Robot Dynamic Path Modification Via Multi-Sensor Data Tracking

by

Tarek Ahmed Fathy Mohamed KHALED

THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, JANUARY 19, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ACKNOWLEDGEMENTS

My first thanks go to my thesis supervisor, Professor Ouassima Akhrif, who accompanied and directed me along this work. I am grateful for her availability, her enthusiasm, her confidence, and her generous help in some of my difficult moments. From the first day, she spared no effort to show up in a time of crisis, she provides all the necessary time, moral, and scientific support necessary for each stage which words fail to thank. Without her support in different situations, the achievement of the present work would have not been possible. I want to thank as well my co-supervisor, Professor Ilian Bonev for his practical and permanent availability, his work hand in hand to prepare practically required experimental setup and provide all their physical and financial requirements, besides his daily monitoring of my work, and his valuable suggestions. I thank him very much for that incredible personal and financial support. I am honored by the presence of all the members of the committee who provided useful advice that helped me to structure my studies during my thesis. I wish to express my gratitude to all my committee members who agreed to review and evaluate this research. Moreover, my thanks would be incomplete, if I do not mention my parents: Ahmed Khaled and Nagwa Raslan, who suffer from my alienation especially in their old age which needs all my support, my wife: Abeer Radwan, who left her life behind and came to accompany me on this difficult journey, as well as my children: Jana Khaled, and Adam Khaled, in addition to all my family, for their unconditional love and support in all possible ways. Finally, my most sincere thanks to my friends and colleagues who have helped me, from near or far, directly or indirectly in the development of this work. Special thanks to the members of ÉTS Control and Robotics Laboratory (CoRo lab).

Modification de chemin dynamique via le suivi des données multi-capteurs

Tarek Ahmed Fathy Mohamed KHALED

RÉSUMÉ

Cette thèse de doctorat propose et valide expérimentalement des stratégies de contrôle non linéaires pour un robot industriel six axes Meca500 de la société Mecademic en utilisant deux approches. La première approche développée vise à fournir un suivi dynamique précis de la trajectoire dans l'espace des vitesses cartésiennes en présence d'incertitude des paramètres et de perturbations indésirables. Alors que l'étalonnage du robot peut améliorer la précision de la position et de l'orientation (pose), le seul moyen pour un utilisateur d'améliorer la précision de la trajectoire dynamique est de «guider» le robot à l'aide d'un capteur externe et d'un algorithme de contrôle fonctionnant sur un ordinateur séparé. À cet effet, les robots industriels, qui sont normalement commandés avec des instructions préprogrammées en mode position, offrent parfois la possibilité de modifier à la volée la pose de l'effecteur terminal du robot. Dans le robot industriel Meca500 de Mecademic, les opérateurs peuvent modifier indirectement la pose de l'effecteur terminal en contrôlant l'articulation du robot ou la vitesse cartésienne. Dans cette thèse, une application pratique d'un schéma de contrôle actif de rejet des perturbations (ADRC) est présentée pour améliorer la précision de trajectoire du Meca500. La correction de trajectoire dynamique est obtenue en mesurant d'abord la distance entre un point fixe et l'infobulle du robot avec un transducteur linéaire (barre à billes QC20-W de Renishaw), puis en transmettant le vecteur de vitesse de l'infobulle au robot (via Ethernet TCP / IP). La précision de la trajectoire (circulaire) du robot est considérablement améliorée pour différentes vitesses TCP du robot. Par exemple, à 50 mm / s, l'erreur radiale maximale est inférieure à 0,100 mm et l'erreur moyenne est de 0,015 mm.

La deuxième approche garantit la manipulation du Meca500 avec une manette à six axes à doigt d'une manière très douce et fluide en évitant le couplage croisé homme-robot, les perturbations du moteur du robot, l'oscillation, le bruit ou la réponse agressive du robot, car la dynamique du système dans ce problème de contrôle de suivi mutuel ainsi que son taux de changement peut changer considérablement en amplitude, en temps et en direction pendant les mouvements normaux du robot. Un nouveau concept de contrôleur à plusieurs étages connectés en série est alors proposé, utilisé et vérifié pratiquement en traçant la réponse mutuelle souris-robot et en capturant deux vidéos pour tous les types de mouvements. Le premier étage du contrôleur utilise la commande floue qui se base sur une description naturelle du comportement du robot par l'humain plutôt qu'une représentation par un modèle mathématique, ce qui donne une solution faisable intégrant des connaissances humaines sur la relation entre l'entrée et la sortie à chaque point de fonctionnement. Le deuxième étage comporte un contrôle d'admittance, connecté en série après le contrôleur flou pour amortir des oscillations qui peuvent apparaître lors de la manipulation du robot. Enfin, et comme troisième étage, un différenciateur de suivi ADRC est incorporé pour filtrer le bruit causé par le contrôle d'admittance, ajouter un retard prévu dans certaines situations de mouvement du robot, puis transformer le signal de position en signal de vitesse qui sera envoyé à l'entrée du robot commandé en mode vitesse. Les résultats expérimentaux montrent les performances améliorées du robot après l’ajout des contrôleurs externes à plusieurs étages.

Mots-clés: Précision du robot, robots industriels, contrôleurs avancés, contrôle actif de rejet des perturbations (ADRC), modification dynamique du chemin, interaction personne-machine, contrôle flou, contrôle d’admittance

Robot Dynamic Path Modification Via Multi-Sensor Data Tracking

Tarek Ahmed Fathy Mohamed KHALED

ABSTRACT

This doctoral thesis proposes and validates experimentally nonlinear control strategies for a six-axis industrial robot Meca500 from Mecademic company using two approaches. The first approach aims to provide a precise dynamic trajectory tracking in Cartesian velocity space in the presence of parameter uncertainty and undesirable disturbances. While robot calibration can improve position and orientation (pose) accuracy, the only way for a user to improve the dynamic path accuracy is by “guiding” the robot with the help of an external sensor and a control algorithm running on a separate computer. For this purpose, industrial robots, which are normally controlled with pre-programmed position-mode instructions, offer sometimes the possibility to modify the pose of the robot end-effector on the fly. In the case of Mecademic’s Meca500 industrial robot, operators can indirectly modify the end-effector pose by controlling the robot joint or Cartesian velocity. In this thesis, a practical application of an active disturbance rejection control (ADRC) scheme is presented to improve the path accuracy of the Meca500. The dynamic path correction is achieved by first measuring the distance between a fixed point and the robot tooltip with a linear transducer (Renishaw’s QC20-W ballbar), and then feeding the tooltip velocity vector to the robot (via Ethernet TCP/IP). The (circular) path accuracy of the robot is significantly improved for different robot TCP velocities. For example, at 50 mm/s, the maximum radial error is less than 0.100 mm, and the mean error is 0.015 mm.

The second approach ensures manipulation of Meca500 with a six-axis finger joystick Space-Mouse in a very soft and smooth manner avoiding human-robot cross-coupling, robot’s motor perturbations, oscillation, noise, or aggressive robot response since the system dynamics in this mutual tracking control problem as well as its rate of change may significantly change in amplitude, time, and direction during the normal robot’s movements. To this end, a novel concept of a multi-stage controller connected in series is proposed and verified practically by plotting the mouse-robot mutual response along with capturing a video for all types of movements. The multi-stage controller starts with a fuzzy logic control block based on a natural description of the mouse-robot system by humans, which is better than creating a mathematical model. This provides a more feasible solution that integrates human knowledge about the relation between the input and the output at each operating point. The next stage is an admittance control block, connected in series and aiming to damp some oscillations that appear during the robot’s manipulation. Finally, an ADRC tracking differentiator is incorporated as a third and final stage to filter the noise caused by the admittance control, add an intended delay in some robot’s motion situations, and transform the position signal into velocity to be sent to the robot in velocity mode. The experimental results show the enhanced robot performance after adding the external multi-stage controller.

Keywords: Robot accuracy, industrial robots, advanced controllers, active disturbance rejection control (ADRC), dynamic path modification, human-machine interaction, fuzzy control, admittance control

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

ADE	Adaptive Differential Evolution
ADRC	Active Disturbance Rejection Control
AI	Artificial Intelligence
AKF	Adaptive Kalman Filter
ANOVA	Analysis Of Variance
BMW	Best Worst Method
CNC	Computer Numerical Control
cobot	collaborative robot
CoRo	Control and Robotics Laboratory
crossentropy	Cross-Entropy loss Function
DOF	n-degree-of-freedom
DPM	Dynamic Path Modification
EBF	Error-Based Form
ESO	Extended State Observer
ÉTS	École de Technologie Supérieure
FLC	Fuzzy Logic Controller
GPI	Generalized Proportional Integral
HRC	Human – Robot Collaboration
iGPS	Global Positioning System

IP	Internet Protocol
IR	Industrial Robot
LED	Light Emitting Diode
MB	Model-Based
MFs	Membership Functions
NLSEF	Non-Linear State Error Feedback
NL	Negative Large
NM	Negative Medium
NMB	Non-Model-Based
NNs	Neural Networks
NS	Negative Small
mse	Mean Square Error
OBF	Output-Based Form
OTG	Online Trajectory Generation
PID	Proportional-Integral-Derivative
PL	Positive Large
PM	Positive Medium
pose	position and orientation
PS	Positive Small
PVL	Parameter Variation Laws

RM	robotic manipulator
sae	Sum of Absolute Error
SCARA	Selective Compliance Assembly Robot Arm
SMC	Sliding Mode Control
SOM	Segment Of Motion
sse	Sum of Squared Error
TCP	Transmission Control Protocol
TD	Tracking Differentiator
TDE	Time-Delay Estimation
TS	Takagi–Sugeno
TSPFF	Tunnel-Shaped Potential Force Filed
VS	Visual Servoing

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

B	Damper
Hz	Hertz (measures/second)
K	Stiffness
kg	Kilogram
M	Inertia
ms	Millisecond
μm	Micrometer
s	Second
t_{SOM}	Time each segment of motion
t_s	Sampling time

INTRODUCTION

Nowadays, there is no doubt that we stand on the outskirts of a revolution known as Industry 4.0, which will introduce a quantum jump to humanity that has never been seen before. The fundamental ways of living, working, and communicating with others will face a significant shift in the sense of generating human free manufacturing environment. Its main objective is to integrate new systems replacing humans to improve industrial production. However, this 4th Industrial Revolution needs to be mature enough for real-life implementation (Oztemel & Gursev, 2020). Robotics is one of the most important exhibitions of the Industry 4.0 technological revolution besides some sensors which are considered another core of this revolution. While many applications that require highly skilled, creative tasks are still being done by humans; it is generally agreed today that robots have already taken the place of humans in the modern manufacturing industry as a factory automation equipment (Manyika, 2012), which fill many disciplines such as material handling, welding, assembly, product inspection, testing, packaging, labeling, palletizing and pick and place. Comparing humans with robots that are located on the production line, it is possible to produce anything faster, cheaper, and with higher quality with robots than it is with humans. Moreover, they can operate in environments with dangerous hazards such as painting, chemical, and nuclear reactors.

Since the majority of robot applications require precise static positioning and fast but not necessarily precise transition from one position to another (e.g., as in pick and place), most robot users need only high repeatability. The typical industrial robot (IR) can repeatedly position its end-effector to a pre-taught position with precision ranging from 0.005 mm to 0.1 mm. Many advanced applications, however, call for accurate dynamic positioning or even accurate path following. These advanced applications are typically in the aerospace, automotive, or consumer electronics sectors. For instance, high position and orientation (pose) accuracy is required in inspection applications, where a sensor such as a touch probe or an Eddy-current probe is mounted on the end-effector of a robot (Greenway, 2000) or where the robot must present a complex part in various orientations to a camera or a 3D sensor. In such applications, users either spend a large amount of time adjusting the desired poses of the robot end-effector or calibrate their robot (and ideally the whole robot cell). This leads the manufacturing society to invest in research and development to cope with customer needs and manufacturer requirements. This is exactly why the control technology of the robots needs to be enhanced to reach the required precision.

In this sense, the thesis aim is to contribute to better robotic development by designing and implementing external advanced controllers. Particularly, there are two objectives of this thesis for the industrial robot system: firstly to promote its dynamic path accuracy in a precise and smooth manner, using real-time feedback data from an external sensor, and secondly to facilitate and improve the human-robot interaction during the operation of hand-guiding the robot, i.e., when the robot operator attempts to manipulate the robot in its workspace at different velocities. To this end, two novel controllers are proposed: the first introduces an innovative approach of designing and implementing a practical active disturbance rejection control (ADRC) scheme in the error-based form (EBF) to boost the accuracy of a Meca500 industrial robot. The regenerated corrected trajectory developed by the ADRC controller improves the robot accuracy on-the-fly without any movement interruption with excellent disturbance rejection capability and a dynamic path accuracy in a range of microns. For the second objective of hand-guiding, a second controller, referred to as a multi-stage controller is proposed to improve efficiently the human-robot interaction. This controller consists of three consecutive blocks connected in series, fuzzy, admittance, and ADRC each with a different function, aimed at ensuring that the human operator can easily manipulate the robot without any aggressive motion, noise, mutual motion difficulties, and oscillations both at low and high speeds.

CHAPTER 1

RESEARCH PROBLEM

1.1 Research challenges

The first challenge: Manufacturers realize that industrial robots are highly repeatable but have low accuracy. It is easy to find repeatability information under industrial robot specifications in manufacturer manuals since robots rely on repeatability to perform their repeated programmed missions; however, one seldom finds any information about the robot's accuracy. Static calibration might enhance the robot accuracy, though the best static industrial robot accuracy in most cases is with an error value equal to more than 300 μm. Furthermore, robot accuracy under a dynamic motion becomes even worse with an error reaching 1 mm or even higher, at high robot velocities.

The second challenge: Industrial robots in the market come with their manufacturer embedded controllers; consequently, the robot system is usually considered as a black box. The robot and the embedded controller's exact dynamic model and their parameters are unknown to the user, as are some of the detailed information behind the embedded control software. This is considered a serious issue that limits the use of high-performance model-based controllers. It also adds uncertainties and complexity to the system from the control point of view because the industrial robot control system has to deal with the nonlinearity in the industrial robot as well as the non-parametric modeling errors due to the manufacturer's controller.

The third challenge: The only approach for a user to improve the dynamic path accuracy is by steering the robot through the use of external sensors and a control algorithm that operates from a separate computer. Towards this end, industrial robots, which are normally controlled with pre-programmed position-mode instructions, in some circumstances put forward the prospect of modifying the position and orientation (pose) of the robot end-effector on the fly. Unfortunately, most external sensors are either noisy or can transmit data at relatively low frequencies (e.g., 10 Hz), which in turn affects the desired accuracy of the robot.

The fourth challenge: The extremely small six-axis industrial robot Meca500 from Mecademic is used in this thesis while moving in a circular trajectory with different linear velocities. This robot identification and modeling are indispensable for simulation to test the advanced control techniques that can be useful before starting its implementation. The main problem is that the difference between the robot input and output velocity vectors command is very small; therefore, it is very hard to be identified, because generally the dynamic difference between system input and output help to precisely identify the required system.

The fifth challenge: Human manipulation of an industrial robot through the control of a finger 6-axis SpaceMouse joystick is a very complicated and tricky task especially at high robot speeds, regardless of the user experience. While using a human-machine interface gives the robot the opportunity to follow the human inputs, it is limited by the robot's slow-motion speeds. It is therefore required to give the robot the opportunity to interpret smoothly human motions and to softly transition from low to high speeds and vice-versa, besides precise movement at robot's low speeds and not to just blindly follow them. The main challenge is that the SpaceMouse is mounted on the robot's TCP end-effector forming an interactive collaboration between the human and the mouse, which means that any small or large mouse cap deflections will cause the robot's TCP to immediately respond to this move while the user is still controlling the mouse. This will affect the required position by the user because of difficulties related to robot-mouse mutual contact's nonlinear behavior, especially for high robot speeds. The human-robot interaction suffers from harsh and aggressive movement especially during the transition between different speeds, in addition to the robot's noise and motor perturbations that are affecting the whole movement.

1.2 Research objectives and methodology

This research will tackle the aforementioned challenges using two proposed essential control configurations: The first configuration aims to ensure precision tracking with very high accuracy and the second configuration proposes to achieve intelligent human-machine interfacing with very smooth, comfortable, no noise, and oscillation free movement. Both configurations will be implemented and validated experimentally on the available robotic system and the devices in ÉTS Control and Robotics Laboratory (CoRo's lab).

1.2.1 Objectives

The first objective: Characterization and analysis of the industrial robot and sensors used in this research are required to identify practically their specifications, sampling time, and to determine their available methods of communications, besides, determining and measuring the practical robot's circular errors, as well as the robot-mouse mutual interaction problems and issues. This phase is extremely important because it allows us to identify the best control technique to be used, tested, and implemented.

The second objective: Robot-controller identification during the robot's different practical experiments is imperative, by collecting the robot's input/output data then using the collected data, different models may be generated. The best-generated model is then chosen based on the data analysis. The model performance is required to be tested with new data sets.

The third objective: Design and implementation of a practical controller based on advanced control techniques, capable of improving the robot accuracy on the fly without any movement interruption, providing good disturbance rejection capability and guaranteeing that the robot's trajectory tracking is fast and highly accurate by industry precision standards (in a range of microns). Moreover, the controller should be able to filter the noise influencing the robot at low and high speeds using an external sensor.

The fourth objective: Design and implementation of a multi-stage controller based on advanced control techniques, using the inferred human intention to give the robot the opportunity to interpret the human motions precisely at low and high robot's speeds and improve efficiently the human-robot interaction. The developed control algorithms should ensure that the human operator can easily manipulate the robot without any aggressive motion, noise, mutual motion difficulties, and oscillations both at low and high speeds.

1.2.2 Methodology

To achieve the previously mentioned objectives, various approaches are introduced throughout this thesis. Hereafter, the different methodologies are classified:

Addressing the first objective: Chapter 3 puts forward the industrial robot of interest (Meca500 from Mecademic) technical specifications, movement control commands, and the different types of communications. In addition, the external feedback sensor (Renishaw's QC20-W ballbar distance-based sensor) technical characteristics are presented, then different practical experiments are established to determine the best method to use this sensor in our real-time application, including the sampling time computations. Next, two online circular trajectory methods are investigated to ascertain the best method for Meca500 to follow the prospective circular path in accordance with the experiment's environment, sensor, and robot constraints. Furthermore, the six-axis SpaceMouse joystick is studied and characterized in detail along with the joystick-robot communications through MATLAB environment, and the robot-joystick mutual control constraints and difficulties.

Addressing the second objective: Chapter I is anticipated to identify Meca500 industrial robot while moving in a circular trajectory at different robot speeds. The robot identification is required for the purpose of simulation to check the advanced control different techniques, tuning the proposed controller, and test its robustness before starting the implementation phase. Accordingly, a static feed-forward neural network approach is employed for the purpose of this robot identification. The input and output data are gathered during the robot's various practical experiments, then in accordance with the collected data, distinct neural models are developed by interchanging between many parameters such as the training optimizer options, number of neurons, performance functions, activation functions, and training algorithms. The superior neural generated model is selected based on data analysis, then its performance is confirmed with new data sets. It has been shown that the best chosen neural network model can accurately model the robot system behavior and emulate the output signals with acceptable ranges of error.

Addressing the third objective: To achieve this objective, a practical controller based upon active disturbance rejection control (ADRC) advanced control technique is introduced. Chapter 4 presents the design of ADRC output-based form, as well as ADRC error-based form, both have the advantages of being non-model-based, simple in implementation with a good disturbance rejection capability, improved transient response, and intuitive tuning capability. The ADRC used here for control design is not the standard one. It is a new ADRC error-based form, which deals only with the error regardless of both the system order/model and the desired trajectory. The implementation was done by setting the controller reference input as the desired error with zero values and dealing with the robot error measured by the ballbar in real-time as the controller output, which is being added to the original robot's desired trajectory as a control action to correct robot dynamic path trajectory. The regenerated corrected trajectory developed by the ADRC controller improves the robot's accuracy on-the-fly without any movement interruption.

Addressing the fourth objective: A new approach towards controller design is presented using an ensemble of six SISO multi-stage controllers one for each movement (three translations and three rotations). The objective is to drive the Meca500 robot's end-effector by human finger mouse manipulation. The robot's movement is required to be very smooth and to be derived in a soft manner avoiding human-robot cross-coupling, robot's motor perturbations, oscillation, noise, or aggressive robot response since the system dynamics in this mutual tracking control problem as well as its rate of change may significantly change in amplitude, time, and direction during the normal robot's movements. The novelty of the proposed controller approach is combining different blocks or stages connected in series such as fuzzy, admittance, and ADRC tracking differentiator controllers as introduced in Figure 5.1 to solve the mouse-robot interaction issues.

1.3 Outlines of the thesis

The organization of this thesis is given as follows: Chapter 1 outlines the research problem including the different challenges, objectives, an overview of methodologies, and the originality of the work. Subsequently, in Chapter 2, the state-of-the-art of the existing literature in this area of research is taken into account and declared. Chapter 3 characterizes the industrial robot system used in this thesis, as well as the different types of sensors utilized in accordance with the robot. In addition, the two various methods of robot motion in circular trajectories are illustrated, and the best method is chosen based on the outcomes of each method. Thereafter, the design of the advanced ADRC controller for enhancing the robot accuracy and its implementation is stated in Chapter 4. Chapter 5 introduces the design and the novel implementation of a multi-stage controller to cope with the robot-human interaction different challenges. An epilogue, the conclusion, and future work are illustrated in the final chapter. Following this, an appendix I on the identification and neural networks modeling of Meca500 while moving in a circular trajectory at different robot speeds is presented.

CHAPTER 2

LITERATURE REVIEW

In recent years, there has been considerable interest in path tracking control of industrial robots. Many researchers have studied industrial robot tracking-related issues such as motion control, offline and online path planning, accuracy enhancement, and dynamic trajectory tracking (Gharaaty, Shu, Joubair, Xie & Bonev, 2018; Kubela, Pochyly & Singule, 2019). In these studies, researchers investigated advanced controllers to achieve the required accuracy for different applications. Various types of external sensors were used in these studies to get the pose of end-effector in real-time which can improve the robot accuracy. Unfortunately, according to manufacturers' requirements, strategies for solving industrial robots control and accuracy problems are still badly needed in the light of the industrial robots' non-linearity, complexity, and dynamic coupling. In addition to the requirement of high accuracy, another subject of great interest to the robot's manufacturers in the last decades is the research on human-machine interaction and the way it addresses applications such as the so-called robot's hand-guiding (Siciliano & Khatib, 2016). Hand-guiding has become a very popular feature in robotics in the last decade as it simplifies significantly the programming of robots, especially in applications where complex paths are to be taught. Such applications include welding, dispensing, and non-contact inspection. In hand-guiding, an operator grasps the end-effector of the robot and moves it in space, while the robot follows with as little resistance as possible. In some cases, it is very difficult to program the robot to achieve the task; instead, it is very easy to teach the robot by human manipulation the required path and positions, then to let the robot repeat the task if needed. Several external sensors can be used to do the aforementioned task; however, this remains a challenge. For instance, using a finger 6-axis 3D mouse to drive the robot's end-effector target to its desired position is not a trivial task, because the interaction should deliver robots intelligent information about the environment that cannot be obtained by simply using external sensors. Furthermore, the human-end effector nonlinear contact as well as the contacts' cross-coupling, and sensor noise may cause oscillations in robot movement which cannot be handled by classical control techniques (Chen & Liu, 2013; Hogan, 1984). Other approaches should be established, dealing with the switching between high and low robot speeds. Therefore, this literature review is concerned with advanced control techniques used by different researchers to emphasize novel solutions for the two scopes of this thesis; tracking control precision issues for industrial robots, and human-machine interaction control applications such as robot hand-guiding. Firstly, a brief introduction to industrial robots is presented along with different types of robots generally used in industry and their programming methods. Besides, the main sources of error which affect the precision of the industrial robots are addressed. Secondly, work concerned with industrial robot's accuracy enhancement is illustrated together with several advanced control techniques used in the literature to address the concerned challenges. Next, this literature review will cover the area of human-machine interaction in order to identify the best possible solutions for the aforementioned challenges. Various authors and many applications use force and torque control as a solution. Our objective is to develop another approach that is more practically convenient especially since in our case, the mouse is mounted on the robot's end-effector itself. Therefore, other controllers are investigated in this literature such as fuzzy and ADRC controllers.

2.1 Industrial Robots

Robots are machines that can carry out different kinds of complex jobs and tasks. Hence, there are many types of robots that have been illustrated in Table 2.1. The most popular and commonly used robot configurations are stationary robots like Cartesian robots, vertically articulated robots, parallel robots, dual-arm robots and, Selective Compliance Assembly Robot Arm (SCARA) as shown in Figure 2.1. Generally, most industrial robots fall into the category of robotic arm system which is composed of an n-degree-of-freedom (DOF) robotic manipulator (RM) mounted on a base. Industrial robots (IRs) include the manipulation ability of a fixed-based manipulator to execute multiple tasks. These systems contain arms (links) mounted on a fixed base, offering the robot the ability to repeatedly accomplish specific tasks with high precision (accuracy and repeatability). In comparison with other IR types, articulated IR could reach their workspace very easily with different orientations, just as a human arm.

Table 2.1 Classifications of general types of robots
Taken from Gencer (2014)

Types of robots				
By Kinematics	Stationary Robots	Cartesian Robots	By Application	Industrial
		Cylindrical		Domestic
		Spherical		Medical
		Scara		Service
		Articulated		Military
		Parallel		Entertainment
	Wheeled Robots	Single wheel (ball)		Hobby and competition
		Two wheels		Space
		Three of more wheels		Flying
	Legged Robots	Bipedal (Humanoid)		Mobile Spherical
		Tripedal		Swarm
		Quadrupedal		Swimmimg
		Hexapod		Others
		Others (octopod, centipede, etc.)		

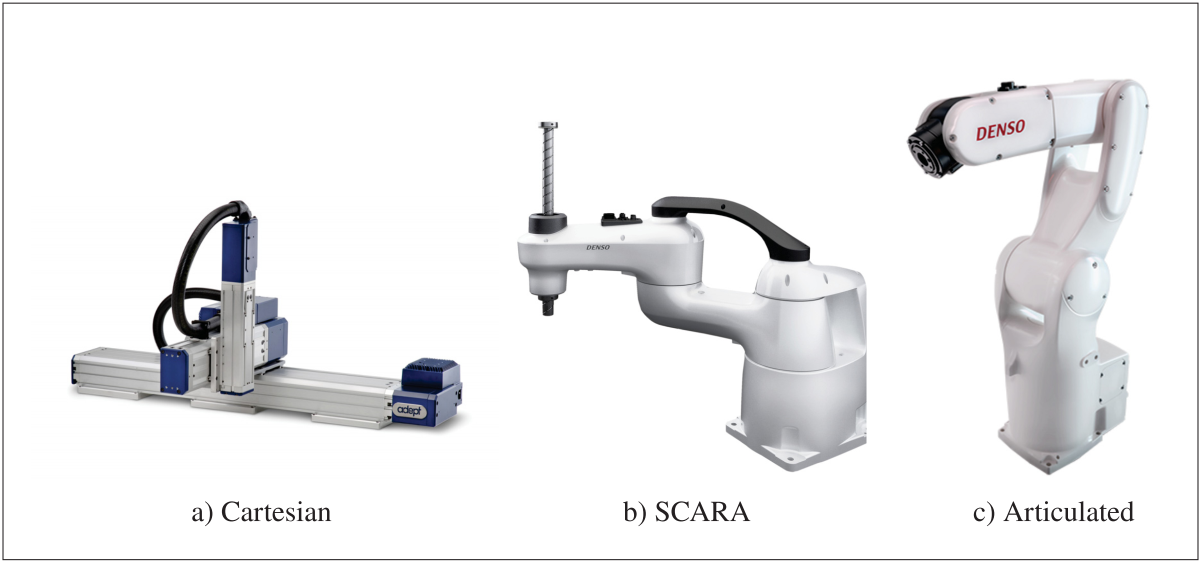


Figure 2.1 Example of robots’ industrial configurations
Taken from Denso (2020)

Industrial robots are taking the hand of human’s day after day in contemporary manufacturing because they offer higher quality, faster response, and the ability to work long times all day and night. They are now considered as basic equipment in the automated production lines offering many solutions to beat manufacturing challenges thanks to their capability of reducing the processing time, variability, increasing productivity, and capacity (Ben-Ari & Mondada, 2018). They can also be used in environments with dangerous hazards such as chemical and nuclear reactors. Typical applications of robots include material handling, welding, painting, assembly, product inspection, testing, packaging, labeling, palletizing, pick and place as presented in Figure 2.2; all these tasks need to be accomplished with high speed, endurance, repeatability, and high accuracy. For more flexibility, some robots are capable of having the same orientation as the objects on which they are operating. Furthermore, some industrial robots contain machine vision subsystems acting as visual sensors in order to monitor the end-effector pose to get a more precise dynamic path, with the help of powerful computers or controllers. Other industrial robots use artificial intelligence to follow a dynamic path or to interpret the human interface. Hence, artificial intelligence has become an increasingly important factor in modern industrial robots.

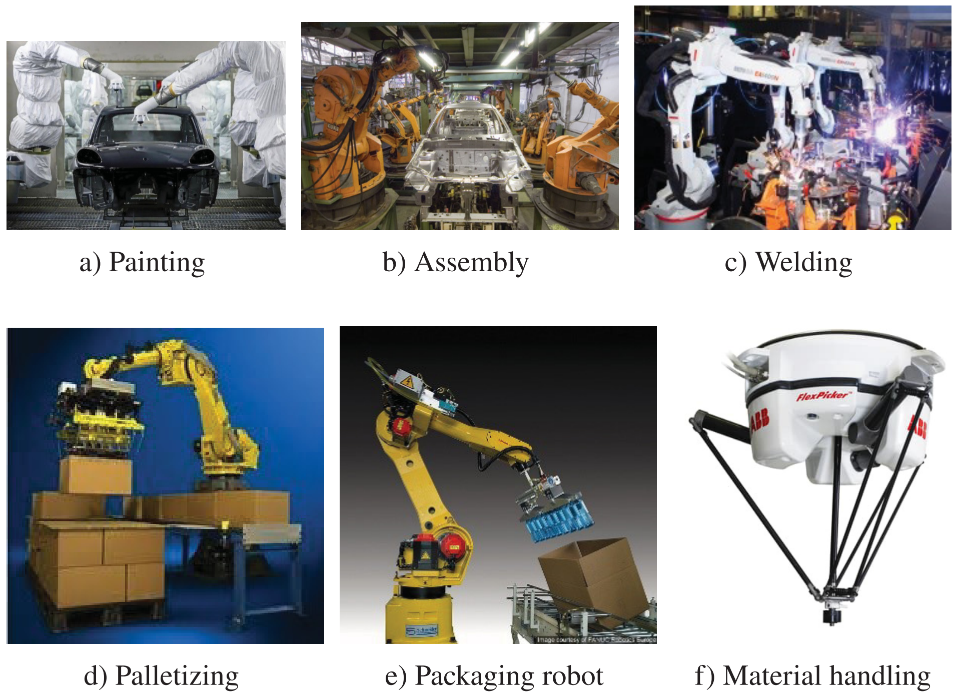


Figure 2.2 Industrial different applications
Taken from RobotWorx (2020)

Unfortunately, while offering very high repeatability, industrial robots still lack high precision accuracy in highly specialized tasks such as gauging, turbine engines metrology, and welding (Manyika, 2012), when a fiber placement head or a laser tool is used as the robot end-effector or in inspection applications, where a sensor such as a touch probe or an Eddy-current probe is mounted on the end-effector of a robot (Greenway, 2000). Another example calling for high path accuracy is when robots are used for spray-painting, where poor accuracy could affect the thickness of the coating and lead to the creation of surface scars and bubbles (Zhang, Wu, Wang & Yu, 2020). Note that pose repeatability is defined as the robot's ability to move back to the same position and orientation. On the other hand, pose accuracy is the robot ability to accurately move to the desired position in 3D space (Shiakolas, Conrad & Yih, 2002; Abderrahim, Khamis, Garrido & Moreno, 2004) as presented in Figure 2.3. The typical industrial robot can repeatedly position its end-effector to a pre-taught position with precision ranging from 0.005 mm to 0.1 mm (Nubiola & Bonev, 2013). Robot positioning errors come from geometric, dynamic, thermal, and system errors (Zhang *et al.*, 2020) as follows:

- Geometric error, due to the difference between the real robot kinematics and the mathematical model programmed inside the robot controller.
- Dynamic error, due to dynamic loads, inertial loading, and structural resonance excited by the motion.
- Thermal error, due to motors, drive mechanism, electronics, and environmental changes.
- System error comes from improper calibration, sensor inaccuracies, drive train backlash, and poorly tuned sensors.

Accordingly, automation and aerospace manufacturers strictly require high robot accuracy in the applications that demand high precision. In such applications (Figure 2.4), users either spend a large amount of time adjusting the desired poses of the robot end-effector or calibrating their robot (and ideally the whole robot cell). However, it is hard to completely solve the accuracy problem, so the aim is just to reduce the error to a desired satisfactory level. To address the robot bad accuracy, two different techniques can be used and are described as follows:

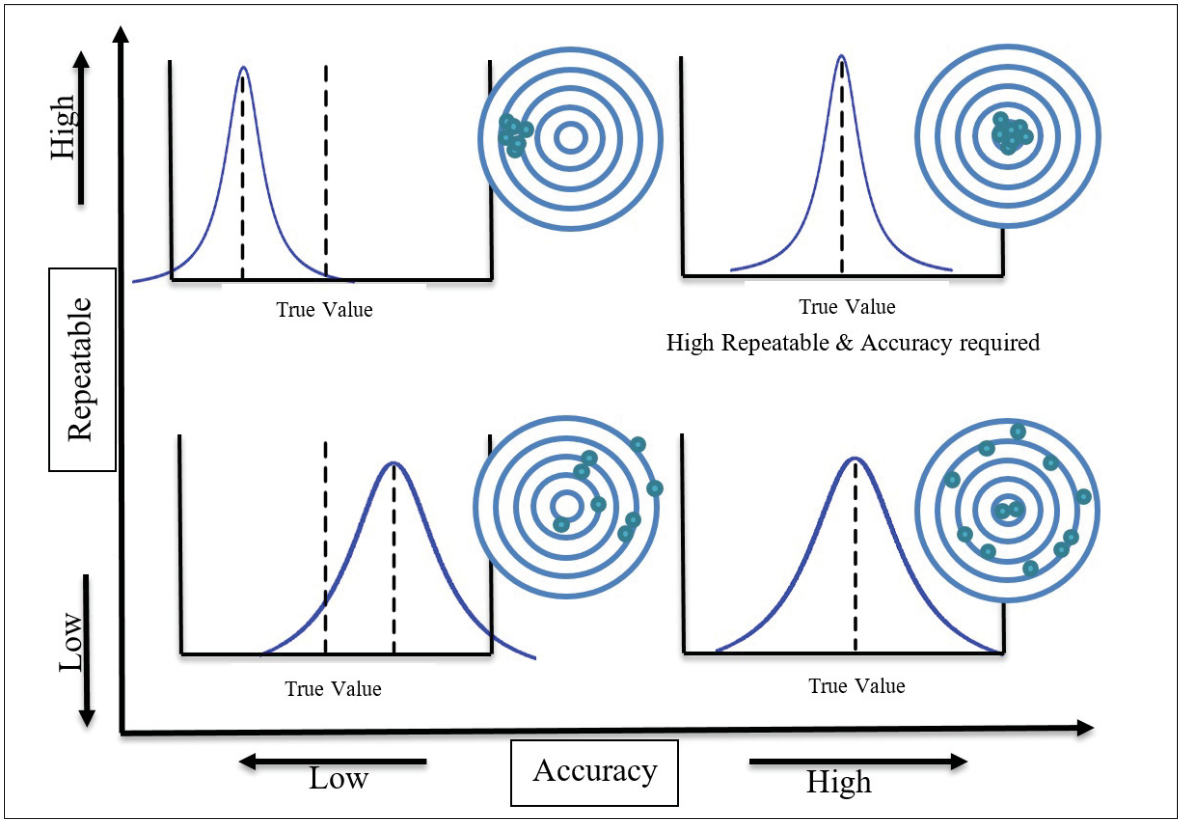


Figure 2.3 Illustration of the difference between robot accuracy and repeatability

- Static robot calibration which is the process of identifying a mathematical model that describes the relationship between robot joint values and robot end-effector pose with higher accuracy than the nominal mathematical model that is embedded in the robot controller (Nubiola & Bonev, 2014). This new mathematical model is either included in the robot controller (when the calibration is performed by the robot manufacturer) or used in an offline programming software (when the calibration is performed by a third party). In both cases, however, calibration makes only static positioning more accurate, since the new mathematical model is applied only for the end poses. Thus, the path between the two poses is not necessarily more accurate than before calibration.
- Adding an external controller which requires that the robot position error be measured by using one or multiple types of external sensors on-the-fly in real-time, then the robot's end-effector reference position is adjusted in the control loop to get the final desired robot's end-effector

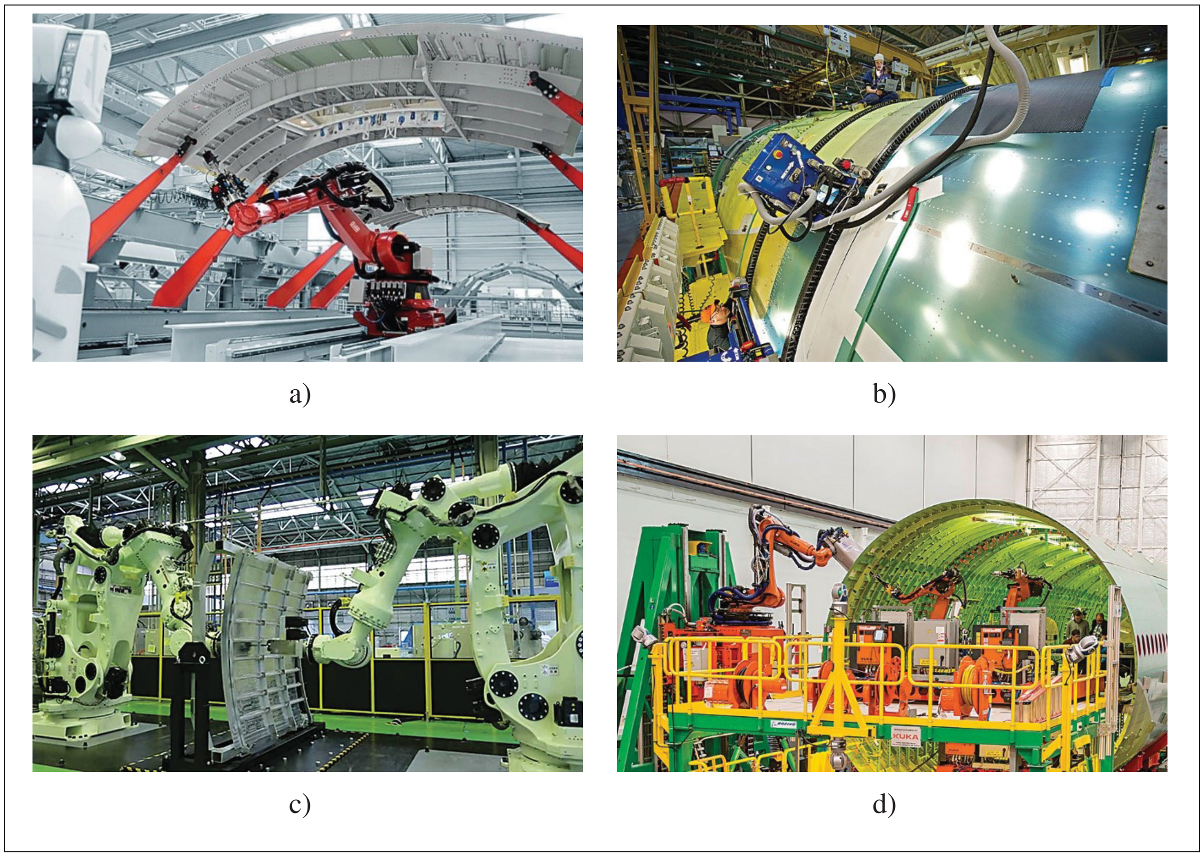


Figure 2.4 Illustration of different applications in aerospace manufactures
Taken from Robotics (2018)

position. This approach enhances the robot's accuracy without any modification in the robot structure or in its manufacturer controller. However, it is important that the proposed external controller be well designed. Moreover, the sensor and controller implementation need to be carefully studied based on different industrial robot types and models. In this thesis, this approach is the one used to obtain the required precision accuracy and achieve better robot path tracking.

2.2 Robot's Accuracy Enhancement

First, while sensing the pose of the robot's end-effector and applying the robot control play a key role in robot coordination task, the resulting precision of the robot is still lower than the requirements of the majority of specific tasks, especially in the aerospace field. The solution for this issue offered by some of the industrial robot manufacturers such as ABB, FANUC, and KUKA is by performing off-site recalculation of the exact kinematic model parameters then calibrating the robot and adding the nominal kinematic parameters of the robot into the controller. Unfortunately, the achieved accuracy gained by this recalculation method is still not sufficient. Furthermore, it is not only a matter of accuracy but also a matter of time because the production needs to be suspended until calibration is done. This is repeated approximately every year depending on the application (Nubiola, Slamani, Joubair & Bonev, 2014). Recently, many researchers have studied the industrial robot's accuracy enhancement using external sensors such as C-Track and laser tracker. In (Gharaaty, Shu, Xie, Joubair & Bonev, 2017), researchers tracked pose of the FANUC M20-iA robot's end-effector in real-time by using photogrammetry based 6D measurement sensor Visual Servoing (VS) C-Track from Creaform and filtered the sensor noise using the root mean square method. An on-line correction using FANUC's dynamic path modification (DPM) option is used to control the robot movement with the help of a Proportional-Integral-Derivative (PID) controller. The robot's accuracy is reduced to 0.050 mm for the position and to 0.050° for the orientation. In parallel, and for the same FANUC M20-iA, researchers in (Shu, Gharaaty, Xie, Joubair & Bonev, 2018) used another technique to filter the C-Track sensor image noises by using an Adaptive Kalman Filter (AKF). The same accuracy is reached, however (0.050 mm and 0.050° for position and orientation respectively).

(Jin, Yu, Li & Ke, 2014) monitored KUKA KR360L280-2 robot's end-effector pose in real-time by using a laser tracker system then calculated the errors and communicated with the robot system via Process Control (OPC) service protocol. The maximum position error was reduced from 1.379 to 0.069 mm, and the maximum orientation error was reduced from 0.712 to 0.013°. On the other hand, in (Schneider, Drust, Diaz Posada & Verl, 2013); the kinematic and dynamic robot models of KUKA were used to predict then compensate the deviations and the online compensation to push the precision beyond 100 μm was achieved thanks to a series of Light Emitting Diodes (LEDs) with three cameras (K600 measurement system), in addition, an improvement of 46.88% was proven by adding a PID controller. Although, these presented results are promising; they do not show the relevance of the approach in high-frequency applications and fast dynamic compensation. (Norman, Schönberg, Gorlach & Schmitt, 2010) used iGPS system as an external position measurement system allowing external control of the robot motions but much more robot interfaces need to be done. (Norman, Schönberg, Gorlach & Schmitt, 2013) presented research work on the use of iGPS for external control of offline programming cooperating robots to achieve position accuracy of ±2 mm. The researchers in (Wang, Mastrogiacomo, Franceschini & Maropoulos, 2011) compared the performance of using iGPS and laser tracker to track KUKA KR240-2 end-effector. The experiments conducted showed that iGPS is clearly more affected by higher robot speed, with distances from the reference trajectory up to 4–5 mm at 1 m/s compared to the 1.2–1.4 mm of the laser tracker but repeatability performances are similar. Laser tracker static and dynamic accuracy are generally better (Wang *et al.*, 2011). Note that all the mentioned methods above used a control scheme such as the one shown in Figure 2.5 that illustrates how the different kind of external sensors can be used to sense the real position of robots' end-effector then to feedback this knowledge to be compared with the desired path and fed to the external controller used by robots manufacturers as a new desired input trajectory (error).

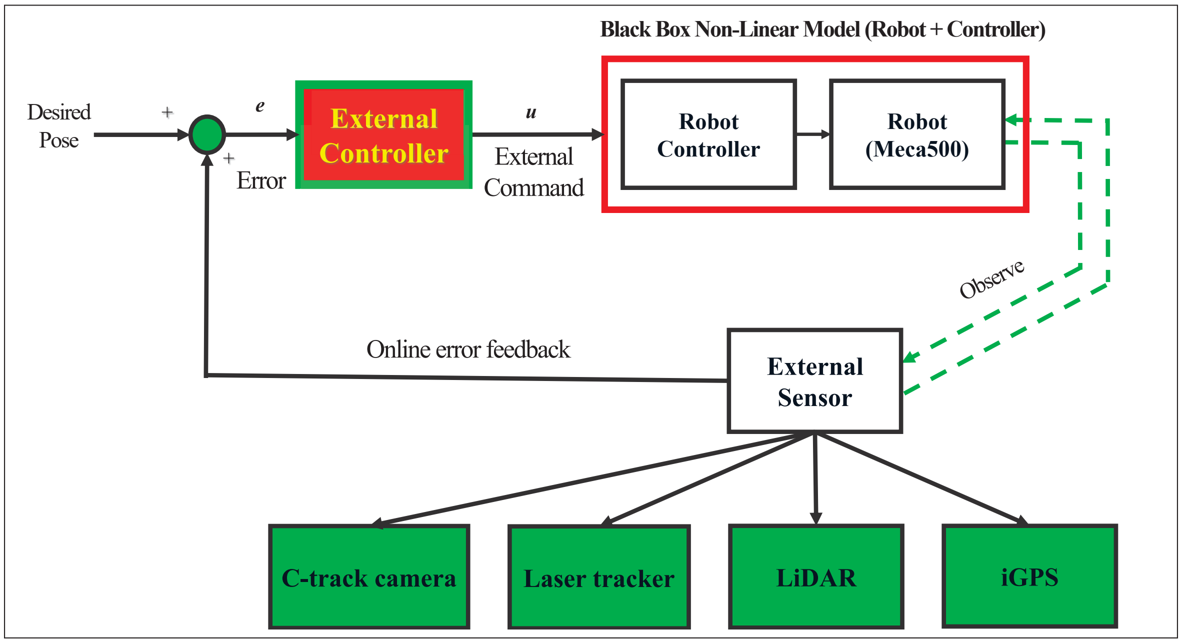


Figure 2.5 The proposed control scheme block diagram with different sensors

To conclude, numerous manufacturers, especially those who work in the aerospace and automotive field, rely on the robot's repeatability but demand more and more accuracy in their high precision applications. For instance, high accuracy is required in gauging, robots to take some measurements such as performing metrology on manufactured parts, in this case, the measuring sensor is mounted on a robot end-effector, thus all the measurements are highly affected with robots position accuracy (Greenway, 2000). Moreover, it is also required when the robots are used for spray-painting, which requires a dynamically high position accuracy with robot's different linear velocities because if not, the thickness of the coating will be affected besides surface scars and bubbles could happen (Zhang *et al.*, 2020). In addition, less error accumulation and high accuracy are an essential demand from aircraft manufacturers when they are using the robot in welding turbine engines metal parts because this task is very critical and accurate, and so on for other various applications.

Moreover and based on the aforementioned discussion, the desired robot trajectory is strictly required to be followed by the robots at different velocities. While the robot accuracy was somewhat enhanced in the above references, the question if combining the use of external sensors with classical linear control techniques used by those researchers gives the most effective way to solve the accuracy problems remains. In other words, is it enough to use classical control techniques to deal with industrial robot accuracy requirements, external sensor noise, and the ambiguity of the embedded manufacturer controller with no access to the robot's real inputs? This thesis proposes to use advanced control techniques to tackle these challenges.

2.3 Robot Hand-Guiding

Hand-guiding has become a very popular feature in robotics in the last decade as it simplifies significantly the programming of robots, especially in applications where complex paths are to be taught. Such applications include welding, dispensing, and non-contact inspection. In hand-guiding, an operator grasps the end-effector of the robot and moves it in space, while the robot follows with as little resistance as possible. Many so-called collaborative robots (cobots) today provide the hand-guiding feature, but most rely on simple readings of the motor currents to sense the intentions of the operator. Best user-experience is achieved when the robot is equipped with both torque sensors in the joints (so that the operator can force individual joint too) and a force/torque sensor at the flange, though excellent results could be achieved when only a force/torque sensor is used.

Of course, user-experience is subjective, and hand-guiding algorithms and their performance differ significantly from one cobot to the other, but even today hand-guiding is still far from the feeling of moving a weightless object in space. In addition, it lacks accuracy and precision. Depending on the robot, there may be problems in reorienting the end-effector or moving it quickly, for example. Some robots may even continue to move briefly after the operator releases the robot. Besides, even a slight mismatch between the actual mass properties of the end-effector and the ones entered by the user can lead to additional problems. Finally, user experience is extremely dependent on where and how the user grabs the end-effector.

Hand-guiding is mainly relying on the contact dynamic interaction between the robot and the surrounding environment that can be inertial, dissipative, or elastic. Thus the use of classical control techniques is not enough to cover the aforementioned issues. However, utilizing the motion control can lead to good results if the task itself is accurately planned, which in turn requires accurate robot's kinematics and dynamics models as well as environmental geometrical and mechanical models. Unfortunately, although the robot's model may be somehow precisely known, the environmental detailed description and its models are difficult to be achieved. In hand-guiding practical application, the positional errors arise because of the interaction contact forces and moments, causing the robot's end-effector deviation from the desired value, on the other hand, the controller reacts hard to keep the tracking and reduce these deviations causing motion oscillations. Consequently, the controller can do an easy task if the mutual compliant behavior between the robot and human interaction is ensured, which can be achieved either in a passive or in an active manner. In the passive approach, the robot end-effector trajectory is revised by the contact forces due to the robot's inherent compliance, i.e., the compliance of the robot's position servos, or the structural compliance of the robot's joints, links, and end-effector. It is a cheap and simple approach because no force/torque sensors are needed;

however, it lacks flexibility, and it can only deal with a small robot's pose. Since no forces are measured, it cannot guarantee that the high contact forces will never occur, in addition, once the end-effector trajectory is planned by the designer, it cannot be changed during the execution time. On the other hand, in the active approach, the contact force and moment measurements are definitely required to be fed to the controller which grants online trajectory modification. Although this approach overcomes the passive drawbacks, it is a complex, slower, and more expensive system. To obtain good performance, however, it can be utilized combined with a certain extent of a passive approach to maintaining the reaction forces lower than a specific threshold (Villani & De Schutter, 2016). Generally, for force control applications, six parameters are essentially required to deliver full-contact force information, i.e., three translation force components and three rotation torques.

Recently, an approach for guiding the industrial robot named sensor-less hand-guiding, which demands precise dynamic parameters (Liu, li, Fang, Han & Zhang, 2020) has been proposed. This force control method relies on the robot's motor current to directly drive the robot, i.e, the force estimation value is calculated based on the robot's motor current. The fundamental concept behind this technique is to accurately estimate the robot parameters such as friction, inertia, and gravity with no need for joints' torque sensors so that the control designer determines the feed-forward action to compensate for the aforementioned robot's parameters during the robot real-time movement. It is a big challenge to compute the friction in all phases, especially when the robot's joint state changes gradually or suddenly while the robot movement changes from moving to stationary or slow movement. Therefore, once these parameters are being accurately identified, the hand-guiding required force becomes much smaller and more relevant. Accordingly, this technique is only valid if the control designer has access to the robot's torque sensors or to its direct-drive motors (Liu *et al.*, 2020).

In (Zhang, Wang, Jing & Tan, 2019), the paper proposed a sensor-less hand-guiding scheme to minimize the external force estimation error by employing a virtual mass and friction model. The 6-DOF robot's dynamic model was built, in addition, the robot's links inertial force and friction are analyzed. A force following control was designed to follow the operator's external forces based on the joint torque which is achieved from the robot's motor current. However, the results were not accurate enough but are acceptable for low speeds, and no results were shown in the paper for high speeds (for safety aspects). The authors in (Lee, Ahn & Song, 2016) had direct access to the 6-DOF robot's joints, so they built the robot dynamic model along with the motor current, in addition, they used the robot's friction model to determine the user's motion intention of the robot end-effector in place of detecting the external user's forces. A sensor-less hand-guiding method relying on torque control was proposed with practical results that ensured its usability; however, the overall sensitivity is lower than the sensor-based technique. The authors still have to build a more accurate robot's friction model to develop the sensor-less approach. In (Moe & Schjølberg, 2013), the presented work developed a communication architecture addressing some challenges, through 5-DOF robot manipulator real-time control via human arm guidance using a cheap sensor (Microsoft Kinect) which was easy to be mounted. The proposed system estimates a user reference signal via hand-guiding, then sends it to the robot controller as a reference. However, it needs more accuracy and responsiveness, and to include a 6-DOF robot in the future to be more relevant. An approach of hand-guiding using a predefined geometric path is introduced in (Hanses, Behrens & Elkmann, 2016), acquiring more precision in surgery applications by splitting the task into offline and online phases, in which joint velocity and acceleration constraints are obligated to prevent actuators saturation effects. This approach was applied only in simulation and no practical work was employed. A new technique to improve the robotic arms hand-guiding is presented in (Müller, Jäkel & Suchy, 2015). A tunnel-shaped potential force field (TSPFF) is introduced to guide the user along a reference trajectory, in which a robust parametrization is identified to enhance the control performance. The simulation results show good control performance with the operator's reaction; however, the practical experimental results deviate from the simulation and the chosen parameters were not valid for the user's comfort. Consequently, the TSPFF still needs to be improved compared to other hand-guiding algorithms. Recently in (Liu *et al.*, 2020), a new friction model is proposed to accurately identify the real robot friction model, then the determining of dynamic parameters related to the external forces by the user is practically evaluated. By doing this, the robot's motion is enhanced especially the joint state's change from moving to stationary or slow movement.

However, an accurate friction model has to be designed to describe the conversion of dynamic and static friction. Although all the aforementioned approaches and control techniques may have good results, they are only applicable if and only if the control designer has access to robots' joints, which is not valid in this study.

On the other side, users do not have access to modify the control algorithms of industrial robots, let alone the robot hardware. Access to controlling directly the robot joint torque is not provided either. Fortunately, some industrial robots offer optional software modules providing different high-level means for controlling the motion of the robot in real-time. For example, ABB Robotics offers the EGM (Externally Guided Motion) option, FANUC Robotics offers the DPM (Dynamic Path Modification) option, and KUKA Robotics offer the RSI option (see (Khaled, Akhrif & Bonev, 2020) for more details). Mecademic, the manufacturer of the Meca500 robot that we use in this study, offers the possibility to control the Cartesian velocity of the robot in real-time. These different options can be used in conjunction with an external force/torque sensor to develop an external controller that governs the hand-guiding feature of the robot.

The interaction control in accordance with sensor-based hand-guiding has been studied in many papers in the last decades, and it earned greater importance with the progress and developments in power electronics, computer power, and sensors (Siciliano & Khatib, 2016; Lahr, Soares, Garcia, Siqueira & Caurin, 2016). The main challenge for this interaction controller and the difficulties facing it come from the mutual interaction between both the sensor used by the operator and the robotics system, i.e., the mutual nonlinear behavior, exactly like our case study. The controller needs to interact with the surrounding environment in a wider range of everyday tasks which are related to direct contact with the environment, such as polishing, drilling, machining, and many other activities (Pires & Bogue, 2009). For that reason, the classical control algorithms are not appropriate to be used with this kind of tasks (Chen & Liu, 2013; Hogan, 1984). Likewise, some other applications can only be implemented through force and torque control, like assembly in precision applications (Chen & Liu, 2013), rehabilitation (Krebs, Hogan, Durfee & Herr, 2006), or surgery assistance (Luo, 2016). This proficiency is needed while using hand-guiding in robotics, in an approach that is not allowing the robot's motion oscillations, the robot's harsh movement, noise, slow and uncomfortable robot's user manipulation. Simultaneously, they must be careful of the amount of the imposed contact forces at certain motion phases, if it exceeds a certain threshold, it will directly cause the aforementioned issues.

The impedance and admittance controllers are considered the most famous control techniques that are addressing the aforementioned control challenges. Since their announcement by Hogan (Hogan, 1984), they gained high interest and became very fast one of the most successful approaches in this field. Despite the fact that many modern applications are designed based on this technique, it is not an easy task to be accomplished practically (Grafakos, Dimeas & Aspragathos, 2016); thus, it is worthwhile to look into an implementation corresponding algorithm. The authors in (Grafakos *et al.*, 2016) gave a clear explanation of implementation on discrete systems, such as micro-controllers, PCs, and industrial robots, they proposed two different implementation methods for the same controller. The methods are successfully modeled, numerically simulated, and validated.

The difficulty arising from the use of the SpaceMouse module comes from the module's high flexibility. That flexibility introduces cross-coupling in the human-robot operation, which causes high oscillations in the robot motion, especially while the Spacemouse is mounted on the robot's end-effector, in addition to the robot's fast movement. Therefore, other approaches should be established, dealing with the switching between high- and low-speed motions associated with the typical hand-guiding process (slow precise motions when teaching a path, followed by faster motion when teaching intermediate poses). To the best of our knowledge, the several advanced control techniques that have been studied in the literature to address the mutual interaction challenges are not enough to be applied in our application, they are not suitable for our case study as was mentioned in (Chen & Liu, 2013; Hogan, 1984). Although the majority of authors and many applications addressed the interaction control by using force and torque control, impedance, stiffness, and admittance control, in this thesis; however, we tackle other techniques of advanced control such as fuzzy logic controllers (FLC) and the active disturbance rejection control (ADRC) that can be employed to emphasize novel solutions for solving the mutual interaction challenges.

2.4 Advanced Control

One of the most important components of robots in general and industrial robots, in particular, is the robot controller, which allows the robot to do its function as well as executing a set of forces and specific motions, it simply regulates the robot's behavior. A good controller determines the interaction level between robot and human then decides on the extent of the robot's versatility and adaptability to human different actions. The question is how to design an efficient and easy-to-use controller capable of controlling the robot very precisely in various types of applications in the existence of unforeseen errors. In the field of industrial robot control, various and different methods have been developed for simulation and real applications by different researchers. In this section, we review the existing types of advanced robot controllers and their implementation, besides the basic role of Artificial Intelligence (AI) is illustrated.

Overall, control systems can be classified into mathematical Model-based (MB) control and practical Non-model-based (NMB) control, each one has its own advantages and disadvantages depending on the robot's desired task and its applications. The MB controller is based on the existence of an accurate dynamic model of the robot which is actually very difficult to build because of the parametric disturbances affecting the robot as laid out in Figure 2.6. Furthermore, the dynamic expression requires a large amount of calculation which is difficult to be achieved in real-time control (Wen, Yu, Zhang, Zhao, Lam, Qin & Wang, 2017). In addition, it is hard to design MB controllers for systems with properties such as time variance, non-linearity, and uncertainty. However, on the other hand, it is very useful and its outcome gives very good results when the model of the robot is well known and is not subjected to high disturbances. On the contrary, the NMB controllers are motion-based controllers that can be used in the case of having real-time data of the robot's end-effector location from available internal and/or external sensors. The greatest advantage of the NMB controllers that they are not designed on the basis of an accurate dynamic model of the system as well as having good robustness to uncertainties. In (Meng, 1995), a comparison study between MB and NMB robot controllers of joint angles position tracking trajectory in terms of error accuracy was done by using 3-types of controllers; classic PD controller computed torque controller, and new adaptive controller.

The highest performance appeared when using the third controller (adaptive controller) that is working with unknown or partially known robot dynamics. It had the advantages of the MB controllers and robustness to uncertainties of the NMB controllers. In (Saied, Chemori, El Rafei, Francis & Pierrot, 2019), high non-linearity, time-varying, and uncertainties were the main reasons that motivated the author to enhance the control of parallel kinematic manipulators. The author demonstrates the strength of MB controllers over NMB controllers by making a comparison between the performances of MB controllers (Augmented PD, and Adaptive Feed-forward with PD) and NMB controllers (PD, PID, and Nonlinear PD) in terms of Cartesian position tracking error and motor torque. This was also fulfilled by real-time experiments on a 4-DOF parallel robot named VELOCE. The results were acceptable, however, the performance of the MB controllers in terms of precision, motion speed, and robustness are still needed to be improved. On the other hand, some other papers demonstrate the strength of NMB controllers over MB controllers, it all depends on the type of application. In (Yang & Jie, 2017), the researchers investigated the ADRC approach for a one-DOF link manipulator, where the extended state observer (ESO) is proposed using a function called fal (Han, 2009) in which the error can reach zero much more quickly in finite time where the gain decreases when the deviation increases. The good effects of the proposed control scheme were proved in simulation and compared to traditional PID, the manipulator follows the desired trajectory more accurately. Researchers in (Yang, Tan & Yue, 2018), combined the backstepping technology with ADRC to illustrate the robustness of the closed-loop system. The proposed NMB system was compared with a MB system with different uncertainties of model parameters and external disturbances. As a result, an improvement of tracking performance was achieved in both tracking accuracy and uncertainty compensation simultaneously by using NMB ADRC controller. Generally, in the case of real-time trajectory applications with a sudden movement and high speed, the NMB is the optimum choice because of the high capability of robustness it has. To conclude, commonly, the trade-off between MB and NMB controllers has been a disputed issue between lots of researchers. The researcher needs to evaluate and choose the best controller on the basis of the system of interest and its applications.

In our case study, since we are dealing with the robot and its controller as a one-unit black box with no direct access to the robot or its joint angles, in addition, our objective is to control the robot dynamically in real-time with the possibility of sudden movement and high speed, we would prefer to work with NMB controller.

2.4.1 Model-based (MB) controllers

Model-based control system is based on mathematical system modeling; it is capable to predict what will happen next, whereas the optimum value is calculated then fed to the model of the plant before being fed to the real plant. It gives a higher general level of view to watch the system and to predict the system behavior. Therefore, the errors can be located and corrected by engineering early in the design thanks to an easy understanding of how the system is behaving. (Wen *et al.*, 2017) used a fuzzy identification and delay compensation based on the force/position hybrid control scheme of the 5-axis parallel robot. This fuzzy identification was applied to solve the parameter uncertainty of the force control problem. The obtained fuzzy model was used in a feedback link and a PI controller was tuned based on this model then a comparison of position tracking curve (with and without MB fuzzy) was made to identify the enhanced accuracy (the error was below 1.5 mm). To compensate for the delay from the input to the output, a Smith predictive compensation method was used, and simulation shows good results. (Abou Elyazed, Mabrouk, Abo Elnor & Mahgoub, 2016) compared the performance of the desired trajectory and the real end-effector position (kinematic behavior) in terms of robustness (with and without end-effector load) using two controllers; dynamic feed-forward controller and computed torque controller experimentally on a 5-DOF robot. The results show the better performance of the feed-forward controller over the computed torque one. However, it does not mention the exact reached accuracy, this is adding some questions especially that our objective is to have a very high accuracy in ranges of micrometers. In (Klančar & Škrjanc, 2007), the author proposed and compared the results of two Model-based controllers on a mobile robot: a model-predictive trajectory-tracking control and a time-varying state feedback controller, then he used a Smith predictor to compensate for the vision-system dead-time. Although the experimental results show that the model predictive controller gives better control results than the state feedback controller, it does not show good results when dealing with large tracking errors and more work is essential to increase the robustness together with obtaining a more accurate model of the robot. Many other researchers proposed various of advanced nonlinear control techniques, such as recurrent fuzzy wavelet neural networks (Yen, Nan & Van Cuong, 2019; Wang, Mai & Mao, 2014)), adaptive control (Brahmi, Laraki, Saad, Rahman, Ochoa-Luna & Brahmi, 2019b; Liu, Zhao & Wen, 2019; Brahmi, Brahmi, Saad, Gauthier & Habibur Rahman, 2019a), sliding mode control (SMC) (Kali, Saad, Benjelloun & Khairallah, 2018; Hacioglu & Yagiz, 2019), model predictive control (Hyatt, Williams & Killpack, 2020), backstepping control (Binh, Tung, Nam & Quang, 2019), $\mathcal{H}\infty$ control (Liu, Tian, Wang & Zhang, 2016) and iterative learning control (Chen, Chu, Freeman & Liu, 2020). Mostly, MB applications depend on well-known information about the mathematical dynamic model of the robot with a small tolerance, but in the case of not having this accurate mathematical dynamic model, in addition to unknown disturbance, it will be useless to use this kind of controller especially in the context of the robot high speed, very tiny error is allowed (in micrometers), and real-time applications.

2.4.2 Non-model-based (NMB) controllers

Apart from model-based controllers which are mainly contingent on accurate knowledge of the system, the Non-model-based (NMB) controller does not require any parameters information of the system; hence no mathematical model of the system is required in the design. In NMB, the design of the controller basis on just reaching and dealing with the plant response, not the model itself because there is no accurate model of the system which matches our case in this research. In (Kumar & Kumar, 2017), the so-called Artificial bee colony algorithm combined with fuzzy PID control was employed to overcome the tracking problem of a 2-DOF robot manipulator. The optimization algorithm estimated the parameters of membership functions (MF) of interval type-2 fuzzy PID controller which provided effective stability and robustness. In addition, by comparing the MF of the interval of type-1 fuzzy PID controller and type-2 fuzzy PID controller, the latter has the best trajectory and robustness for model uncertainties and disturbance rejection.

2.4.2.1 Active Disturbance Rejection Control (ADRC) review

Another interesting non-model-based promising approach namely the Active Disturbance Rejection Control (ADRC) can effectively hit out the target. It was proposed for the first time by Han (Han, 1995, 1999, 2009), and it is applicable to nonlinear, n^{th} order, time-varying, multi-input, multi-output systems (Han, 1995). In addition, several results have shown that both linear and non-linear plants exhibit better and more robust performance than classic control theory and feedback linearization technique, independently of mathematical models of plants using ADRC method (Han, 2009; Gao, 2009; Nowicki, Madoński & Kozłowski, 2015). A good comparison between the ADRC methodology and the well-known results of classic control theory PID was introduced in (Han, 1999), in addition to the conducted study in (Nowicki *et al.*, 2015) that investigates similarities between them on the basis of the disturbance decoupling and the feedback linearization applied on a single link manipulator with flexible joint dynamics. Note that from the theoretical point of view, despite the second-order ESO convergence and stability were only proved and analyzed recently in (Huang & Han, 2000; Gao, 2015; Das, Mehta & Roy, 2020), Table 2.2 illustrates the advantages of ADRC over PID. In practice, however, ADRC has been applied to a broad range of different engineering applications, such as motor motion and speed control (Wu & Huang, 2019; Suhail, Bazaz & Hussain, 2020), flight control, attitude tracking of rigid spacecraft (Luo, Sun, Wu, Sun, Chen & He, 2019; Lotufo, Colangelo, Perez-Montenegro, Canuto & Novara, 2019), robot control (Abdallah & Fareh, 2019; Cheng, Tu, Zhou & Zhou, 2019; Arcos-Legarda, Cortes-Romero & Tovar, 2016; Guanyu, Cheng, Zhenbang & Huibin, 2019; Chen, Sun, Xu & Wang, 2019), and other industrial control systems such as low-velocity compensation of brushless DC servo, control for superconducting RF cavities, the boiler turbine-generator control systems, under-actuated mechanical systems, stabilization of axial flow compressors, and micro-electro-mechanical systems gyroscope (Madonski, Shao, Zhang, Gao, Yang & Li, 2019; Xue, Zhang, Sun & Fang, 2020). As a matter of fact, ADRC is screening almost all control engineering domains.

Numerical simulation and real-time experiments were held to test the proposed ADRC controller including a comparison with the regular PID controller in terms of angular position trajectory and the results proved the better performance of the ADRC over the regular PID controller. (Jiang, Qiu, Wu & He, 2016b) solved the problem of self-balancing control for two-wheeled self-balancing robot by using ADRC whose parameters are adjusted by adaptive differential evolution (ADE) algorithm and the steering control problem by using the tracking differentiator (TD) with PID controller. The simulation results show the effectiveness of the proposed controller with fast adjusting speed of the robot, high precision, and strong robustness.

Table 2.2 Advantages of ADRC over PID

	PID	**ADRC**
Set point	Is often given as a step function (Not appropriate for most of dynamic systems)	Simple differential equation to be used as a transient profile generator
Derivative part	Sensitive to noise	Noise tolerant due to tracking differentiator
Control law	Linear weighted sum	The power of nonlinear control feedback
Integral part	Introduces saturation and reduced stability margin due to phase lag	The integrated control problem is transformed to of total disturbance estimation and rejection

The reality that ADRC holds certain particularity in conception, straightforwardness in implementation, and superb performance in engineering practical applications. To name but a few, it can deal with a wide range of uncertainties, it has improved transient response, and it can be implemented very simply. Wherefore, in the field of robotics, there are many applications of ADRC address the trajectory tracking problem, with both simulation and experimental work: for a tomographic robotic system (Wen *et al.*, 2017), for a Delta robot trajectory tracking problem with uncertain dynamic model (Castañeda, Luviano-Juárez & Chairez, 2014), and for self-balancing control for two-wheeled self-balancing robot (Jiang, Qiu, Wu & He, 2016a). For a 1-DOF manipulator, several versions of ADRC were used (Yang & Jie, 2017), in combination with backstepping technique (Yang *et al.*, 2018), and with an add-on observer-based PD control in (Xue, Madonski, Lakomy, Gao & Huang, 2017).

As can be seen, the range of ADRC applications seems broad, therefore, it can be a possible solution for the low accuracy tracking performances of industrial robots and can overcome the side effect of integral feedback to the closed-loop system. Accordingly, (Vera, Luviano, Santos-Cuevas & Chairez, 2017) used the ADRC controller in the trajectory tracking for a tomographic robotic system, the controller was useful for controlling the class of uncertain nonlinear systems represented. External disturbance, uncertain dynamics, non-modeled effects were treated as a function of time to estimate the robot velocity then canceled it in the feedback loop. In (Gao, Huang & Han, 2001b), it was shown that both linear and non-linear plants show better performance independently of mathematical models of plants using ADRC method. Furthermore, in (Gao, 2006b), the researchers estimated the unknown dynamics and disturbances and actively compensated them in real-time which gave the feedback control the advantages to be more robust and less dependent on the mathematical model of the physical process. Although much experimental and analytical work should be done in the field of ADRC controller, it still shows promising results that can well be applied to our case.

In (Xue *et al.*, 2017), an add-on “module” was proposed which is a special variation on ADRC and which can be combined with observer-based PD controls to increase the robustness of the system. Simulation and experimental validation were done on 1-DOF manipulator and results show that the angular position responses are close to the desired output values with the existence of several mass changes and external disturbances, which justified the effectiveness of the proposed modularized ADRC. ADRC based on Generalized Proportional Integral observer (ADRC with GPI) was developed in (Arcos-Legarda *et al.*, 2016) to control the dynamic walking of a 5-DOF bipedal robot, the proposed controller was divided into two control loops: the external loop that was responsible for generating the walking pattern and the internal control loop which had the task of tracking references generated by the external loop. The tracking references and disturbances rejection improved using ADRC with GPI when compared to the classical controller, however, it does not give a smooth enough trajectory. The researcher in (Desai, Patre & Pawar, 2018), investigated the effect of external disturbances, measurement noise, and parametric variation on a tank level in real-time environment. The slow tracking problem was solved by proposing an adaptive rate limitation which allows the control signal to change rapidly within this limitation. The results were compared to the PI controller and gave a better transient response. However, this application (tank level) had the advantage of limited system variation, but in the case of 6-DOF robots, the large amount of system variation may be a challenge. Although much analytical work remains to be done in the field of ADRC control, the ADRC scheme shows promising results in practice and can be applied to our case.

Motivated by the above references, we focus in this thesis on employing ADRC to solve a very specific problem related to industrial robot accuracy by designing and implementing an external advanced ADRC controller in error-based form, using real-time feedback data from an external distance-based sensor (Renishaw's ballbar). In the literature, different path following control algorithms has been applied to know industrial robots such as ABB or FANUC. This is the first time, however, that such an application is used on the Meca500, a quite novel robot in the market whose the software option has only been recently released. While this robot arm is highly accurate for static positioning, our objective is to keep the same degree of accuracy in micrometers as per industry standards even in dynamic mode, i.e., while the end-effector is following different paths at different speeds. Despite the constraints of black-box modeling, limits on movement and accelerations as well as noise and low transmission frequency of sensors, the proposed controller overcomes all those challenges and all the industrial robot sources of errors illustrated in Figure 2.6 with advantages of high accuracy, small overshoot, noise filtering capabilities, robustness to different payloads and strong capacity of resisting disturbances.

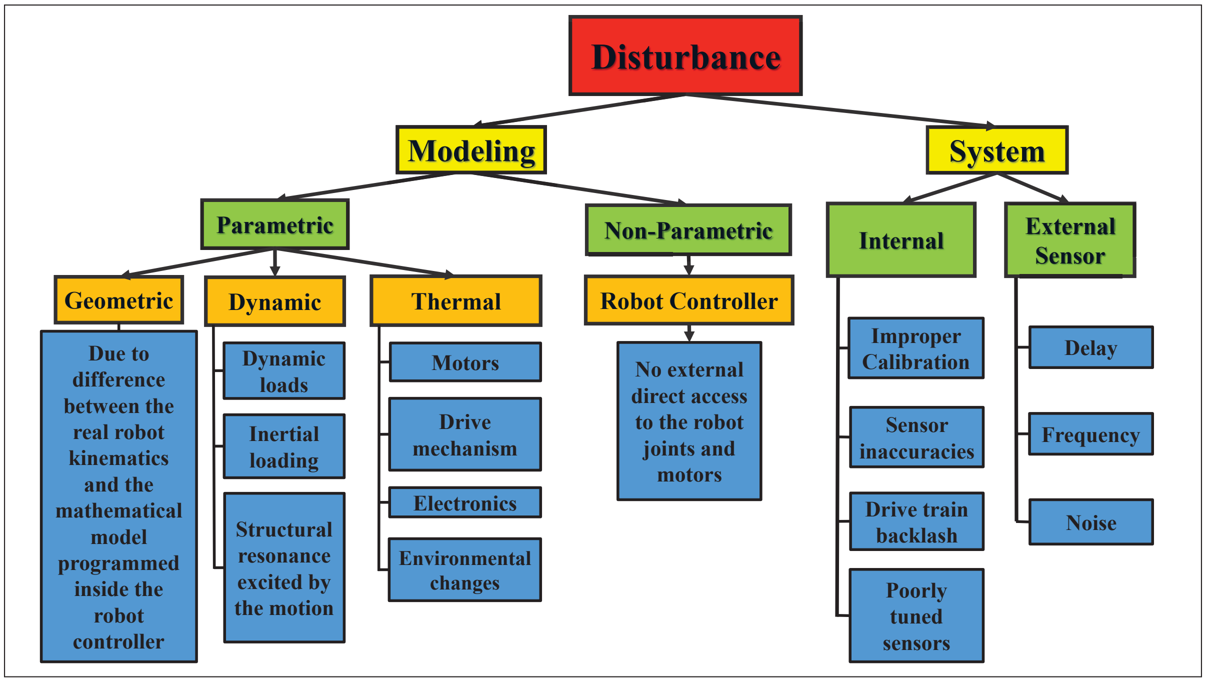


Figure 2.6 The classification of industrial robot disturbance

2.4.2.2 Fuzzy Logic Control (FLC) review

Fuzzy logic is one of artificial intelligence (AI) techniques that centers around algorithms used to replicate human philosophy taking different decisions and actions in machines. Where the system process data cannot be accessed, fuzzy algorithms are used, because they have the ability to analyze the system uncertainties mathematically, i.e, the information in a gray environment. Fuzzy logic provides a practical, inexpensive solution for controlling complex or ill-defined systems. Despite its contradictory-sounding name, fuzzy logic offers a rigorous framework for solving many types of control problems. Rule-based fuzzy controllers require less code and memory and don't need heavy number-crunching or complex mathematical models to operate. All that is needed is a practical understanding of the overall system behavior improvement in reducing noise effects on the tracking error when used in Laser tracking systems (Ying Bai, Hanqi Zhuang & Roth, 2005). Around four decades of FLC have disclosed that systems based on FLC have been employed for various applications in different areas since Professor L.A.

Zadeh of the American University of California developed fuzzy mathematics for the first time in 1965 (Zadeh, 1973). Generally speaking, FLC provides a practical, economical solution for controlling complicated, tricky, or obscure systems that are hard can be described by a mathematical method. Thus, fuzzy control techniques have been a powerful tool to deal with uncertain nonlinear systems (Wen *et al.*, 2017). It has a great attraction by many researchers and scholars to enhance and develop the related methodologies, which is growing in different social and natural science fields. FLC systems can communicate, extract the input-output data linguistic information, then describe the system dynamics in the local region by the fuzzy rules designed by the user. Therefore, fuzzy models have been applied in many and different applications (Chien, Chen, Tsai & Chen, 2010; Huang, Li & Chen, 2009; Wang, Tanaka & Griffin, 1996), it is distinct from black-box traditional modeling techniques.

Through distinct fuzzy models, the modeling technique of Takagi-Sugeno (T–S) is the most attractive because of its fundamental modeling and desirable system parametric properties (Ying, 1999; Tanaka & Sugeno, 1992). It is introduced as a method of fuzzy model identification method by Japan's Takagi and Sugeno in 1985 (Liang, Chen & Xu, 2013). It is modeling the nonlinear systems using the linear dynamic equations of local sub-models. Thus, it is easy to design the controller and analyze the system by using modern control systems. Authors in (Cao & Frank, 2000; Dong, Wang, Ho & Gao, 2010; Hua & Ding, 2011) developed control algorithms on the basis of T-S fuzzy model for modeling nonlinear systems. In addition, researchers in (Wang & Fei, 2014) illustrate T-S technique for nonlinear time-delay systems. In (Su, Shi, Wu & Song, 2012), a novel method is proposed to filter the design of T-S fuzzy in the case of discrete-time systems with time delay. In (Xu, Cui, Li, Yao, Tian & Zhou, 2020), built a digital IR model on the basis of digital manufacturing features model, he uses the FLC incorporation with Best Worst Method (BWM) to drive the weights of the proposed model in the simulation. However, in implementation, the model does not reflect the real state of the physical IR system, due to inaccurate delayed conditions.

To conclude, FLC is deployed to model the unknown nonlinear system input and output data, in reliance on human experience, which is considered a practical solvent for our mouse-robot interaction control problem. This mutual interrelationship can be approximated by T-S fuzzy model based on the aforementioned discussion. Once, the system's practical performance and the required controller behavior is well recognized by the robot user and the controller designer.

2.4.2.3 Admittance Control review

The interaction control has been studied in many papers in the last decades, and it earned a wider margin of importance with the progress and developments in power electronics, computer power, and sensors (Siciliano & Khatib, 2016; Lahr *et al.*, 2016). The main challenge of the interaction control that it has to give the robots better information about the environment that cannot be gained using traditional sensors. Likewise, the limitation of some applications that can only be implemented through force and torque control, take, for instance, assembly in precision applications (Chen & Liu, 2013), rehabilitation high assistance mechanisms (Krebs *et al.*, 2006), or surgery assistance (Luo, 2016). In manipulation applications, proficiency is needed in the robotic hand fingers or a gripper, in an approach that not allowing the grasped objects to slide. Simultaneously, they must be careful of the amount of the imposed contact forces, if it exceeds a certain threshold, it will cause some damage to the object. Further, while industrial applications are spreading and being enhanced day after day, it needs to interact with the surrounding environment in a wider range of everyday tasks, such as painting, polishing, drilling, machining, and many other activities, i.e., the tasks related to direct contact with the environment) (Pires & Bogue, 2009). The main challenge for this interaction controller the difficulties facing it due to the mutual interaction between the sensor used by the operator and the robotics system, i.e., the nonlinear behavior and performance, exactly like our case study. For that reason, the classical control algorithms are not appropriate for these tasks (Chen & Liu, 2013; Hogan, 1984). One of the most famous techniques dealing with this control challenge is the impedance and admittance controls, since its announcement by Hogan (Hogan, 1984), it gained high interest and becomes very fast one of the most successful in its field.

Despite the fact that many modern applications are designed based on this technique, it is not an easy task to be accomplished practically (Grafakos *et al.*, 2016); thus, it is worthwhile to be employed as an implementation corresponding algorithm. The author in (Grafakos *et al.*, 2016) gave a clear explanation on implementations to discrete systems, such as micro-controllers, PCs, and industrial robots, he proposed two different implementation methods for the same controller. The methods are successfully modeled, numerically simulated, and validated. Commonly, in haptic applications, impedance and admittance control are the two main control classes that are used. Impedance controllers can be used when the system accepts a displacement as input — which is measured — and react with an effort (force) as an output. Ideally, the systems controlled by this method should have some specific proprieties, such as low inertia and friction, because if these forces are not properly compensated, the user will consequently feel these forces. On the other hand, the admittance control is employed when the system is capable of receiving the force as input and imposes a displacement as an output (Grafakos *et al.*, 2016). This concept defines the implementation that is desired for each system. The mouse environment in our case study accepts the forces on 6-axis and gives deflections in position, which in turn can be transformed into velocities to be sent to the robot of interest in velocity mode, therefore this is defining an admittance control.

Notwithstanding the fact that the admittance control turned out to be a famous method for designing force controllers in human-robot interaction, in which takes an input force, and gives the desired motion as an output, e,g., position, velocity, or acceleration, to the robot's manufacturer controller, the conventional one has fixed parameters which cannot be modified on the basis of different states of mutual motion. (Grafakos *et al.*, 2016) presented a novel approach of variable admittance control to enhance the system insightfully, to eliminate the trade-offs legacy of the fixed admittance control. The results were promising, however, the application was on a large 4-DOF intelligent assist device has large inertia and significant friction as shown in Figure 5.14. For our application case study, while the robot is very small and fast, it even worst in performance if it is implemented. To be specified, the related mutual interaction between the very tiny motion of the six-axis joystick SpaceMouse and the very fast response of the small robot has different oscillations and errors, and a proper way of driving the mouse should be investigated. For instance, the author in (Underwood & Gallimore, 2010) proves that using one hand to control the robot compared to having translation controlled by one hand and rotation controlled by the other is performed with fewer errors in both translation and rotation, and despite that, the error was increasing and the motion was uncontrollable when moving in the robot's high speeds. This error was related to the cross-coupling between translation and rotation when all six DOF were controlled simultaneously. At this point, it is worthwhile to mention that researcher in (Underwood & Gallimore, 2010) mentioned that the device settings in some applications may considerably shorten the training time for the robot operator to manipulate the robot, and he also illustrates that different people may acquire different devise setting levels, which is needed to be changed with each user system experiences. However, our objective is to conduct our proposed control methodologies to be auto-adjusted regardless of user experiences. In (Martins, Cunha & Morgado, 2012), the author proposed the measures of system usability which need settling and learning time to achieve the performance speed. However, our objective is to design the controller to auto adapt with various motion conditions so that the robot can operate in a smooth manner without any oscillation or noise, regardless of the user system's experiences.

CHAPTER 3

SYSTEM CHARACTERIZATION AND ANALYSIS

In this chapter, we first introduce the industrial robot of interest and its motion control command options including the available types of communication with the robot. Secondly, the technical specifications of the real-time feedback sensor used for the proposed controller feedback are introduced. Next, we present the experimental setup, then we investigate the best online trajectory generator by proposing two methods to make the robot follow a circular path in velocity mode, in addition, the sensor buffer reading methods are investigated to determine the right sampling time that can be selected in the experiments in real-time. The first method is done by receiving the robot's pose feedback option while the second one is accomplished without using this option. The results show that using the second method of designing the robot's circular motion gives an appropriate solution for satisfying a stable and fixed sampling time. Finally, the six-axis SpaceMouse joystick used for the human-machine interface is described, particularly its communication mode and the inherent difficulties in using the SpaceMouse for hand-guiding.

3.1 Mecademic's Meca500 industrial robot

The industrial robot used in this research is Mecademic's Meca500—a particularly small, compact, and precise six-axis robot arm as shown in Figure 3.1. This robot is situated in CoRo laboratory of École de technologie supérieure (ÉTS). The robot's controller is embedded in the robot's base, and the arm weighs only 4.5 kg. The rated payload of the robot is 0.5 kg while the maximum payload is 1 kg under certain conditions (Mecademic, 2020), and its position repeatability is 0.005 mm. Unlike other industrial robots, the Meca500 can only be remotely commanded by feeding it with high-level instructions from an external computing device, over either EtherCAT or TCP/IP. Figure 3.2 shows the workspace and dimensions of the robot, while Table 3.1 illustrates the main technical specification (Mecademic, 2020).

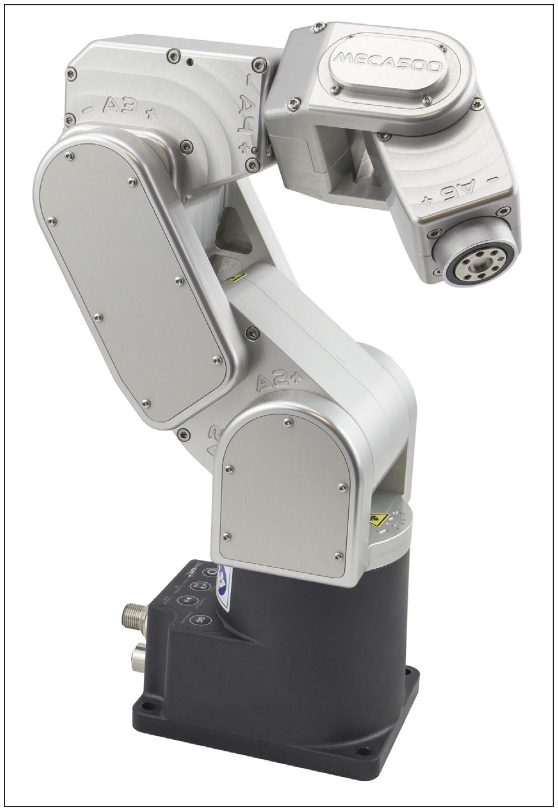


Figure 3.1 MECA500 (R3)
industrial robot from Mecademic
Taken from Mecademic (2020)

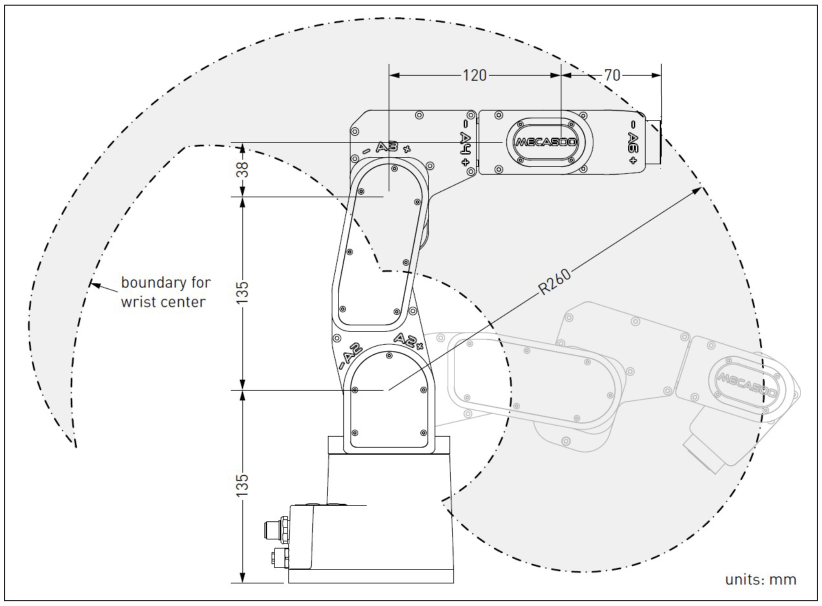


Figure 3.2 The dimensions of Meca500
Taken from Mecademic (2020)

Table 3.1 Meca500 technical specifications
Taken from Mecademic (2020)

Parameter	**Value**
Position Repeatability	0.005 mm
Rated payload	0.5 kg
Max. payload	1 kg under special condition
Weight of robot arm	4.5 kg
Range for joint 1	[-175°, 175°]
Range for joint 2	[-70°, 90°]
Range for joint 3	[-135°, 70°]
Range for joint 4	[-170°, 170°]
Range for joint 5	[-115°, 115°]
Range for joint 6	[-36,000°, 36,000°]
Max. speed for joint 1	150°/s
Max. speed for joint 2	150°/s
Max. speed for joint 3	180°/s
Max. speed for joint 4	300°/s
Max. speed for joint 5	300°/s
Max. speed for joint 6	500°/s
Max. TCP linear velocity in joint mode	More than 2,000 mm/s
Max. TCP linear velocity in Cartesian mode	500 mm/s

3.1.1 Robot movement control commands

Generally, it is important to define how the users want the robot's end-effector to move to its target, either by specifying the desired position and orientation (pose) of end-effector or by rotating the robot joints to a desired joint set. The Meca500 has two modes that can be used individually to program the robot to move around: position mode and velocity mode (Mecademic, 2020), as described below:

- In the traditional control method denoted as "position mode", a user commands the robot to move by either specifying the desired end-effector pose or the desired joint values. On one hand, when the robot end-effector must follow a linear path in Cartesian space (i.e., pose values), the robot Cartesian space motion commands MoveLin, MoveLinRelTRF, and MoveLinRelWRF should be used. In this case, the required linear and angular velocities can also be specified using the robot's commands SetCartLinVel and SetCartAngVel, respectively, besides specifying the linear and angular acceleration using SetCartAcc. However, the desired velocities are not guaranteed, because if the robot is close to singularity even with a tiny velocity, the joints may rotate at maximum velocity. On the other hand, when the robot end-effector must follow a linear path in joint space (i.e., rotate to a certain joint set), then the robot joint space motion commands MoveJoints or MovePose should be used. Similarly, the velocity and acceleration of joints can be specified using robot commands SetJointVel and SetJointAcc, respectively. Finally, it is very important to know that in position mode, once the robot starts executing a motion command, the remainder of the trajectory cannot be modified (the robot can only be stopped) as mentioned in (Mecademic, 2020).

- In the alternative control method known as “velocity mode”, it is the desired Cartesian velocity of the robot’s end-effector or the desired velocities of the joints that can be continuously fed to the robot (as frequently as every 2 ms). As soon as the robot receives a new velocity-mode motion command, the robot executes it without stopping. Thus, the velocity mode can be used for real-time trajectory following in conjunction with an external sensor. Therefore, it can be used if some advanced applications are to be used such as dynamic path corrections, force control, or telemanipulation. This control mode is associated with the velocity mode motion commands MoveLinVelTRF, MoveLinVelWRF, and MoveJointsVel. The effect of the velocity mode motion will continue until the end of the time duration specified by the command SetVelTimeout, which ranges from 0.001 s to 1 s. Besides, if the robot runs into a singularity, the motion will be automatically stopped. The joints acceleration for all velocity-mode commands can be assigned by the command SetJointAcc, while the acceleration of the end-effector can be specified by the command SetCartAccs.

To be concluded, if we are trying to control the robot in the position mode in real-time, the robot has to reach a specific position then stops before being able to move to a new position, which is causing an accumulation of errors and delay while updating the position in real-time while using our proposed controller. Therefore, we choose to operate the robot Meca500 in velocity mode, since in this mode, the trajectory can be updated on the fly during the robot motion without any movement interruption, stop, discretization, or changes in velocity and/or acceleration values since we are far from the robot's singularity. The influence of the robot motion in position and velocity modes settings is illustrated in Figure 3.3.

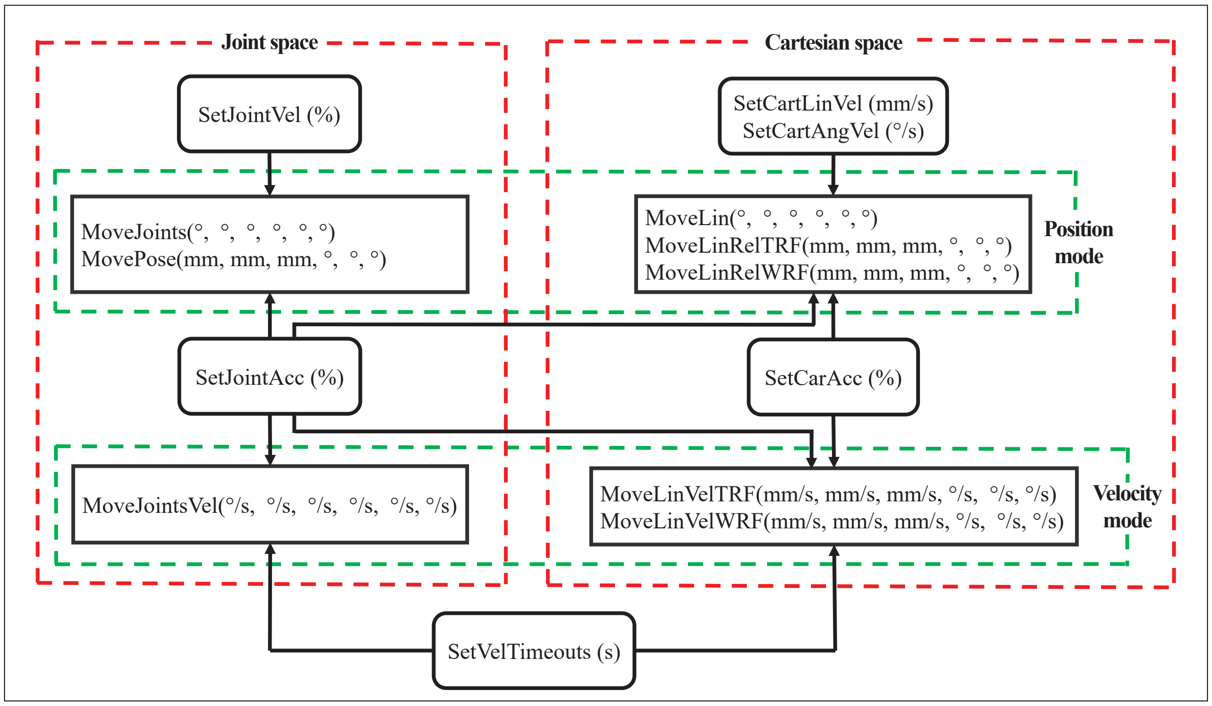

Joint space
Cartesian space
SetJointVel (%)
SetCartLinVel (mm/s)
SetCartAngVel (°/s)
MoveJoints(°, °, °, °, °, °)
MovePose(mm, mm, mm, °, °, °)
MoveLin(°, °, °, °, °, °)
MoveLinRelTRF(mm, mm, mm, °, °, °)
MoveLinRelWRF(mm, mm, mm, °, °, °)
Position mode
SetJointAcc (%)
SetCarAcc (%)
MoveJointsVel(°/s, °/s, °/s, °/s, °/s, °/s)
MoveLinVelTRF(mm/s, mm/s, mm/s, °/s, °/s, °/s)
MoveLinVelWRF(mm/s, mm/s, mm/s, °/s, °/s, °/s)
Velocity mode
SetVelTimeouts (s)

Figure 3.3 Settings that influence the robot motion in position and velocity modes
Taken from Mecademic (2020)

3.1.2 Communication with Meca500

The Meca500 can only be remotely commanded by feeding it with high-level instructions from an external computing device, over either EtherCAT or TCP/IP via Ethernet through a computer or a PLC (Mecademic, 2020). The default Ethernet via TCP/IP protocol method is chosen to communicate to the Meca500 robot using a PC. In this case, the robot Meca500 is using null-terminated ASCII strings that are being transmitted to PC through a default IP address (192.168.0.100). There are two available ports to communicate with the robot:

1. A default TCP/IP port (10000), it is known as the control port, all the movement and robot setting commands can be sent via this port.

2. A feedback TCP/IP port (10001), the robot will continuously send pose feedback over this port after homing and movement at the rate specified by the (SetMonitoringInterval) command in seconds, ranging from 0.001 s to 1 s.

At this end, it is worthwhile to mention that we are using two methods to design the robot's circular motion, the first method is designed by using both robot's communication ports in real-time (the default port to send the desired movement to the robot and the other feedback port to receive the current robot's TCP pose) while the other method is designed by using only the default control port in real-time. However, the results of using these two ports are analyzed later to determine if using the feedback port in real-time is relevant while designing the robot's circular motion, or it is better to use only the default control port to send the required movement to the robot without the needs to receive the current pose by the robot feedback port.

3.2 The External Wireless Ballbar Sensor QC20-W

Given the high precision of the Meca500, Renishaw's QC20-W telescoping ballbar was selected as a real-time external sensor for the purposes of this study only. The QC20-W is a linear transducer with a measurement range of ±1 mm, and an accuracy of ±0.001 mm, embodied in a telescoping bar of 100 mm nominal length, having a precision 0.5-inch steel ball at each end. The QC20-W is a standard tool for evaluating the positioning performance of CNC machine tools (Esmaeili & Mayer, 2020; Ding, Wu, Huang, Song & Zhang, 2019) for analyzing the accuracy of the CNC machine tools (Renishaw, 2016), by following circular trajectories. It is often used in robotics as well, for both evaluation (Slamani, Joubair & Bonev, 2015) and calibration purposes (Nubiola & Bonev, 2014; Yang, Guo & Kong, 2020). It is often used in robotics too (Čep, Malotová, Kratochvíl, Stančeková, Czán & Jakab, 2018a) and it has the capability to measure radius variations while rotation around a fixed point, it provides very accurate measures of any variations in the test circle radius traced by the machine during the test. Contouring performance such as circular deviation and circularity can be calculated. The sensor data can be transmitted to PC using Bluetooth Class 2 module. The sensor is powered by battery, in addition to a LED indicator built into the housing which shows battery, communications, and fault status as shown in Figure 3.4. The ballbar kit contains a 100 mm long ballbar assembly and 50, 150, and 300 mm long extension bars, which gives the ballbar an option to be used to measure different distances (Renishaw, 2016). Besides, Table 3.2 illustrates its main technical specifications. In this study, the original length of the ballbar (100 mm) will be used without any extension bars, this is because the robot is tiny and its workspace dimension is very limited as shown in Figure 3.2.

Of course, in an actual industrial application where accurate robot guiding is required, a non-contact sensor will be used, most probably a distance one. Thus, we used the QC20-W to emulate such a sensor mainly because it is very accurate and precise. However, our work is directly applicable if another high-accuracy sensor is used, as long as it supports similar or better transmission rates.

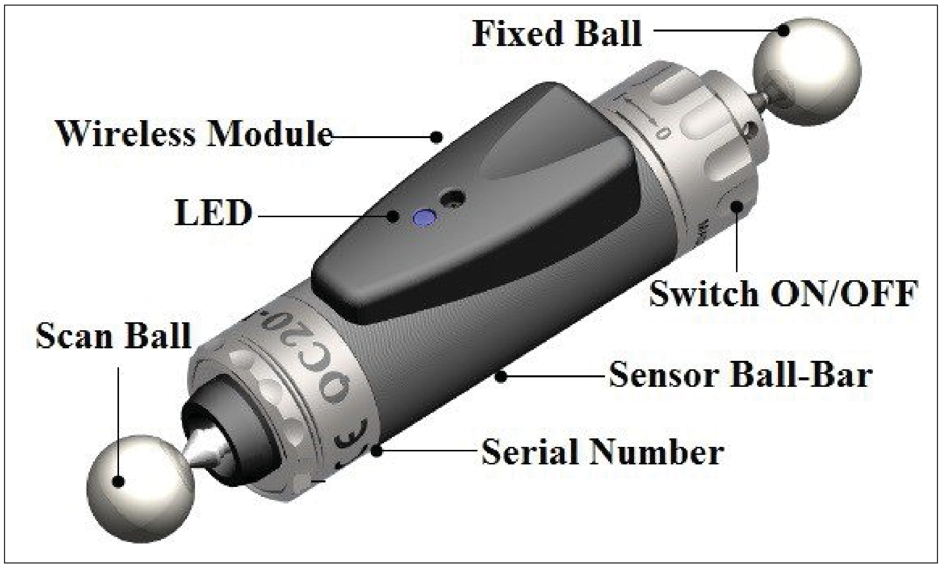


Figure 3.4 The QC20-W ballbar sensor
Taken from Čep *et al.* (2018b)

Table 3.2 QC20-W ballbar technical specifications
Taken from Renishaw (2016)

Parameter	**Value**
Sensor resolution	0.1 μm
Ballbar measurement accuracy	± 0.7 μm
Ballbar measurement range	± 1.0 mm
Sensor stroke	-1.25 mm to +1.75 mm
Maximum sample rate	1000 Hz
Data transmission Bluetooth, Class 2	10 m typical

3.2.1 Ballbar Calibration kit

The ballbar is supplied with a calibrator kit shown in Figure 3.5 to calibrate the length of a ballbar. The manufactured material of the calibrator has a temperature expansion coefficient of almost zero. When it is used with the calibrator, the ballbar calculates absolute (rather than relative) errors for axis scaling and radial deviation values. We always calibrate the ballbar sensor each time before starting any experiments by following the calibration procedures stated in (Renishaw, 2016) and similar to what can be seen in the calibration part of this video:.https://www.youtube.com/watch?v=3lYp1TFhJTw, except that the calibration value is manually added to our generated program in MATLAB before each experiment.

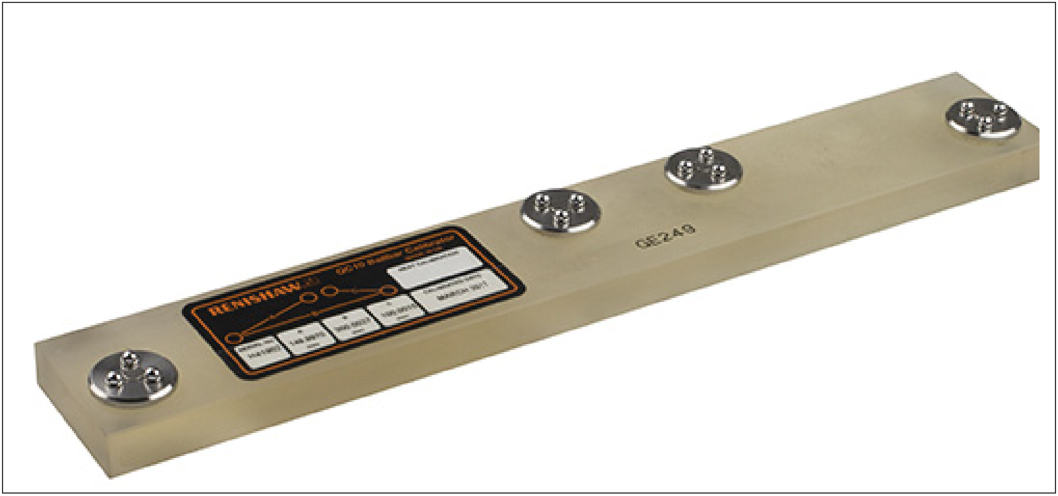


Figure 3.5 The QC20-W BallBar Calibrator
Taken from Renishaw (2016)

3.2.2 Ballbar Sampling Time

The QC20-W sampling rate is 1000 Hz (Renishaw, 2016), meaning that measurement is recorded by the ballbar buffer every 1 ms. The latter can be defined as the buffer recording sampling time. However, due to the Class-2 Bluetooth protocol used by the ballbar for transmitting measurements, the rate of feeding results to the PC is about 28.5 Hz. Thus, the buffer reading time is about 35 ms, but these measurements are repeated during the reading process and they are not updated to new values except after approximately 100 ms (i.e., sampling rate of 10 Hz), which leads to a new definition of buffer updated reading time. In real-time applications, the buffer updated reading time is more important than buffer recording and reading times, because it is considered as the real sampling time in which a new updated value is received from the ballbar buffer, then it is used as feedback for the proposed controller on the fly in real-time. Table 3.3 illustrates all the related sampling times.

In order to get all the information about the aforementioned ballbar buffer times, we introduce with details the two methods that we tested experimentally to be able to read from the ballbar buffer:

1. The first method is to read the last reading value from the ballbar recording buffer, in which the user can read the updated buffer reading value every 100 ms, despite the user can read the stored values in the ballbar buffer every 30–40 ms (Buffer reading time), it is useless because they are repeated. In this sense, we intentionally wait until a time of 100 ms is elapsed to be able to get an updated considerable value. Hence, the minimum sampling time for the ballbar that can be used in real-time is 100 ms. On other words, we are limited by the ballbar updated sampling time of 100 ms (i.e., the sampling rate of 10 Hz), because experimentally this is the time that the ballbar data is being updated in real-time. If the sampling time is reduced, the ballbar reading data will be repeated and will not reflect the real reading data for each segment of motion, which is extremely important for the controller feedback especially with this high precision required accuracy, every micrometer of deviation counts.

2. The second method is to read all the readings accumulated in the recording ballbar buffer at the end of the robot movement (i.e., after the circular motion is completed), which include all the recorded values at the buffer recorded sampling time of 1 ms, (i.e., sampling rate of 1000 Hz). Nevertheless, this method cannot be used in real-time applications.

In this chapter, the aforementioned two methods of reading the ballbar buffer are being used to evaluate and identify the system. One is used in the real-time application and the other is used to determine the reference trajectory for the proposed controller (will be discussed later).

Note: The previous two methods are not stated in the ballbar manual (Renishaw, 2016). We recorded the received ballbar values while experimentally using the ballbar sensor. Then the results were analyzed, as illustrated in subsection 3.4.3. We are using a ballbar API in order to communicate with the ballbar sensor via MATLAB environment then to receive the data from the ballbar buffer in real-time.

3.3 Online Circular Trajectory Generator and Experimental Setup

Generally, the concept of the Online Trajectory Generation (OTG) is the ability to recalculate the robot trajectory at any time instant, because of dynamic changes to the target pose (Liu, 2002). In our case, such target pose changes are due to the feedback provided by the ballbar, because the location of the target is only available after the beginning of the movement, which requires an on-line update of the reference trajectory (Zhao, 2015). In addition, robot movement is always under some constraints specified by the robot companies such as the maximum Cartesian/joint velocity and acceleration values. The regenerated trajectory must be updated “on-the-fly” without any movement interruptions, discretization, or changes in velocity and acceleration values. This project falls into the concept of OTG because the ballbar external sensor is used in real-time, and the regenerated trajectory will be updated “on-the-fly” by this sensor. Consequently, the buffer updated reading time mentioned in subsection 3.2.2 is considered as the sampling time (100 ms) for the whole upcoming circular trajectory experiments.

The ballbar is designed to measure the circular motion variation; therefore, the robot movement has to be in a circle trajectory, which can be designed for this Meca500 robot either by using position mode or velocity mode. However, the velocity mode is preferable as been mentioned in subsection 3.1.1, especially in the advanced applications which require dynamic path corrections or tele-manipulation (Mecademic, 2020). Therefore in this project, the desired trajectory is presented in the velocity mode. At this end, it is worthwhile to mention that following a circular path in position mode is trivial and can be done by discretizing the circle in small linear segments of pre-calculated constant length. Following the same path in velocity mode is more complicated for a user who has no knowledge of how exactly does the robot responds to requests for changes in the desired robot velocity. At this end, in order to figure out a suitable method before going on and design the external controller to design the robot's circular path in velocity mode, we are proposing two methods: (1) the first method is by using the two robot's communication ports mentioned in subsection 3.1.2 to send the robot's movement commands then receiving the robot's TCP pose feedback; (2) the second method is by just sending the robot's movement commands without receiving the robot's TCP pose feedback. Both these designing methods are illustrated, evaluated then discussed in order to choose the best method relevant to our controller.

3.3.1 Robot Setup

The experiment design and setup have been chosen based on a range of factors, such as robot workspace, ballbar radius limitations, and buffer updated sampling time. This design is very critical for results validation; thus, a comparative study is proposed here to analyze and validate the experimental results. The base of the Meca500 is attached to the horizontal surface of a rigid table. A custom fixture weighing 0.5 kg is attached to the robot's flange. The fixture holds a 0.5-inch magnetic nest. A rigid holder with a 0.5-inch magnetic nest is mounted to the table. The center of the precision ball that is attached to the nest on the robot's end-effector coincides with the robot's TCP (tool center point), whereas the center of the other precision ball defines the center of the desired circular path (fixed with respect to the robot's base) as shown in Figure 3.6. The setup and the circular motion in our work are similar to what can be seen in this video:.https://youtu.be/P0bnKaWPrcA. In addition, another custom plastic fixture weighing only 15 g is attached on the top of the main fixture in some experiments in order to add an extra 0.5 kg load during the whole circular motion, as shown in Figure 3.7a, or during a part of it, by dropping the load as shown in Figure 3.7b.

3.3.2 Circular path Setup

The desired trajectory is a circle with a radius r of 100 mm that matches the adopted ballbar measured distance in which the robot's TCP moves in yz-plane. The fixed circle central point is denoted as $(\mathbf{y}_o, \mathbf{z}_o)$, while the desired starting point of the robot's TCP trajectory is denoted as $(\mathbf{y}_{d_0},\ \mathbf{z}_{d_0})$, their values are (0, 172 mm) and (0, 272 mm) related to the robot base, respectively. The mentioned coordinates of the circle center and the starting trajectory points have been chosen according to Meca500 robot workspace shown in Figure 3.2 and the adopted ballbar length of 100 mm, this position is chosen to avoid being close to robot singularity configurations during the circular motion.

3.3.3 Robot Commands

To move Meca500 robot in velocity mode, the command MoveLinVelWRF is used to make the robot's end-effector moves with specified Cartesian velocity with respect to the robot's base, which is specified by six arguments $(\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z)$ as follows:

- The first three arguments are the components of the instantaneous linear velocity of end-effector in mm/s, ranging from -1000 to 1000 mm/s. In this application, $\dot{x}$ is fixed to 0 mm/s, while the values of $\dot{y}$ and $\dot{z}$ change along the circle trajectory based on the required robot's linear velocity.
- The second three arguments are the components of the instantaneous angular velocity of end-effector in °/s, ranging from -300 to 300 °/s. In this application, all arguments are fixed to 0 °/s.

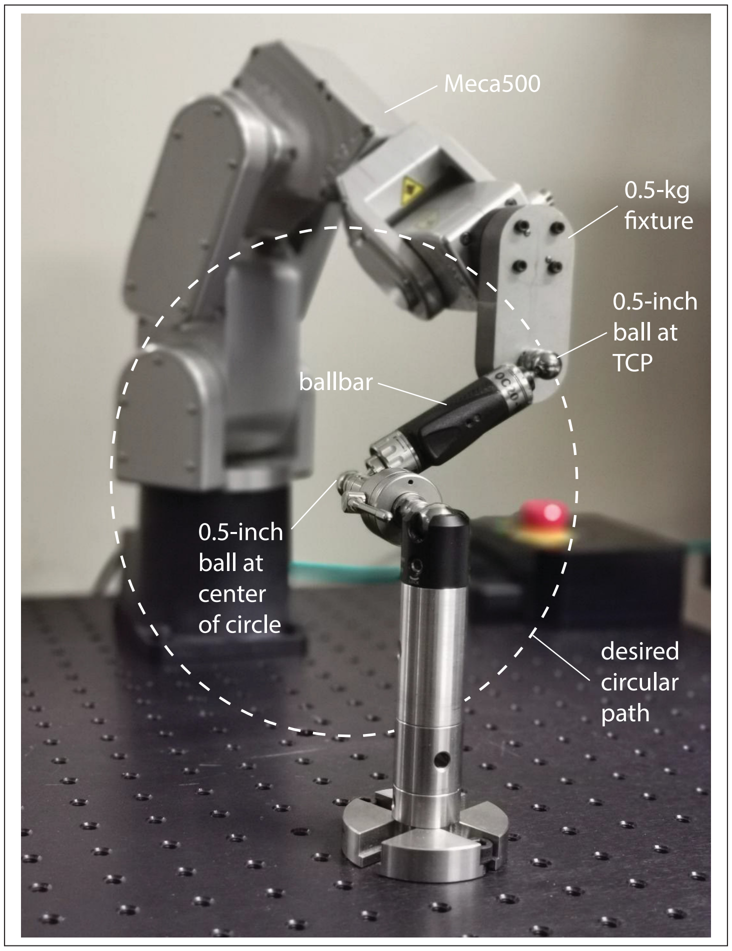
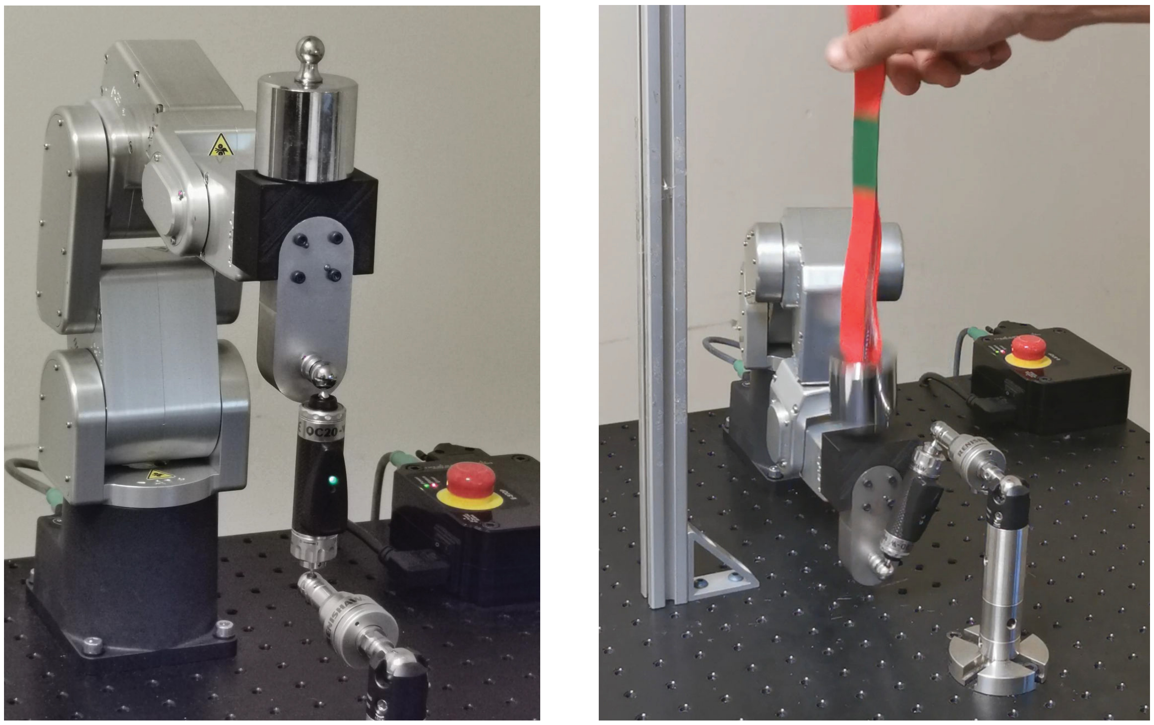


Figure 3.6 Experimental setup of the standard robot's circular motion featuring the Meca500 six-axis industrial robot and the QC20-W telescoping ballbar

In this work, because the circular path is designed in yz-plane, the x-component of the TCP velocity remains zero and the angular velocities of the end-effector will be kept at zero too. Hence, the six arguments specified for the MoveLinVelWRF command will be $(0, \dot{y}, \dot{z}, 0, 0, 0)$. In addition, the timeout of executing the MoveLinVelWRF command is being set to its maximum value of 1 s, this means that the robot will stop after one second unless it receives another command (Mecademic, 2020). This timeout must be equal to or greater than the experiments' sampling time of 100 ms. Table 3.3 illustrates all the robot, ballbar, and circle setup parameters. In the first method of designing the robot's circular path in velocity mode, only one Cartesian



a) Robot's full payload of 1 kg b) Dropping an extra load of 0.5 kg

Figure 3.7 Experimental setup featuring the Meca500 six-axis industrial robot and the QC20-W telescoping ballbar

linear velocity of 10 mm/s has been chosen to be used in analysis and evaluation, in the second method; however, the same circular path experiment is repeated four times using different robot Cartesian linear velocities (25, 50, 75 and 100 mm/s), while joint and Cartesian accelerations and blending option, are fixed to 100 %. The ballbar sensor is attached during all experiments, while the measured errors are stored and plotted during each experiment.

3.4 The First Method of Designing the Robot's Circular Trajectory

It is based on receiving the current robot's end-effector point coordinates (y_{e_n}, z_{e_n}) of each segment of motion (SOM) from the robot position feedback using the feedback TCP/IP port (10001) during the whole circular motion. At the current SOM index (n), the radial position vectors are computed from coordinates positions (y_{e_n}, z_{e_n}) to (y_o, z_o). For instance, at the first

Table 3.3 Experimental Parameters Setup

Parameters			**Values**
Robot	Velocity mode	Timeout (s)	1
		Cartesian linear velocity (mm/s)	25, 50, 75, 100
		Joint acceleration (%)	100
		Cartesian acceleration (%)	100
		Blending (%)	100
	TCP/IP	Control port (10000)	On
		Feedback port (10001)	On
		Monitoring interval (s)	1
		Buffer reading time (ms)	10
Ballbar	Distance measured (mm)		100
	Buffer recording time (ms)		1
	Buffer reading time (ms)		30-40
	Buffer updated time (ms)		100
Circle	Radius (mm)		100
	Circle center (mm)	$\mathbf{y}_o$	0
		$\mathbf{z}_o$	172
	End-effector starting point (mm)	$\mathbf{y}_{d_0}$	0
		$\mathbf{z}_{d_0}$	272

SOM of index n=1, the first radial position vector $\mathbf{v}_{r_0}$ is computed from (y_{e_0}, z_{e_0}) to (y_o, z_o), then being rotated by 90° anticlockwise to get the first tangential position vector $\mathbf{v}_{t_0}$. The same procedures are applied for other vectors such as $\mathbf{v}_{r_1}$ and $\mathbf{v}_{t_1}$, as shown in Figure 3.8. Consequently, and by following the same procedures, all the velocity vectors are computed and sent to the robot each SOM via robot's velocity command (MoveLineVelWRF) through the control TCP/IP port (10000). Note that without getting the robot end-effector positions feedback (y_{e_n}, z_{e_n}) each SOM, via the feedback TCP/IP port (10001), nothing can be computed using this method.

Hence, the final velocity vectors can be obtained by the following steps:

1. Computing the robot's TCP desired radial position vectors $\mathbf{v}_{r_n}$ between the circle origin (y_o, z_o) and the points on the circle circumference (y_{e_n}, z_{e_n}) each SOM then computing its

related unit vector $\hat{\mathbf{v}_{r_n}}$ as:

$$\mathbf{v}_{r_n} = \begin{bmatrix} y_{e_n} - y_o \\ z_{e_n} - z_o \end{bmatrix}, \tag{3.1}$$

$$\hat{\mathbf{v}_{r_n}} = \mathbf{v}_{r_n} / \|\mathbf{v}_{r_n}\|_2, \tag{3.2}$$

where $\|\cdot\|_2$ denotes the ℓ_2-norm and n is the current SOM index.

2. Computing the desired tangential position unit vector $\hat{\mathbf{v}_{t_n}}$, which concerns with the motion direction, by rotating the radial position vector $\hat{\mathbf{v}_{r_n}}$, 90° anticlockwise as:

$$\hat{\mathbf{v}_{t_n}} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \hat{\mathbf{v}_{r_n}}, \tag{3.3}$$

3. Determining the desired robot's TCP velocity υ_r, then computing the required velocity vector $\mathbf{v}_{v_n}$ each SOM by:

$$\mathbf{v}_{v_n} = \upsilon_r \hat{\mathbf{v}_{t_n}}, \tag{3.4}$$

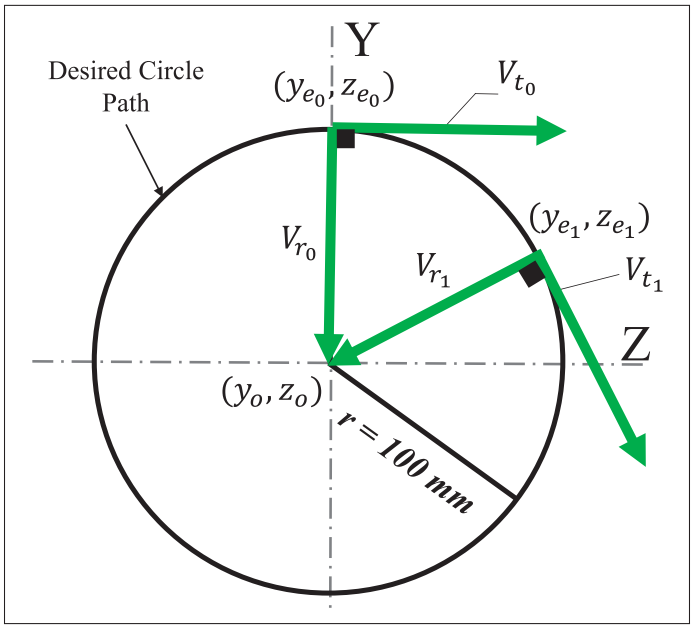
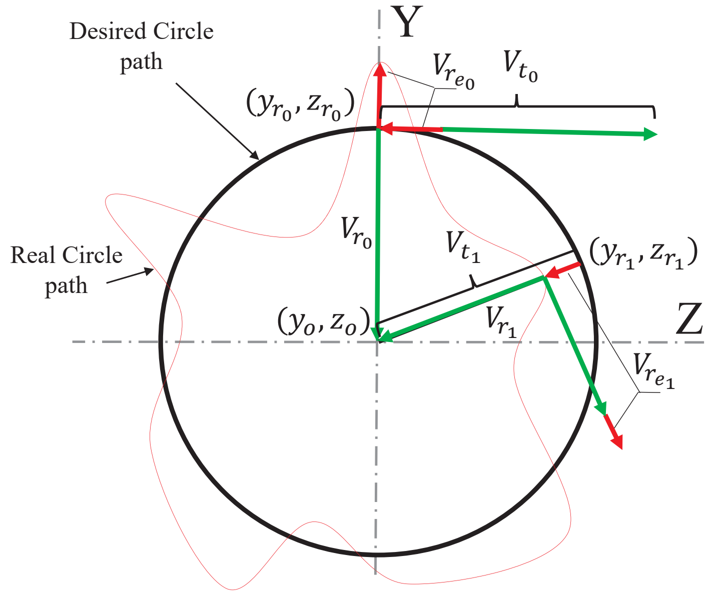
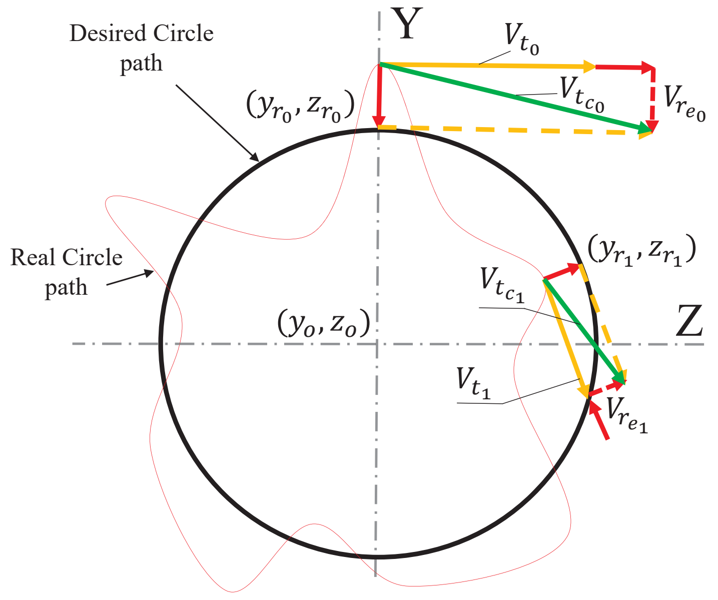


Figure 3.8 Rotation of radial position vectors



a) Calculating the robot's TCP real radial vector $\mathbf{v}_{r_n}$ including the error vector $\mathbf{v}_{r_{e_n}}$



b) Calculating the robot's TCP corrected tangential vector $\mathbf{v}_{\hat{t_{c_n}}}$

Figure 3.9 The first method of designing the robot's circular trajectory

3.4.1 Error Computation

Unfortunately, the accuracy of the robot pose feedback, which is used to calculate the desired radial position vector $\mathbf{v}_{r_n}$ is not perfect, which in turn, affects related desired tangential position vector $\mathbf{v}_{t_n}$ accuracy used to move along the circular trajectory. Figure 3.9a shows the real radial and tangential position vectors which includes the desired position vectors in green color plus the real error vectors $\mathbf{v}_{r_{e_n}}$ in red color. In order to be more accurate, a ballbar sensor is used besides the robot position feedback to determine the real error vector $\mathbf{v}_{r_{e_n}}$ of $\mathbf{v}_{r_n}$ for each SOM. Accordingly, to cancel the effect of $\mathbf{v}_{r_{e_n}}$ on the real circle path (red color) in each SOM, $\mathbf{v}_{r_{e_n}}$ has been inverted to the opposite direction, then has been added to the real $\mathbf{v}_{t_n}$ which includes the error, to obtain the required final corrected position vector $\mathbf{v}_{t_{cn}}$ as shown in Figure 3.9b. This gives the robot the opportunity to correct the position each SOM to enhance the accuracy of tracking the reference trajectory. However, an advanced controller is needed to control the magnitude $\mathbf{v}_{t_{cn}}$, and to overcome any sudden changes or high fluctuations with different speeds. **Theoretically, in the ideal desired case** the feedback position has no error, hence, $\mathbf{v}_{t_n}$ will be error-free and the circle trajectory will be perfect, i.e., the amplitude of $\mathbf{v}_{r_n}$ is exactly equal to the circle radius of 100 mm. However, in practice, the amplitude of $\mathbf{v}_{r_n}$ is greater or less than the circle radius which is presented in Figure 3.9a by $\mathbf{v}_{r_0}$ and $\mathbf{v}_{r_1}$, respectively. Therefore, on one hand the robot feedback position error e_{r_n} can be calculated by computing the difference between the amplitude of the radial vector and the radius of the circle as:

$$e_{r_n} = 100 - \|\mathbf{v}_{r_n}\|_2. \tag{3.5}$$

While, on the other hand, the ballbar sensor error e_{b_n} can be calculated using the first method in subsection 3.2.2, in which e_{b_n} is assigned the value of the last reading from the ballbar recording buffer.

There are different methods to merge both mentioned errors (e_{r_n} and e_{b_n}) in the computations of the desired corrected tangential vectors $\mathbf{v}\hat{_{t_{cn}}}$ during controlling the robot motion for each SOM, which will be discussed in detail in the next subsection 3.4.2. However, assuming that the equivalent error e_{equ} is the summation of both errors in each SOM (just as an assumption to resume the computational method equations) as:

$$e_{equ_n} = e_{r_n} + e_{b_n}. \tag{3.6}$$

Considering the computation of $\hat{\mathbf{v}_{t_n}}$ in equation (3.3), $\hat{\mathbf{v}_{r_n}}$ in equation (3.2) and e_{equ_n} in equation (3.6), we can compute the corrected tangential vector $\hat{\mathbf{v}_{t_{cn}}}$ by:

$$\hat{\mathbf{v}_{t_{cn}}} = \hat{\mathbf{v}_{t_n}} - (e_{equ_n}\hat{\mathbf{v}_{r_n}}), \tag{3.7}$$

Finally, the desired corrected velocity vector $\hat{\mathbf{v}_{v_{cn}}}$ can be calculated by replacing equation (3.4) with:

$$\hat{\mathbf{v}_{v_{cn}}} = \upsilon_r \hat{\mathbf{v}_{t_{cn}}}. \tag{3.8}$$

3.4.2 Error Consideration

The new corrected tangential position vector $\hat{\mathbf{v}_{t_{cn}}}$ required by the robot to move around the circle is computed in each SOM. As mentioned in the previous subsection, the radial position vector $\mathbf{v}_{r_n}$ in equation (3.1), is mainly based on robot position feedback, while the corrected tangential position vector $\hat{\mathbf{v}_{t_{cn}}}$ in equation (3.7), is computed from both robot and ballbar error feedback based on the equivalent error e_{equ_n} in equation (3.6). Obviously, e_{equ_n} can be any combination of these two errors (robot and/or ballbar error), i.e., the mean, median, maximum, minimum, or a weighted sum of them. Our main concern is to know how to compute this equivalent error in the most efficient way to get accurate information and reflection of the robot real error. Apparently, it is preferred to use the ballbar error, which is more accurate than the robot error. Nevertheless, due to the way $\mathbf{v}_{r_n}$ is computed, as in equation (3.1), the contribution of the robot error e_{rn} is already fully involved in the computation of e_{equ_n} in equation (3.6). In order to analyze the system response using different error considerations, we propose to begin with three methods of calculating e_{equ_n} in equation (3.6), which are:

1. The robot feedback error e_{r_n} as:

$$e_{equ_n} = e_{r_n}. \tag{3.9}$$

2. The average error values of both robot feedback error e_{b_n} and ballbar error e_{b_n} as:

$$e_{equ_n} = \frac{e_{r_n} + e_{b_n}}{2}. \tag{3.10}$$

3. The ballbar feedback error e_{b_n} as:

$$e_{equ_n} = e_{b_n}. \tag{3.11}$$

The objective of these experiments is to provide the results for a comparative study and to have practical verification of the circular path planning in velocity mode using this first method. Therefore, three experiments are done according to different values of e_{equ_n} proposed in the previous sub-subsection. Hence, various aspects of the robot and ballbar specifications were taken into consideration. This planning ahead was performed to ensure that the results of each experiment properly reflect the real scenario in the best possible way before going ahead and start robot model identification, control simulation, and implementation phase. Circular trajectory planning on robot velocity mode is used typically as illustrated in section 3.4, the experiments setup parameters are summarized in Table 3.3. To ensure the accuracy of the experimental data, the synchronization in sampling time and values between the robot and ballbar error needs to be done during each SOM. If this synchronization is not achieved, the computation of e_{equ_n} will be incorrect.

3.4.3 Time and Value Synchronization

In the following, we briefly delve into the details of doing a synchronization between the robot and the ballbar in the time and the value.

- **Time Synchronization:** as mentioned in subsection 3.2.2, the ballbar sampling rate is 1000 Hz, meaning that measurement is recorded by the ballbar buffer every 1 ms as stated on ballbar manual (Renishaw, 2016), regardless of the minimum time of reading from the ballbar recording buffer which is every 30-40 ms as proved experimentally, see Table 3.3. Two methods of reading from the ballbar buffer are introduced in subsection 3.2.2. After more investigation, we discovered that the ballbar is designed to record the robot errors and show them at the end of movement (Renishaw, 2016). Sometimes, it is used in real-time, but with slow sampling time, more than 100 ms. In other words, due to its slow reading time, it is used to plot the robot or machine errors, but not to control the robot movement. Therefore, in real-time applications, the buffer updated reading time is considered rather than the recording sampling rate. Ballbar buffer reading time is experimentally validated when the first method of reading the ballbar buffer is used, it is noticed that the ballbar updating reading sampling time is 100 ms, that is much slower than the robot reading sampling time of 30-40 ms, i.e., for every three or four readings from the robot, one reading of the ballbar is updated. That is to say, the robot buffer reading sampling rate is approximately 28.5 Hz versus ballbar updating recording sampling rate of 10 Hz. To conclude, we can say that although the readings of ballbar are accumulated inside the recording buffer every 1 ms, they can be received from the latter to the user interface every 30-40 ms, and cannot be updated before a time of 100 ms is elapsed. Therefore, the sampling time for all the experiments is chosen to be equal to 100 ms taking into consideration that matching the ballbar reading time with that from the robot movement commands is essential.

- **Values Synchronization:** not only the time but also the reading values between robot and ballbar is needed to be synchronized, as there is a difference in the error reading values between the robot and ballbar which is fluctuating among – 0.2 and 0.2 mm as shown in the experiment graph in Figure 3.14.
- **Synchronization solution:** based on the aforementioned discussion, by considering the difference in reading sampling time and that in the reading values between the robot and ballbar, the best way to use this error in producing the tangential position vector $\mathbf{v}_{t_n}$ is to take values accumulated in ballbar buffer even if the latest value is not updated yet in the same time of obtaining each reading from the robot position feedback when using the second method mentioned in that section proposed by equation (3.10). Further, the most expressive calculation method of e_{equ_n} is employed as proposed in subsection 3.4.2. This is anticipated to give error values with better consistency than switching between robot and ballbar error values once the latter is updated. The three equivalent error e_{equ_n} calculation methods resulted graphs are shown in blue color in Figures 3.13b, 3.16a and 3.19b.

3.4.4 Methods of Generating Reference Trajectory for the Proposed Controller

For the proposed controller, it is essential to determine the robot desired trajectories in yz-plane during the whole circular motion to be compared by the controller with the real system output then and based on that, the controller will take its corrected action. In our case, the angle of rotation θ_n of each SOM plays a vital role in determining y_d and z_d trajectories of the circle which are given by:

$$y_{d_n} = r \sin \theta_n. \tag{3.12}$$

$$z_{d_n} = r \cos \theta_n + 172. \tag{3.13}$$

where r is the desired circle radius (100 mm), n is the current SOM index and the bias value 172 mm is the z-coordinate of the center of the circle as illustrated in Table 3.3. The key point is being able to accurately compute the angle of rotation θ_n of each SOM during the circular motion. Therefore, three different methods are proposed to compute θ_n and then to compute the desired trajectories y_d and z_d, thus and upon the results, the best relevant method is chosen. In the first two methods, θ_n is computed at each SOM during the robot's movement in its circular motion, while in the third method, θ_n is computed after the robot completes its whole circular movement. The three calculation methods that can be used to compute angle θ_n are summarized as follows:

1. The angle $\theta_n(1)$ of the first method is computed by applying the inverse tangent (arctangent) function to the quotient of the robot feedback positions of y_{r_n} and z_{r_n} coordinates, that is:

$$\theta_n(1) = \text{atan2}\big(y_{r_n}, z_{r_n}\big). \tag{3.14}$$

   Then substituting by this computed $\theta_n(1)$ in equations (3.12 and 3.13) to calculate the desired positions y_d and z_d which are plotted in red and denoted as $y_d(1)$ and $z_d(1)$ on Figures 3.10 and 3.11, respectively.

2. The angle $\theta_n(2)$ of the second method is computed from the calculated traveled distance y_{s_n} in Y direction and z_{s_n} in Z-direction, which is expressed as:

$$y_{s_n} = \mathbf{v}_r t_{som}. \tag{3.15}$$

$$z_{s_n} = 100. \tag{3.16}$$

where t_{som} is the time of each SOM. Hence, the rotation angle is given by:

$$\theta_n(2) = \text{atan2}\big(y_{s_n}, z_{s_n}\big). \tag{3.17}$$

Then substituting by this computed $\theta_n(2)$ in equations (3.12 and 3.13) to calculate the desired positions y_d and z_d which are plotted in yellow and denoted as $y_d(2)$ and $z_d(2)$ on Figures 3.10 and 3.11, respectively.

3. The rotation angle $\theta_n(3)$ of the third method is computed after the robot finishes circle trajectory, by dividing the radial circumference by the total number of segments as:

$$\theta_n(3) = \frac{2pi}{N}. \tag{3.18}$$

where N is the total number of segments of the whole trajectory. Then substituting by this computed $\theta_n(3)$ in equations (3.12 and 3.13) to calculate the desired positions y_d and z_d which are plotted in purple and denoted as $y_d(3)$ and $z_d(3)$ on Figures 3.10 and 3.11, respectively.

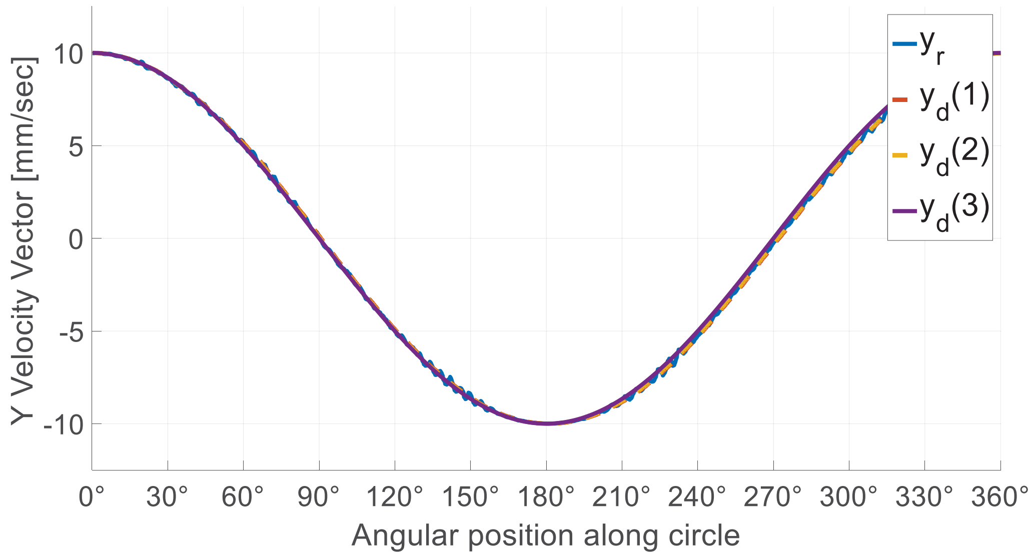
The aforementioned three methods of computing the desired trajectory in Y and Z directions of the circle are the regular approaches to have the reference circular trajectory for the controller, but we are also proposing an irregular approach that can be used as a reference trajectory to the control system. This is an error-based approach, which considers that the reference trajectory is the error itself not the geometrical coordinates of Y and Z directions. In this approach, the required reference error is used as an input raw data to the system, which is zero. Thus, if the output error from the system is not zero, the proposed controller adapts the system input to force the output being zero. This will be discussed in more detail in section 4.3.

The results of the related graphs which are presented in Figures 3.10 and 3.11 illustrate: (1) robot real circular trajectory computed by the ballbar in Y, Z directions denoted as y_r, z_r, in blue color; (2) the first calculation method of the desired reference $y_d(1)$, $z_d(1)$, in red color; (3) the second method of the desired reference $y_d(2)$, $z_d(2)$, in yellow color; and (4) the third method of the desired reference $y_d(3)$, $z_d(3)$, in purple color. By analyzing the result in these graphs, we notice that the robot's actual real values (blue color) in Y and Z directions are mostly fluctuating around $y_d(2)$ and $z_d(2)$ (yellow color) reference trajectories of the second method. Generally, as a result of these graphs, we can not find a definite conclusion, because a lot of experiments need to be done and a rigorous analysis method such as computing the RMS error should be used to get confident results and have the right conclusion. Consequently, we can say that it is better to start with the irregular approach mentioned above for more results consistent and confident and to be apart from the uncertainties related to the regular approach of the three methods mentioned above.

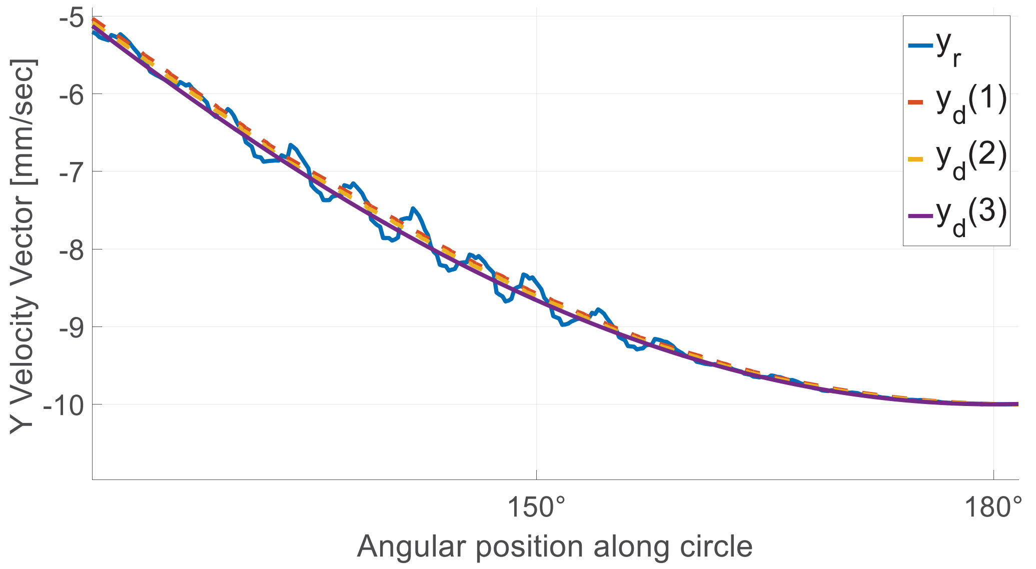
3.4.5 Results Analysis

In order to compute the corrected velocity vector $\mathbf{v}_{\hat{v_{c_n}}}$ in equation (3.8) which is required to move the robot, the equivalent error has been assumed to be the summation of both ballbar and robot errors as in equation (3.6), see subsection 3.4.1. However, we will test all the three methods of computing the equivalent error mentioned in subsection 3.4.2. The results are divided into three main sections according to the equivalent error e_{equ_n} calculations used in each experiment and each section includes the following results graphs:

- The ballbar error e_{b_n} computed by the first ballbar calculation method (Last reading) and the second calculation method (All recorded buffer readings) which are used to read from the ballbar recorded buffer as mentioned in subsection 3.2.2.
- The robot error e_{r_n} calculated from the robot feedback based on the difference between the radial position vector amplitude and the circle radius in equation (3.5) during each SOM.
- The equivalent error e_{equ_n} of robot e_{r_n} and ballbar e_{b_n} errors as mentioned in subsection 3.4.2 and computed by equations (3.9, 3.10 and 3.11).

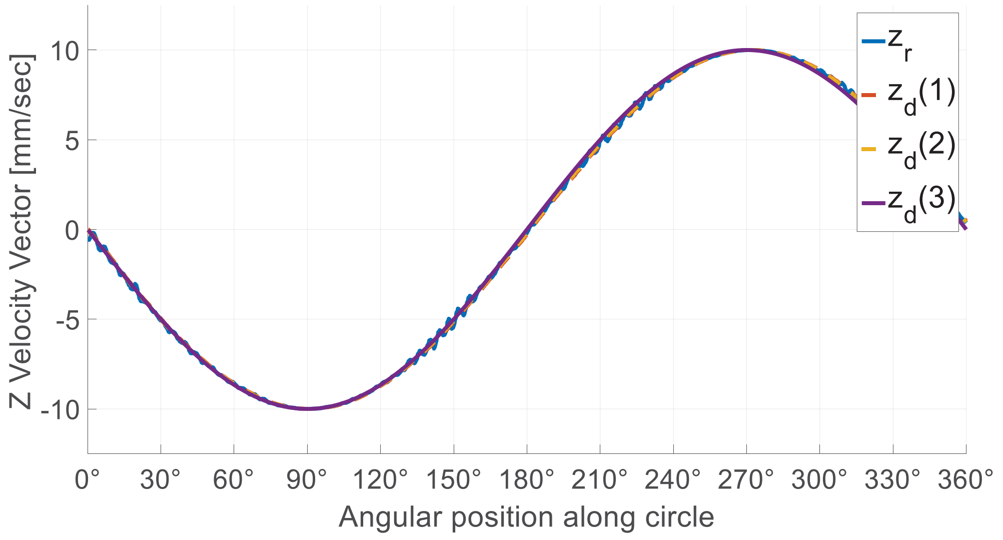


a) Robot's real and desired reference trajectories calculated by the three methods

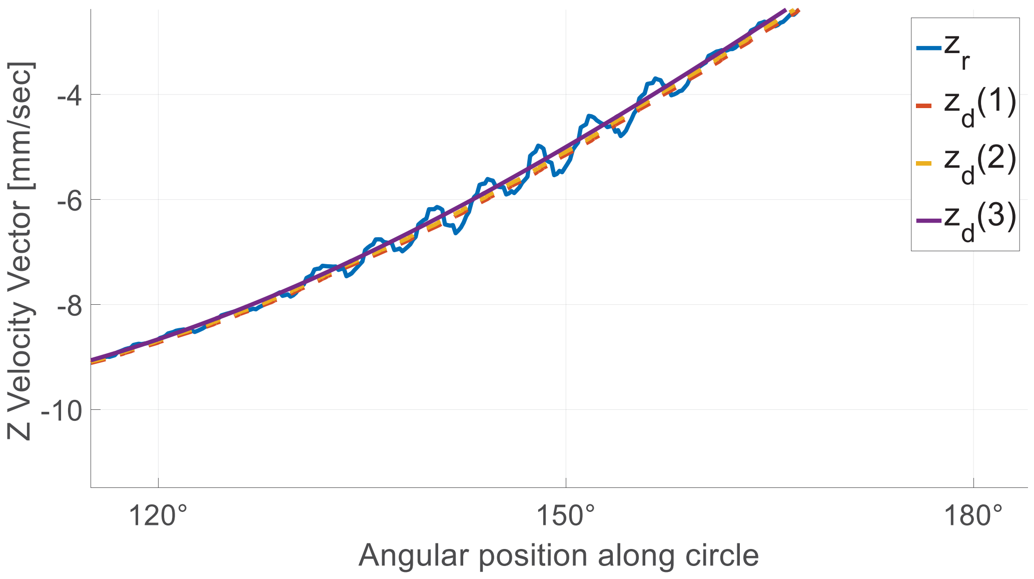


b) Zooming in the first half of the robot circular motion

Figure 3.10 Real robot trajectory y_r, in blue color and desired reference trajectories: $y_d(1)$ of first calculation method, in red color; $y_d(2)$ of second calculation method, in yellow color; and $y_d(3)$ of third calculation method, in purple color, in Y direction



a) Robot's real and desired reference trajectories calculated by the three methods

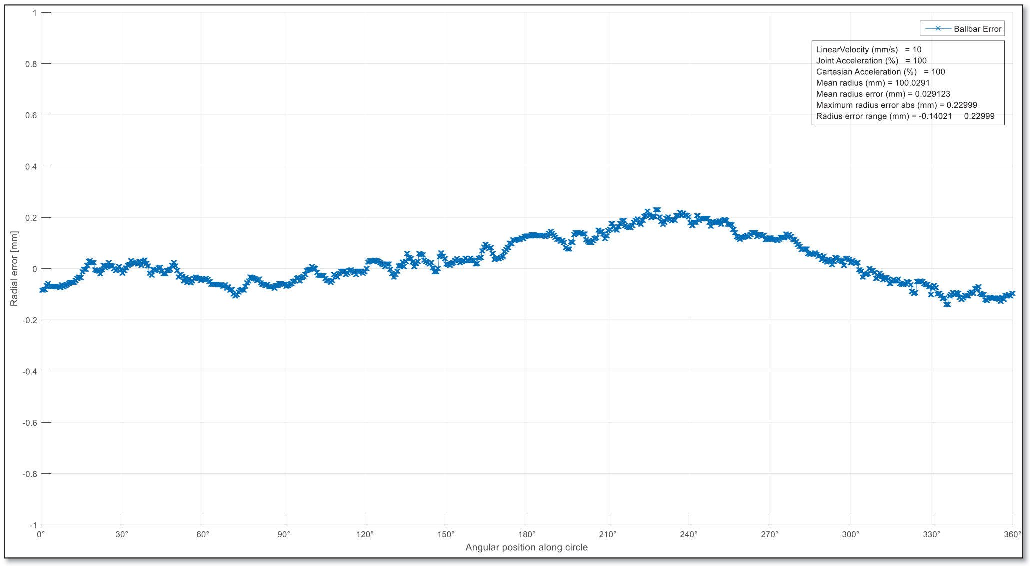


b) Zooming on the first half of the robot circular motion

Figure 3.11 Real robot trajectory z_r, in blue color and desired reference trajectories: $z_d(1)$ of first calculation method, in red color; $z_d(2)$ of second calculation method, in yellow color and $z_d(3)$ of third calculation method, in purple color, in Z direction

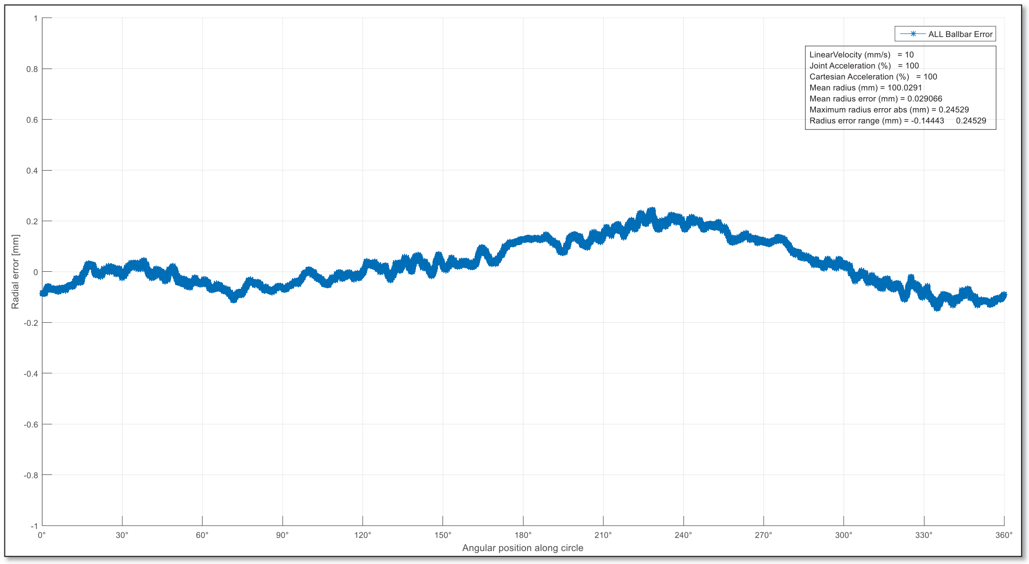
3.4.5.1 Robot circular movement using Robot Error only

The first method is employed to compute the equivalent error e_{equ_n} in equation (3.9) while the robot moving in a circular motion at TCP low velocity of 10 mm/s, as a result, we can see that the circular motion is not perfect, and the ballbar displays circular deviations along the whole circle. Comparing the two methods of reading from the ballbar buffer, it is noticed that the ballbar radius errors are similar in ranges as seen in Figure 3.12, the ranges are –0.14 to 0.229 mm and -0.14 to 0.245 mm, respectively, in spite of the density of the readings outcomes in Figure 3.12b is higher than the one in Figure 3.12a, that is because all the ballbar buffer readings with a full-sampling rate (1000 Hz) of the second method is used to plot the graph of Figure 3.12b. Analyzing the results, seemingly these radius variations measured by the ballbar exist because the robot positions have some errors and these positions are not exactly in the right place as that the robot expected to be, i.e., they are not perfect. On the contrary, the robot error e_{r_n} in Figure 3.13a which is measured by the robot's positions feedback, shows as if the e_{r_n} has no error during the whole circular motion, i.e., the robot assumes that the positions are perfect (error free), whereas the ballbar measures error variations during the same robot's circular motion, which is more precise than e_{r_n} feedback and contradict with robot's error results. Figure 3.13b shows the results of three types of error: (1) average error, in blue color; (2) robot error, in red color; and (3) Ballbar error, in yellow color. Obviously, the figure demonstrates how the robot error is misleading showing zero values while the ballbar buffer measures the errors. The average error and the ballbar error, in this case, are not used to correct the robot motion but only the robot error in equation (3.9), they are just recorded during the same robot's circular motion and plotted in the same graph. This ballbar error reflects the real robot error circular motion deviations without adding any controller which can be used in the future to be compared with the error in the robot's position mode, supposing that both modes are measured at the same sampling rate and time. Logically, they should be similar in values. Therefore, the title of this sub-subsection can be named as corrections with zero corrections or robot velocity movement with no corrections since the robot thinks the end-effector is in the ideal pose while we are depending only on the robot's feedback to move in the circular motion.



```
LinearVelocity (mm/s)   = 10
Joint Acceleration (%)   = 100
Cartesian Acceleration (%)   = 100
Mean radius (mm) = 100.0291
Mean radius error (mm) = 0.029123
Maximum radius error abs (mm) = 0.22999
Radius error range (mm) = -0.14021     0.22999
```

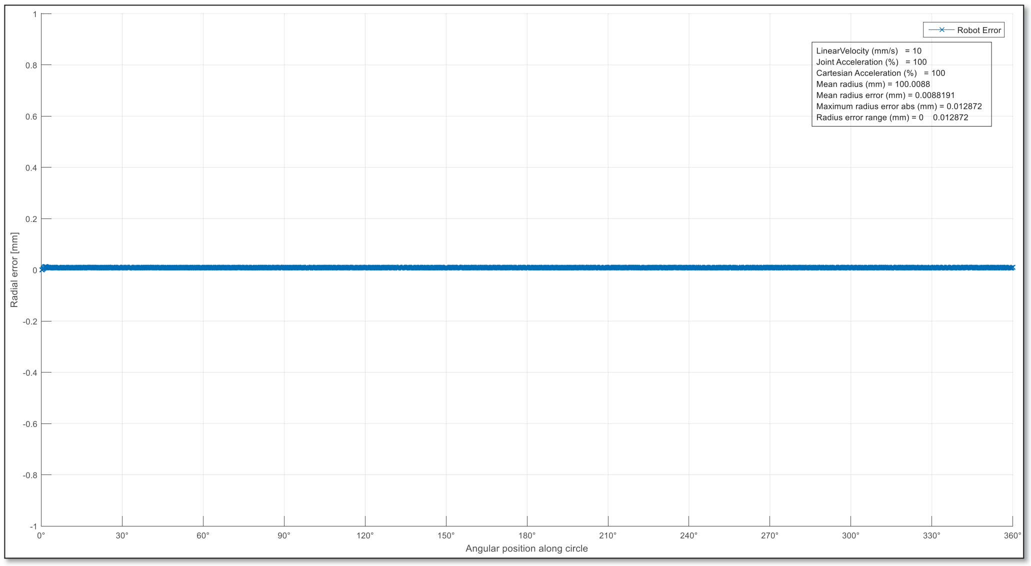
a) The first method "Ballbar buffer Last reading"



```
LinearVelocity (mm/s)   = 10
Joint Acceleration (%)   = 100
Cartesian Acceleration (%)   = 100
Mean radius (mm) = 100.0291
Mean radius error (mm) = 0.029066
Maximum radius error abs (mm) = 0.24529
Radius error range (mm) = -0.14443     0.24529
```

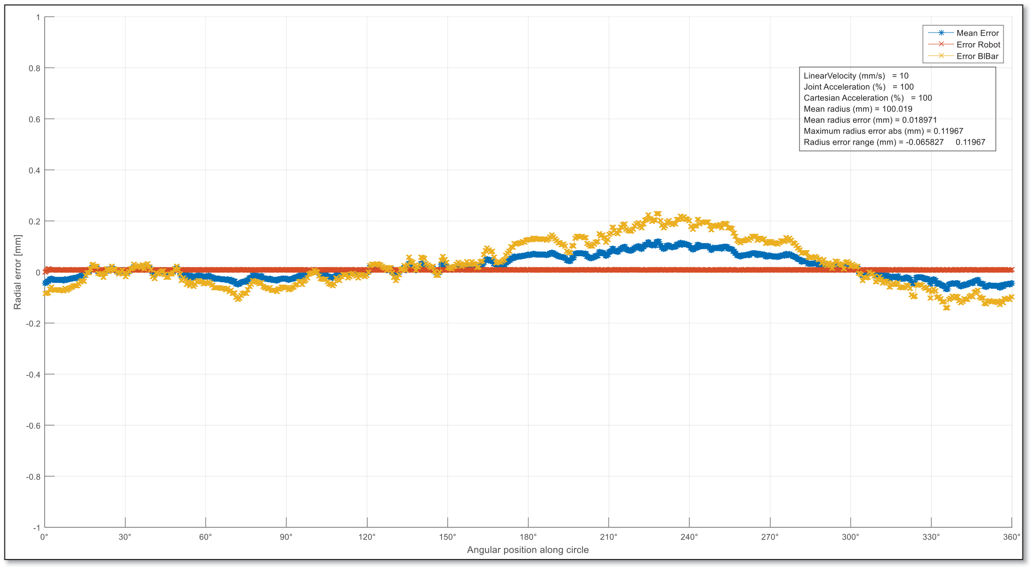
b) The second method "Ballbar buffer All readings"

Figure 3.12 The difference between the two methods of reading the ballbar buffer error e_{b_n} when the equivalent error e_{equ_n} is equal to only the robot error e_{r_n}



LinearVelocity (mm/s) = 10
Joint Acceleration (%) = 100
Cartesian Acceleration (%) = 100
Mean radius (mm) = 100.0088
Mean radius error (mm) = 0.0088191
Maximum radius error abs (mm) = 0.012872
Radius error range (mm) = 0 0.012872

a) Robot Error e_{r_n}



LinearVelocity (mm/s) = 10
Joint Acceleration (%) = 100
Cartesian Acceleration (%) = 100
Mean radius (mm) = 100.019
Mean radius error (mm) = 0.018971
Maximum radius error abs (mm) = 0.11967
Radius error range (mm) = -0.065827 0.11967

b) Robot error, ballbar error and the average of both errors

Figure 3.13 Error graphs when the equivalent error e_{equ_n} is equal to only the robot error e_{r_n}

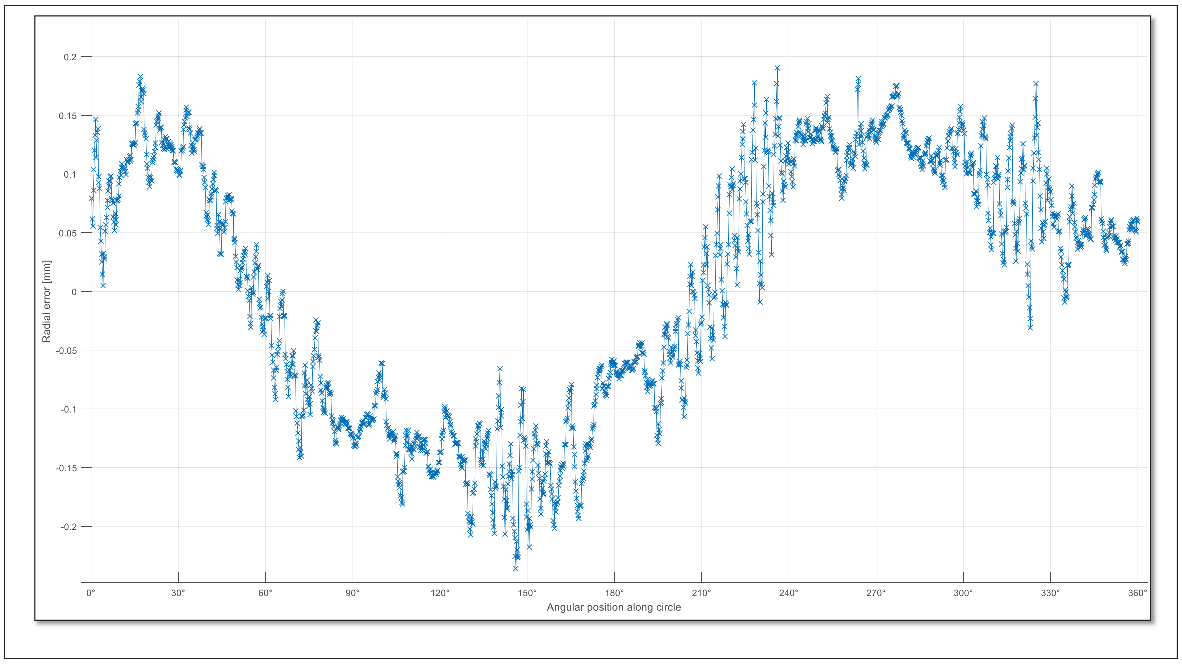
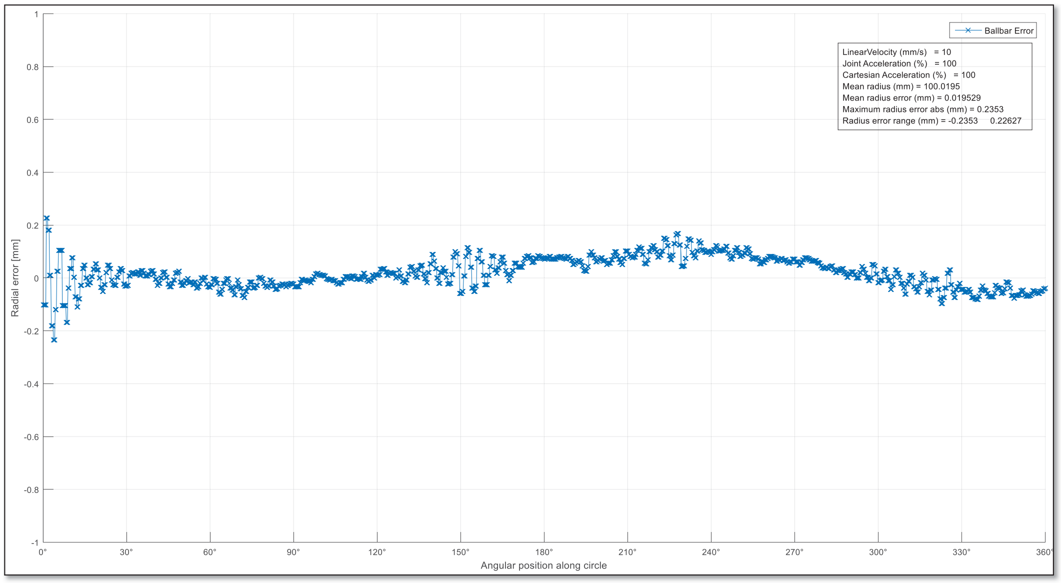


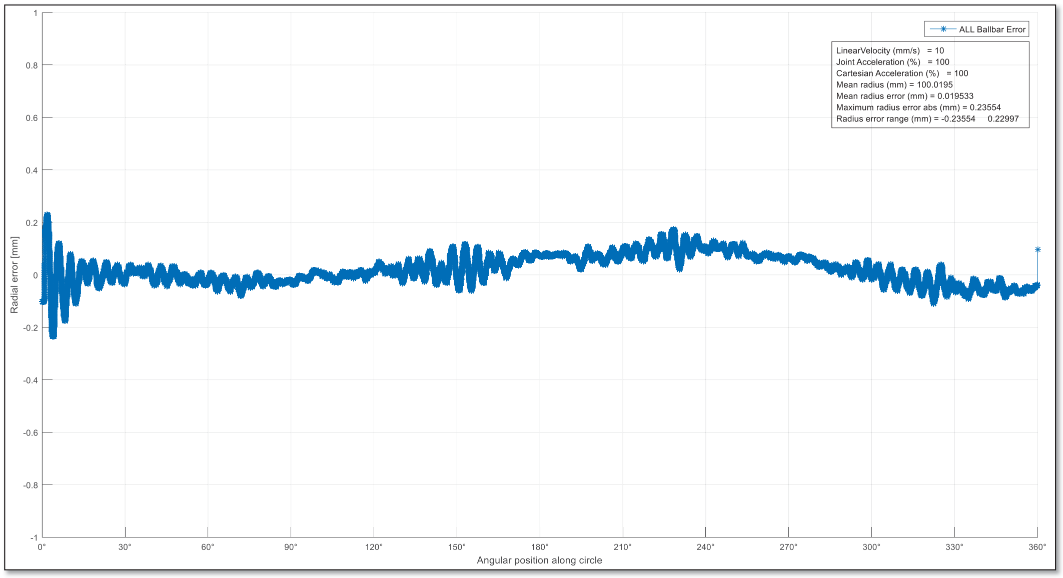
Figure 3.14 Difference in reading values between e_{r_n} and e_{b_n}

3.4.5.2 Robot circular movement using the Average Error of both robot and ballbar errors

Similarly, by applying the second method of the equivalent error e_{equ_n} computation in equation (3.10) at the same robot's TCP velocity of 10 mm/s, it can be said that the circular deviations along the whole circle exist as seen on ballbar error graphs in Figure 3.15. The two methods of reading from the ballbar buffer in Figures 3.15a and 3.15b are again frequently similar as can be seen from one graph to another with the same previous remark of density outcomes between the two Figures. With regard to that in this case, the robot error in Figure 3.16b, demonstrates circular deviations the same as what the ballbar sensor attests. The ballbar radius error range is –0.235 to 0.226 mm, while the robot radius error range is –0.151 to 0.37 mm. Likewise, Figure 3.16a justifies the results of three types of error: (1) average error, in blue color; (2) robot error, in red color; and (3) Ballbar error, in yellow color. In this case, the robot error (red color) confirms a similar ballbar error (yellow color) but in the opposite direction as shown in the zoom view of Figure 3.17. Although at the first glance, this contradicts the hypotheses that both errors should be in the same direction except that the values may be different, with deep thought, it is a logical consequence since the ballbar error which is involved in the robot's motion this time as in equation (3.10), deviates from the perfect positions (zeros values) presented by robot error in Figure 3.13a of the previous subsection. In other words, the robot is ordered to correct its movement in the opposite direction to the equivalent error, this concludes that the robot is doing a good job based on the corrected velocity vectors given to the robot. As a result, for instance, at the beginning of motion at angular position of 10° in Figure 3.17, the robot error values are in the opposite direction of the equivalent error e_{equ_n} (average error in this case), this e_{equ_n} documents the error used to correct $\mathbf{v}_{\hat{t_{cn}}}$ in equation (3.8), that is why we have some oscillations in the ballbar error in Figure 3.16a. In other words, the correction action we take is based on the average of the ballbar error and the robot reaction to the correction error, each SOM. It is worthwhile to mention that the ballbar error is the only error that attests and reflects the real robot error in all cases. However, particularly in this method of e_{equ_n} calculations, the correction action of $\mathbf{v}_{\hat{t_{cn}}}$, is computed from merging between both robot and ballbar errors calculations.

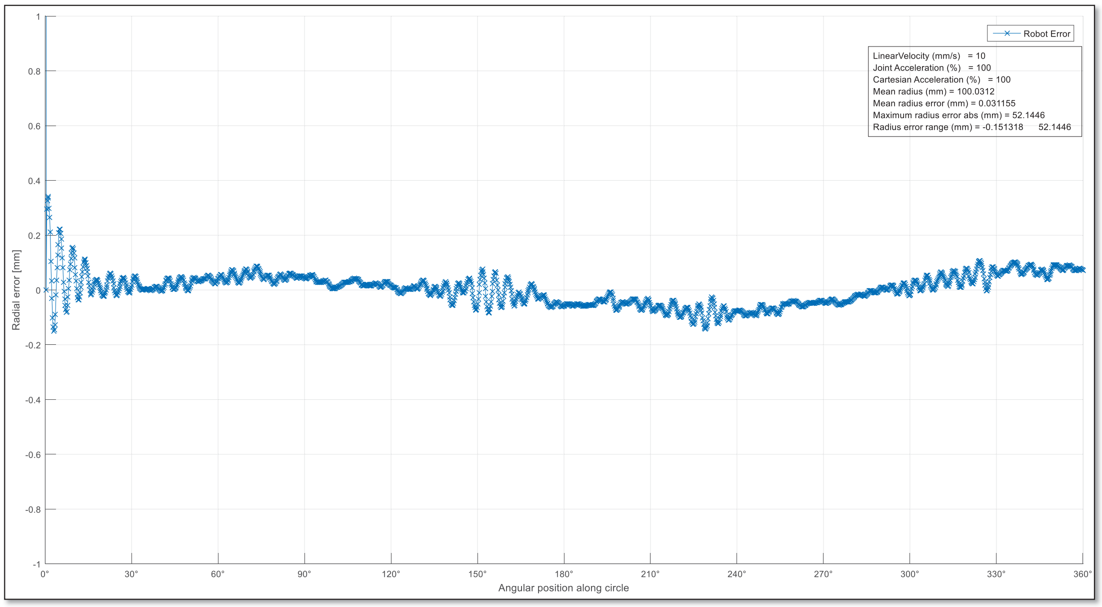


a) The first method "Ballbar buffer Last reading"



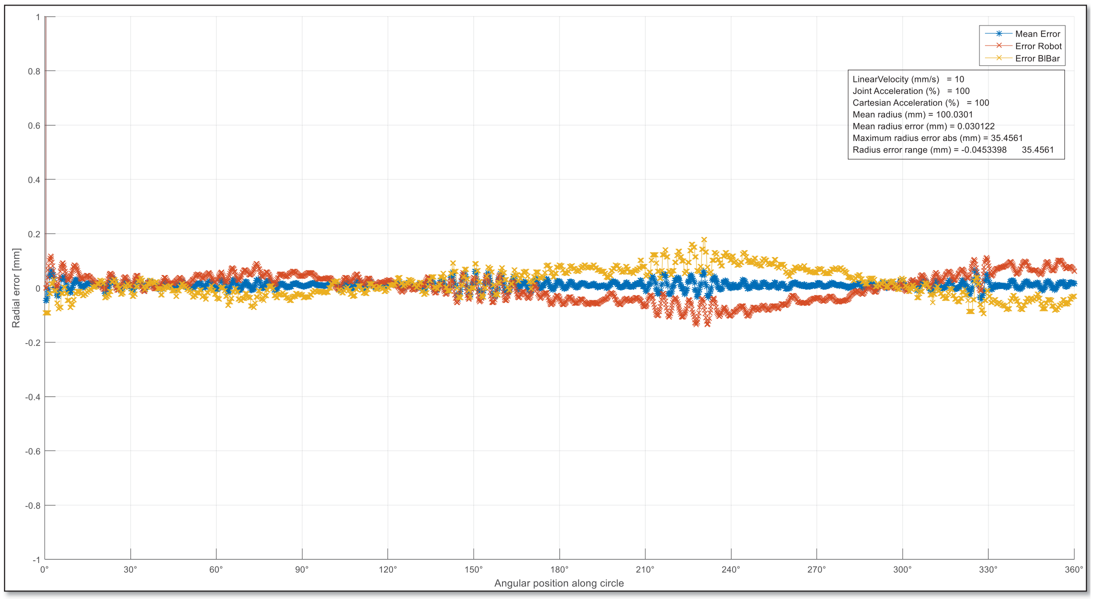
b) The second method "Ballbar buffer All readings"

Figure 3.15 The difference between the two methods of reading the ballbar buffer error e_{b_n} when the equivalent error e_{equ_n} is equal to the average of both robot and ballbar error values



LinearVelocity (mm/s) = 10
Joint Acceleration (%) = 100
Cartesian Acceleration (%) = 100
Mean radius (mm) = 100.0312
Mean radius error (mm) = 0.031155
Maximum radius error abs (mm) = 52.1446
Radius error range (mm) = -0.151318 52.1446

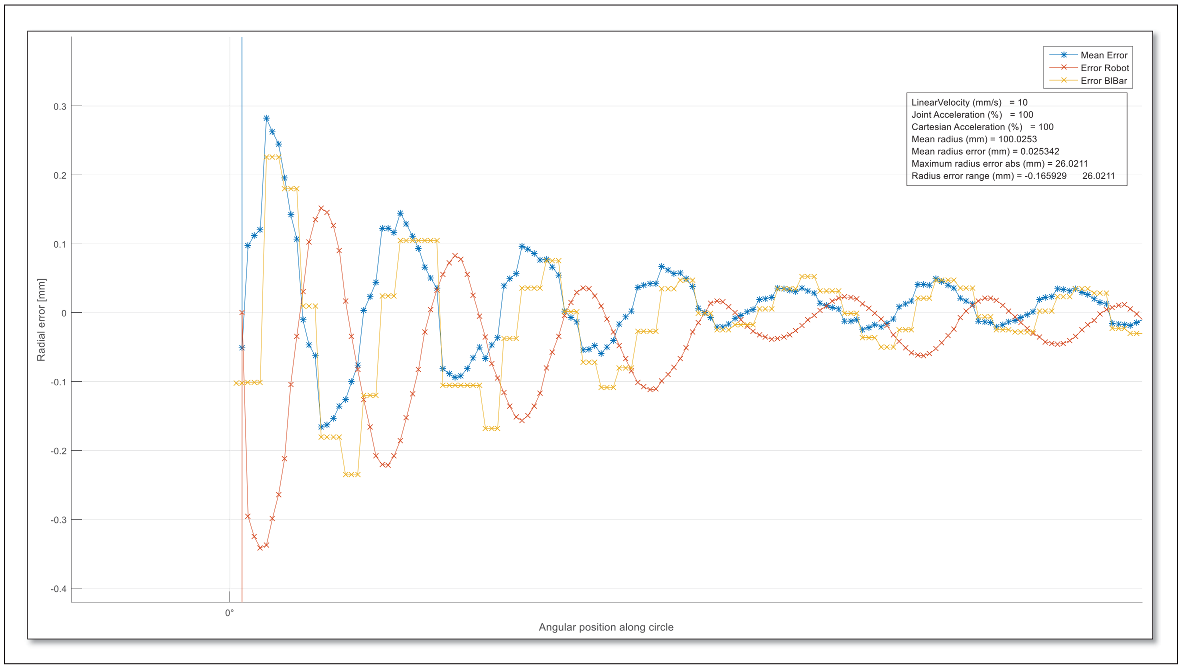
a) Robot Error e_{r_n}.



LinearVelocity (mm/s) = 10
Joint Acceleration (%) = 100
Cartesian Acceleration (%) = 100
Mean radius (mm) = 100.0301
Mean radius error (mm) = 0.030122
Maximum radius error abs (mm) = 35.4561
Radius error range (mm) = -0.0453398 35.4561

b) Robot error, ballbar error and the average of both errors

Figure 3.16 Error graphs when e_{equ_n} is equal to the average of both robot and ballbar error values

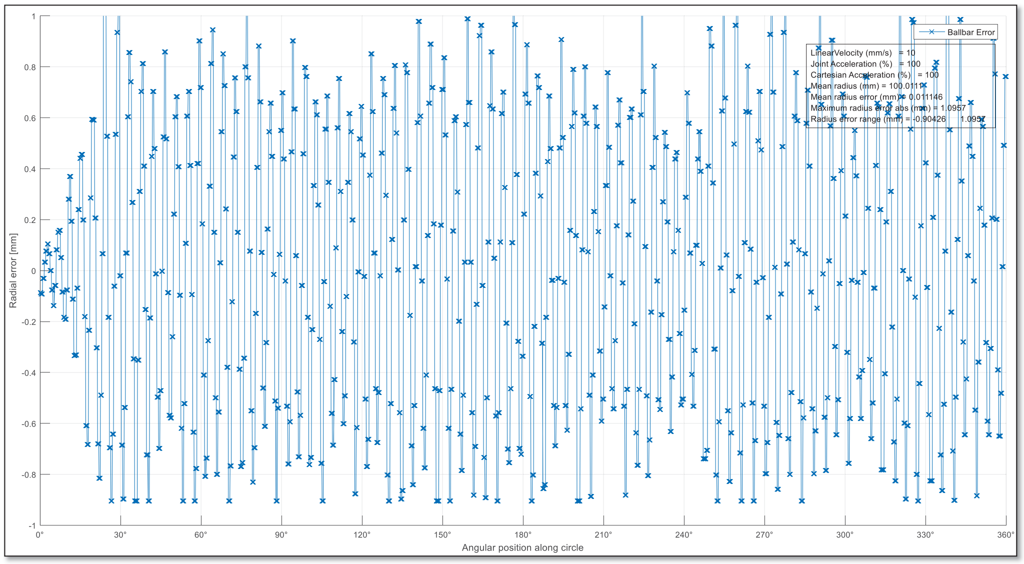


```
LinearVelocity (mm/s)   = 10
Joint Acceleration (%)   = 100
Cartesian Acceleration (%)   = 100
Mean radius (mm) = 100.0253
Mean radius error (mm) = 0.025342
Maximum radius error abs (mm) = 26.0211
Radius error range (mm) = -0.165929     26.0211
```

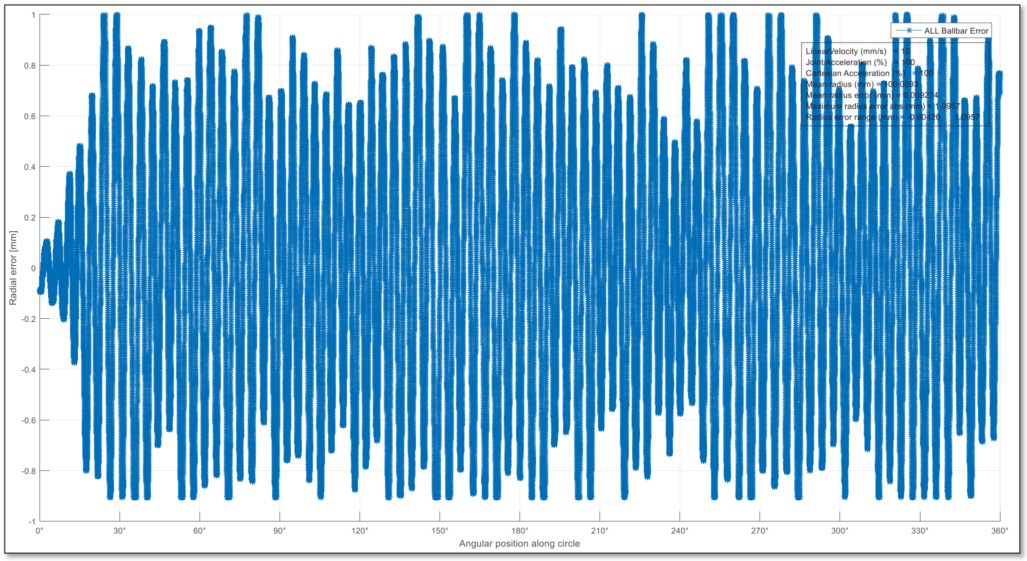
Figure 3.17 Zoomed view of the graph showing errors difference directions, when e_{equ_n} is equal to the average of both robot and ballbar error values

3.4.5.3 Robot circular movement using Ballbar Error only

Finally, by considering only the ballbar error when calculating e_{equ_n} in equation (3.11), we emphasize the same conclusion about ballbar error existence, similarity in radius error ranges and the ballbar buffer readings density outcomes of Figures 3.18a and 3.18b. Comparing Figure 3.19a with Figure 3.19b, we can conclude that all the robots and ballbar errors are in the same directions except that error values may be different. Supposedly, on the spot, this conflicts with the direction antibiosis assumption in the previous sub-subsection, but with deep analysis, we conclude that this is obviously again considered as a logical consequence since the ballbar error, in this case, accustoms to the direct corrections of $\mathbf{v}_{\hat{v_{cn}}}$ in equation (3.8) with no contribution of the robot errors, consequently, the robot is showing its error in the same direction of ballbar error. This validates our assumption in the previous sub-subsection that when the robot reaction error each SOM is not involved in this e_{equ_n} calculation method here, all the errors meet in the same direction and almost similar in the values as well.

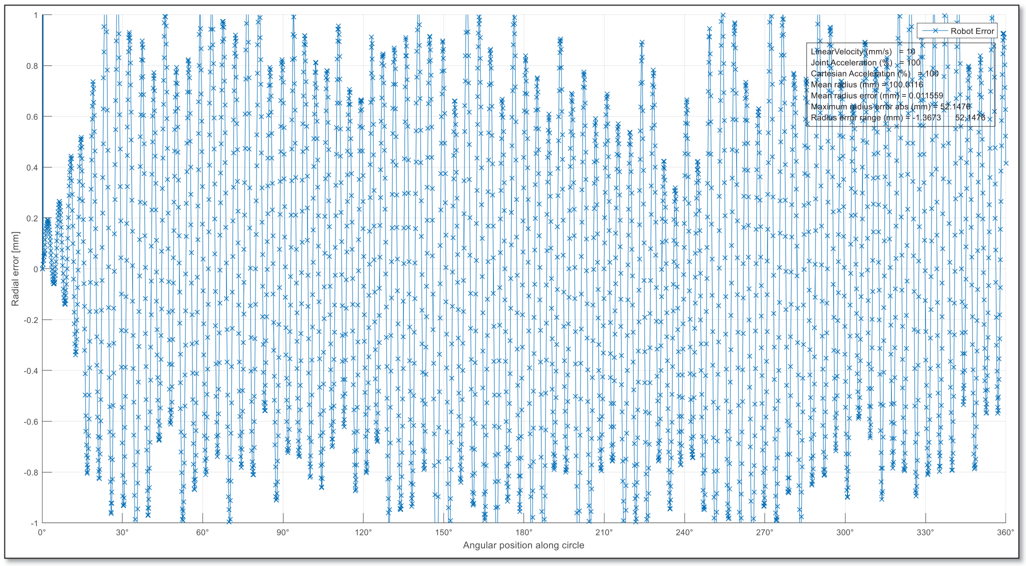


a) The first method "Ballbar buffer Last reading"

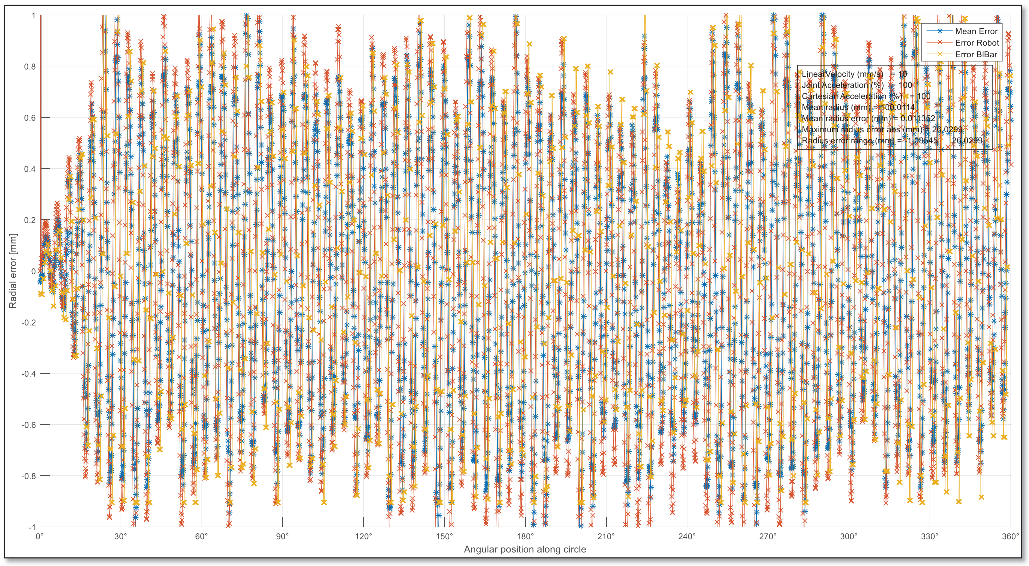


b) The second method "Ballbar buffer All readings"

Figure 3.18 The difference between the two methods of reading the ballbar buffer error e_{b_n} when the equivalent error e_{equ_n} is equal to only the ballbar error e_{b_n}



a) Robot Error e_{r_n}



b) Robot error, ballbar error and the average of both errors

Figure 3.19 Error graphs when the equivalent error e_{equ_n} is equal to only the ballbar error e_{b_n}

3.4.6 Outcomes of the first method of designing the robot's circular trajectory

Based on the aforementioned analysis and discussion, we determine the type of robot movement control commands that can be used in the desired robot's circular motion; besides, the ballbar reading buffer method. Moreover, the final robot correction method is chosen, in addition to an irregular approach is introduced in terms of determining the robot reference trajectory that is required by the proposed controller as follows:

1. In terms of Robot Movement Control Commands:

   Ostensibly, the circular trajectory can be designed either by using position mode or velocity mode, as recognized from subsection 3.1.1. Notwithstanding, the velocity mode is preferable to be used in advanced applications (Mecademic, 2020), as long as dynamic path corrections are required. Therefore, in this project, the desired trajectory is presented in the velocity mode.

2. In terms of ballbar Reading Buffer Method:

   As reported by subsection 3.2.2, there are two methods to read from the ballbar buffer. The first method will be used on the grounds that, it is the only method that can be used in an on-line trajectory. Concurrently, all the values accumulated inside the ballbar buffer each SOM will be taken into consideration, even if the latest value is not updated yet; exactly as mention in the first paragraph of this subsection. However experimentally, we discovered while using this first method of the robot's circular motion that the sampling time is not fixed to 0.1 s has been adjusted, although it was fixed in the MATLAB code. By doing more analysis, we found that the reason for this issue is because of MATLAB computational delay, besides communication delay time using the two robot's ports, that's why we are proposing the second method of the robot's circular motion in the following section 3.5, which guarantees to have the fixed sampling time we desired.

3. In terms of Robot Movement Correction Methods:

   The trade-off between using the second and/or the third methods of calculation's acknowledgment in subsection 3.4.2 is not a trivial task. It is best to start using the third method of carrying out the corrections using only the ballbar error since it is the only error that attests and reflects the real robot error in all cases, as reported by experimental results in subsection 3.4.5. However, ballbar slow reading sampling time combined with robots fast-moving, demonstrated the movement oscillations in Figures 3.18 and 3.19, which is hard to be controlled. On the other hand, to merge the robot and ballbar errors while the circular movement is taking place is not correct because the robot reaction is considered in the correction; read the last paragraph of sub-subsection 3.4.5.2.

4. In terms of Robot Reference Trajectories:
   According to the aforementioned discussion in subsection 3.4.4, it is better to use the irregular approach (error-based approach) which considers that the reference trajectory is the error itself not the geometrical coordinates of Y and Z directions. This will exclude the uncertainties around determining which calculation method of the regular approach can reflect the real robot performance. The required reference errors will be used later in designing the identified robot model for the purpose of simulation as input raw data to the system, which is zero values. If the output errors from the identified and/or the real robot systems are not zero values, the proposed controller adapts the system input to force the output to be zero values.

In this method, the amount of measured data can be considered enormous, although the output data is huge, it has been used in the previous statistical analysis. The aim of that analysis is to draw the previous conclusions, together with the other observations. The aim of the previous long discussion is to generalize the results to a wider phenomenon if there is no indication of confounding variables that may pollute the results. However, we discovered that the real sampling time of all the experiments using this method was not fixed and was fluctuating between 0.085 to 0.115 s. This sampling time instability affects the robot circular motion. Thus, we cannot count on this method to accurately controlling the robot in real-time. Hence, we designed the robot circular motion using a different method in the following section 3.5.

3.5 The Second Method of Designing the Robot's Circular Trajectory

The previous conclusion of the first method in subsection 3.4.6 has been observed following the circular path using this second method. For instance, the robot's circular motion in yz-plane in velocity mode, can be achieved by sending different velocity vectors at specific times (e.g.,sampling time) to the robot. This is started by dividing the circular path into a pre-calculated number of points in yz-plane based on experiment sampling time, robot's linear velocity, and distance traveled each SOM. Then by computing the position vector between every two successive points, the related desired velocity vector can be computed. Figure 3.20a illustrates the desired circular path in yz-plane, including position and velocity vectors, polar angles and distance traveled each SOM. To delve into detailed steps, the circular trajectory can be achieved in velocity mode by the following steps:

1. Determine the sampling time t_s of sending the velocity vector commands to the robot and for the fact that we want to measure the radial error using the ballbar as feedback to the proposed controller, it is better to synchronize the sampling time with the ballbar buffer updated reading time of 100 ms. Rather than that, if the sampling time is reduced, the ballbar reading data will be repeated and will not reflect the real reading data each SOM, which is extremely important for the controller feedback. Thus, the experiment sampling time is adjusted to:

$$t_s = 0.1 \text{ s}. \tag{3.19}$$

2. Determine the desired robot TCP velocity υ_r, then compute the fixed distance d traveled by the robot between every two successive corner points during each SOM as:

$$d = \upsilon_r t_s. \tag{3.20}$$

3. Compute the polar step angle θ_p between those successive corner points from the distance traveled d and the desired circle radius, $r = 100$ mm, as:

$$\theta_p = 2 \sin^{-1}\left(\frac{d}{2r}\right). \tag{3.21}$$

The polar angle θ_p is a fixed value during the whole robot circular motion in the same experiment. It changes only when the TCP velocity υ_r changes in a new experiment.

4. Compute the total polar angle θ_n as:

$$\theta_n = n\theta_p, \tag{3.22}$$

where n is the current SOM index.

5. Compute the robot's TCP desired coordinates in the yz-plane, y_{d_n} and z_{d_n}, at each SOM as:

$$y_{d_n} = r \sin \theta_n \tag{3.23}$$

$$z_{d_n} = r \cos \theta_n + 172, \tag{3.24}$$

where 172 is the z coordinate of the center of the circle.

6. Compute the robot's TCP position vectors $\mathbf{v}_{p_n}$ between every two successive points in the yz-plane and their unit vectors $\hat{\mathbf{v}}_{p_n}$ at each SOM as:

$$\mathbf{v}_{p_n} = \begin{bmatrix} y_{d_{n+1}} - y_{d_n} \\ z_{d_{n+1}} - z_{d_n} \end{bmatrix}, \tag{3.25}$$

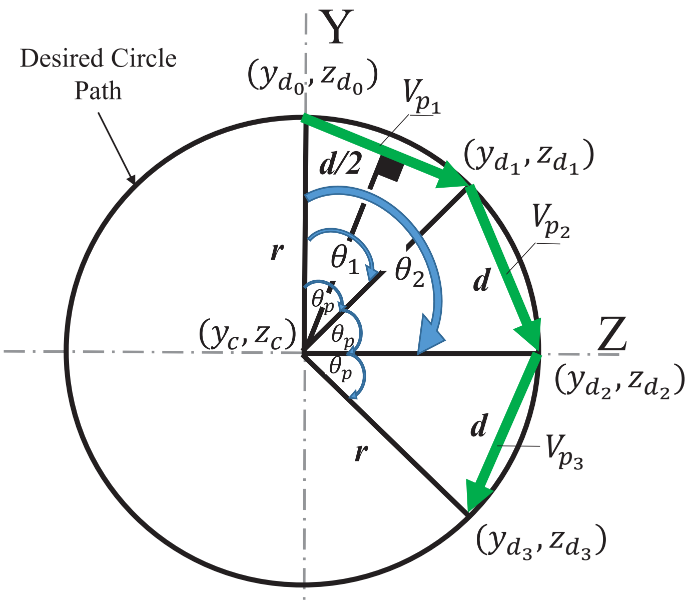
$$\hat{\mathbf{v}}_{p_n} = \mathbf{v}_{p_n} / \|\mathbf{v}_{p_n}\|, \tag{3.26}$$

where $\|\cdot\|$ denotes the ℓ_2-norm.

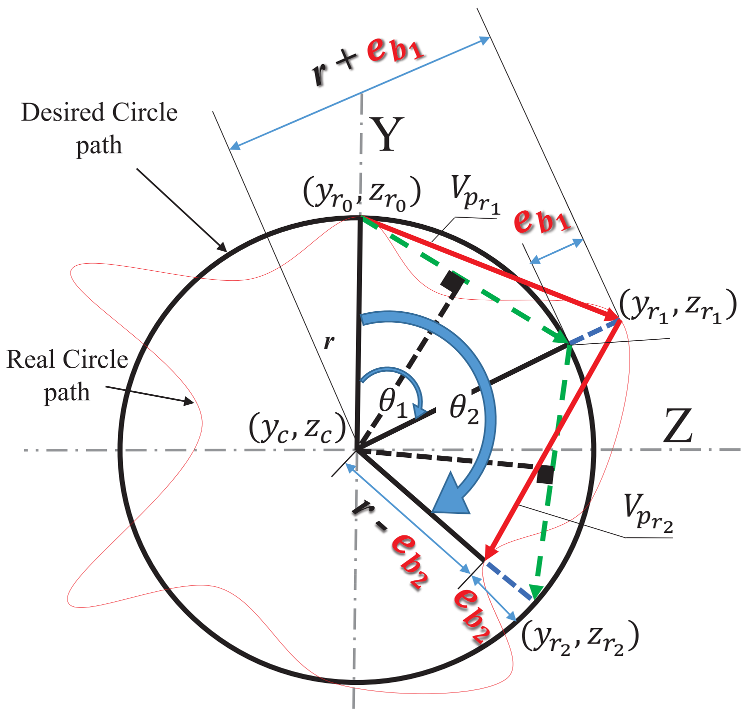
7. Compute the robot's TCP velocity vectors $\mathbf{v}_{v_n}$ from $\hat{\mathbf{v}}_{p_n}$ and the desired robot's TCP velocity υ_r as:

$$\mathbf{v}_{v_n} = \upsilon_r \hat{\mathbf{v}}_{p_n}. \tag{3.27}$$

The velocity vector determines the direction and the magnitude only in the yz-plane which is changing during the motion of each SOM, whereas the x-component of the TCP velocity vector remains zero. The angular velocities of the end-effector will be kept at zero too. Hence, the six arguments specified for the MoveLinVelWRF command will be $(0, \mathbf{v}_{v_n(1)}, \mathbf{v}_{v_n(2)}, 0, 0, 0)$.



a) Calculating the desired robot's TCP position vectors for the desired circular path



b) Calculating the estimated real robot's TCP position vectors for the real circular path

Figure 3.20 The second method of designing the robot's circular trajectory

3.5.1 Real Circular Motion Path of the second method

The previous subsection 3.5 shows how to determine the desired robot circular path supposing that, the robot rotation around a fixed point from the circle center has no error, and there are no variations in the measured circle radius. However, in practice, there are radius error variations measured by the ballbar at each SOM. These variations are not only due to the inherent inaccuracies of the robot but also are due to the fact that the robot cannot respond instantaneously to a change in the TCP velocity vector. We must therefore take into account the error measured by the ballbar sensor, e_{b_n} in polar coordinates, then measure the robot's TCP real coordinates in yz-plane y_{r_n} and z_{r_n} as follows:

$$y_{r_n} = (r + e_{b_n}) \sin\theta_n \tag{3.28}$$

$$z_{r_n} = (r + e_{b_n}) \cos\theta_n + 172, \tag{3.29}$$

Accordingly and because we are only able to measure the rotation error by the ballbar in real-time, we can only estimate the real position and velocity vectors based on this measured error, by replacing the desired coordinates in yz-plane y_{d_n} and z_{d_n} in equations (3.25, 3.26, and3.27) by the real ones y_{r_n} and z_{r_n}. Therefore, the robot's TCP real position and velocity vectors can be estimated as follows:

$$\mathbf{v}_{p_{r_n}} = \begin{bmatrix} y_{r_{n+1}} - y_{r_n} \\ z_{r_{n+1}} - z_{r_n} \end{bmatrix}, \tag{3.30}$$

$$\hat{\mathbf{v}}_{p_{r_n}} = \mathbf{v}_{p_{r_n}} / \|\mathbf{v}_{p_{r_n}}\|, \tag{3.31}$$

$$\mathbf{v}_{v_{r_n}} = \upsilon_r \hat{\mathbf{v}}_{p_{r_n}}. \tag{3.32}$$

The aforementioned real robot's TCP position and velocity vectors have been presented only to emphasize how the real vectors are affected by the error measured by the ballbar compared to the desired ones. Figure 3.20b highlights and justifies the error in the robot's TCP coordinates, radius variation, and position vectors.

3.5.2 Outcomes of the second method of designing the robot's circular trajectory

We note that the sampling time is fixed using this second method unlike the first one, which gives us the confidence that the results related to the second method reflect the real robot error thanks to its stable sampling time. Therefore, we choose this method to move the robot in its circular motion because of the advantage of the fixed sampling time. In this sense, we have the confidence to start the second phase of robot model identification with the right input and output data, which in turn, reflects the real movement of the robot Meca500. At this level, the generated robot model can be used in the simulation phase, to adjust the controller parameters, test, and validate the proposed control system response before going ahead and implement this controller to the real system of the robot.

3.6 SpaceMouse Module

A more affordable alternative and one that forces the user to always handle the end-effector, in the same way, is the use of a 6-axis joystick. Undoubtedly, the most popular six-axis joystick module is one developed by 3Dconnexion. The module has been used in the well-known SpaceMouse, but also in the teach pendants of the industrial robots manufactured by KUKA. Recently, KUKA has introduced the KUKA ready2_pilot, which is an add-on based on 3Dconnexion's SpaceMouse module for hand-guiding traditional KUKA industrial robots. Therefore, this mouse has been selected in this thesis as a human-machine interface (finger joystick SpaceMouse) for manipulating the industrial robot Meca500 in 6-DOF, it could manipulate the robot's spatial position and angular orientation. It is an optimized industrial version of the SpaceMouse from 3Dconnexion company (3dconnexion, 2020), the module core is an optoelectronic sensor which gives intuitive control of 3D complex movements (6-DOF) with one hand. Note that the 3Dconnexion SpaceMouse Compact is proving itself worldwide inside 3D mice and it is originally developed for CAD applications to mainly deliver an intuitive, effortless, and precise 3D navigation that cannot be experienced by using a standard mouse and keyboard (3dconnexion, 2020). For modern engineers, architects, and designers, it is also considered an ideal tool to review 3D designs and explore 3D spaces. This industrial SpaceMouse Module from 3Dconnexion has a dust-proof sealing, a smaller dead zone, and a higher spring tension. The mouse weight is 60 g, while its height is 52 mm, and its natural initial position has the zero value in all directions (3dconnexion, 2020). It is connected to the computer via USB Port which is recognized by the operating system as a standard joystick with 6-axes. The mouse is capable of detecting small inputs of human hand manipulation; therefore, humans are able to directly manipulate the robot's end-effector so as to guide the robot to its target, moreover, they can assign a specific trajectory for the robot to follow; therefore, it is considered an ideal input device for teaching robots which covers both rotation and translation movements due to its 6-axis sensor. Simply, it can replace two conventional joysticks in many applications.

Thanks to the small compact SpaceMouse iconic, small installation depth and pure design, it can be mounted on the Meca500 robot flange by attaching a custom fixture. For the experimental setup, the base of the Meca500 is attached to the same horizontal surface of a rigid table shown in Figure 3.22. A custom fixture weighing 200 g is attached to the robot's flange to support and hold the mouse on the robot's flange itself as shown in Figure 3.23. By simply push, pull, twist or tilt the mouse controller cap in the directions, where axis 1, 2, 3 are translation axes and 4, 5, 6 are rotation axes as the reference coordinate frames shown in Figure 3.21, Meca500 can freely move in its workspace enabling manipulation up to 6-DOF (three transnational, three rotational).

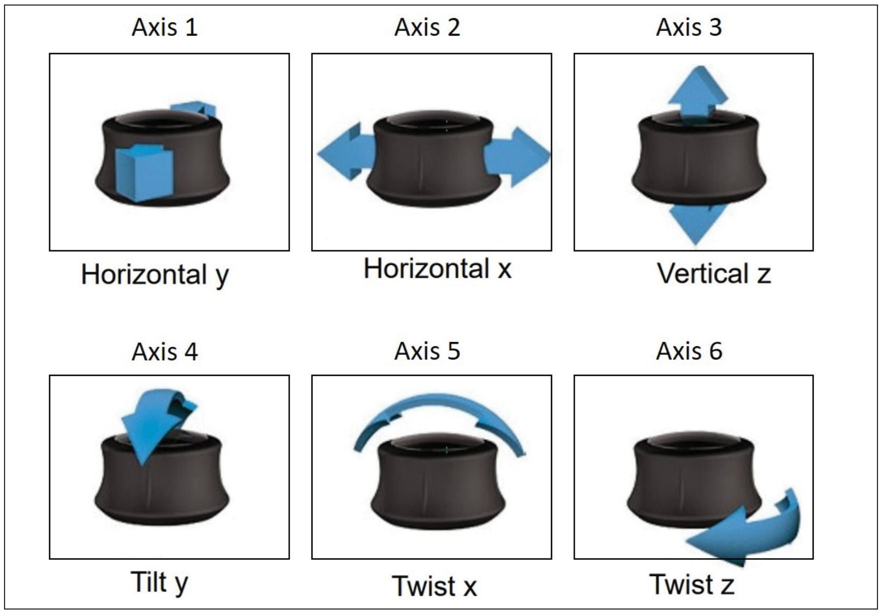


Figure 3.21 The SpaceMouse reference coordinate frame
Taken from 3dconnexion (2020)

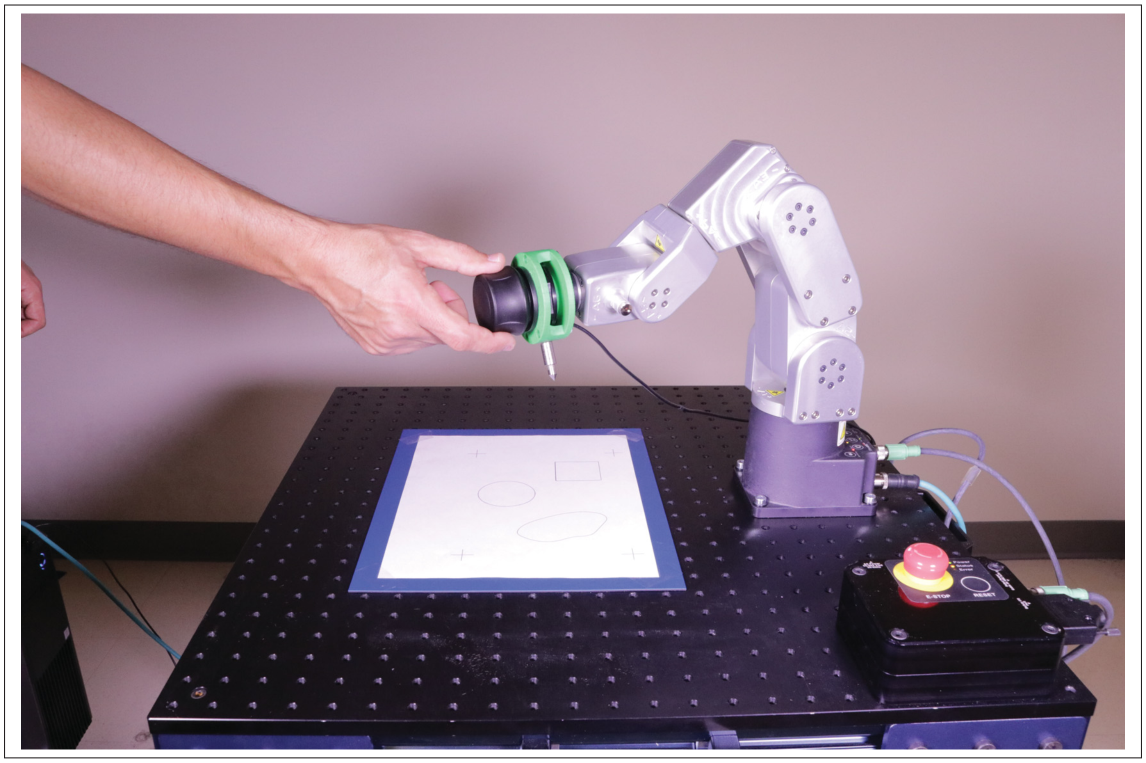
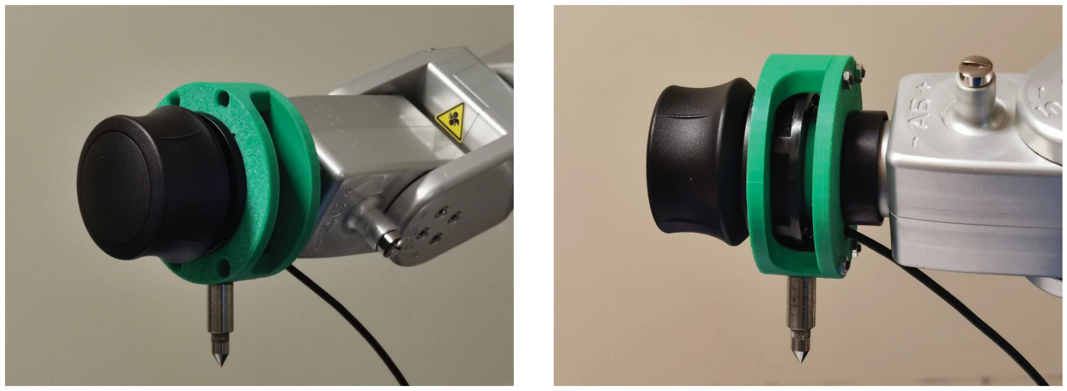


Figure 3.22 Experimental setup featuring the Meca500 six-axis industrial robot and the SpaceMouse Module from 3Dconnexion company



a) A front view of the mouse controller cap from 3Dconnexion company

b) A side view of the mouse controller cap and its fixture

Figure 3.23 Experimental setup featuring the SpaceMouse Module from 3Dconnexion company and its custom fixture on the Meca500 TCP

3.6.1 The SpaceMouse motion commands

As mentioned in subsection 3.1.1, we choose again to operate the Meca500 in velocity mode using the SpaceMouse, since the robot's trajectory can be updated online during the robot motion. This velocity mode involves giving the Meca500 end-effector the velocity vectors obtained from the SpaceMouse in all six degrees of freedom. In order to obtain these velocity vectors using the MATLAB environment, a mouse object is first created using the function vrspacemouse which is capable of interfacing with the SpaceMouse inputs. Then, thanks to the vrspacemouse object's several properties, the SpaceMouse input device cab behavior can be influenced by the user. The mouse properties can be read or modified using dot notation in which the user can read the mouse position axis i (Figure 3.21). Thus, the mouse object outputs are considered the position and orientation in the form of a roll, pitch, and yaw angles. Finally, in order to make the mouse inputs as intuitive as possible, mapping the mouse translations and rotations position outputs to match the robot linear and angular velocity inputs are done by the following steps:

1. Reading the mouse positions of multiple axes output vector which includes the mouse translation and rotation values.

2. Getting the mouse velocity vector by differentiating the mouse position vector achieved by the previous step.
3. Determine the maximum required robot's linear and angular velocities by the user. For the Meca500, the maximum robot's linear and angular velocities that can be selected by the user are ±1000 mm/s and ±300 °/s, respectively (see subsection 3.3.3). The user can also select any maximum desired values below the maximum robot's values depending on the required task.
4. Mapping the mouse velocity vector of the previous step 2 to match the robot's linear and angular velocity vector as follows:

   - The natural position of the mouse cap controller of the position vector of zero which leads to a mouse velocity vector of zero as well is mapped as a robot's linear and angular velocity of zero.
   - The maximum value of the mouse cap translation axes of ±1 in both directions for each axis is mapped as a robot's maximum linear and angular velocity assigned by users in step 3. For instance, if the user selects the maximum robot linear velocity of the Meca500 to be ±600 mm/s according to his task, then the maximum mouse value of ±1 is mapped to the robot's linear velocity of ±600 mm/s. In the same way, if he selects the maximum robot's angular velocity to be ±250 °/s, then the maximum mouse cap rotational axes of ± 1 are mapped to the maximum robot's angular velocity of ±250 °/s selected by the user. Any other mouse value exerted by the user during the mouse movement greater than −1 or less than +1 is mapped to its corresponding robot's velocity compared to the maximum user-selected robot's velocity value as demonstrated in Figure 3.24.

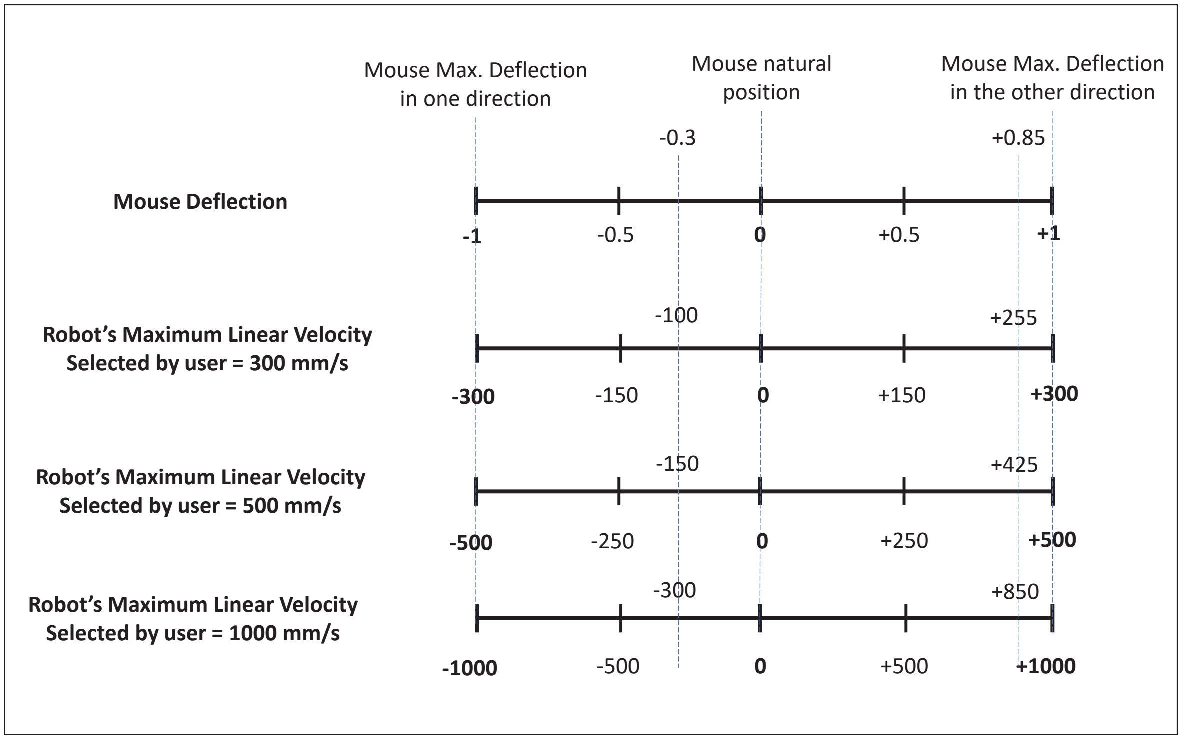

Mouse Max. Deflection in one direction
Mouse natural position
Mouse Max. Deflection in the other direction
-0.3
+0.85
Mouse Deflection
-1
-0.5
0
+0.5
+1
Robot's Maximum Linear Velocity Selected by user = 300 mm/s
-100
+255
-300
-150
0
+150
+300
Robot's Maximum Linear Velocity Selected by user = 500 mm/s
-150
+425
-500
-250
0
+250
+500
Robot's Maximum Linear Velocity Selected by user = 1000 mm/s
-300
+850
-1000
-500
0
+500
+1000

Figure 3.24 Demonstration of mapping the mouse deflections to robot's linear velocity values

Following the previous steps, a script is developed to calculate the mouse input to its corresponding robot's velocity movement. Through this we are able to collect the mouse data in real-time as the speed changes, then to feed the related mapped linear and angular speeds to the robot. The influence of the robot motion in position and velocity modes settings is illustrated in Figure 3.3.

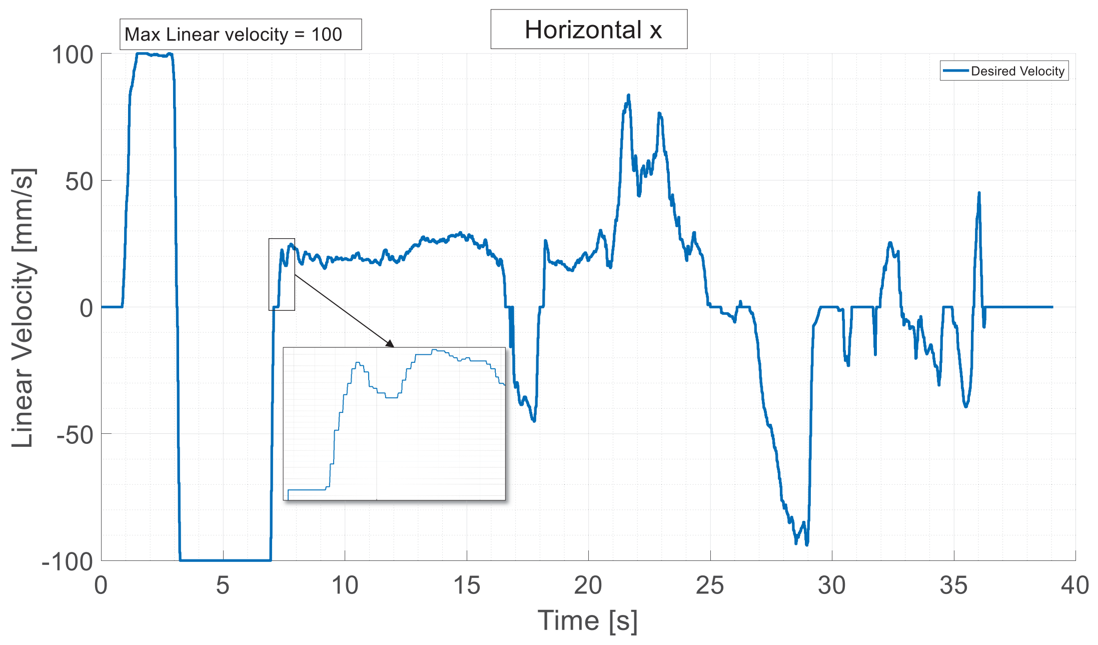
3.6.2 The SpaceMouse control challenges

To control the SpaceMouse in a compensatory desired tracking task when a 6-DOF industrial robot is being manipulated is difficult especially at the robot's high speeds, regardless of the user experience. The challenge increases because the mouse is mounted on the robot's TCP end-effector forming an interactive collaboration between both humans and mouse, which means that any small or larger mouse cap deflections will cause the robot's TCP to move while the user is still controlling the mouse, this will affect the required position by the user, especially at the robot's high speeds. Some researchers addressed the mouse challenges while controlling the robot when the mouse is not mounted on the robot's TCP itself, in other words, the mouse is apart from the robot. For instance, the author in (Underwood & Gallimore, 2010) proves that using one hand to control the robot is performed with few errors when the mouse is apart from the robot's end-effector (i.e the mouse is not mounted on the end-effector itself), and despite that, the mouse motion was difficult to be controlled. This error was related to the cross-coupling between translation and rotation when all six DOF were controlled simultaneously, which makes it very hard to be controlled. In our case, when the mouse is mounted on the robot's TCP itself and not apart, it is even worse. Besides the aforementioned cross-coupling issue, the mouse movement is very sensitive to the tiny exerted forces by the user, which in turn is directly affecting the robot movement. To delve into details, once there is any tiny displacement between the mouse and its center, the robot moves according to this displacement very fast, which makes it leads the mouse, at this moment, the displacement of the mouse from its center becomes in the opposite direction, without any intention or control from the user, which in turn makes the robot responds again to this new displacement pushing it to immediately respond and moves to the opposite direction. This is exactly the point that makes the robot oscillate around the required position, consequently the mouse may leave the user's hand itself and the robot's motion becomes more aggressive causing rush movement, especially at the robot's high speeds. We believe and agree that each mouse has its own stiffness which in turn affects the minimum robot's speed that can be selected before reaching this oscillation behavior.

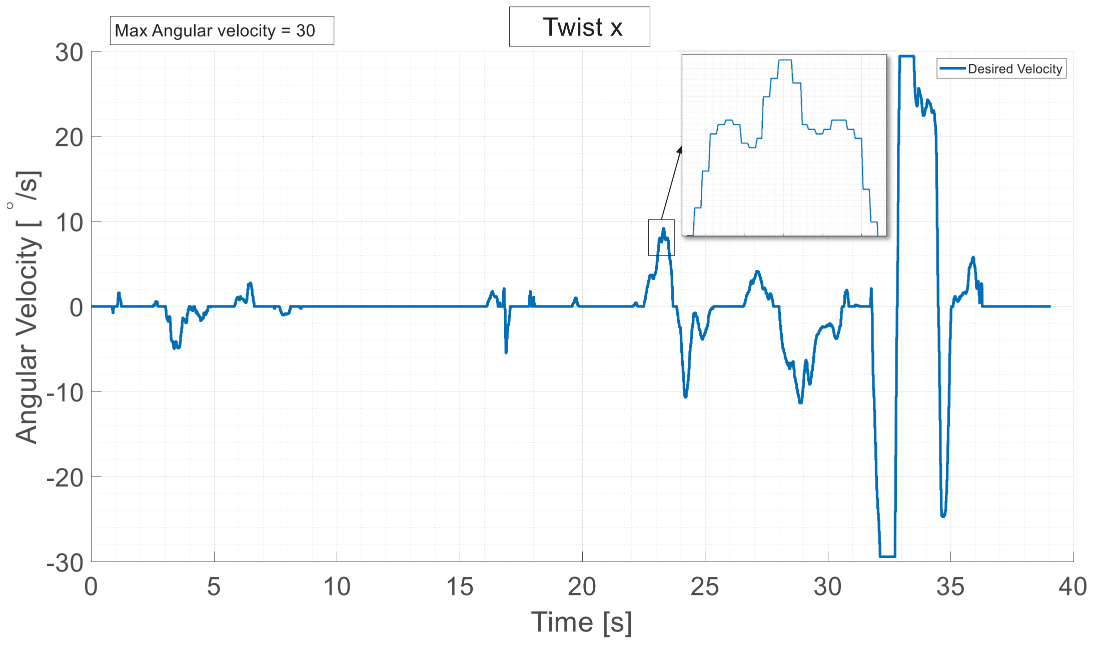
In our case, the mouse from 3Dconnexion company has the mouse proprieties according to the data-sheet in (3dconnexion, 2020). Upon doing many experiments in which we manipulate the Meca500 by the SpaceMouse, we can say that, when the Meca500 linear and angular velocities are equal to or less than 100 mm/s and 30 °/s, respectively, the robot's motion stays smooth. However, once the robot's speeds exceed the aforementioned values, the response of the Meca500 towards the mouse manipulation (translation and rotation) starts to oscillate and the robot's movement starts to be more aggressive and sometimes the mouse suddenly leaves the user's hand. To address this issue, Figure 3.25a illustrates the mouse-robot mutual behavior when the mouse is deflecting horizontally in axis-2 while the robot is responding by translating in the x-direction at the robot's maximum linear velocity of 100 mm/s. In addition, Figure 3.25b shows this behavior when the mouse is being twisted in axis-6 (see Figure 3.21) while the robot is responding by rotating in the x-direction at the robot's maximum angular velocity of 30 °/s. As can be seen, there is no oscillation in the graph and the motion is very smooth, except that there are some perturbations due to the robot's joint motors (zoomed in Figure. 3.25), which will be solved later after adding our proposed external controller. On the contrary, this robot's behavior is changed when we attempt to increase the robot's maximum linear and angular velocities than the previous values of 100 mm/s and 30 °/s, respectively. For instance, if the robot's linear and angular speeds are 150 mm/s and 44 °/s, respectively, while the mouse is moving and twisted in the same previous axes and directions, a large-scale oscillation starts to appear as can be clearly seen in Figures 3.26a and 3.26b. We also strive to raise the robot's maximum linear velocity to be 250, 350, 450, and 600 mm/s, and the robot's maximum angular velocity to be 73, 103, 132, and 176 °/s, then we plot the results in Figures 3.27 and 3.28. We discovered that the oscillation increases on a large-scale proportionally with the robot linear and angular velocities as can be clearly seen in these figures. This error is understandable because it is related to the cross-coupling between the mouse and the robot manipulation especially that the mouse is mounted on the robot's TCP while the robot is rotating and also translating at the same time in addition to continuously changing the robot's direction and speeds. The fact that they are attempting input into translation or rotation while also trying to decrease and/or increase the motion or even change the direction in both translation and rotation, which is affecting their mutual translation and rotation performance represented by the aforementioned figures; thus, both the related translation and rotation resulted in worse performance compared to those of the robot's slower motion in Figure 3.25. Therefore, we need to add an external controller as laid out in Figure 5.2 to counter the oscillations and defeat those challenges.

Note 1: All the aforementioned plots are not purely describing the desired mouse deflection, this may be misleading, they are showing the mutual response between the desired mouse deflection and the real robot's deflection from the mouse center including the cross-coupling effect. Therefore, we can say that when fewer oscillations appear in these plots, this means that the robot moves in a smooth manner. In other words, one should note that these plots yield the measured mutual position and not only the desired one, i.e., the error between the real and the desired motion.

Note 2: To manipulate the robot at a very low speed trying to avoid the aforementioned issues is not comfortable at the user end. The optimum is to comply with the user demand by moving fast when teaching the intermediate poses followed by slower precise motion when teaching a path, in addition, this behavior should also be kept when the user attempts to move from one direction to its opposite, or when he attempts to do small/large scale of mouse deflection.

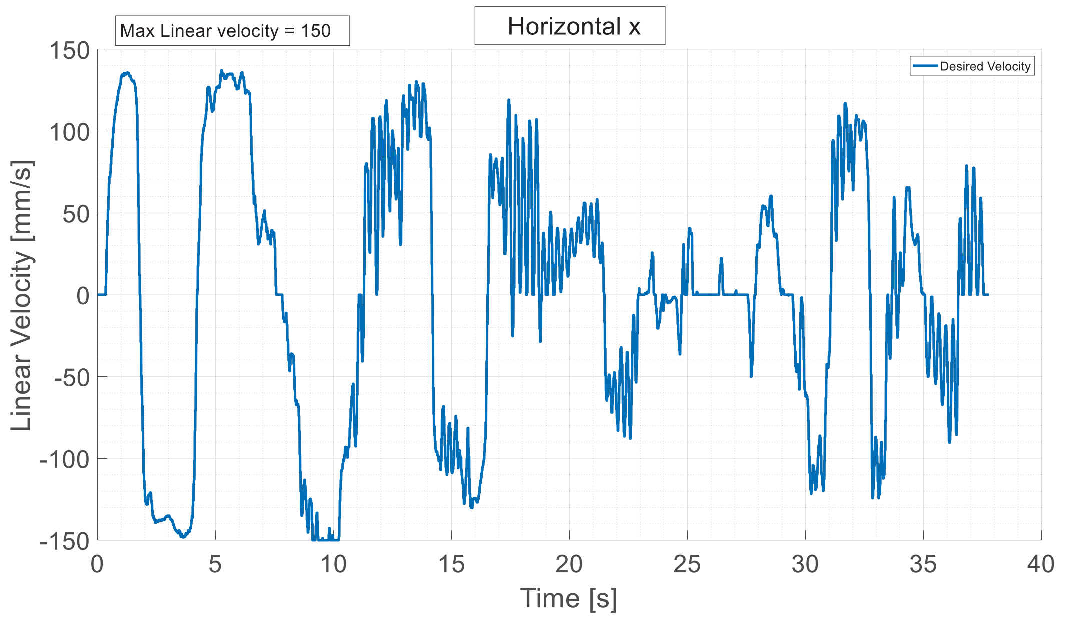


a) Horizontal motion in x direction axis 2

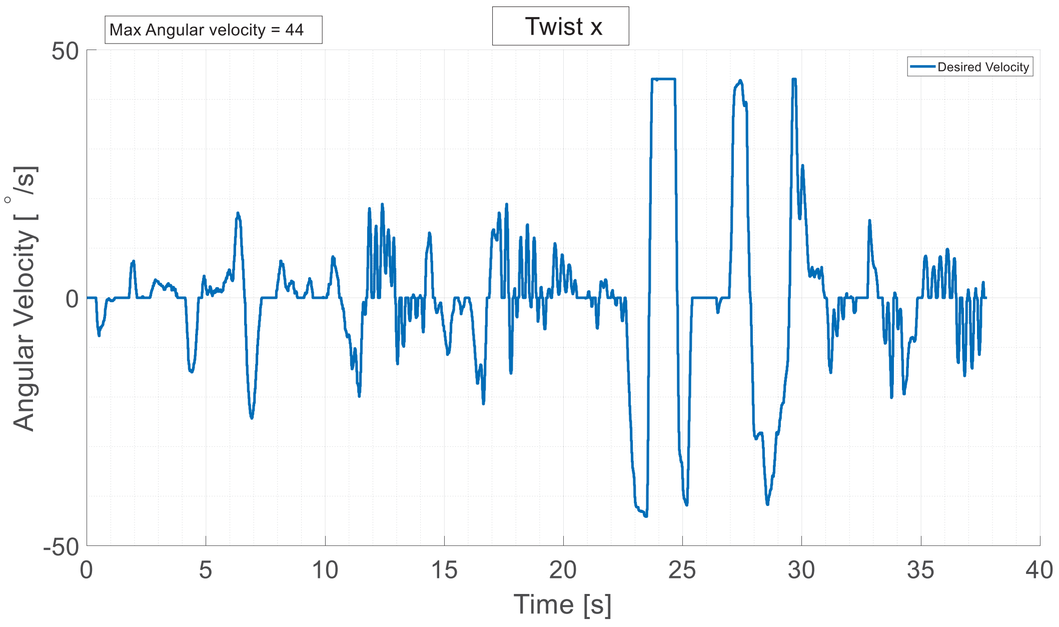


b) Twisting motion in x direction axis 5

Figure 3.25 Robot's desired motion and twisting in x directions at maximum linear and angular speeds of 100 mm/s and 30 °/s, respectively

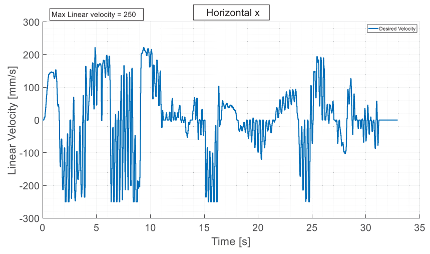


a) Horizontal motion in x direction axis 2

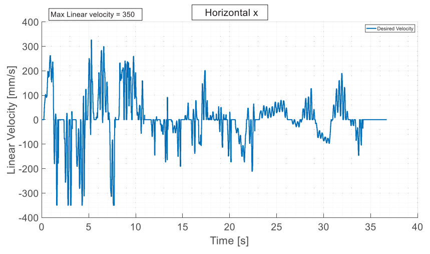


b) Twisting motion in x direction axis 5

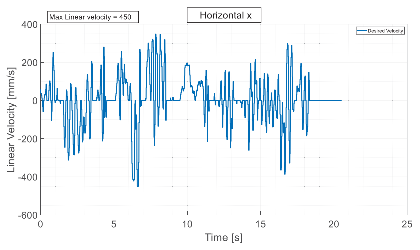
Figure 3.26 Robot's desired motion and twisting in x directions at maximum linear and angular speeds of 150 mm/s and 44 °/s, respectively



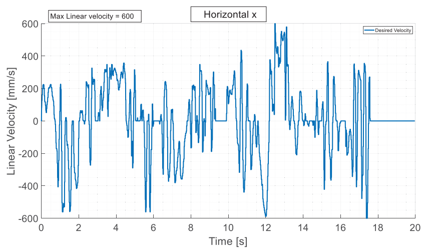
a) Max. robot's linear velocity of 250 mm/s



b) Max. robot's linear velocity of 350 mm/s

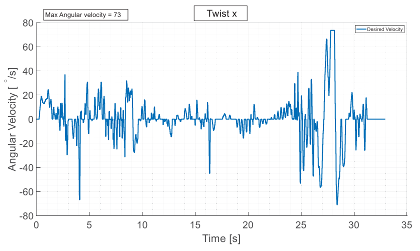


c) Max. robot's linear velocity of 450 mm/s

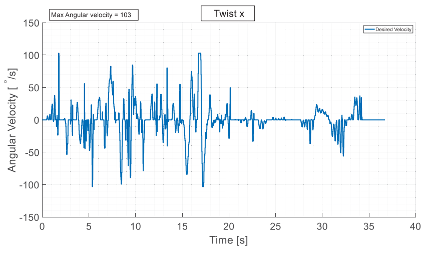


d) Max. robot's linear velocity of 600 mm/s

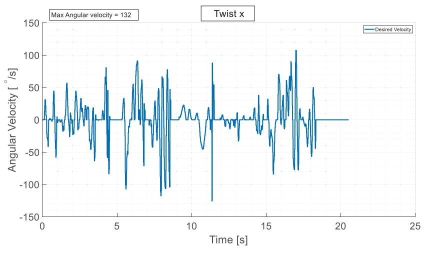
Figure 3.27 Different maximum robot's linear velocities while moving in horizontal x-direction axis 2



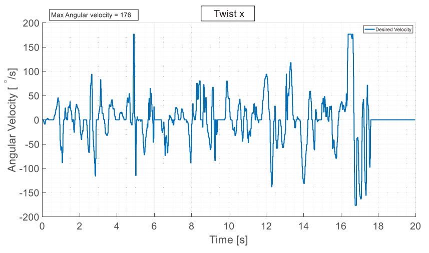
a) Max. robot's angular velocity of 73 °/s



b) Max. robot's angular velocity of 103 °/s



c) Max. robot's angular velocity of 132 °/s



d) Max. robot's angular velocity of 176 °/s

Figure 3.28 Different maximum robot's angular velocities while twisting in x-direction axis 5

CHAPTER 4

ADRC PROPOSED CONTROL DESIGN FOR ENHANCING THE ROBOT'S PATH ACCURACY

4.1 Introduction

Many advanced and high-performance control techniques exist in the literature and are presented in Section 2.4. For industrial robots, however, there is a theory-practice gap: control methods developed in academia are seldom used in this industry where the choice is usually to sacrifice the robot performance for the sake of design simplicity (Madoński, 2016). Many robot applications in the literature use advanced control algorithms (adaptive, non-linear, etc.) as well as sophisticated filtering techniques with very good simulation results. At the implementation stage, however, the required high level of accuracy (micrometer errors) is not necessarily achieved because of the complexity of the controller, the transmission rate, and hardware constraints.

Our first control objective is to design a simple practical controller based on the advanced control technique of ADRC, to ensure that the robot's trajectory tracking is highly accurate by industry standards and fast while rejecting the disturbances affecting the robot using an external sensor. Thanks to ADRC nature which inherits the properties of the proportional-integral-derivative (PID) controller while improving its characteristics, it gives us the required practical simplicity. It also embraces the power of nonlinear feedback and puts it in full use by using nonlinear gains in a PID structure. The total disturbance (internal or external) treated as a function of time and that's exactly how it can be estimated and canceled in the feedback loop (Han, 1999). ADRC essential philosophy is to view the total uncertainty as a new extended state of the system, and then estimate it with possible other non-accessible states of the system by an extended state observer (ESO) and finally, the control action compensates this uncertainty in real-time (Han, 1995; Guo & Zhao, 2015). In addition and thanks to this ADRC disturbance estimation in real-time, both parametric and non-parametric robot disturbance shown in Figure 2.6, can be effectively compensated by the feedback loop. To conclude, ADRC has its concept uniqueness, the advantages of being non-model-based, simple in implementation with a disturbance rejection by good estimation, improved transient response, and intuitive tuning capability. Thus, the ADRC meets our first objective.

In order to assess the performance of our ADRC controller, different experiments were executed with four TCP velocities (25, 50, 75, and 100 mm/s) in three cases as demonstrated in Figures 3.6, and 3.7, and starting with an initial ballbar error of 0.16 mm, which comes from teaching the robot the center of the circle, then moving it upwards a distance of 100 mm. These experiments are accomplished with the manufacturer robot controller only and after adding the external loop of the proposed control algorithm ADRC to the manufacturer robot controller, as follows:

1. Testing the robot's normal operating performance with the robot's rated payload of 0.5 kg.
2. Testing the robot's performance with the robot's full payload of 1 kg, to check the controller robustness.
3. Testing the performance under a unit step disturbance signal (sudden extra load of 0.5 kg) during the robot's circular motion at an angular position between 190° and 230° for the rest of its motion.

Moreover, to demonstrate the advantages and superiority of the proposed ADRC robustness, a non-model based SMC algorithm (Gurumurthy & Das, 2020) is also experimentally implemented as an external loop controller in the above third case instead of the ADRC, then the results are compared, contrasted, and analyzed.

Remark. *Recently, SMC has attracted great interest in the field of robotics due to its implementation simplicity and robustness against uncertainties. For a fair comparison, we implemented a simple model-free approach while preserving robustness by using time-delay estimation (TDE) to estimate the disturbance (Ebrahimi, 2014).*

4.2 ADRC Output-based Form (OBF)

The ADRC technique on which our proposed controller is based is mainly composed of three parts: tracking differentiator (TD), extended state observer (ESO), and nonlinear state error feedback (NLSEF) (Xia & Fu, 2013). The block diagram of the proposed control scheme is shown in Figure 4.1. These three parts are relatively independent of each other and at the same time they are very different, we will explain in detail how each part works individually then the system overall proprieties will be concluded.

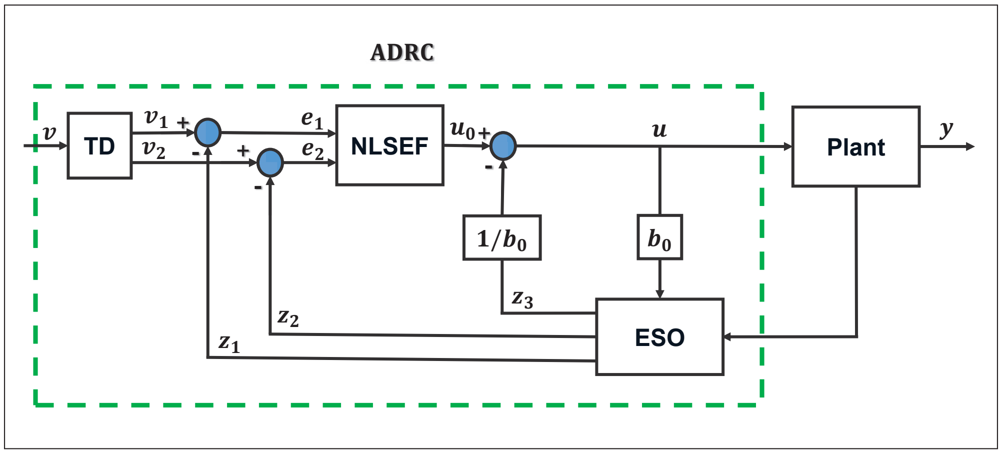


Figure 4.1 The structure of ADRC algorithm

4.2.1 Tracking Differentiator (TD)

The tracking differentiator plays an essential role of the arrangement of an appropriate transient process of a given input reference signal v by setting value v_1 and its differential v_2, then provides noise-free and robust differential signals from that input signal to the next ADRC part (Han, 2009). This part solves the problems related to the PID derivative term which creates large discrepancies when there are noises measurement in the input signal. Han's work on non-linear differentiators (Han, 1995) inspires plenty of researchers (Han, 1999; Gao *et al.*, 2001b; Gao, 2006b) to start using the principle of TD, taking the example of TD second order which can be described by:

$$\begin{cases} \dot{v_1} &= v_2 \\ \dot{v_2} &= fhan(v_1 - v(t), v_2, r, h_0) \end{cases} \tag{4.1}$$

where $v(t)$ denotes the input reference signal, r is the speed factor that makes the transient response faster while r increases, h_0 is the filter factor and $fhan(v_1 - v(t), v_2, r, h_0)$ is fastest control comprehensive function defined as follows:

$$\begin{cases} d & = rh_0^2 \\ a_0 & = h_0 v_2 \\ y & = (v_1 - v(t)) + a_0 \\ a_1 & = \sqrt{d(d + 8 \mid y \mid)} \\ a_2 & = a_0 + sign(y)(a_1 - d)/2 \\ s_y & = (sign(y + d) - sign(y - d))/2 \\ a & = (a_0 + y - a_2)s_y + a_2 \\ s_a & = (sign(a + d) - sign(a - d))/2 \\ fhan & = -r(\frac{a}{d} - sign(a))s_a - rsign(a) \end{cases} \tag{4.2}$$

In practice, r drives the convergence speed of the reference signal derivative estimator. TD has the advantage of being noise tolerant regardless of its application conditions. Some examples can be seen in the book of (Guo & Zhao, 2016) regarding the linear and non-linear tracking differentiators that may slightly differ depending on the type of system application but at the same time remain globally similar. Retaining the TD properties in (Gao, Hu & Jiang, 2001a), it is filtering the noise with a much lower phase shift than the conventional linear filters and has a better signal-to-noise ratio. The TD, therefore, solves the problem raised by Han as to the PID derivative term. That is why the PID has often used as a PI in practice because the signal noise is deriving producing large errors in relation to its true value which has been resolved by the TD presented above. At this end, we have to mention that the TD is not an essential component of the ADRC, indeed several examples of applications can be found in (Gao, 2006b; Li, Li & Zheng, 2016; Ma, Xia, Li & Chang, 2016) where the ADRC does not include TDs. The controller still can be designed without it, the same has been done by PI instead of PID. It is a matter of compromising between controller quality and complexity.

Note 1: In our first objective case (Accuracy enhancement), we are dealing directly with the radial error which is measured by the ballbar as illustrated in Figure 4.3. Hence, the controller reference signal is not the desired trajectory but the desired value of the error which is zero. Consequently, the input reference signal v in Figure 4.1 as well as the signal v_1 and its derivative v_2 are zero. Therefore, we will not be using the TD part in our ADRC design in the first objective which simplifies the control structure and does not affect the controller quality.

Note 2: In our second objective case (Hand-guiding), we will fully use the tracking differentiator to remove the signal noise and to exert an intended delay to part of the signal, we will also automatically control the filtering factor h_0 which is responsible for determining the amount of filtering and the delay.

4.2.2 Extended State Observer (ESO)

In general, knowing all the states of the entire system is rare, mostly only a part of it is known through the output signal. In other words, the rest of the system states can usually be known from the data provided by the output signal by building an estimator. Luenberger is one of the most well-known estimators which is simple and very effective, and by the estimator gain, the designer can acquire the convergence speed of the actual system state. Despite this is working very well in linear systems, large gains amplify the estimator initial error causing large peaks of errors. Using nonlinear gains is one of the important solutions to this problem, for instance, the estimator sliding mode in (Edwards & Spurgeon, 1998). New estimator (ESO) however, proposed by Han (Han, 1995) for the first time taking into account the internal and external disturbance. The ESO is the core of the ADRC controller, it estimates the state value z_1 of the output signal y, the state value z_2 of the differential of the output signal, which is dependent on the control input u and, finally, the extended state z_3 which represents the system external disturbances and uncertainties (Han, 2009). ESO for a system of a second order can be expressed as:

$$\begin{cases} e &= z_1 - y \\ \dot{z_1} &= z_2 - \beta_1 e \\ \dot{z_2} &= z_3 - \beta_2 f_{al}(e, \alpha_1, h) + bu \\ \dot{z_3} &= -\beta_3 f_{al}(e, \alpha_2, h) \end{cases}, \tag{4.3}$$

where z_1, z_2 and z_3 are the observer outputs, e is the error, β_1, β_2 and β_3 are the observer gains, which are interrelated and must be carefully tuned, α_1 and α_2 are the nonlinear coefficients, and h is the sampling time. The nonlinear function f_{al} is given by (Han, 2009) :

$$f_{al}(e, \alpha, \delta) = \begin{cases} \frac{e}{\delta^{1-\alpha}}, & \mid e \mid \leq \delta \\ sgn(e) \mid e \mid^{\alpha}, & \mid e \mid > \delta \end{cases} \tag{4.4}$$

It has been shown that the estimator convergence speed and robustness depend more on the choice of the fal functions. The researcher (Dabin, 2018) noted in articles dealing with ESO, two types of functions: either linear functions which are easy to adjust or non-linear functions such as fal function in the equation. 4.3. These functions have been found empirically and must nevertheless be specified because of the complexity of its underlying mathematics.

For the nonlinear function of fal in equation 4.4, it is mainly dependent on the linear space parameter δ (precision coefficient), which is tuned to increase proportionally with the linear velocity as shown in Table 4.1. This is because the characteristic of the function fal is that the gain decreases when the deviation increases. It can be seen in Figure 4.2 that the parameter δ is controlling the linearity of the curve and determining to what extent the gain in our case can be linearly related to the error itself. In other words, the gain is linear when the error falls into a small region around zero with bounded δ, as shown in the figure, this is exactly one of the major benefits of Han's fal function. Therefore, and because the robot's error increases with the higher linear velocities in addition to the robot's step itself, we increased the value of this parameter with the higher velocities.

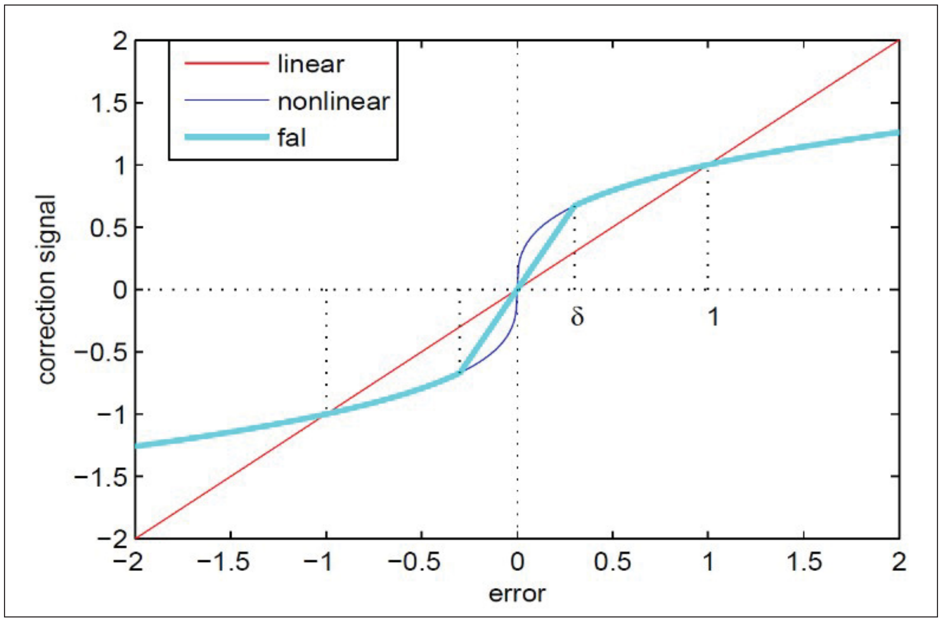


Figure 4.2 Comparison of linear and non-linear gains
Taken from Dabin (2018)

As can be seen, the system dynamics do not to be considered or explained, this dynamic in the quantity is only to be estimated, and by doing this, any modeling errors that could be disturbed by the system dynamics can be compensated. The additional state variable z_3 represents the estimation of the total system uncertainties including the system unknown dynamics plus disturbance. Since z_3 estimates these disturbances, they can be canceled and rejected in the feedback loop of the control law in equation (4.7).

The output signal y is equal to the ballbar error e_b. Thus, the error e in equations (4.3) and (4.4) is calculated as:

$$e = z_1 - e_b \tag{4.5}$$

4.2.3 Nonlinear State Error Feedback (NLSEF)

It is considered as a nonlinear control law, because it generates the final control value u based on the state errors of the system between TD - ESO outputs $e_1,\ e_2$ and the estimated extended state value z_3 of the disturbance. Generally, the errors will be large with lower gains and small with higher gains (Han, 2009). The NLSEF is given by:

$$\begin{cases} e_1 &= v_1 - z_1 \\ e_2 &= v_2 - z_2 \\ u_0 &= k_1 f_{al}(e_1, \alpha_1, \delta) + k_2 f_{al}(e_2, \alpha_2, \delta) \end{cases} \tag{4.6}$$

where e_1, e_2 are the output errors, u_0 is a nonlinear combination of error signal and its differential and k_1, k_2 are the proportional and differential control coefficient gains. The controller is designed as:

$$u = u_0 - \frac{z_3}{b_0}, \tag{4.7}$$

where b_0 is the system gain and is not equal to zero.

In our first objective case (Accuracy enhancement), we have the following remarks:

1. v_1 and v_2 in equation (4.6) are equal to zero, thus, the output errors are equal to:

$$\begin{cases} e_1 &= -z_1 \\ e_2 &= -z_2 \end{cases} \tag{4.8}$$

2. The control action u_n at each SOM is added to the robot end-effector desired coordinates by the same error projection of equations (4.11 and 4.12) as:

$$y_{c_n} = y_{d_n} + u_n \sin\theta_n \tag{4.9}$$

$$z_{c_n} = z_{d_n} + u_n \cos\theta_n \tag{4.10}$$

   where y_{c_n}, z_{c_n} are the corrected robot coordinates each SOM, n is the current SOM index.

3. As mentioned in Chapter 1, the robot under the dissertation comes with an embedded and closed robot controller with the desired trajectory to be tracked as input. Thus, the proposed ADRC controller will be an external loop that generates the enhanced desired

trajectory and feeds it to the controlled robot. The information needed by the controller is not derived from previous knowledge of robot mathematical model, but rather from the error feedback information from the end-effector observed by the external ballbar sensor. The control action u will be added to the desired trajectory before forwarding it to the robot. The block diagram in Figure 4.3 shows the proposed controller structure incorporating the external sensor to get the real-time data of end-effector real position. The tracking error e_{b_n} measured by the ballbar is defined as the difference between the robot reference desired coordinates (y_{d_n}, z_{d_n}) and real coordinates (y_{r_n}, z_{r_n}). This error is the polar radius error which combines the errors in both y and z coordinates. Therefore, the projection of this error on the y-plane e_y and the z-plane e_z at each SOM can be calculated as:

$$e_{y_n} = y_{r_n} - y_{d_n} = e_{b_n} \sin\theta_n \tag{4.11}$$

$$e_{z_n} = z_{r_n} - z_{d_n} = e_{b_n} \cos\theta_n \tag{4.12}$$

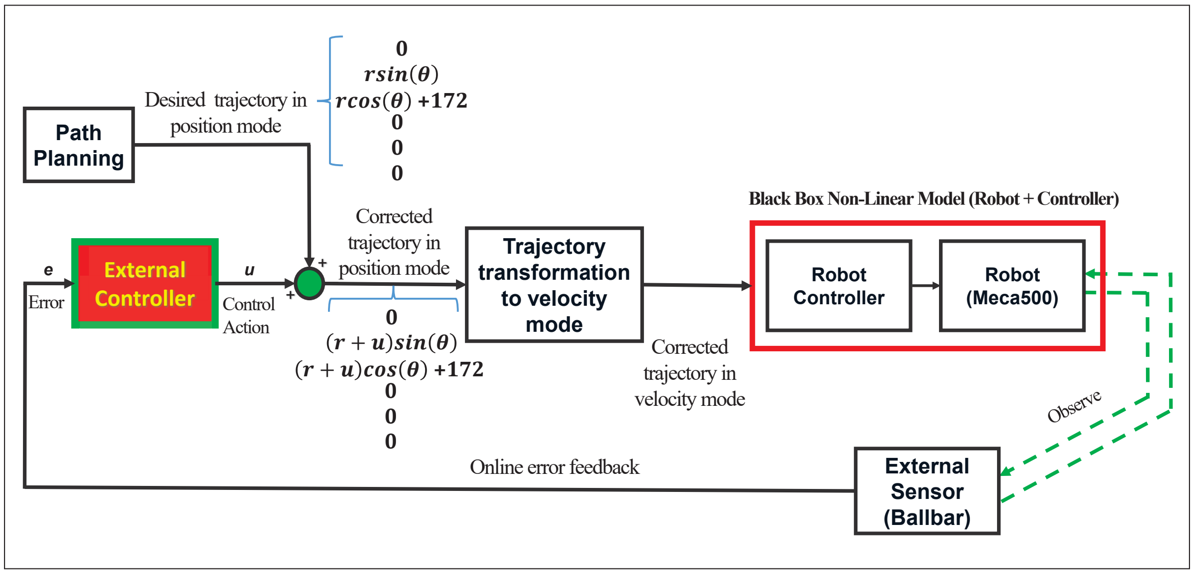

Path Planning
Desired trajectory in position mode
0
rsin(θ)
rcos(θ) +172
0
0
0
Black Box Non-Linear Model (Robot + Controller)
e
Error
External Controller
u
Control Action
+
+
Corrected trajectory in position mode
0
(r + u)sin(θ)
(r + u)cos(θ) +172
0
0
0
Trajectory transformation to velocity mode
Corrected trajectory in velocity mode
Robot Controller
Robot (Meca500)
Observe
Online error feedback
External Sensor (Ballbar)

Figure 4.3 The structure of ADRC algorithm

4.2.4 Summary of the ADRC in OBF

Referring to Figure 4.1, the three ADRC parts explained above can be better understood. The ADRC controller's main function is to trace the design of the command. Firstly, the TD takes either the reference signal or the output signal (based on TD role, in the scheme TD is derived by the reference signal), then TD generates its derivative. By accessing the state variables given by the output and also by computing its derivatives, we will have the system feedback information. The differences between the estimated state variables and reference signals (and its derivatives) will then be forwarded to the NLSEF. The command produced by the NLSEF will therefore represent the system's desired dynamics which will be unfortunately disrupted by the internal and external disturbance (sources of errors classified in Figure 2.6). The ESO, therefore, estimates in real-time the difference between these polluted dynamics and the real dynamics. The NLSEF generates the final control value realizing the desired dynamics in offsetting the uncertainties of our system and the external disturbances.

4.3 ADRC Error-Based Form (EBF)

It is always desirable in practice to have a more industry-friendly controller design, which enables straightforward implementation, simplicity, robustness, and explicitly expressing the feedback error-to-control signal channel. It is also preferable to be applied to a wide range of industry platforms. We implicitly know that the majority of industrial controllers are led by PID in error based form. Therefore, the main inspiration is to make the previous ADRC output based form even simpler in the design and tuning process, which makes it widely applicable to the industry. In this context, (Madonski *et al.*, 2019) reformulated ADRC OBF into a more industry-familiar PID-like error based form. By doing this, the author provided different industrial platforms with online disturbance rejection capabilities. ADRC can be expressed in error-based form instead of a typical output-based form which is independently developed in (Zhang, 2017; Michałek, 2016). They reconstructed the typical ADRC into an industrial 1-DOF form making it easier to be implemented in real applications or to swiftly replace the existing controllers.

The output-based form of the ADRC controller presented in the previous section is the standard and most popular one in the literature. In our case, however, and since the measured output is the error between the desired and real trajectory, we will be using a different version of the ADRC, referred to in the literature as error-based ADRC (Madonski *et al.*, 2019; Huang & Yin, 2019). The proposed error-based solution stabilizes the ballbar output error signal to zero using the control action u as shown in Figure 4.3. Thus, the observer equation (4.3) can be written in discrete error-based form as:

$$\begin{cases} \epsilon = z_{1_n} - e_{b_n} \\ z_{1_{n+1}} = z_{1_n} + \tau(z_{2_n} - \beta_1 \epsilon) \\ z_{2_{n+1}} = z_{2_n} + \tau(z_{3_n} - \beta_2 \epsilon + b_0 u_n) \\ z_{3_{n+1}} = z_{3_n} - \tau \beta_3 \epsilon \end{cases} \tag{4.13}$$

where ϵ is the new controller error-based output, e_{b_n} is the ballbar feedback error and τ is the sampling time. Finally, u can be directly considered as the disturbance rejection controller action as:

$$u_{n+1} = \frac{z_{3_{n+1}} + k_1 z_{1_{n+1}} + k_2 z_{2_{n+1}}}{b_0} \tag{4.14}$$

It is worthwhile mentioning that one of the best advantages of using this error-based form in our case study is that by using the ballbar sensor feedback, the controller performs well regardless of the lack of knowledge of the robot system order and mathematical model. It is considered as a plug-and-play controller which can be used for any system using feedback from any external sensor, once sufficient information about the error is available. It also supports industrial applications where the reference time derivatives trajectory tracking is not available because they are treated as part of the total disturbance (Madonski *et al.*, 2019).

4.3.1 EBF stability analysis

ADRC stability has been investigated in the presence of uncertainty parameters in different studies such as (Shao & Gao, 2017) and (Wu & Guo, 2018). For the generalized error-based form, a complete stability analysis was performed by (Madonski *et al.*, 2019) and a theorem which extends the stability results of (Shao & Gao, 2017) to the proposed EBF was provided.

4.3.2 Robustness test

An extra weight (0.5 kg) has been added on the top of the robot's end-effector at the beginning of its circular motion with different robot's linear velocities in order to test the system robustness by pushing the robot to its maximum limit (full payload of 1 kg). This test is showing the ability of the controller to deal with various payloads. Note that the original custom fixture is weighing 0.5 kg and the extra weight has been added on top of that as shown in Figure 3.7a.

4.3.3 Disturbance rejection analysis

In this dissertation, different experiments were executed with different robot linear velocities and a sudden weight (0.5 kg) was dropped onto the robot's end-effector. The drop was done from approximately 50 mm during the robot circular motion at a rotation angle between 190° and 230° as shown in Figure 3.7b, in order to investigate the system response under unit step disturbance.

4.3.4 Tuning process and parameters adjustment

One of the most challenging parts of the ADRC control technique is the tuning process, especially in the case of high order systems (Chu, Wu & Sepehri, 2019). Since we are dealing with a black box system (robot and controller) with internal and external disturbances, the tuning process using model-based classical methodology is not available. For instance, the use of the popular pole-placement approach for tuning the observer gains (Gao, 2006a) will be time-consuming and with no guarantee because the system knowledge is not available. ESO disturbance estimation quality is directly affected by its parameter tuning, hence the choice of optimal parameters is crucial for improving the controller performance.

An empirical approach was used, where the observer and controller gains tuning process is performed manually based on trial-and-error attempts and on the hunch and experience of the user. Note that in the ADRC technique, the observer gains can be independently tuned from the controller gains. To simplify the tuning process and to reduce the number of trial-and-error attempts, a parameterization methodology has been introduced in (Guanyu *et al.*, 2019; Gao, 2006a), in which the user can choose the observer bandwidth ω_o and the controller bandwidth ω_c considered as design parameters. The parameterization is proposed as (Cui, Tan, Li, Wang & Wang, 2020):

$$\beta_1 = 3\omega_o;\ \beta_2 = 3\omega_o^2;\ \beta_3 = \omega_o^3 \tag{4.15}$$

where ω_o is the observer bandwidth.

$$k_1 = \omega_c^2;\ k_2 = 2\zeta\omega_c^2 \tag{4.16}$$

where ω_c, ζ are the controller bandwidth and the damping ratio respectively. In (Gao, 2006a), the author proposed to set the observer bandwidth to be three to five times the controller bandwidth, i.e., $\omega_0 \approx 3\sim5\omega_c$.
Note that the observer bandwidth is limited by the sampling time for discrete-time systems, therefore it is preferable to start with tuning the ESO, since it affects directly the inner loop of the whole feedback control system. Han's fal function plays an important role to make the gain linear while the error falls into the region around zero. The parameter δ in equation (4.4) determines to what extent the gain can be linearly related to the error itself which is proportionally increasing with the robot's TCP velocities. Once the estimation process is achieved without any undesired noises, then the tuning of the control loop can be started. The tuning of the controller parameters k_1, k_2 is similar to the tuning of proportional and derivative gains of PD controller. The same concept of the trade-off between the response speed and the overshoot of system response is applied. It is preferable to start to increase first the proportional term, and then to compensate for the overshoot by increasing the derivative gain. The integration part was intentionally omitted because it is already included in the ESO. The selected setting parameters are listed in Table 4.1.

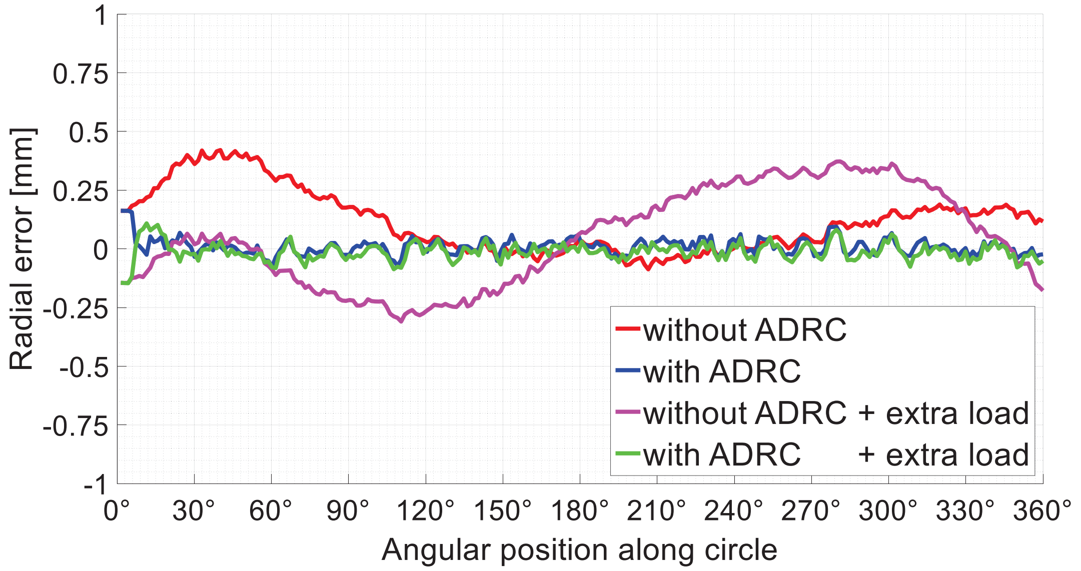
Table 4.1 ADRC parameters

ADRC Parameters		b_0	δ	α_1	α_2	β_1	β_2	β_3	k_1	k_2	h_0
Linear Velocity (mm/s)	**25**	0.6	0.4	0.25	1	1.9	1.15	02	0.02	0.31	0.1
	50	0.5	0.6			2.4	1.9	0.5	0.04	0.4	
	75	0.5	0.6			2.4	1.9	0.5	0.04	0.4	
	100	0.5	0.6			3.6	4.3	1.7	0.09	0.6	

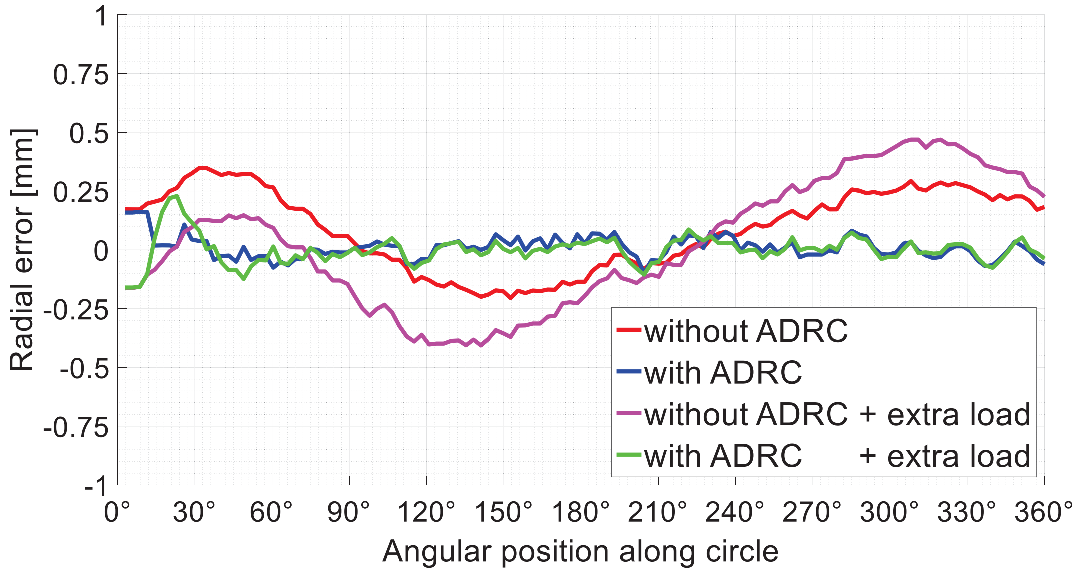
4.4 The experimental results of the proposed ADRC controller

The results of the aforementioned three cases of the experiments, without and with ADRC, are shown for the first case in red and blue, for the second case in magenta and green in Figures 4.4, and 4.5, and for the third case in red and blue in Figures 4.6, and 4.7, respectively. In addition, the maximum and mean values of the absolute radial errors (i.e., the radial path deviation) of the curves shown in Figures 4.4, and 4.5 are shown in Figure 4.8. The non-model based SMC algorithm performance is given in magenta color in Figures 4.6 and 4.7 after applying the controller in the same aforementioned third test.

The results are demonstrated as follows. In the first part, the results highlight the robot's bad performance before applying the proposed ADRC controller for the three cases at four different robot's linear velocities (25, 50, 75, and 100 mm/s). In the second part, the performance is exposed after applying the ADRC algorithm for the three cases, in addition to the robustness test by applying the aforementioned SMC algorithm instead of the ADRC in the third case, while the robot is moving at the same linear velocities of the first part. All the associated results are demonstrated in Table 4.2.

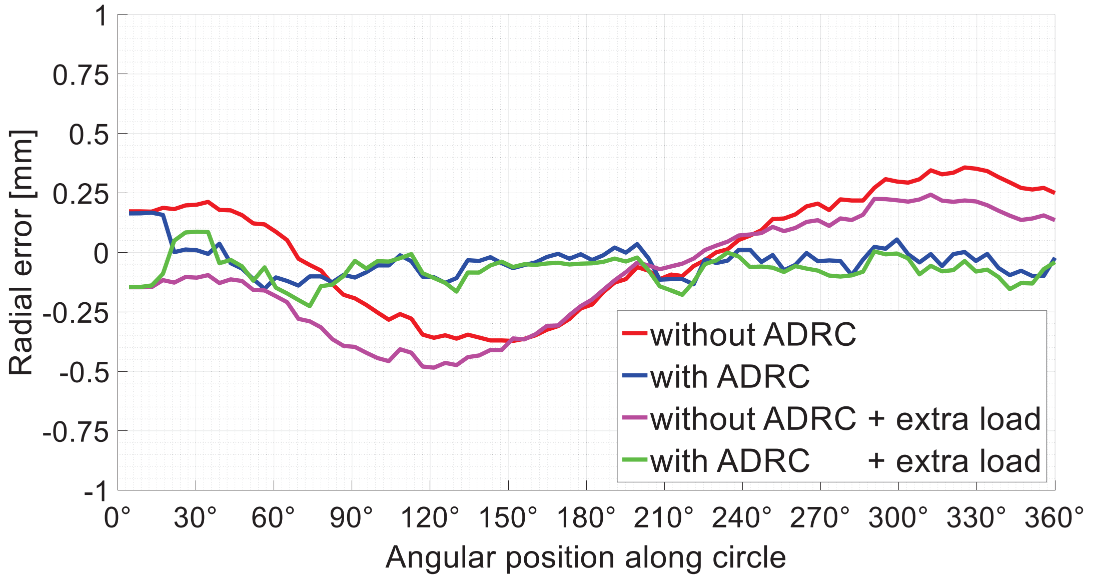


a) TCP velocity of 25 mm/s

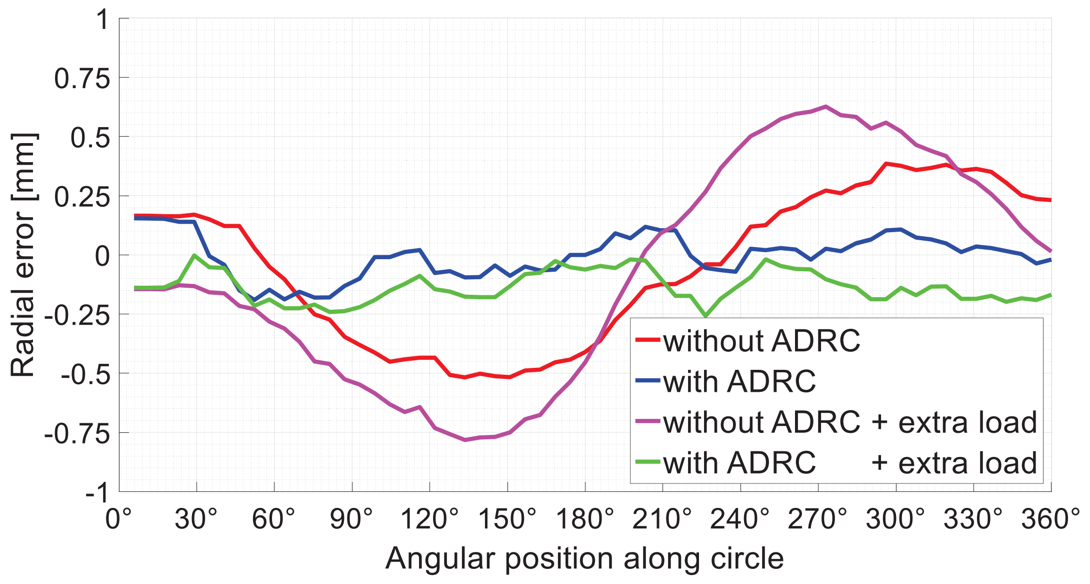


b) TCP velocity of 50 mm/s

Figure 4.4 Robot's radial error measured with the ballbar at different TCP velocities, with and without ADRC controller, with robot's rated payload (0.5 kg) and with robot's full payload (1 kg), at robot's low speeds

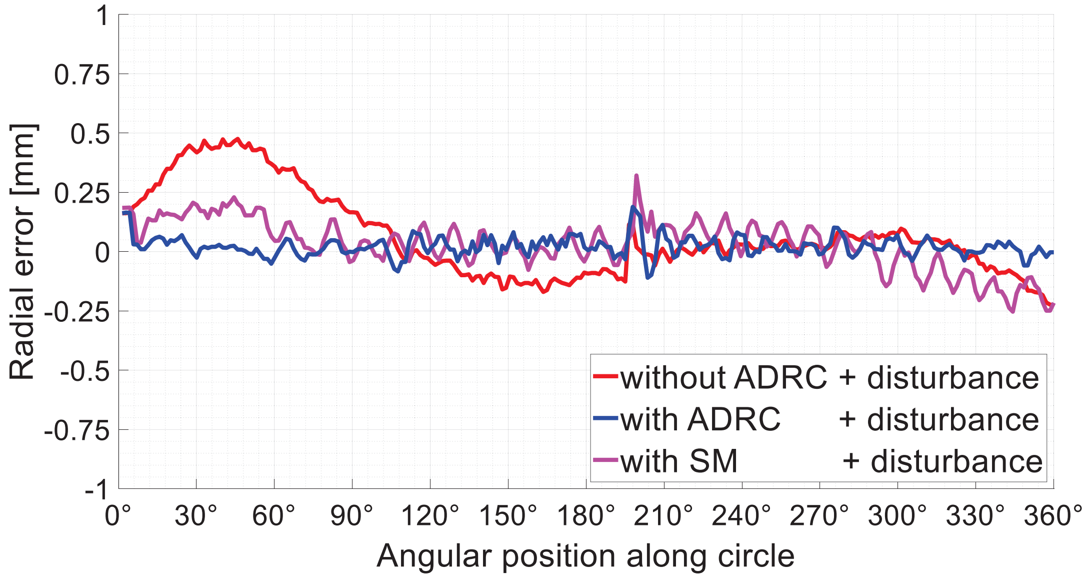


a) TCP velocity of 75 mm/s

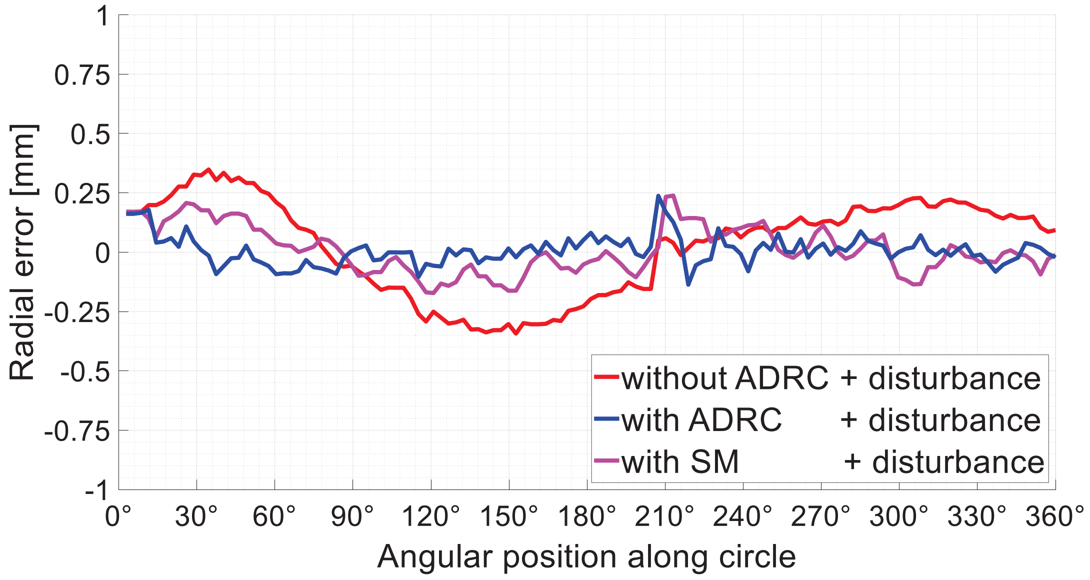


b) TCP velocity of 100 mm/s

Figure 4.5 Robot's radial error measured with the ballbar at different TCP velocities, with and without ADRC controller, with robot's rated payload (0.5 kg) and with robot's full payload (1 kg), at robot's high speeds

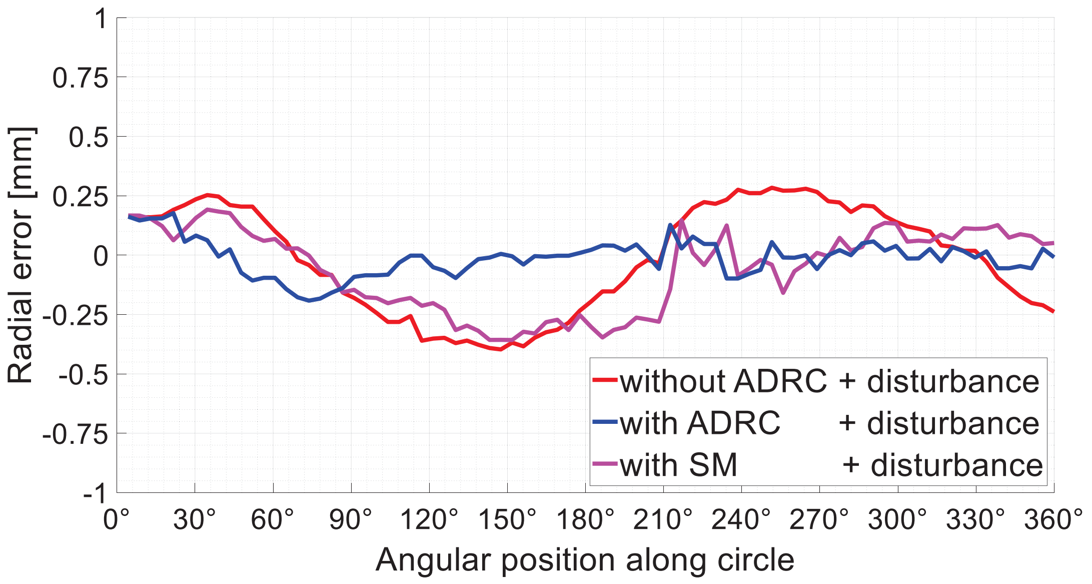


a) TCP velocity of 25 mm/s

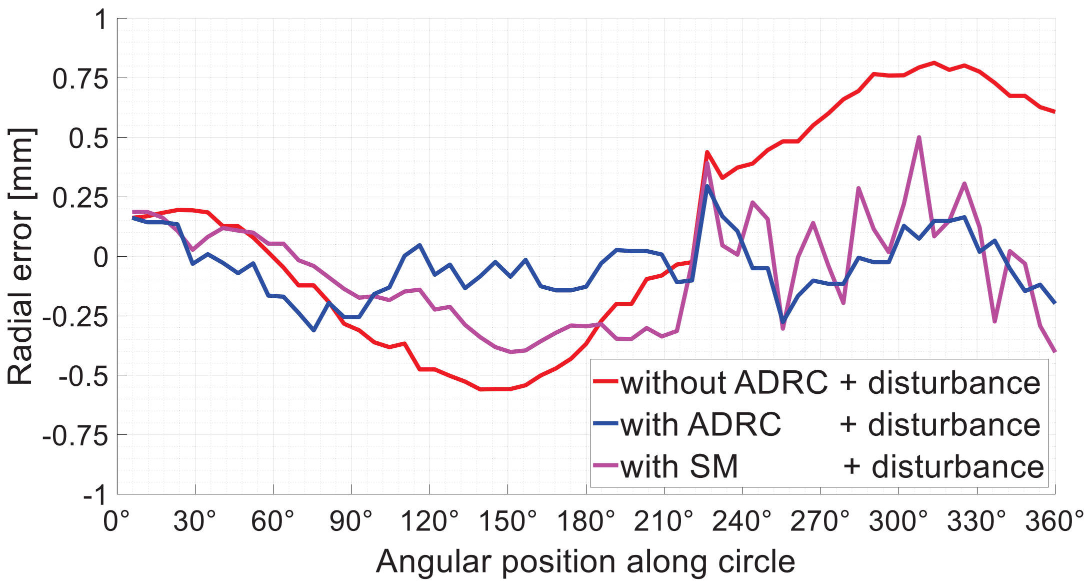


b) TCP velocity of 50 mm/s

Figure 4.6 Robot's radial error measured with the ballbar at different TCP velocities, with ADRC controller, SMC controller and the sudden introduction of an extra load at angular position between 190° to 230°, at robot's low speeds



a) TCP velocity of 75 mm/s



b) TCP velocity of 100 mm/s

Figure 4.7 Robot's radial error measured with the ballbar at different TCP velocities, with ADRC controller, SMC controller and the sudden introduction of an extra load at angular position between 190° to 230°, at robot's high speeds

4.4.1 Before applying the ADRC controller

The idea of these four experiments is to present the robot's actual circular path error using the precise ballbar sensor from Renishaw company with a measurement range of ±1 mm, and accuracy of ±0.001 mm, under the different aforementioned robot's linear velocities, and for each case individually. It is very important to layout this error and to plot it very precisely using the ballbar which provides very accurate measures of any variations in the test circle radius traced by the robot during its circular path. By recording and plotting the ballbar error reading it from its buffer in real-time, we are able to measure the robot's radial error and to figure out to what extent this error can reach while applying the three aforementioned cases and before implementing the ADRC controller. We generally choose the plot colors to be in red for the first and third cases, and in magenta for the second case to express the robot's bad performance with low accuracy before applying the proposed ADRC algorithm in Figures 4.4, and 4.5. In addition, the absolute radial errors of those curves (i.e., the radial path deviation) are introduced by its maximum and mean values of the absolute radial errors and are laid out in Figure 4.8.

4.4.1.1 The first case

In this first case, we are testing the robot's normal operating performance with the robot's rated payload of 0.5 kg. The plots of the robot's rated payload (0.5 kg) in red color in Figures 4.4 and 4.5 illustrate that the starting initial ballbar error is around 0.16 mm while the path deviation increases rapidly when the desired TCP velocity increases. In this case, the robot is controlled in velocity mode without the use of any feedback or external controller and under the action of only the robot's embedded controller. The robot is not capable to keep up with this initial error for the whole path and it swings between upward and downward while its movement along the circle circumference. Now and after we clearly see how the robot is responding in the velocity mode, and after we determine to what extent can the error reach during different robot's velocities, we are able to judge the proposed ADRC controller, not only in this case but also when the other cases are being applied.

Two important remarks must be made here. Firstly, note that the robot's path performance is more than two-times better when the robot is controlled in position mode, but radial errors could still surge up to 0.5 mm at TCP velocity of 170 mm/s (not shown here). Secondly, note that we have used the same 0.1 s dictated by the limits of the ballbar, in order to make a fair comparison. A much smaller sampling time (e.g., 0.002 s) could lead to results similar to those in the case of position mode control. Nevertheless, without external feedback, it is impossible to guarantee a maximum path deviation of 0.2 mm, as required in some industrial applications.

4.4.1.2 The second case

In the second case, the robot's performance is being tested with the robot's full payload of 1 kg as exhibited in Figure 3.7a, to check later the proposed controller robustness. The idea is to monitor the robot circular motion in an extreme condition, in other words, to draw an idea about the variability in the error between the half payload of the first case and the full one of 0.5 kg and 1 kg, respectively. This is also delineating whether the proposed controller will be effective and robust or not. The plots are presented in magenta color in Figures 4.4 and 4.5, where the same initial point position of the ballbar of 0.2 mm/s is selected before putting the extra load on the top of the robot's end-effector, to put the robot in the same exact condition of the first case. However, the starting point of the motion is shifted down from 0.2 mm/s to be -0.18 mm/s due to the effect of this full payload. The same observations can also be seen on the plots in magenta color related to the robot's full payload (1 kg), the path deviation increases rapidly when the desired TCP velocity increases, while the robot is controlled in velocity mode without the use of any feedback or external controller and under the action of only the robot's embedded controller. Overall, the robot movement thereafter becomes worst as can be seen when the full payload is applied.

4.4.1.3 The third case

Finally, in the third case, we need to test the robot's performance experimentally under an exerted disturbance, to monitor the robot's performance under external disturbance, and to decide later whether the proposed controller has the capability of rejecting the disturbance or not. We agree that the unit step disturbance (load variations) especially with this tiny error is the best disturbance that can be selected among the set-point changes and noise. Therefore, an extra load of 0.5 kg is suddenly dropped on the top of the robot's end-effector during the robot's circular motion at an angular position between 190° and 230° for the rest of its motion as demonstrated in Figure 3.7b. The robot's performance thereafter is analyzed and monitored under this unit step disturbance signal (sudden extra load of 0.5 kg). The plots in red color in Figures 4.6 and 4.7 represents the same performance of the first case of the error increase proportionally with higher robot's TCP velocities, except after the moment of the dropping of the extra weight (step signal disturbance).

4.4.2 After applying the ADRC controller

In this section, we repeat the same four experiments for each case individually under the different aforementioned robot's linear velocities to present the robot's actual circular path error, however, this time after applying the ADRC proposed controller. It is very important to record the error after implementing the ADRC algorithm and to plot it very precisely using the same ballbar under starting from the same initial ballbar position of 0.2 mm/s. Then to compare these new results after the controller to those before it. Here, we choose to plot the graphs in blue color for the first case, and in green color for the second one, in order to distinguish them from the ones before the controller (in red and magenta colors), except for the results related to SMC which is applied only for the third case, and after applying the SMC algorithm, the plots are in magenta color.

4.4.2.1 The first case

In this first case, we are testing the robot's normal operating performance with the robot's rated payload of 0.5 kg after applying the ADRC controller starting from the initial ballbar error position of 0.2 mm. The results of the ballbar readings, in this case, are plotted in blue color in Figures 4.4 and 4.5. The plots are showing magnificent improvement in the robot response, starting from the initial error of 0.2 mm/s, ADRC forces the robot to immediately reach the zero values and to keep up with around the zero line for almost the whole remaining radial path especially at the robot's low speeds of 25 and 50 mm/s. The fact is that although the path deviation increases when the desired TCP velocity increases, the absolute path deviation remains well below 0.2 mm for all the robot's TCP velocities. This clearly demonstrates how the ADRC controller enhanced the robot performance during the normal operation with the rated payload of 0.5 kg compared to its performance before the external controller, thanks to the topology behind the ADRC error-based form discussed in section 4.3.

4.4.2.2 The second case

In order to test the robot's performance while pushing the robot to its maximum limits, a full payload of 1 kg is added on the top of the robot's TCP as presented in Figure 3.7a during the whole robot's circular motion at each speed. These experiments go directly to test the ADRC robustness which assists to determine whether the controller still be able to do its job and direct the robot to its zero line at the case of the robot's full payload (1 kg). Obviously, a good ADRC controller performance is noticeable at the beginning of the motion under the robot's full payload forcing the robot's error to be zero, and the rest of the robot's performance remains the same as the robot's rated payload thereafter, which is displayed in green color in Figures 4.4 and 4.5. A faster sampling time would have undoubtedly yielded even better results. The ADRC algorithm is succeeded not only to force the robot to immediately reach the zero values, but also to remain on the zero line for the rest of its motion, which is reflecting a very good controller's robustness for both low and high robot's speeds.

4.4.2.3 The third case

Finally, we are eager to see what the ADRC controller performs after exerting an external sudden disturbance, to inspect if the ADRC performance is good enough to reject a sudden disturbance during the robot motion. In addition, we want to test if any other controller is capable to do the same job of rejecting this disturbance, e.g., the famous SMC algorithm. We have two methods to exert this intended disturbance, one by impeded it in the MATLAB code (i.e., change suddenly the desired circular radius), or another method, by dropping a sudden extra load of 0.5 kg on the robot's end-effector support experimentally. We choose the second method because it is more practical and reflects a real disturbance that might happen during the real robot's motion. Then we record and plot the ballbar error to investigate the controller response if can overcome this sudden change and brings the robot's error to its zero line, or not. Therefore, a unit step disturbance signal is exerted by dropping a sudden extra load of 0.5 kg during the robot's circular motion at an angular position between 190° and 230° for the rest of its motion. We always tried to drop this load at the exact place each time during the robot's circular movement, to have a better comparison between the cases before and after our external controller.

The performance of the ADRC scheme is also confirmed in this third case. Nevertheless, the controller is able to stabilize and minimize the resulting ballbar error at and after the moment of dropping the extra weight 0.5 kg on the top of the robot's end-effector, thanks to the feature of ADRC disturbance rejection as shown in Figures 4.6 and 4.7 in blue color. We can clearly notice that the extra load is suddenly affecting the motion at the moment of the drop (i.e., a large change in the robot error), however, the controller took the right action directly in the next SOM, and instantly the controller brings the error to zero compared to the case before applying the controller, which confirms its low convergence time. The ADRC disturbance rejection performance is also affirmed when we decided to remove the ADRC and implement the SMC instead. Although we choose a medium sliding mode tracking bandwidth for the SMC, taking into account during its design the trade-off between the system tracking accuracy and the well-known SMC chattering phenomena (Ebrahimi, 2014), unfortunately, we couldn't get better performance than ADRC. As can be seen from the results in magenta color in Figures 4.6 and 4.7, the tracking accuracy is good at the beginning of motion and before the moment of dropping the extra weight; however, the chattering phenomena appears after that and brought excessive high-frequency measurement noises especially with high robot velocities, in addition, the convergence time is affected. On the contrary, the ADRC is being able to overcome the sudden disturbance smoothly with no chattering effect and with fast convergence time.

Table 4.2 Comparison of robot error under different robot linear velocities without/with ADRC controller starting from an angular position of 30°

Parameters		**Linear velocity (mm/s)**	**Rated payload (0.5 kg)**		**Full payload (1 Kg)**		**With disturbance**		
			Without ADRC	**With ADRC**	**Without ADRC**	**With ADRC**	**Without ADRC**	**With SM**	**With ADRC**
Radius error range (mm)	**min**	**25**	-0.087	-0.076	-0.310	-0.087	-0.230	-0.253	-0.110
	max		0.421	0.094	0.373	0.081	0.475	0.321	0.189
	min	**50**	-0.205	-0.099	-0.310	-0.087	-0.343	-0.172	-0.137
	max		0.348	0.081	0.065	0.102	0.348	0.238	0.237
	min	**75**	-0.372	-0.153	-0.310	-0.087	-0.398	-0.358	-0.192
	max		0.357	0.054	0.065	0.109	0.284	0.192	0.127
	min	**100**	-0.517	-0.191	-0.228	-0.087	-0.560	-0.404	-0.311
	max		0.385	0.119	0.065	0.109	0.813	0.501	0.295
Mean absolute deviation (mm)		**25**	0.105	0.024	0.185	0.025	0.124	0.079	0.029
		50	0.150	0.032	0.101	0.026	0.176	0.073	0.039
		75	0.216	0.040	0.108	0.031	0.201	0.152	0.052
		100	0.285	0.066	0.080	0.027	0.434	0.193	0.096

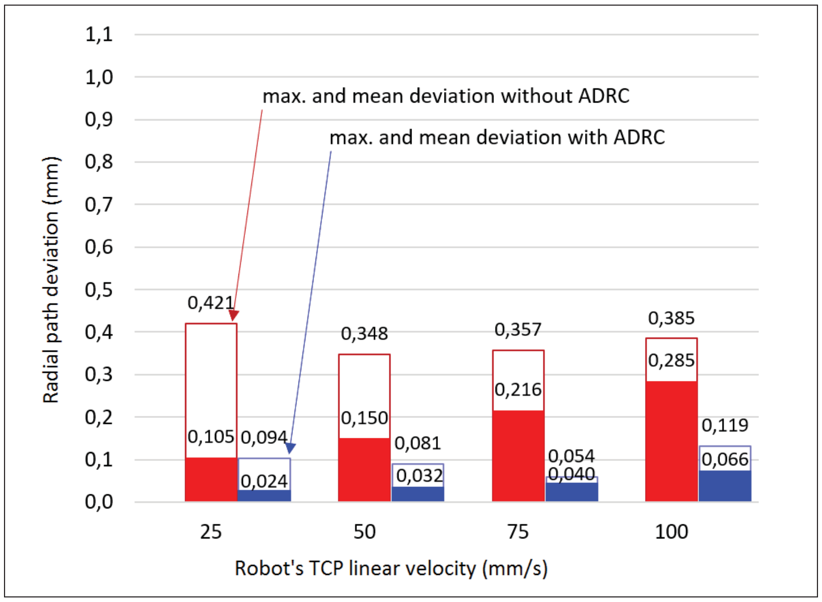


Figure 4.8 Maximum and mean radial path deviations with and without ADRC controller for different robot TCP velocities

CHAPTER 5

DESIGN OF MULTI-STAGE INTERACTION CONTROL FOR ROBOT HAND-GUIDING

This chapter addresses the second main objective of this thesis: the development of a novel controller capable of providing robots with smooth hand-guiding capabilities at different speeds. More specifically, the objective is to make the Meca500 robot end-effector follow the finger mouse during human manipulation in a very smooth and soft manner avoiding human-robot cross-coupling, robot's motor perturbations, oscillation, noise, or aggressive robot response. Note that the system dynamics and the rate of change in this mutual tracking control problem may significantly change in amplitude, time, and direction during the normal robot's movements. The proposed controller is multi-stage, i.e., it consists of multiple blocks connected in series and based respectively on fuzzy, admittance, and ADRC control techniques as indicated in Figure 5.1. The controller stages have each a specific function aimed at integrating the inferred human intention, to solve the mouse-robot interaction issues stated above, and to obtain a smooth and high performance combined human-robot movement. The fuzzy logic controller which represents the first stage is designed to cope with the complex human-robot interaction, followed by an admittance controller, designed to damp the oscillations that might appear in the robot's slow motion. Finally, in the third stage, an active disturbance rejection control (ADRC) tracking differentiator (TD) is exploited to filter the noise and add an intended delay to the signal for better performance. As mentioned in section 3.6, the mouse has 6-DOF with six outputs, one for each axis introduced in Figure 3.24. The control architecture used consists of an ensemble of six single-input, single-output (SISO) control systems which means that each axis is treated and controlled individually, with its own multi-stage controller as put forward in Figure 5.2.

The human intention inference is first clarified in this chapter, followed by the approach used to design the different controllers and the practical tuning of each one of them. To evaluate the performance of the proposed controller, multiple experiments have been performed, involving both fine path following and rapid displacements. One of these experiments has been filmed and presented. In addition, the response from the 6-axis sensor is shown for the system both before and after the implementation of the proposed controller.

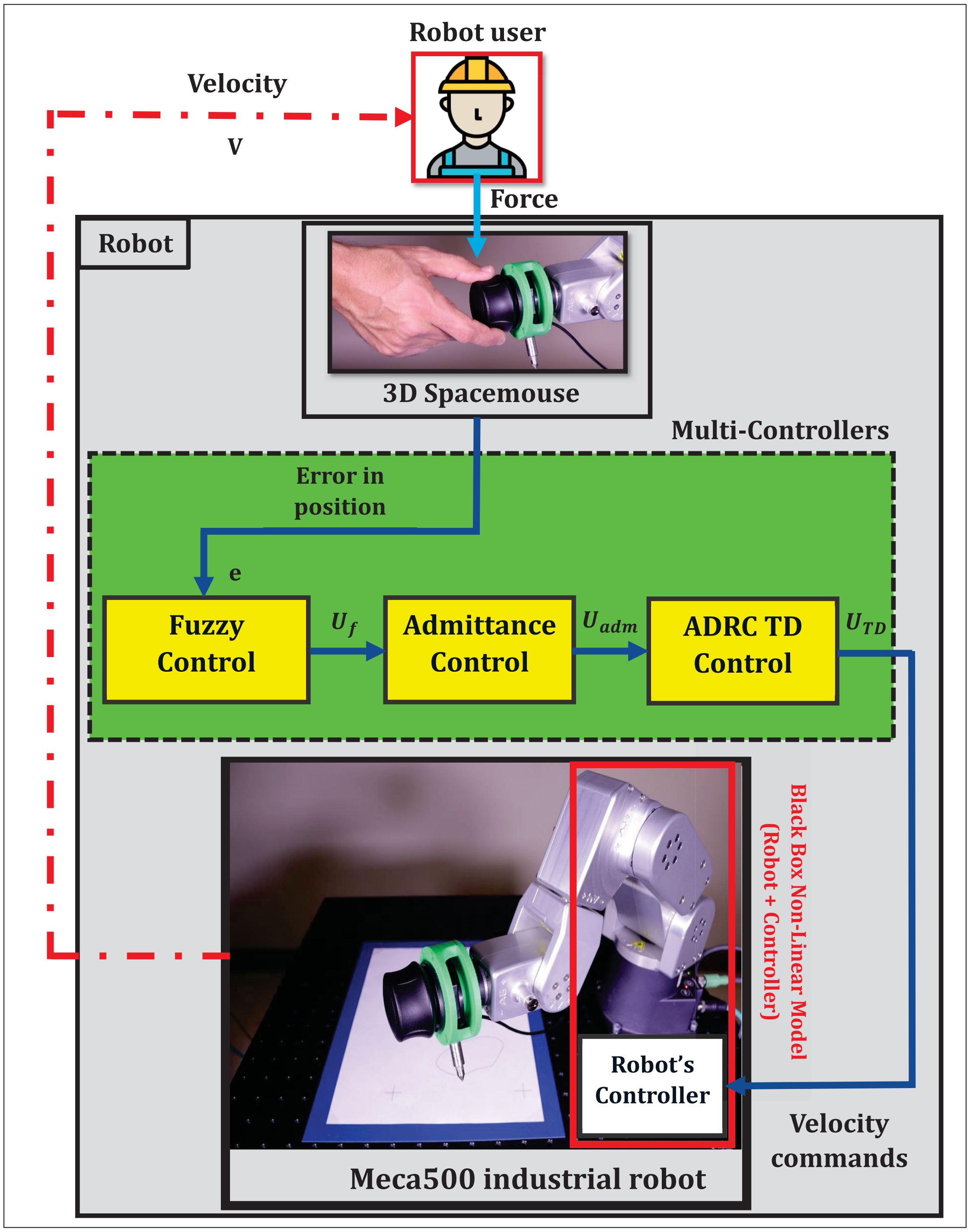


Figure 5.1 The proposed external controller block diagram of the mouse-robot system

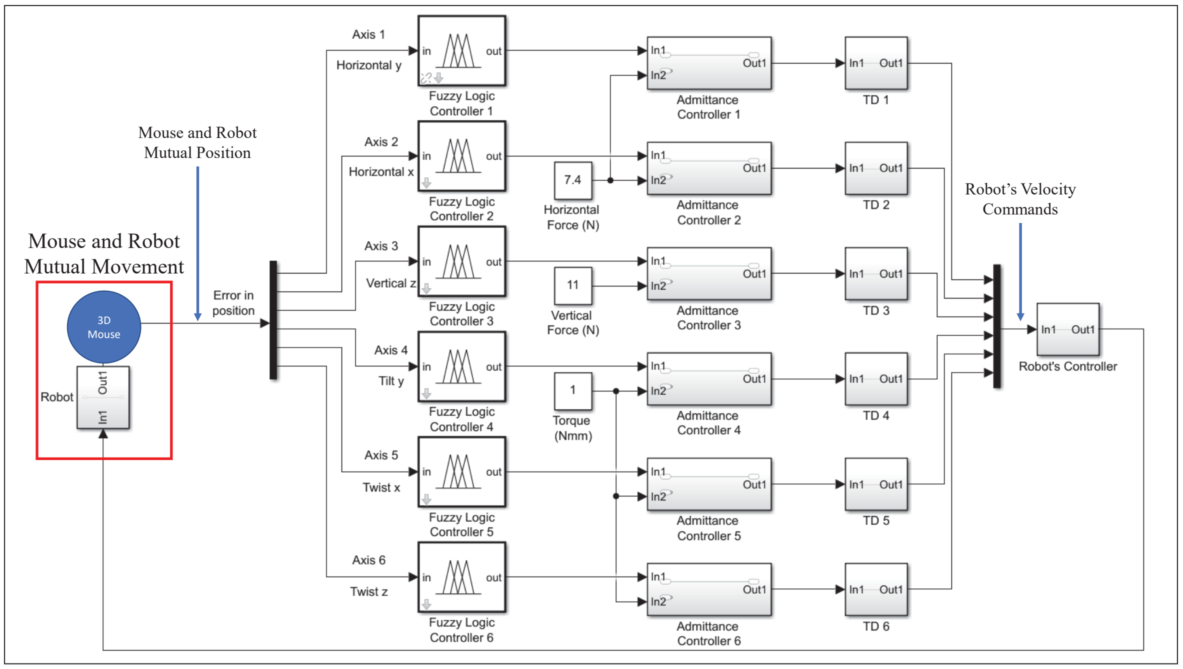


Figure 5.2 The structure of the Mouse-Robot SISO Control block diagram

5.1 Human intention and the need for the proposed controllers

The physical interaction between a human and a robot is an area that promotes tasks like heavy object collaborative lifting or programming by demonstration. During such tasks, the human manipulates the robot by applying external forces and torques to it by sensors (Dimeas, Moulianitis, Papakonstantinou & Aspragathos, 2016). For safe interaction and efficient manipulation, it is important to predict the robot reaction to this interaction and make sure it operates within its limits and capability. In order to do so, the need to regulate the user's movement with respect to the robot plays an essential role and is considered the key for solving the human-robot interaction problem. The main challenge is the unstructured environment between the robot and the human which causes the aforementioned issues (see subsection 3.6.2). There are three human interaction cases that can be considered when the user manipulates the robot with the SpaceMouse, namely accelerate, stop, or reverse direction. Two controller inputs can be considered in this case, the error in mouse position (deflection from its center) and the rate of change of this error, while its output might be a position or a velocity. In the first human interaction case, if the user starts to accelerate the robot, the desired mouse position (deflection) will be in the direction of the real robot velocity. In order to help the user to accelerate the robot, the resultant controlled desired position output should be generated by the controller in medium or large value to match the magnitude and the direction of both the desired position and robot real velocity. For the second human interaction case, if the user wants to stop the robot, the desired mouse position must be in the direction opposite to the robot's real velocity direction. In accordance with the controller actions and from intuition, to help the user to stop the robot, ideally, the controller output will be transformed from the current state to the lower one, i.e., from large output to medium, from medium to small, or from small to zero. In the third human interaction case, if the user wants to reverse the direction of motion, two phases will be tracked: a deceleration phase until the robot velocity reaches zero, then followed by an acceleration phase in the opposite direction. Here, the same concept of the previous first and second human interaction cases is applied, nevertheless, in this case, the user does not want to stop the moving robot but rather reverse its direction.

The fuzzy logic controller (FLC) is firstly proposed to detect the human intention and to determine whether the user intends to accelerate or decelerate, in accordance with the mouse position direction if it is in the same direction or opposite to the robot's real velocity. However, the FLC performance results have some oscillations in high robot speeds. Therefore, the second controller (admittance control) was designed to damp the oscillation that might be generated when FLC is applied, especially if a high maximum robot's velocity is selected by the user (see section 3.6.1). In the admittance control, it is possible to include the magnitude of the mutual interaction force between the mouse and the robot in the analysis (the force will be smaller if the user intends to stop). After the admittance control implementation, noise is introduced by the force/torque sensor (SpaceMouse); additionally, the robot velocity may still be too fast when changing from one direction to its opposite, which affects the motion and causes oscillations. Accordingly, the third controller (just the tracking differentiator (TD) part of an ADRC) is added to do four tasks; (1) filtering the noise, (2) adding an intended delay when the motion is at high speed, (3) controlling the amount of noise filtration and the intended delay based on the desired mouse deflection, and (4) converting the mouse deflections into velocities to be sent to the robot.

5.2 Fuzzy Logic Controller (FLC)

Fuzzy logic is an artificial intelligence (AI) technique that deals with approximate reasoning algorithms used to emulate human thinking and decision-making in machines. These algorithms are used in applications where process data cannot be represented in binary form, in other words, it is a method of mathematically dealing with information that is "gray" in nature (Ying Bai *et al.*, 2005). The design of FLC is standard as it takes the control decision by following specific steps. The three main actions performed by a fuzzy logic controller are; (1) Fuzzification, (2) Fuzzy processing, and (3) Defuzzification. Input and output membership functions (MFs) are first defined, secondly, a number of IF–THEN rules are constructed to hold the required system knowledge and then mapped to a lookup table in a process called a fuzzification. Thirdly, an interference mechanism evaluates the control rules based on the lookup table and decides which of them is relevant at the current time. Finally, a defuzzification process transforms the concluded outputs reached by the interference mechanism into relevant input to the plant. The main structure of the proposed FLC corresponding to our case is portrayed in Figure 5.3.

Generally, the fuzzy control space is partitioned into small regions according to different input conditions, and for continuity, each region is usually overlapped by its neighbors. The two major FLC's categories are the Mamdani type and the Takagi–Sugeno (TS) type, the only difference between the two is that the first type generates the control actions by using fuzzy numbers, while the latter uses linear functions of the input variables to make the control decision (Reznik, 1997). It is useful at this point to explain the similarities and differences between fuzzy logic control and other artificial intelligence techniques. Both artificial intelligence techniques and fuzzy logic control use a set of IF-THEN rules which describe what action is to be taken if a certain set of conditions is met. Artificial intelligence rule bases, however, have a finite number of control points - one control point for every IF-THEN rule. In a fuzzy rule base, there are still a limited number of IF-THEN rules, but an infinite number of control points is possible because a fuzzy rule base maps membership values to corresponding control values. This means that a fuzzy rule base recognizes information that is fuzzy or partially true in nature, and can partially "fire" or invoke more than one rule at any one time (Zadeh, 1973).

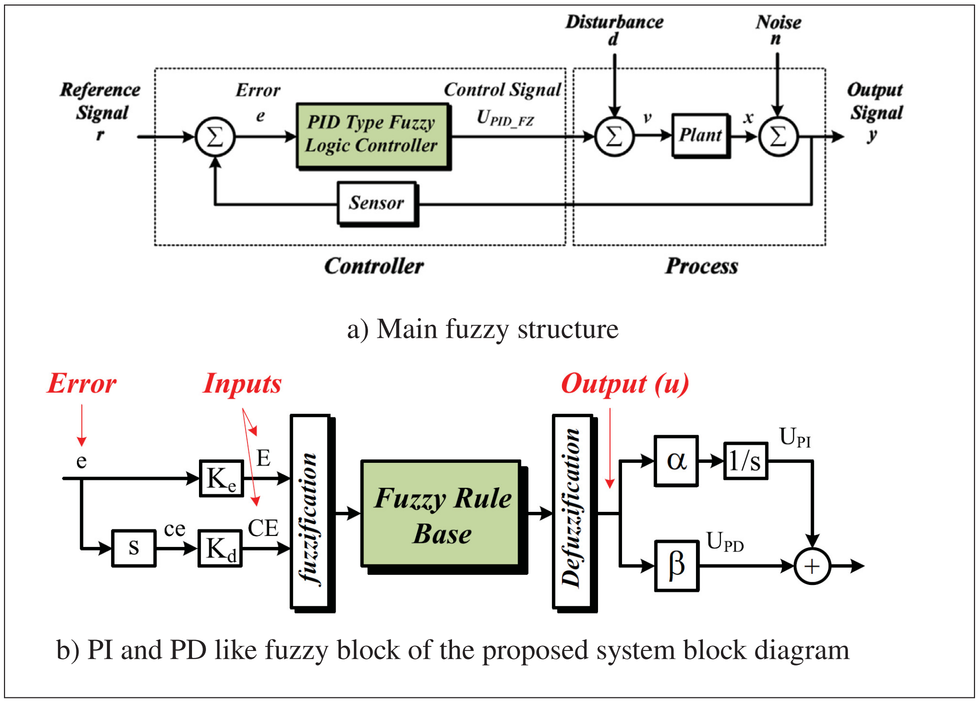


Figure 5.3 PID like fuzzy control closed-loop structure
Taken from Fereidouni *et al.* (2015)

5.2.1 Fuzzification

Fuzzification is a process of receiving the input data, also known as a fuzzy variable, and then analyzing it according to user-defined charts called Membership Functions (MFs). It assigns the input data into a grade from logic 0 to 1 based on how well it fits into MFs. It can also have many shapes depending on the data set. MFs are made up of connecting line segments defined by line endpoints. Each MF can have up to three line segments with a maximum of four endpoints. The grade at each endpoint must have a value of 0 or 1, and each fuzzy controller input can have several MFs, with nine being the maximum. MFs have six types of shapes; triangular, z-shape, trapezoidal, s-shape, sigmoid, and Gaussian (The MathWorks, 2020b). Besides the MFs, fuzzification also has another component called labels, which defines each MF and it spans from the data range's minimum point label to its maximum point label, and in between such as; negative large (NL), negative medium (NM), negative small(NS), Zero (ZR), positive small (PS), positive medium (PM), and positive large (PL) (Reznik, 1997).

5.2.2 Fuzzy processing

It is a process which analyzes the input data as defined by MFs to determine the right control output data then performs two actions, the rule evaluation, and the fuzzy outcome calculation. On one hand, the rule evaluation uses a reasoning process composed of IF-THEN rules, each providing a response or outcome. Basically, a rule is activated, or triggered, if an input condition satisfies the IF part of the rule statement. This results in a control output based on the THEN part of the rule statement. In a fuzzy logic system, many rules may exist, corresponding to one or more IF conditions. A rule may also have several input conditions, which are logically linked in either an AND or an OR relationship to trigger the rule's outcome as demonstrated in Figure 5.4b. On the other hand, the fuzzy outcome calculation is generated once a rule is triggered, this means that the input data belongs to a membership function that satisfies the rules IF statement, and this is the time that the rule will generate an output outcome. This fuzzy output is composed of one or more membership functions (with labels), which have grades associated with them. The outcome's membership function grade is affected by the grade level of the input data in its input membership function. However, the output membership function that is selected for the final output value depends on the user's programming of the IF...THEN rules (Reznik, 1997), as shown in Figure 5.5a.

5.2.3 Defuzzification

Defuzzification is the process responsible for computing the outcome values corresponding to each label, then the fuzzy controller generates the final decision control output. In other words, the defuzzification process examines all of the rule outcomes after they have been logically added and then computes a value that will be the final output of the fuzzy controller. The PC sends then this value to the output module. Thus, during defuzzification, the controller converts the fuzzy output into a real-life data value relevant to the plant (Reznik, 1997).

5.2.4 Design of our proposed FLC

In view of the fact that the main controller design is premised on a SISO control technique and since the 6-DOF mouse has six outputs, one for each axis as demonstrated in Figure 3.24, each axis has thus been designed to have its own fuzzy control system. The proposed PID like fuzzy controller (PID-FLC) system for each axis has one input, which is the mouse position feedback error e (deviation of this axis from its center after the robot responds to the desired motion)[1], and one output $U_{PID_{FZ}}$, which is the fuzzy control action in position as outlined in Figure 5.3a. The inner fuzzy part has two inputs, the mouse position error e and the rate of change of this position error signal ce as demonstrated in the fuzzy part of Figure 5.3b. The figure parameters e, ce, E, and CE indicate the error, change of error, normalized error, and normalized change of error, respectively; while the parameters K_e, K_d, α, and β are two inputs and two outputs scaling factors, respectively, because both the inputs and the outputs have to be normalized to match the MFs-input and output signals (Mudi & Pal, 2001). Each input and each output uses seven membership functions (MFs) each as shown in Figure 5.4a. We use Takagi–Sugeno (TS) fuzzy system to generate control actions that use linear functions of the input variables and take the required control decision. The mouse fuzzy tracking system is designed with the aid of MATLAB/SIMULINK software, fuzzy logic designer toolbox (The MathWorks, 2020b).

The PID like fuzzy has three widely accepted and relatively simple controllers: PD, PI, and PID (Åström, Hägglund, Hang & Ho, 1992). The mathematical representation of these controllers can be written as:

$$U_{PD_{FZ}} = K_P\, e + K_D \frac{de}{dt} = K_p(e + T_d \frac{de}{dt}),\ K_D = \ K_p T_d, \tag{5.1}$$

$$U_{PI_{FZ}} = K_P\, e + K_I \int edt = K_p(e + \frac{1}{T_i} \int edt),\ K_I = \ K_p/T_i, \tag{5.2}$$

[1] The mouse feedback error in position e is the user desired reference signal (i.e., the desired mouse deflection) subtracted from the sensor feedback which is the robot responding motion that affects the user desired mouse deflection. Therefore, in our case, the mouse deflection readings are directly the error e and not the desired deflection, because the mouse is mounted on the robot's end-effector and real-time deflection is affected by each movement of the robot.

$$U_{PID_{FZ}} = K_P\, e + K_I \int edt + K_D \frac{de}{dt} = K_p(e + \frac{1}{T_i} \int edt + T_d \frac{de}{dt}). \tag{5.3}$$

where e is the mouse position feedback error, K_P is the proportional gain, K_I is the integral gain, K_D is the derivative gain, T_i is the integral time, and T_d is the derivative time.

In this work, to control the robot via the mouse (which is mounted on the robot's end-effector) as smoothly and precisely as possible, we need to use different fuzzy sets for each of the following variables: mouse position feedback error, rate of change error, and control position output. In traditional set theory, the membership of an object belonging to a set can be one of two values: 0 or 1. The proposed system has a set of linguistic variables to represent the output control signal, five triangular MFs, one S-shape MF, and one Z-shape MF as shown in Figure 5.4a, and defined in sub-subsection 5.2.1. To establish the structure of the FLC, the triangle shapes of the membership function were used. It was supposed that the mouse position is "zero" in an imprecise way. If the position is zero and the change in position is a small negative, then the control output is small negative and vice versa. If the position is small negative and the position change is a small negative, then the control output is a small negative. For example, if the position is actually 0.3 and the mouse velocity is 0.7, the value of the triangle MF at that point would be 0.622 as demonstrated in Figure 5.5a.

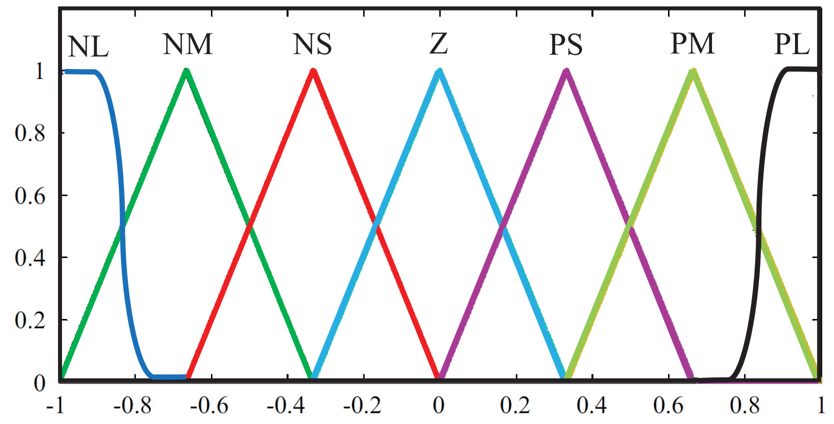
These rules are reasonable and straightforward resembling human reasoning. It was found experimentally that using the Z, S functions, at the ends of MFs and triangle-shaped MFs at the rest of MFs yielded smoother tracking performance and faster time responses than FLCs that used either Trapezoidal or Gaussian MFs, the limits of the input variables and MFs have been designed (mapped) to match the limitation of the real inputs and the relations between them. The most important thing in FLC system is the design process of the membership functions for inputs and outputs, and the design process of a fuzzy if-then rule knowledge base. For a rule base to be valid, it must incorporate information about every possible condition that the system can be expected to encounter. Each unique combination of conditions is corresponds to a control decision in the form of a rule. In this tracking problem, two variables are considered for each axis with each variable breaking its domain into seven input membership functions or conditions.

Thus a total of 49 rules were constructed for the vertical tracking axis control, a sample of these rules is demonstrated in Figure 5.4b. These rules are formulated one by one, and then the whole rules set is analyzed and considered to be complete if making any combination of the inputs fired at least one rule, consistent if it does not contain any contradictions, and continuous if it does not have neighboring rules with output fuzzy sets that have an empty intersection. Once the lookup table is constructed, no further modification of its structure or entries is ever attempted. Therefore, the fuzzy set, or membership functions, and control IF-THEN rule are combined together to form the lookup table as laid out in Table 5.1. The outcome of the procedure is a fuzzy variable that is crisp. This design method is used to calculate the fuzzy output in real-time and this makes the robot more concise and easier to be manipulated.

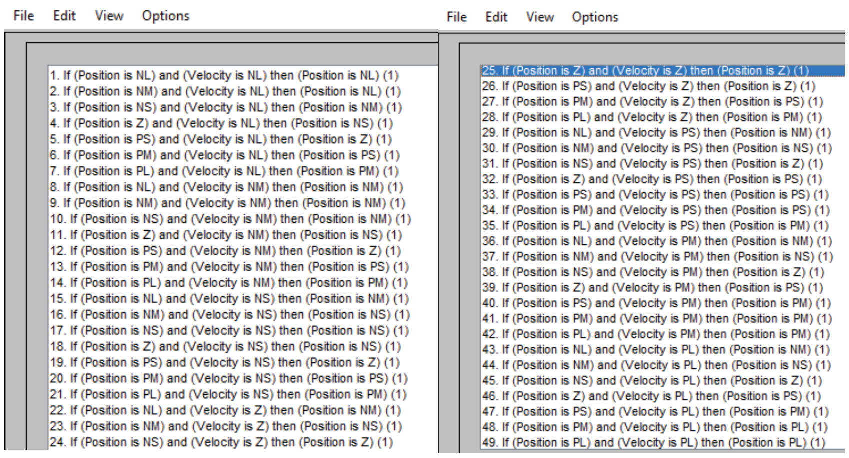
Table 5.1 The proposed system truth table

Position / Rate	**PL**	**PM**	**PS**	**Z**	**NS**	**NM**	**NL**
PL	PL	PM	PM	PM	PM	PM	PM
PM	PL	PM	PS	PS	PS	PS	PS
PS	PM	PM	PS	Z	Z	Z	Z
Z	PS	PS	PS	Z	NS	NS	NS
NS	Z	Z	Z	Z	NS	NM	NM
NM	NS	NS	NS	NS	NS	NM	NL
NL	NM	NM	NM	NM	NM	NM	NL

To better understand how the fuzzy rules are working, one should concentrate on the details of the tracking problem and consider each step. It is desired to position the tracking robot so that it is in line with the center of the mouse. If the robot is far out of position with respect to the mouse center, then one could make the rule "If the mouse position is LARGE, then the control output is LARGE" which makes sense. If the platform is seriously out of line with the target, then a large control force is needed to move the robot quickly back to its desired position. Likewise, if only a small discrepancy exists between the robot and the mouse, then one could derive the control rule "If the mouse position is SMALL, then the control output is SMALL", which also makes sense. If there is only a small inconsistency between the robot and the mouse, then only a small



a) The Fuzzy membership functions shapes for the controller’s two inputs and one output

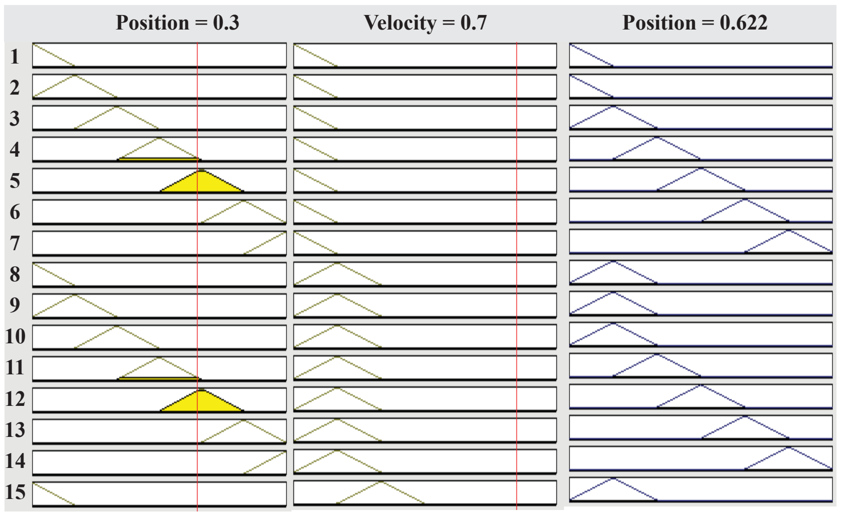


```
1. If (Position is NL) and (Velocity is NL) then (Position is NL) (1)
2. If (Position is NM) and (Velocity is NL) then (Position is NL) (1)
3. If (Position is NS) and (Velocity is NL) then (Position is NM) (1)
4. If (Position is Z) and (Velocity is NL) then (Position is NS) (1)
5. If (Position is PS) and (Velocity is NL) then (Position is Z) (1)
6. If (Position is PM) and (Velocity is NL) then (Position is PS) (1)
7. If (Position is PL) and (Velocity is NL) then (Position is PM) (1)
8. If (Position is NL) and (Velocity is NM) then (Position is NM) (1)
9. If (Position is NM) and (Velocity is NM) then (Position is NM) (1)
10. If (Position is NS) and (Velocity is NM) then (Position is NM) (1)
11. If (Position is Z) and (Velocity is NM) then (Position is NS) (1)
12. If (Position is PS) and (Velocity is NM) then (Position is Z) (1)
13. If (Position is PM) and (Velocity is NM) then (Position is PS) (1)
14. If (Position is PL) and (Velocity is NM) then (Position is PM) (1)
15. If (Position is NL) and (Velocity is NS) then (Position is NM) (1)
16. If (Position is NM) and (Velocity is NS) then (Position is NS) (1)
17. If (Position is NS) and (Velocity is NS) then (Position is NS) (1)
18. If (Position is Z) and (Velocity is NS) then (Position is NS) (1)
19. If (Position is PS) and (Velocity is NS) then (Position is Z) (1)
20. If (Position is PM) and (Velocity is NS) then (Position is PS) (1)
21. If (Position is PL) and (Velocity is NS) then (Position is PM) (1)
22. If (Position is NL) and (Velocity is Z) then (Position is NM) (1)
23. If (Position is NM) and (Velocity is Z) then (Position is NS) (1)
24. If (Position is NS) and (Velocity is Z) then (Position is Z) (1)
25. If (Position is Z) and (Velocity is Z) then (Position is Z) (1)
26. If (Position is PS) and (Velocity is Z) then (Position is Z) (1)
27. If (Position is PM) and (Velocity is Z) then (Position is PS) (1)
28. If (Position is PL) and (Velocity is Z) then (Position is PM) (1)
29. If (Position is NL) and (Velocity is PS) then (Position is NM) (1)
30. If (Position is NM) and (Velocity is PS) then (Position is NS) (1)
31. If (Position is NS) and (Velocity is PS) then (Position is Z) (1)
32. If (Position is Z) and (Velocity is PS) then (Position is PS) (1)
33. If (Position is PS) and (Velocity is PS) then (Position is PS) (1)
34. If (Position is PM) and (Velocity is PS) then (Position is PS) (1)
35. If (Position is PL) and (Velocity is PS) then (Position is PM) (1)
36. If (Position is NL) and (Velocity is PM) then (Position is NM) (1)
37. If (Position is NM) and (Velocity is PM) then (Position is NS) (1)
38. If (Position is NS) and (Velocity is PM) then (Position is Z) (1)
39. If (Position is Z) and (Velocity is PM) then (Position is PS) (1)
40. If (Position is PS) and (Velocity is PM) then (Position is PM) (1)
41. If (Position is PM) and (Velocity is PM) then (Position is PM) (1)
42. If (Position is PL) and (Velocity is PM) then (Position is PM) (1)
43. If (Position is NL) and (Velocity is PL) then (Position is NM) (1)
44. If (Position is NM) and (Velocity is PL) then (Position is NS) (1)
45. If (Position is NS) and (Velocity is PL) then (Position is Z) (1)
46. If (Position is Z) and (Velocity is PL) then (Position is PS) (1)
47. If (Position is PS) and (Velocity is PL) then (Position is PM) (1)
48. If (Position is PM) and (Velocity is PL) then (Position is PL) (1)
49. If (Position is PL) and (Velocity is PL) then (Position is PL) (1)
```

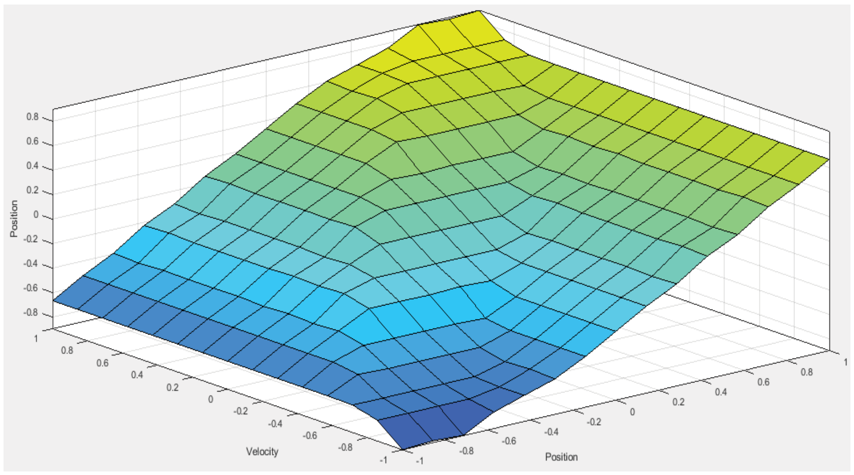
b) The Fuzzy membership functions rules based

Figure 5.4 The proposed fuzzy system using MATLAB/SIMULINK exploring the fuzzy membership functions

correction is needed. Despite all the aforementioned discussion, another factor is involved in our situation, which is the support of the mouse itself on the moving robot’s end-effector. Adding the mutual directional information between both the robot and the mouse while moving, one gives the control outputs further meaning, and here we are proposing the second fuzzy input to be the rate of change of the moving mouse. An example of a rule that takes both variables into account is as follows: If the error is POSITIVE LARGE, AND the mouse velocity is POSITIVE



a) A sample of the proposed fuzzy rules outcomes



b) The proposed system control 3D surface

Figure 5.5 The proposed fuzzy rules outcomes and the surface exploring the relations between them using MATLAB/SIMULINK Taken from The MathWorks (2020b)

LARGE, then the control output is POSITIVE LARGE. If the error is NEGATIVE SMALL, AND the mouse rate is NEGATIVE LARGE, then the control output is NEGATIVE MEDIUM. If the target is well to one side of the center point of the tracking device, and if the tracking device is already moving quickly toward the center point, then little, if any extra effort is needed by the controller to place the tracking platform back on mark. These rules simply mean that if the robot tracking platform is displaced to one side of the center point, then a force is needed in the same direction to bring the robot back in line. All of the rules above are valid, but they incorporate knowledge of two input variables in the control decision. The tracking problem is considered, however, to include information about two variables; position and rate of change of position. Later, the results acknowledge that using fuzzy control alone is not enough to achieve the thesis's second objective; therefore, an admittance control is connected in series with the fuzzy control to get better performance.

5.3 Admittance Control

Impedance and admittance control are two of the most popular control techniques relating force and position used in robotics. Impedance controllers can be used when the system accepts a displacement as input and reacts with an effort (force) as an output. Ideally, the systems controlled by this method should have low inertia and friction, because if these forces are not properly compensated, they will be fully felt by the user. On the other hand, admittance control is employed when the system is capable of receiving the force as input and imposes a displacement as an output (Grafakos *et al.*, 2016). The SpaceMouse environment in our case study accepts the human forces on the 6-axis of the mouse cap and gives deflections in position, which in turn can be transformed into velocities to be sent to Meca500 robot in velocity mode; therefore admittance control is a good candidate in our case. The idea behind admittance/impedance control is to introduce mathematically new dynamics (mass, friction and stiffness coefficients) in order to shape the the closed loop system dynamics. The detailed analysis behind this concept can be found in (Hogan, 1987). Generally, impedance control is better when the environment is rigid (i.e., high stiffness environment), it acts as a spring in this case but it does so with low accuracy. On the other hand, the admittance control approach is preferable when the environment is flexible (i.e., low stiffness) (Zhao, 2015).

Since the human interaction in our case is usually carried out with low stiffness while applying the required forces and torques on the SpaceMouse cap to steer the robot's end-effector, admittance control is used to map the forces/torques which are typically measured by the 3D SpaceMouse Module to the required robot's positions. The SpaceMouse deflections of each axis are firstly read by the PC, then the required exchange force/torque related to each axis are calculated. Once the latter is known, the user's hand-guiding manipulation can be integrated into the desired admittance behavior, then the desired robot's positions are generated. To explain more the admittance control principle, we firstly define the exchange work between two bodies as the scalar product between the force and the displacement (Lahr *et al.*, 2016) which can be presented by the following equation:

$$dW = F \bullet dX \tag{5.4}$$

where dW is the exchange work, dX is the mouse feedback position error.

If we suppose that the force is zero, then the exchange work is null and the system can be controlled by purely position control, however, in practice, this is not applicable and many applications have an existing work exchange, resulting in the need for force control as well. Since the robot is controlled in velocity mode, we are only capable of reading the SpaceMouse interaction forces by using equation 5.4. The exchange work of each axis which is defined as the percentage of force related to the SpaceMouse deflection is computed via multiplying the maximum actuation force/torque stated in the SpaceMouse manufacturer data-sheet in (3dconnexion, 2020), i.e., vertical actuation force of 11 N, horizontal actuation force of 7.4 N, and torque of 171 Nmm, by the mouse desired position, i.e., in our case the FLC output since the admittance is connected in series after the FLC controller, as presented in Figure 5.2. For instance, in the case when the robot is moving in a vertical direction, the current exchange work (i.e., the related force) is as:

$$dW_H = 11\ dX_H \tag{5.5}$$

where dW_H is the exchange work (admittance control input) in the vertical direction, dX_H is the FLC mouse required position in the vertical direction.

At this point, the admittance control is introduced as a second order dynamic system, where the system parameters have to be identified by the control designer, i.e., the spring "stiffness" (K), inertia (M), and damper "energy dissipation" (B) coefficients of a simplified mechanical model as illustrated in Figure 5.6a. Since admittance needs the interaction force as an input and displacement as an output, the transfer function can be written as:

$$\frac{X}{F}(s) = \frac{1}{Ms^2 + Bs + K} \tag{5.6}$$

where $X(s)$ is the displacement, and $F(s)$ is the interaction force, both are in Laplace domain, M is the virtual inertia component, B is the virtual damper component, and K is the virtual stiffness of the equivalent system representing human-robot interaction.

For implementation purposes, a discretized version of the admittance controller equations is needed. In (Lahr *et al.*, 2016), two methods of implementation are introduced, the integration and the discretization methods as laid out in Figures 5.6b, and 5.6c. Both methods present a very good response, however, in (Lahr *et al.*, 2016), the sources of uncertainties such as compliance in the joints, in the force sensor, and in the environment were neglected in their presented model. We chose to use the discretization method because it is easier to implement, in addition, it takes into account the sources of uncertainties which are very present in our case (see perturbation zoomed-in Figure 3.25) as well as different oscillations which cannot be neglected [2].

For discretization purposes, Tustin's approximation was used to go from Laplace domain to Z domain as in (Lahr *et al.*, 2016), i.e we substitute s in equation 5.6 by the following:

$$s = \frac{2}{T_s}\frac{z-1}{z+1} \tag{5.7}$$

where T_s is the sampling time.

[2] The model uncertainties and unknown environment are inevitable difficulties for admittance/impedance control. To reduce the negative impacts from these problems, many researchers combine admittance/impedance control with some advanced control techniques such as robust, adaptive, and learning control techniques (Song, Yu & Zhang, 2017). In our case, we used a (TD) in series with the admittance control in the implementation phase to overcome these challenges, thus, the sources of uncertainties are included then rejected via the TD proposed controller in this thesis.

The full deduction of the difference equations from Laplace is stated in Appendix of (Lahr *et al.*, 2016). Here, the final formula is presented for the open-loop block diagram in Figure 5.6b. The values of M, B, and K are first estimated using initial experimental tests and tuning by trial and error and then kept constants. The only line to be implemented is equation (5.8).

$$\begin{aligned} x(k) = [&T_s^2 f(k) + 2T_s^2 f(k-1) + T_s^2 f(k-2) - (2KT_s^2 - 8M)x(k-1) \\ &-(4M - 2BT_s + KT_s^2)x(k-2)] * 1/(4M + 2BT_s + KT_s^2) \end{aligned} \tag{5.8}$$

The pseudo-code of the discretization method is displayed in Algorithm 5.1, which includes the second-order system gains presented in Figure 5.6a.

Algorithm 5.1 Discretization Method Algorithm
Taken from Lahr *et al.* (2016)

```
$M \leftarrow user\ defined\ inertia$;
$B \leftarrow user\ defined\ damper$;
$K \leftarrow user\ defined\ stiffness$;
$x(0) \leftarrow x_0$;
$f(0) \leftarrow f_0$;
$loop$ :;
$f(k) \leftarrow updated\ force\ value\ from\ sensor$;
$x(k) = [T_s^2 f(k) + 2T_s^2 f(k-1) + T^2 f(k-2) - (2KT_s^2 - 8M)x(k-1) - (4M - 2BT_s +$
    $KT_s^2)x(k-2)] * 1/(4M + 2BT_s + KT_s^2)$;
$x(k-2) \leftarrow x(k-1)$;
$x(k-1) \leftarrow x(k)$;
$f(k-2) \leftarrow f(k-1)$;
$f(k-1) \leftarrow f(k)$;
goto $loop$.
```

The role of the admittance control is mainly to damp the oscillations that still exist in some phases during the robot movement even after using fuzzy control. Since these oscillations appear when the robot's operator adjusts the maximum speed of the robot at high values (more than 100 mm/s), the admittance control is employed to take an action typically at two phases, first, when the robot's operator attempts to suddenly change the direction, and secondly, when he/she desires very slow motions (below 50 mm/s).

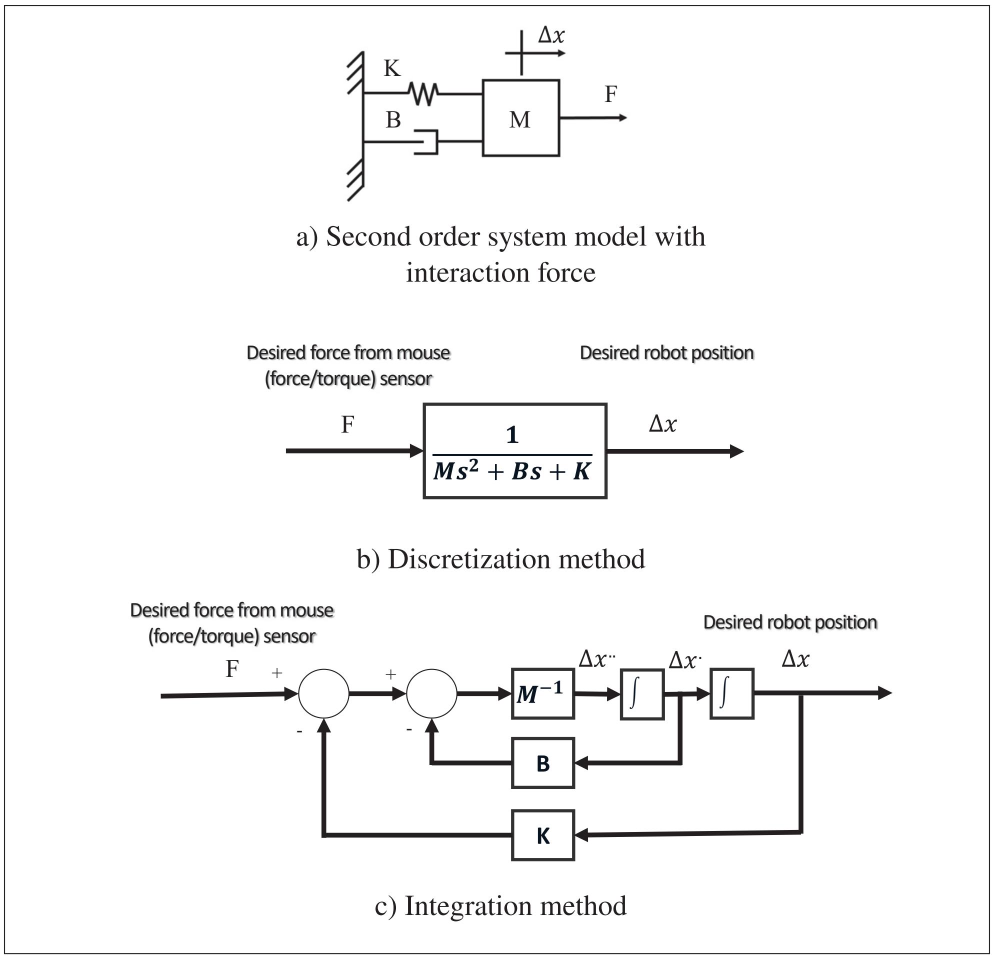

Δx
K
F
B
M
a) Second order system model with interaction force
Desired force from mouse (force/torque) sensor
Desired robot position
F
$\frac{1}{Ms^2 + Bs + K}$
Δx
b) Discretization method
Desired force from mouse (force/torque) sensor
Desired robot position
F
+
+
-
-
M^{-1}
$\Delta\ddot{x}$
$\Delta\dot{x}$
Δx
B
K
c) Integration method

Figure 5.6 Admittance control block diagram
Taken from Lahr *et al.* (2016)

Therefore, in accordance with equation 5.8, we tuned the damping, stiffness, and mass parameters on the basis of the online estimation of the human hand stiffness, to prevent the robot from reaching low performance during human-robot interaction. Usually, high admittance parameters are required when the robot user performs fine movements and vice versa when large accelerations movements are desired. We designed and adjusted the admittance parameters which would help in reducing the virtual mouse desired position when the robot's real velocity is small either in the same or opposite directions in the sense of small interaction force in this case. The most important and sensitive parameter is the damping coefficient which has a major effect on human reception than virtual mass (Grafakos *et al.*, 2016). We do so because while the fuzzy control that precedes the admittance control takes care of the whole robot movement in all cases, it fails to overcome the oscillation generated from the acceleration or deceleration phases if the mouse movement is respectively in the same or opposite direction as the robot real velocity. The admittance overcomes these issues by adding a small delay while the robot is in the acceleration or deceleration phase, in addition, it also attenuates more the small signals in low robot's speeds to give better accuracy and precision.

5.4 ADRC tracking differentiator (TD)

In this section, the tracking differentiator (TD), which represents the first part of the ADRC controller as described in subsection 4.2.1 is fully employed to filter the noise generated by the admittance control. In addition, TD is exploited to delay the robot movement, typically, when the robot operator attempts to move from one direction to its opposite at high speed, while not affecting the robot's movement at low speeds. The filtering factor h_0 in equation 4.1 is automatically tuned based on the robot's current speed and movement as compared to mouse deflection. TD has a vital function giving its input signal a relevant transient response and provides a robust noise-free differential signal. To recapitulate, TD has been assigned to do four tasks:

- Filtering the admittance output noise including the robot motor perturbations.
- Adding a specific delay once the desired movement transfers from high to low deflection or vice versa.
- Automatically tuning the filtering factor h_0 value related to the mouse-robot feedback error.
- Converting the input mouse position (deflection) into velocity output to be sent to the robot's controller in a velocity mode.

Following equations 4.1 and 4.2, the reference signal $v_1(t)$ (admittance noisy output signal) is differentiated to be $v_2(t)$ by Han's function ($fhan$). The main two parameters influencing $v_2(t)$ are r; the convergence speed of the reference signal and h_0; the filtering factor. In our case, we seek to filter all the signal noises including the robot's motors perturbations zoomed-in Figure 3.25. To do so, the filtering factor has to be adjusted to achieve the required performance.

Note that increasing h_0 leads to an increase in signal time delay without changing its amplitude. Analyzing the human-robot interaction, it can be observed that when the user is moving steadily in any direction at low speed, there is no need to add any delay, the mouse-end-effector follows exactly the reference signal with no oscillation nor bouncing around. Adding a delay is much needed, however, at high speeds, for example when the user decides to suddenly change the direction in a short time. The robot, in this case, is leading the desired mouse position and may bounce around this position resulting in an oscillation. The delay helps relatively slow down the robot while reaching the appropriate high speeds. To achieve this, we perform an online tuning of the filtering factor h_0 by increasing its value to add more delay when the robot starts the acceleration or deceleration phase and decreasing its value to the minimum when the robot is moving at low speed. The concept of this tuning operation is based on defining the filtering factors at linear and angular velocities $h_{0_{t_L}}$ and $h_{0_{t_A}}$ as power functions of the mouse deflection error, see equation 5.11 and equation 5.12 below where T_s is the sampling time, and d_m is the mouse deflection error. We can then change the delay by controlling the exponents 5.9 and 5.10 on the basis of the amount of mouse deflection (error) and the maximum desired speed selected by the user, same as controlling the exponent n in the function x^n as shown in Figure 5.7. We then determine the amount of the exponent value by reading the user-selected maximum linear and angular velocities, then dividing them by the manufacturer maximum robot's velocities, i.e., the linear velocity of 1000 mm/s and angular velocity of 300 mm/s (see robot documentation in (Mecademic, 2020)). The result is powered to 10 (the maximum power value we need) in case of linear velocity and to 4 in case of angular velocity, then rounded to the nearest decimal value as in the following equations 5.9 and 5.10

$$G_L = round(V_{L_{max}}/1000)^{10}, \tag{5.9}$$

$$G_A = round(V_{A_{max}}/300)^{4}, \tag{5.10}$$

$$h_{0_{t_L}} = T_s\ |d_m|^{G_L}, \tag{5.11}$$

$$h_{0_{t_A}} = T_s\ |d_m|^{G_A}. \tag{5.12}$$

where G_L, G_A are the power of the mouse deflection error at robot linear and angular velocity, respectively, $V_{L_{max}}$, $V_{A_{max}}$ are the maximum selected linear and angular robot's velocity by the user.

Note: The oscillation and the noise in the case of linear robot's speeds are higher than those of angular speeds, therefore, the power of the filtering factor (G_L) is higher for linear speeds than the one for angular speeds (G_A).

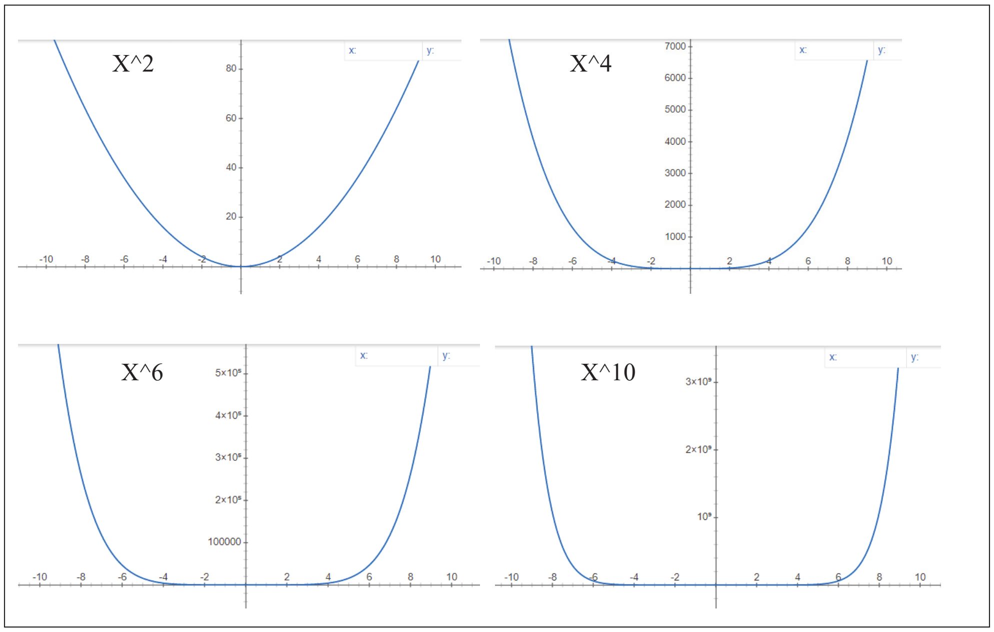


Figure 5.7 Influences of different power for variable X

5.5 Experimental results of the proposed multi-stage controller

The unique challenge in developing the hand-guiding algorithm for this hardware combination comes from the high-flexibility of the 6-axis sensor, which reduces to a minimum coupling between the robot and the hand of the operator. To cope with this flexibility, a unique advanced control technique is put forward. The objective is to make the robot track the mouse manipulation by humans easily without any aggressive motion, noise, mutual motion difficulties, or oscillations. To evaluate the performance of the proposed controller, multiple experiments have been performed, involving both fine path following and rapid displacements. One of these experiments has been filmed and presented. In addition, the response from the 6-axis sensor is shown for the system both before and after the implementation of the proposed controller. The usability of the robot-mouse system can be established by analyzing the performance of mouse-robot errors while the user is manipulating the robot and interacting with the system, in particular by plotting the errors and recording a video to note the cues of easy robot manipulating in high and low speeds, and when performing a precise task. The robot is tested by following the contour of specific shapes surfaces and moving very precisely, in applications such as inputting silicon or polishing. The purpose is to ensure that user interaction is more effective, efficient, comfortable, and satisfactory.

As discussed in the sections above, the proposed controller combines three controllers connected in series, which are the fuzzy logic control, the admittance control, and the TD part of the ADRC control. We first implement the fuzzy logic control because a human can describe the mutual system naturally better than creating a mathematical model, thanks to fuzzy rules which incorporate human knowledge. Although the implemented fuzzy controller does a good job for the robot response to mutual motion (will be described in detail later), it was not able to overcome the oscillation problem especially with high robot's speeds as seen in Figures 5.8, 5.9, and 5.10. Hence, we use an admittance control in series with the fuzzy control to overcome this issue, i.e., the fuzzy output is the admittance input. The admittance control, in turn, damps the oscillations but brings noise to the signal as presented in Figures 5.11, 5.12, and 5.13, which needs to be filtered. Here, the idea of using the ADRC tracking differentiator (TD), which was intentionally neglected in the proposed controller (ADRC in EBF) of the first objective (see the end-note of subsection 4.2.1), is pursued. The TD has the capability not only to filter the noise but also to add an intended delay to the robot's motion especially at very high robot's speeds. In this context, the TD is connected in series with the admittance control, i.e., the admittance control output is the TD input signal. The mouse-robot system including the proposed SISO controllers block diagram is shown in Figure 5.2.

To address the robot-mouse tracking mutual performance and to cover all the possible cases, two approaches are used:

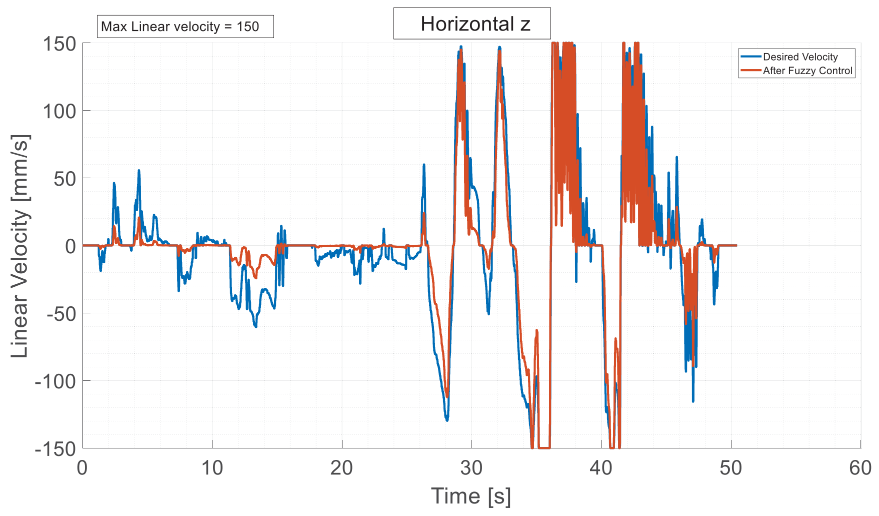
1. **In the first approach,** we recorded the mouse deviation values (error signal) during the robot movement for different linear and angular velocities which shows the signal real-time oscillations, noise, and delay during the robot's motion. The results are presented in a specific sequence to cover all the cases: First, before (in blue color) and after each stage individually (in red after fuzzy and in yellow after admittance control) , then after the implementation of the proposed multi-stage controllers, again with the response in blue for the case before any controller and in purple for the response of the system with the multi-stage controller. The individual responses (red and yellow) are also plotted on the same graphs for reference.
   a. Before and after each controller individually
      - The real mouse error signal without any controller in blue color.
      - Fuzzy control in red color.
      - Admittance control in yellow color.
   b. After multi-stage controller's implementation, in which the robot motion is controlled by the multi-stage controller's output in purple color, and the results of other controllers are just plotted as a reference as:
      - The real mouse error signal without any controller in blue color.
      - The output of the fuzzy controller after controlling the real mouse error signal in red color.
      - The output of the admittance controller in series after the fuzzy controller in yellow.
      - The output of the TD controller in series after the admittance controller in purple.
2. **In the second approach,** we filmed a video to present the visual mouse to robot movement at the robot's linear and angular velocities of 250 mm/s and 73°/s, respectively. The video is displaying the user while trying to move very slowly then very fast in the robot's workspace in all mouse directions as mentioned in Figure 3.21, in addition, it is shown how the user is capable of moving very slowly and precisely while trying to move on the circumferences of a circle, square, ellipse, and middle of four different cross shapes, as can be seen in this video: .https://www.youtube.com/watch?v=-jRZKi8_Obg&feature=youtu.be.

5.5.1 Outcomes of implementing each controller individually

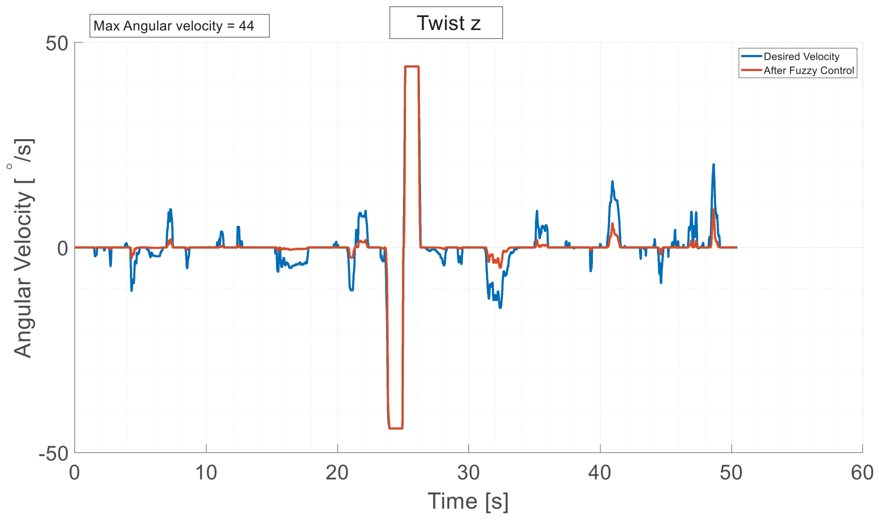
5.5.1.1 Fuzzy Control Outcomes

After many mouse-robot manipulation experiments, we discovered that the oscillation starts to appear after the robot's linear speed of 100 mm/s as mentioned in the mouse control challenges in subsection 3.6.2. This is due to the fact that when the user adjusts the robot's maximum speed at a value higher than 100 mm/s (for instance, 150 mm/s), then when the user attempts to manipulate the mouse with a very small deflection in ranges from 0 to ± 0.4 for example, and in different directions in a very short time, the robot moves with a linear velocity that is proportional to the user-selected maximum speed, say in ranges from 0 to ± 60 mm/s for this example (see Figure 3.24 and more explanations in subsection 3.6.1). Therefore, the robot moves faster in this case and leads the desired mouse deflection at a certain time, however, when the user tries to move in the same direction for a period of time without any change in direction, the robot responds very well.

Hence, we decided to design a fuzzy controller to sense the mouse-robot error and its rate of change, then to take the right control decision by reducing the desired robot's speed to lower speeds than the anticipated real ones in the case when there are multiple direction changes in the error and its rate of change while keeping the same desired robot's high speed if the robot mouse deflection is steady and moving in the same direction. To better describe the fuzzy role, Figure 5.8b represents the robot's desired angular velocity of 44 °/s performance while the user twists the mouse in z-direction after implementing only the fuzzy control. As can be seen, the blue plot represents the mouse real deflection during the robot movement; while, the red plot displays the fuzzy output; here, we can visualize that the red graph attenuates the blue graph signal at low speeds starting from time 0 to around 22 s, nonetheless, from time 22 s to 27 s, the red graph is following exactly the blue graph, because, at this time, the user is moving in the same direction and for a longer period of time, and exerting a high rate of velocity change reaching the selected maximum speed; thereafter, the red graph is again attenuated in low speeds. Although the fuzzy control did a great job in the sense of the introductory required task, the oscillation appears again with higher robot's speeds, for instance, when the maximum speed is 150 mm/s and at time 28 s to the end of the motion, the oscillation can be seen as laid out in Figure 5.8a. Generally, fuzzy performance is better in mouse rotation desired motions (twisting and tilting) as shown in Figure 5.10, than the translation ones in Figure 5.9.

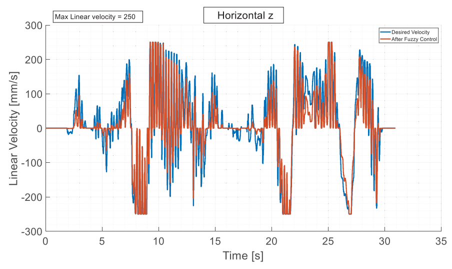


a) Desired linear velocity of maximum 150 mm/s

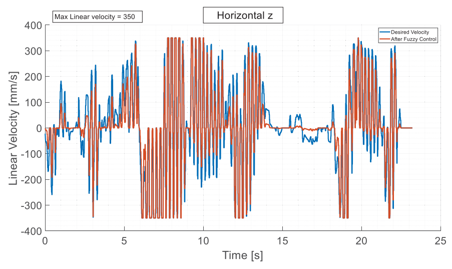


b) Desired angular velocity of maximum 44°/s

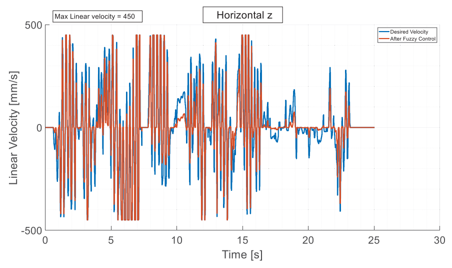
Figure 5.8 Robot's performance at desired maximum linear and angular velocities while moving in horizontal z-direction, and twisting in z-direction, respectively, both after adding only the fuzzy control



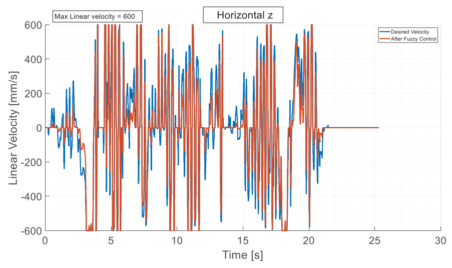
a) Desired linear velocity of Max. 250 mm/s



b) Desired linear velocity of Max. 350 mm/s

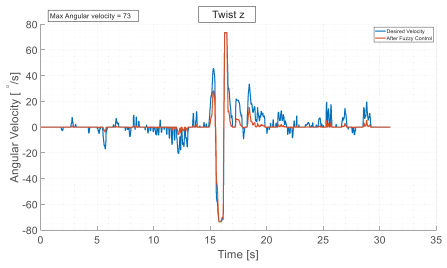


c) Desired linear velocity of Max. 450 mm/s

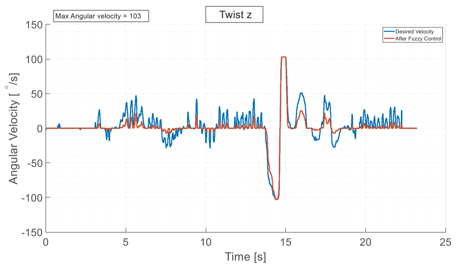


d) Desired linear velocity of Max. 600 mm/s

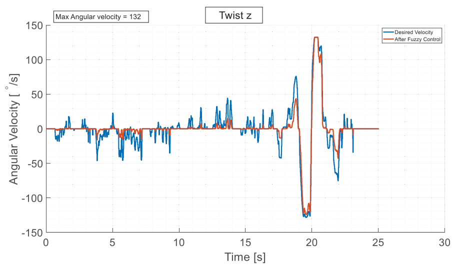
Figure 5.9 Robot's performance at different linear velocities while moving in horizontal z-direction after adding only the fuzzy control



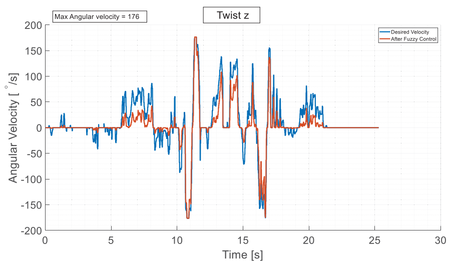
a) Desired angular velocity of Max. 73°/s



b) Desired angular velocity of Max. 103°/s



c) Desired angular velocity of Max. 132°/s



d) Desired angular velocity of Max. 176°/s

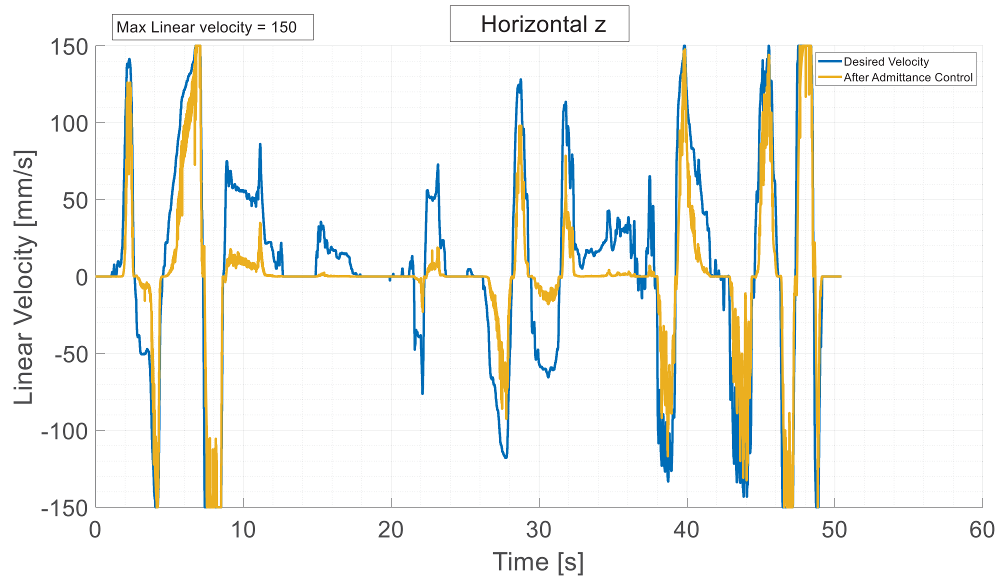
Figure 5.10 Robot's performance at different angular velocities while twisting in z-direction after adding only the fuzzy control

5.5.1.2 Admittance Control Outcomes

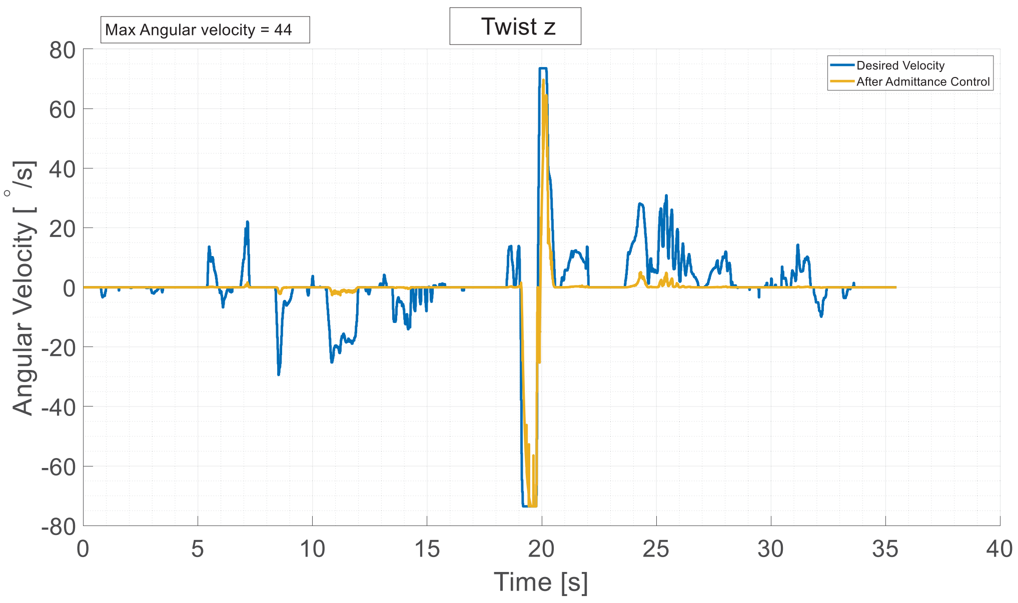
The main purpose of using the admittance control in this stage is to better control the acceleration and deceleration phases by appending a small delay to the signal before the robot reaching the maximum speeds, and to attenuate the signal more than what fuzzy control does at the robot's low speeds. Although the admittance control enhances the performance and removes the oscillation, it unfortunately added noises to the signal. The small delay can hardly be seen in the graphs except in the zoomed sections, but at the same time, it practically enhances the robot motion than using fuzzy control alone. In this sub-subsection we are only concerned with the outcomes of the admittance control when it is applied only without any other controller, however, to better see the plot details of the intended small delay, the desired attenuation, and the unwanted noise, we have to observe the admittance after it is connected in series with the fuzzy control. Figure 5.15 is added to distinguish between the same signal under fuzzy control only in red plots of Figure 5.15a and after adding the admittance control in yellow plots of Figure 5.15b. The zoom parts A and B of the latter figure show the intended small delay added when the robot accelerates and decelerates reaching maximum velocities, respectively, in addition, the unwanted noise can be seen in the same graph. The zoom part C represents the actual mouse deflection in blue color while it is passing through two attenuation phases, first with fuzzy control in the red signal, and finally with admittance control in the yellow signal. Generally speaking, the admittance control did a great job towards the required attenuation and the intended small delay, however, at the same time, it added unwanted noise to the signal which is needed to be filtered.

All the aforementioned figures are showing the robot under both fuzzy and admittance controllers, thus, in order to prove that the robot's performance under only the admittance control is not enough and to show how the oscillations occur, we have to see the performance under only the admittance control. Figures 5.11, 5.12, and 5.13 are showing that applying only the admittance control will not achieve the desired performance because of the multi-oscillation at high robot's speeds and the noise generated at each part of the signal. Therefore, the TD part from ADRC controller is introduced after to be connected in series to filter the noise and handle the remaining existing oscillations.

Note: At this point, an important question floats to the surface, why in our admittance control design we did not tune all the parameters together, i.e., the mass, spring, and damper, to have better performance by applying only the admittance control instead of employing the proposed multi-stage controller in SISO technique. To answer this question, we have to glance the different robot's motion phases stated in section 5.1. The answer concludes that the admittance might be helpful when the robot is heavy in weight with large inertia and significant friction in which the motion is very slow when the high precision is not considered, in which the parameters can be easily tuned. However, it will not be applicable with Meca500 (small robot), particularly at the robot's high speeds that are selected by the robot's user in this second objective. Delving into details, if we decide to increase the damper to help the robot's user to perform tiny and fine movements, with smoother and limited response, the virtual mass should not be too low, despite that, in practice, the mass is too heavy. Even so, if it is succeeded to find the appropriate tune parameter, it will not be comfortable for the user during the manipulation especially when he decides to suddenly move in large movement which might be too rigid in this case to freely move. On the contrary, decreasing the damper at high speeds is causing high oscillation and bouncing around the same point even if the controller designer attempts to increase the spring parameter, because it will be so hard to stop the movement at a certain location, and will cause bouncing around this position due to the spring effect. For instance, see Figure 5.11 at low linear and angular speeds of 150 mm/s and 44°/s, respectively, and compare it with Figures 5.12 and 5.13, the performance is better in lower speed except for noise existence compared to what happen in high speeds which have higher oscillations and relative aggressive motion. There is a proposed solution for this issue in some papers which proposing to use what is called a variable admittance control, for instance, such as presented in (Lecours, Mayer-St-Onge & Gosselin, 2012), though the intelligent assist device (IAD) used in this application has large inertia and significant friction, see Figure 5.14, which is not applicable in our case.

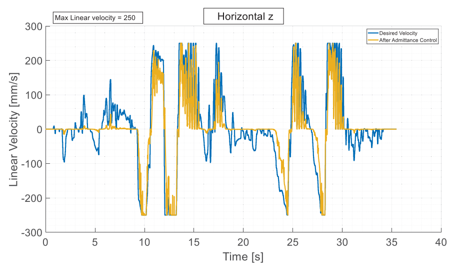


a) Desired linear velocity of Max. 150 mm/s

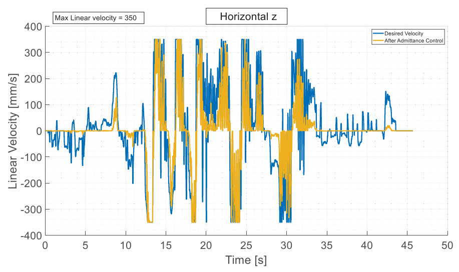


b) Desired angular velocity of Max. 30°/s

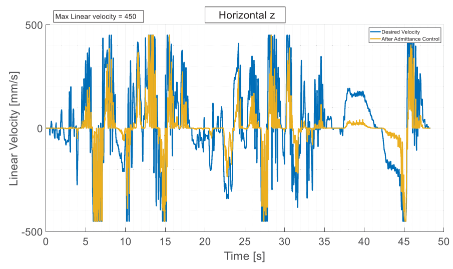
Figure 5.11 Robot's performance at desired maximum linear and angular velocities while moving in horizontal z-direction, and twisting in z-direction, respectively, both after adding only the admittance control



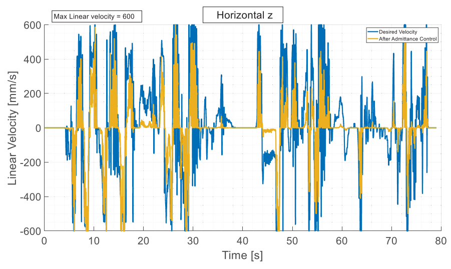
a) Desired linear velocity of Max. 250 mm/s



b) Desired linear velocity of Max. 350 mm/s

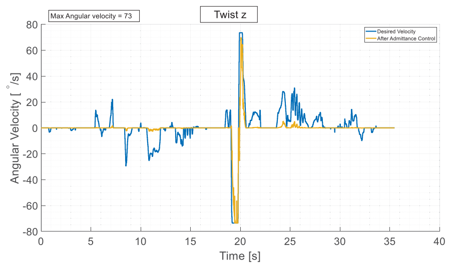


c) Desired linear velocity of Max. 450 mm/s

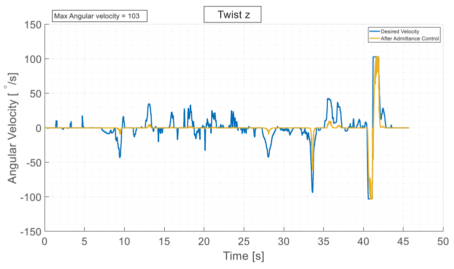


d) Desired linear velocity of Max. 600 mm/s

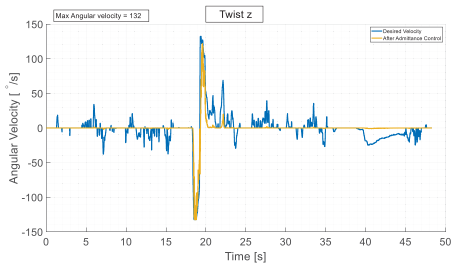
Figure 5.12 Robot's desired velocity performance while moving in horizontal z-direction after adding only the admittance control



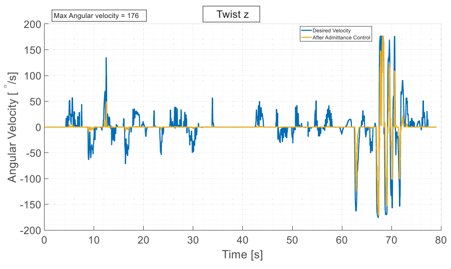
a) Desired angular velocity of Max. 73°/s



b) Desired angular velocity of Max. 103°/s



c) Desired angular velocity of Max. 132°/s



d) Desired angular velocity of Max. 176°/s

Figure 5.13 Robot's desired velocity performance while twisting in z-direction after adding only the admittance control

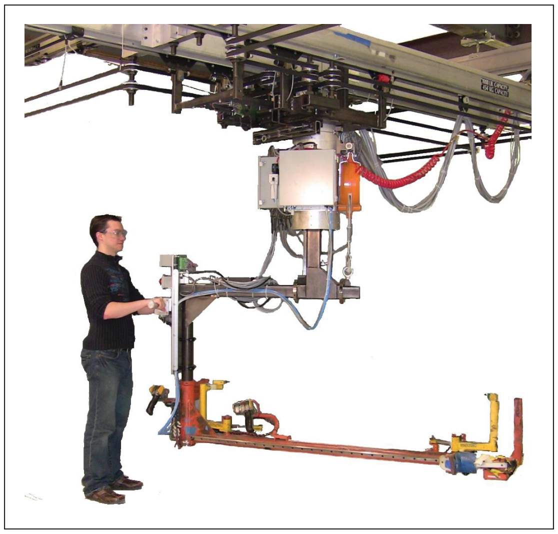
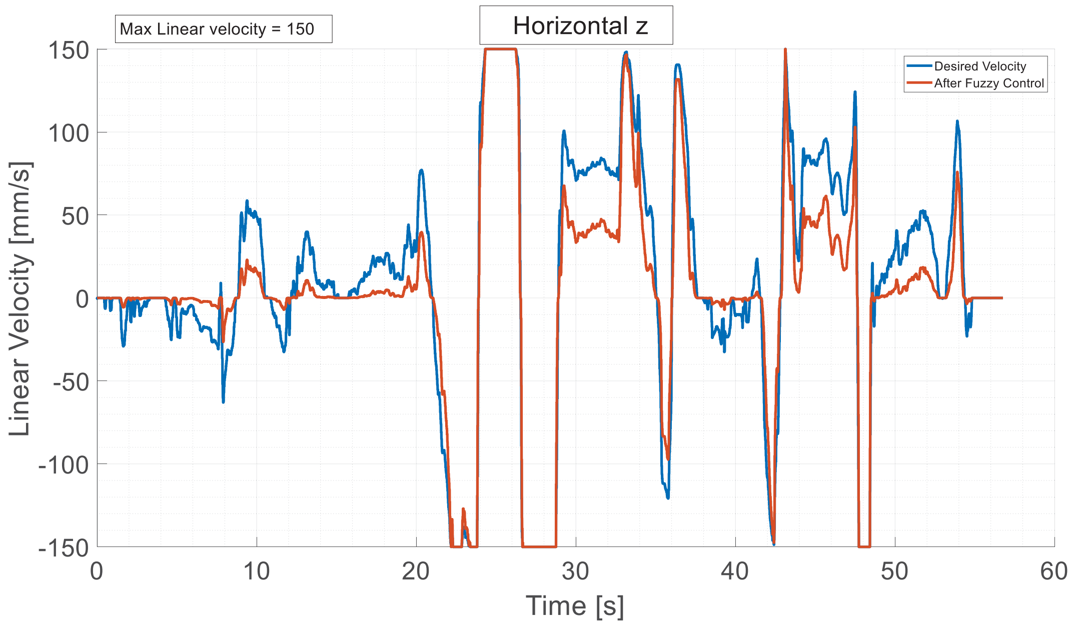


Figure 5.14 Prototype of the IAD
Taken from Lecours *et al.* (2012)

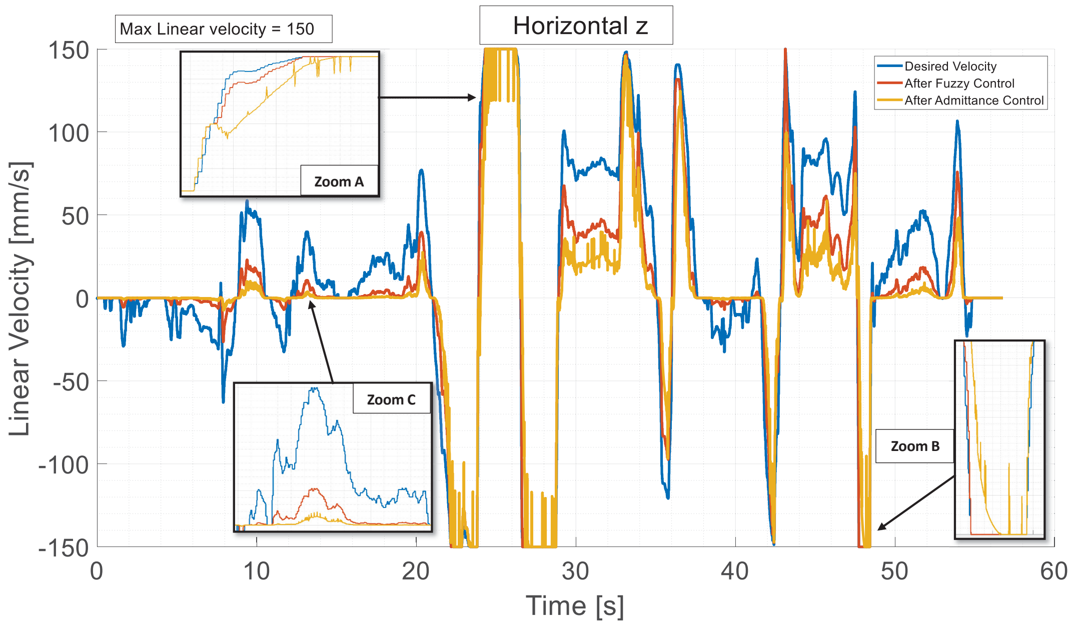
5.5.2 The outcomes after implementing the multi-stage controller

After all the aforementioned discussion, here in this sub-subsection, the robot performance under the proposed multi-stage controller shown in Figure 5.2 is presented. The controller output is plotted in purple color and has the legend (After TD Control) because this is the output after the last stage of the series of controllers. Although the robot performance is plotted only under the multi-stage controllers, the error (real desired signal) and the other two controllers' outputs are also plotted. Starting from the real desired signal (before controller), it can be observed that the plots in blue color at different robot's linear and angular velocities in Figures 5.17, 5.18, and 5.19 have very low oscillations compared to those in Figures 3.26, 3.27, and 3.28. This highlights the good effect of this controller on the global error. Then the plots in red and yellow of the latter figures demonstrate the performance after the fuzzy and admittance control, respectively, while as mentioned above, the plots in purple represent the performance after the final stage of the TD controller. As can be seen, the plots in purple color in Figure 5.17a and 5.18a are showing how the TD output signal filtered the admittance output signal in yellow color, not only this but also added delays at the end of the acceleration and deceleration phases. For instance, the purple color in the latter figure from time 23 s to 29 s represents how the delay is built up then remains steady when it reaches the maximum velocity after a certain time, on the other hand, from time 44 s to time 50 s, the same figure demonstrates the delay building up phase, but this time, it does not remain steady because the user changes the movement much faster. In addition, the amount of this delay is increased when the maximum robot velocity is increased as demonstrated in Figure 5.18b. From time 17 s to 21 s, the signal takes much more time to reach the maximum robot's velocity than the signal in Figure 5.18a, which means that controlling the filtering factors by equations (5.11 and 5.12) are taking effect. The robot perturbations are also filtered and removed (see zoom part of Figure 5.17a). Note that in Figure 5.17b which represents the controller performance for angular velocities, there is a peak at time 37 s where the purple plot is higher (exactly double) than the yellow one. This was done by design to make the hand-guiding more comfortable for the operator in high speeds. The same goes for Figure 5.19a.

To conclude, the multi-stage controller is giving very good results for all the four tasks; (1) filtering the noise including the robot perturbations, (2) adding an intended delay when the motion is at high speed, (3) controlling the amount of noise filtration and the delay based on the desired mouse deflection, and (4) converting the mouse deflections into velocities to be sent to the robot. To address the robot-mouse tracking mutual performance, two main methods are used. In the first method, we recorded the mouse deviation during the robot movement in different robot's linear and angular velocities in which we can see the real-time oscillation, delay, and noise during the robot's motion. We agree and acknowledge that the mouse deviation does not purely describe the real robot's movement but the mutual deviation between them, see Note 1 in subsection 3.6.2. However, the graph plots give a very good idea and draw actual mutual response which reflects the real performance. In the second method, we filmed a video to present the visual human to robot interaction at the robot's linear and angular velocities of 250 mm/s and 73°/s, respectively. This video is displaying the user while he is trying to move the robot in its workspace at a very slow speed then suddenly very fast by manipulating the robot in all mouse directions mentioned in Figure 3.21, in addition, the video is showing how the user is capable to move very slow and precise while trying to move at the boundaries and on the circumferences of a circle, square, ellipse, and in the middle of four different cross shapes.

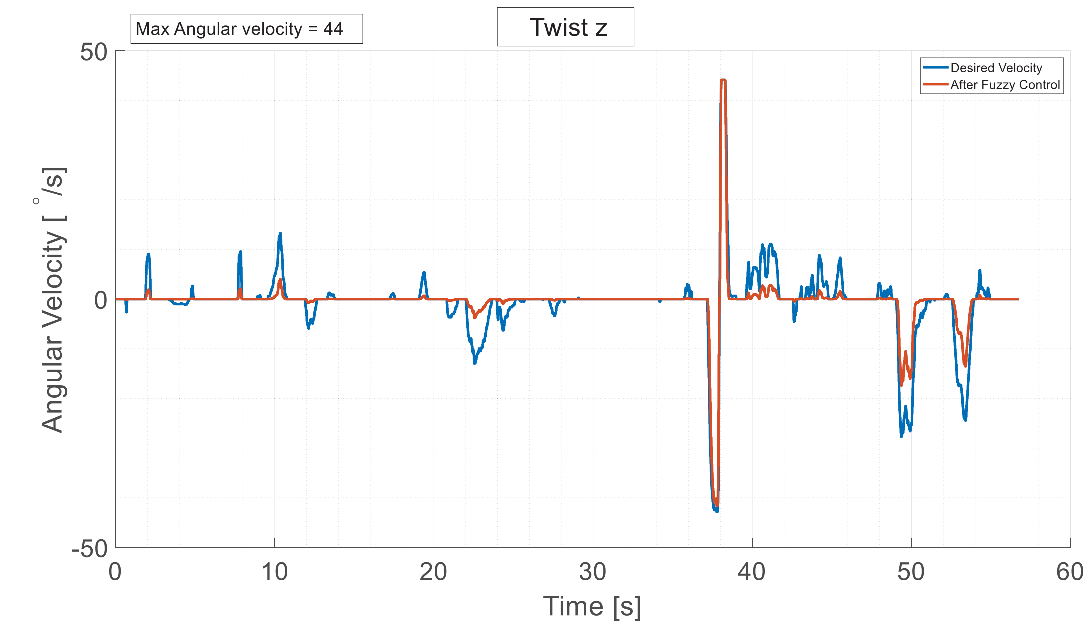


a) After adding the fuzzy control

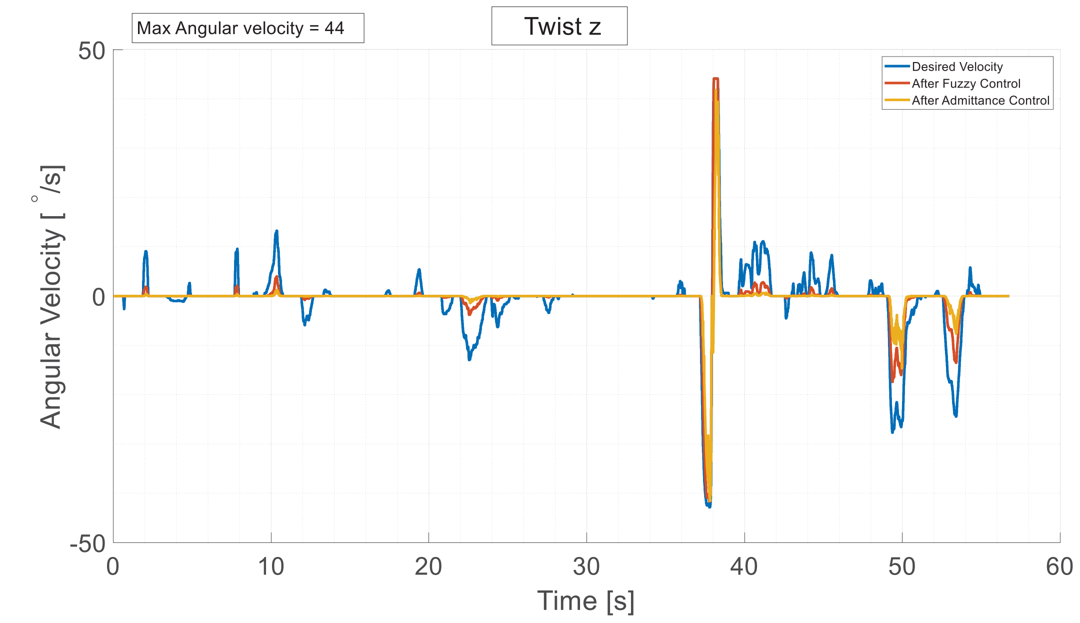


b) After adding the fuzzy and admittance control

Figure 5.15 Robot's performance at a desired linear velocity of maximum 150 mm/s while moving in horizontal z-direction after adding the proposed controllers is series

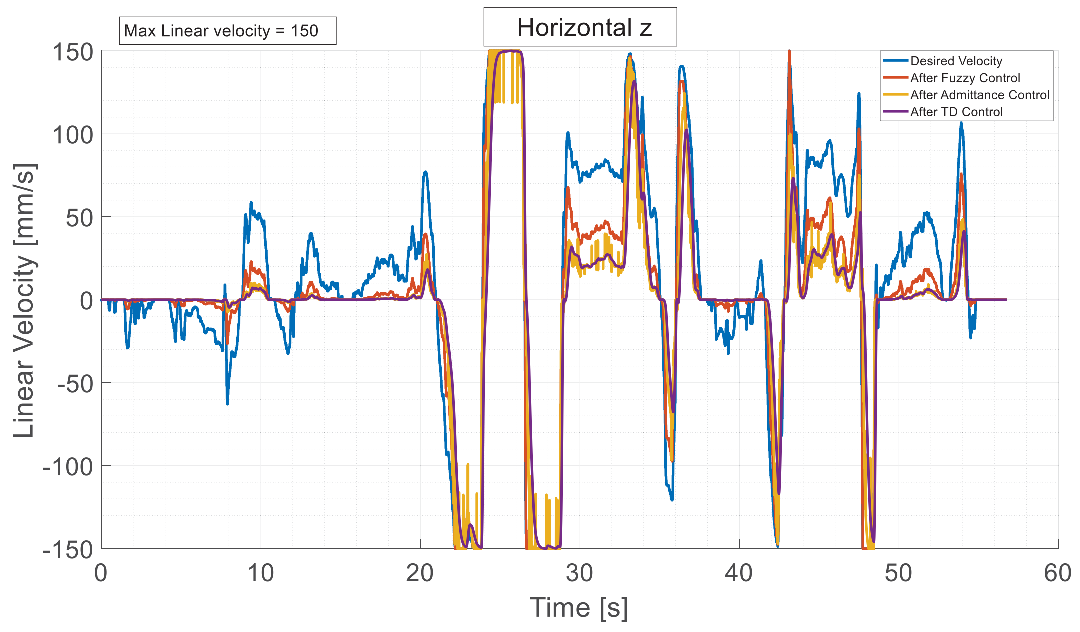


a) After adding the fuzzy control

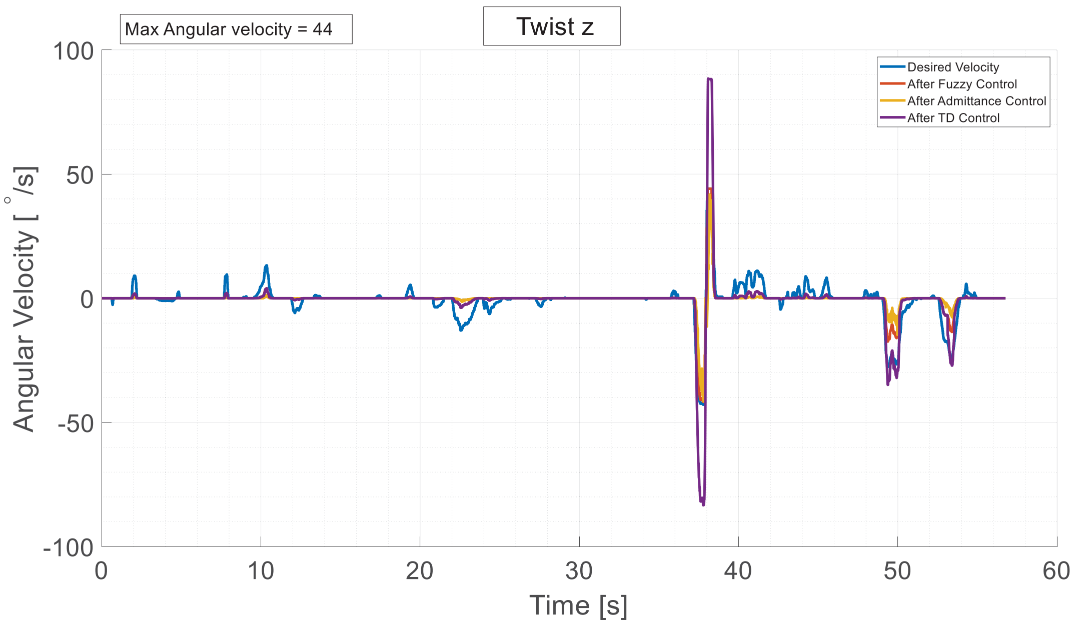


b) After adding the fuzzy and admittance control

Figure 5.16 Robot's performance at a desired angular velocity of maximum 44°/s while twisting in z-direction after adding the proposed controllers is series

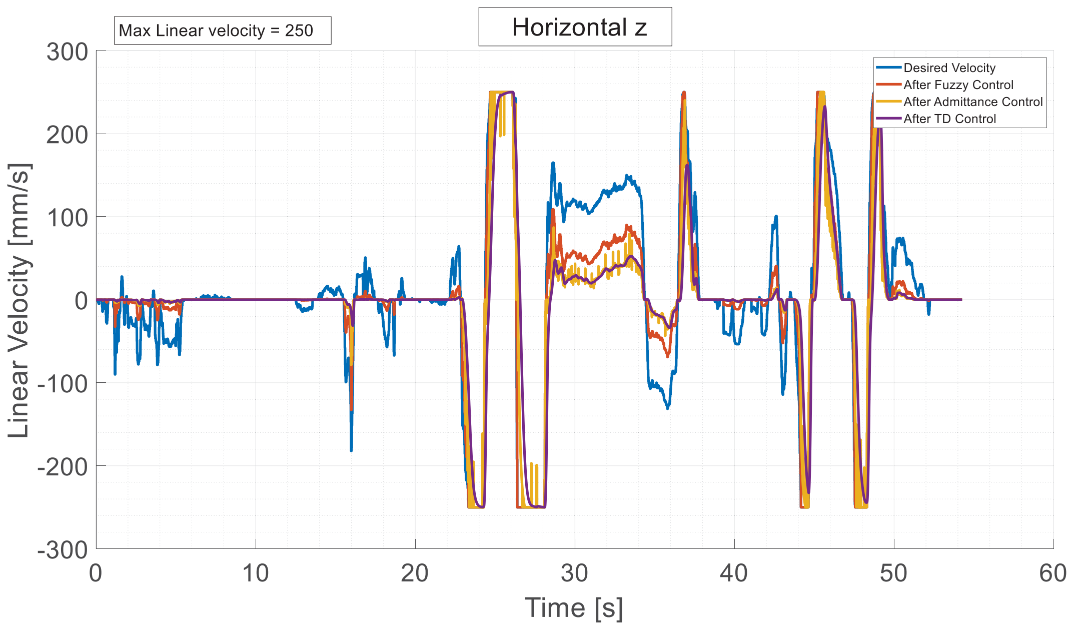


a) Desired linear velocity of maximum 150 mm/s

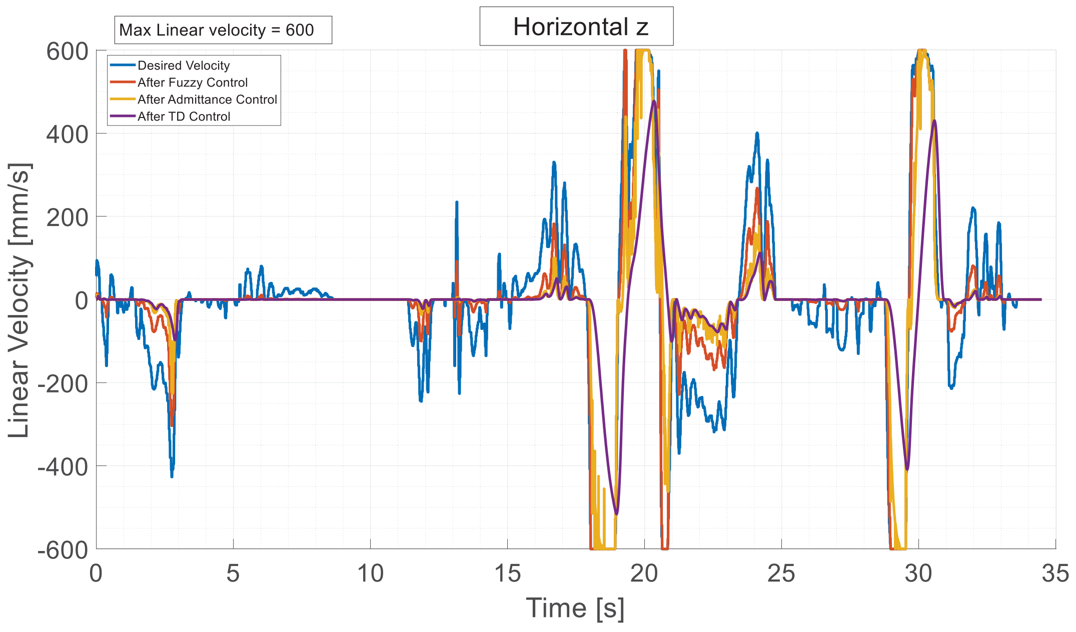


b) Desired angular velocity of maximum 44°/s

Figure 5.17 Robot's performance at desired maximum linear and angular velocities while moving in horizontal z-direction, and twisting in z-direction, respectively, both after adding the fuzzy, admittance, and TD controllers in series

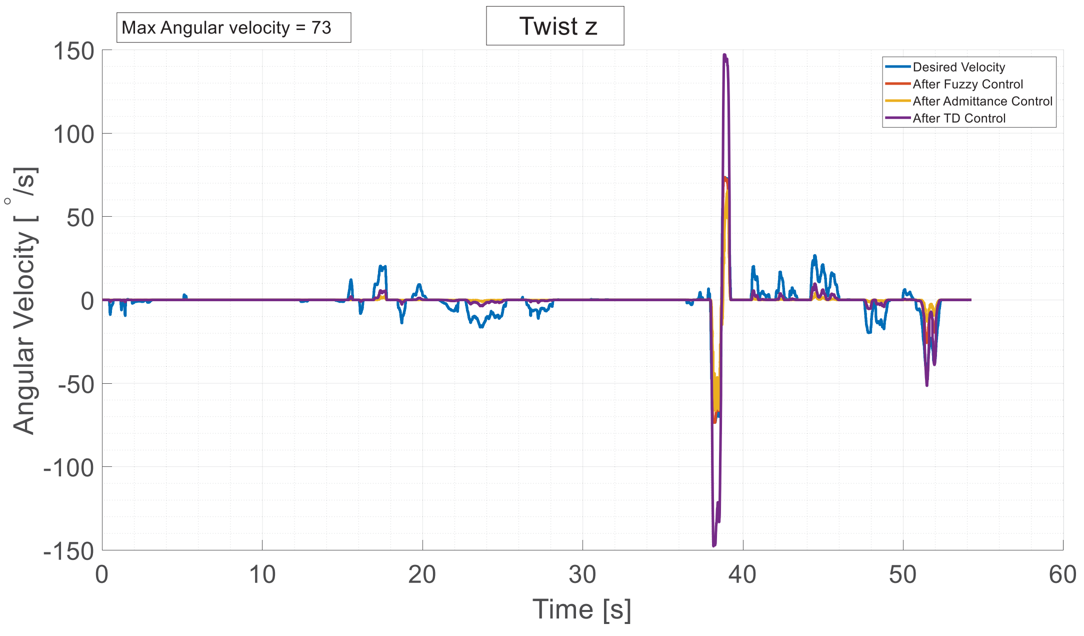


a) Desired linear velocity of maximum 250 mm/s

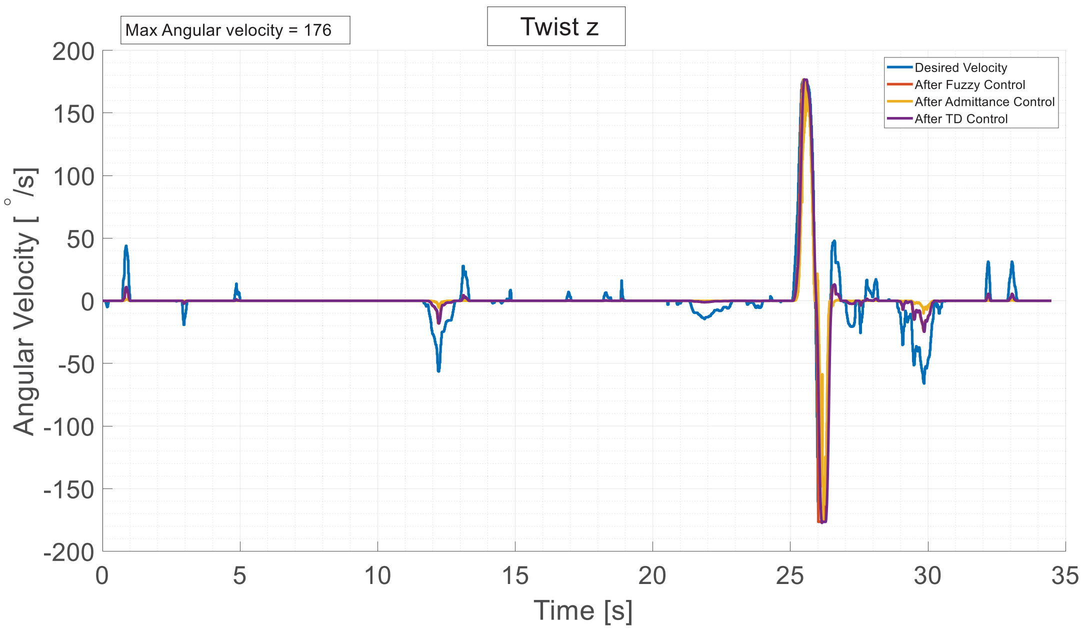


b) Desired linear velocity of maximum 600 mm/s

Figure 5.18 Robot's performance at different desired maximum linear velocities while moving in horizontal z-direction after adding the fuzzy, admittance, and TD controllers in series



a) Desired angular velocity of maximum 73°/s



b) Desired angular velocity of maximum 176°/s

Figure 5.19 Robot's performance at different desired maximum angular velocities while twisting in z-direction after adding the fuzzy, admittance, and TD controllers in series

CONCLUSION AND RECOMMENDATIONS

Conclusion

A novel approach of implementation and evaluation of a practical ADRC scheme was presented to improve the accuracy of a Meca500 industrial robot arm using real-time feedback data from an external distance sensor. The new concept of dealing with the error regardless of both the system order/model and the desired trajectory was proven using the EBF proposed controller. The implementation was done by setting the controller reference input as the desired error with zero values and dealing with the robot error measured by the ballbar in real-time as the controller output, which is being added to the original robot's desired trajectory as a control action to correct robot dynamic path trajectory. The regenerated corrected trajectory developed by the ADRC controller improves the robot's accuracy on-the-fly without any movement interruption.

It can be noticed that before adding an additional step signal, the experimental results validate that the proposed ADRC scheme can improve considerably the robot tracking accuracy. The ADRC controller makes the error converge quickly and with no significant overshoot or important steady-state error. The performance is acknowledged after exerting the additional step signal, in which a temporary error can be seen in low Cartesian Linear velocities for the first few seconds in the output signal with limited overshoot. However, for high TCP velocities, it takes more time for the controller to make the right action and to compensate for the error. This is because the sampling time is not small enough for large TCP velocities. The ADRC performance is also confirmed by comparing the controller's disturbance rejection with that of a well-known SMC with TDE. This work is directly applicable if another IR is chosen while using the same high-accuracy sensor or any other sensor, as long as it supports similar or better transmission rates.

In addition, for the human interaction challenge, a unique advanced technique is put forward for manipulating Meca500 with a SpaceMouse. The novel concept of multi-stage controllers connected in series was verified practically by plotting the mutual response good effect and capturing a video for all types of movements. The multi-stage controllers started with fuzzy control since humans, in this case, can describe the system naturally better than creating a mathematical model, fuzzy rules have the advantages of incorporating such human knowledge. Hence, it gives a more feasible solution featuring human knowledge about the relation between the input and the output at each operating point, instead of trying to accurately model the system. However, the fuzzy controller fails to cover all the human interface cases; consequently, an admittance control is introduced to be connected in series after fuzzy control. Usually, the robot user manipulates the mouse by applying the required forces and torques on it, then it is mapped by the admittance controller to the desired control displacement. This controller enhanced human interaction and robot performance but added some noise. At this level, the ADRC tracking differentiator which has been neglected in the first objective was been fully incorporated to filter the noise, add an intended delay, and transform the position signal into velocity to be sent to the robot as an input in the robot's velocity mode. TD was connected in series after the admittance control.

The experimental results assured the enhanced robot performance after adding the external multi-stage controllers while supporting mouse on the robot's TCP. The controller tackles the challenges related to the operator switching between high- and low-speed motions associated with his typical hand-guiding process (slow precise motions when teaching a path, followed by faster motion when teaching intermediate poses). Comparing the mouse-robot mutual response before and after the multi-stage controllers, portrayed a complete vision about the success of removing the unwanted oscillation, noise, perturbations, and the robot's aggressive motions. Moreover, the filmed a video introduced how the different kinds of robot movements are smooth and comfortable, the user had been able to move suddenly from slow to fast motion and vice versa with no evidence of oscillation or difficulties in movement. Furthermore, seeking precision while moving at very low speeds, the video filmed shows how the user succeeded to move on the circumference of a circle, square, ellipse, and cross dot, thanks to the smart and advanced action of the multi-stage controllers.

Future work

In future works, applying different types of other path tracking trajectories with different types of sensors could be included. Moreover, an advanced algorithm could be used to automatically tune the ADRC parameters. In addition, a combination of other techniques with the sliding mode could be considered to overcome the chattering effect. A variable admittance control could also be implemented instead of using constant admittance parameters. Moreover, large numbers of robot operators may be involved to test the robot performance. An ANOVA test could be employed in future experiments.

APPENDIX I

ROBOT IDENTIFICATION IN CIRCULAR MOTION MODE

1. Introduction

The aim of this chapter is to accurately identify Mecademic's Meca500, a very small six-axis industrial robot while performing movements in a circular trajectory with different robot's linear velocities. This robot identification is indispensable for simulation to test the advanced control techniques before starting the implementation phase. The difference between the robot input and output velocity vectors is very small in ranges of micrometers (µm), which makes the identification procedure very hard because of the limited numerical resolution of MATLAB which ultimately performs the computations required for identification. A static feed-forward neural network approach is employed for the purpose of this robot identification. The input and output data are collected during robot different practical experiments, then based on the collected data, different neural models are generated by interchanging between many parameters such as the training optimizer options, number of neurons, performance functions, activation functions, and training algorithms. The best neural generated model is chosen following the data analysis, then its performance is tested with completely new data sets. It has been shown that the best chosen neural network model can accurately represent the robot behavior with acceptable error ranges.

2. Neural Network Modeling

In the scope of using an external controller and for its design purposes, a robot and its controller are required to be mathematically modeled to be used in the closed-loop simulation before moving forward to the implementation phase. Therefore, accurate identification of the robot model is essential for testing the proposed controller algorithms.

Since the robot with its embedded controller dynamics and parameters are unknown to us, a data-driven based approach may offer a good solution for the black box modeling environment.

Neural Networks (NNs) have proved to be a valuable identification tool giving a very good performance in modeling black box systems even when little information of the system is available. NNs are also a good tool for real-time applications because of their high computational speed, they can also be used for multi-input multi-output (MIMO) systems. Thus, they will be useful for modeling 6-DOF IRs in both joint or Cartesian space position mode, which requires six joints or six positions and orientations (pose), and/or modeling the robot in velocity mode, which requires six linear and angular velocity vector commands (which is our case study).

The neural network's fundamental idea is inspired by the human nervous system neurons' behavior and its network of massively parallel data processing formed by these neurons and provides a good approach to deal with the complexity of physics modeling in industrial applications. It is composed of consolidation of various learning transfer functions and neurons with connections including the layouts between them. In the identification process, the NN executes users' explicit tasks by setting the weights of the neurons stored information, and their connections numerically (Chiang, Chang & Chang, 2004). There have many successful applications of NNs for systems modeling. In (Bekey & Goldberg, 1993; Gu Fang & Dissanayake, 1995; Carlevarino, Martinotti, Metta & Sandini, 2000), the authors showed how the NNs represented the relations between system inputs and outputs by weighted connections for robot modeling. Recently, a dynamic model was developed for billiards robot by NNs in (Gao, He, Zhan & Gao, 2016), and for a redundant five joints robot in (Kern Molina, Jamett Dominguez, Urrea Onate & Torres Salamea, 2014).

Although NNs are classified into static and dynamic, we choose to model Mecademic's Meca500 IR with static NN because it is the simplest form with no feedback elements nor network delays. The topology behind this type of NN can be configured using the system's dimension input layer, the number of hidden layers, and the output layers, thus, it is based on feed-forward conversion from input to output layers as shown in Figure I-1.

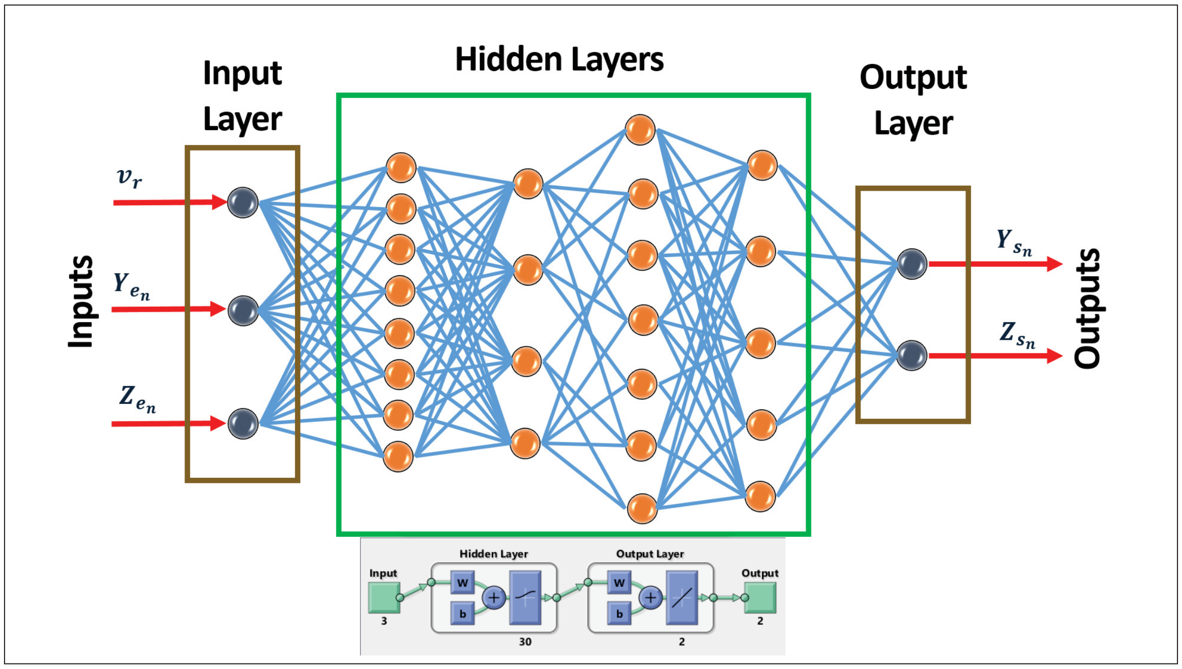


Figure-A I-1 Static Neural Network structure propagation

3. Modeling approach with Neural network

Neural network approach can be designed following specific steps:

1. Data collection.
2. Network configuration, creation and initialization.
3. Network training phase.
4. Network validation phase.
5. Network testing phase.

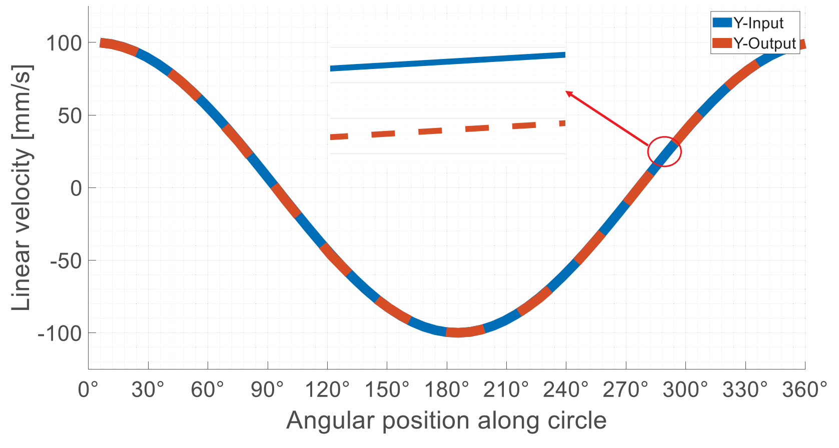
3.1 Collecting data-driven sets

The use of experimental data to and from the robot is necessary to generate good neural models. In other words, different experiments with different robot linear velocities are adopted for data collection. Note that, the number of data samples for each experiment, i.e., the total number of robot steps during the circular motion, is inversely proportional to robot linear velocity.

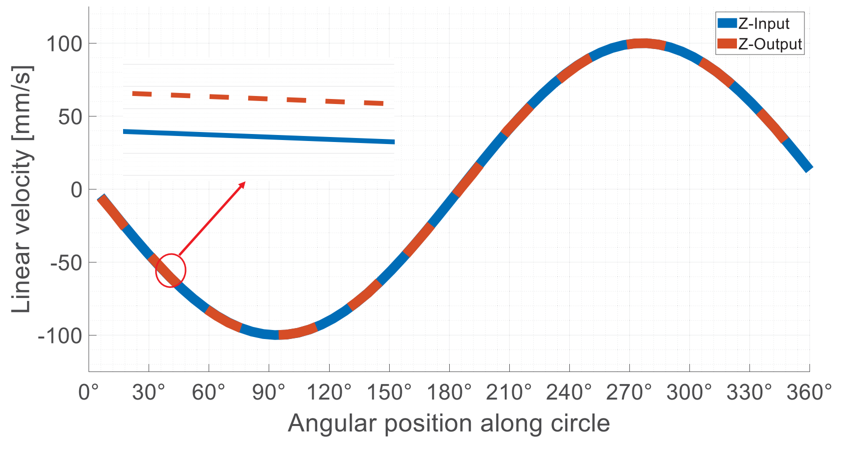
To this end, the NN network in this study is designed to include three input data parameters (Robot different linear velocities υ_r, desired reference TCP's two coordinates Y_{e_n} and Z_{e_n}) and two output data parameters (Real reference TCP's two coordinates Y_{s_n} and Z_{s_n}). All the input and output data gathered from all the practical experiments of different linear velocities have been divided into three sets of data: (1) 75 % (first part for NN training phase), (2) 15 % (intermediate part for NN test phase), and (3) 10 % (final part for validation phase), these phases aim to minimize the modeling error as much as possible. A part of real robot input data (desired velocity vectors in both directions) compared to the output data (real velocity vectors in both directions) at a linear velocity of 100 mm/s is displayed in Figure I-2, while the circular robot error in both directions at this velocity is exposed in Figure 4.5b. Moreover, a block diagram of the method used to collect all the data from the robot during all the experiments is shown in Figure I-3. In addition, Table I-1 illustrates some samples of this data, to give an idea of the values of this data during the different experiments. As mentioned before, the difference between the inputs and the outputs is very small, which is causing a big challenge for the NNs to be able to accurately model the robot system.

Table-A I-1 Samples of robot input and output data with different robot's linear velocities

Input Data			**Output Data**	
Linear Velocity	**Y**	**Z**	**Y**	**Z**
5	5	-0.0125	4.953	-0.6829
5	4.9999	-0.0374	4.9998	-0.0365
5	4.9996	-0.0624	4.9996	-0.0614
.....	.....	....	....	....
10	9.9998	-0.0499	9.9998	-0.0499
10	9.9988	-0.1499	9.9660	-0.8231
10	9.9968	-0.2499	9.9968	-0.2514
.....	.....	.....	.....	.....
100	99.8749	-5	99.842	-5.613
100	98.876	-14.950	98.867	-15.004
.....	.....	.....	.....	.....
150	149.577	-11.250	149.518	-12.006
.....	.....	.....	....	.....
185	184.206	-17.112	184.138	-17.830
.....	.....	.....	.....	.....



(a) In Y direction



(b) In Z direction

Figure-A I-2 Input vs output velocity vectors at linear velocity of 100 mm/s

3.2 Building up the Static Neural Model

In this section, a static feed-forward multi-layer NN is proposed to model the multi-input multi-output (MIMO) robotic system. A mapping with fixed weights based on the robot data following the second method in section 3.5 is carried out to map the relation between robot inputs and outputs. A single hidden layer is used, while the number of both the nodes and the system inputs in the input layer are similar, also, the number of neurons and system outputs in

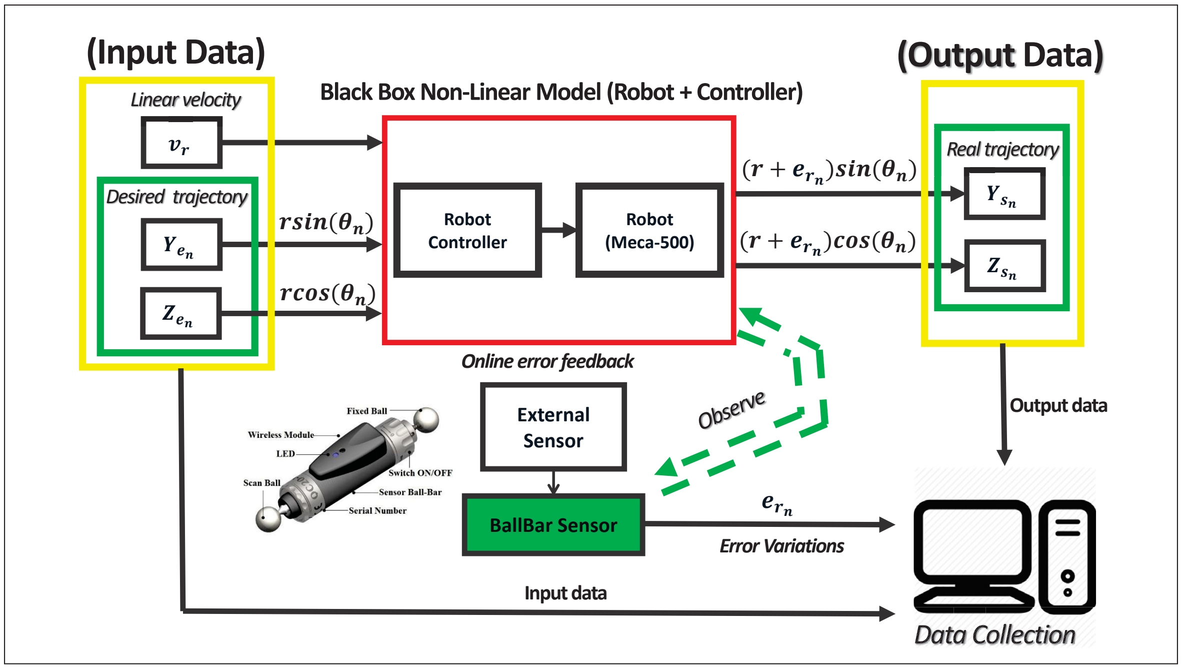


Figure-A I-3 Input/output data block diagram

the output layer are equal as well, as shown in Figure I-10. The number of system inputs in the input layer is assigned to three parameters (υ_r, Y_{e_n}, Z_{e_n}), compared to the number of target outputs parameters of two in the output layer (Y_{s_n},Z_{s_n}). In order to have to determine the best NN model which is the closest to the real robot system, we generate an extensive computer program using MATLAB environment which develops several NN models based on the main framework of MATLAB Deep Learning Toolbox (Ibrahem, Akhrif, Moustapha & Staniszewski, 2019). The program generates a diverse set neural models by changing different parameters including random changing of data sets in each computational cycle as follows:

- Interchanging between the three training Optimizer Options (ADAM, RMSProp and SGDM).
- Starting from one neuron to 36 neurons.
- Swapping between NN performance functions (mae, mse, sae, sse, and crossentropy).
- Switching between two activation functions (tansig and logsig).
- Interpolation of three training algorithms: (trainlm) for Levenberg-Marquardt algorithm, (trainscg) for Scaled conjugate gradient algorithm, and (trainbr) for Bayesian regularization algorithm.

The program is instructed to start with ADAM as a training optimizer option, then to go through changing the other parameters (neurons number, training algorithms, performance, and activation functions) during the training and validation phases. After that, the program will stop to begin analyzing the data of all the generated NN models, this is to dictate the quality of the resultant neural models. Finally, the program will start the same cycle again with the other two training optimizer (RMSProp and SGDM), respectively. All the NN models generated by the three training algorithms including the change of all the parameters during each training cycle are recorded.

Note that the training of each computational cycle is repeated just three times, because repetition more than three, requires a large computational effort while it only grants limited improvements. The performance functions used in the training process determine the stop parameter in the sense of minimizing the function output until reaching its minimum value of 10^{-6}, these functions can be described as follows:

1. Mean Absolute Error Function (mae):

$$mae = \frac{1}{n}\sum_{i=1}^{n} \mid e_{r_i} \mid \tag{A I-1}$$

   where $i \geq 1 \in \mathbb{N}$, e_{r_i} is the error between the actual and the predicted output, is the number of each sample, n is the total number of samples and μ is the mean radius error which is computed by:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} e_{r_i} \tag{A I-2}$$

2. Mean Square Error Function (mse):

$$mse = \frac{1}{n}\sum_{i=1}^{n} {e_{r_i}}^2 \tag{A I-3}$$

3. Sum of Absolute Error Function (sae):

$$sae = \sum_{i=1}^{n} \mid e_{r_i} \mid \tag{A I-4}$$

4. Sum of Squared Error Function (sse):

$$sse = \sum_{i=1}^{n} {e_{r_i}}^2 \tag{A I-5}$$

5. Cross-Entropy loss Function (crossentropy):

$$H(p,q) = -\sum_{x \in \mathcal{X}} p(x) * log\ q(x) \tag{A I-6}$$

where p and q are discrete probability distributions, and $\mathcal{X}$ is the given set.

In the validation phase of the trained NN models which starts directly after the training phase, the program uses different practical data sets which are considered as new unseen data for the NN generated models. All the data of the training and validation phases of each computational cycle is stored in a matrix form including all the generated NN models and its related number of neurons, optimizer training option used, network structure, performance, and activation functions used. The mean absolute error (MAE) had been chosen as an efficient function in training and validation phases for distinguishing between the generated NN models. This function is selected because, in some robot velocities, the error generated by the robot in the first half of the circle is equal to the second half but in the opposite direction. For instance, if we use the mean error to evaluate this error, the mean, in this case, will not describe the real error in an appropriate way; however, MAE is giving the average distance between each data point and the mean, therefore it gives an idea about the variability in an error data set since we are only interested in the deviations of the error values and not whether they are above or below the mean.

3.3 Selection Method of the Best Generated NN Model

In order to evaluate the generated NN models and then to select the best NN model, we categorized all the NN models' output data into three groups based on the influence of neurons number versus the overall MAE as follows:

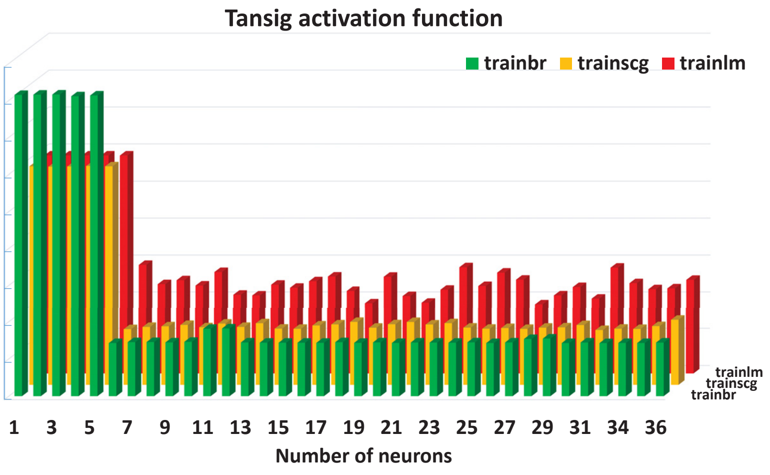
1. **First Group**: contains (tansig) activation functions, number of neurons, and the three training algorithms (trainlm, trainscg and trainbr) as shown in Figure a.
2. **Second Group**: contains (logsig) activation functions, number of neurons and the three training algorithms (trainlm, trainscg and trainbr) as demonstrated in Figure b.
3. **Third Group**: contains the best results of the previous two groups with the different performance functions (mae, mase, sae, sse and crossentropy) as exposed in Figure I-5.

The results of these three groups for the first optimizer option (ADAM) are displayed in Figures I-4 and I-5, while for the second optimizer option (RMSProp) is demonstrated in Figures I-6 and I-7. Finally, the third optimizer option (SGDM) is exposed in Figures I-8 and I-9. The best NN model has been selected based on the minimum value of MAE summarized in Table I-2.

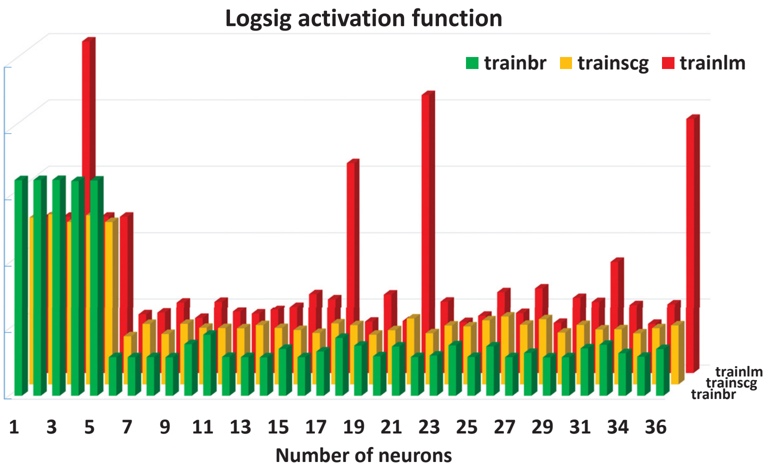
As can be seen from the figures, on one hand, the best training algorithm for all the three optimizer options is Bayesian regularization algorithm (trainbr). Although it requires more training time, it gives good generalized NN models for small, difficult, and noisy data. On other hand (logsig) gives better results with (ADAM) optimizing option while tansig is better for the other two optimizer options (RMSProp and SGDM).

From Table I-2, we can conclude that the best NN model using ADAM option is generated by the number of 30 neurons using sse as a performance function and achieving the minimum MAE of 0.0345 mm, while the model with 16 neurons, is the best for RMSProp option using mae as a performance function and having the minimum MAE of 0.0361 mm. Finally, the NN model with a number of neurons of 31 recorded the best value of a minimum MAE of 0.0333 mm using mae performance function. The comparison of the best of all the three optimizing options showed that the best NN model is obtained by using the third optimizing option (SGDM) when the neuron number is equal to 31 as a result of (tansigm) training algorithm with Bayesian regularization algorithm and (trainbr) activation function, this model has the min MAE of 0.0333 mm. Therefore, this model is the one that will be used in modeling the Meca500 robot since it provides great accuracy with the structure presented in Figure I-10.

Generally, the overall MAE is inversely proportional to the number of neurons, however, in some cases, a lower number of neurons gives better results than higher if the data is over-fitted. For instance, the model generated by 30 neurons gives better results than the one generated by 34 neurons when using ADAM optimizing option as presented in Table I-2.



(a) Tansig activation function



(b) Logsig activation function

Figure-A I-4 Influence of number of neurons using ADAM optimizing option with different activation functions and trainbr training algorithms with different performance functions

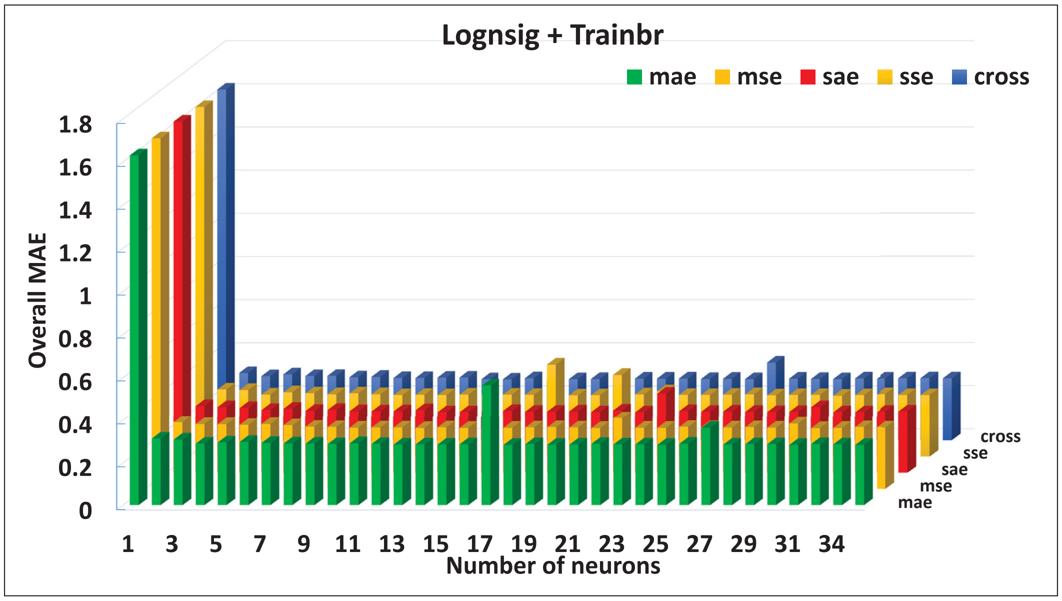


Figure-A I-5 Influence of number of neurons using ADAM optimizing option with Logsig activation function and trainbr training algorithms with different performance functions

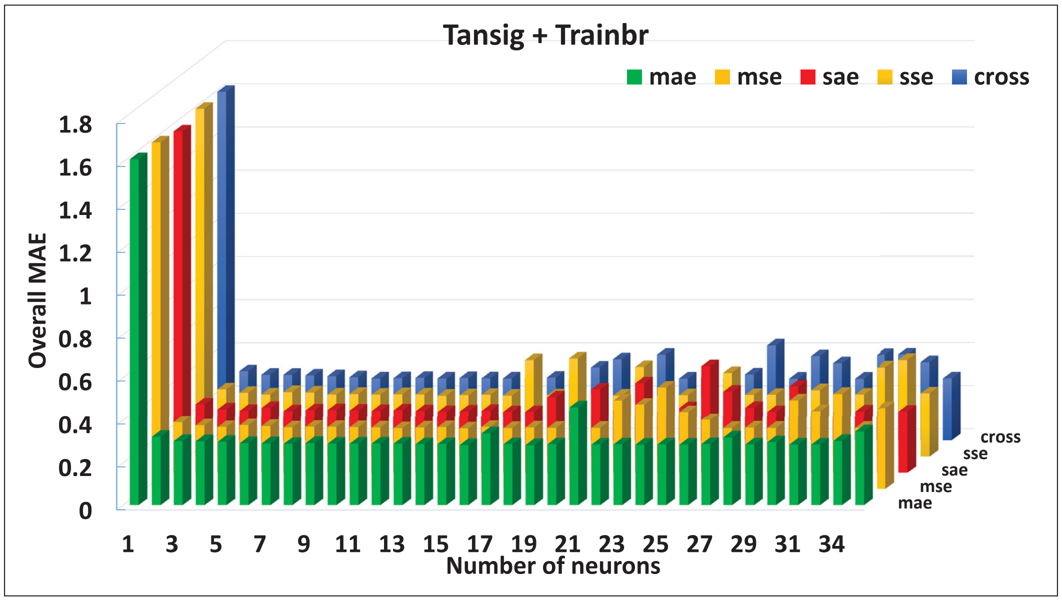
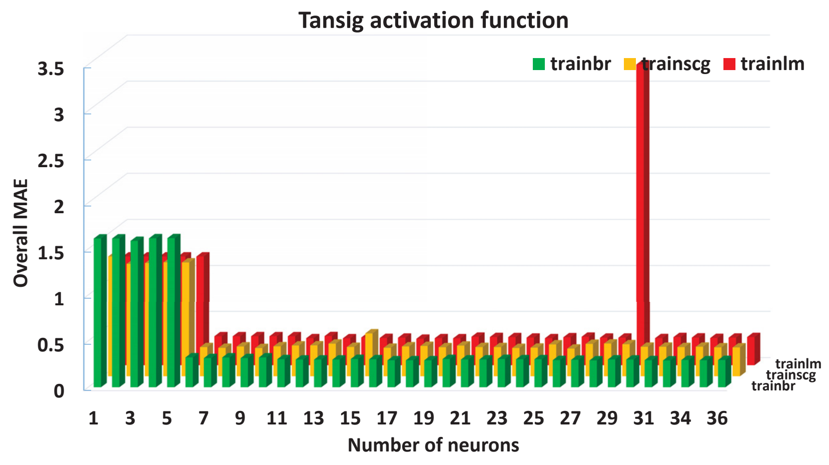
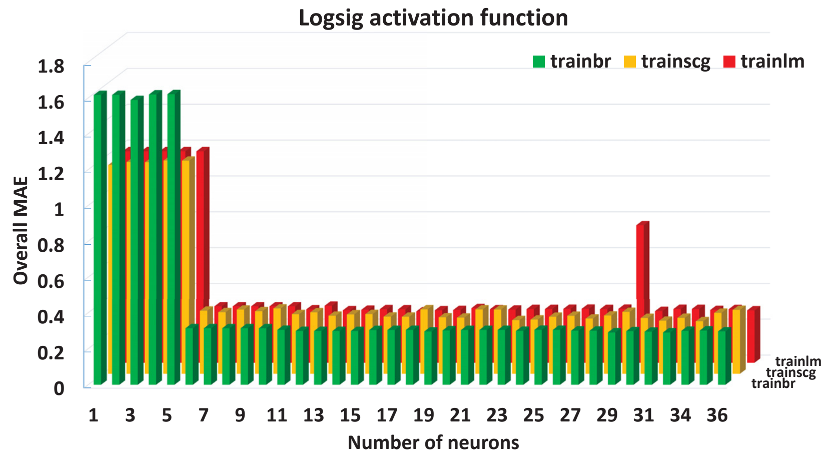


Figure-A I-6 Influence of number of neurons using RMSProp optimizing option with Tansig activation function and trainbr training algorithms with different performance functions



(a) Tansig activation function

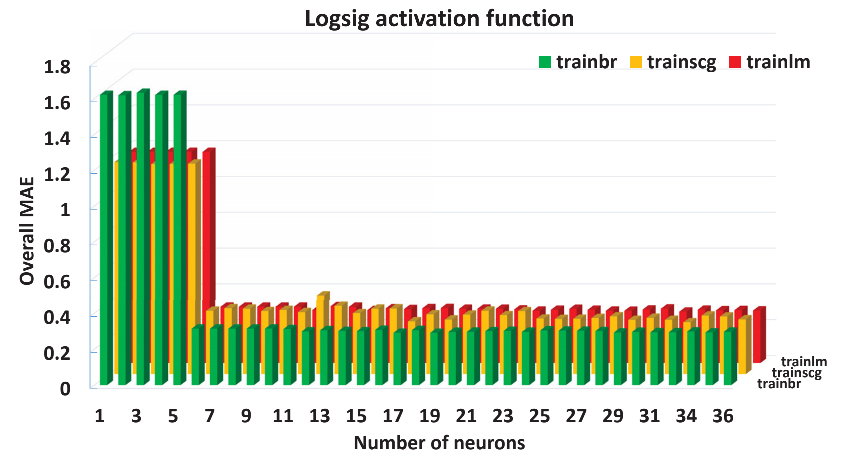


(b) Logsig activation function

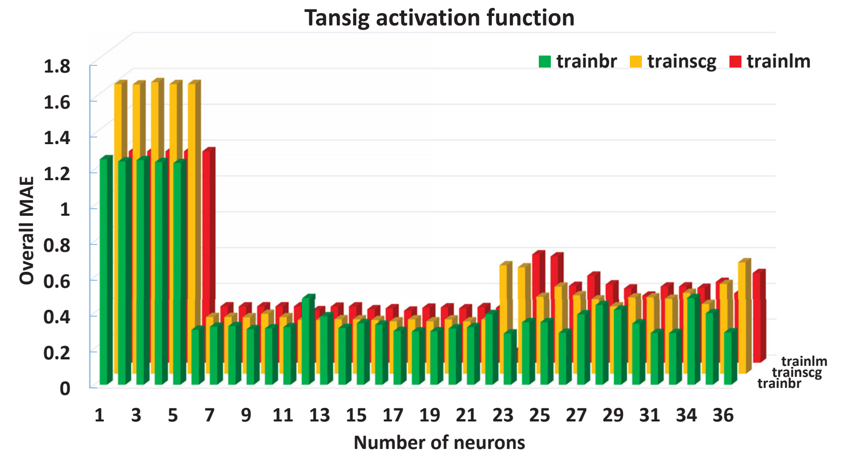
Figure-A I-7 Influence of number of neurons using RMSProp optimizing option with different activation functions and trainbr training algorithms with different performance functions

3.4 Testing the generated best Neural Model with the Real robot Model

In this section, a testing phase is introduced to test the performance of the chosen NN best model (SGDM optimizer/neuron number 31). This is considered an advanced phase since the validation phase already validated the generated models from the training phase by using new data sets as mentioned in section 3.1. We are testing the model by applying robot velocity vectors for both low linear velocity of 7 mm/s, the intermediate linear velocity of 97 mm/s,



(a) Tansig activation function



(b) Logsig activation function

Figure-A I-8 Influence of number of neurons using SGDM optimizing option with different activation functions and trainbr training algorithms with different performance functions

and high linear velocity of 167 mm/s as model inputs, because these are totally new unseen data for the model not even during the previous phases of training and validation. Then, the model velocity vector outputs are compared to the real ones, the error is stored then plotted in Figure I-11, Figure I-12, and I-13. The results show a good NN model performance with a very small MAE not exceeding 0.15 mm in robot low velocity, 0.52 mm in robot intermediate velocity, and 1.5 mm in robot high velocity in both directions Y and Z.

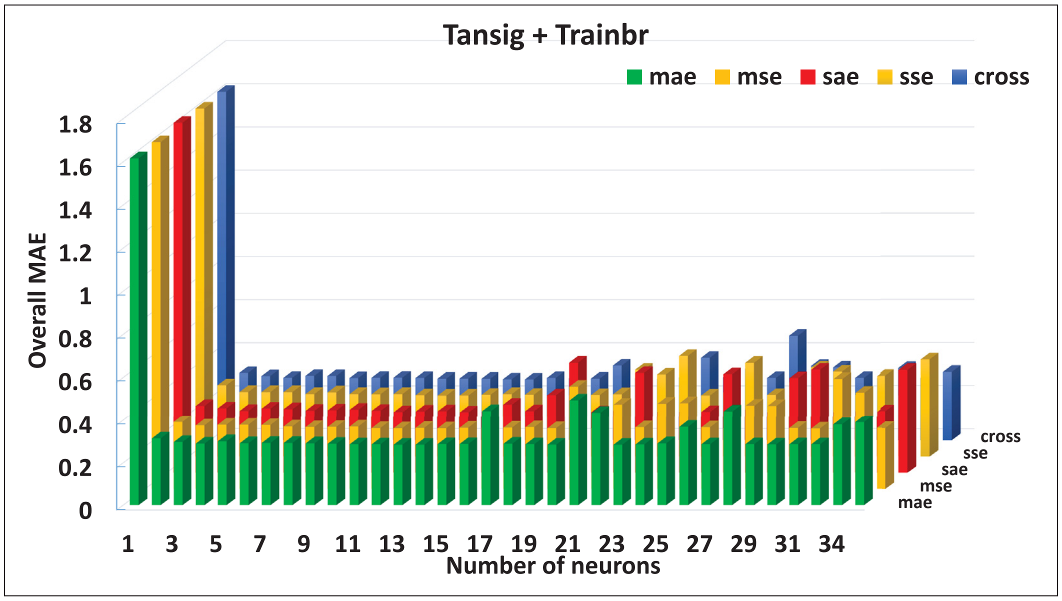


Figure-A I-9 Influence of number of neurons using SGDM optimizing option with Tansig activation function and trainbr training algorithms with different performance functions

Table-A I-2 Summary of the three groups best results using different optimizer options

Optimizer option	**N# of neurons**	**Performance function**	**Training algorithm**	**Activation function**	**min MAE (mm)**
ADAM	34	mae	trainbr	logsig	0.0362
	31	mse			0.0355
	22	sae			0.0363
	30	sse			0.0345
	14	crossentropy			0.0374
RMSProp	16	mae		tansig	0.0361
	15	mse			0.0374
	23	sae			0.0367
	12	sse			0.0376
	17	crossentropy			0.0362
SGDM	23	mae			0.0351
	31	mse			0.0333
	14	sae			0.0379
	17	sse			0.0353
	15	crossentropy			0.0369

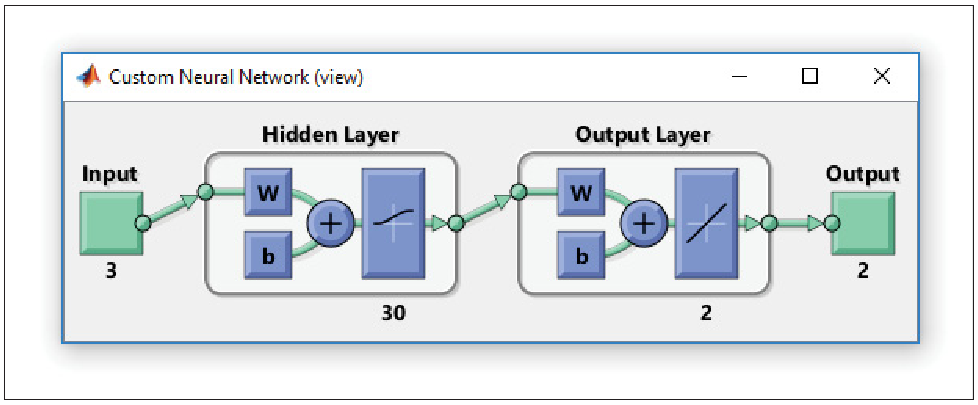
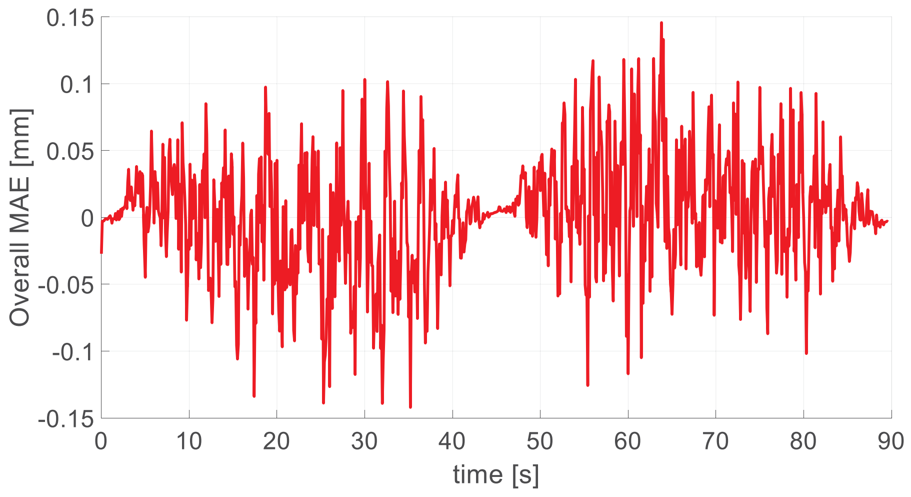


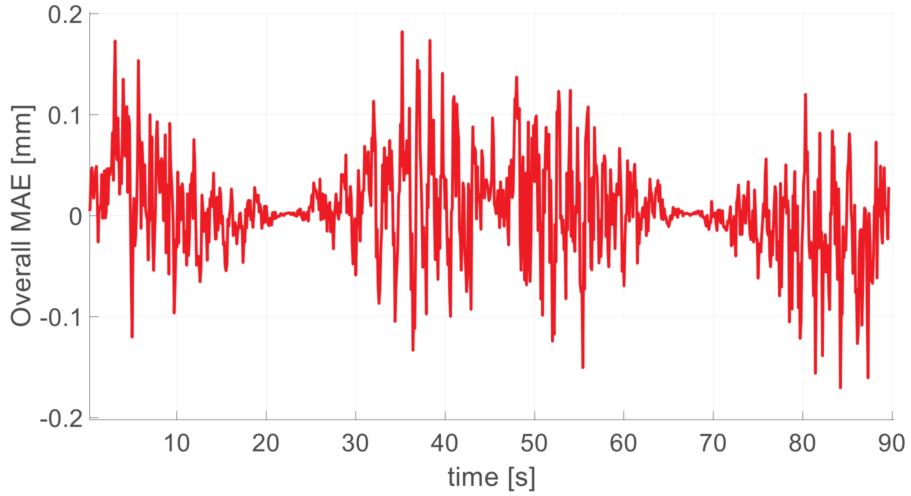
Figure-A I-10 The best chosen Static NN model based on data analysis
Taken from The MathWorks (2020a)

4. Conclusion

It can be concluded that the generated neural model can faithfully recreate the behavior of a real Meca500 robot system while performing movements in a circular trajectory with different robot's linear velocities. It has succeeded in predicting the error between the robot inputs and outputs. In this research, the problem of analyzing the generated NN models with the non-uniqueness of the parameter values problem solution is considered. Solving it, our program is used to evaluate, analyze, and judge the performance of the generated NN models, then to select the best model through analyzing the final model data. A static feed-forward neural network model of the Meca500 robot is developed assuming a single network layered structure changing all the available parameters such as the number of neurons, performance functions, training optimizer options, activation functions, and training algorithms. It generates a model that can be used for controller real-time simulation before its implementation phase. Using the best-generated model after analysis, the effectiveness of the model performance to determine the real robot system is proved using new and unseen data for the model with low and high robot linear velocity to cover a wide range of different robot velocities. The results show that the best chosen neural network model accurately identified the real robot system and simulate its behavior with small errors.

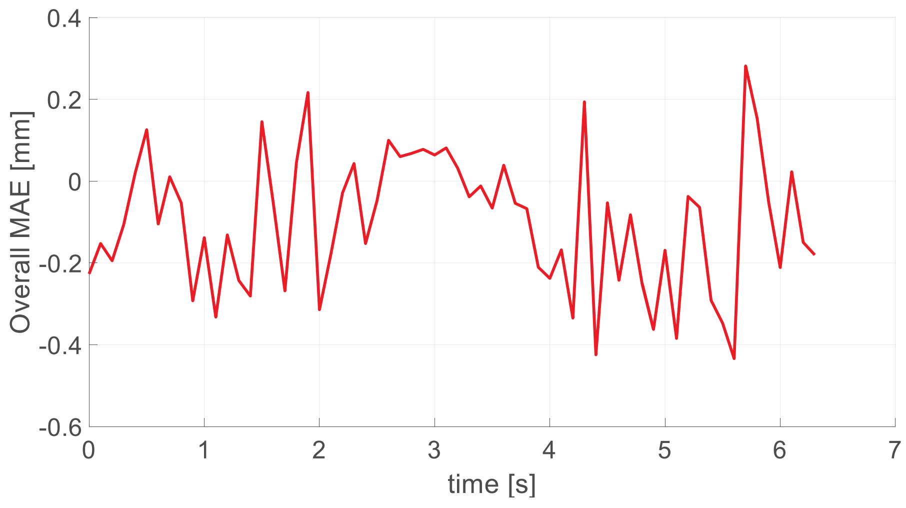


(a) In Y direction

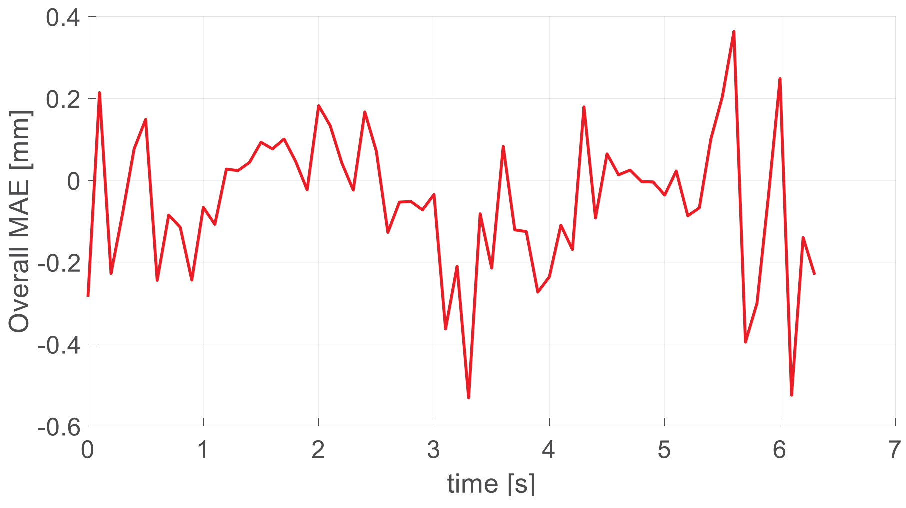


(b) In Z direction

Figure-A I-11 The Error between the Real robot and Neural model velocity vector outputs at robot linear velocity of 7 mm/s

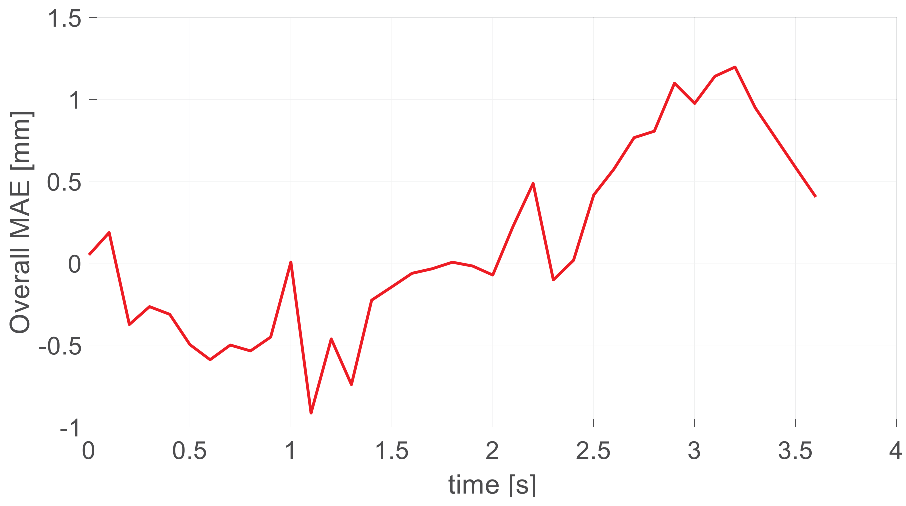


(a) In Y direction

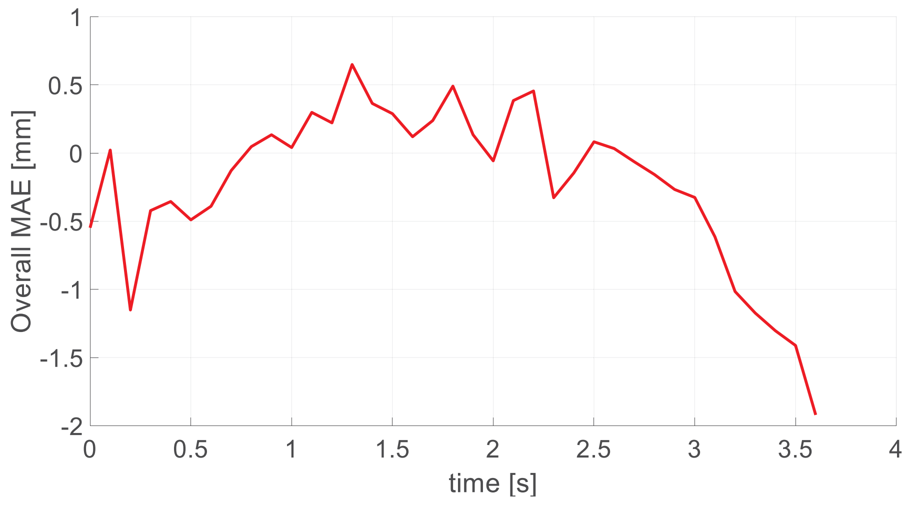


(b) In Z direction

Figure-A I-12 The Error between the Real robot and Neural model velocity vector outputs at robot linear velocity of 97 mm/s



(a) In Y direction



(b) In Z direction

Figure-A I-13 The Error between the Real robot and Neural model velocity vector outputs at robot linear velocity of 167 mm/s

BIBLIOGRAPHY

3dconnexion. (2020). SpaceMouse® Compact User Guide. Consulted at https://3dconnexion.com/us/product/spacemouse-module/.

Abdallah, M. A. & Fareh, R. (2019). Tracking Control of Serial Robot Manipulator using Active Disturbance Rejection Control. *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, pp. 1–5.

Abderrahim, M., Khamis, A., Garrido, S. & Moreno, L. (2004). Accuracy and calibration issues of industrial manipulators. *Industrial robotics: programming, simulation and application*, 131–146.

Abou Elyazed, M. M., Mabrouk, M. H., Abo Elnor, M. & Mahgoub, H. M. (2016). Trajectory planning of five DOF manipulator: dynamic feed forward controller over computed torque controller. *The International Conference on Applied Mechanics and Mechanical Engineering*, 17(17th International Conference on Applied Mechanics and Mechanical Engineering), 1–14.

Arcos-Legarda, J., Cortes-Romero, J. & Tovar, A. (2016). Active disturbance rejection control based on generalized proportional integral observer to control a bipedal robot with five degrees of freedom. *2016 American Control Conference (ACC)*, pp. 3928–3933.

Bekey, G. A. & Goldberg, K. Y. (Eds.). (1993). *Neural Networks in Robotics*. Springer US. doi: 10.1007/978-1-4615-3180-7.

Ben-Ari, M. & Mondada, F. (2018). Robots and Their Applications. In *Elements of Robotics* (pp. 1–20). Cham: Springer International Publishing. doi: 10.1007/978-3-319-62533-1_1.

Binh, N. T., Tung, N. A., Nam, D. P. & Quang, N. H. (2019). An adaptive backstepping trajectory tracking control of a tractor trailer wheeled mobile robot. *International Journal of Control, Automation and Systems*, 17(2), 465–473.

Brahmi, B., Brahmi, A., Saad, M., Gauthier, G. & Habibur Rahman, M. (2019a). Robust adaptive tracking control of uncertain rehabilitation exoskeleton robot. *Journal of Dynamic Systems, Measurement, and Control*, 141(12).

Brahmi, B., Laraki, M. H., Saad, M., Rahman, M., Ochoa-Luna, C. & Brahmi, A. (2019b). Compliant adaptive control of human upper-limb exoskeleton robot with unknown dynamics based on a Modified Function Approximation Technique (MFAT). *Robotics and Autonomous Systems*, 117, 92–102.

Cao, Y.-Y. & Frank, P. M. (2000). Analysis and synthesis of nonlinear time-delay systems via fuzzy control approach. *IEEE Transactions on fuzzy systems*, 8(2), 200–211.

Carlevarino, A., Martinotti, R., Metta, G. & Sandini, G. (2000, jul). An incremental growing neural network and its application to robot control. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, 5, 323-328 vol.5.

Castañeda, L. A., Luviano-Juárez, A. & Chairez, I. (2014). Robust trajectory tracking of a delta robot through adaptive active disturbance rejection control. *IEEE Transactions on control systems technology*, 23(4), 1387–1398.

Čep, R., Malotová, Š., Kratochvíl, J., Stančeková, D., Czán, A. & Jakab, T. (2018a). Diagnosis of machine tool with using Renishaw ball-bar system. *MATEC Web of Conferences*, 157, 01006.

Čep, R., Malotová, Š., Kratochvíl, J., Stančeková, D., Czán, A. & Jakab, T. (2018b, 01). Diagnosis of machine tool with using Renishaw ball-bar system. *MATEC Web of Conferences*, 157, 01006.

Chen, H. & Liu, Y. (2013). Robotic assembly automation using robust compliant control. *Robotics and Computer-Integrated Manufacturing*, 29(2), 293–300.

Chen, H., Sun, X., Xu, S. & Wang, Y. (2019). Robust Stabilization of Extended Nonholonomic Chained-Form Systems with Dynamic Nonlinear Uncertain Terms by Using Active Disturbance Rejection Control. *Complexity*, 2019.

Chen, Y., Chu, B., Freeman, C. T. & Liu, Y. (2020). Generalized iterative learning control with mixed system constraints: A gantry robot based verification. *Control Engineering Practice*, 95, 104260.

Cheng, X., Tu, X., Zhou, Y. & Zhou, R. (2019). Active Disturbance Rejection Control of Multi-Joint Industrial Robots Based on Dynamic Feedforward. *Electronics*, 8(5), 591.

Chiang, Y.-M., Chang, L.-C. & Chang, F.-J. (2004). Comparison of static-feedforward and dynamic-feedback neural networks for rainfall-runoff modeling. *Journal of Hydrology*, 290(3), 297–311.

Chien, T.-L., Chen, C.-C., Tsai, M.-C. & Chen, Y.-C. (2010). Control of AMIRA's ball and beam system via improved fuzzy feedback linearization approach. *Applied Mathematical Modelling*, 34(12), 3791–3804.

Chu, Z., Wu, C. & Sepehri, N. (2019). Active Disturbance Rejection Control Applied to High-order Systems with Parametric Uncertainties. *International Journal of Control, Automation and Systems*, 17(6), 1483–1493.

Cui, W., Tan, W., Li, D., Wang, Y. & Wang, S. (2020). A Relay Feedback Method for the Tuning of Linear Active Disturbance Rejection Controllers. *IEEE Access*, 8, 4542–4550.

Dabin, V. (2018). *Control of a quadricopter by active disturbance rejection*. (Master's thesis, École Polytechnic de Montréal).

Das, P., Mehta, R. & Roy, O. (2020). Stability Analysis of a Nonlinear MIMO System using Fractional Order Active Disturbance Rejection Controller. *Available at SSRN 3517372*.

Denso. (2020). Product categories. Consulted at https://www.denso-wave.com/en/robot/.

Desai, R., Patre, B. & Pawar, S. N. (2018). Active disturbance rejection control with adaptive rate limitation for process control application. *2018 Indian Control Conference (ICC)*, pp. 131–136.

Dimeas, F., Moulianitis, V. C., Papakonstantinou, C. & Aspragathos, N. (2016). Manipulator performance constraints in cartesian admittance control for human-robot cooperation. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3049–3054.

Ding, S., Wu, W., Huang, X., Song, A. & Zhang, Y. (2019). Single-axis driven measurement method to identify position-dependent geometric errors of a rotary table using double ball bar. *The International Journal of Advanced Manufacturing Technology*, 101(5-8), 1715–1724.

Dong, H., Wang, Z., Ho, D. W. C. & Gao, H. (2010). Robust H_∞ Fuzzy Output-Feedback Control With Multiple Probabilistic Delays and Multiple Missing Measurements. *IEEE Transactions on Fuzzy Systems*, 18(4), 712-725. doi: 10.1109/TFUZZ.2010.2047648.

Ebrahimi, A. (2014). Regulated model-based and non-model-based sliding mode control of a MEMS vibratory gyroscope. *Journal of Mechanical Science and Technology*, 28(6), 2343–2349.

Edwards, C. & Spurgeon, S. (1998). *Sliding mode control: theory and applications*. Crc Press.

Esmaeili, S. & Mayer, J. (2020). An Integrated Geometric and Hysteretic Error Model of a Three Axis Machine Tool and Its Identification With a 3D Telescoping Ball-Bar. *Journal of Manufacturing and Materials Processing*, 4(1), 24.

Fereidouni, A., Masoum, M. A. & Moghbel, M. (2015). A new adaptive configuration of PID type fuzzy logic controller. *ISA transactions*, 56, 222–240.

Gao, J., He, Q., Zhan, Z. & Gao, H. (2016, apr). Dynamic modeling based on fuzzy Neural Network for a billiard robot. *2016 IEEE 13th International Conference on Networking, Sensing, and Control (ICNSC)*, pp. 1-4.

Gao, Z. Q. (2009). A paradigm shift in feedback control system design. *Proceedings of the American Control Conference*, pp. 2451–2457.

Gao, Z. (2006a). Scaling and bandwidth-parameterization based controller tuning. 6, 4989–4996.

Gao, Z. (2006b). Active disturbance rejection control: a paradigm shift in feedback control system design. *2006 American control conference*, pp. 7–pp.

Gao, Z. (2015). Active disturbance rejection control: from an enduring idea to an emerging technology. *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pp. 269–282.

Gao, Z., Hu, S. & Jiang, F. (2001a). A novel motion control design approach based on active disturbance rejection. *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No. 01Ch37228)*, 5, 4877–4882.

Gao, Z., Huang, Y. & Han, J. (2001b). An alternative paradigm for control system design. *Proceedings of the 40th IEEE conference on decision and control (Cat. No. 01CH37228)*, 5, 4578–4585.

Gencer, O. (2014). *Robotic Arm Controlling & Simulation in MATLAB SIMULINK*. (Master's thesis, T.R.Istanbul Aydin University, Istanbul, Turkey). Consulted at https://www.slideshare.net/OuzGener/bitirme-tezim.

Gharaaty, S., Shu, T., Xie, W.-F., Joubair, A. & Bonev, I. A. (2017). Accuracy enhancement of industrial robots by on-line pose correction. *2017 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, pp. 214–220.

Gharaaty, S., Shu, T., Joubair, A., Xie, W. F. & Bonev, I. A. (2018). Online pose correction of an industrial robot using an optical coordinate measure machine system. *International Journal of Advanced Robotic Systems*, 15(4), 1729881418787915. doi: 10.1177/1729881418787915.

Grafakos, S., Dimeas, F. & Aspragathos, N. (2016). Variable admittance control in pHRI using EMG-based arm muscles co-activation. *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 001900–001905.

Greenway, B. (2000). Robot accuracy. *Industrial Robot: An International Journal*.

Gu Fang & Dissanayake, M. W. M. G. (1995, nov). Neural networks for modelling robot forward dynamics. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 5, 2715-2719 vol.5.

Guanyu, Q., Cheng, P., Zhenbang, X. & Huibin, G. (2019). Trajectory tracking control based on linear active disturbance rejection controller for 6-DOF robot manipulator. *HIGH TECHNOLOGY LETTERS*, (4), 347–354.

Guo, B.-Z. & Zhao, Z.-L. (2015). Active disturbance rejection control: Theoretical perspectives. *Communications in Information and Systems*, 15(3), 361–421.

Guo, B.-Z. & Zhao, Z.-L. (2016). *Active disturbance rejection control for nonlinear systems: An introduction*. John Wiley & Sons.

Gurumurthy, G. & Das, D. K. (2020). Terminal sliding mode disturbance observer based adaptive super twisting sliding mode controller design for a class of nonlinear systems. doi: https://doi.org/10.1016/j.ejcon.2020.05.004.

Hacioglu, Y. & Yagiz, N. (2019). Fuzzy robust backstepping with estimation for the control of a robot manipulator. *Transactions of the Institute of Measurement and Control*, 41(10), 2816–2825.

Han, J.-Q. (1999). Nonlinear design methods for control systems. *IFAC Proceedings Volumes*, 32(2), 1531–1536.

Han, J. (1995). A class of extended state observers for uncertain systems. *Control and decision*, 10(1), 85–88.

Han, J. (2009). From PID to active disturbance rejection control. *IEEE transactions on Industrial Electronics*, 56(3), 900–906.

Hanses, M., Behrens, R. & Elkmann, N. (2016). Hand-guiding robots along predefined geometric paths under hard joint constraints. *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1-5. doi: 10.1109/ETFA.2016.7733600.

Hogan, N. (1987). Stable execution of contact tasks using impedance control. *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, 4, 1047-1054. doi: 10.1109/ROBOT.1987.1087854.

Hogan, N. (1984). Impedance control: An approach to manipulation. *1984 American control conference*, pp. 304–313.

Hua, C. & Ding, S. X. (2011). Decentralized networked control system design using T–S fuzzy approach. *IEEE Transactions on fuzzy systems*, 20(1), 9–21.

Huang, C.-J., Li, T.-H. S. & Chen, C.-C. (2009). Fuzzy feedback linearization control for MIMO nonlinear system and its application to full-vehicle suspension system. *Circuits, Systems and*

Signal Processing, 28(6), 959.

Huang, C. & Yin, Y. (2019). Wind Turbine Pitch Control Based on Error-based ADRC Approach Optimized by Brain Storm Optimization Algorithm. *2019 1st International Conference on Industrial Artificial Intelligence (IAI)*, pp. 1–6.

Huang, Y. & Han, J. (2000). Analysis and design for the second order nonlinear continuous extended states observer. *Chinese science bulletin*, 45(21), 1938.

Hyatt, P., Williams, C. S. & Killpack, M. D. (2020). Parameterized and GPU-Parallelized Real-Time Model Predictive Control for High Degree of Freedom Robots. *arXiv preprint arXiv:2001.04931*.

Ibrahem, I., Akhrif, O., Moustapha, H. & Staniszewski, M. (2019). Neural networks modelling of aero-derivative gas turbine engine: a comparison study.

Jiang, L., Qiu, H., Wu, Z. & He, J. (2016a). Active disturbance rejection control based on adaptive differential evolution for two-wheeled self-balancing robot. *2016 Chinese Control and Decision Conference (CCDC)*, pp. 6761–6766.

Jiang, L., Qiu, H., Wu, Z. & He, J. (2016b). Active disturbance rejection control based on adaptive differential evolution for two-wheeled self-balancing robot. *2016 Chinese Control and Decision Conference (CCDC)*, pp. 6761–6766.

Jin, Z., Yu, C., Li, J. & Ke, Y. (2014). A robot assisted assembly system for small components in aircraft assembly. *Industrial Robot: An International Journal*.

Kali, Y., Saad, M., Benjelloun, K. & Khairallah, C. (2018). Super-twisting algorithm with time delay estimation for uncertain robot manipulators. *Nonlinear Dynamics*, 93(2), 557–569.

Kern Molina, J., Jamett Dominguez, M., Urrea Onate, C. & Torres Salamea, H. (2014). Development of a neural controller applied in a 5 DOF robot redundant. *IEEE Latin America Transactions*, 12(2), 98-106.

Khaled, T. A., Akhrif, O. & Bonev, I. A. (2020). Dynamic Path Correction of an Industrial Robot using a Distance Sensor and an ADRC Controller. *IEEE/ASME Transactions on Mechatronics*, 1-1.

Klančar, G. & Škrjanc, I. (2007). Tracking-error model-based predictive control for mobile robots in real time. *Robotics and autonomous systems*, 55(6), 460–469.

Krebs, H., Hogan, N., Durfee, W. & Herr, H. (2006). Rehabilitation robotics, orthotics, and prosthetics. *Textbook of neural repair and rehabilitation*, 2, 165–181.

Kubela, T., Pochyly, A. & Singule, V. (2019). High Accurate Robotic Machining based on Absolute Part Measuring and On-Line Path Compensation. *2019 International Conference on Electrical Drives Power Electronics (EDPE)*, pp. 143-148. doi: 10.1109/EDPE.2019.8883912.

Kumar, A. & Kumar, V. (2017). Artificial bee colony based design of the interval type-2 fuzzy PID controller for robot manipulator. 602–607.

Lahr, G. J., Soares, J. V., Garcia, H. B., Siqueira, A. A. & Caurin, G. A. (2016). Understanding the implementation of impedance control in industrial robots. *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)*, pp. 269–274.

Lecours, A., Mayer-St-Onge, B. & Gosselin, C. (2012). Variable admittance control of a four-degree-of-freedom intelligent assist device. *2012 IEEE International Conference on Robotics and Automation*, pp. 3903-3908. doi: 10.1109/ICRA.2012.6224586.

Lee, S.-D., Ahn, K.-H. & Song, J.-B. (2016). Torque control based sensorless hand guiding for direct robot teaching. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 745–750.

Li, J., Li, R. & Zheng, H. (2016). Quadrotor modeling and control based on linear active disturbance rejection control. *2016 35th Chinese Control Conference (CCC)*, pp. 10651–10656.

Liang, Y.-W., Chen, C.-C. & Xu, S. S.-D. (2013). Study of Reliable Design Using TS Fuzzy Modeling and Integral Sliding Mode Control Schemes. *International Journal of Fuzzy Systems*, 15(2).

Liu, C., Zhao, Z. & Wen, G. (2019). Adaptive neural network control with optimal number of hidden nodes for trajectory tracking of robot manipulators. *Neurocomputing*, 350, 136–145.

Liu, G., li, Q., Fang, L., Han, B. & Zhang, H. (2020). A new joint friction model for parameter identification and sensor-less hand guiding in industrial robots. *Industrial Robot: the international journal of robotics research and application*, ahead-of-print. doi: 10.1108/IR-03-2020-0053.

Liu, H., Tian, X., Wang, G. & Zhang, T. (2016). Finite-Time H_∞ Control for High-Precision Tracking in Robotic Manipulators Using Backstepping Control. *IEEE Transactions on Industrial Electronics*, 63(9), 5501–5513.

Liu, S. (2002). An on-line reference-trajectory generator for smooth motion of impulse-controlled industrial manipulators. *7th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 02TH8623)*, pp. 365–370.

Lotufo, M. A., Colangelo, L., Perez-Montenegro, C., Canuto, E. & Novara, C. (2019). UAV quadrotor attitude control: An ADRC-EMC combined approach. *Control Engineering Practice*, 84, 13–22.

Luo, R. C. (2016). Assistive robot endoscopic system with intuitive maneuverability for laparoscopic surgery and method thereof. Google Patents. US Patent App. 14/597,672.

Luo, S., Sun, Q., Wu, W., Sun, M., Chen, Z. & He, Y. (2019). Accurate flight path tracking control for powered parafoil aerial vehicle using ADRC-based wind feedforward compensation. *Aerospace Science and Technology*, 84, 904–915.

Ma, D., Xia, Y., Li, T. & Chang, K. (2016). Active disturbance rejection and predictive control strategy for a quadrotor helicopter. *IET Control Theory & Applications*, 10(17), 2213–2222.

Madonski, R., Shao, S., Zhang, H., Gao, Z., Yang, J. & Li, S. (2019). General error-based active disturbance rejection control for swift industrial implementations. *Control Engineering Practice*, 84, 218–229.

Madoński, R. (2016). *On active disturbance rejection in robotic motion control*. (Ph.D. thesis).

Manyika, J. (2012). *Manufacturing the future: The next era of global growth and innovation*. McKinsey Global Institute.

Martins, M., Cunha, A. & Morgado, L. (2012). Usability test of 3Dconnexion 3D mice versus keyboard+ mouse in Second Life undertaken by people with motor disabilities due to medullary lesions. *Procedia Computer Science*, 14, 119–127.

Mecademic. (2020). Meca500 (R3) User and Programming Guide. Consulted at https://www.mecademic.com/resources/documentation.

Meng, Q.-H. M. (1995). Comparison study of model-based and non-model-based robot controllers. *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, 1, 61–66.

Michałek, M. M. (2016). Robust trajectory following without availability of the reference time-derivatives in the control scheme with active disturbance rejection. *2016 American Control Conference (ACC)*, pp. 1536-1541.

Moe, S. & Schjølberg, I. (2013). Real-time hand guiding of industrial manipulator in 5 DOF using Microsoft Kinect and accelerometer. *2013 IEEE RO-MAN*, pp. 644-649. doi: 10.1109/ROMAN.2013.6628421.

Mudi, R. K. & Pal, N. R. (2001). A note on fuzzy PI-type controllers with resetting action. *Fuzzy Sets and Systems*, 121(1), 149 - 159. doi: https://doi.org/10.1016/S0165-0114(99)00143-8. Formal Methods for Fuzzy Modeling and Control.

Müller, F., Jäkel, J. & Suchy, J. (2015). Tunnel-shaped potential force fields for improved hand-guiding of robotic arms. *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 429-434. doi: 10.1109/MMAR.2015.7283914.

Norman, A., Schönberg, A., Gorlach, I. & Schmitt, R. (2010). Cooperation of industrial robots with indoor-GPS. *Proceedings of the international conference on competitive manufacturing*, pp. 215–224.

Norman, A. R., Schönberg, A., Gorlach, I. A. & Schmitt, R. (2013). Validation of iGPS as an external measurement system for cooperative robot positioning. *The International Journal of Advanced Manufacturing Technology*, 64(1-4), 427–446.

Nowicki, M., Madoński, R. & Kozłowski, K. (2015). First look at conditions on applicability of ADRC. *2015 10th International Workshop on Robot Motion and Control (RoMoCo)*, pp. 294–299.

Nubiola, A. & Bonev, I. A. (2013). Absolute calibration of an ABB IRB 1600 robot using a laser tracker. *Robotics and Computer-Integrated Manufacturing*, 29(1), 236–245.

Nubiola, A. & Bonev, I. A. (2014). Absolute robot calibration with a single telescoping ballbar. *Precision Engineering*, 38(3), 472–480.

Nubiola, A., Slamani, M., Joubair, A. & Bonev, I. A. (2014). Comparison of two calibration methods for a small industrial robot based on an optical CMM and a laser tracker. *Robotica*, 32(3), 447–466.

Oztemel, E. & Gursev, S. (2020). Literature review of Industry 4.0 and related technologies. *Journal of Intelligent Manufacturing*, 31(1), 127–182.

Pires, J. N. & Bogue, R. (2009). Finishing robots: a review of technologies and applications. *Industrial Robot: An International Journal*.

Renishaw, Q.-W. (2016). Renishaw, QC20-W. Issued. Consulted at https://www.renishaw.com/en/ballbar-testing-explained--6818.

Reznik, L. (1997). *Fuzzy Controllers Handbook: How to Design Them, How They Work*. Elsevier Science. Consulted at https://books.google.ca/books?id=ar0-84SktfQC.

Robotics. (2018). Aerospace Applications. Consulted on 2018-04-18 at https://www.robotics.org/blog-article.cfm/The-Latest-Uses-of-Industrial-Robotics-in-Aerospace-Applications/110.

RobotWorx. (2020). Industrial Robot Applications. Consulted on 2020-01-22 at https://www.robots.com/applications.

Saied, H., Chemori, A., El Rafei, M., Francis, C. & Pierrot, F. (2019). From non-model-based to model-based control of pkms: a comparative study. In *Mechanism, Machine, Robotics and Mechatronics Sciences* (pp. 153–169). Springer.

Schneider, U., Drust, M., Diaz Posada, J. & Verl, A. (2013). Position control of an industrial robot using an optical measurement system for machining purposes.

Shao, S. & Gao, Z. (2017). On the conditions of exponential stability in active disturbance rejection control based on singular perturbation analysis. *International Journal of Control*, 90(10), 2085–2097.

Shiakolas, P., Conrad, K. & Yih, T. (2002). On the accuracy, repeatability, and degree of influence of kinematics parameters for industrial robots. *International journal of modelling and simulation*, 22(4), 245–254.

Shu, T., Gharaaty, S., Xie, W., Joubair, A. & Bonev, I. A. (2018). Dynamic path tracking of industrial robots with high accuracy using photogrammetry sensor. *IEEE/ASME Transactions on Mechatronics*, 23(3), 1159–1170.

Siciliano, B. & Khatib, O. (2016). *Springer handbook of robotics*. Springer.

Slamani, M., Joubair, A. & Bonev, I. A. (2015). A comparative evaluation of three industrial robots using three reference measuring techniques. *Industrial Robot: An International Journal*.

Song, P., Yu, Y. & Zhang, X. (2017). Impedance Control of Robots: An Overview. *2017 2nd International Conference on Cybernetics, Robotics and Control (CRC)*, pp. 51-55. doi: 10.1109/CRC.2017.20.

Su, X., Shi, P., Wu, L. & Song, Y.-D. (2012). A novel approach to filter design for T–S fuzzy discrete-time systems with time-varying delay. *IEEE Transactions on Fuzzy Systems*, 20(6), 1114–1129.

Suhail, S. A., Bazaz, M. A. & Hussain, S. (2020). Active Disturbance Rejection Control Applied to a DC Motor for Position Control. In *Proceedings of ICETIT 2019* (pp. 437–448). Springer.

Tanaka, K. & Sugeno, M. (1992). Stability analysis and design of fuzzy control systems. *Fuzzy sets and systems*, 45(2), 135–156.

The MathWorks, I. (2020a). Deep Learning Toolbox. Natick, Massachusetts, United State. Consulted at https://www.mathworks.com/products/deep-learning.html.

The MathWorks, I. (2020b). Fuzzy Logic Toolbox for MATLAB. Natick, Massachusetts, United State. Consulted at https://www.mathworks.com/products/fuzzy-logic.html.

Underwood, S. & Gallimore, J. J. (2010). One versus Two-Handed Six Degree-of-Freedom Compensatory Tracking in 3D and the Effects of Practice. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 54(28), 2432–2436.

Vera, P., Luviano, A., Santos-Cuevas, L. & Chairez, I. (2017). Trajectory tracking adaptive disturbance rejection controller for a tomographic robotic system. *2017 4th International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 0377–0382.

Villani, L. & De Schutter, J. (2016). Force Control. In Siciliano, B. & Khatib, O. (Eds.), *Springer Handbook of Robotics* (pp. 195–220). Cham: Springer International Publishing. doi: 10.1007/978-3-319-32552-1_9.

Wang, H. O., Tanaka, K. & Griffin, M. F. (1996). An approach to fuzzy control of nonlinear systems: Stability and design issues. *IEEE transactions on fuzzy systems*, 4(1), 14–23.

Wang, S. & Fei, J. (2014). Robust adaptive sliding mode control of MEMS gyroscope using T–S fuzzy model. *Nonlinear Dynamics*, 77(1-2), 361–371.

Wang, Y., Mai, T. & Mao, J. (2014). Adaptive motion/force control strategy for non-holonomic mobile manipulator robot using recurrent fuzzy wavelet neural networks. *Engineering Applications of Artificial Intelligence*, 34, 137 - 153.

Wang, Z., Mastrogiacomo, L., Franceschini, F. & Maropoulos, P. (2011). Experimental comparison of dynamic tracking performance of iGPS and laser tracker. *The international journal of advanced manufacturing technology*, 56(1-4), 205–213.

Wen, S., Yu, H., Zhang, B., Zhao, Y., Lam, H.-K., Qin, G. & Wang, H. (2017). Fuzzy identification and delay compensation based on the force/position control scheme of the 5-DOF redundantly actuated parallel robot. *International Journal of Fuzzy Systems*, 19(1), 124–140.

Wu, H. & Huang, J. (2019). Control of Induction Motor Drive based on ADRC and Inertia Estimation. *2019 IEEE International Electric Machines & Drives Conference (IEMDC)*, pp. 1607–1612.

Wu, Z.-H. & Guo, B.-Z. (2018). Approximate decoupling and output tracking for MIMO nonlinear systems with mismatched uncertainties via ADRC approach. *Journal of the Franklin Institute*, 355(9), 3873–3894.

Xia, Y. & Fu, M. (2013). *Compound control methodology for flight vehicles*. Springer.

Xu, W., Cui, J., Li, L., Yao, B., Tian, S. & Zhou, Z. (2020). Digital twin-based industrial cloud robotics: Framework, control approach and implementation. *Journal of Manufacturing Systems*. doi: 10.1016/j.jmsy.2020.07.013.

Xue, W., Madonski, R., Lakomy, K., Gao, Z. & Huang, Y. (2017). Add-on module of active disturbance rejection for set-point tracking of motion control systems. *IEEE Transactions on Industry Applications*, 53(4), 4028–4040.

Xue, W., Zhang, X., Sun, L. & Fang, H. (2020). Extended State Filter based Disturbance and Uncertainty Mitigation for Nonlinear Uncertain Systems with Application to Fuel Cell Temperature Control. *IEEE Transactions on Industrial Electronics*.

Yang, P., Guo, Z. & Kong, Y. (2020). Plane kinematic calibration method for industrial robot based on dynamic measurement of double ball bar. *Precision Engineering*, 62, 265–272.

Yang, Y. & Jie, T. (2017). One-DOF link manipulator control through active disturbance rejection approach. *2017 36th Chinese Control Conference (CCC)*, pp. 1028–1032.

Yang, Y., Tan, J. & Yue, D. (2018). Prescribed performance control of one-DOF link manipulator with uncertainties and input saturation constraint. *IEEE/CAA Journal of Automatica Sinica*, 6(1), 148–157.

Yen, V. T., Nan, W. Y. & Van Cuong, P. (2019). Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators. *Neural Computing and Applications*, 31(11), 6945–6958.

Ying, H. (1999). Analytical analysis and feedback linearization tracking control of the general Takagi-Sugeno fuzzy dynamic systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29(2), 290–298.

Ying Bai, Hanqi Zhuang & Roth, Z. S. (2005). Fuzzy logic control to suppress noises and coupling effects in a laser tracking system. *IEEE Transactions on Control Systems Technology*, 13(1), 113-121. doi: 10.1109/TCST.2004.833653.

Zadeh, L. A. (1973). Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on systems, Man, and Cybernetics*, (1), 28–44.

Zhang, B., Wu, J., Wang, L. & Yu, Z. (2020). Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot. *Robotics and Computer-Integrated Manufacturing*, 63, 101923.

Zhang, H. (2017). *Information driven control design: a case for PMSM control*. (Ph.D. thesis, Cleveland State University).

Zhang, S., Wang, S., Jing, F. & Tan, M. (2019). A Sensorless Hand Guiding Scheme Based on Model Identification and Control for Industrial Robot. *IEEE Transactions on Industrial Informatics*, 15(9), 5204-5213. doi: 10.1109/TII.2019.2900119.

Zhao, R. (2015). *Trajectory planning and control for robot manipulations*. (Ph.D. thesis).

Åström, K., Hägglund, T., Hang, C. & Ho, W. (1992). Automatic Tuning and Adaptation for PID Controllers - A Survey. *IFAC Proceedings Volumes*, 25(14), 371 - 376. doi: https://doi.org/10.1016/S1474-6670(17)50762-4. 4th IFAC Symposium on Adaptive Systems in Control and Signal Processing 1992, Grenoble, France, 1-3 July.