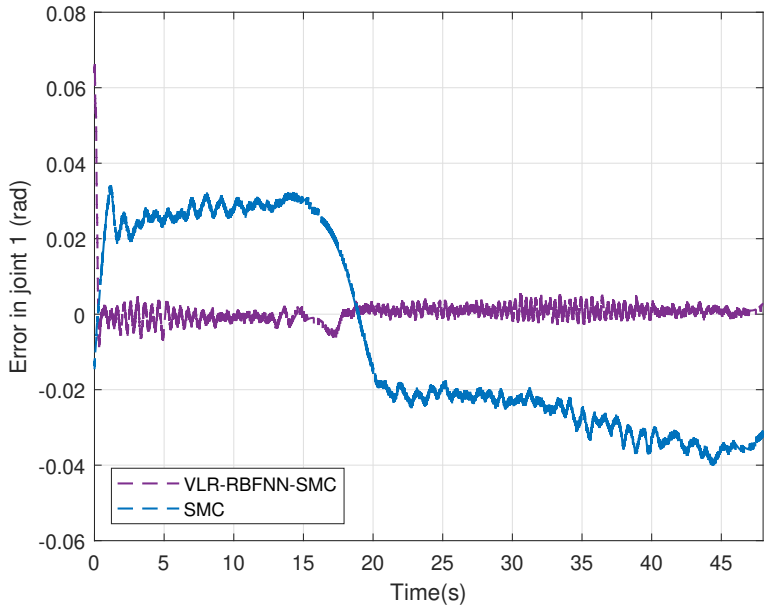


Figure 3.60: Trajectory 1 tracking error for joint 1 with no payload

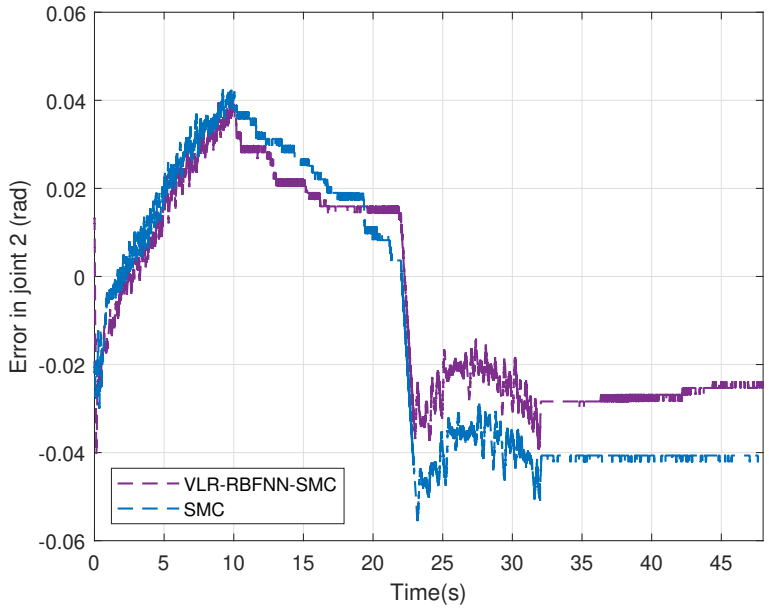


Figure 3.61: Trajectory 1 tracking error for joint 2 with no payload

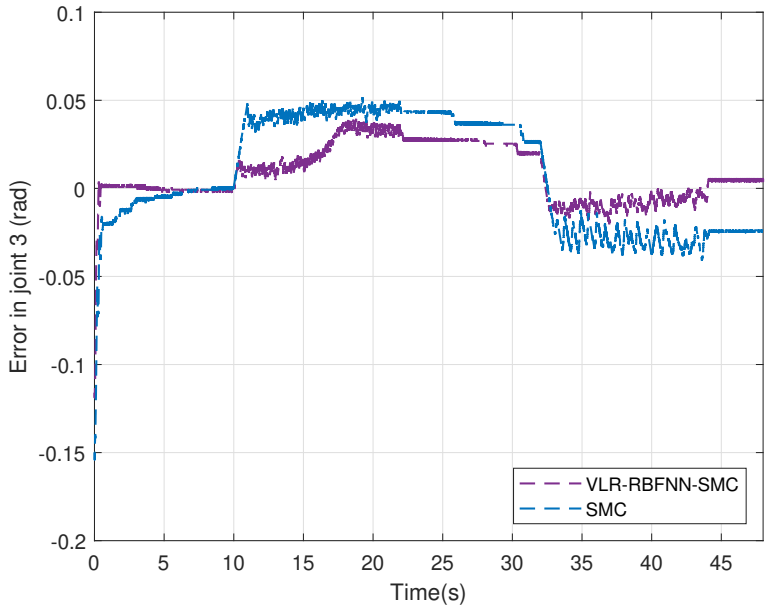


Figure 3.62: Trajectory 1 tracking error for joint 3 with no payload

Table 3.1, shows the absolute mean and maximum error of the tracking from the first trajectory of the controllers. We can see that the proposed method can outperform PID and SMC when carrying no payload. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.1: Comparison of experimental errors for trajectory 1 with no payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	**0.0017**	0.0662	0.0075	**0.0122**	0.0260	0.0401
$\lvert e_{\theta_2} \rvert$ (rad)	**0.0229**	**0.0412**	0.0264	0.0572	0.0313	0.0559
$\lvert e_{\theta_3} \rvert$ (rad)	**0.0138**	0.1189	0.0176	**0.0333**	0.0290	0.1542
$\lvert e_x \rvert$ (m)	**0.0056**	**0.0130**	0.0060	0.0185	0.0097	0.0201
$\lvert e_y \rvert$ (m)	**0.0023**	0.0295	0.0032	**0.0078**	0.0108	0.0185
$\lvert e_z \rvert$ (m)	**0.0113**	0.0480	0.0152	**0.0370**	0.0188	0.0718

(b) 54 **g payload:**

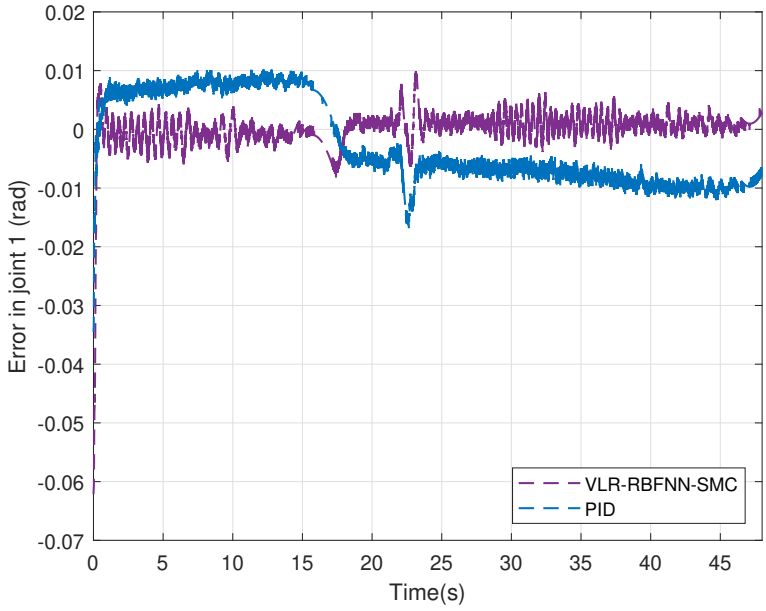


Figure 3.63: Trajectory 1 tracking error for joint 1 with 54 g payload

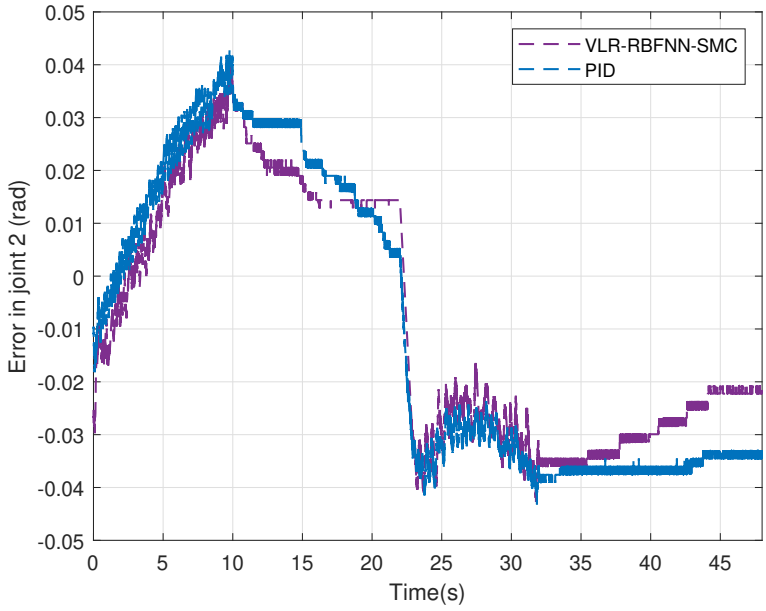


Figure 3.64: Trajectory 1 tracking error for joint 2 with 54 g payload

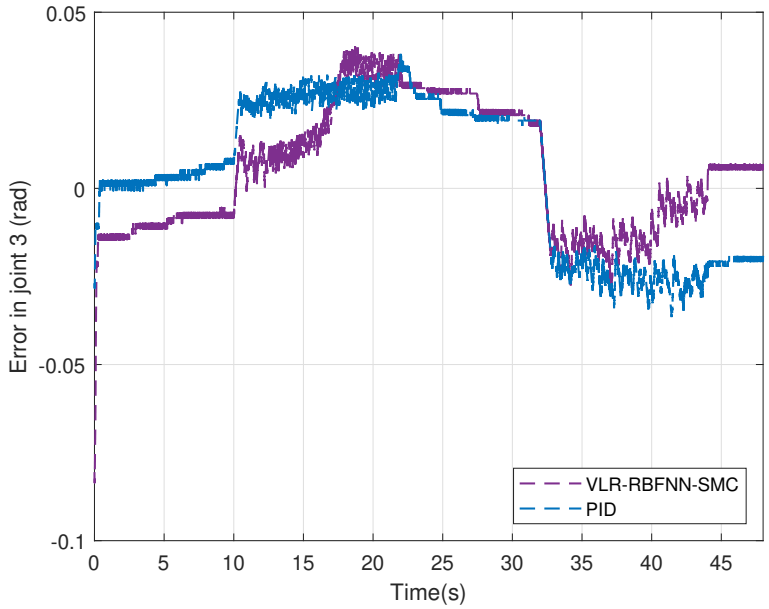


Figure 3.65: Trajectory 1 tracking error for joint 3 with 54 g payload

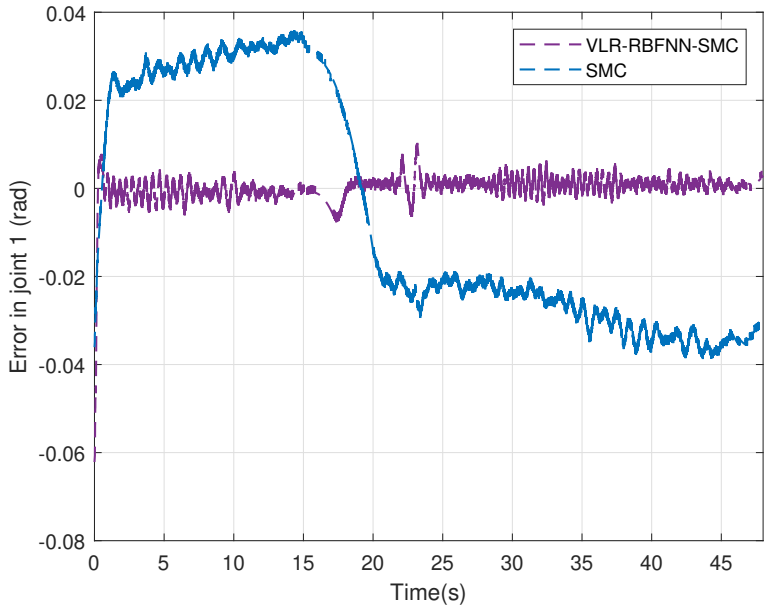


Figure 3.66: Trajectory 1 tracking error for joint 1 with 54 g payload

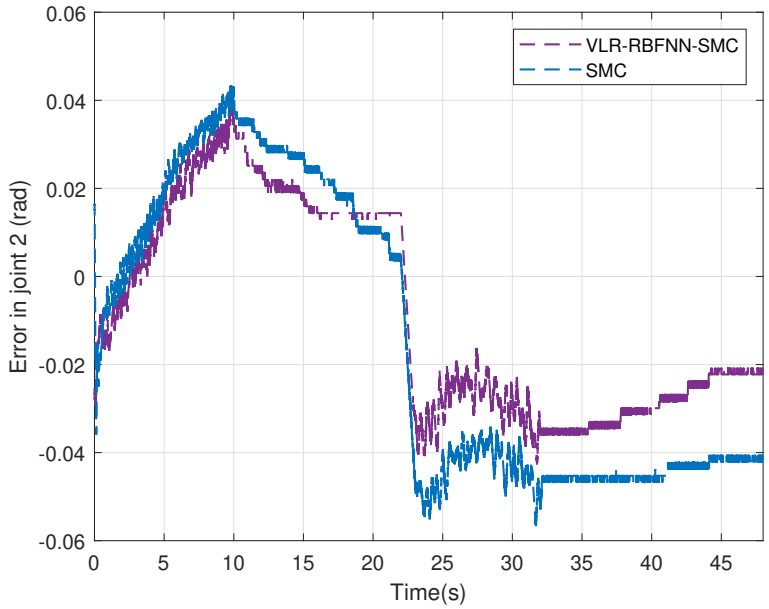


Figure 3.67: Trajectory 1 tracking error for joint 2 with 54 g payload

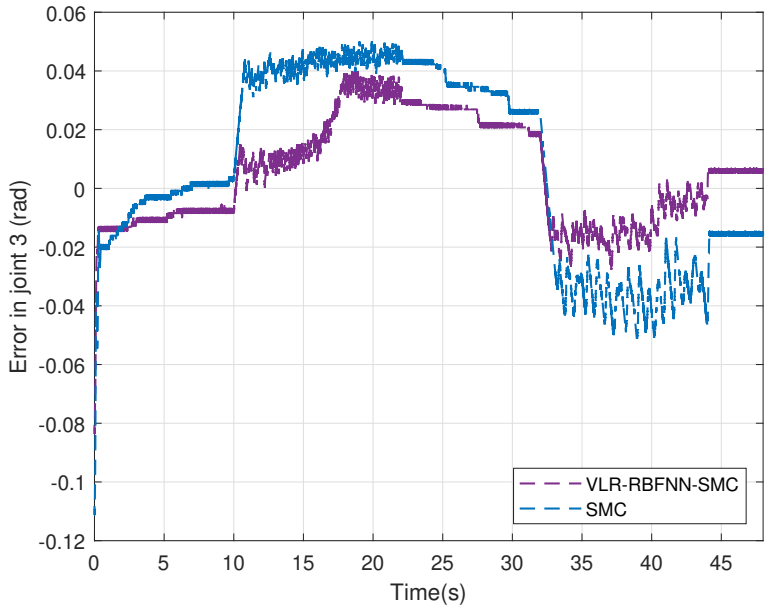


Figure 3.68: Trajectory 1 tracking error for joint 3 with 54 g payload

Table 3.2, shows the absolute mean and maximum error of the tracking from the first tra-

jectory of the controllers. We can see that the proposed method can outperform PID and SMC when carrying a payload of 54 g. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.2: Comparison of experimental errors for trajectory 1 with 54.0 g payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	**0.0018**	0.0621	0.0075	**0.0345**	0.0269	0.0388
$\lvert e_{\theta_2} \rvert$ (rad)	**0.0235**	**0.0427**	0.0276	0.0432	0.0369	0.0568
$\lvert e_{\theta_3} \rvert$ (rad)	**0.0159**	0.0836	0.0198	**0.0384**	0.0288	0.1112
$\lvert e_x \rvert$ (m)	**0.0055**	0.0135	0.0061	**0.0127**	0.0097	0.0204
$\lvert e_y \rvert$ (m)	**0.0023**	0.0284	0.0033	**0.0156**	0.0111	0.0192
$\lvert e_z \rvert$ (m)	**0.0116**	0.0461	0.0165	**0.0312**	0.0200	0.0425

(c) 102 g payload:

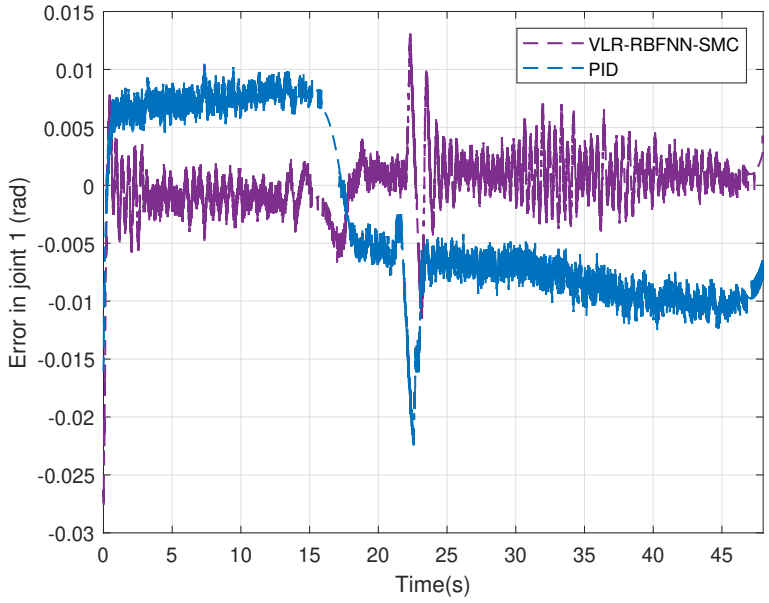


Figure 3.69: Trajectory 1 tracking error for joint 1 with 102.6 g payload

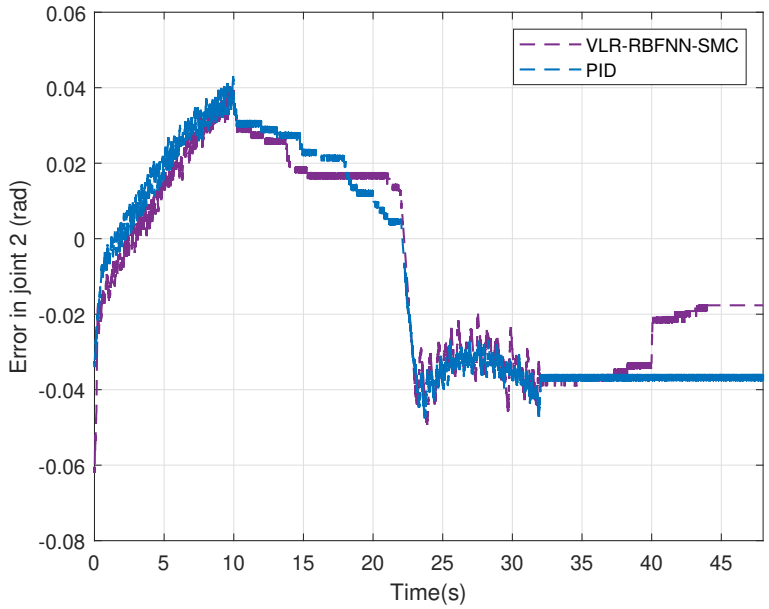


Figure 3.70: Trajectory 1 tracking error for joint 2 with 102.6 g payload

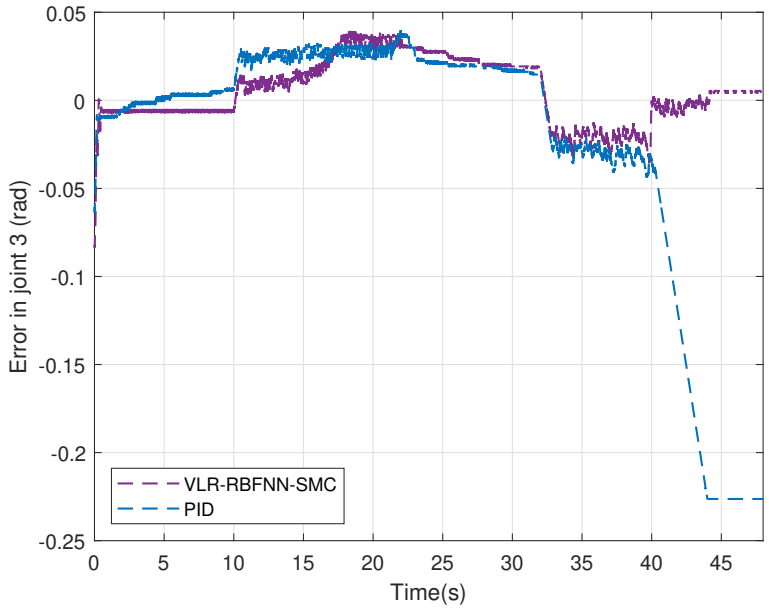


Figure 3.71: Trajectory 1 tracking error for joint 3 with 102.6 g payload

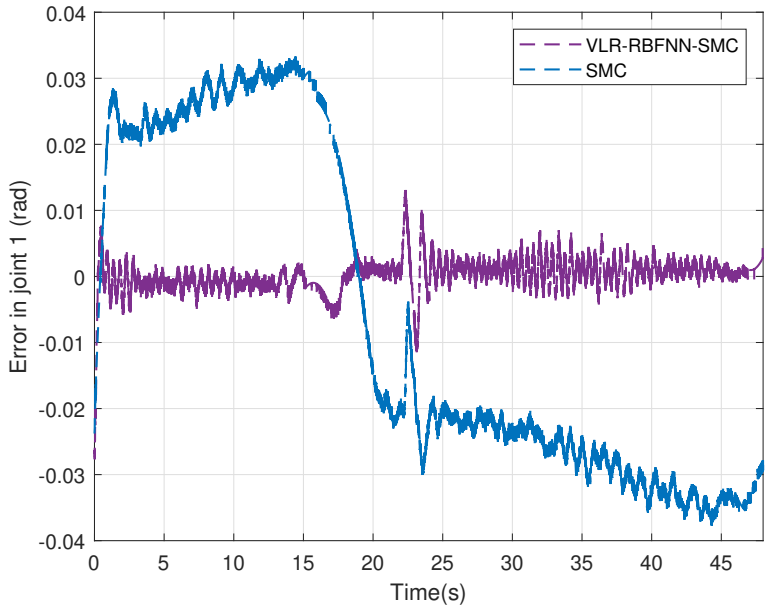


Figure 3.72: Trajectory 1 tracking error for joint 1 with 102.6 g payload

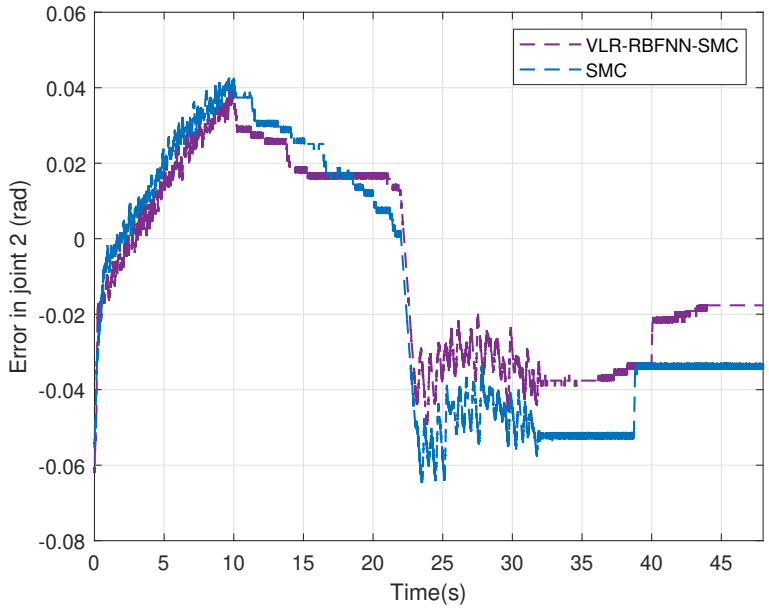


Figure 3.73: Trajectory 1 tracking error for joint 2 with 102.6 g payload

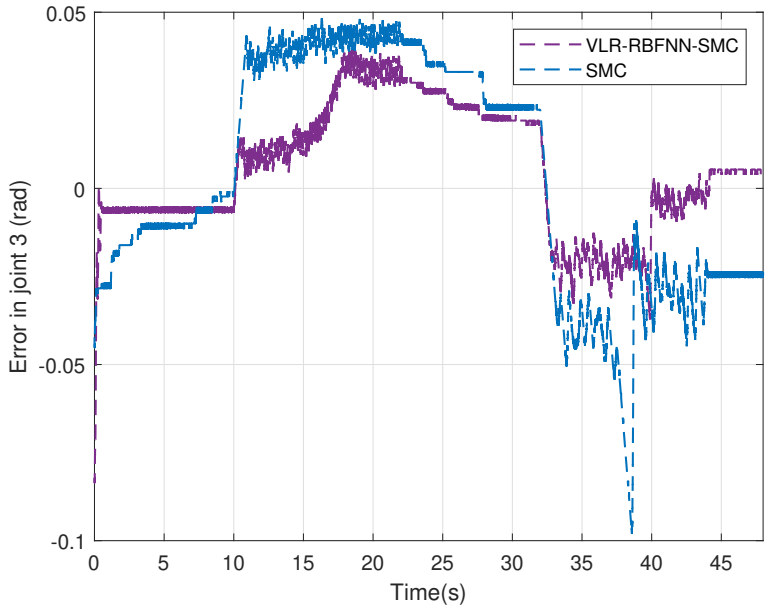


Figure 3.74: Trajectory 1 tracking error for joint 3 with 102.6 g payload

Table 3.3, shows the absolute mean and maximum error of the tracking from the first trajectory of the controllers. We can see that the proposed method can outperform PID and SMC when carrying a payload of 102.6 g. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.3: Comparison of experimental errors for trajectory 1 with 102.6 g payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	**0.0018**	0.0277	0.0078	**0.0226**	0.0255	0.0378
$\lvert e_{\theta_2} \rvert$ (rad)	**0.0246**	0.0620	0.0286	**0.0484**	0.0335	0.0649
$\lvert e_{\theta_3} \rvert$ (rad)	**0.0157**	**0.0836**	0.0460	0.2263	0.0297	0.0980
$\lvert e_x \rvert$ (m)	0.0053	**0.0173**	**0.0045**	0.1065	0.0088	0.0181
$\lvert e_y \rvert$ (m)	**0.0021**	0.0130	0.0039	**0.0128**	0.0107	0.0185
$\lvert e_z \rvert$ (m)	**0.0123**	0.0617	0.0274	**0.0116**	0.0203	0.0598

3.4.6 Comparison with PID and SMC for Trajectory 2

The experiments conducted in subsection 3.4.5 are repeated here for the second trajectory. For a fair comparison, the gains of the VLR-RBFNN-SMC, the PID controller, and the sliding mode control were manually tuned for a better control performance and are given as follows:

The gains for the VLR-RBFNN-SMC controller for the second trajectory are:

$$\varepsilon_N = \begin{bmatrix} 0.15 & 0.15 & 0.15 & 0.15 \end{bmatrix} \tag{3.57}$$

$$\varepsilon_D = \begin{bmatrix} 0.35 & 0.35 & 0.35 & 0.35 \end{bmatrix} \tag{3.58}$$

$$\mathbf{K} = \begin{bmatrix} 25 & 0 & 0 & 0 \\ 0 & 30 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 20 \end{bmatrix} \tag{3.59}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 \\ 0 & 0 & 15 & 0 \\ 0 & 0 & 0 & 15 \end{bmatrix} \tag{3.60}$$

The gains for the PID controller for the second trajectory are:

$$\mathbf{K_p} = \begin{bmatrix} 14 & 0 & 0 & 0 \\ 0 & 14 & 0 & 0 \\ 0 & 0 & 14 & 0 \\ 0 & 0 & 0 & 14 \end{bmatrix} \tag{3.61}$$

$$\mathbf{K_i} = \begin{bmatrix} 0.003 & 0 & 0 & 0 \\ 0 & 0.003 & 0 & 0 \\ 0 & 0 & 0.003 & 0 \\ 0 & 0 & 0 & 0.003 \end{bmatrix} \tag{3.62}$$

$$\boldsymbol{\lambda}_{pid} = \begin{bmatrix} 13 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 \\ 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 13 \end{bmatrix} \tag{3.63}$$

$$\mathbf{K_d} = \mathbf{K_p}\boldsymbol{\lambda}_{pid} \tag{3.64}$$

The gains for the SMC controller for the second trajectory are:

$$\mathbf{K_{smc}} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{3.65}$$

$$\boldsymbol{\eta}_{smc} = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \tag{3.66}$$

$$\boldsymbol{\lambda}_{smc} = \begin{bmatrix} 13 & 0 & 0 & 0 \\ 0 & 13 & 0 & 0 \\ 0 & 0 & 13 & 0 \\ 0 & 0 & 0 & 13 \end{bmatrix} \tag{3.67}$$

The controller's tracking errors for the VLR-RBFNN-SMC and the classical PID are plotted against one another. Fig. 3.75, Fig. 3.76, Fig. 3.77 show the tracking errors with no payload. Fig. 3.81, Fig. 3.82, Fig. 3.83 show the tracking errors for a 54 g payload. Fig. 3.87, Fig. 3.88, Fig. 3.89 show the tracking errors for a 102.6 g payload. It can be observed that the proposed controller is more robust against payload variation given that we have smaller tracking errors compared to the classical PID controller when submitted to the same conditions.

The controller's tracking errors for the VLR-RBFNN-SMC and the classical SMC are plotted against one another. Fig. 3.78, Fig. 3.79, Fig. 3.80 show the tracking errors with no payload. Fig. 3.84, Fig. 3.85, Fig. 3.86 show the tracking errors for a 54 g payload. Fig. 3.90, Fig. 3.91, Fig. 3.92 show the tracking errors for a 102.6 g payload. It can be observed that the proposed controller is more robust against payload variation given that we have smaller tracking errors compared to classical sliding mode when submitted to the same conditions.

(a) No payload:

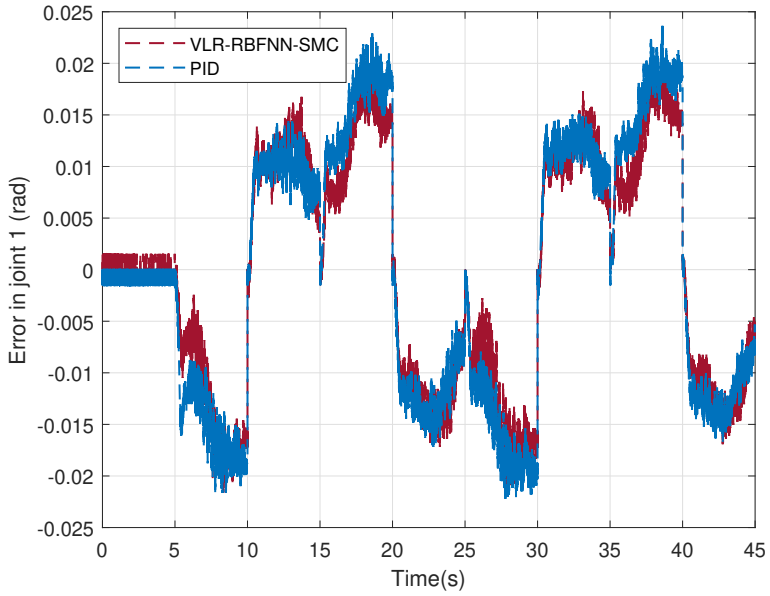


Figure 3.75: Trajectory 2 tracking error for joint 1 with no payload

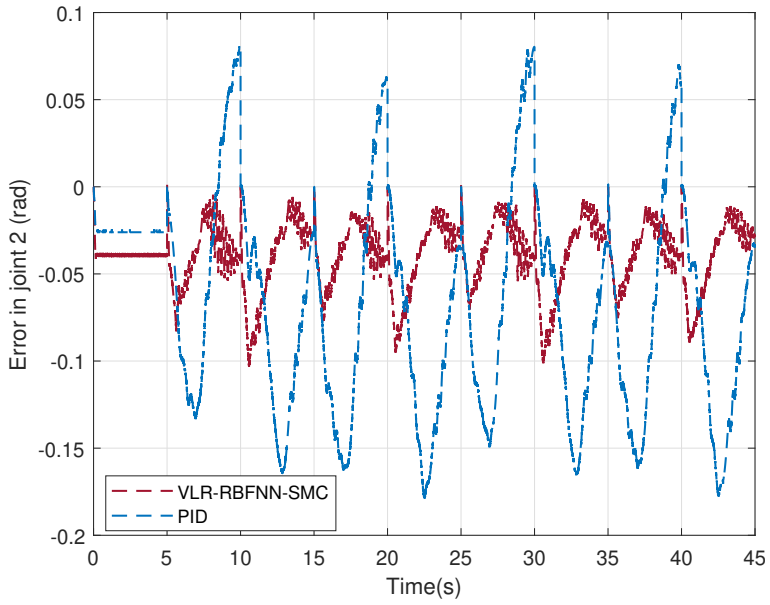


Figure 3.76: Trajectory 2 tracking error for joint 2 no payload

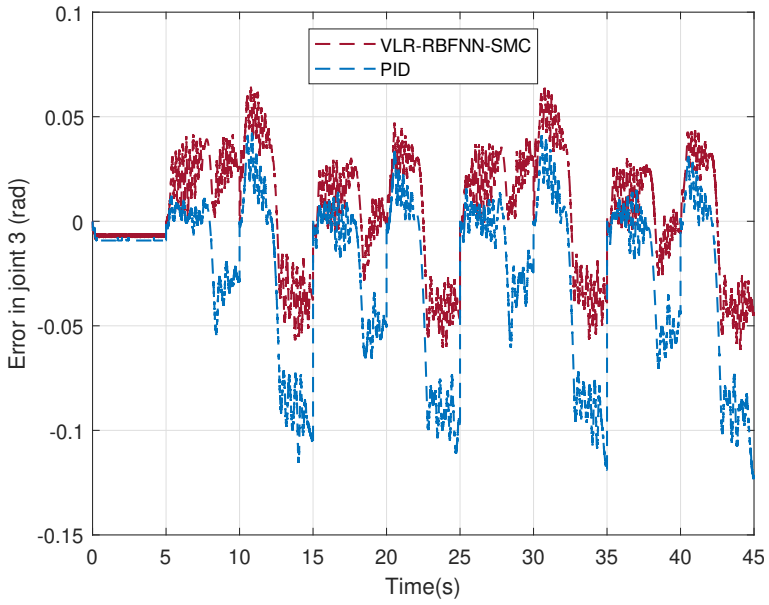


Figure 3.77: Trajectory 2 tracking error for joint 3 no payload

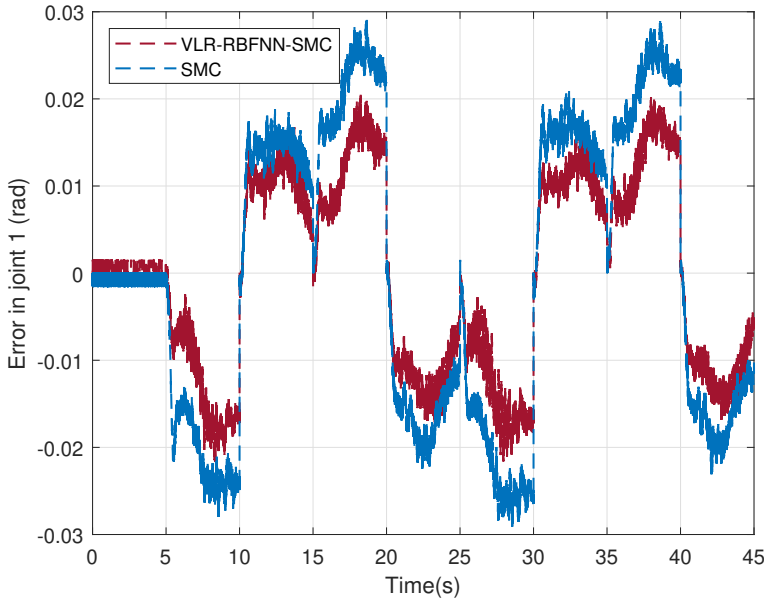


Figure 3.78: Trajectory 2 tracking error for joint 1 with no payload

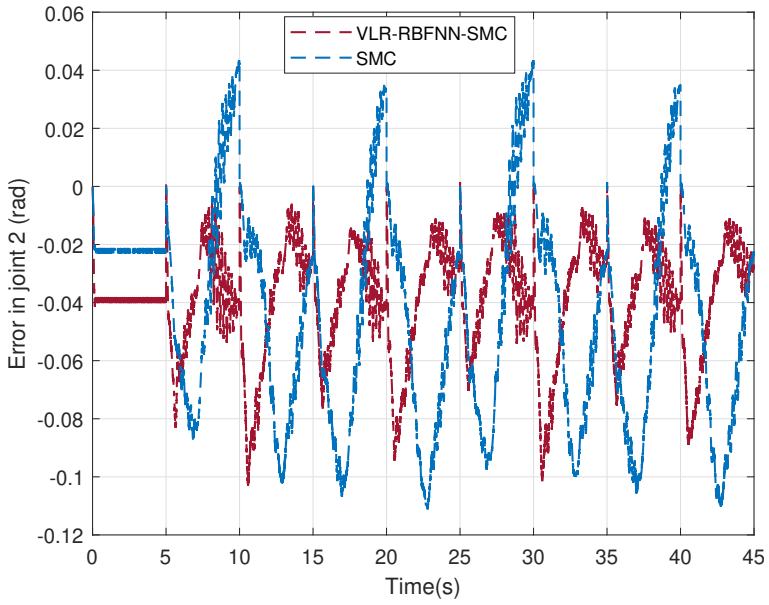


Figure 3.79: Tracking error for joint 2 with no payload

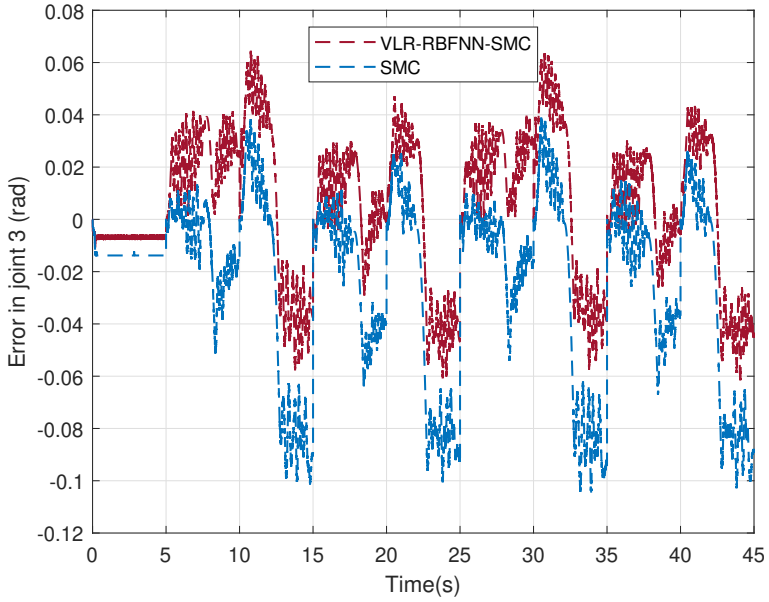


Figure 3.80: Trajectory 2 tracking error for joint 3 with no payload

Table 3.4, shows the absolute mean and maximum error of the tracking from the second trajectory of the controllers. We can see that the proposed method can outperform PID and SMC when

carrying no payload. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.4: Comparison of experimental errors for trajectory 2 with no payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	**0.0096**	**0.0220**	0.0115	0.0237	0.0153	0.0289
$\lvert e_{\theta_2} \rvert$ (rad)	**0.0413**	0.1292	0.0766	0.1789	0.0499	**0.1132**
$\lvert e_{\theta_3} \rvert$ (rad)	0.0255	**0.0875**	**0.0115**	0.1233	0.0358	0.1370
$\lvert e_x \rvert$ (m)	**0.0072**	0.0264	0.0107	0.0336	0.0077	**0.0229**
$\lvert e_y \rvert$ (m)	**0.0047**	**0.0189**	0.0072	0.0212	0.0060	0.0251
$\lvert e_z \rvert$ (m)	**0.0160**	**0.0462**	0.0361	0.0933	0.0289	0.0826

(b) 54 g payload:

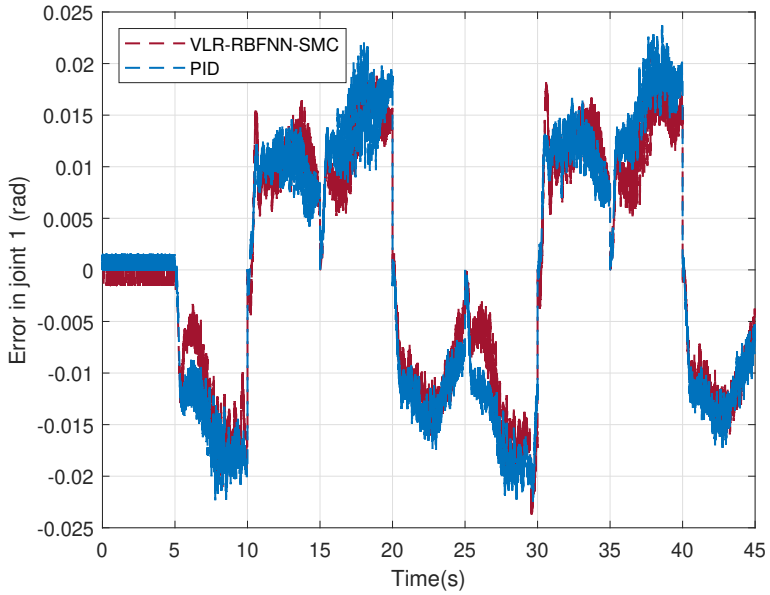


Figure 3.81: Trajectory 2 tracking error for joint 1 with 54 g payload

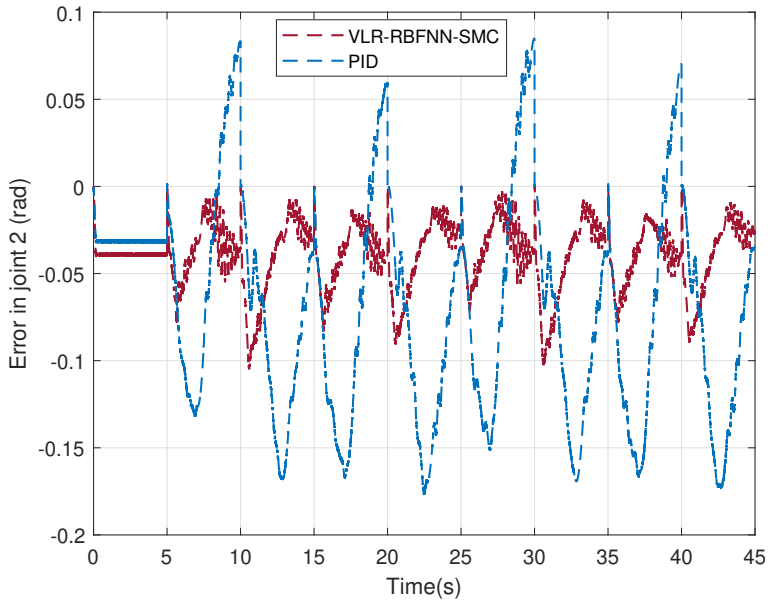


Figure 3.82: Trajectory 2 tracking error for joint 2 with 54 g payload

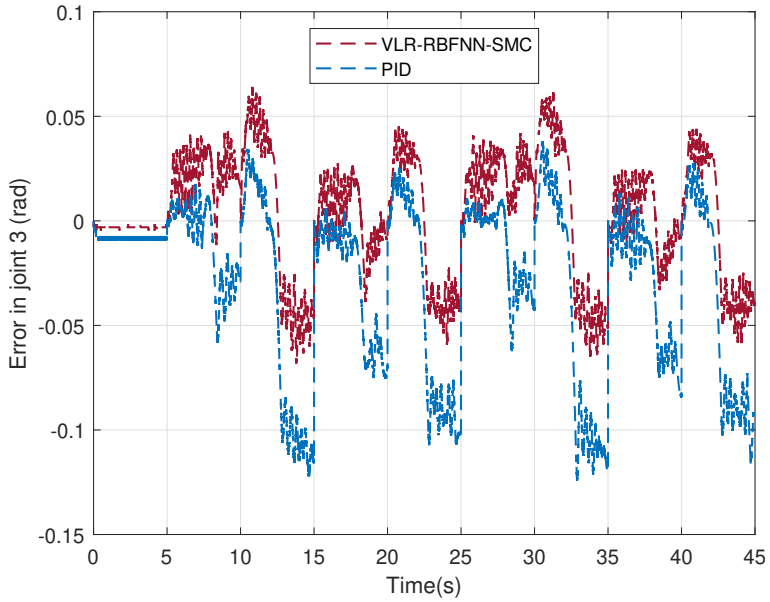


Figure 3.83: Trajectory 2 tracking error for joint 3 with 54 g payload

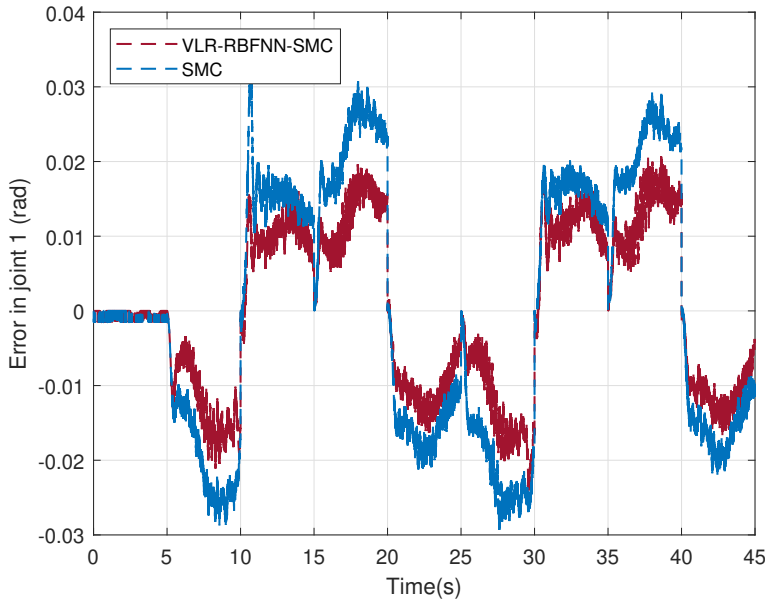


Figure 3.84: Trajectory 2 tracking error for joint 1 with 54 g payload

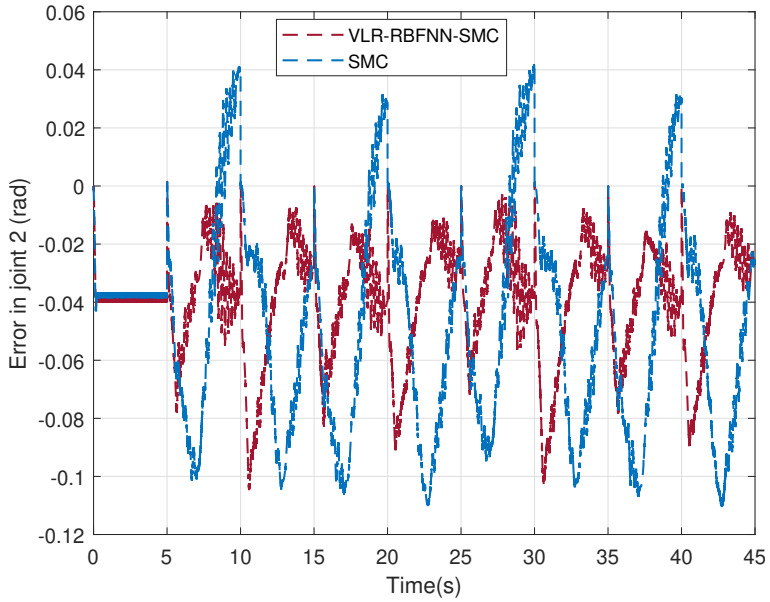


Figure 3.85: Trajectory 2 tracking error for joint 2 with 54 g payload

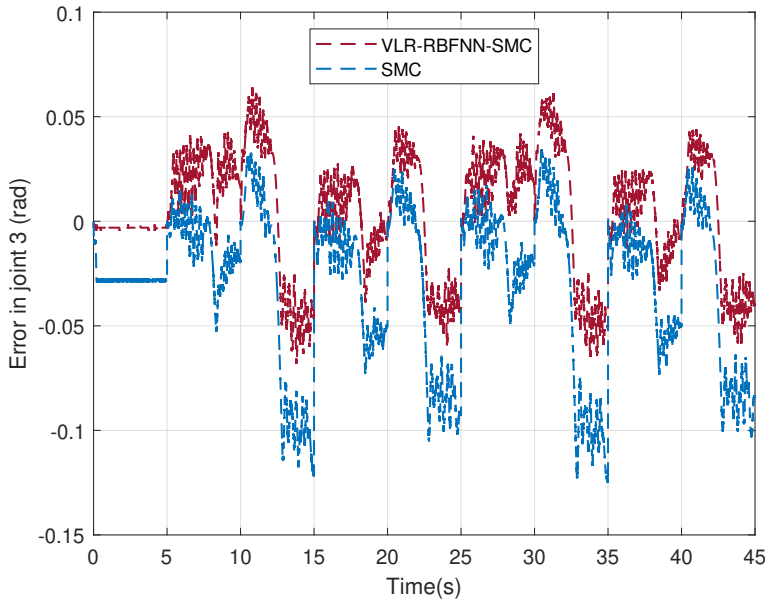


Figure 3.86: Trajectory 2 tracking error for joint 3 with 54 g payload

Table 3.5, shows the absolute mean and maximum error of the tracking from the second trajectory of the controllers. We can see that the proposed method can outperform PID and SMC when carrying a payload of 54 g. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.5: Comparison of experimental errors for trajectory 2 with 54.0 g payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	**0.0100**	0.0240	0.0113	**0.0238**	0.0155	0.0324
$\lvert e_{\theta_2} \rvert$ (rad)	**0.0396**	**0.1050**	0.0790	0.1771	0.0519	0.1103
$\lvert e_{\theta_3} \rvert$ (rad)	**0.0246**	**0.0684**	0.0351	0.1246	0.0347	0.1250
$\lvert e_x \rvert$ (m)	**0.0074**	**0.0255**	0.0104	0.0323	0.0088	0.0238
$\lvert e_y \rvert$ (m)	**0.0049**	**0.0201**	0.0072	0.0265	0.0060	0.0210
$\lvert e_z \rvert$ (m)	0.0166	**0.0255**	**0.0104**	0.0323	0.0293	0.0768

(c) 102.6 g payload:

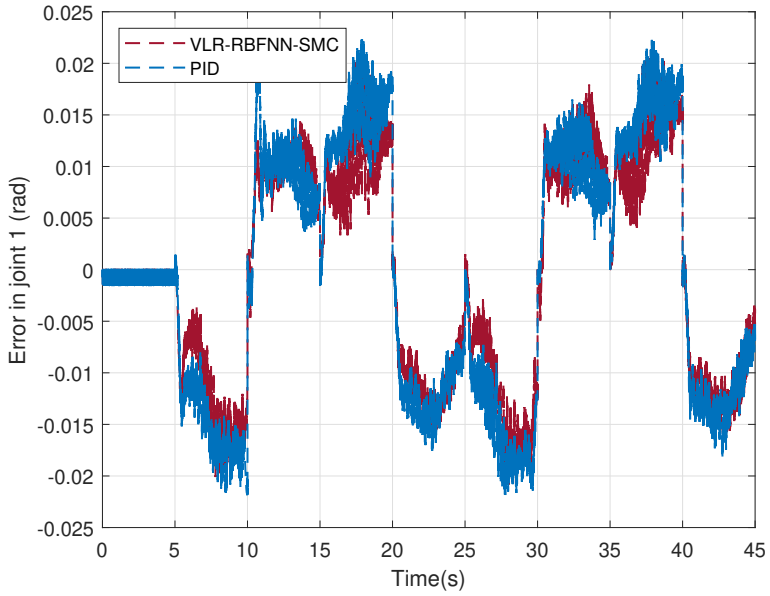


Figure 3.87: Trajectory 2 tracking error for joint 1 with 102.6 g payload

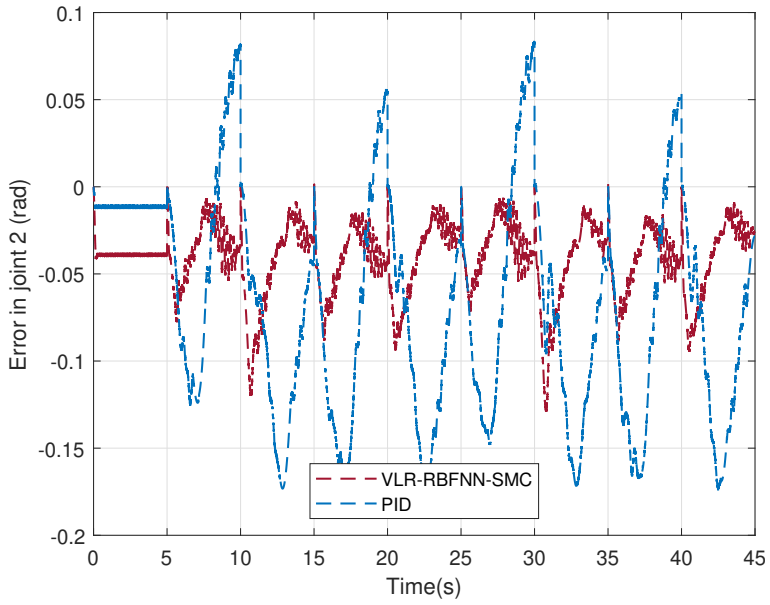


Figure 3.88: Trajectory 2 tracking error for joint 2 with 102.6 g payload

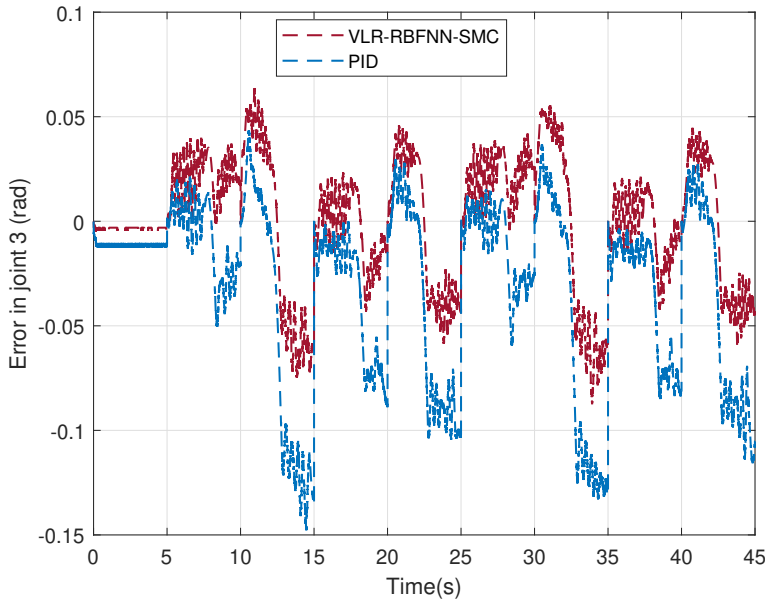


Figure 3.89: Trajectory 2 tracking error for joint 3 with 102.6 g payload

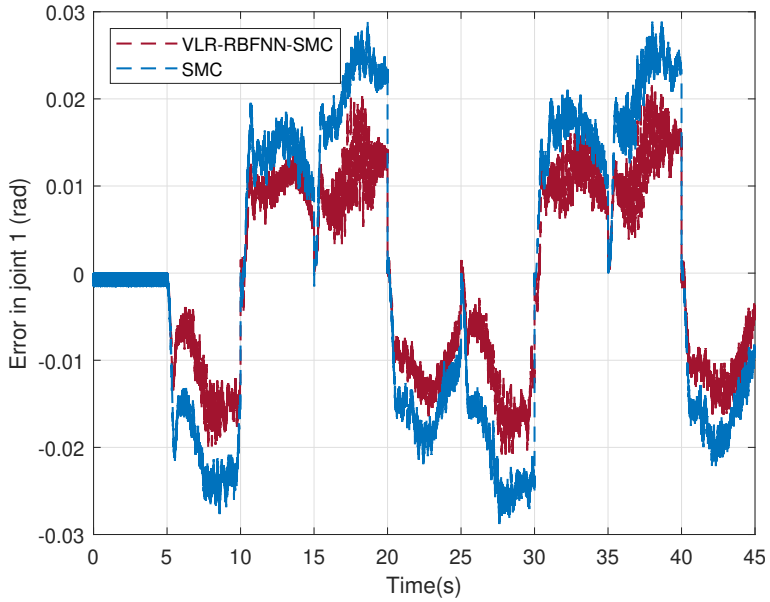


Figure 3.90: Trajectory 2 tracking error for joint 1 with 102.6 g payload

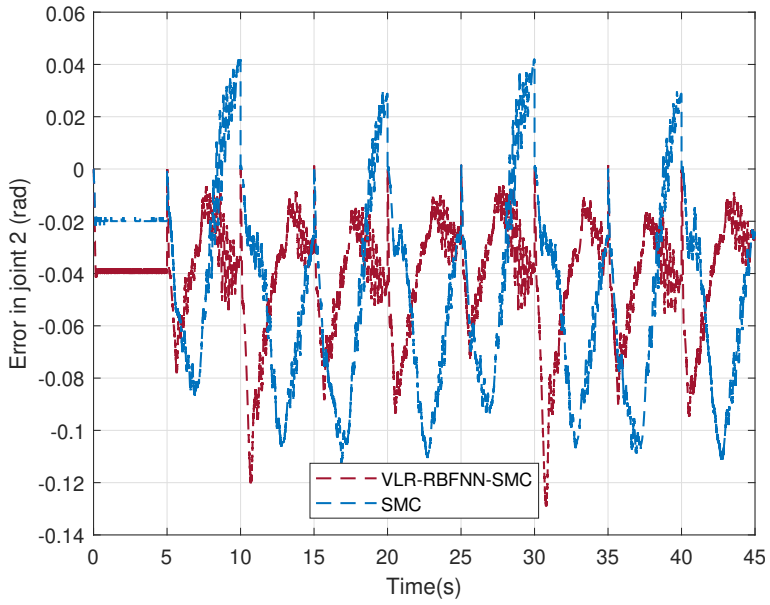


Figure 3.91: Trajectory 2 tracking error for joint 2 with 102.6 g payload

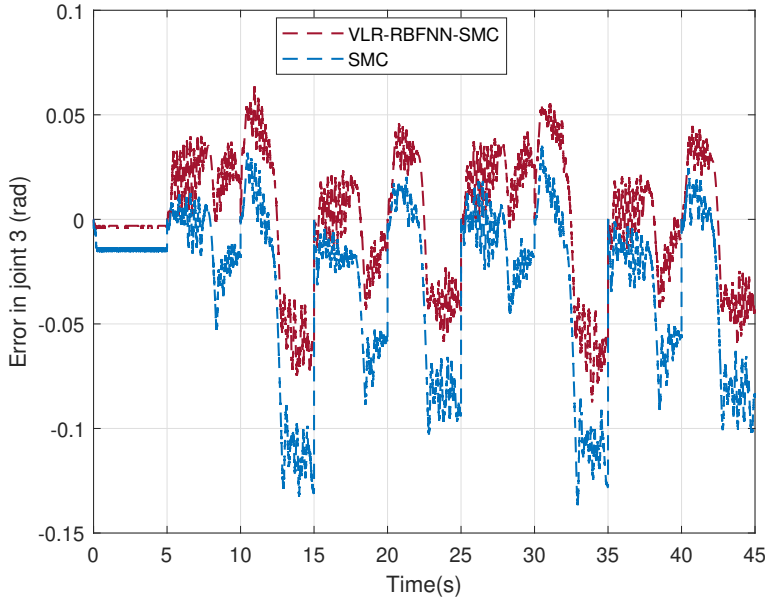


Figure 3.92: Trajectory 2 tracking error for joint 3 with 102.6 g payload

Table 3.6, shows the absolute mean and maximum error of the tracking from the second trajectory of the controllers. We can see that the proposed method can outperform PID and SMC

when carrying a payload of 102.6 g. For the reader's convenience, the overall smallest errors were highlighted.

Table 3.6: Comparison of experimental errors for trajectory 2 with 102.6 g payload

	VLR-RBFNN-SMC		**PID**		**SMC**	
	Mean	**Maximum**	**Mean**	**Maximum**	**Mean**	**Maximum**
$\lvert e_{\theta_1} \rvert$ (rad)	$\mathbf{0.0096}$	$\mathbf{0.0220}$	0.0382	0.0226	0.0153	0.0289
$\lvert e_{\theta_2} \rvert$ (rad)	$\mathbf{0.0413}$	0.1292	0.0768	0.1743	0.0499	$\mathbf{0.1132}$
$\lvert e_{\theta_3} \rvert$ (rad)	$\mathbf{0.0255}$	$\mathbf{0.0875}$	0.0382	0.1478	0.0358	0.1370
$\lvert e_x \rvert$ (m)	0.0182	0.0264	0.0092	0.0296	$\mathbf{0.0077}$	$\mathbf{0.0229}$
$\lvert e_y \rvert$ (m)	$\mathbf{0.0047}$	$\mathbf{0.0189}$	0.0071	0.0271	0.0060	0.0251
$\lvert e_z \rvert$ (m)	$\mathbf{0.0182}$	$\mathbf{0.0462}$	0.0383	0.1009	0.0289	0.0826

3.4.7 Comparison with Back-stepping Controller from Literature

In [67], the authors proposed a feedback controller by combining a backstepping controller with a high-gain observer. They aimed to track the end-effector in x and y coordinates given a desired trajectory. The backstepping controller was designed to drive the system states to follow the trajectory and the observer was designed to estimate the unknown states of the system.

The QArm was used to evaluate their method, which is the same hardware used in this work, so we believe this would be a fair approach to demonstrate our control achievements. Therefore, to compare the proposed controller performance, we implement it to follow the desired trajectory shown in Eq. (3.68), where the frequency is given as $f = 0.125$ Hz, same as [67].

$$\begin{aligned} x(t) &= 0.4 + 0.1\text{sin}(2\pi f t) \\ y(t) &= 0.2 + 0.1\text{cos}(2\pi f t) \end{aligned} \tag{3.68}$$

The gains of the VRL-RBFNN-SMC were chosen as:

$$\boldsymbol{\eta} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix} \tag{3.69}$$

$$\mathbf{K} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 47 & 0 & 0 \\ 0 & 0 & 37 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \tag{3.70}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.71}$$

where $\boldsymbol{\eta} = \varepsilon_N + \varepsilon_D$.

The control performance for the trajectory tracking of Eq. (3.68) is shown from Fig. 3.93 to Fig. 3.95.

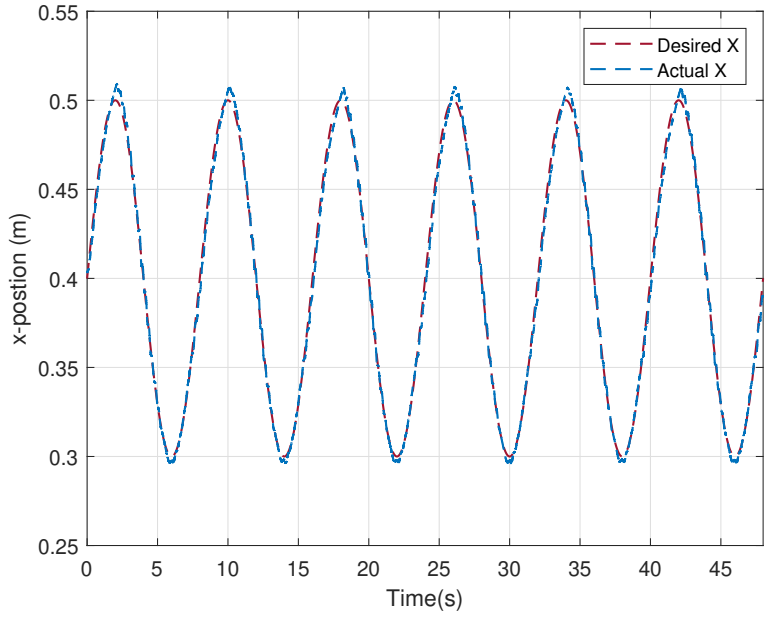


Figure 3.93: Trajectory tracking for x-direction

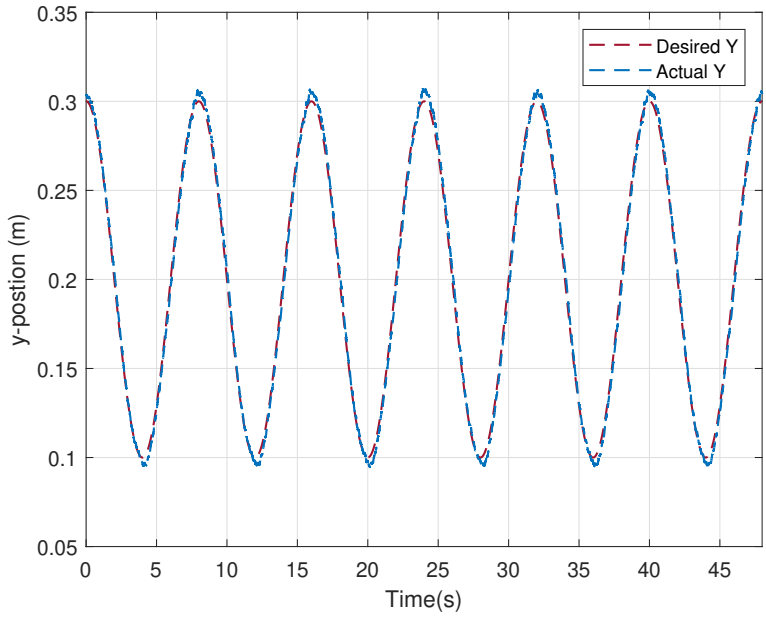


Figure 3.94: Trajectory tracking for y-direction

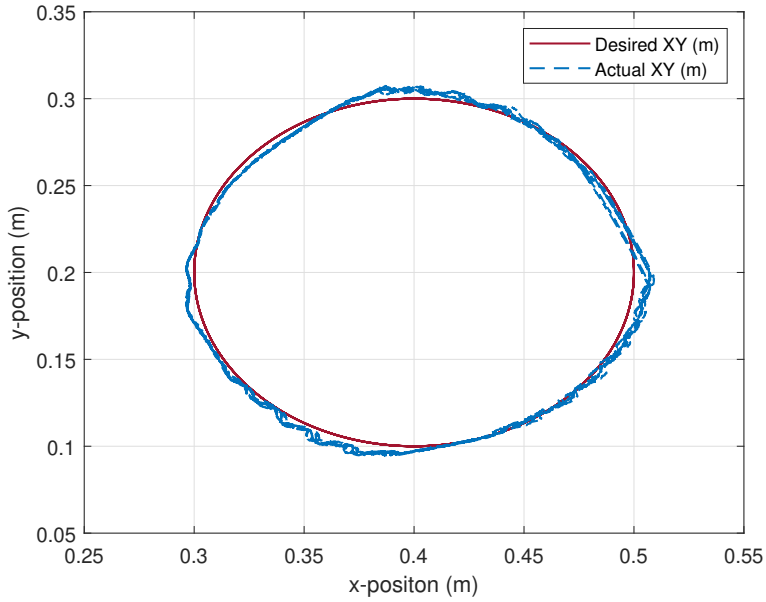


Figure 3.95: Path following for x-y directions

The tracking errors in x-y directions are shown in Fig. 3.96. We can see that the absolute error

in the x direction does not go above 0.01 m and the absolute error in the y direction does not go above 0.008m. In contrast to [67], where a trajectory tracking error for the end-effector was of less than 3%, the proposed controller can achieve a tracking error for the end-effector in x-y directions with a total error of less than 2.12% as can be seen in Fig. 3.97.

where x_d and y_d are the desired x-y positions and x_m and y_m are the actual x-y positions.

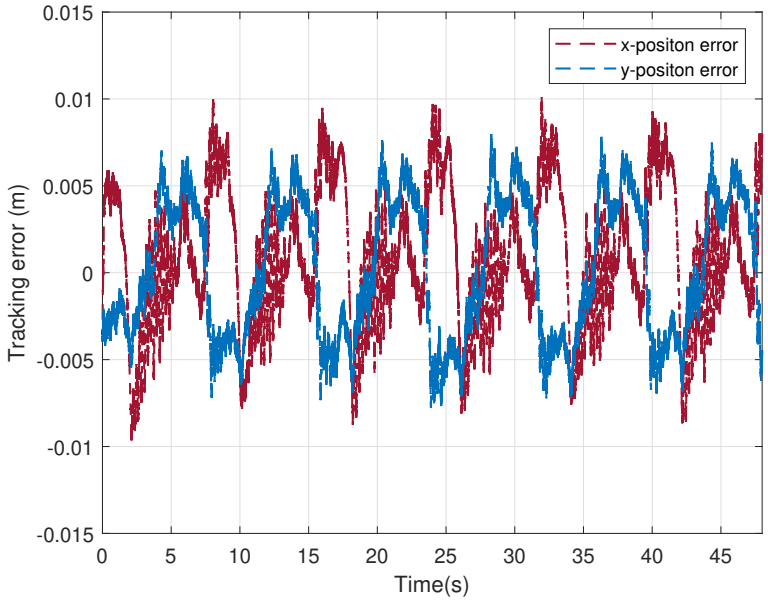


Figure 3.96: Tracking error for x-y directions

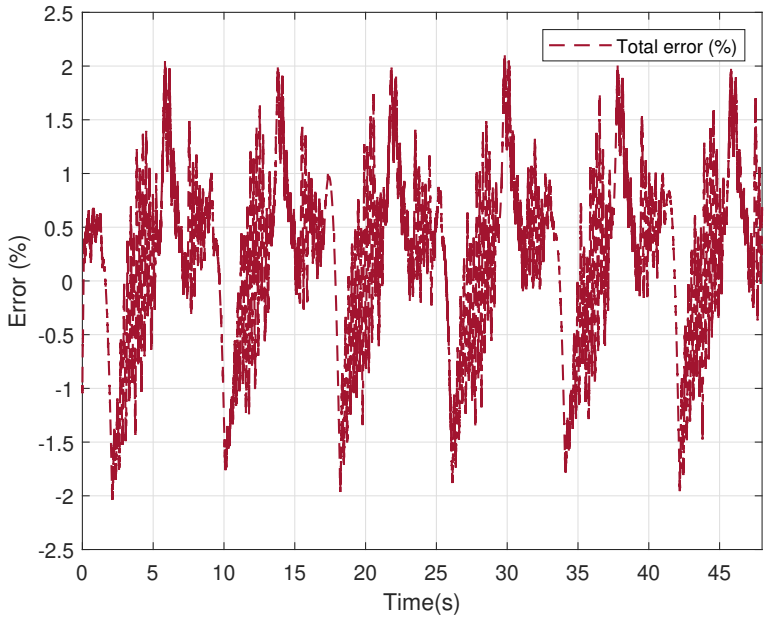


Figure 3.97: x-y tracking error percentage

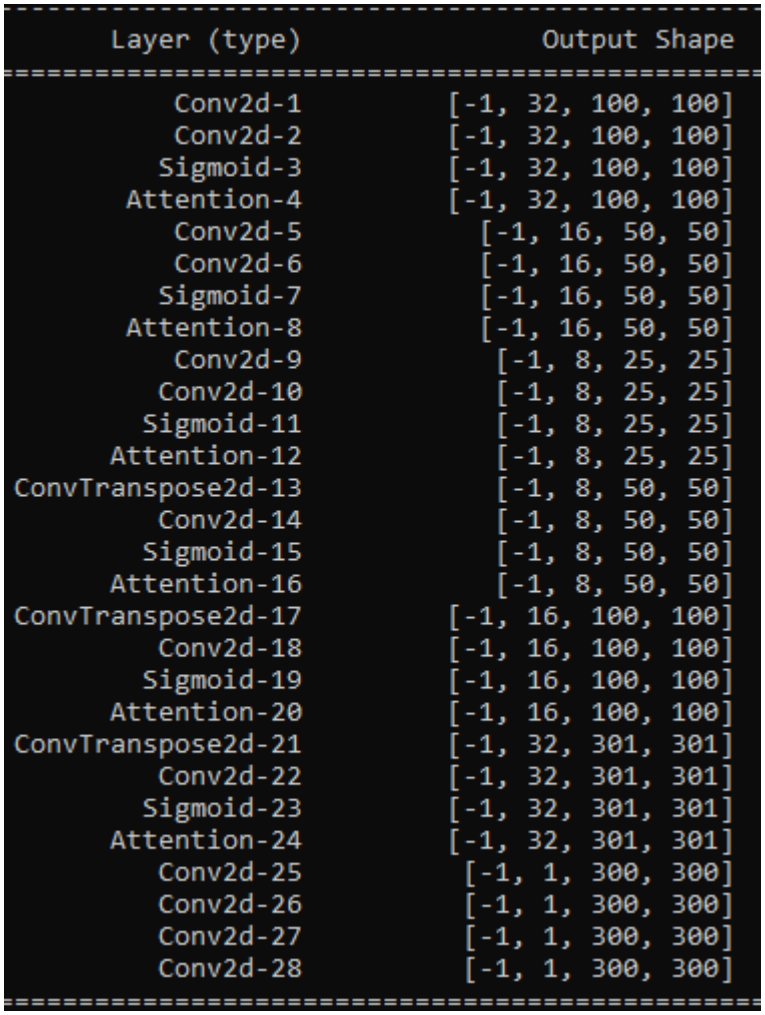
3.4.8 Implementation for Autonomous Manipulation

Accurately following control commands is not enough for precise autonomous manipulation by itself. The robot should also be able to perceive and interact with its surroundings considering the information feedback from its sensors. Therefore, acquiring the position and orientation of the object is important information to achieve an autonomous manipulation task successfully.

Deep learning techniques have been largely used to implement detection methods. The Generative Grasping Convolutional Neural Network (GGCNN) is a grasp detection neural network developed by [68]. The real-time model can be used for closed-loop grasping. This neural network can detect the grasp candidates on a single stage, taking the depth camera as input and directly outputting the grasp detection.

For this implementation, the GGCNN is modified by adding layers consisting of a convolution layer and a sigmoid function layer to the model to enhance the performance of the algorithm. The structure of the layers can be seen in Fig. 3.98. The model takes a 300x300 depth image as an

input and outputs the best grasp detection.



Layer (type)	Output Shape
Conv2d-1	[-1, 32, 100, 100]
Conv2d-2	[-1, 32, 100, 100]
Sigmoid-3	[-1, 32, 100, 100]
Attention-4	[-1, 32, 100, 100]
Conv2d-5	[-1, 16, 50, 50]
Conv2d-6	[-1, 16, 50, 50]
Sigmoid-7	[-1, 16, 50, 50]
Attention-8	[-1, 16, 50, 50]
Conv2d-9	[-1, 8, 25, 25]
Conv2d-10	[-1, 8, 25, 25]
Sigmoid-11	[-1, 8, 25, 25]
Attention-12	[-1, 8, 25, 25]
ConvTranspose2d-13	[-1, 8, 50, 50]
Conv2d-14	[-1, 8, 50, 50]
Sigmoid-15	[-1, 8, 50, 50]
Attention-16	[-1, 8, 50, 50]
ConvTranspose2d-17	[-1, 16, 100, 100]
Conv2d-18	[-1, 16, 100, 100]
Sigmoid-19	[-1, 16, 100, 100]
Attention-20	[-1, 16, 100, 100]
ConvTranspose2d-21	[-1, 32, 301, 301]
Conv2d-22	[-1, 32, 301, 301]
Sigmoid-23	[-1, 32, 301, 301]
Attention-24	[-1, 32, 301, 301]
Conv2d-25	[-1, 1, 300, 300]
Conv2d-26	[-1, 1, 300, 300]
Conv2d-27	[-1, 1, 300, 300]
Conv2d-28	[-1, 1, 300, 300]

Figure 3.98: Model architecture

For this case study, the proposed controller is combined with a modified GGCNN to realize autonomous payload manipulation. The detection module is implemented in Python and returns the centroid coordinates of the detection bounding box as well as the depth distance of the object from the camera. This information is then sent to the Matlab/Simulink module, which uses the camera parameters to acquire the object pick coordinates. Since the object place coordinate is known, the autonomous manipulation is then realized.

3.4.8.1 Camera Calibration

Camera calibration is an important aspect of computer vision, especially when implementing detection algorithms. The calibration estimates the camera's intrinsic and extrinsic parameters that are essential to map 3D objects in world coordinates to 2D image coordinates and vice-versa, their relationship can be seen in Fig. 3.99. The extrinsic parameters project to the 3D world coordinates into 3D camera coordinates and consist of a 3x3 rotation matrix and a 3x1 translation matrix. We can visualize the extrinsic parameters from the calibration in Fig. 3.101.

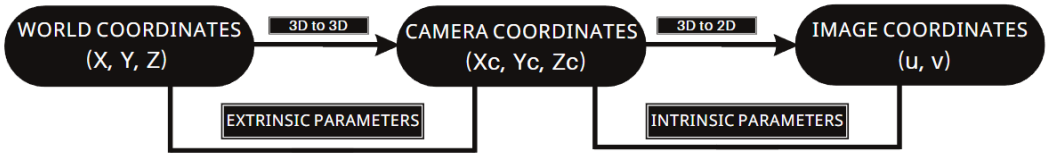


Figure 3.99: Coordinates transformation

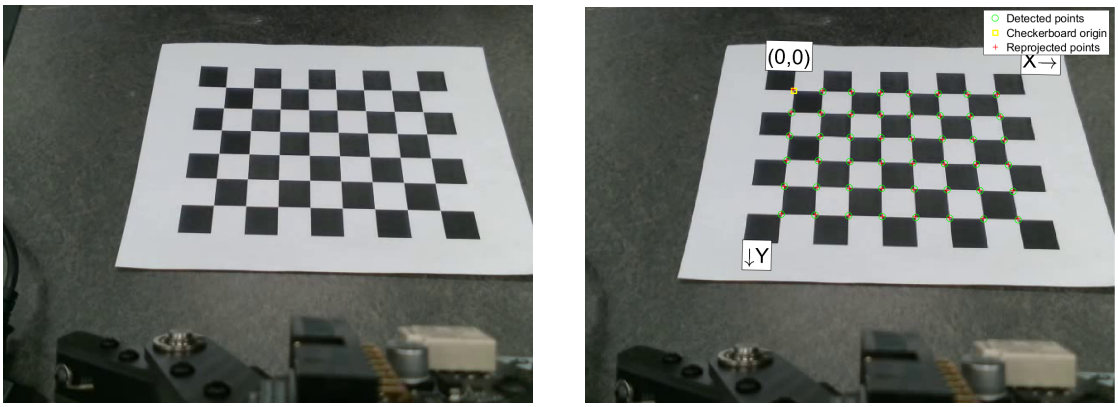
The intrinsic parameters project the 3D camera coordinates into 2D image coordinates and is defined by:

$$I = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.72}$$

where f_x and f_y are the focal length in pixels, c_x and c_y are the optical center and s is the skew coefficient.

The calibration was done using the Camera Calibrator from Matlab [69]. A minimum of 20 images are necessary for good calibration, in this case, 28 images of an 8x6 chessboard acquired from the camera located at the top of the QArm end-effector were uploaded to the calibrator app. Fig. 3.100a is an example of the images uploaded. After that, we ran the calibration to detect the intersection points on the chessboard images as shown in Fig. 3.100b.

The results from the calibration can be seen in Fig. 3.101 which represents the visualization



(a) Sample image from the calibration

(b) Detected points on the chessboard

Figure 3.100: Calibration image samples

of the extrinsic parameters and Fig. 3.102 which shows the calibration accuracy by displaying the error per image. The mean projection error per image for this calibration is 0.85 pixels. The precision requirements for the applications of the camera calibration will determine if the errors need to be further improved. To do that, one can take more calibration images or exclude the images with higher projection errors from the calibration.

3.4.8.2 Training and Evaluation

The neural network was trained on the Cornell dataset [70] for a 100 epochs, which is a neural network hyperparameter that consists of the number of times that all training data passes through the algorithm [71]. The dataset was split as follows: 80% for training and 20% for validation. Some sample detections on the validation dataset are shown in Fig. 3.103.

The modified GGCNN model performed 81% on the Cornell validation dataset when compared to the original model, which got 76.4% for the GGCNN architecture using the pre-trained weights provided by the authors. The success rate was computed based on the IoU (Intersection over Union) method which is given by Eq. (3.73).

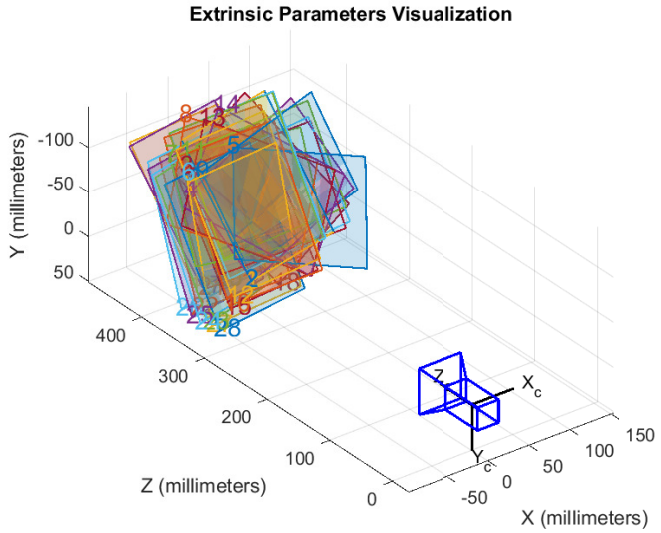


Figure 3.101: Camera extrinsic parameters

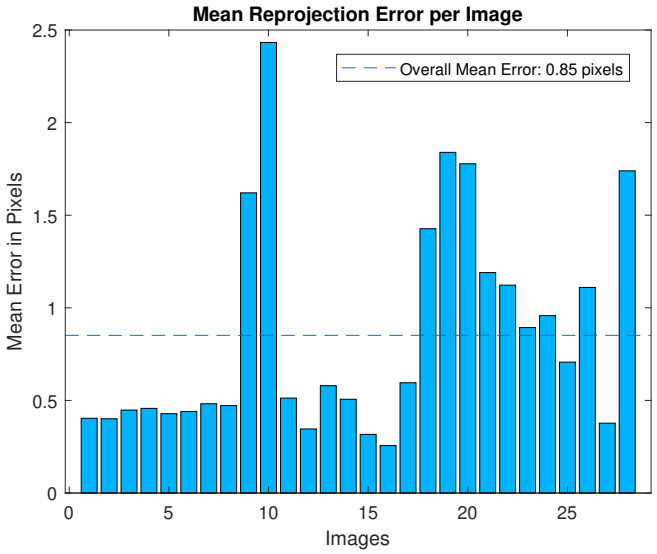


Figure 3.102: Reprojection error per image

$$IoU = \frac{P_c}{P_c + P_f} \tag{3.73}$$

where P_c are the correct predictions and P_f are the failed predictions.

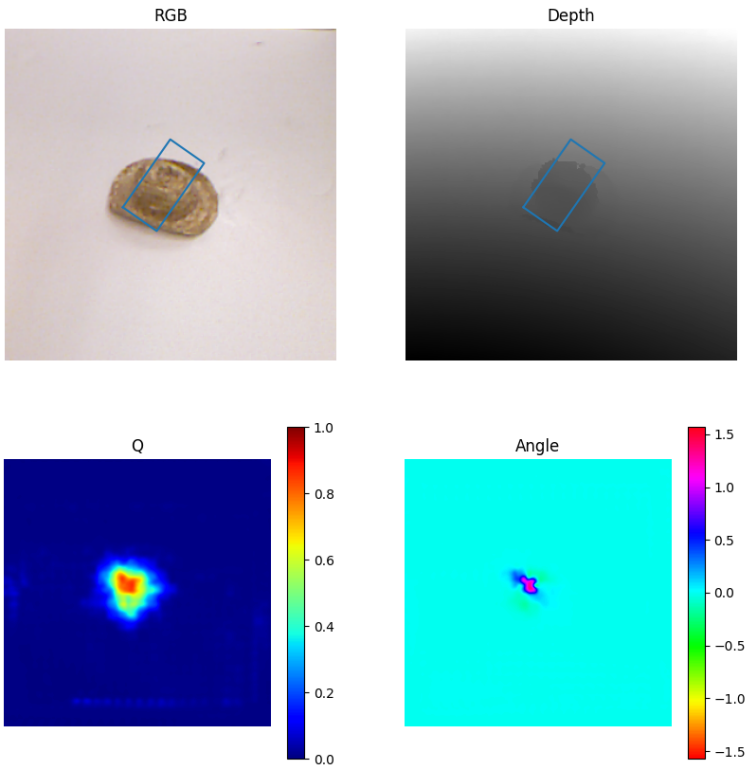


Figure 3.103: Grasp detection on Cornell validation dataset. The top left image shows the detection on the RGB image. The top right image shows the detection on the depth image. The bottom left image shows the grasp quality and the bottom right image shows the angle in radians for grasping

3.4.8.3 Grasping Trials

The modified neural network model was implemented on hardware to execute a pick-and-place task. The object's place position is known, however, the object's pick position is unknown. The picking coordinate is acquired from the detection. The centroid coordinates of the bounding box are acquired and from that, we can get the depth distance of the object to the camera. With this information, we can transform the object location from the image frame to the world frame by using the parameters estimated from the camera calibration.

That being said, a Python script was implemented to run inference on the camera, realize the detections, and return the detection centroid coordinates, depth distance, and wrist angle. The detections are resolved in 0.0333 seconds so then this model is feasible for real-time applications. The information obtained from the detections is then sent to Matlab/Simulink using the socket interface via TCP/IP communication to generate the control commands and trajectory to drive the system to realize the task. Fig. 3.104 and Fig. 3.109 show the servoing module where the robot is doing the detection. Once the detection is realized and the position of the object in the workspace is obtained, the robot moves to the pick location, picks the object Fig. 3.105, moves it to the place location at the white container Fig. 3.106, and places it Fig. 3.107.

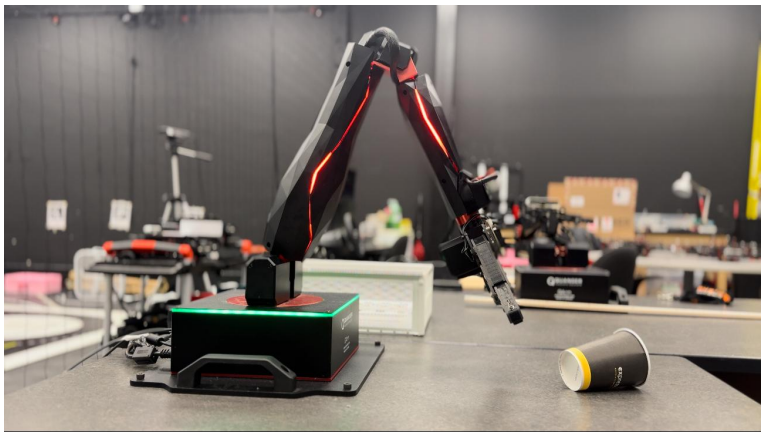


Figure 3.104: Robot realizing the detection

The validate the performance of the proposed method three sets of tests were realized. The first trial consisted of trying to grasp a red cup for 10 times. The second one was repeating the first trial but with a black cup. And finally, 10 attempts of grasping both red and black cups were performed, in order to see how the proper method adapts to different objects, considering that the cups have different colors and sizes.

For the first trial, the robot was able to successfully pick and place the object 6 times out of 10. For the second trial, the robot was able to successfully pick and place the object 7 times out of

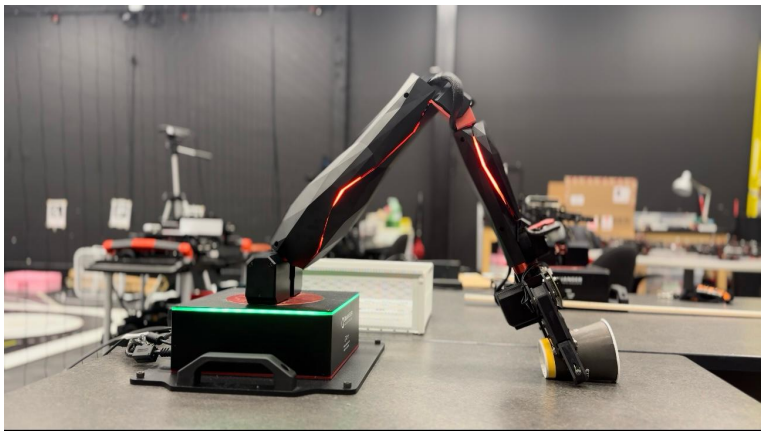


Figure 3.105: Robot picking the object

Figure 3.106: Robot moving to the place position

10. For the last trial, the robot was able to successfully pick and place the objects 5 times out of 10. Samples of the detections can be seen in Fig. 3.109 and Fig. 3.108.

3.4.8.4 Trajectory Tracking for Grasping

The trajectory is designed as a set of waypoints. As mentioned before, the pick waypoint is acquired from the detection, and the place location is known. The joints and xyz-direction tracking errors are shown in Fig. 3.110 and Fig. 3.111.

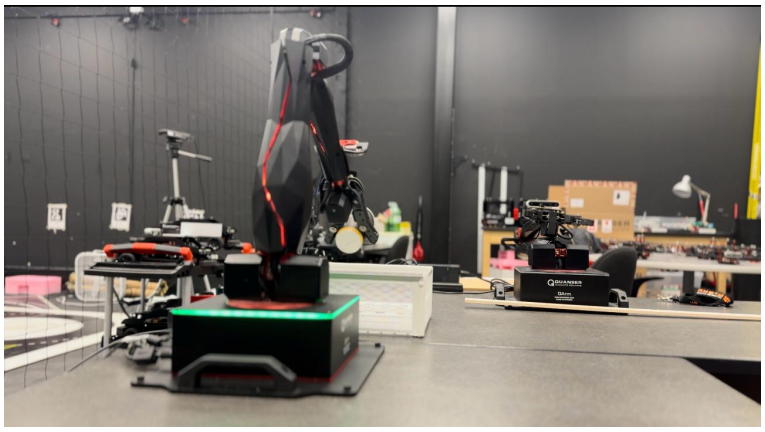


Figure 3.107: Robot placing the object

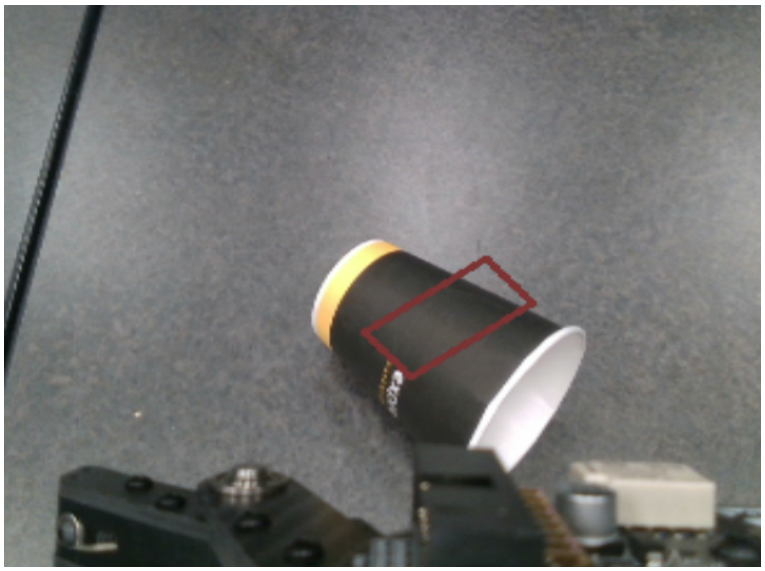


Figure 3.108: Grasp detection for the black cup from the camera view

The gains for the VLR-RBFNN-SMC controller for the autonomous manipulation are:

$$\varepsilon_N = \begin{bmatrix} 0.15 & 0.15 & 0.15 & 0.15 \end{bmatrix} \tag{3.74}$$

$$\varepsilon_D = \begin{bmatrix} 0.35 & 0.35 & 0.35 & 0.35 \end{bmatrix} \tag{3.75}$$

Figure 3.109: Grasp detection for the red cup from the camera view

$$\mathbf{K} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix} \tag{3.76}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} 12 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 0 & 12 \end{bmatrix} \tag{3.77}$$

Since the desired trajectory generation is not continuous, when the robot moves from one waypoint to another an overshoot when changing waypoints occurs. For this reason, we can see a few jumps in the tracking errors. However, since these are pointed spikes, the overall control performance is similar to the ones achieved in the previous trajectories as we can see in Table 3.7 that shows the controller's absolute mean and median errors.

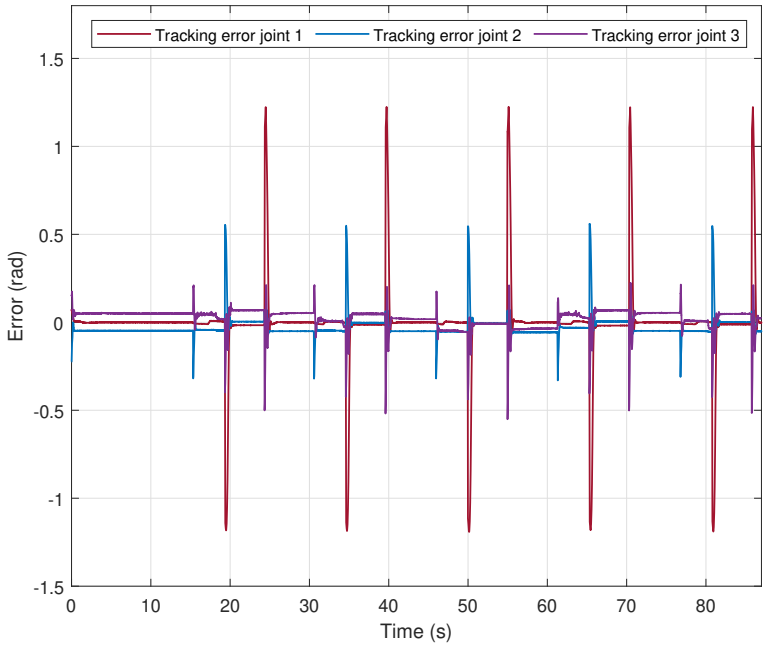


Figure 3.110: Joint tracking errors

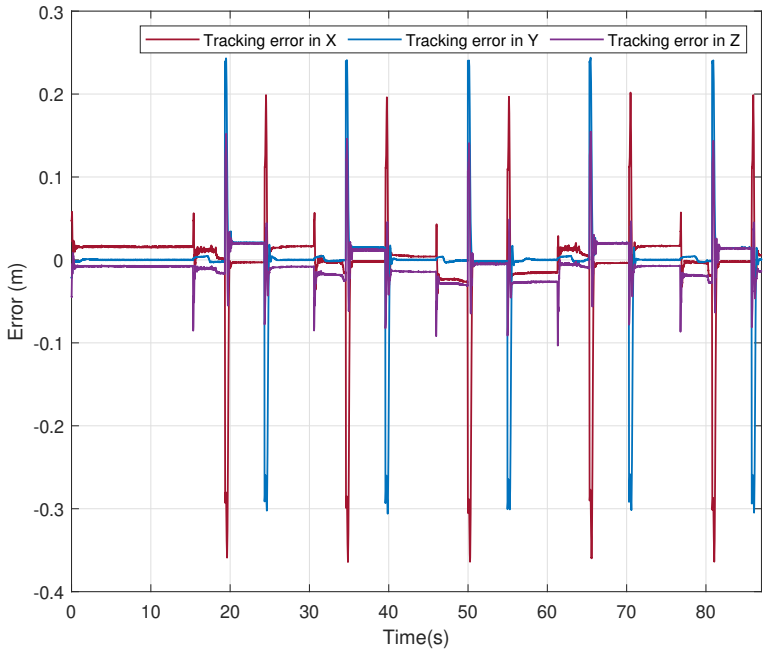


Figure 3.111: xyz-directions tracking errors

Table 3.7: Experimental errors for the trajectory tracking of the autonomous manipulation

	VLR-RBFNN-SMC	
	Mean	**Median**
$\lvert e_{\theta_1} \rvert$ (rad)	0.0581	0.0047
$\lvert e_{\theta_2} \rvert$ (rad)	0.0490	0.0473
$\lvert e_{\theta_3} \rvert$ (rad)	0.0490	0.0479
$\lvert e_x \rvert$ (m)	0.0205	0.0139
$\lvert e_y \rvert$ (m)	0.0158	0.0014
$\lvert e_z \rvert$ (m)	0.0166	0.0135

4 Conclusions and Future Work

4.1 Conclusions

This thesis focused on robust control for trajectory tracking in robotic payload transportation which can contribute to the advancements of autonomous warehouse implementation or other scenarios that involve robotic manipulation of payloads. A significant topic in autonomous robotic manipulation was addressed: control design for trajectory tracking.

A neural network-based sliding mode control with a variable learning rate was proposed to follow a desired trajectory. Extensive experiments were realized to validate the proposed model performance. First, the VLR-RBFNN-SMC estimation performance is compared to the RBFNN-SMC with a fixed learning rate for two trajectories. From the results, it was shown that the proposed approach can suppress the overshoot when estimating the dynamics. Second, VLR-RBFNN-SMC was compared with classical PID and SMC for different trajectories under payload variations. In the results, it was shown that the VLR-RBFNN-SMC can outperform the classical controllers, especially under complex scenarios (larger payload and wider joint angle variations). Third, the VLR-RBFNN-SMC was compared to a controller from the literature [67] to follow a desired path. The proposed approach can achieve a total tracking error for the end-effector in x-y directions of less than 2.12% in comparison with less than 3% achieved in [67]. Finally, the proposed controller was implemented with a grasp detection algorithm to realize an autonomous task.

For the first trial, the task succeeded 60% of the time. For the second trial, the task succeeded 70% of the time. For the third trial which was a mix of two objects, the task succeeded 50% of the time.

Compared to other approaches the contributions of this work consist of:

- Instead of considering the nominal model and approximating the uncertainties, this work considers that the dynamics model is completely unknown, using a learning-based approach to estimate the system model.
- The proposed controller was extensively validated and evaluated under different conditions with different types of trajectories. The results showed that the proposed controller outperformed other alternatives including some controllers from the literature.
- The proposed controller can be smoothly integrated with grasping techniques to address autonomous payload manipulation tasks.

4.2 Future Work

The work of this thesis can likely be extended to investigate different aspects of the control design problem. This thesis considered estimating the overall unknown dynamics of the robotic system. However, some forces have a greater impact on the system's performance, for example, the gravitational force. Therefore, other applications of robot manipulators may require more robust compensation for these forces acting on the system. Consequently, designing a pipeline such as an observer or estimator that can improve the compensation for the action of these forces on the robot's joints might be a good topic to explore. Furthermore, studying and investigating methods for trajectory generation that can avoid obstacles is a good topic to further extend and contribute to this work. Additionally, extending the implementation of this controller to be integrated with autonomous manipulation by investigating grasp detection and vision-based techniques is an open research topic.

References

[1] Quanser, *QArm User Manual*, 2021.

[2] QUANSER, "QArm: Modern Manipulator Arm for Robotics Courses and Research," https://www.quanser.com/products/qarm/ (2022/02/12).

[3] J. Liu, *Radial Basis Function (RBF) Neural Network Control for Mechanical Systems*. Springer, 2013.

[4] D. Morrison, A. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, A. Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, C. Lehnert, I. Reid, P. Corke, and J. Leitner, "Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 7757–7764. DOI: 10.1109/ICRA.2018.8463191

[5] Canadian Space Agency, "About Canadarm2," https://www.asc-csa.gc.ca/eng/iss/canadarm2/about.asp (2022/02/12).

[6] M. De Stefano, H. Mishra, A. M. Giordano, R. Lampariello, and C. Ott, "A relative dynamics formulation for hardware-in-the-loop simulation of on-orbit robotic missions," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3569–3576, 2021. DOI: 10.1109/LRA.2021.3064510

[7] K. S. E. S. S. S. Sayour, Malak H., "Autonomous robotic manipulation: Real-time, deep-learning approach for grasping of unknown objects," *Journal of Robotics*, 2022. DOI: 10.1007/s10846-021-01536-6

[8] K. H. K. S. H. Son, Jeongwoo, "A review on robust control of robot manipulators for future manufacturing," *International Journal of Precision Engineering and Manufacturing*, 2023. DOI: 10.1007/s12541-023-00812-9

[9] F. H. X. T. W. F. Wu, Yuxiang, "Adaptive neural fixed-time sliding mode control of uncertain robotic manipulators with input saturation and prescribed constraints," *Neural Processing Letters*, 2022. DOI: 10.1007/s11063-022-10788-8

[10] T. H.-V.-A. A. K. K. Tran, Duc-Thien, "Adaptive nonsingular fast terminal sliding mode control of robotic manipulator based neural network approach," *International Journal of Precision Engineering and Manufacturing*, 2021. DOI: 10.1007/s12541-020-00427-4

[11] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-driven model predictive control for trajectory tracking with a robotic arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3758–3765, 2019. DOI: 10.1109/LRA.2019.2929987

[12] R. Cheng, L. Li, H. Li, S. Cheng, X. Fang, and Y. Fan, "Fixed-time fault-tolerant sliding mode control for robot manipulator," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 581–586. DOI: 10.23919/CCC55666.2022.9901736

[13] Dynamixel, "XM540-W270-R Servo Motor," https://emanual.robotis.com/docs/en/dxl/x/xm540-w270/ (2023/10/12).

[14] Dynamixel, "XM430-W350-R Servo Motor," https://emanual.robotis.com/docs/en/dxl/x/xm430-w350/ (2023/10/12).

[15] QUANSER, "QLabs Virtual QArm," https://www.quanser.com/products/qlabs-virtual-qarm/ (2022/02/12).

[16] Quanser, *QArm Instructor Academic Resources*, 2021.

[17] V. Spong, *Robot Dynamics and Control*. Springer, 2018, ch. 3 and 4.

[18] X. Li and W. Yu, "A systematic tunning method of PID controller for robot manipulators," in *2011 9th IEEE International Conference on Control and Automation (ICCA)*, 2011, pp. 274–279. DOI: 10.1109/ICCA.2011.6138081

[19] 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. DOI: 10.1109/ACCESS.2020.3014348

[20] 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. DOI: 10.1109/ACCESS.2020.3014348

[21] N. W. Y.-Q. N. V. Long, Mai Thang, "Adaptive robust self-tuning PID fault-tolerant control for robot manipulators," *International Journal of Dynamics and Control*, 2024. DOI: 10.1007/s40435-023-01197-3

[22] K. S. S. A. Loucif, Fatiha, "Whale optimizer algorithm to tune PID controller for the trajectory tracking control of robot manipulator," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, 2019. DOI: 10.1007/s40430-019-2074-3

[23] H. Li, B. Yang, S. Jiang, and J. Luo, "Robot manipulator trajectory tracking with uncertainties based on adaptive sliding control," in *2023 IEEE International Conference on Unmanned Systems (ICUS)*, 2023, pp. 1434–1439. DOI: 10.1109/ICUS58632.2023.10318441

[24] Q. Ai, Q. Yang, M. Li, X. Feng, and W. Meng, "Implementing multi-dof trajectory tracking control system for robotic arm experimental platform," in *2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, 2018, pp. 282–285. DOI: 10.1109/ICMTMA.2018.00075

[25] H. Zhao, C. Lv, and J. Chen, "Robust adaptive trajectory tracking control for robot manipulator with friction disturbances," in *2022 34th Chinese Control and Decision Conference (CCDC)*, 2022, pp. 5293–5298. DOI: 10.1109/CCDC55256.2022.10033977

[26] L. Wang and S. Guan, "Research on trajectory tracking control for scara manipulator of tea picking robot," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 2621–2625. DOI: 10.23919/CCC50068.2020.9189044

[27] R. F. A. Khan, K. Rsetam, Z. Cao, and Z. Man, "Singular perturbation-based adaptive integral sliding mode control for flexible joint robots," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 10, pp. 10 516–10 525, 2023. DOI: 10.1109/TIE.2022.3222684

[28] M. Ghafarian, B. Shirinzadeh, A. Al-Jodah, and T. K. Das, "Adaptive fuzzy sliding mode control for high-precision motion tracking of a multi-dof micro/nano manipulator," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4313–4320, 2020. DOI: 10.1109/LRA.2020.2996065

[29] Y. Chunyu, W. Xiao, Z. Xin, and Z. Linna, "Safe trajectory tracking of manipulator based on sliding mode control," in *2020 7th International Conference on Information, Cybernetics, and Computational Social Systems (ICCSS)*, 2020, pp. 353–358. DOI: 10.1109/ICCSS52145.2020.9336908

[30] C. E. Boudjedir, M. Bouri, and D. Boukhetala, "An enhanced adaptive time delay control-based integral sliding mode for trajectory tracking of robot manipulators," *IEEE Trans-*

actions on Control Systems Technology, vol. 31, no. 3, pp. 1042–1050, 2023. DOI: 10.1109/TCST.2022.3208491

[31] X. Wang and J. Qian, "Adaptive sliding mode neural network-based composite control of robot manipulators for trajectory tracking," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 434–439. DOI: 10.1109/CCDC49329.2020.9164305

[32] S. Jiang, J. Zhao, F. Xie, J. Fu, X. Wang, and Z. Li, "A novel adaptive sliding mode control for manipulator with external disturbance," in *2018 37th Chinese Control Conference (CCC)*, 2018, pp. 3024–3028. DOI: 10.23919/ChiCC.2018.8483990

[33] R. Prakash, L. Behera, S. Mohan, and S. Jagannathan, "Dual-loop optimal control of a robot manipulator and its application in warehouse automation," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 262–279, 2022. DOI: 10.1109/TASE.2020.3027394

[34] M. A. Arteaga-Peréz, J. Pliego-Jiménez, and J. G. Romero, "Experimental results on the robust and adaptive control of robot manipulators without velocity measurements," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2770–2773, 2020. DOI: 10.1109/TCST.2019.2945915

[35] B. Esmaeili, M. Salim, M. Baradarannia, and A. Farzamnia, "Data-driven observer-based model-free adaptive discrete-time terminal sliding mode control of rigid robot manipulators," in *2019 7th International Conference on Robotics and Mechatronics (ICRoM)*, 2019, pp. 432–438. DOI: 10.1109/ICRoM48714.2019.9071819

[36] C. Yang, D. Huang, W. He, and L. Cheng, "Neural control of robot manipulators with trajectory tracking constraints and input saturation," *IEEE Transactions on Neu-*

ral Networks and Learning Systems, vol. 32, no. 9, pp. 4231–4242, 2021. DOI: 10.1109/TNNLS.2020.3017202

[37] 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. DOI: 10.1109/LRA.2020.2975727

[38] L. M.-W. Z. Qiao Nan, Wang Lihui, "The sliding mode controller with improved reaching law for harvesting robots," *Journal of Intelligent Robotic Systems*, 2021. DOI: 10.1109/TCYB.2016.2555307

[39] M. Van, S. S. Ge, and H. Ren, "Finite time fault tolerant control for robot manipulators using time delay estimation and continuous nonsingular fast terminal sliding mode control," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1681–1693, 2017. DOI: 10.1109/TCYB.2016.2555307

[40] Z. Li and J. Zhai, "Event-triggered-based sliding mode asymptotic tracking control of robotic manipulators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 3, pp. 1266–1270, 2024. DOI: 10.1109/TCSII.2023.3319645

[41] X. Wang, A. Wang, X. Wang, and D. Wang, "Trajectory tracking control of industrial manipulator using adaptive type-2 fuzzy sliding mode controller," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 3621–3626. DOI: 10.23919/CCC50068.2020.9188960

[42] C. Yang, R. Yang, J. Na, and F. Chen, "Adaptive rbfnn control of robot manipulators with finite-time convergence," in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 42–47. DOI: 10.1109/IECON.2016.7792980

[43] C. Zhu, Y. Jiang, and C. Yang, "Fixed-time neural control of robot manipulator with global stability and guaranteed transient performance," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 1, pp. 803–812, 2023. DOI: 10.1109/TIE.2022.3156037

[44] L. Zhang and L. Cheng, "An adaptive neural network control method for robotic manipulators trajectory tracking," in *2019 Chinese Control And Decision Conference (CCDC)*, 2019, pp. 4839–4844. DOI: 10.1109/CCDC.2019.8832715

[45] Y. Sun, Y. Gao, Y. Zhao, Z. Liu, J. Wang, J. Kuang, F. Yan, and J. Liu, "Neural network-based tracking control of uncertain robotic systems: Predefined-time nonsingular terminal sliding-mode approach," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 10, pp. 10 510–10 520, 2022. DOI: 10.1109/TIE.2022.3161810

[46] W. He, Y. Sun, Z. Yan, C. Yang, Z. Li, and O. Kaynak, "Disturbance observer-based neural network control of cooperative multiple manipulators with input saturation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1735–1746, 2020. DOI: 10.1109/TNNLS.2019.2923241

[47] D. Pavlichenko and S. Behnke, "Real-robot deep reinforcement learning: Improving trajectory tracking of flexible-joint manipulator with reference correction," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 2671–2677. DOI: 10.1109/ICRA46639.2022.9812023

[48] Z. Xu, W. Huang, Z. Li, L. Hu, and P. Lu, "Nonlinear nonsingular fast terminal sliding mode control using deep deterministic policy gradient," *Applied Sciences*, vol. 11, no. 10, 2021. DOI: 10.3390/app11104685. https://www.mdpi.com/2076-3417/11/10/4685

[49] S. Cao, L. Sun, J. Jiang, and Z. Zuo, "Reinforcement learning-based fixed-time trajectory tracking control for uncertain robotic manipulators with input saturation," *IEEE Transac-*

tions on Neural Networks and Learning Systems, vol. 34, no. 8, pp. 4584–4595, 2023. DOI: 10.1109/TNNLS.2021.3116713

[50] C. Wang, W. Cheng, H. Fu, and Z. Zhang, "Disturbance-observer-based tracking control for a two-link robotic manipulators with external disturbances," in *2019 Chinese Control Conference (CCC)*, 2019, pp. 491–495. DOI: 10.23919/ChiCC.2019.8865252

[51] D. Chwa and H. Kwon, "Nonlinear robust control of unknown robot manipulator systems with actuators and disturbances using system identification and integral sliding mode disturbance observer," *IEEE Access*, vol. 10, pp. 35 410–35 421, 2022. DOI: 10.1109/ACCESS.2022.3163306

[52] W. Zheng and M. Chen, "Sliding mode tracking control of a two-link robotic manipulator using nonlinear disturbance observer," in *2017 4th International Conference on Information, Cybernetics and Computational Social Systems (ICCSS)*, 2017, pp. 192–197. DOI: 10.1109/ICCSS.2017.8091410

[53] W. Cheng, Z. Zhang, and Y. Wu, "Disturbance-observer-based tracking control for a flexible-joint robotic manipulator with external disturbance," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 3406–3409. DOI: 10.1109/CCDC49329.2020.9164122

[54] M. A. Y. Abdallah and R. Fareh, "Tracking control of serial robot manipulator using active disturbance rejection control," in *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 2019, pp. 1–5. DOI: 10.1109/ICASET.2019.8714470

[55] A. J. Humaidi, H. M. Badr, and A. R. Ajil, "Design of active disturbance rejection control for single-link flexible joint robot manipulator," in *2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018, pp. 452–457. DOI: 10.1109/ICSTCC.2018.8540652

[56] Y. Lu, L. Han, J. Liu, and S. Li, "Model predictive tracking control for rigid manipulators with disturbance rejection," in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 2705–2710. DOI: 10.23919/CCC55666.2022.9902123

[57] R.-D. Xi, X. Xiao, T.-N. Ma, and Z.-X. Yang, "Adaptive sliding mode disturbance observer based robust control for robot manipulators towards assembly assistance," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6139–6146, 2022. DOI: 10.1109/LRA.2022.3164448

[58] Y. Huang, L. Cheng, Z. Li, W. Gao, H. Lu, and L. Wei, "Backstepping sliding mode control for robot manipulator via nonlinear disturbance observer," in *2019 Chinese Control Conference (CCC)*, 2019, pp. 3220–3224. DOI: 10.23919/ChiCC.2019.8865316

[59] X. Yao, L. Zhang, and W. X. Zheng, "Uncertain disturbance rejection and attenuation for semi-markov jump systems with application to 2-degree-freedom robot arm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 9, pp. 3836–3845, 2021. DOI: 10.1109/TCSI.2021.3091533

[60] J. Su, J. Yang, and S. Li, "Finite-time disturbance rejection control for robotic manipulators based on sliding mode differentiator," in *2013 25th Chinese Control and Decision Conference (CCDC)*, 2013, pp. 3844–3849. DOI: 10.1109/CCDC.2013.6561619

[61] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 1st ed. Wiley, 2006.

[62] W. He, Y. Chen, and Z. Yin, "Adaptive neural network control of an uncertain robot with full-state constraints," *IEEE Transactions on Cybernetics*, vol. 46, no. 3, pp. 620–629, 2016. DOI: 10.1109/TCYB.2015.2411285

[63] J. Liu and X. Wang, *Advanced Sliding Mode Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 81–96. https://doi.org/10.1007/978-3-642-20907-9_3

[64] P. Podder, M. Mehedi Hasan, M. Rafiqul Islam, and M. Sayeed, "Design and implementation of butterworth, chebyshev-i and elliptic filter for speech signal analysis," *International Journal of Computer Applications*, vol. 98, no. 7, p. 12–18, July 2014. DOI: 10.5120/17195-7390. http://dx.doi.org/10.5120/17195-7390

[65] X. Yin, L. Pan, and S. Cai, "Robust adaptive fuzzy sliding mode trajectory tracking control for serial robotic manipulators," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 101884, 2021. DOI: https://doi.org/10.1016/j.rcim.2019.101884. https://www.sciencedirect.com/science/article/pii/S0736584518302060

[66] D. Yao, H. Li, R. Lu, and Y. Shi, "Distributed sliding-mode tracking control of second-order nonlinear multiagent systems: An event-triggered approach," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3892–3902, 2020. DOI: 10.1109/TCYB.2019.2963087

[67] M. Al Saaideh, A. M. Boker, and M. Al Janaideh, "Robust output feedback controller for a serial robotic manipulator with unknown nonlinearities and external disturbances," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5298–5303. DOI: 10.1109/ICRA48891.2023.10160921

[68] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *CoRR*, vol. abs/1804.05172, 2018. http://arxiv.org/abs/1804.05172

[69] Bouguet, J. Y., "Camera Calibration Toolbox for Matlab," https://www.mathworks.com/help/vision/ref/cameracalibrator-app.html (2024/03/12).

[70] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013. DOI: 10.15607/RSS.2013.IX.012

[71] S. Afaq and S. Rao, "Significance of epochs on training a neural network," *International Journal of Scientific and Technology Research*, vol. 9, no. 06, pp. 485–488, 2020.