Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers

Lencho Duguma Fufa

A Thesis Submitted to

The Department of Electrical Power and Control Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Electrical Power and Control Engineering (Control Engineering)

Office of Graduate Studies

Adama Science and Technology University

June 2022

Adama, Ethiopia

Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers

Lencho Duguma Fufa
Advisor: Dr. Endalew Ayenew

A Thesis Submitted to
The Department of Electrical Power and Control Engineering
School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Electrical Power and Control Engineering (Control Engineering)

Office of Graduate Studies
Adama Science and Technology University

June 2022

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Lencho Duguma Fufa _____ _____

Name of Student Signature Date

RECOMMENDATION

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled "Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers," prepared under my guidance by Lencho Duguma Fufa, submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Control Engineering. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Endalew Ayenew _____ _____ _____

Major Advisor Signature Date

APPROVAL SHEET

I, the advisors of the thesis entitled "Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers" and developed by Lencho Duguma Fufa, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Endalew Ayenew		
Major Advisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Lencho Duguma Fufa, have read and evaluated the thesis entitled "Trajectory Tracking of a Two-Wheeled Mobile Robot Using Backstepping and Nonlinear PID Controllers" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Control Engineering.

Chairperson	Signature	Date
Professor Gang Gyoo Jin		June 22, 2022
Internal Examiner	Signature	Date
Dr. Lebsework Negash		June 23/2022
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon the submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department of Graduate Council (DGC) and School Graduate Committee (SGC).

Department Head	Signature	Date
School Dean	Signature	Date
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to the Almighty God for giving me strength in this difficult situation to complete this thesis journey. Without his help, all these achievements would not be imaginable.

I would also like to express my deepest gratitude to my advisor and mentor, Dr. Endalew Ayenew, for his invaluable advice, inspiration, support, and continued encouragement to make this thesis work possible and fruitful.

Last but not least, I would like to express my gratitude to my family, friends, colleagues, and others who directly or indirectly assisted me during the completion of this work. It would be a blessing to have you around during this time.

Thank you.

Lencho Duguma Fufa

TABLE OF CONTENTS

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ASTU	Adama Science and Technology University
BSC	Backstepping Controller
CoM	Center of Mass
DC	Direct Current
2DOF	Two Degree of Freedom
3DOF	Three Degree of Freedom
DWMR	Differential-drive Wheeled Mobile Robots
GA	Genetic Algorithm
IAE	Integral Absolute Error
ISE	Integral Square Error
ITAE	Integral Time-Weighted Absolute Error
MR	Mobile Robot
MIMO	Multi-Input Mult-Output
NN	Neural Network
NPID	Nonlinear Proportional, Integral, and Derivative
PID	Proportional, Integral, and Derivative
PSO	Particles Swarm Optimization
SMC	Sliding Mode Controller
TWMR	Two-Wheeled Mobile Robot
WMR	Wheeled Mobile Robot

ABSTRACT

Many researchers have become interested in wheeled mobile robot (WMR) trajectory tracking control due to the increased application of mobile robots over the past three decades in the industry, in the military, at home, and in public service. Classically, the movement of WMR is controlled depending on its kinematic model. In practical applications, both the dynamic and kinematic models of robots and external disturbance and uncertainty affect system performance. In this research, a backstepping controller combined with a nonlinear proportional, integral, and derivative (NPID) controller is proposed for the trajectory tracking of a two-wheeled mobile robot (TWMR). The kinematic and dynamic models of the WMR are considered. The dynamic modeling was derived using a Lagrangian approach, and the stability of the system was achieved using the Lyapunov stability function. A backstepping controller is used for the position control and the NPID controller for the velocities of the robot. The proposed controller gains are optimized using a genetic algorithm (GA). The overall system with the proposed method is simulated on Matlab/Simulink software. The results show the best trajectory tracking performance in the presence of unknown disturbances and initial position change with a minimum error. In addition to this, the results show that the proposed controller outperforms the GA-based BSC+PID controller in terms of root-mean-square (RMS) of trajectory tracking error (47.36% in a linear and 60.32% in a nonlinear reference trajectory). The proposed controller also has better tracking performance than a GA-based BSC controller (which means trajectory tracking error is improved by 74.35% in the linear reference trajectory and 67.95% in the nonlinear reference trajectory). Finally, trajectory tracking of TWMR using backstepping and an NPID controller was performed effectively in the presence of unknown disturbance and initial position change.

Keywords: *Backstepping and NPID Controller, Trajectory Tracking, Two-Wheeled Mobile Robot, Genetic Algorithm, Lyapunov Stability Analysis.*

CHAPTER 1

1. INTRODUCTION

1.1. Background of Mobile Robot

A mobile robot is one of the robotic vehicles that can move its surroundings and is not fixed to a given physical location. It can be autonomous or semi-autonomous, which means it can navigate in an uncontrolled environment without needing a personal operator or electro-mechanical guidance device. A wheeled mobile robot (WMR) is a mobile robot that navigates its surroundings using a wheel's rotation. It is widely used because it can substitute humans in many fields. For instance, in industry, the military, public and private sector services, and manufacturing and distribution firms (Cen & Singh, 2021).

When the application of a WMR increases, the performance acquired from the robot becomes a critical issue for researchers. The robot's performance (in the case of trajectory tracking) depends on the position error and orientation error for kinematic control, while a velocity error will affect the performance in the case of dynamic control. The movement or navigation of a WMR depends on three control problems: reference trajectory tracking, line or path following, and point stabilization (Fierro & Lewis, 1997):

The path following problem: is a simple problem that requires only the angular velocity change to decrease the distance between a given position and the current robot position, i.e., the path is time-independent.

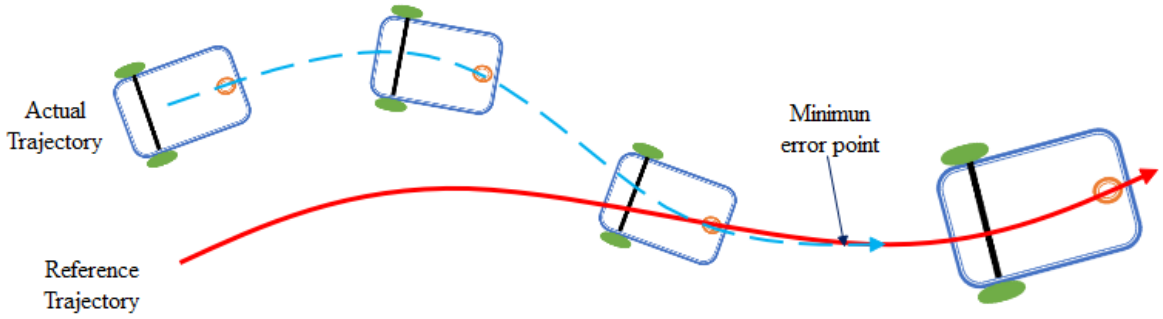

Figure 1.1: An example of a path or line tracking task

Reference trajectory tracking problem: The robot follows a track based on a given trajectory. The predefined trajectory is time-dependent in this problem and needs a control law to predict its future behavior. This navigation problem is widely used in mobile robot trajectory tracking control applications.

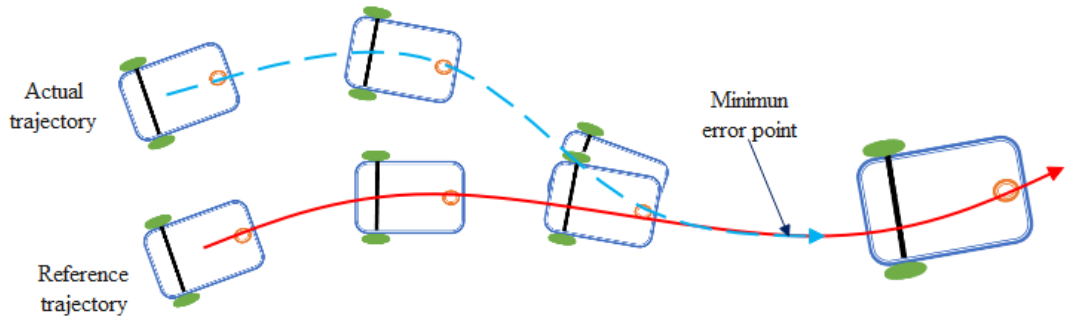

Figure 1.2: An example of a reference trajectory tracking task

Posture stabilization problem: In this case, the system tries to reach the desired posture goal along any path. This tracking problem is challenging if there is an obstacle in its way.

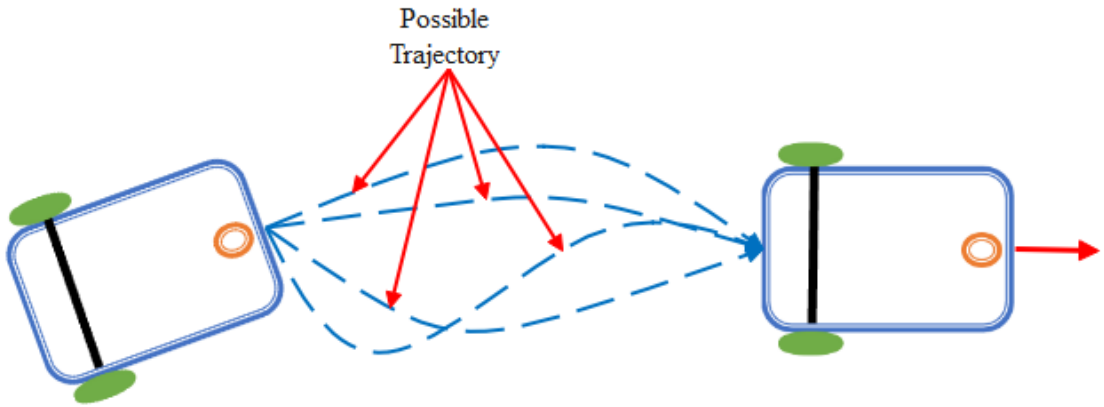

Figure 1.3: An example of a posture stabilization task

One of the main objectives of reference tracking is to control a robot's position on a predetermined trajectory with the minimum position and orientation error. Due to nonholonomic restrictions in nonholonomic robots, trajectory tracking and motion control are not independent. Trajectory tracking of nonholonomic robots in the robotic domain involves determining trajectories from a starting point to a final point while considering mechanical limitations and guiding the robot to follow the proposed trajectories.

Several papers on the WMR's trajectory control problem have been presented. Most of them are focused on a nonholonomic constraint that depends on posture stabilization and trajectory tracking because path following is a simple and time-function independent trajectory (Uddin, 2018). Posture stabilization intends to stabilize the robot at a reference position, whereas trajectory tracking makes the robot follow a predefined trajectory.

The two-wheeled mobile robot (TWMR) used in this work has two standard fixed wheels actuated by two similar DC motors and a free castor wheel. The freewheeler is used to

balance the robot's body frame. Many control methods have been applied to control a WMR trajectory tracking, for example, a kinematics-based backstepping controller (Ren et al., 2017), PID controller (Chang & Jin, 2013), Fuzzy logic controller (Moqbel Obaid et al., 2016; Q. Xu et al., 2014), Nonlinear PID controller. Furthermore, various optimization techniques (like; Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Neural Network (NN)) are used to optimize the controller gains.

In this work, a backstepping controller combined with a nonlinear proportional, integral and derivative (NPID) controller is proposed for trajectory tracking control of a WMR. The proposed control method includes two control loops: outer and inner loops. The outer loop controller is used to control the robot's position, while the inner loop controller controls its velocity. A backstepping controller was proposed for the outer loop, while an NPID controller was used for the inner loop control. The backstepping plus NPID controller tracking performance will be compared with that of a backstepping controller combined with a PID controller and a backstepping-only controller. In the case of the proposed controller's capability, smoothness, and robustness tests, unknown disturbances are considered.

1.2. Statement of the Problem

This work includes developing a nonholonomic TWMR trajectory tracking control using a backstepping controller combined with an NPID controller. In wheeled mobile robotics, trajectory tracking is essential for navigation control problems. Solving and working on such a problem needs significant effort to achieve better performance from a system. Hence, trajectory tracking aims to maintain a robot on a reference trajectory by controlling its position and orientation, i.e., the error between the reference and actual positions converges to zero as time goes to infinity. However, the trajectory tracking the performance of TWMR, which is presented in the previous work, is less efficient, and the tracking error is not as small as desired. The reason is that only the kinematics model for the trajectory tracking of a TWMR is considered and neglects velocity control that affects the position of the system. In addition to this, the effects of unknown disturbance must also be considered.

Therefore, trajectory tracking is an attractive and interesting area for research to study the effects of the above problem. Thus, the trajectory tracking and velocity control of WMR are investigated. A backstepping controller combined with an NPID controller is presented in this study to overcome the above problem in the trajectory tracking control of a TWMR.

1.3. Objectives

1.3.1. General Objective

The main goal of this thesis work is to control the trajectory tracking of a TWMR using a backstepping plus NPID controller.

1.3.2. Specific Objectives

The specific objectives to achieve this thesis work are;

- Model the kinematics and dynamics models of a nonholonomic TWMR,
- Generate the reference trajectory for TWMR,
- Design a backstepping with an NPID controller for trajectory tracking of TWMR,
- Apply the GA optimization solver to optimize controller parameters for trajectory tracking of TWMR,
- Simulate the overall system using MATLAB/SIMULINK software to analyze the trajectory tracking performance of the proposed controller.

1.4. Significance of the Study

The WMR is an autonomous mobile robot powered by a DC motor. Such a type of robot uses an electric energy source (specifically a battery), which plays a vital role in decreasing the environmental pollution. Many organizations, particularly warehouses or distribution companies, employ carts or forklifts to deliver loaded items from one area to another. A cart, for example, is difficult for the elderly and disabled to operate. Similarly, a forklift, or a small vehicle employed in a warehouse or distribution company, uses gasoline as its energy source.

However, this has a side effect. Using fuel as an energy source increases environmental pollution and costs. Moreover, it makes the work environment less safe. As a result, the suggested system uses electric energy sources and provides safety for all users. The cost required for electricity is much lower than the cost of fuel energy.

1.5. Scope of the Study

This thesis work focuses on the control of trajectory tracking of a differential drive type WMR using backstepping combined with an NPID controller. The GA optimization tool is used to optimize the controller gain parameter. In addition to this, a mathematical model (kinematics and dynamics models) has been derived for a two-wheel differential-drive

mobile robot with one passive wheel. The derived mathematical model of the system is executed using MATLAB/SIMULINK (R2021a) software.

1.6. Limitations of the Study

This thesis work is limited to the trajectory control of a two-wheeled differential-drive mobile robot with a single passive wheel. Furthermore, mathematical models of a robot were developed by considering the steering of two wheels. i.e., no rolling, skidding, or neglecting the effect of the passive wheel on system dynamics. There is no hardware implementation because the cost of acquisition, timing, and component availability are also challenges for hardware implementation. The system is executed using MATLAB/SIMULINK (R2021a) software.

1.7. Thesis Organization

This work can be organized as follows:

Chapter 1: Introduction

- This chapter discusses the introduction of trajectory tracking of mobile robots, the thesis problem of a statement, objectives, significance, and limitations.

Chapter 2: Literature Review

- Discuss previous related work in WMR trajectory tracking.

Chapter 3: TWMR System Modeling and Methodology

- This chapter discusses the system modeling of TWMR.

Chapter 4: Controller Design

- This chapter covers the design of trajectory generation and control methods.

Chapter 5: Results and Discussion

- This chapter will present closed-loop system simulation, results, and discussion.

Conclusion and Recommendation:

- A conclusion and future work recommendations for the researcher are provided.

CHAPTER 2

2. LITERATURE REVIEW

2.1. Overview

This chapter introduces a brief discussion on mobile robots, including a WMR and the types of wheels used in WMR applications. Various related works are reviewed and discussed, along with their benefits and drawbacks.

2.2. Mobile Robot

Many researchers and academicians have recently been interested in mobile robots (MR). This is due to the robot replacement of personnel and standard transport methods in industry, military, health, communication, public and private sector service, a distribution company, and transportation. It is a widely used and studied robotics application because of its ability to navigate in an uncontrolled environment.

A mobile robot is an autonomous or semi-autonomous robotic vehicle that navigates on the ground, in the air, or underwater using wheels, legs, tracks, wings, or hybrids. Its navigation mechanisms can mainly walk, roll, leap, run, slide, skate, swim, or fly, depending on the environmental interaction. Mobile robots can be categorized into different groups depending on their application, navigation, automation, and locomotion. Based on their movement, mobile robots can be classified as follows (Rubio et al., 2019):

1. Land-based
    - Wheeled mobile robots
    - Mobile robots that walk or have legs
    - Mobile Robots with Tracked Slip/Skid Locomotion
    - hybrid mobile robots.
2. Air-based
    - Unmanned-Aerial Vehicles
3. Water-based
    - Unmanned-Underwater Vehicles

A legged mobile robot is a mobile robot that is used in rough terrain environments where walking is usually preferred. In contrast, WMRs are used on smoother surfaces and less terrain because they are faster and more agile. A WMR is more sensitive to obstacles than a

legged mobile robot. Unmanned-Aerial Vehicles (UAVs) or Unmanned Underwater Vehicles are used when we need to operate in three-dimensional spaces.

2.3. Wheeled Mobile Robot (WMR)

Autonomous WMRs are mobile robots that employ wheels for mobility. A WMR plays a critical role in transportation, logistics, and distribution companies. It travels on a smooth surface and non-rugged terrain where employing wheels is possible. A WMR is easier to design, build, and program than a tracked or legged mobile robot and is less expensive and less complicated. Another advantage of WMR is that it does not impose any significant balancing challenges because the robot is usually in contact with the surface. However, WMRs are less effective in traveling through obstacles like rocky terrain, sharp surfaces, or low-friction zones. The WMR can be differential drive, car type, omnidirectional, or synchro drive, depending on its drive system (Rubio et al., 2019).

2.3.1. Differential Drive WMR

A differential-drive WMR comprises two standard fixed-powered wheels mounted on the robot platform's left and right sides. The two wheels are propelled independently. For balance and stability, one or two passive castor wheels are used. The differential drive is the most basic mechanical drive system because it does not require the rotation of the driven axis. Depending on the signs of the motor speeds, the robot moves straight forward, backward, and turns right or left (Tzafestas, 2014).

- If the wheels actuated at the same speed, the robot moves straight forward or backward, i.e., it depends on the sign of motor speeds.
- If one wheel is spinning faster than the other, the robot is at the turning point.
- If all two wheels are spinning at the same speed but have opposite signs, the robot will revolve around the midpoint.

Figure 2.1: An example of a differential drive WMR (Dong et al., 2014)

2.3.2. Types of Wheels

Four types of wheels are utilized in WMR applications (Dong et al., 2014):

1. Fixed conventional wheel,
2. Castor wheel,
3. Swedish wheel and,
4. Ball or spherical wheel.

The fixed standard and castor wheel are directional because they rotate around a single axis. The main difference is that the conventional wheel can steer without side effects, whereas the castor wheel rotates around an offset axis and imparts force to the robot chassis during steering. Figure 2.2 shows an example of wheels used in mobile robots (Tzafestas, 2014).

A. Fixed standard wheel B. Castor wheel C. Swedish wheel

Figure 2.2: Wheels employed in mobile robot

Swedish and spherical wheels have less directionality than standard wheels. The Swedish wheel operates similarly to a fixed standard wheel but provides low resistance in the opposite direction. On the other hand, the Swedish wheel can kinematically move despite being propelled only along one principal axis, which is a significant advantage of this design.

2.4. Related Works

In recent times, many researchers have proposed a control mechanism to control the trajectory of a mobile robot using different control techniques. These control techniques were based on linear or nonlinear control ideas. The trajectory tracking control of a mobile robot depends on tracking control problems like the reference trajectory problem and a posture stabilization problem, as mentioned in the previous chapter.

This thesis work concentrated on the trajectory tracking of TWMR using a backstepping controller combined with an NPID controller. In order to get the best values for the controller gains, genetic algorithm (GA) optimization methods are used. In the following sections, previous related work will be reviewed and discussed in addition to the gap obtained.

2.4.1. Trajectory Tracking Control Using a Backstepping Controller

A backstepping controller for MR trajectory tracking control was presented in the literature (Hassani et al., 2020). This work presents the kinematic and dynamic control of a WMR separately. The controller gains are adjusted using GA optimization. The result shows that better trajectory tracking is achieved in the circular, sinusoidal, and lemniscate reference trajectory. The robot will perform poorly if the velocity and mass of the system are changed because the dynamic model of a system affects the system's performance in a real-time application.

A feedback linearization-based control method for trajectory tracking of WMR is introduced (Benchouche et al., 2021). A control method based on feedback linearization is used for velocity control, while a backstepping controller is used for position control. The obtained results show the effects of the dynamic model on system performances. However, the proposed method's capability to handle the error at the start of a simulation is less. i.e., there is more overshoot. Suppose the robot's initial position changes; bringing it back to its reference trajectory is challenging. In addition to this, the proposed method cannot be tested in the presence of an unknown disturbance environment.

2.4.2. Trajectory Tracking Control Using an NPID Controller

In a paper published by (Mohamed & Hamza, 2019), an NPID (Proportional-Integral-Derivative Neural Network) controller was developed for the differential drive WMR (DWMR) trajectory tracking issue. The proposed controller is applied to the mobile robot's kinematic model, and the PID-NN parameters are tuned by the particle swarm optimization method. The obtained result is robust and gives good performance with fewer tracking errors. However, the mobile robot control mechanism should consider the kinematic and dynamic models because, in real applications, system dynamics affect the robot's performance.

2.4.3. Trajectory Tracking Control Using Backstepping Controller Combined with Other Controllers

An NPID controller is proposed in the literature (KALYONCU & DEMİRBAŞ, 2017; Zangina et al., 2020) for the trajectory tracking of robots. A proposed control algorithm uses a kinematic-based controller to control robot coordinates and a PID controller for motor speed control. The proposed method achieves better trajectory tracking. However, getting a controller parameter by trial and error takes more time and does not give better performance when compared with optimization techniques.

Yousuf et al. introduced backstepping combined with a PID controller for trajectory tracking problems. The kinematic and dynamic models of the framework are derived and modeled. The performance of a proposed method is tested with three trajectories: circular, infinity, and straight. The system is stable and follows its desired trajectory with a minimum error in all cases. However, the limitation of this approach is the lack of optimization (Yousuf et al., 2021).

Mai et al. presented a backstepping and adaptive fuzzy PID controller for the trajectory control of autonomous MR. A PID controller is used for the system's velocity and steering angle control, while the backstepping controller is responsible for the position. A fuzzy controller guarantees the adaptability of the PID controller. The obtained results provide a better tracking response at infinity reference input than the BSC combined PID controller. However, the performance of the proposed system in the presence of unknown disturbances and uncertainties is not evaluated (Mai et al., 2021).

Xu et al. presented a combined backstepping with a fractional-order PID controller for trajectory tracking of MR (L. Xu et al., 2022). This work describes kinematic and dynamic models of the system with skidding and sliding. A beetle swarm optimization algorithm is applied for controller gain optimizations. The obtained response is compared with a backstepping combined with a PID controller, and the proposed algorithm has a better tracking response. However, the proposed method has a significant tracking error; if an unknown disturbance and system uncertainty occurs, the system performs less.

In general, the following tracking control approaches are widely used in mobile robots (Yildirim & Savaş, 2013): state feedback linearization, sliding-mode control (SMC), backstepping control (BSC), computed torque, adaptive control, and intelligent control. When the stability of the tracking control algorithm, the calculated load of hardware, and operability in the actual application are all taken into account in these approaches, the backstepping control strategy becomes one of the most preferred. However, when the mobile robot has an initial position change or reference trajectory change, this control technique causes substantial velocity changes, i.e., the velocities in the system change quickly. In practice, this is not easy to realize.

Better trajectory tracking control of WMR is achieved by considering the disturbances, noise, or internal model changes (uncertainty) that will affect the system in real-time applications. Including a WMR system dynamic model is vital for good trajectory tracking and stabilization. The contribution of this work is the design of a backstepping controller

combined with an NPID controller for stabilization and trajectory control of TWMR. The proposed strategy consists of two approaches. A backstepping controller controls the robot's positions, and NPID controls the robot's velocity. The Lyapunov function justifies the stability of the system. Kinematic and dynamic models of a proposed framework are considered. Additionally, the controller parameters are also optimized using a GA optimization tool. The system's performance is analyzed with unknown disturbances and different initial positions.

Table 2.1: Summary of related works

Authors	**Title**	**Method**	**Achieved**	**Gap**
Mohamed and Hamza (2019)	*"Design PID neural network controller for trajectory tracking of differential drive mobile robot based on PSO"*	PID and NN controller with PSO optimization applied to a kinematic model of the system.	Better tracking response for infinity and star reference trajectory tracking	Using only a kinematic model of the system and ignoring velocity control of the robot
Hassani et al. (2020)	*"Backstepping tracking control for nonholonomic mobile robot"*	Kinematic control for trajectory tracking (BSC) and dynamic control for real-time robot control	Trajectory tracking is achieved in circular, sinusoidal, and lemniscate reference trajectory	The results are not tested for the presence of disturbances and uncertainties
Yousuf et al. (2021)	*"Dynamic modeling and tracking for nonholonomic mobile robot using PID and backstepping"*	PID controller for velocity control and position control via backstepping	The response is stable and follows a reference path in circular, infinity, and straight paths	Use trial and error to get a gain parameter
Mai et al. (2021)	*"A combined backstepping and adaptive fuzzy PID approach for trajectory tracking of autonomous mobile robots"*	Backstepping and adaptive fuzzy PID controllers are applied for position and velocity control.	The obtained results give a better tracking response on the infinity reference input	The results are not tested for the presence of disturbances and uncertainties
Xu et al. (2022)	*"A combined backstepping and fractional-order PID controller to trajectory tracking of mobile robots"*	BSC and FOPID controller with a beetle swarm optimization is used, and sliding and skidding are considered in system modeling	The proposed system has a better tracking response than BSC combined with a PID controller.	The results are not tested for the presence of disturbances and uncertainties

CHAPTER 3

3. TWMR SYSTEM MODELING AND METHODOLOGY

3.1. Overview

This chapter will give details of the proposed system models and modeling, the procedures followed, and the materials used for report writing and result analysis. The mathematical model derivation, representation, controllability, and an open-loop systems simulation analysis will be presented.

3.2. Materials Used

This research uses MATLAB/R2021a, Microsoft Office (MS) 2016, MathType 7.0, and Mendeley Desktop software tools. The MATLAB software consists of a technical toolbox and Simulink, which is used to simulate the system. Microsoft Office is used to edit the thesis documentation and presentation, and MathType 7.0 is used to write mathematical formulas and equations. Mendeley Desktop is for organizing and inserting references, citations, and bibliographies into a document.

3.3. Methods or Procedures Followed

The following procedure, shown in Figure 3.1, will be used throughout this thesis to accomplish a proposed objective. The related papers and literature are selected and reviewed for a WMR control mechanism depending on the selected title and identified problem.

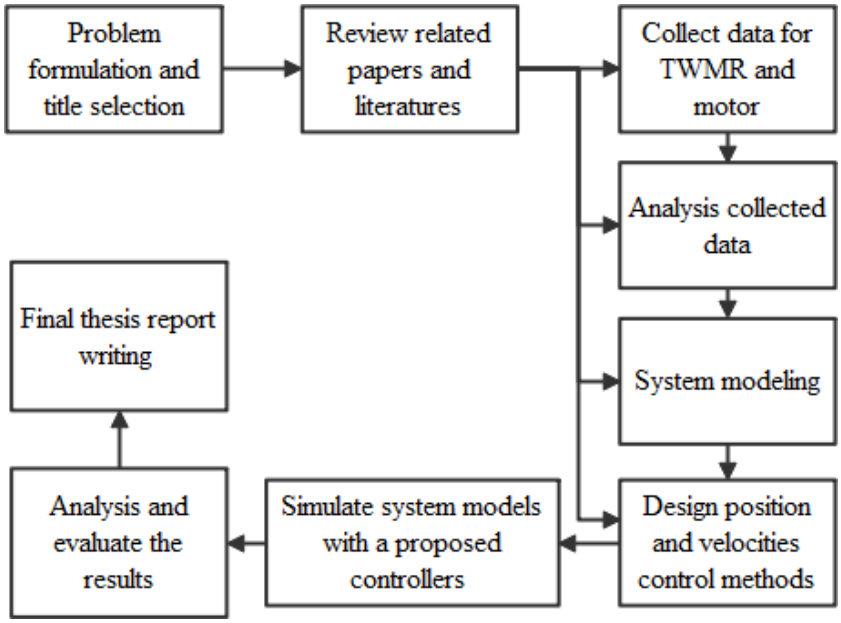

Figure 3.1: Proposed workflow or procedures

All the data was gathered from secondary sources via online data access with the help of a skilled person and a researcher. The collected data from selected literature is evaluated and analyzed. Based on data collected for a TWMR, the kinematics and dynamics of the system modeling were developed. A control law for the position and velocity of a system was proposed for developed WMR models. i.e., a backstepping controller is used to control the position of a robot, and an NPID controller will be used to control the velocity of a WMR robot.

The proposed system models with a proposed controller were finally simulated on MATLAB/Simulink software to test and analyze the controller's performance. Finally, the thesis report was organized and documented.

3.4. The Block Diagram of TWMR with the Proposed Control Method

This work focused on the trajectory tracking of TMWR. The proposed system has two standard fixed wheels that are actuated by two similar DC motors and one free castor wheel to balance the robot body. i.e., the right wheel is actuated by a right motor, and the other motor also actuates the left wheel. Each motor is controlled separately to achieve a differential drive mechanism. Trajectory tracking control of TWMR aims to eliminate the trajectory tracking error to zero, i.e., the tracking error signal goes to zero when time goes to infinity. As shown in Figure 3.2, a two-loop control method is used to control the robot's position and velocity.

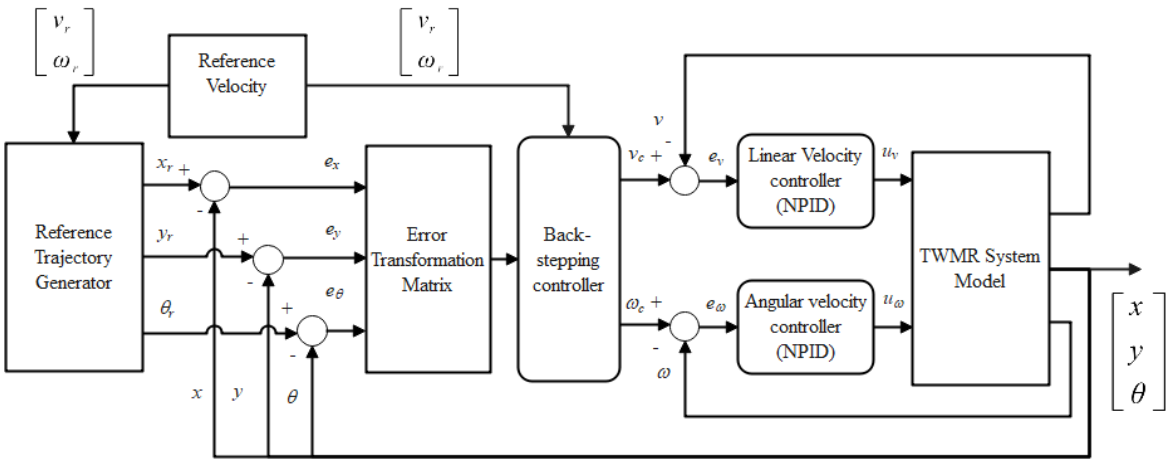

Figure 3.2: Proposed Block diagram of TWMR

The positions of a mobile robot are represented by two coordinates (x and y positions) with a heading angle of theta (θ) in radian. The trajectory generator generates a reference trajectory depending on the reference velocity inputs (linear and angular velocity). The error

transformation matrix transforms a robot's previous error signals into the current position, and a backstepping controller will control these transformed tracking errors. The BSC controller generates a velocity control input as a function of tracking errors and a reference velocity input. A generated velocity control input is used to stabilize the dynamic model of TWMR.

In addition to this, an NPID controller is used to control the velocity of a system. It generates a command control by comparing a velocity control input from a backstepping controller with the actual velocity of the system. A system with a proposed control law will have its stability guaranteed by the Lyapunov stability function. Nonlinear feedback linearization techniques are also applied to linearize the system. System dynamics and kinematic modeling are derived and explained briefly in the next topic.

3.5. Modeling and Description of the TWMR System

System modeling of a differential-drive mobile robot platform in this work includes kinematic and dynamic modeling as well as modeling of the system actuators. For a system, kinematic modeling is concerned with the geometric relationships and mathematics of motion without considering influencing forces. On the other hand, dynamic modeling is the study of motion in which forces and energy are represented and included, while actuator modeling is required to determine the interaction between the control signal and the mechanical system's input. A derivation of an appropriate system model makes a system more understandable and controllable.

3.5.1. Representing Robot Coordinate

The primary function of the coordinate systems is to represent the robot's position. There are two coordinate systems that are widely used for system model development: Cartesian and Polar coordinates. Hence, because of the ease and simple representation of Cartesian coordinates, system modeling is done in Cartesian coordinates. The following cartesian coordinate systems are used for mobile robot modeling and control purposes:

- ✓ *The inertial reference frame (X_I, Y_I):* This is a fixed coordinate system in the plane of the robot. It defines a random inertial base on the plane as the global reference frame from some origin.
- ✓ *Robot frame (X_R, Y_R):* This is the coordinated system attached to the robot's body frame. It defines two axes relative to the robot chassis.

To specify the robot's position on the coordinate configuration frame, the relationship between the global frame and the robot's local frame is determined, as shown in Figure 3.3.

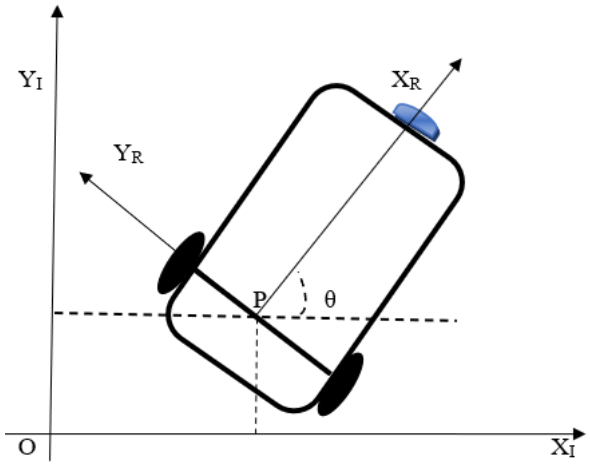

Figure 3.3: Inertial and robot frame coordinate representation

The axes X_I and Y_I are an arbitrary inertial frame on the plane as the global reference frame from the origin *O*. The basis (X_R, Y_R) is two local robot frame axes relative to *P* on the robot chassis. The position of P in the global reference frame is specified by the coordinates *x* and *y*, and the angular difference between the global and local frames is given by θ. The essential issue in a coordinate system representation is to map the motion along the axes of the global reference frame to the motion along the axes of the local robot frame. The inertial frame and the robot frame can be represented as follows (Ben Jabeur & Seddik, 2021);

$$\begin{cases} q_I = \begin{bmatrix} x_I & y_I & \theta_I \end{bmatrix}^T \\ q_R = \begin{bmatrix} x_R & y_R & \theta_R \end{bmatrix}^T \end{cases} \tag{3.1}$$

The mapping of the two frames can be done through a standard orthogonal transformation matrix. This matrix is used to map the motion of the global reference frame (X_I, Y_I) to the movement of the local robot frame (X_R, Y_R) as follows:

$$\dot{q}_R = R(\theta)\,\dot{q}_I \tag{3.2}$$

The local robot frame (X_R, Y_R) to the global reference frame (X_I, Y_I) was mapped as:

$$\dot{q}_I = R^{-1}(\theta)\,\dot{q}_R \tag{3.3}$$

$$R(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where $R(\theta)$ is an orthogonal transformation matrix. A relationship shown in Equation (3.2) is also helpful for robot velocities in a robot, and inertial frames, which is an essential task in the kinematic equations drive of a TWMR in the next topic.

3.5.2. Kinematic Modeling of TMWR

Robot kinematics modeling is concerned with the spatial configuration of robots, the relationships between their geometric parameters, and the constraints imposed on their trajectories. The geometrical structure of the robot determines the kinematic equations. A mobile robot, for example, can have one or more wheels with or without motion constraints. The concept of a kinematic model is helpful in understanding system dynamics, stability, and control methods.

A kinematic equation establishes the robot's speed $\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T$ as a function of wheels speed $(\dot{\varphi}_r, \dot{\varphi}_l)$ and robot geometric parameters can be categorized as follows;

- ✓ Forward kinematics from:

$$\dot{q} = f(\dot{\varphi}_r, \dot{\varphi}_l, L, r, \theta) \tag{3.5}$$

- ✓ Inverse kinematics form:

$$[\dot{\varphi}_r \quad \dot{\varphi}_l]^T = f(\dot{x}, \dot{y}, \dot{\theta}) \tag{3.6}$$

3.5.2.1. The Forward Kinematics Model of Differential Drive WMR

A differential drive of WMR is shown in Figure 3.4, which has two wheels with a radius of *r* and a distance *L* from the center *P*, and free castor wheels to balance the robot setup body. The parameters that are used to derive kinematic and dynamic models are presented as follows (Martins & Bertol, 2022):

- *L*: The distance between the actuated wheel and the axis of symmetry in the y-direction,
- *C*: is the center of mass (CoM) at the center of gravity of the robot platform,
- *d*: The distance between the CoM and the driving wheel axis.
- *P*: The intersection of the axis of symmetry with the driving wheel axis,
- *r*: The radius of each wheel,
- m_c: a mass of WMR devoid of wheels and motors,
- m_w: The combined mass of each wheel and motor,
- m_t: The total mass of the robot,

- *Ic*: The inertia of WMR without wheels and motor about the y-axis through P,
- *Iw*: The inertia of each wheel and motor around the wheel axis,
- *Im*: The inertia of each wheel and motor about the y-axis is parallel to the wheel plane,
- *I*: total inertia of the robot,
- φ_r, φ_l: angular displacement of the right and left wheels,
- $\dot{\varphi}_r, \dot{\varphi}_l$: Angular velocity of the right and left wheels,
- v, ω: The robot's linear and angular velocity,
- w : velocity vector of linear and angular velocity.

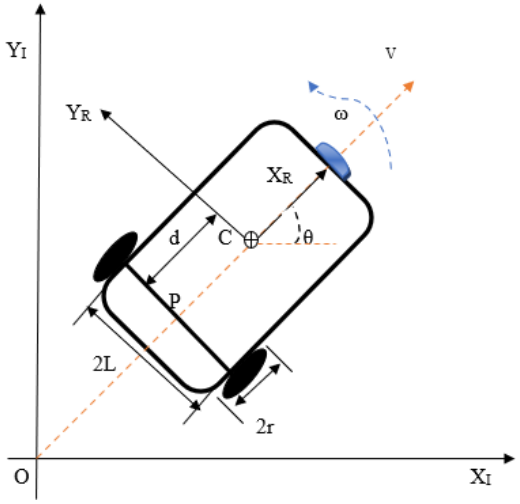

Figure 3.4: 2D velocity coordinate configuration of TWMR

From Equation (3.5) a forward kinematic model of DWMR is a function of the robot's wheel velocity and geometric parameters, representing the robot's velocities in an inertial frame. The velocity coordinate represents the kinematic model problem, as shown below (Benchouche et al., 2021):

$$\dot{q}_I = [\dot{x}_I \quad \dot{y}_I \quad \dot{\theta}_I]^T = f(\dot{\varphi}_r, \dot{\varphi}_l, L, r, \theta) \tag{3.7}$$

From Equations (3.2) and (3.3), we can compute the robot motion in a global reference frame on a local robot frame and vice versa. Therefore, to calculate the contribution of each wheel in a movement of the robot in the local robot frame $\dot{q}_R$ the following methods are employed (Siegwart R, Nourbakhsh IR, 2011).

In Figure 3.4, suppose that a robot moves forward along the positive *x*-plane of the local frame in alignment with a local robot frame. First, let us calculate the contribution of each wheel's rotating speed to the translation speed at point *P* in the direction of the positive *x*-plane of the local frame. Suppose one wheel spins while the other wheel is stationary, then

the robot moves instantaneously at half speed since *P* is the halfway point between two wheels, that is:

$$\begin{cases} \dot{x}_{R1} = \frac{1}{2} r \dot{\varphi}_r \\ \dot{x}_{R2} = \frac{1}{2} r \dot{\varphi}_l \end{cases} \tag{3.8}$$

In a differential drive mechanism, Equation (3.8) is added together to get the contribution of $\dot{x_R}$ component. The contribution $\dot{x_R}$ component is;

$$\dot{x}_R = \frac{1}{2} r \left(\dot{\varphi}_r + \dot{\varphi}_l \right) \tag{3.9}$$

The contribution of $\dot{y_R}$ is zero because there is no sideways movement in the robot reference frame. Therefore, the contribution of $\dot{y_R}$ component is:

$$\dot{y}_R = 0 \tag{3.10}$$

Finally, the contribution of a rotational component to the robot reference frame is computed independently and added together. Let us consider the right wheel, and a forward movement of this wheel results in an anti-clockwise rotation at point *P*. Suppose the right wheel rotates alone, and the robot pivots along with the left wheel and the robot chassis. The angular velocity, $\dot{\theta}_r$ at point *P* due to wheel movement along the arc of a circle of radius *2L* is given as:

$$\dot{\theta}_r = \frac{r}{2L} \dot{\varphi}_r \tag{3.11}$$

The same procedure is applied to the left wheel, except the forward spin of the left wheel results in a clockwise rotation at point P:

$$\dot{\theta}_l = -\frac{r}{2L} \dot{\varphi}_l \tag{3.12}$$

Moreover, by adding Equations (3.11) and (3.12) the rotational velocity component is given as:

$$\dot{\theta}_R = \frac{r}{2L} (\dot{\varphi}_r - \dot{\varphi}_l) \tag{3.13}$$

Therefore, by combining Equations (3.9), (3.10) and (3.13) the translational speed and the rotational velocity in the robot reference frame as follows:

$$\dot{q}_I = R^{-1}(\theta)\begin{bmatrix} \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l) \\ 0 \\ \frac{r}{2L}(\dot{\varphi}_r - \dot{\varphi}_l) \end{bmatrix} \tag{3.14}$$

Moreover, the inverse orthogonal transformational matrix $R^{-1}(\theta)$ is:

$$R^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

By substituting Equation (3.15) into Equation (3.14) a DWMR velocity in an inertial frame became:

$$\dot{q}_I = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l) \\ 0 \\ \frac{r}{2L}(\dot{\varphi}_r - \dot{\varphi}_l) \end{bmatrix} = \frac{r}{2}\begin{bmatrix} (\dot{\varphi}_r + \dot{\varphi}_l)\cos(\theta) \\ (\dot{\varphi}_r + \dot{\varphi}_l)\sin(\theta) \\ \frac{1}{L}(\dot{\varphi}_r - \dot{\varphi}_l) \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{r}{2}\begin{bmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ \frac{1}{L} & -\frac{1}{L} \end{bmatrix}\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} \tag{3.17}$$

The kinematic equation obtained above shows the DWMR velocity in an inertial frame.

3.5.2.2. Nonholonomic Kinematic Constraints

A mobile robot's holonomy can be defined in two ways: holonomic and nonholonomic. Nonholonomic systems have mathematically nonintegrable constraints, whereas holonomic systems have integrable constraints. On the other hand, a holonomic system can move in any direction in the configuration space; i.e., robot movement is not restricted by direction. In contrast, nonholonomic systems are those in which the velocities and other derivatives of the position are constrained, i.e., a robot's motion is limited. Nonholonomic robots are common because of their basic design and ease of operation. It also has fewer degrees of freedom than holonomic mobile robots by nature.

Several assumptions on wheel motion will cause the robot kinematic constraints to simplify the effort needed: movement on a horizontal axis; point contact between the wheels and ground; pure rolling, no slipping, skidding, or sliding; and no friction for rotation around contact points; steering axes orthogonal to the surface; and wheels connected by a rigid frame.

In Figure 3.3, a wheel presents three kinematic constraints on DWMR. The first constraint is that DMWR cannot slide sideways, i.e., no non-slipping constraint exists. That only moves in the direction of the symmetry axis of the actuated wheels. Algebraically, this constraint is written as (Mohareri et al., 2012):

- ✓ No lateral slip constraints:

$$\dot{y}\cos(\theta) - \dot{x}\sin(\theta) - d\,\dot{\theta} = 0 \tag{3.18}$$

where $\dot{x}$ and $\dot{y}$ the robot velocity components in the inertial frames. The other constraints are related to the motion of the wheels, which means the actuated wheels cannot rotate in the wrong direction. i.e., there are only pure rolling constraints.

- ✓ Pure rolling constraints:

$$\begin{cases} \dot{x}\cos(\theta) + \dot{y}\sin(\theta) + L\,\dot{\theta} - r\,\dot{\varphi}_r = 0 \\ \dot{x}\cos(\theta) + \dot{y}\sin(\theta) - L\,\dot{\theta} - r\,\dot{\varphi}_l = 0 \end{cases} \tag{3.19}$$

By rewriting Equation (3.19), we obtain:

$$\begin{cases} v + L\omega = r\,\dot{\varphi}_r \\ v - L\omega = r\,\dot{\varphi}_l \end{cases} \tag{3.20}$$

Since,

$$v = \dot{x}\cos(\theta) + \dot{y}\sin(\theta) \tag{3.21}$$

$$\omega = \dot{\theta} \tag{3.22}$$

Equation (3.20) are used to relate the velocities of DWMR, and the angular velocities of wheels within the center of mass (CoM) are written as:

$$\begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \phi \begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & L \\ 1 & -L \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.23}$$

where ϕ is the velocity conversion matrix. Similarly, a robot velocity vector is expressed as:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \phi^{-1} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} \tag{3.24}$$

The three-constraint in Equations (3.18) and (3.19) can be written as a generalized coordinate vector relation. The relationship is represented in a matrix form like as (Gao et al., 2021):

$$A(q)\,\dot{q} = 0 \tag{3.25}$$

where q is the coordinate configuration space written as a function of the *x*-axis, *y*-axis, heading angle theta, right, and left wheel displacements as follows:

$$q = [x \quad y \quad \theta \quad \varphi_r \quad \varphi_l]^T \tag{3.26}$$

Furthermore, its time derivatives are given by:

$$\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\theta} \quad \dot{\varphi}_r \quad \dot{\varphi}_l]^T \tag{3.27}$$

where φ_r and φ_l are right and left angular displacement, and its time derivatives are equivalent to the angular speed of wheels $\dot{\varphi}_r$ and $\dot{\varphi}_l$, respectively. By rewriting Equation (3.25) as:

$$A(q)\,\dot{q} = \begin{bmatrix} -\sin(\theta) & \cos(\theta) & -d & 0 & 0 \\ \cos(\theta) & \sin(\theta) & L & -r & 0 \\ \cos(\theta) & \sin(\theta) & -L & 0 & -r \end{bmatrix} \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} & \dot{\varphi}_r & \dot{\varphi}_l \end{bmatrix}^T \tag{3.28}$$

The obtained equation is functional when driving dynamic modeling and considering a constraint. If the inertia and mass of the wheels are neglected, DWMR satisfies the pure rolling and non-slipping constraint conditions (Fierro & Lewis, 1997). As a result, the nonholonomic matrix *A(q)* is reduced to the matrix shown below:

$$A(q) = [-\sin(\theta) \quad \cos(\theta) \quad -d] \tag{3.29}$$

The reduced constraints show that displacements occur only in the direction of the symmetry axis of the actuated wheels. Then the generalized coordinate vector is modified as:

$$q = [x \quad y \quad \theta]^T \tag{3.30}$$

It is important to emphasize that the system has a configuration space *n*, the generalized coordinate vector q, and the number of constraints given by *p* so that the velocity vector is of dimension $m = n - p$; in this case, $m = 2$ (corresponds to the degrees of freedom). The Jacobian matrix *S(q)* is formed by a set of linearly independent and smooth vector fields distributed in the null space of *A(q)* are used to transform velocity term in an inertial frame (Rabbani & Memon, 2021). i.e.,

$$A(q)S(q) = 0 \tag{3.31}$$

The Equation (3.31) is used to find auxiliary velocity as a function of time $w(t) \in R^{p \times 1}$ for all *t*. The relationship between coordinate *q* and velocity vector *w(t)* can be written as:

$$\dot{q} = S(q)w(t) \tag{3.32}$$

where $w(t)$ is the velocity vector given as $w(t) = [v \quad \omega]^T$ and *S(q)* is given by;

$$S(q) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \tag{3.33}$$

The Jacobian matrix of Equation (3.33) has utilized the DWMR velocity terms of $[\dot{x} \quad \dot{y} \quad \dot{\theta}]^T$, i.e.,

$$\dot{q} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.34}$$

The kinematic model equation obtained in Equation (3.34) is used throughout this thesis for system control and simulation. In the next topic, a dynamic model of a DWMR is derived.

3.5.3. Dynamic Modeling of TWMR Using Lagrangian Approach

A mobile robot system having an n-dimensional configuration space *R*, with generalized coordinates $q = [q_1, q_2, \ldots, q_n]$ and subject to m input constraints, can be described as (Mohareri et al., 2012):

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = E(q)\tau - A^T(q)\rho \tag{3.35}$$

where *H(q)* is an inertial matrix represented as $n \times n$ positive definite matrix, $C(q,\dot{q})$ is Coriolis and Centripetal torques matrix represented as $n \times n$, $F(\dot{q})$ is surface friction force matrix, $G(q)$ is a gravitational vector, ρ is a vector associated with Lagrange multiplier, *E(q)* is the control input of $n \times m$ transformation matrix, $A^T(q)$ is matrix associated with kinematic constraints, τ is input torque vector and, τ_d an unknown disturbance. Dynamic modeling of the mobile robot is determined in two ways:

- *Lagrangian approach:* which is an energy-based approach,
- *Newton-Euler approach:* is based on vector mechanics.

Lagrange's equations are differential equations in which the energies of the system and the work done instantaneously in time are considered. The derivation of the Lagrange equation for holonomic systems requires that the generalized coordinates be independent. The Lagrangian approach is used to determine a dynamic model because it is simpler to formulate and implement than the Newton-Euler approach. Precisely knowing all the forces applied to the system is challenging in the Newton-Euler approach.

The general form of the Lagrange method for the case of DWMRs with fixed standard wheels is given by (Ahmad Abu Hatab, 2013):

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = E(q)\tau - A^T(q)\rho \tag{3.36}$$

where q and $\dot{q}$ are given as in Equation (3.26) and (3.27). The total energy of a Lagrangian system is given as $L = T - W$, where T is kinetic energy, and W is potential energy. A DWMR has zero potential energy because movement is always along a horizontal plane, with no vertical direction movement. In the simple formula, the total potential energy is given as:

$$W = 0 \tag{3.37}$$

And total kinetic energy as:

$$T = \frac{1}{2}\dot{q}^T H(q)\dot{q} \tag{3.38}$$

Equation (3.36) rearranged in standard form as:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} = E(q)\tau - A^T(q)\rho \tag{3.39}$$

A Coriolis and Centripetal torques matrix are dependent on the posture and velocity with the $n \times n$ dimension given as follows:

$$C(q,\dot{q})\dot{q} = \frac{d}{dt}\left[H(q)\right]\dot{q} - \frac{\partial T}{\partial q} = \frac{d}{dt}\left[H(q)\right]\dot{q} - \frac{1}{2}\frac{\partial}{\partial q}\left[\dot{q}^T H(q)\dot{q}\right] \tag{3.40}$$

However, the Equation (3.39) is a disturbance-free system dynamic equation that is not physically applicable because of uncertainty and disturbance due to mechanical design and robot body interaction, like friction between components, wheels, and other body contacts. Considering a dynamic system in Equation (3.39) with uncertainties and disturbances, we get in the form of:

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + \tau_d = E(q)\tau - A^T(q)\rho \tag{3.41}$$

where $\tau_d - n \times 1$ vector denotes uncertainties and disturbances, $A(q) - p \times n$ matrix associated with nonholonomic constraints, $\rho - p \times 1$ vector of a constraint force (Lagrangian multiplier), $\tau - p \times 1$ vector control or input torques.

The dynamics system obtained in Equation (3.41) has the following properties to determine the stability analysis of the control system (Tzafestas, 2014):

- *Property 3.1: Limitation* - The matrix *H(q),* the norm of a matrix $C(q,\dot{q})$, and the vector τ_d are bounded,
- *Property 3.2: Skew-symmetry* - The matrix $\dot{H}(q) - 2C(q,\dot{q})$ is skew-symmetric.

3.5.3.1. Dynamic Model Using Wheel Constraint

A DWMR has two standard fixed wheels. These wheels are powered separately by a motor to achieve a differential drive. From the Equation (3.38), we have kinetic energy by considering the dynamics of the wheels. The following assumptions are used to derive the DWMR kinetic energy, and individual kinetic energy is obtained as follows (Benchouche et al., 2021):

- ✓ Kinetic energy (T_1) of WMR without the wheels,
- ✓ Kinetic energy (T_2) of actuated wheels in the plane only,
- ✓ Kinetic energy (T_3) of all wheels considering the orthogonal plane.

Total kinetic energy *T* is given as;

$$T = T_1 + T_2 + T_3 \tag{3.42}$$

A. Kinetic energy (T_1) of WMR without the wheels:

This kinetic energy is obtained by considering the horizontal v_x and vertical velocity v_y. Figure 3.5 shows the DWMR velocities representation of the mobile robot's free-body diagram given in Figure 3.4.

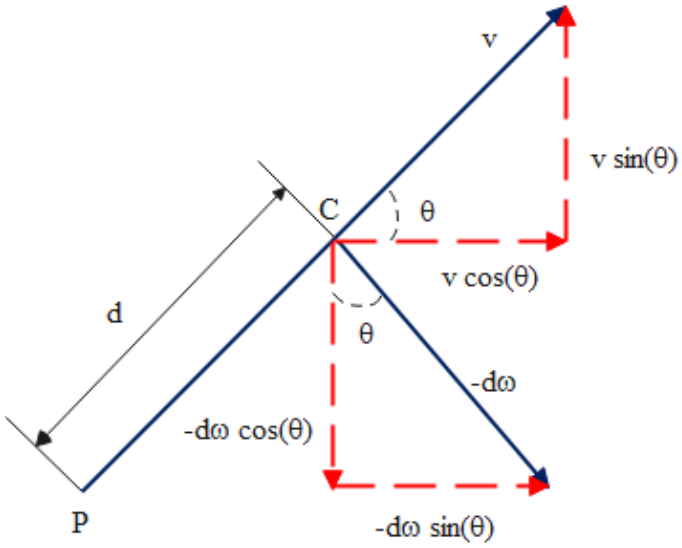

Figure 3.5: Velocity representation of DWMR

From the velocity representation of differential drive WMR in Figure 3.5 and Equation (3.34) , we will get v_x and v_y as shown below:

$$\begin{cases} v_x = \dot{x} + d\,\dot{\theta}\sin(\theta), \\ v_y = \dot{y} - d\,\dot{\theta}\cos(\theta), \end{cases} \text{ and } \omega = \dot{\theta} \tag{3.43}$$

We get the kinetic energy of DWMR without a wheel as:

$$T_1 = \frac{1}{2}\left(m_c \left(v_x^2 + v_x^2 \right) + I_c\,\dot{\theta}^2 \right) \tag{3.44}$$

By substituting Equation (3.43) into (3.44), we get T_1 in the following form:

$$\begin{aligned} T_1 &= \frac{1}{2}\left(m_c \left(\left(\dot{x} + d\,\dot{\theta}\sin(\theta) \right)^2 + \left(\dot{y} - d\,\dot{\theta}\cos(\theta) \right)^2 \right) + I_c\,\dot{\theta}^2 \right) \\ &= \frac{1}{2} m_c \left(\dot{x}^2 + \dot{y}^2 - 2\,\dot{y}\,d\,\dot{\theta}\cos(\theta) + 2\,\dot{x}\,d\,\dot{\theta}\sin(\theta) + d^2\,\dot{\theta}^2 \right) + \frac{1}{2} I_c\,\dot{\theta}^2 \end{aligned} \tag{3.45}$$

B. Kinetic energy (T_2) of actuated wheels in the plane only:

This kinetic energy is based on the actuated wheels in a wheel's plane. It is calculated based on wheel velocity, i.e., right and left wheels.

$$\begin{bmatrix} \dot{x}_r \\ \dot{y}_r \end{bmatrix} = \begin{bmatrix} v_x + L\dot{\theta}\cos(\theta) \\ v_y + L\dot{\theta}\sin(\theta) \end{bmatrix}, \text{ and } \begin{bmatrix} \dot{x}_l \\ \dot{y}_l \end{bmatrix} = \begin{bmatrix} v_x - L\dot{\theta}\cos(\theta) \\ v_y - L\dot{\theta}\sin(\theta) \end{bmatrix} \tag{3.46}$$

Then the kinetic energy T_2 is expressed as follows:

$$T_2 = \frac{1}{2} m_w \left(\dot{x}_r^2 + \dot{y}_r^2 + \dot{x}_l^2 + \dot{y}_l^2 \right) + \frac{1}{2} I_{mr} \dot{\theta}^2 + \frac{1}{2} I_{ml} \dot{\theta}^2 \tag{3.47}$$

After substituting a velocity Equation (3.46) into (3.47) and rearranging it, we get:

$$T_2 = m_w \left(\dot{x}^2 + \dot{y}^2 + 2\dot{x}\, d\, \dot{\theta}\sin(\theta) - 2\dot{y}\, d\, \dot{\theta}\cos(\theta) \right) + \dot{\theta}^2 \left(m_w \left(d^2 + L^2 \right) + I_m \right) \tag{3.48}$$

where $I_m = I_{mR} = I_{mL}$

C. Kinetic energy (T_3) of all wheels considering the orthogonal plane:

This kinetic energy of a DWMR is obtained by considering the moment of inertia of wheels around the axis that passes through them. The kinetic energy T_3 is given as:

$$T_3 = \frac{1}{2} I_w \dot{\varphi}_r^2 + \frac{1}{2} I_w \dot{\varphi}_l^2 = \frac{1}{2} I_w \left(\dot{\varphi}_r^2 + \dot{\varphi}_l^2 \right) \tag{3.49}$$

Finally, the total kinetic energy in Equation (3.42) became:

$$T = \frac{1}{2} m_t \left(\dot{x}^2 + \dot{y}^2 \right) + m_t \left(\dot{x}\, d\, \dot{\theta}\sin(\theta) - \dot{y}\, d\, \dot{\theta}\cos(\theta) \right) + \frac{1}{2} I \dot{\theta}^2 + \frac{1}{2} I_w \left(\dot{\varphi}_r^2 + \dot{\varphi}_l^2 \right) \tag{3.50}$$

where $m_t = m_c + 2m_w$, $I = m_c d^2 + I_c + 2m_w \left(d^2 + L^2 \right) + 2I_m$ and $\dot{\theta} = \omega$ as in Equation (3.22).

3.5.3.2. Dynamic Model Configuration and Simplification

The total Lagrangian energies of the above kinetic and potential energies were given as:

$$L = T - W, \text{ where, } W = 0 \tag{3.51}$$

Therefore, total Lagrangian energy is equivalent to kinetic energy since there is no vertical movement, as shown below:

$$L = \frac{1}{2}\left(m_t \dot{x}^2 + m_t \dot{y}^2 + I \dot{\theta}^2 + I_w \dot{\varphi}_r^2 + I_w \dot{\varphi}_l^2 \right) + m_t \dot{x} d \dot{\theta} \sin(\theta) - m_t \dot{y} d \dot{\theta} \cos(\theta) \qquad (3.52)$$

By substituting Equation (3.52) in Equation (3.36), we get the following results. From a generalized coordinate vector, we had q as in Equation (3.26) and $\dot{q}$ as in Equation (3.27). So, the partial differential equations of Lagrangian energy with a specified coordinates are given as follows;

$$\frac{\partial L}{\partial \dot{x}} = m_t \dot{x} + m_t d \dot{\theta} \sin(\theta) \qquad (3.53)$$

$$\frac{\partial L}{\partial \dot{y}} = m_t \dot{y} - m_t d \dot{\theta} \cos(\theta) \qquad (3.54)$$

$$\frac{\partial L}{\partial \dot{\theta}} = I \dot{\theta} + m_t \dot{x} d \sin(\theta) - m_t \dot{y} d \cos(\theta) \qquad (3.55)$$

$$\frac{\partial L}{\partial \dot{\varphi}_r} = I_w \dot{\varphi}_r, \quad \text{and} \quad \frac{\partial L}{\partial \dot{\varphi}_l} = I_w \dot{\varphi}_l \qquad (3.56)$$

The time derivative of the Equation (3.53) through Equation (3.56) is as follows:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{x}} \right) = m_t \ddot{x} + m_t d \ddot{\theta} \sin(\theta) + m_t d \dot{\theta}^2 \cos(\theta) \qquad (3.57)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{y}} \right) = m_t \ddot{y} - m_t d \ddot{\theta} \cos(\theta) + m_t d \dot{\theta}^2 \sin(\theta) \qquad (3.58)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}} \right) = I \ddot{\theta} + m_t d \left(\ddot{x} \sin(\theta) + \dot{x} \dot{\theta} \cos(\theta) - \ddot{y} \cos(\theta) + \dot{y} \dot{\theta} \sin(\theta) \right) \qquad (3.59)$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}_r} \right) = I_w \ddot{\varphi}_r; \quad \text{and} \quad \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\varphi}_l} \right) = I_w \ddot{\varphi}_l \qquad (3.60)$$

The other derivatives are obtained as follows:

$$\frac{\partial L}{\partial x} = 0, \quad \frac{\partial L}{\partial y} = 0, \quad \frac{\partial L}{\partial \theta} = m_t \dot{x} \dot{\theta} d \cos(\theta) + m_t \dot{y} \dot{\theta} d \sin(\theta), \quad \frac{\partial L}{\partial \varphi_r} = 0, \quad \frac{\partial L}{\partial \varphi_l} = 0 \qquad (3.61)$$

By substituting Equations (3.57) to (3.60) into Equation (3.36); the left side matrices in Equation (3.41) are given as follows:

$$H(q)=\begin{bmatrix} m_t & 0 & m_t d\sin(\theta) & 0 & 0 \\ 0 & m_t & -m_t d\cos(\theta) & 0 & 0 \\ m_t d\sin(\theta) & -m_t d\cos(\theta) & I & 0 & 0 \\ 0 & 0 & 0 & I_w & 0 \\ 0 & 0 & 0 & 0 & I_w \end{bmatrix} \tag{3.62}$$

$$C(q,\dot{q})=\begin{bmatrix} 0 & 0 & m_t d\,\dot{\theta}\cos(\theta) & 0 & 0 \\ 0 & 0 & m_t d\,\dot{\theta}\sin(\theta) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.63}$$

From Equation (3.29), we have a reduced nonholonomic constraints matrix by neglecting wheel inertia. Therefore, the displacements occur only in the direction of the symmetry axis. So, the matrix in Equation (3.62) and (3.63) can be reduced to a 3x3 matrix, and Equation (3.39) is rewritten as:

$$\begin{bmatrix} m_t & 0 & m_t d\sin(\theta) \\ 0 & m_t & -m_t d\cos(\theta) \\ m_t d\sin(\theta) & -m_t d\cos(\theta) & I \end{bmatrix}\ddot{q}+\begin{bmatrix} 0 & 0 & m_t d\,\dot{\theta}\cos(\theta) \\ 0 & 0 & m_t d\,\dot{\theta}\sin(\theta) \\ 0 & 0 & 0 \end{bmatrix}\dot{q}=\begin{bmatrix} F_x \\ F_y \\ \tau \end{bmatrix}+\begin{bmatrix} D_x \\ D_y \\ D_\theta \end{bmatrix} \tag{3.64}$$

Let us say the force produced by the right wheel is F_r, by the left wheel F_l and the corresponding force along the x-axis is F_x and y-axis is F_y. The input transformation matrix E(q) depends on the relationship between force and torque and the robot's position (Benchouche et al., 2021).

$$F_r=\frac{\tau_r}{r},\ \text{and}\ F_l=\frac{\tau_l}{r} \tag{3.65}$$

However, force produced by both wheels along the *x*-axis and *y*-axis with torque is given as:

$$\begin{bmatrix} F_x \\ F_y \\ \tau \end{bmatrix}=\begin{bmatrix} (F_r+F_l)\cos(\theta) \\ (F_r+F_l)\sin(\theta) \\ (F_r-F_l)L \end{bmatrix}=\frac{1}{r}\begin{bmatrix} (\tau_r+\tau_l)\cos(\theta) \\ (\tau_r+\tau_l)\sin(\theta) \\ (\tau_r-\tau_l)L \end{bmatrix}=\frac{1}{r}\begin{bmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ L & -L \end{bmatrix}\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{3.66}$$

Thus, the input transformation matrix *E(q)* is obtained as follows;

$$E(q) = \frac{1}{r}\begin{bmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ L & -L \end{bmatrix} \tag{3.67}$$

According to the constraints matrix as in Equation (3.28), force constraint is expressed as:

$$\begin{bmatrix} D_x \\ D_y \\ D_\theta \end{bmatrix} = \begin{bmatrix} m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))\dot{\theta}\cos(\theta) \\ -m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))\dot{\theta}\sin(\theta) \\ m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))d\dot{\theta} \end{bmatrix} \tag{3.68}$$

The matrix representation of the above constraint forces is as follows:

$$\begin{bmatrix} m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))\dot{\theta}\cos(\theta) \\ -m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))\dot{\theta}\sin(\theta) \\ m_t(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))d\dot{\theta} \end{bmatrix} = A^T(q)\rho \tag{3.69}$$

However, from Equation (3.29), we had *A(q)* and its transpose $A^T(q)$ and vector of a constraint force (Lagrangian multipliers) ρ in Equation (3.69), is become:

$$A^T(q) = \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ -d \end{bmatrix}, \quad \rho = -m_t\dot{\theta}\left(\dot{x}\cos(\theta) + \dot{y}\sin(\theta)\right) \tag{3.70}$$

Thus, the dynamic modeling equation in Equation (3.64) is obtained as:

$$\begin{aligned} &\begin{bmatrix} m_t & 0 & m_t d\sin(\theta) \\ 0 & m_t & -m_t d\cos(\theta) \\ m_t d\sin(\theta) & -m_t d\cos(\theta) & I \end{bmatrix}\ddot{q} + \begin{bmatrix} 0 & 0 & m_t d\dot{\theta}\cos(\theta) \\ 0 & 0 & m_t d\dot{\theta}\sin(\theta) \\ 0 & 0 & 0 \end{bmatrix}\dot{q} \\ &= \frac{1}{r}\begin{bmatrix} \cos(\theta) & \cos(\theta) \\ \sin(\theta) & \sin(\theta) \\ L & -L \end{bmatrix}\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} + \begin{bmatrix} -\sin(\theta) \\ \cos(\theta) \\ -d \end{bmatrix}\left(-m_t\dot{\theta}(\dot{x}\cos(\theta) + \dot{y}\sin(\theta))\right) \end{aligned} \tag{3.71}$$

The above dynamic equation can be transformed into a proper system representation for control and simulation purposes. The transformation can eliminate a constraint term from

the dynamic equation. Let us define two matrices as having a shift as follows (Benchouche et al., 2021; Mohareri et al., 2012):

$$w(t) = \begin{bmatrix} v & \omega \end{bmatrix}^T \tag{3.72}$$

$$S(q) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \tag{3.73}$$

So, we can rewrite Equation (3.32) as:

$$\dot{q} = S(q)w(t) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.74}$$

By differentiating the Equation (3.74) and using w instead $w(t)$, we have:

$$\ddot{q} = \dot{S}(q)w + S(q)\dot{w} \tag{3.75}$$

Now by substituting Equation (3.75) in (3.35), the following representation will be obtained:

$$H(q)[\dot{S}(q)w + S(q)\dot{w}] + C(q,\dot{q})\left[S(q)w\right] + F(\dot{q}) + G(q) + \tau_d = E(q)\tau - A^T(q)\rho \tag{3.76}$$

By rearranging the above equation, we get:

$$H(q)\dot{S}(q)w + H(q)S(q)\dot{w} + C(q,\dot{q})S(q)w + F(\dot{q}) + G(q) + \tau_d = E(q)\tau - A^T(q)\rho \tag{3.77}$$

To eliminate the constraint matrix $A^T(q)\rho$ multiply Equation (3.77) by $S^T(q)$ as follows:

$$S^T(q)\left[\begin{array}{c} H(q)\dot{S}(q)w + H(q)S(q)\dot{w} + C(q,\dot{q})S(q)w + F(\dot{q}) + G(q) + \tau_d \\ = E(q)\tau - A^T(q)\rho \end{array}\right] \tag{3.78}$$

From Equation (3.31) we have $S^T(q)A^T(q) = 0$. Constraints free dynamic equation is given as:

$$\begin{aligned} &\left[S^T(q)H(q)S(q)\right]\dot{w} + \left[S^T(q)C(q,\dot{q})S(q) + S^T(q)H(q)\dot{S}(q)\right]w \\ &+S^T(q)F(\dot{q}) + S^T(q)G(q) + S^T(q)\tau_d = S^T(q)E(q)\tau \end{aligned} \tag{3.79}$$

To rewrite the Equation (3.79), let us define dynamic equations as follows;

$$\overline{H}(q)\dot{w}(t) + \overline{C}(q,\dot{q})w(t) + \overline{F}(\dot{q}) + \overline{G}(q) + \overline{\tau}_d = \overline{E}(q)\tau \tag{3.80}$$

where

$$\overline{H}(q) = S^T(q)H(q)S(q) \tag{3.81}$$

$$\overline{C}(q,\dot{q}) = S^T(q)C(q,\dot{q})S(q) + S^T(q)H(q)\dot{S}(q) \tag{3.82}$$

$$\overline{F}(\dot{q}) = S^T(q)F(\dot{q}) = 0 \tag{3.83}$$

$$\overline{G}(q) = S^T(q)G(q) = 0 \tag{3.84}$$

$$\overline{\tau}_d = S^T(q)\tau_d \tag{3.85}$$

$$\overline{E}(q) = S^T(q)E(q) \tag{3.86}$$

$$S^T(q)A^T(q) = 0 \tag{3.87}$$

By ignoring the unknown disturbance vector, the simplified dynamic model is given as:

$$\begin{bmatrix} m_t & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} \dot{v}(t) \\ \dot{\omega}(t) \end{bmatrix} + \begin{bmatrix} 0 & -m_t d\,\dot{\theta} \\ m_t d\,\dot{\theta} & 0 \end{bmatrix}\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \frac{1}{r}\begin{bmatrix} 1 & 1 \\ L & -L \end{bmatrix}\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} \tag{3.88}$$

The dynamic model in Equation (3.88) represents a nonholonomic system's behavior in a new robot or local coordinates, i.e., the Jacobian matrix *S(q)* transforms the velocity w in coordinates of the model of the mobile base to the velocities $\dot{q}$ in cartesian coordinates. As a result, the properties of the original dynamics are preserved for the new set of coordinates.

The dynamics system obtained in Equation (3.80) has the following properties to determine the stability analysis of the control system (Martins & Bertol, 2022):

- *Property 3.3: Limitation* - The matrix $\overline{H}(q)$, the norm of a matrix $\overline{C}(q,\dot{q})$, and the vector $\overline{\tau}_d$ is all bounded;
- *Property 3.4: Skew-symmetry* - The matrix $\dot{\overline{H}}(q) - 2\overline{C}(q,\dot{q})$ is skew-symmetric.

Proof: By substituting the time derivative of the $\overline{H}(q)$ and $\overline{C}(q,\dot{q})$ in $\dot{\overline{H}}(q) - 2\overline{C}(q,\dot{q})$, we get:

$$\dot{\overline{H}}(q) = \dot{S}^T(q)H(q)S(q) + S^T(q)\dot{H}(q)S(q) + S^T(q)H(q)\dot{S}(q), \tag{3.89}$$

$$\begin{aligned}\dot{\overline{H}}(q) - 2\overline{C}(q,\dot{q}) = \dot{S}^T(q)H(q)S(q) + S^T(q)\dot{H}(q)S(q) + S^T(q)H(q)\dot{S}(q) \\ -2\left(S^T(q)C(q,\dot{q})S(q) + S^T(q)H(q)\dot{S}(q) \right),\end{aligned} \tag{3.90}$$

$$\begin{aligned}\dot{\overline{H}}(q) - 2\overline{C}(q,\dot{q}) = \dot{S}^T(q)H(q)S(q) - [\dot{S}^T(q)H(q)S(q)]^T \\ + S^T(q)[\dot{H}(q) - 2C(q,\dot{q})]S(q)\end{aligned} \tag{3.91}$$

Since $\dot{H}(q) - 2C(q,\dot{q})$ is skew-symmetry, Equation (3.91) is also skew-symmetry. As a result, the derived system dynamic model is asymptotically stable. The dynamic model in the Equation (3.88) is used in system control and simulation throughout this thesis work.

3.5.3.3 Actuator Dynamic Modelling

In the case of DWMRs, the wheels are driven by two similar DC motors that have both mechanical and electrical dynamics. The electrical part of the motor dynamic model equation can be obtained by using the Kirchhoff voltage law as follows (Jabeur, 2019):

$$R_a i_a + L_a \frac{di_a}{dt} + v_b = v_s \tag{3.92}$$

where R_a is the resistance of armature winding, L_a is leakage inductance in the armature winding, i_a is motor armature current, v_s and v_b are voltage source and back electromotive force voltage, respectively. The mechanical part of the motor dynamic model equation can be obtained by using Newton's second law as follows:

$$J_m \frac{d\omega_m}{dt} + B_m\omega_m - \tau_{lo} = \tau_m \tag{3.93}$$

where J_m is the motor's inertia, ω_m the angular speed of a motor, B_m equivalent damping constant, τ_{lo} and τ_m load and motor torque, respectively. Also, the torque of the motor with armature current and the speed of the motor with back emf are related as follows:

$$\tau_m = k_m i_a \tag{3.94}$$

$$v_b = k_b\omega_m \tag{3.95}$$

The parameters k_m and k_b are the torque constant and the back electromotive force voltage constant, respectively. By rearranging and taking Laplace transform of Equations (3.92) and (3.93), the dynamic model of the motor becomes:

$$\begin{cases} v_s - v_b = \left(sL_a + R_a\right) I_a \\ \tau_m - \tau_{lo} = \left(sJ_m + B_m\right) \omega_m \end{cases} \tag{3.96}$$

By substituting Equations (3.94) and (3.95) in the Equation (3.96) with neglecting load torque, we get the transfer function of a motor as follows:

$$\frac{\omega_m(s)}{v_s(s)} = \frac{k_m}{\left(sL_a + R_a\right)\left(sJ_m + B_m\right) + k_m k_b} \tag{3.97}$$

The dynamic equation of the motor from Equation (3.97) is added to the dynamic equation of the TWMR to study and analyze how the derived system can satisfy a differential drive method for WMR locomotion during open-loop system simulation.

3.5.4. Controllability Analysis of TWMR

The primary considerations that influence the selection of a robot in a robotics system are controllability and stability. For a robot with kinematic constraints, we need to analyze them and determine if they affect the system's controllability. Nonholonomic constraints limit the robot's capability to move in its primary motion directions (basic movements such as driving straight and spinning) but not in the constrained direction. However, by combining basic movements, such a robot may attain any desired configuration.

According to the Lie bracket, a system satisfies the rank condition of Lie algebra (e.g., controllability rank condition or rank accessibility condition). If the involutive distribution has a rank of n, the robot can reach any point q in the configuration space. Chow's theorem states that "the system is controllable if its involutive distribution involving Lie brackets has a rank equal to the configuration space dimension (n)." Some robots can be controlled if their constraints are nonholonomic and their involutive distribution has a rank of n (Klančar et al., 2017). For two vector fields g_1 and g_2 the Lie-bracket is a third vector field g_3 is defined as (Coelho & Nunes, 2003):

$$g_3 = \left[g_1, g_2\right](q) = \frac{\partial g_2}{\partial q} g_1(q) - \frac{\partial g_1}{\partial q} g_2(q) \tag{3.98}$$

where $[g_1, g_2] = -[g_2, g_1]$ and $[g_1, g_2] = 0$ for constant vector fields g_1 and g_2.

A nonholonomic constraint represented in Equation (3.25) and matrix A(q) is defined as Equation (3.29); therefore, the system has a three-dimensional configuration space ($n=3$)

and a constraint (*p=1*) with a velocity vector (*m = n–p = 2*). The independent field vectors g_1 and g_2 from Jacobian matrix S(q) as in Equation (3.73):

$$g_1 = [\cos(\theta) \quad \sin(\theta) \quad 0]^T \text{ , and } g_2 = [-d\sin(\theta) \quad d\cos(\theta) \quad 1]^T \tag{3.99}$$

By using the Lie-bracket provided in Equation (3.98), we will get the third field vector g_3;

$$g_3 = [g_1, g_2] = \frac{\partial g_2}{\partial q} g_1(q) - \frac{\partial g_1}{\partial q} g_2(q) = \begin{bmatrix} \sin(\theta) \\ -\cos(\theta) \\ 0 \end{bmatrix} \tag{3.100}$$

If the system in Equation (3.32) is completely integrable (holonomic), then it is involutive, i.e., $rank(g_1\ g_2) = rank(g_1\ \ g_2\ [g_1, g_2])$.

$$rank\left\{\begin{bmatrix} \cos(\theta) & -d\sin(\theta) & \sin(\theta) \\ \sin(\theta) & d\cos(\theta) & -\cos(\theta) \\ 0 & 1 & 0 \end{bmatrix}\right\} = 3 \tag{3.101}$$

However, the system is not completely integrable (holonomic) because the rank of the two vector fields, (*rank (g_1 g_2) = 2*), is not equal to the Lie brackets' ($rank(g_1\ g_2\ [g_1, g_2]) = 3$). To conclude, the dimension of the generalized configurable vector is equal to the rank of a Lie bracket. i.e., *n = 3*, which makes the system controllable and nonholonomic according to Chow's theorem. In DWMR, a system's stability is not challenging because the platform has two wheels and one free castor wheel that is used to keep the balance of a system.

3.6. Open-Loop System Modeling Simulation Analysis

The open-loop system simulation is used to check whether a system's derived mathematical model satisfies the mobile robot's differential drive mechanism. The proposed method is simulated using Matlab and Simulink (R2021a) toolbox software. Table 3.1 shows the TWMR and DC motor parameters specification. A kinematics and dynamics model of a DWMR in Equations (3.23), (3.34) and (3.88) with the motor dynamic model equation are simulated. The Matlab code, functions, and Simulink block diagram of an open-loop system model used for the following results are included in the Appendices.

If the two motors have equal speed in magnitude but different signs in a differential drive system, the robot rotates around an arc of the wheel. The robot's direction is forward if the two motors have similar speeds in magnitude with the same +ve sign (the +ve sign is for forward movement). The robot moved backward when a similar magnitude with the -ve

speed sign was applied to the motor. If both motor speeds are not equal in magnitude but have the same sign, the robot's movement is at a turning point or curvature.

Table 3.1: TMWR and motor parameters specifications (Martins & Bertol, 2022)

Dynamics and Kinematic models Parameters		Actuator Parameters	
Parameters	Values with unit	Parameters	Values with unit
m_t	120 kg	k_m	0.2247Nm/A
L	0.33 m	k_b	0.02 Vs/rad
d	0.10 m	La	0.01 H
r	0.135 m	Ra	6 Ω
I	15.0656 kg-m^2	$\tau_{m_{max}}$	20.45 Nm
Ic	11.4180 kg-m^2	$V_{S\ max}$	24 Vdc (maximum motor voltage)
Iw	0.0456 kg-m^2	$I_{a\ max}$	4.0 A (Maximum motor current)
Im	0.7293 kg-m^2		

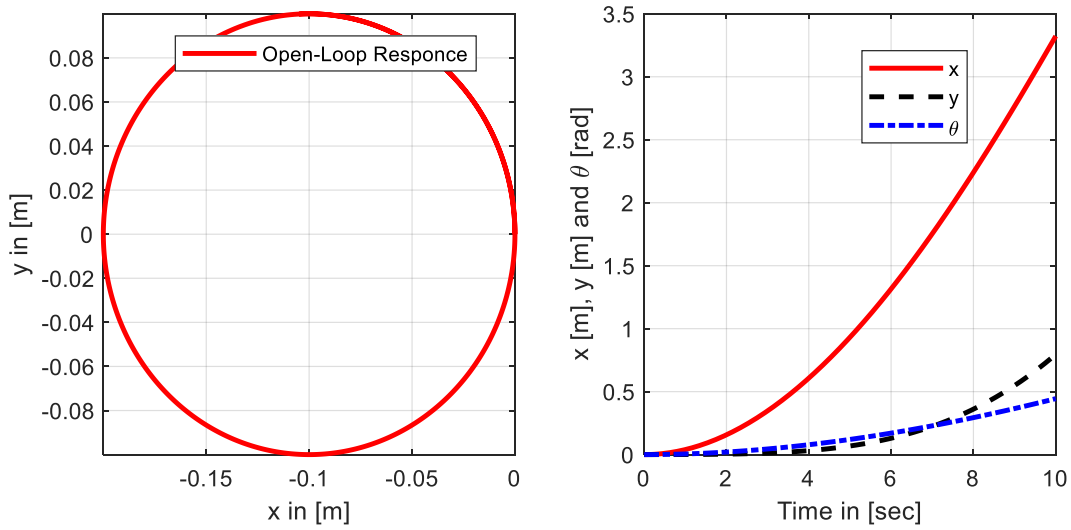

Figure 3.6: Open-loop trajectory tracking and robot pose for the same magnitude but a different input sign

As seen in Figure 3.6, the robot will rotate around a center point when we have equal opposite-sign motor speed. The robot is supposed to revolve around itself in response to such input. It makes a circle and rotates around it because of the misalignment between its center of mass (CoM) and center of rotation. The angles between the robot's reference frame and the local frame are increased, which makes the system unstable because the robot revolves around the midpoint of the body frame.

The other test that can be verified the model is by giving the same sign but not equal input to the robot. Figure 3.7 and Figure 3.8 also show an open-loop robot response with different magnitudes and the same sign of the input speed of the motor. As a result, the robot tries to verify the turning point condition during its movement. A robot's posture response is unbounded and unstable for a given input.

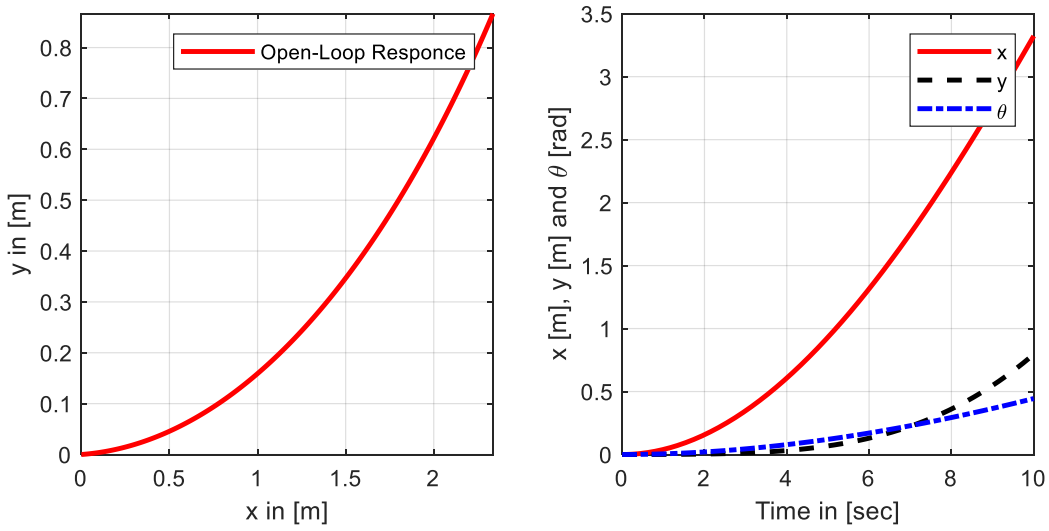

Figure 3.7: Open-loop trajectory tracking and robot pose for different magnitudes, but the same +ve sign input

The other test employed for the derived system modeling is that a robot moves forward and backward. As seen in Figure 3.9, the robot moves from its initial point to the left and right, respectively. The result shows that a robot's position tracking response is unbounded within a given input.

An open-loop simulation result shows that the derived TWMR system models (kinematic and dynamic) satisfy the differential drive method. An open-loop robot position does not satisfy the desired position tracking because the system response is not bounded or converged in all conditions, which makes it unstable and challenging to control in an interesting region.

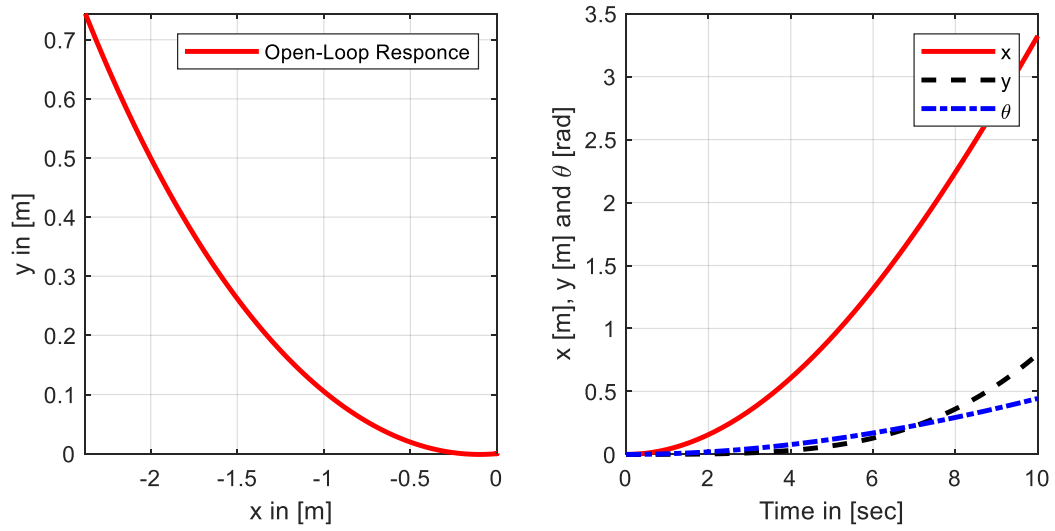

Figure 3.8: Open-loop trajectory tracking and robot pose for a different magnitude but the same -ve sign input

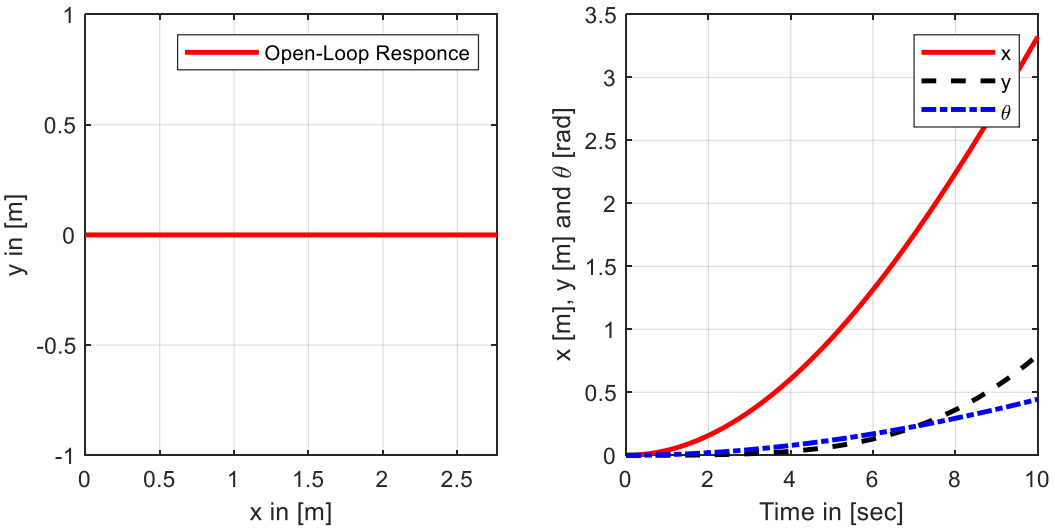

Figure 3.9: Open-loop trajectory tracking and robot pose for the same magnitude and the same +ve sign input

Generally, the system is highly nonlinear and affected by systems input changes, parameter uncertainty, and unknown disturbances. The control law that solves the aforementioned problem is designed and explained in the following chapter.

CHAPTER 4

4. CONTROLLER DESIGN

4.1. Reference Trajectory Generation

The main goal of trajectory tracking of nonholonomic TWMR is to eliminate the tracking error between the desired and actual trajectory. In a mobile robot, reference tracking problems are widely used and researched by numerous researchers because they use a control law to predict the future robot state. Hence, the reference trajectory generation of a TWMR is an essential task for the smooth movement of a robot. A nonholonomic mobile robot trajectory track can be described and formulated as follows (Hwang et al., 2013):

$$\begin{cases} \dot{x_r} = v_r \cos(\theta_r) \\ \dot{y_r} = v_r \sin(\theta_r) \\ \dot{\theta_r} = \omega_r \end{cases} \tag{4.1}$$

where x_r and y_r are reference robot positions in the x and y axes, θ_r is the reference angle between the local and the inertial reference frames, and v_r and ω_r are reference velocity inputs. From Equation (4.1) the velocity profile has two components: the x-axis and y-axis velocity components. The total tangential (linear) velocity of a robot is:

$$v_r = \left(v_x(t)^2 + v_y(t)^2\right)^{\frac{1}{2}} = \left(\dot{x}_r(t)^2 + \dot{y}_r(t)^2\right)^{\frac{1}{2}} \tag{4.2}$$

The robot orientation angle θ_r that satisfies a nonholonomic constraint is given as follows:

$$\theta_r = \tan^{-1}(\dot{y}_r(t) / \dot{x}_r(t)) \tag{4.3}$$

By differentiating the Equation (4.3) , the reference angular velocity ω_r of a robot is obtained as:

$$\omega_r = \frac{d\theta_r}{dt} = \frac{\ddot{y}_r(t)\,\dot{x}_r(t) - \ddot{x}_r(t)\,\dot{y}_r(t)}{\left(\dot{x}_r(t)\right)^2 + \left(\dot{y}_r(t)\right)^2} \tag{4.4}$$

Moreover, in a generalized vector, the reference trajectory and velocity are represented by:

$$q_r = [x_r \quad y_r \quad \theta_r]^T \tag{4.5}$$

$$w_r = [v_r \quad \omega_r]^T, w_r > 0, \forall t \tag{4.6}$$

For smooth and perfect velocity control, the tracking error between the reference and actual trajectory is near zero (ideally zero). Mathematically, $\lim_{t\to\infty}(q_r - q) = 0$. The following topic introduces a velocity control law as a function of position error, and reference velocity is proposed to reduce trajectory tracking errors.

4.2. Backstepping Controller (BSC) Design

In a control system, dealing with a nonlinear system is challenging because in a nonlinear system, the system's state does not change linearly. Many nonlinear controllers have been developed for nonlinear systems control. The backstepping (BSC) and sliding mode controller (SMC) are widely used nonlinear controllers. A backstepping controller is used in this work and is explained in the next topic.

4.2.1. Introduction to the Backstepping Approach

A backstepping control method is a nonlinear controller designed in a recursive way that combines the choice of a Lyapunov function with the design of a feedback controller. A Lyapunov stability analysis function guarantees the global asymptotic stability of the designed controller (Vaidyanathan, 2021).

A BSC is widely applicable in the area of nonlinear control fields. The backstepping method achieves the Lyapunov stabilization by collectively shifting all the eigenvalues in a favorable direction in the complex plane rather than assigning individual eigenvalues. Unlike optimal control, which is challenging to solve, the Riccati operator has a higher order of PDEs (Smith et al., 2016). These control methods are used only for strict feedback systems, i.e., the lower triangular form of a system. Let us consider the strict-feedback system shown in Figure 4.1.

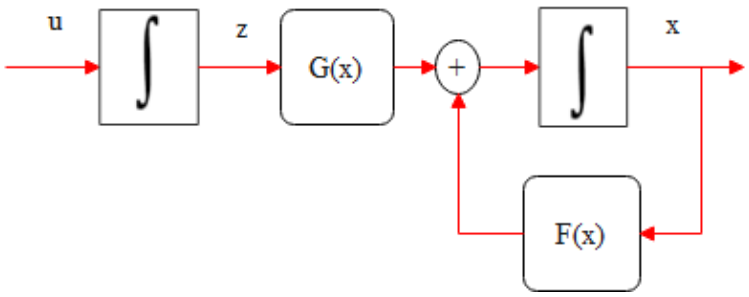

Figure 4.1: The Strict feedback system model

For a subsystem, we have the following forms (Raptis & Valavanis, 2011):

$$\dot{x} = f(x) + g(x)z \tag{4.7}$$

$$\dot{z} = u \tag{4.8}$$

where $(x, z) \in R^{n+1}$ is the state vector, u is scalar input to the system, $f(x)$ and $g(x)$ are known smooth functions in which $f_i(x)$ is vanishing at the origin. The aims of designing a state feedback control law are such that $(x, z) \to 0$ as $t \to \infty$ or to treat z as a virtual control input used to stabilize a subsystem x. According to Figure 4.1, a system in Equations (4.7) and (4.8) connected in cascade. Now let us assume a smooth feedback control law $z = \sigma(x)$ with $\sigma(0) = 0$; the system in Equation (4.7) has become:

$$\dot{x} = f(x) + g(x)\sigma(x) \tag{4.9}$$

For a Lyapunov function $V(x)$ for a selected control law by using lie derivative is given as:

$$\frac{\partial V}{\partial x}\left[f(x) + g(x)\sigma(x)\right] \leq - W(x) \tag{4.10}$$

where $W(x)$ is a positive definite matrix, by adding and subtracting $g(x)\sigma(x)$ from the right-hand side of Equation (4.7) , then it became:

$$\dot{x} = f(x) + g(x)\sigma(x) + g(x)\left[z - \sigma(x)\right] \tag{4.11}$$

$$\dot{z} = u \tag{4.12}$$

The error between pseudo control $\sigma(x)$ and z is:

$$e_z = z - \sigma(x) \tag{4.13}$$

Rewriting Equations (4.11) and (4.12) with an error in Equation (4.13) as:

$$\dot{x} = \left[f(x) + g(x)\sigma(x)\right] + g(x)e_z \tag{4.14}$$

$$\dot{e_z} = u - \dot{\sigma}(x) \tag{4.15}$$

The derivative of pseudo control input $\sigma(x)$ is computed as follows:

$$\dot{\sigma} = \frac{\partial \sigma}{\partial x}\left(f(x) + g(x)z\right) \tag{4.16}$$

Now setting a nominal control input $u = \upsilon + \dot{\sigma}$, where $\upsilon \in R$, the transformed system, has the following form:

$$\begin{cases} \dot{x} = \left[f(x) + g(x)\sigma(x) \right] + g(x)e_z \\ \dot{e_z} = \upsilon \end{cases} \tag{4.17}$$

A system in Equation (4.17) is similar to the system in Equations (4.7) and (4.8). However, the system in Equations (4.7) and (4.8) is stable at origin such that $(x, z) \to 0$ as $t \to \infty$, while in a transformed system using auxiliary control input $\sigma(x)$ has been backstepped through the integrator form $u = \upsilon + \sigma(x)$. For an initial and transformed system, the overall stabilization depends on Lyapunov function knowledge. By using a Lyapunov function candidate as:

$$V_n(x, z) = V(x) + \frac{1}{2}e_z^2 \tag{4.18}$$

One obtains $\dot{V_n}$ as:

$$\dot{V_n}(x, z) = \frac{\partial V}{\partial x}\left[f(x) + g(x)\sigma(x) \right] + \frac{\partial V}{\partial x} g(x)e_z + e_z\upsilon \le -W(x) + \frac{\partial V}{\partial x} g(x)e_z + e_z\upsilon \tag{4.19}$$

The control input υ is chosen as:

$$\upsilon = -\frac{\partial V}{\partial x} g(x) - ke_z, \quad \text{k} > 0 \tag{4.20}$$

By substituting a control input υ in Equation (4.19) we will obtain:

$$\dot{V_n} \le -W(x) - ke_z^2 \tag{4.21}$$

The system is uniformly asymptotically stable at the origin $(x = 0, z = 0)$ due to $\sigma(0) = 0$, and $\lim_{t \to \infty} e_z = 0$ then, the origin $(x = 0, z = 0)$ is asymptotically stable. The control law is:

$$u = \frac{\partial \sigma}{\partial x}\left[f(x) + g(x)z \right] - \frac{\partial V}{\partial x} g(x) - k\left[z - \sigma(x) \right] \tag{4.22}$$

The main advantage of the backstepping control method is that it allows one to design a control law that stabilizes the internal subsystem with an auxiliary velocity control input and allows the system to step back out to the outer system with a control law.

4.2.2. Kinematic-Based Backstepping Controller

The kinematic-based backstepping controller is a nonlinear controller whose control law is designed based on a nonlinear kinematic controller. A kinematic controller is a nonlinear controller developed based on the kinematics model. The first kinematic-based backstepping controller used to control a nonholonomic mobile robot system was proposed by (Kanayama et al., 1990). Numerous researchers widely used this work for WMR stability and trajectory tracking control.

The kinematic-based backstepping controller can be applied with PID, fuzzy logic controller, SMC, and NN controllers (Esmaeili et al., 2017; Hassan & Saleem, 2022; Mai et al., 2021; L. Xu et al., 2022; Yousuf et al., 2021). A kinematic-based backstepping controller with a system model is proposed, as shown in Figure 4.2.

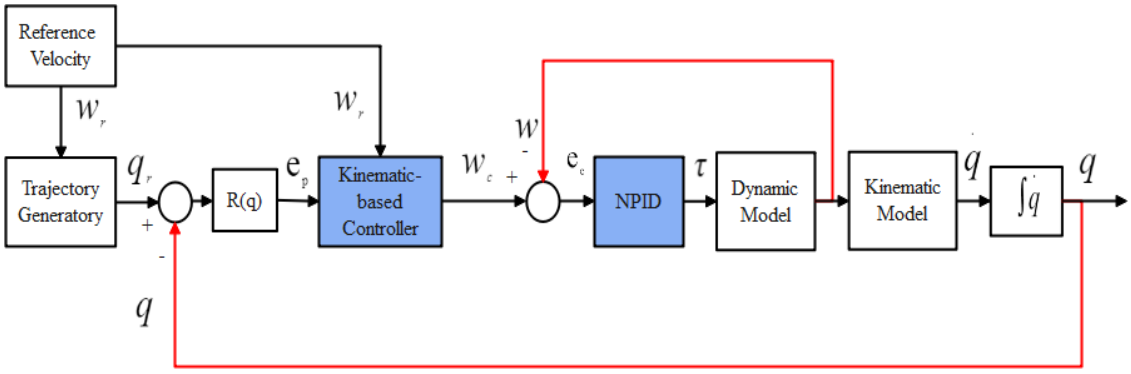

Figure 4.2: Kinematic-based controller and NPID controller with a proposed system

In trajectory tracking control, two postures of a mobile robot should be defined to control a robot navigation problem: a reference posture q_r and a current robot posture q, as shown in Figure 4.2. The reference posture represents the final goal of the system, and the current pose is the real-time pose at the moment. Let us consider the following references and the current position configuration for controller design:

$$q_r = [x_r \quad y_r \quad \theta_r]^T, \text{ and } q = [x \quad y \quad \theta]^T \tag{4.23}$$

The main objective of kinematic-based control is to determine the auxiliary velocity control of a system to track a given trajectory. It is determined by a reference velocity and a described posture configuration error. The pose error of a robot's local frame is defined as:

$$e_p = \begin{bmatrix} e_x \\ e_y \\ e_\theta \end{bmatrix} = R(\theta)[q_r - q] = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{4.24}$$

If $q_r = q$ the error posture configuration $e_p = 0$, and if $q_r > q$ the robot is behind the goal, then $e_p > 0$. Let us differentiate error posture configuration separately $\dot{e}_p = [\dot{e}_x \quad \dot{e}_y \quad \dot{e}_\theta]^T$, as given below, with perfect velocity control $[v \quad \omega]^T = [v_r \quad \omega_r]^T$ for simplification (Martins & Bertol, 2022).

- ✓ *The derivative of e_x is provided as:*

$$\begin{aligned} \dot{e_x} &= (\dot{x_r} - \dot{x})\cos(\theta) + (\dot{y_r} - \dot{y})\sin(\theta) - (x_r - x)\dot{\theta}\sin(\theta) + (y_r - y)\dot{\theta}\cos(\theta) \\ &= e_y\omega - v + \dot{x_r}\cos(\theta) + \dot{y_r}\sin(\theta) \end{aligned} \tag{4.25}$$

However, from the Equation (4.24) we had $e_\theta = \theta_r - \theta$ and rearranged in terms of θ as $\theta = \theta_r - e_\theta$. Then Equation (4.25) re-written and simplified as:

$$\begin{aligned} \dot{e_x} &= e_y\omega - v + \dot{x_r}\cos(\theta_r - e_\theta) + \dot{y_r}\sin(\theta_r - e_\theta) \\ &= \mathrm{e}_y\omega - v + \dot{x_r}\left(\cos(\theta_r)\cos(e_\theta) + \sin(\theta_r)\sin(e_\theta)\right) \\ &\quad + \dot{y_r}\left(\sin(\theta_r)\cos(e_\theta) - \cos(\theta_r)\sin(e_\theta)\right) \end{aligned} \tag{4.26}$$

Using the relation $\dot{x_r}\, sin(\theta_r) = \dot{y_r}\, cos(\theta_r)$ the final equation becomes:

$$\dot{e_x} = e_y\omega - v + v_r\cos(e_\theta) \tag{4.27}$$

- ✓ *The derivative of e_y is provided as:*

$$\dot{e_y} = -(\dot{x_r} - \dot{x})\sin(\theta) + (\dot{y_r} - \dot{y})\cos(\theta) - (x_r - x)\dot{\theta}\cos(\theta) + (y_r - y)\dot{\theta}\sin(\theta) \tag{4.28}$$

By applying Equation (3.74) and Jacobian matrix *S(q)*, we will get the following equation:

$$\dot{e_y} = -(d + e_x)\omega - \dot{x_r}\sin(\theta) + \dot{y_r}\cos(\theta) \tag{4.29}$$

By following a similar step in $\dot{e_x}$ derivation, Equation (4.29) is simplified in the following forms:

$$\begin{aligned} \dot{e_y} &= -(d + e_x)\omega - \dot{x_r}\sin(\theta_r - e_\theta) + \dot{y_r}\cos(\theta_r - e_\theta) \\ &= -(d + e_x)\omega + v_r\sin(e_\theta) \end{aligned} \tag{4.30}$$

- ✓ *The derivative of e_θ is provided as follows:*

$$\dot{e_\theta} = \dot{\theta_r} - \dot{\theta} = \omega_r - \omega \tag{4.31}$$

Therefore, posture error dynamics $\dot{e_p}$ are provided by:

$$\dot{e_p} = \begin{bmatrix} \dot{e_x} & \dot{e_y} & \dot{e_\theta} \end{bmatrix}^T = \begin{bmatrix} \omega\, e_y - v + v_r \cos(e_\theta) \\ -\omega\left(d + e_x\right) + v_r \sin(e_\theta) \\ \omega_r - \omega \end{bmatrix} \tag{4.32}$$

Suppose the center of mass (CoM) of a WMR is on a wheel axis ($C = P$), i.e., $d = 0$. The error posture configuration is simplified as follows:

$$\dot{e_p} = \begin{bmatrix} \dot{e_x} & \dot{e_y} & \dot{e_\theta} \end{bmatrix}^T = \begin{bmatrix} \omega\, \mathrm{e}_y - v + v_r \cos(e_\theta) \\ -\omega e_x + v_r \sin(e_\theta) \\ \omega_r - \omega \end{bmatrix} \tag{4.33}$$

The control law, which makes a system asymptotically stable, will be designed based on the pose error dynamics obtained in Equation (4.24). This control law calculates auxiliary velocity control inputs as a function of pose error and input velocities based on the pose error's time derivatives, as shown in Equation (4.33) using the Lyapunov theory.

4.2.2.1. Lyapunov Stability Analysis

System stability can be analyzed in different ways in a control system. The Lyapunov function stability analysis is widely used to stabilize a system at an equilibrium point. Any Lyapunov function candidate *V(q)* must satisfy the following properties: (Hadi & Younus, 2020):

1. $V(q)$ this function and its derivatives are continuous;
2. $V(q) = 0$, for all $q = 0$;
3. $V(q) > 0$, for all $q \neq 0$;
4. $\dot{V}(q) < 0$, for all $q \neq 0$.

Let us propose a scalar function *V* as a Lyapunov function candidate (Mai et al., 2021):

$$V = \frac{1}{2}\left(e_x^2 + e_y^2\right) + \frac{1}{K_y}\left(1 - \cos(e_\theta)\right) \tag{4.34}$$

Therefore, for $K_y > 0$ one can see $V \geq 0$ otherwise, $V = 0$ if $e_p = 0$ and $V > 0$ if $e_p \neq 0$. The above candidate function V is a positive definite function. i.e., the Lyapunov function candidate satisfies properties 1, 2, and 3. A time derivative of a Lyapunov function *V* with a derivative posture error as in Equation (4.33) is given as:

$$\dot{V} = \dot{e_x}\, e_x + \dot{e_y}\, e_y + \frac{\dot{e_\theta} \sin(e_\theta)}{K_y} = -e_x \left[v - v_r \cos(e_\theta) \right] - \sin(e_\theta) \left[\frac{(\omega_r - \omega)}{K_y} - v_r e_y \right] \leq 0 \quad (4.35)$$

The velocity control input that satisfies the Lyapunov function candidate by using a backstepping control law is formulated as follows:

$$w_c = \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} K_x e_x + v_r \cos(e_\theta) \\ \omega_r + K_y v_r e_y + K_\theta v_r \sin(e_\theta) \end{bmatrix} \quad (4.36)$$

where K_x, K_y, and K_θ are positive constants adjusted to reduce the posture error. The velocity control rule described above is called a kinematic-based backstepping controller, and its stability is verified using the Lyapunov function.

Proposition 4.1: If the Equation (4.36) is used as a control law for an auxiliary velocity control input, $\dot{e_p} = 0$ it is a stable equilibrium point for the reference velocity vector $w_r > 0$ (Kanayama et al., 1990).

P*roof:* Let us rewrite posture error dynamics in Equation (4.33) with an auxiliary velocity input as:

$$\dot{e_p} = \begin{bmatrix} \dot{e_x} \\ \dot{e_y} \\ \dot{e_\theta} \end{bmatrix} = \begin{bmatrix} \left(\omega_r + v_r \left(K_y e_y + K_\theta \sin(e_\theta) \right) \right) \mathrm{e}_y - K_x e_x \\ -\left(\omega_r + v_r \left(K_y e_y + K_\theta \sin(e_\theta) \right) \right) \mathrm{e}_x + v_r \sin(e_\theta) \\ -v_r \left(K_y e_y + K_\theta \sin(e_\theta) \right) \end{bmatrix} \quad (4.37)$$

The time derivatives of a Lyapunov function with control rule and pose error are given as:

$$\dot{V} = \dot{e_x}\, e_x + \dot{e_y}\, e_y + \frac{1}{K_y} \dot{e_\theta} \sin(e_\theta) = -K_x e_x^2 - \frac{K_\theta}{K_y} v_r \sin^2(e_\theta) \leq 0 \quad (4.38)$$

It is clear that $\dot{V} \leq 0$, otherwise, $\dot{V} = 0$ if $e_p = 0$ and $\dot{V} < 0$ if $e_p \neq 0$. Therefore, the derivative of a Lyapunov function is a negative-definite function. The system is asymptotically stable at an equilibrium point $e_p = 0$ under the condition that reference velocity $w_r = [v_r\ \omega_r]^T$ and proposed constant positive gains (K_x, K_y, and K_θ) are bounded and continuous.

As in the above demonstration, the proposed system is stable with any positive combination of controller gains. However, the value of this combination affects system performance. In

the case of non-oscillatory and not too slow response, we should find optimal controller parameter values. A genetic algorithm is used for controller tuning purposes.

4.2.3. BSC Design for Dynamic Control Using Feedback Linearization

The following controller design is a backstepping controller based on the feedback linearization technique. This method will improve the trajectory tracking response by linearizing the nonlinear system dynamics. Feedback linearization techniques are used throughout, changes in variables, and appropriate control input to linearize a nonlinear system to an equivalent linear system. This linearization method aims to create a control input that transforms between the new input and output (Klančar et al., 2017).

The dynamic model of a nonholonomic WMR is linearized in this document using a nonlinear feedback linearization technique. Figure 4.3 shows a system model with a proposed backstepping controller with a nonlinear feedback linearization torque.

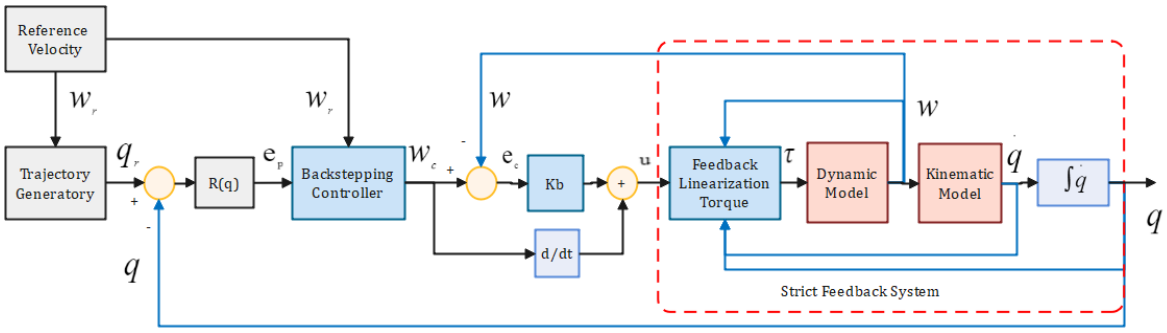

Figure 4.3: Backstepping controller with a proposed system model

The control rule proposed in Equation (4.36) and the selected velocity control input *w(t)* are used for a kinematic model for the TWMR system. For the actual system to apply nonlinear feedback linearization, a controlled velocity *w(t)* is converted to a torque control $\tau(t)$. From the system model derived in the previous chapter, we had a complete nonholonomic TWMR motion equation. i.e., kinematic and dynamic equations as:

$$\dot{q} = S(q)w(t) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{4.39}$$

$$\overline{H}(q)\dot{w}(t) + \overline{C}(q,\dot{q})w(t) + \overline{F}(\dot{q}) + \overline{G}(q) + \overline{\tau}_d = \overline{E}(q)\tau \tag{4.40}$$

We can linearize a nonlinear dynamic model of nonholonomic mobile robots by applying a nonlinear feedback linearization technique. Let us define an auxiliary velocity input $u =$

$\dot{w}(t)$, and w instead of $w(t)$ by applying a feedback linearization technique, we had a feedback torque as (Benchouche et al., 2021):

$$\tau = \overline{E}^{-1}(q)\left[\overline{H}(q)u + \overline{C}(q,\dot{q})w + \overline{F}(\dot{q}) + \overline{G}(q) + \overline{\tau}_d\right] \tag{4.41}$$

In this torque feedback linearization, it is assumed that all dynamical quantities; $\overline{H}(q)$, $\overline{C}_n(q,\dot{q}), \overline{F}(\dot{q}), \bar{G}(q)$ are precisely known and $\overline{\tau}_{\mathrm{d}}$ are assumed to be zero. That is;

$$\tau = \overline{E}^{-1}(q)\left[\overline{H}(q)u + \overline{C}(q,\dot{q})w\right] \tag{4.42}$$

The transformation of Equation (4.40) to Equation (4.41) is achieved by a nonlinear feedback linearization, which linearizes and simplifies the system's dynamic equation. And we can convert a dynamic model to the following forms by neglecting unknown disturbances $\overline{\tau}_d$:

$$\begin{cases} \overline{H}(q)\dot{w} + \overline{C}(q,\dot{q})w = \overline{E}(q)\left[\overline{E}^{-1}(q)\left(\overline{H}(q)u + \overline{C}(q,\dot{q})w\right)\right] \\ \dot{w} = u \end{cases} \tag{4.43}$$

From Equation (4.39) and auxiliary velocity u, we had:

$$\begin{cases} \dot{q} = S(q)w(t) \\ \dot{w}(t) = u \end{cases} \tag{4.44}$$

A system model in Equation (4.44) can be linearized by using feedback linearization because the system has a form of strict feedback system as follows:

$$\begin{aligned} \dot{x} &= f(x) + g(x)z \\ \dot{z} &= u \end{aligned} \tag{4.45}$$

At the equilibrium point, a system is asymptotically stable, i.e., $(x,z) = 0$; *f(x)* and *g(x)* are known smooth functions. The main idea of feedback control is to stabilize a system at its origin, $(x,z) = 0$. The controller that stabilizes a system at equilibrium points $(x,z) = 0$ is called a backstepping controller.

The linearized TWMR system in Equation (4.44) has a similar structure to a strict feedback system in Equation (4.45). Therefore, a backstepping controller can stabilize a linearized

system at equilibrium. The proposed nonlinear feedback acceleration control input used to stabilize the system in Equation (4.44) is given as follows (Fierro & Lewis, 1997):

$$u = \dot{w}_c + K_b(w_c - w) \tag{4.46}$$

where K_b is a positive definite matrix given as;

$$K_b = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \tag{4.47}$$

We had a kinematic controller w_c from Equation (4.36) as:

$$w_c = \begin{bmatrix} v_c \\ \omega_c \end{bmatrix} = \begin{bmatrix} v_r \cos(e_\theta) + K_x e_x \\ \omega_r + K_y v_r e_y + K_\theta v_r \sin(e_\theta) \end{bmatrix} \tag{4.48}$$

Moreover, its time derivatives become as follows:

$$\begin{aligned} \dot{w}_c &= \begin{bmatrix} \dot{v}_r \cos(e_\theta) - v_r \dot{e}_\theta \sin(e_\theta) + K_x \dot{e}_x \\ \dot{\omega}_r + K_y \dot{v}_r e_y + K_y v_r \dot{e}_y + K_\theta \dot{v}_r \sin(e_\theta) + K_\theta v_r \dot{e}_\theta \cos(e_\theta) \end{bmatrix} \\ &= \begin{bmatrix} \dot{v}_r \cos(e_\theta) \\ \dot{\omega}_r + \dot{v}_r \left(K_y e_y + K_\theta \sin(e_\theta) \right) \end{bmatrix} + \begin{bmatrix} K_x & 0 & -v_r \sin(e_\theta) \\ 0 & K_y v_r & K_\theta v_r \cos(e_\theta) \end{bmatrix} \dot{e}_p \end{aligned} \tag{4.49}$$

If reference velocity inputs are constants, the time derivatives of the auxiliary velocity control input become:

$$\dot{w}_c = \begin{bmatrix} K_x & 0 & -v_r \sin(e_\theta) \\ 0 & K_y v_r & K_\theta v_r \cos(e_\theta) \end{bmatrix} \dot{e}_p \tag{4.50}$$

The control law proposed in Equation (4.46) is also valid for time-varying reference velocity, $w_r(t) = [v_r(t) \quad \omega_r(t)]^T$. The Lyapunov stability function that reaches the stability of a proposed backstepping controller given in Equation (4.46) will be explained in the following.

4.2.3.1. Lyapunov Stability Analysis

Proposition 4.2: For a nonholonomic dynamics motion equation in Equations (4.39) and (4.40) the nonlinear feedback control law u have proposed as in Equation (4.46) with a nonlinear feedback control input is given by Equation (4.41). Therefore, origin $e_p = 0$ is an asymptotically stable point, and a velocity vector of the system satisfies $w \to w_c$ as $t \to \infty$ (Fierro & Lewis, 1997).

Proof: Define an auxiliary velocity error:

$$e_c = \begin{bmatrix} e_v \\ e_\omega \end{bmatrix} = [w_c - w] = \begin{bmatrix} v_r \cos(e_\theta) + K_x e_x - v \\ \omega_r + K_y v_r e_y + K_\theta v_r \sin(e_\theta) - \omega \end{bmatrix} \tag{4.51}$$

From an auxiliary velocity control, we had $\dot{w}(t) = u$, then Equation (4.46) is rearranged as:

$$\dot{w}(t) - \dot{w}_c = K_b(w_c - w) \tag{4.52}$$

By comparing Equations (4.51) and (4.52) we obtain auxiliary velocity error derivatives as:

$$\dot{e}_c = -K_b\left(w_c - w\right) = -K_b e_c \tag{4.53}$$

The equation above shows that the system is asymptotically stable, and the velocity vector satisfies $w \to w_c$ as $t \to \infty$. On the other hand:

$$\lim_{t\to\infty} e_v = 0 \text{ and } \lim_{t\to\infty} e_\omega = 0 \tag{4.54}$$

The Lyapunov function candidate V that guarantees the stability of a system with a proposed control rule is given in the following forms:

$$V = K_x\left(e_x^2 + e_y^2\right) + \frac{2K_x}{K_y}\left(1 - \cos(e_\theta)\right) + \frac{1}{K_b}\left(e_v^2 + \frac{K_x}{K_y K_\theta v_r} e_\omega^2\right) \tag{4.55}$$

where $V \geq 0$ and $V = 0$ only if $e_p = 0$ and $e_c = 0$, which is considered as an equilibrium point. The time derivatives of the Lyapunov candidate V are:

$$\dot{V} = 2K_x\left(\dot{e}_x e_x + \dot{e}_y e_y\right) + \frac{2K_x}{K_y}\dot{e}_\theta \sin(e_\theta) + \frac{2}{K_b}\left(\dot{e}_v e_v + \frac{K_x}{K_y K_\theta v_r}\dot{e}_\omega e_\omega\right) \tag{4.56}$$

By substituting the derivative of pose error configuration and auxiliary error vector with a kinematic controller, we finally get a simplified function as follows:

$$\dot{V} = -K_x^2 e_x^2 - \frac{K_x K_\theta}{K_y} v_r \sin^2(e_\theta) - \left(e_v + K_x e_x\right)^2 - \frac{K_x}{K_y K_\theta v_r}\left(e_\omega + K_\theta v_r \sin(e_\theta)\right)^2 \tag{4.57}$$

The derivative of a candidate function is a negative definite function, i.e., $\dot{V} \leq 0$, and the error is $e = \begin{bmatrix} e_p^T & e_v^T \end{bmatrix}^T$ bounded. Consider that $e_c \to 0$ as $t \to \infty$, then the limit of the Lyapunov function derivatives as $t \to \infty$ is:

$$\begin{aligned}\lim_{t\to\infty}\dot{V} &= \lim_{t\to\infty}\left(-K_x^2e_x^2-\frac{K_xK_\theta}{K_y}v_r\sin^2(e_\theta)-\left(e_v+K_xe_x\right)^2-\frac{K_x\left(e_\omega+K_\theta v_r\sin(e_\theta)\right)^2}{K_yK_\theta v_r}\right)\\ &= -2\lim_{t\to\infty}\left(K_x^2e_x^2+\frac{K_xK_\theta}{K_y}v_r\sin^2(e_\theta)\right)\end{aligned} \tag{4.58}$$

Since $\dot{V}\le 0$ we know that V does not increase, and it converges to some constant value, therefore, $\dot{V}$ goes to zero as $t\to\infty$:

$$\lim_{t\to\infty}\dot{V}=\lim_{t\to\infty}\left(K_x^2e_x^2+\frac{K_xK_\theta}{K_y}v_r\sin^2(e_\theta)\right)=0 \tag{4.59}$$

From the above equation, we get the following:

$$\lim_{t\to\infty}e_x=0,\ \text{and}\ \lim_{t\to\infty}e_\theta=0 \tag{4.60}$$

And from Equation(4.31), we had $\dot{e_\theta}=\omega_r-\omega$ then,

$$\lim_{t\to\infty}\dot{e_\theta}=\lim_{t\to\infty}\left(\omega_r-\omega\right)=0 \tag{4.61}$$

By considering an auxiliary angular velocity error e_ω in Equation (4.51) with Equations (4.60) and (4.61) we have:

$$\lim_{t\to\infty}\left(-\omega+\omega_r+K_yv_re_y+K_\theta v_r\sin(e_\theta)\right)=\lim_{t\to\infty}\left(K_yv_re_y\right)=0 \tag{4.62}$$

From the above equation, by assuming $v_r>0$, we get:

$$\lim_{t\to\infty}e_y=0 \tag{4.63}$$

Therefore, the equilibrium point $e=\left[e_p^T\quad e_v^T\right]^T=0$ is uniformly asymptotically stable. The above stability analysis results show that the backstepping controller in Equation (4.46) and a nonlinear feedback torque in Equation (4.41) will make a complete nonholonomic mobile robot system. The tracking performance of a proposed control depends on kinematic-based controller gains (K_x, K_y, and K_θ) and backstepping controller gains (K_1 and K_2).

4.3. Nonlinear PID Controller

4.3.1. Introduction to PID Controller

The PID controller has been widely used among numerous classical industrial process control controllers since 1930. Because of its simplicity in structure and ease of implementation, PID is widely applicable in closed-loop system control (Zaidner et al., 2010). As its name describes, a PID controller is a combination of proportional, integral, and derivative functions. It has three control behaviors (Song, 2018).

The proportional controller (P): gives an output proportional to the error signal. The resulting error is multiplied by the proportional gain constant to get the output. However, offset has always existed in this controller.

The Integral controller (I): provides a control action to eliminate the steady-state error over a period of time until the error value reaches zero. This controller improves the limitations of a P-controller. However, overshoot will be the limitation of this I-type controller, and it cannot predict the future state of the error signal. It is used with a combination of P-controllers (PI).

The derivative controller (D): will overcome the I-controller's limitation by anticipating an error's future state. Its output depends on the rate of error change with respect to time, multiplied by a derivation constant.

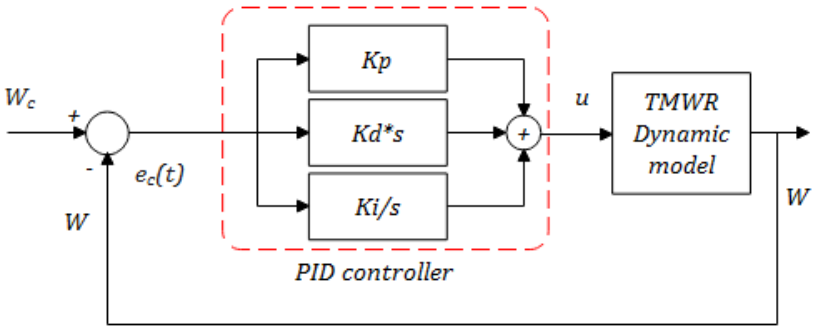

Figure 4.4: Basic PID controller structure with TWMR dynamic model

The time-domain representation of the standard PID controller is given as follows;

$$u(t) = K_p\left[e_c(t) + \frac{1}{T_i}\int_0^t e_c(t)dt + T_d\frac{de_c(t)}{dt}\right] = K_p e_c(t) + K_i\int_0^t e_c(t)dt + K_d\frac{de_c(t)}{dt} \tag{4.64}$$

$$u(s) = K_p + K_i\frac{1}{s} + K_d s \tag{4.65}$$

where u is PID controller output, K_p is the proportional gain, T_i is the integral time, T_d is the derivative time, $K_i = K_p/T_i$ is the integral gain, $K_d = K_p T_d$ is the derivative gain in the case of a parallel PID controller. An error signal $e_c(t)$ is given as the difference between the control and actual velocity values ($e_c(t) = w_c - w$). The integral of the error function is accumulated to achieve better tracking performance. A derivative predicts the rate of change of an error to prevent an overshoot. For TWMR velocity control, a PID controller is designed as follows:

$$u_k = K_{pk} + K_{ik}\frac{1}{s} + K_{dk}s, \quad \text{where k = 1, 2} \tag{4.66}$$

where $u_k = [u_1, u_2]^T = [u_v, u_\omega]^T$ controller output, and k is the number of controllers.

4.3.2. Nonlinear PID Controller for TWMR Velocity Control

An NPID controller whose architecture is similar to the conventional PID controller is proposed. However, the integral action contains a nonlinear function while the proportional and derivative actions are linear. Hence, the error input to the integral action is scaled by a nonlinear gain function in the product of the error and the nonlinear gain. An NPID controller used in work is similar to an NPID controller in work (JIN & SON, 2019). A proposed nonlinear PID controller block diagram is shown in Figure 4.5.

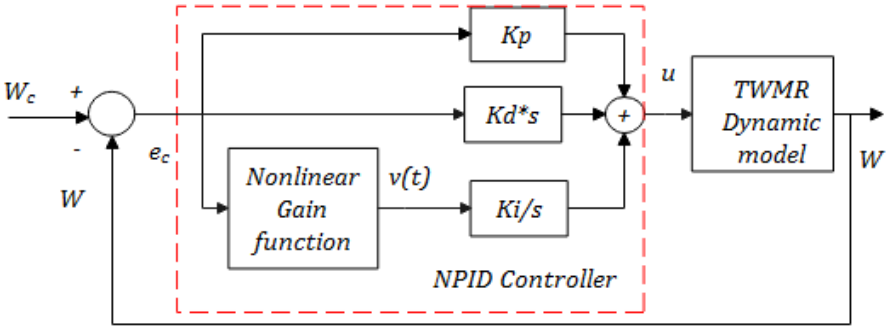

Figure 4.5: Nonlinear PID controller with TWMR dynamic model

The time-domain equation of the NPID controller for velocity control of TMWR is given as:

$$\begin{aligned} u_k(t) &= K_{pk}\left[e_c(t) + \frac{1}{T_{ik}}\int_0^t v(t)dt + T_{dk}\frac{de_c(t)}{dt}\right] \\ &= K_{pk}\, e_c(t) + K_{ik}\int_0^t v(t)dt + K_{dk}\frac{de_c(t)}{dt} \end{aligned} \tag{4.67}$$

The nonlinear scaled error function $v(t)$ in the integral and derivative control action is given as (Garbaabaa et al., 2020) :

$$v(t) = k(e)e_c(t), \text{ where, } k(e) = \exp\left(-\frac{e_c(t)^2}{2\Delta w_c^2}\right) \tag{4.68}$$

where $u_k = [u_1, u_2]^T = [u_v, u_\omega]^T$ is controller output, $k(e)$ is a nonlinear gain function, $\Delta w_c \neq 0$ is controlled velocity change, i.e., $\Delta w_c = w_c(i) - w_c(i-1)$ and i denote the discrete instant of time. According to (JIN & SON, 2019), a typical value of Δw_c is 0.5, 1, and 1.5.

A nonlinear PID controller output in Equation (4.67) with a nonlinear gain function as in Equation (4.68) is used throughout this thesis. A controller's gain parameters are adjusted using genetic algorithm optimization techniques to get an optimal value for better performance tracking.

4.4. Overall Proposed System Modeling

The WMR system model derived in Chapter 3 is a nonlinear system. A nonlinear system needs an appropriate control mechanism to control and stabilize at an equilibrium point. As in Equations (4.43) and (4.44) a backstepping control rule that stabilizes a nonholonomic system from inner to outer subsystem was designed. The auxiliary velocity control input is used to stabilize the internal subsystem at the equilibrium point $(e_c = 0)$ as $t \to \infty$ and step back out to the external subsystem at the equilibrium point $(e_p = 0)$ as $t \to \infty$. This could be achieved by using a Lyapunov stability function.

The overall system modeling using system dynamics as in Equations (3.34) and (3.88) with a proposed control rule in Equation (4.46) combined with an NPID controller as in Equation (4.67) will be designed as shown in Figure 4.6.

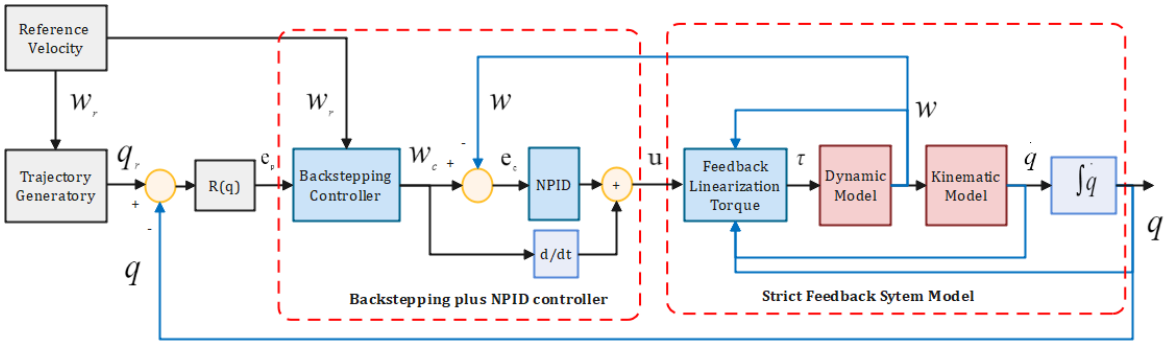

Figure 4.6: Overall system model with Backstepping and NPID controller

The trajectory tracking performance of a backstepping controller combined with a PID controller is compared with that of a backstepping controller plus an NPID controller. All

controller gains are tuned using a genetic algorithm to get optimal values for better trajectory tracking with the possible minimum posture and velocity errors.

4.5. Controller Gains Optimization by Using a Genetic Algorithm

4.5.1. Introduction to Genetic Algorithm (GA)

A genetic algorithm is a random search algorithm used to solve complex optimization problems. It employs probabilistic transition rules rather than deterministic rules and iteratively evolves a population of potential solutions known as individuals or chromosomes. Each iteration of the algorithm is referred to as a "generation." The evolution of solutions is mimicked using a fitness function and genetic operators such as selection (reproduction), crossover, and mutation (Messom, 2002).

The genetic algorithm has the following phases; initial population, fitness function, selection, crossover and mutation. Figure 4.7 shows the pseudo-code of the GA technique process. The process begins with a set of individuals called a population.

Start
Set t = 0;
Generate initial population $P(t)$;
Compute the fitness of an individual in $P(t)$;
do while < Stop condition for not satisfied >
Set t = t+1;
Select from individual $P(t-1)$ to set a tentative population $\overline{P}(t)$;
Perform Cross-over individual in $\overline{P}(t)$;
Perform Mutation of an individual in $\overline{P}(t)$;
Compute the fitness of an individual in a new population $P(t)$;
end while
Output the best individual $P(t)$ as the best solution;
Stop

Figure 4.7: Pseudo-code of the genetic algorithm

The fitness function determines how fit an individual is, which means an individual's ability to compete with others. The fittest individuals are chosen, and their genes are passed down to the next generation through selection. Crossover occurs when each pair of parents is mated, crossover points are selected randomly from within a gene, and mutation occurs to maintain diversity within the population and prevent premature convergence. The algorithm

terminates if the population has converged (i.e., does not produce significantly different offspring from the previous generation). Then it is said that the GA has provided a solution to our problem. This process keeps on iterating, and in the end, a generation with the fittest individuals will be found. The fittest individual is considered a solution to a problem (Alam et al., 2021).

In a nature-inspired optimization tool solver, there is no analytical proof that the obtained solution is globally optimal or not. The searching mechanism is based on probabilistic random functions. However, in order to select an optimization tool for an appropriate system, we considered the particular behavior of the solvers. For example, convergence rate, the program's complexity, the type of cost function, and data handling capacity are a few behaviors we should consider. Therefore, GA optimization can handle a multi-objective function, is easily understandable, and programming is less complex. It is a widely used optimization tool solver in mixed or continuous/discrete systems.

4.5.2. Objective Function

The GA optimization tunes a controller to obtain optimal parameters with a possible minimum objective function. The objective function is obtained by using integral error criteria: integral time-weighted absolute error (ITAE), integral absolute error (IAE), or integral square error (ISE). The time function of the objective/cost function using integral error criteria is as follows:

$$ISE = \int_0^{t_f} e^2(t)dt,\ IAE = \int_0^{t_f} |e(t)|dt,\ \text{ and }\ ITAE = \int_0^{t_f} t\,|e(t)|dt \tag{4.69}$$

where *e(t)* represents the difference between the desired and actual values, and t_f is the time duration. Because it is easily analyzed, ISE is frequently utilized for optimum control. IAE has more sensitivity than ISE because it uses the absolute magnitude of the error. However, ITAE penalizes an error that persists long, resulting in more significant discrimination than IAE or ISE (JIN & SON, 2019).

A TMWR is a system with a multi-input and multi-output (MIMO). The cost function used to minimize is a sum of individual cost functions. That is, the cost function is rewritten as (Al-Mayyahi et al., 2016);

$$ITAE = \int_0^{t_f} \left(t\ \left|e_p(t)\right| + t\ \left|e_c(t)\right| \right)\ dt \tag{4.70}$$

where $e_p = R(\theta)[q_r - q]$ is pose error and $e_c = w_c - w$ is a velocity error. This cost function should be reduced to minimum values for better tracking performance response.

4.5.3. Parameter Settings of the Genetic Algorithm

The following parameters are used in Matlab/Simulink R2021a software simulation to obtain backstepping, PID, and NPID controller gains parameters. A kinematic-based backstepping has three gains $(K_x, K_y,$ and $K_\theta)$ as well as backstepping gains (k_1 and k_2) optimized to reduce a tracking error to the minimum possible value. Similarly, PID and NPID controllers have three parameters $(K_p, K_i,$ and $K_d)$. However, the dynamic model has two controlled variables: angular velocity and linear velocity.

In GA optimization techniques, the optimal solution to a problem is obtained depending on a parameter we set during simulation. The best solution to a problem is obtained if more populations, a maximum number of generations, and more sampling time are selected. However, it takes a long time to finish the simulation. If a small population and a few generations are set, a solution to a problem is quickly found, but the solution is less accurate. Therefore, the selection of a number of populations, generations, and lap time is based on the performance needed from the system.

Table 4.1 shows a breakdown of the parameters used in the gain optimization simulations for the backstepping plus NPID and backstepping plus PID controllers. Genetic algorithm optimization is executed in Matlab using Matlab-code or an optimization toolbox app. A Matlab code and Simulink function used for system simulation and optimization are provided in Appendices A and B. It is important to note that the overall system is optimized simultaneously for better performance and to overcome the effects of one loop on the other.

Table 4.1: Parameter settings of the genetic algorithm optimization

Parameters of the Algorithm	
Maximum generation	30
Population size	30
No of parameters	9
Lower Boundary (LB)	[0 0 0 0 0 0 0 0 0]
Upper Boundary (UB)	[100 1 1 100 1 1 100 100 100]

CHAPTER 5

5. RESULTS AND DISCUSSION

5.1. Introduction

The mathematical modeling of TWMR with a proposed control algorithm is discussed and explained in the previous section. The backstepping controller combined with an NPID controller is designed for TWMR trajectory tracking control. The proposed controller is simulated on Matlab/Simulink R2021a software, as shown in Figure 5.1. The proposed control method performance and the capability to track a given trajectory by changing the initial position and its robustness by applying an unknown disturbance are discussed.

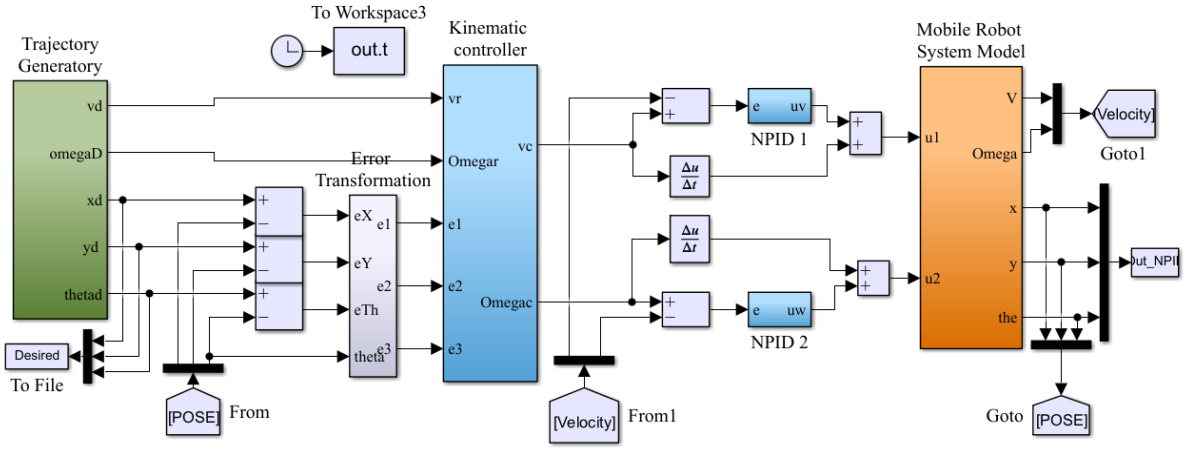

Figure 5.1: Matlab/Simulink system simulation block diagram

For a simple understanding, the designed system in Figure 5.1 is described as follows:

- The dark green color denotes a trajectory generator,
- The light-blue color denotes the proposed controllers (backstepping and NPID controller),
- The yellow color denotes a TWMR system model.

5.2. Simulation Results

This thesis work optimizes controller gains (backstepping plus PID controller, backstepping plus NPID controller, and backstepping-only controller) by the genetic algorithm. The simulation is done with a solver with a fixed step size of 0.001 sec and a simulation stop time of 5 sec. The solver type is a fourth-order differential equation (Runge-Kutta). As shown in Table 5.1 and Table 5.2, the best values of controller gains with minimized

objective function (ITAE) values are obtained. ITAE is used to reduce a cost function in this work because it is used for an error that persists for a long time.

Table 5.1: Backstepping and NPID controller gains optimized by genetic algorithm

Backstepping and NPID Controller gain parameters								
NPID 1			NPID 2			BSC		
Kp1	Ki1	Kd1	Kp2	Ki2	Kd2	Kx	Ky	K_θ
71.4881	0.0441	0.9483	99.8071	0.0251	0.1802	33.3095	98.7610	10.2058
ITAE = 0.0940967								

Table 5.2: Backstepping and PID controller gains optimized by genetic algorithm

Backstepping and PID Controller gain parameters								
PID 1			PID 2			BSC		
Kp1	Ki1	Kd1	Kp2	Ki2	Kd2	Kx	Ky	K_θ
48.1210	1.5982	9.1775	96.3155	0.0193	1.1368	9.1938	53.4033	14.2411
ITAE = 0.1095								

5.2.1. Trajectory Tracking of TMWR using Backstepping and NPID Controller

To test a controller's performance for a capability to track a given reference trajectory, adaptability, and robustness in an unknown disturbance and model uncertainty, we consider two scenarios for the reference trajectory.

A linear reference trajectory (straight shape) is applied to a motion and is given as:

$$\begin{cases} x_r = t \\ y_r = 2*t \end{cases}, \text{ for all } t \geq 0 \tag{5.1}$$

Furthermore, a nonlinear reference trajectory (eight-shape) is given as:

$$\begin{cases} x_r = -5*\sin(t/5) \\ y_r = 10*\sin(t/10) \end{cases}; \text{ for all } t \geq 0 \tag{5.2}$$

The reference trajectory and velocity are generated according to Equation (4.1) to Equation (4.4), derived in the previous section.

5.2.1.1. Tracking Response of a Linear Reference Trajectory (Straight-Shape)

In linear reference trajectory tracking, a robot rapidly follows a given trajectory because the controller can easily anticipate the future behavior of a tracking line or tracking error. The Root-Mean Square (RMS) value of the tracking error is used to compute the performance of a robot in its numerical value.

Figure 5.2 and Figure 5.3 show that a robot tracks a given reference trajectory with a minimum tracking error at $q_0 = [0, 0, 0]^T$ initial states between 0 and 10 sec. In a given time range, a robot tracking performance response in the x-direction position has a short settling time of 0.2 sec and a peak value of 0.0114 m with a peak time of 0.03 sec, as in Figure 5.2.

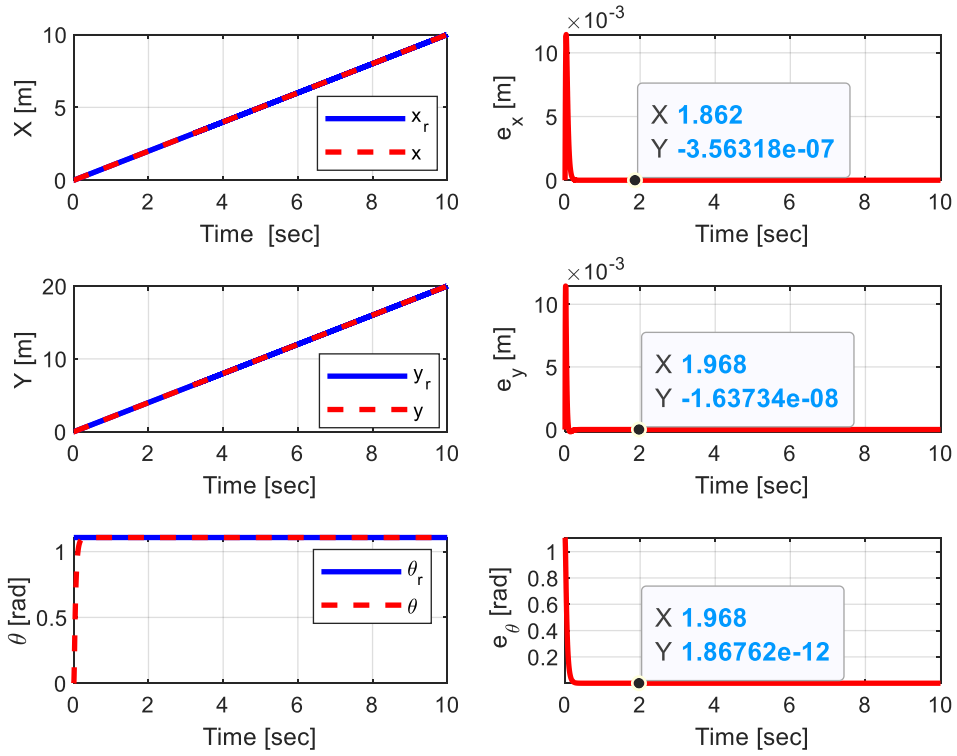

Figure 5.2: x, y, and θ variables tracking response and tracking error with an initial position of [0, 0, 0]

In addition, the y-axis tracking error has a 0.1 sec settling time with a peak value of 0.0115 m at 0.015 sec. Regarding its heading angle, the robot needs 0.2 sec to reach a reference trajectory. On the other hand, the robot's position error (trajectory tracking error) is calculated from position tracking errors in the x and y-direction. Mathematically, expressed as $\left(e_x^2 + e_y^2\right)^{0.5}$.

As shown in Table 5.3, the RMS value of tracking errors achieved by a proposed controller in the case of linear trajectory tracking is minimum with [0, 0, 0] initial robot state.

Table 5.3: RMS values of linear trajectory tracking error with [0, 0, 0] initial state

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
RMS value	0.0008	0.0006	0.0613	0.0010

The result clearly shows that a robot needs only about 0.2 sec to track a reference trajectory when it starts its movement from an [0, 0, 0] initial position, as shown by the tracking error in Figure 5.3. The trajectory tracking error has only a 0.0152 m peak value at 0.02 sec. Therefore, the robot tracking performance achieved by a proposed controller with [0, 0, 0] initial states is almost perfect, regardless of the peak value at the start of a movement, which converges to zero quickly.

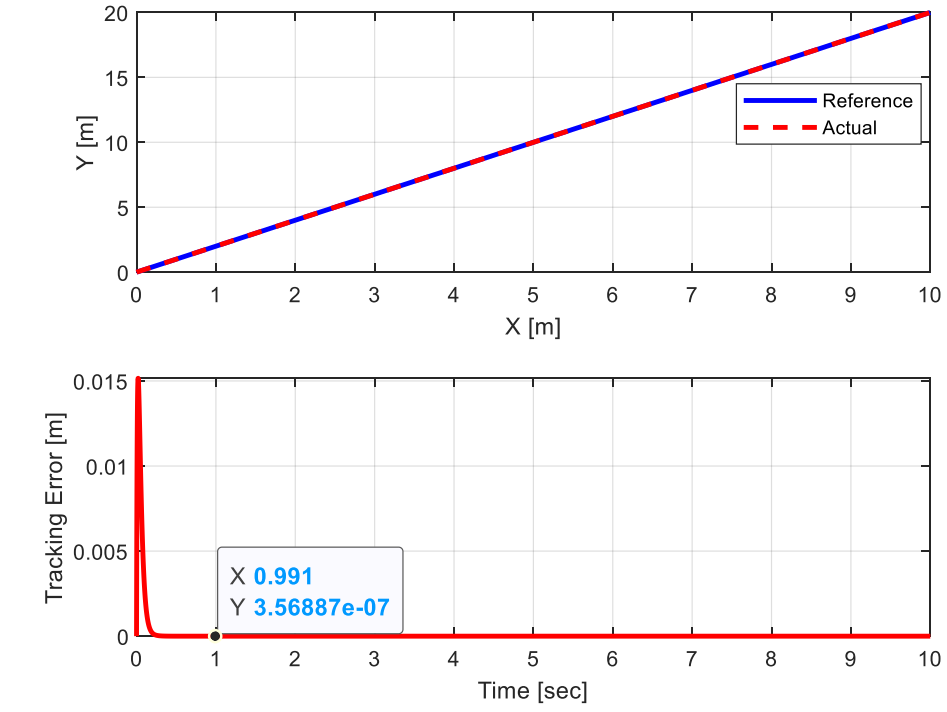

Figure 5.3: Linear trajectory tracking and position error with [0, 0, 0] initial position

However, a limitation of the backstepping controller is that it produces a significant overshoot response when the robot's initial position is changed. In this case, the robot runs at high speed. In order to test the capability of the robot to track a given reference trajectory with a proposed method, changing the initial state and applying an unknown disturbance to the system is essential.

Trajectory tracking performance with [2, 1, pi/2] initial robot position

The trajectory tracking response is shown in Figure 5.4, and Figure 5.5 shows that the robot follows its reference trajectory with [2, 1, pi/2] initial position change. From Figure 5.4, the robot has a considerable peak value at the start of motion; then, this value is decreased to zero in a short time. Thus, the proposed method rejects the effect of initial position changes on a robot's performance with a short settling time and a slight overshoot.

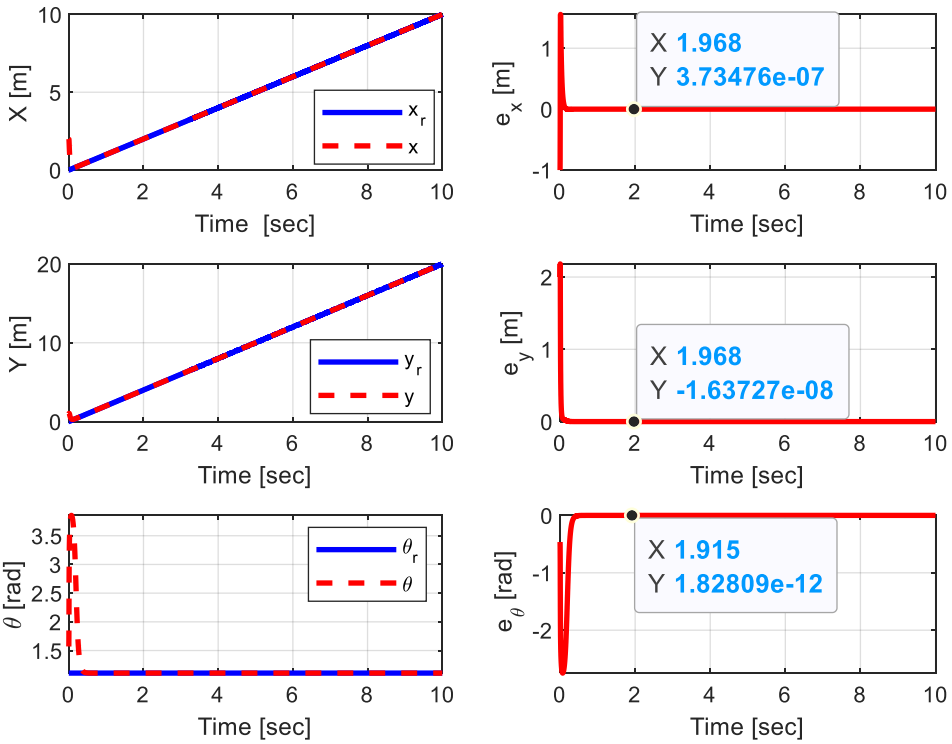

Figure 5.4: x, y, and θ variables tracking response and tracking error with the initial position of [2, 1, pi/2]

Figure 5.5 shows the tracking performance of a robot with a proposed control method when it starts its movement from [2, 1, pi/2] at the initial position. As we have seen, the obtained results are a smooth response and converge quickly to their steady-state error. The trajectory tracking error of the robot initially has a considerable peak value due to the initial position change. However, this error is reduced to almost zero in a short time.

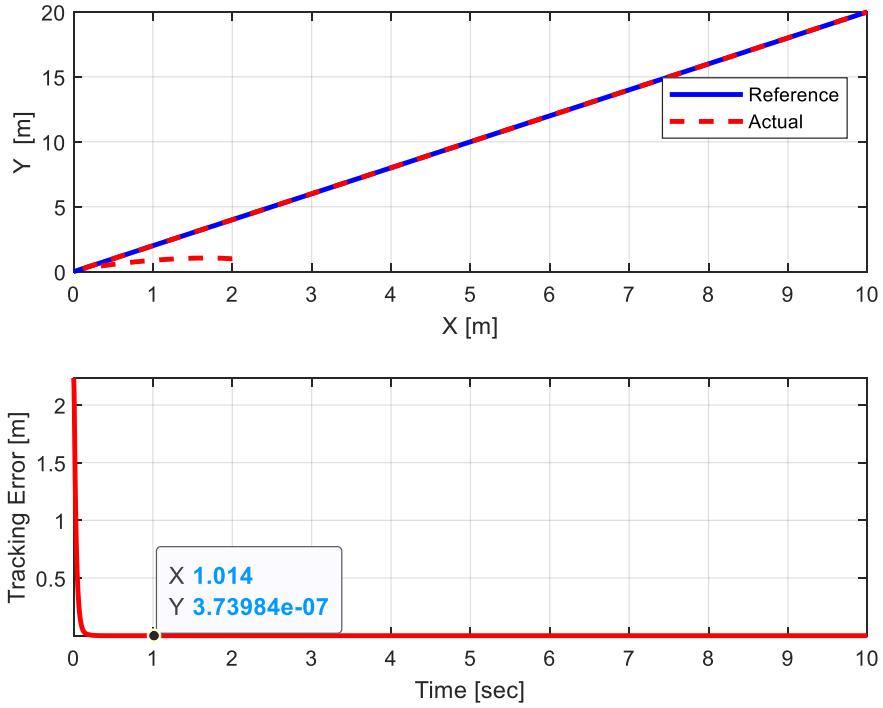

Figure 5.5: Linear trajectory tracking and tracking error with [2,1, pi/2] initial position

As a result, as shown in Table 5.4, the proposed controller has a small RMS value of tracking error, and its trajectory tracking response is also preferable and acceptable.

Table 5.4: RMS values of linear trajectory tracking error with [2, 1, pi/2] as an initial state

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
RMS value	0.0746	0.0694	0.3436	0.1018

Trajectory tracking performance with [3, 0, 2pi/3] initial robot position

Similarly, if the robot starts its movement at the initial position of [3, 0, 2pi/3], its trajectory tracking performance is shown in Figure 5.6. The result shows that the proposed controller stabilizes and controls the robot on a given track within an initial position change. Table 5.5 shows RMS tracking error values with [3, 0, 2pi/3] initial robot position.

Table 5.5: RMS values of linear trajectory tracking error with [3, 0, 2pi/3] initial state

Tracking errors RMS value	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
	0.1306	0.0727	0.2480	0.1494

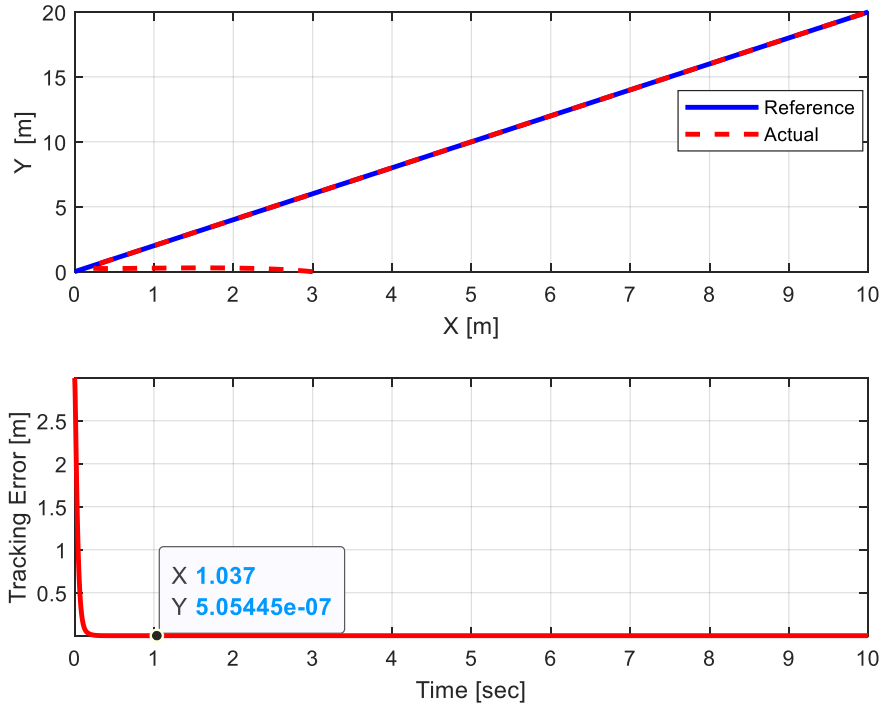

Figure 5.6: Linear trajectory and tracking error response with [3, 0, 2pi/3] initial position

As we have seen from the tracking responses, it is clear that the proposed system achieved the best trajectory tracking performance for a given reference trajectory with a small tracking error (i.e., almost negligible). In addition, a controller adapts to a robot's initial state change. Therefore, the trajectory tracking and stabilization of the robot with the proposed controller are almost perfect on a given reference trajectory. Thus, the proposed controller can easily predict the future behavior of a tracking error of a TWMR in the case of a linear reference trajectory.

5.2.1.2. Tracking Response of a Nonlinear Reference Trajectory (Eight-Shape)

For the second scenario, a nonlinear trajectory (eight-shape) is considered as a reference trajectory input. Since the objective of the backstepping controller is to stabilize a robot at an equilibrium point, the robot's performance at the initial state [0, 0, 0] is shown in Figure 5.7 to Figure 5.11. In this case, the reference robot angle is a time-dependent function, not a constant.

As shown in Figure 5.7, the robot trajectory tracking of a sinusoidal reference trajectory in the x-direction is almost perfect control with a tracking error peak value of 0.0077 m at 0.021 sec of peak time regardless of undershoot at the start of the movement.

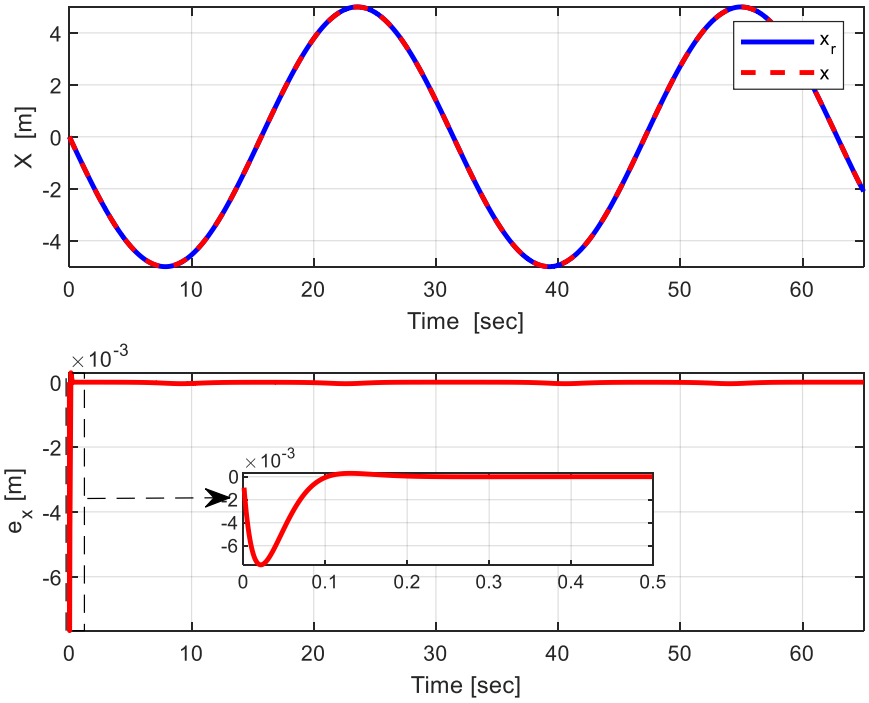

Figure 5.7: x-position tracking response and tracking error with [0, 0, 0] initial position (nonlinear trajectory)

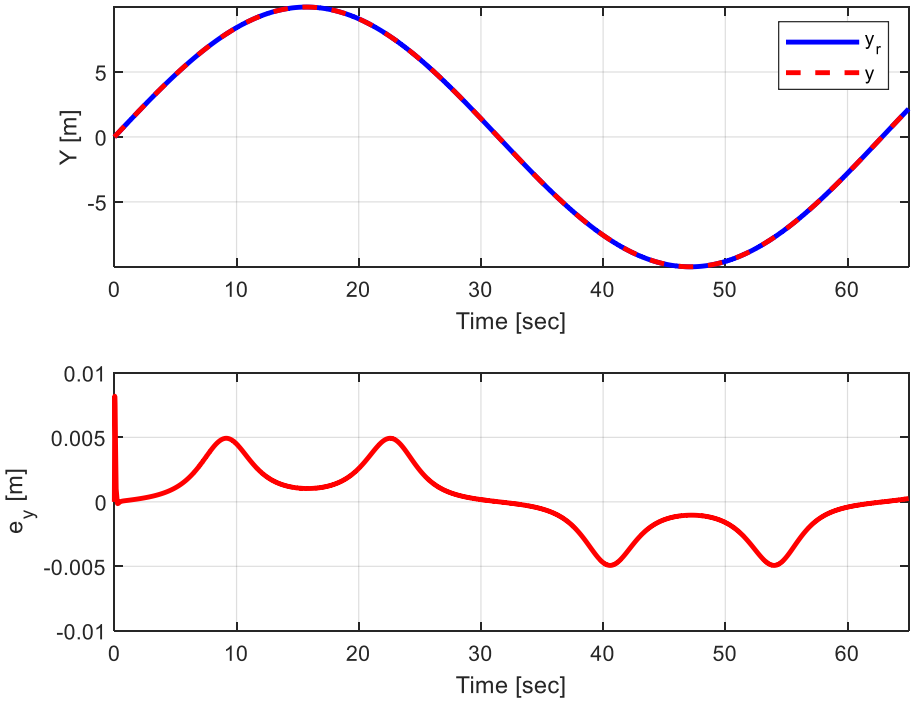

Figure 5.8: y-position tracking response and tracking error with [0, 0, 0] an initial position (nonlinear trajectory)

Furthermore, a robot's tracking performance in the y-position has a peak tracking error of about 0.0082 m with a 0.031 sec as in Figure 5.8. After the robot settled on its reference trajectory, the tracking error only varied between-0.0049 m and +0.0049 m within the reference trajectory.

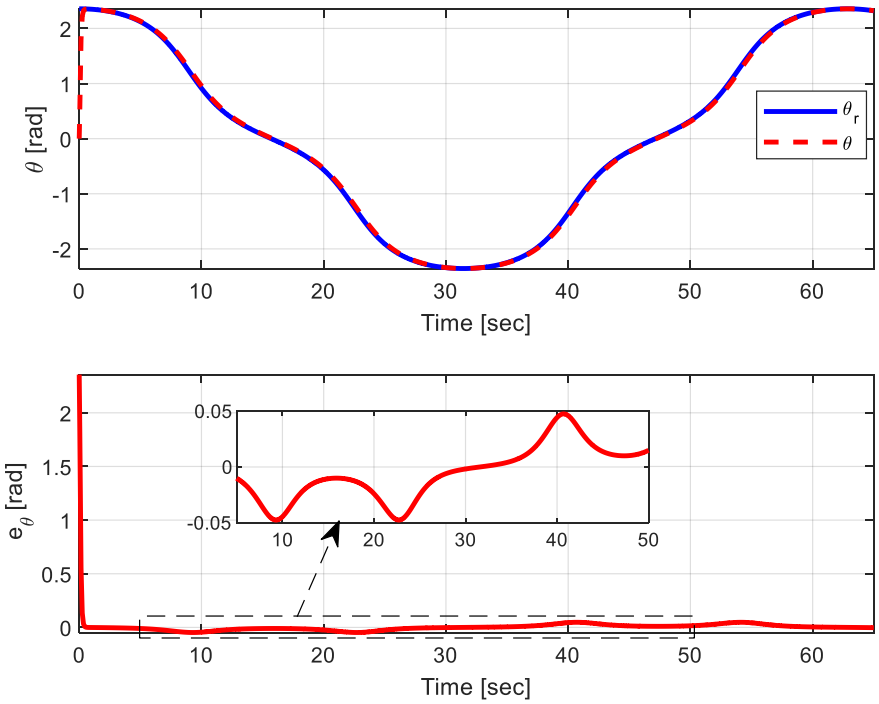

Figure 5.9: Angle tracking response and tracking error with the initial position [0, 0, 0] (nonlinear trajectory)

As shown in Figure 5.9, since the robot's heading angle depends on position changes in y-direction and x-direction, the reference steering angle at the start of motion is not zero. The robot's steering angle quickly converges to zero. Moreover, the result in Table 5.6 shows that a robot tracks a given trajectory with small RMS tracking error values.

Table 5.6: RMS values of the eight-shape trajectory tracking error with [0, 0, 0] initial state

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
RMS value	0.0002	0.0025	0.0799	0.0025

The robot smoothly tracks its reference trajectory in the x and y positions with the possible minimum tracking error when it starts its movement at the same initial position as the reference trajectory. From the trajectory tracking response shown in Figure 5.10 and the tracking error in Figure 5.11, the robot tracks an eight-shape trajectory with an error peak

value of 0.0111 m at a 0.024 sec peak time. According to the demonstrated results, the robot's trajectory tracking performance with a [0, 0, 0] initial state is preferable with a minimum tracking error.

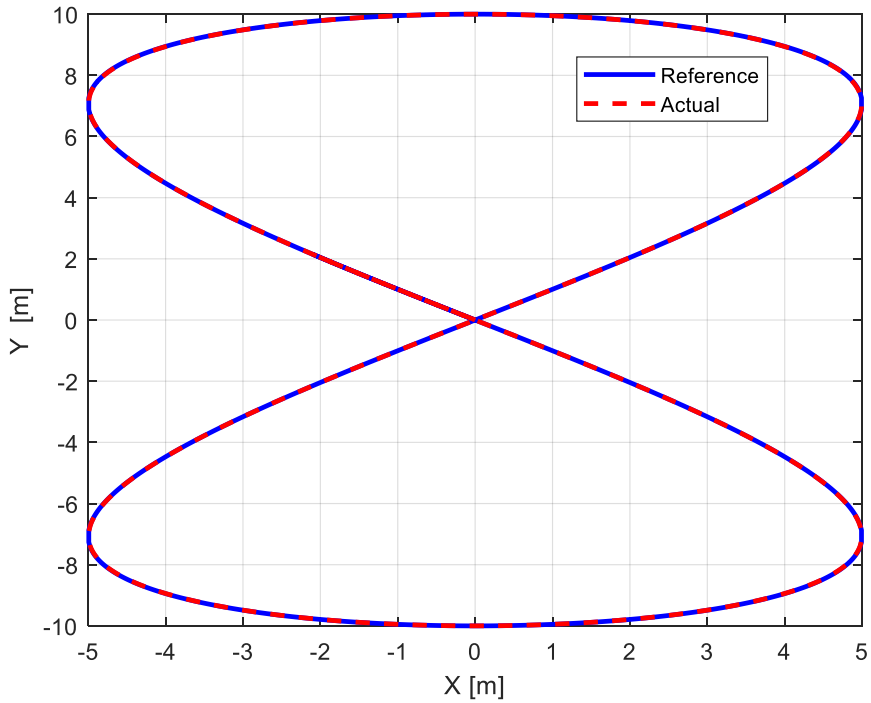

Figure 5.10: Eight-shape trajectory tracking response with initial position [0, 0, 0]

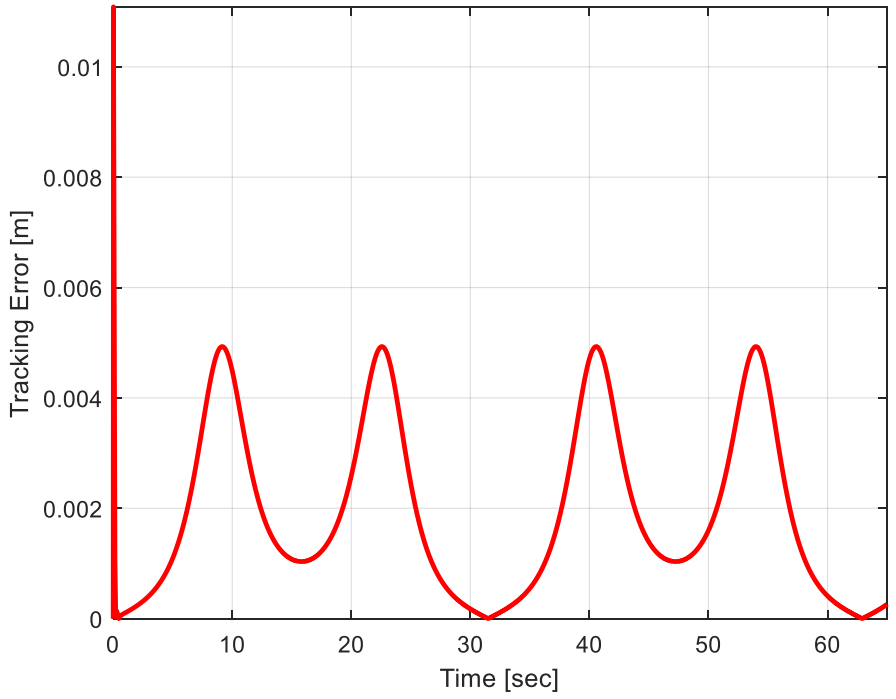

Figure 5.11: Eight-shape trajectory tracking error with the initial position [0,0,0]

In the trajectory tracking problem, if the initial position is changed, it imposes more overshoot on the system's transient response. If the change is much farther from the initial position of the reference trajectory, the system may become unable to track a given path, making it unstable and challenging to control in a reference position.

Trajectory tracking performance with [3, 2, pi/2] initial robot position

Figure 5.12 to Figure 5.16 show the trajectory tracking with the initial position [3, 2, pi/2]. The results show that the robot follows its reference trajectory with a possible minimum tracking error, as shown in Table 5.7.

Table 5.7: RMS values of trajectory tracking error with an initial state [3, 2, pi/2]

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
RMS value	0.0484	0.0428	0.0849	0.0646

The obtained results revealed that the x position, y position, and robot angle tracking errors converged in a short settling time regardless of the significant error at the start of a motion due to the initial position change. i.e., the robot needs more energy to track a target path at the beginning and then return to its position.

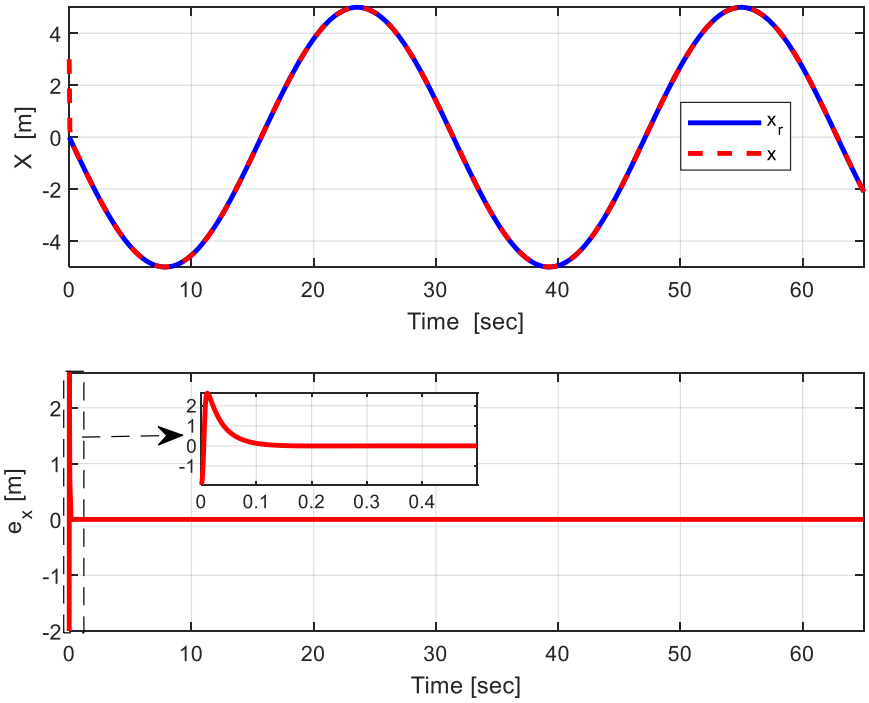

Figure 5.12: x position tracking and tracking error with initial position [3, 2, pi/2] (nonlinear)

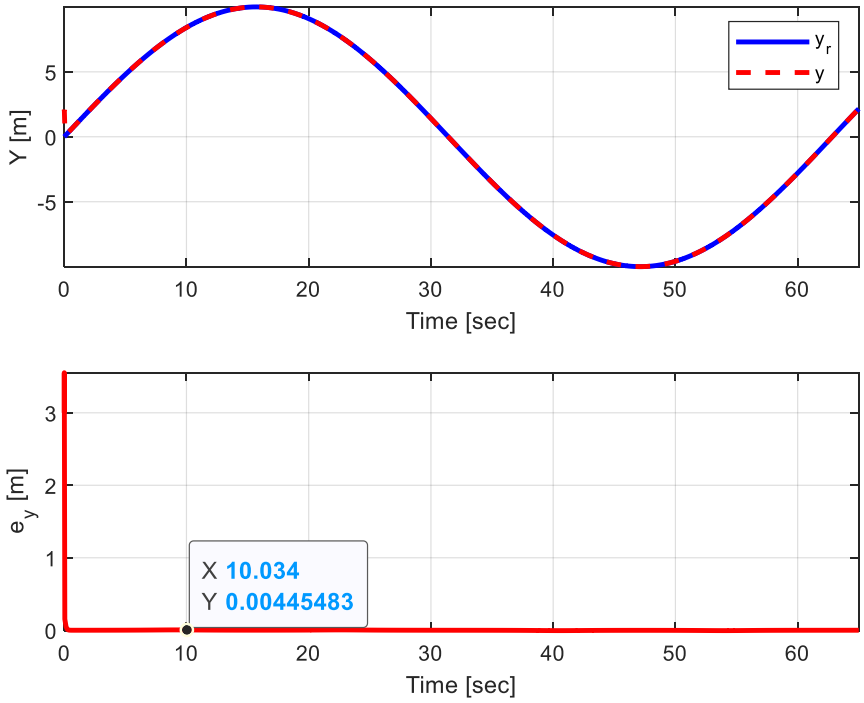

Figure 5.13: y position tracking and tracking error with initial position [3, 2, pi/2] (nonlinear)

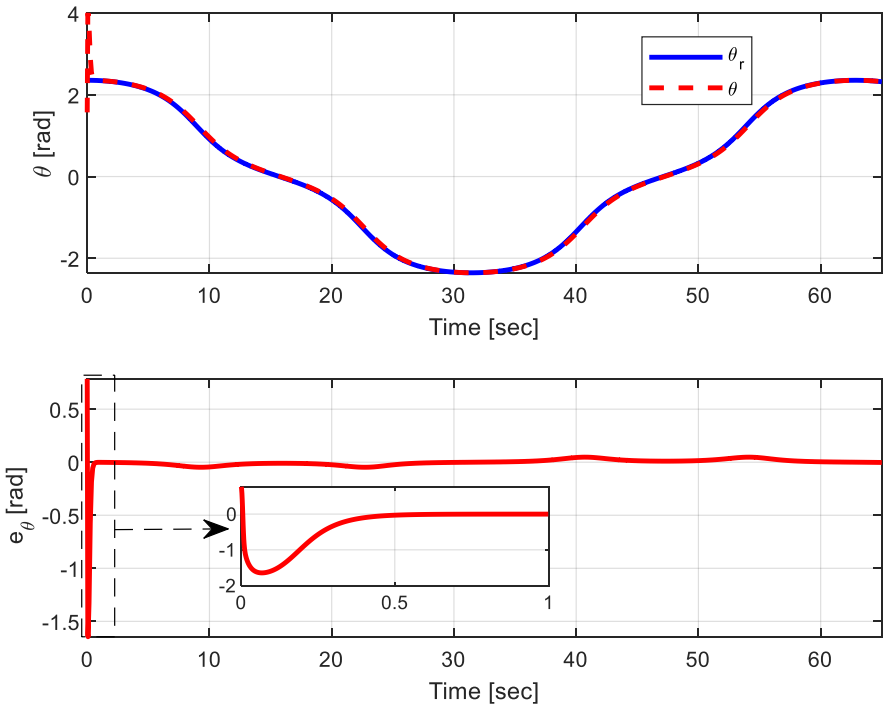

Figure 5.14: Angle tracking and tracking error with the initial position [3, 2, pi/2] (nonlinear)

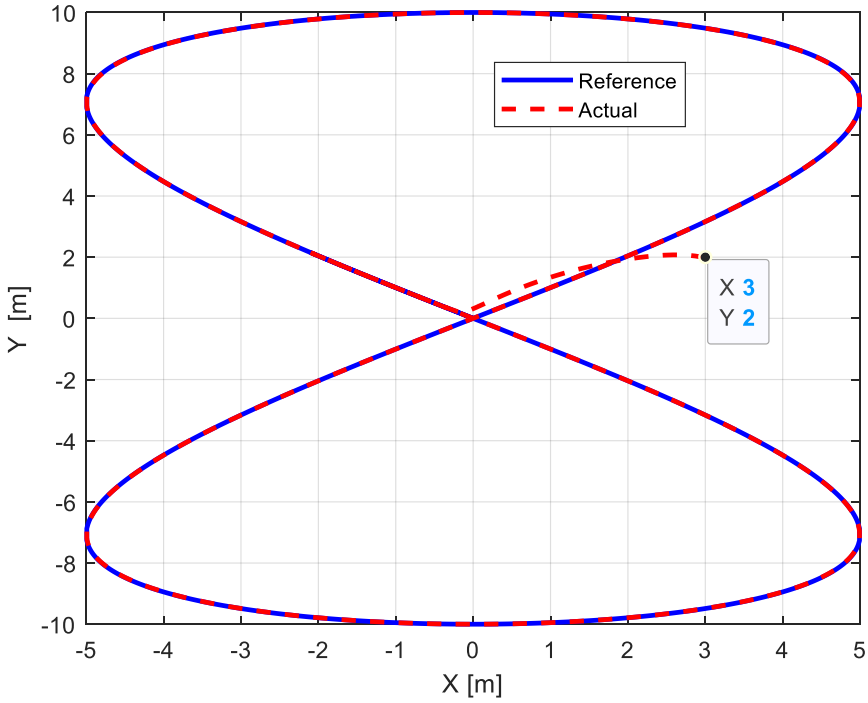

Figure 5.15: Eight-shape trajectory tracking response with the initial position [3, 2, pi/2]

In addition, the demonstrated results show that a robot tracks its reference trajectory with a minimum tracking error within an initial state change, as shown in Figure 5.16. The results show that the proposed controller is the fastest to respond to initial position changes.

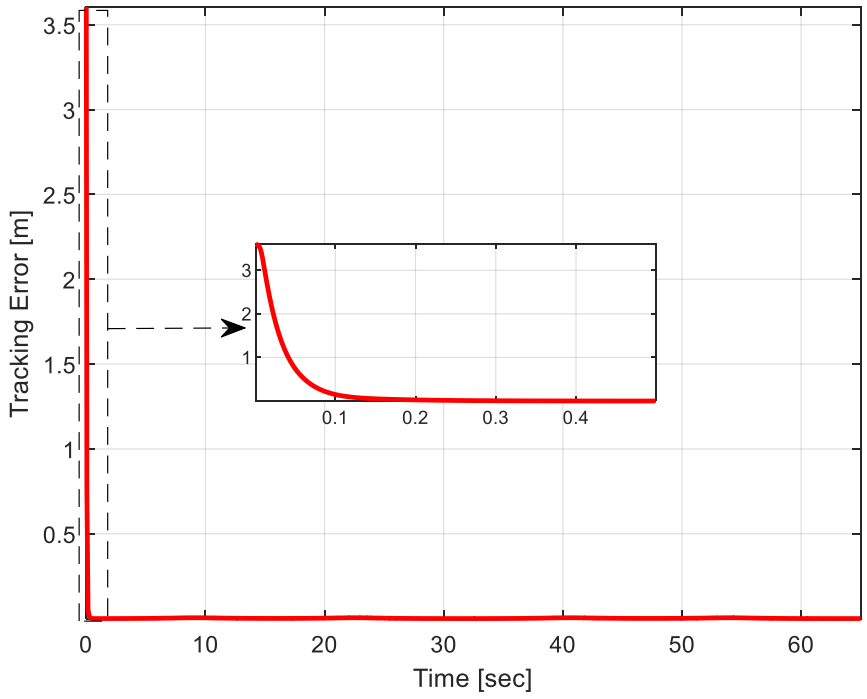

Figure 5.16: Robot trajectory tracking error with [3, 2, pi/2] initial state

Trajectory tracking performance with [0, -3, 2pi/3] initial robot position

Figure 5.17 depicts a trajectory tracking result of the robot with an initial state of [0, -3, 2pi/3]. In this case, the robot position in the x-direction is not changed, while the initial values of the y-direction and the robot angle are changed to -3 m and 120^0, respectively. The robot still tracks its reference trajectory with a minimum tracking error, as shown in Table 5.8.

Table 5.8: RMS values of tracking error with initial state [0, -3, 2pi/3]

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
RMS value	0.0586	0.0173	0.0467	0.0611

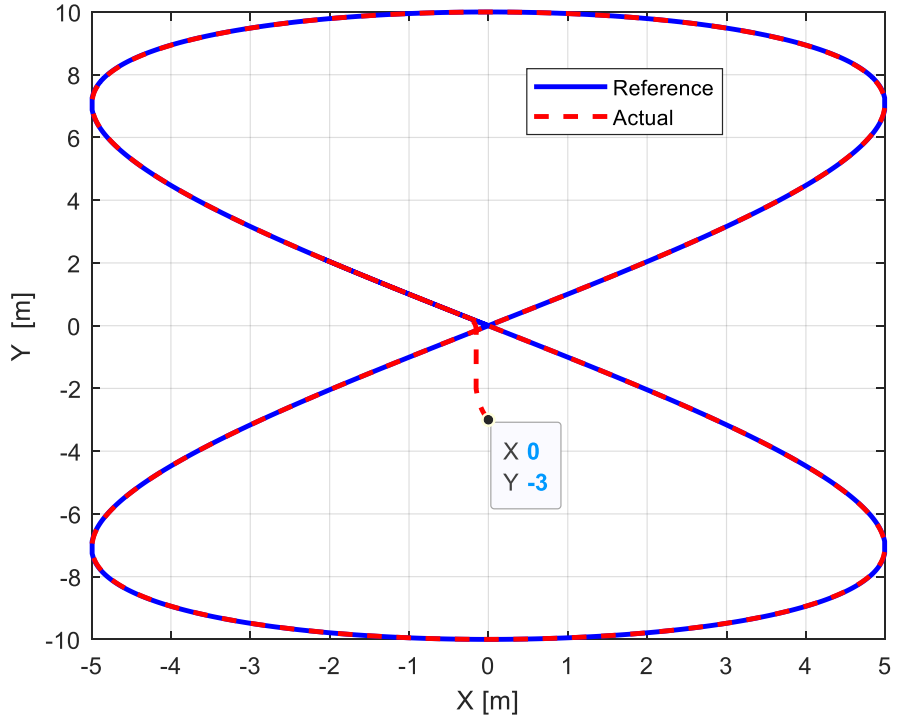

Figure 5.17: Eight-shape trajectory tracking response with the initial position [0,-2, 2pi/3]

From the three cases, i.e., for q_0 = [0, 0, 0], [3, 2, pi/2] and [0, -3, 2pi/3] of an initial position change, someone sees that the robot tracking performance that is achieved is smooth and preferable within a sinusoidal reference input. The proposed control method achieves a better tracking response regardless of the considerable peak value of tracking errors because the robot's initial position changes. All simulation results justify that the proposed method for trajectory tracking TWMR is preferable with minimum tracking error. In addition to this, a robot can handle an initial position change.

5.2.2. Velocity Tracking of TMWR using Backstepping and an NPID Controller

There are three controlled variables in the trajectory tracking control of TWMR that are controlled by two control inputs. Two-degrees-of-freedom (2DoF) velocity control inputs are used to control the 3DoF of controlled variables (x, y, and θ). A backstepping control method is helpful for mapping 2DoF control inputs to 3DoF controlled variables. If there are two loop control methods, the inner loop control method should be faster than the outer loop control method. Therefore, an NPID controller should respond quickly to eliminate velocity errors and achieve better tracking performance.

Figure 5.18 and Figure 5.19 show the controlled input velocity (solid blue line), auxiliary velocity control or controller output (dash-dot black line), and the actual velocity of a robot (dash-red line) tracking response results for a linear reference trajectory. A robot has only linear velocity in a linear trajectory because of an infinity turning point.

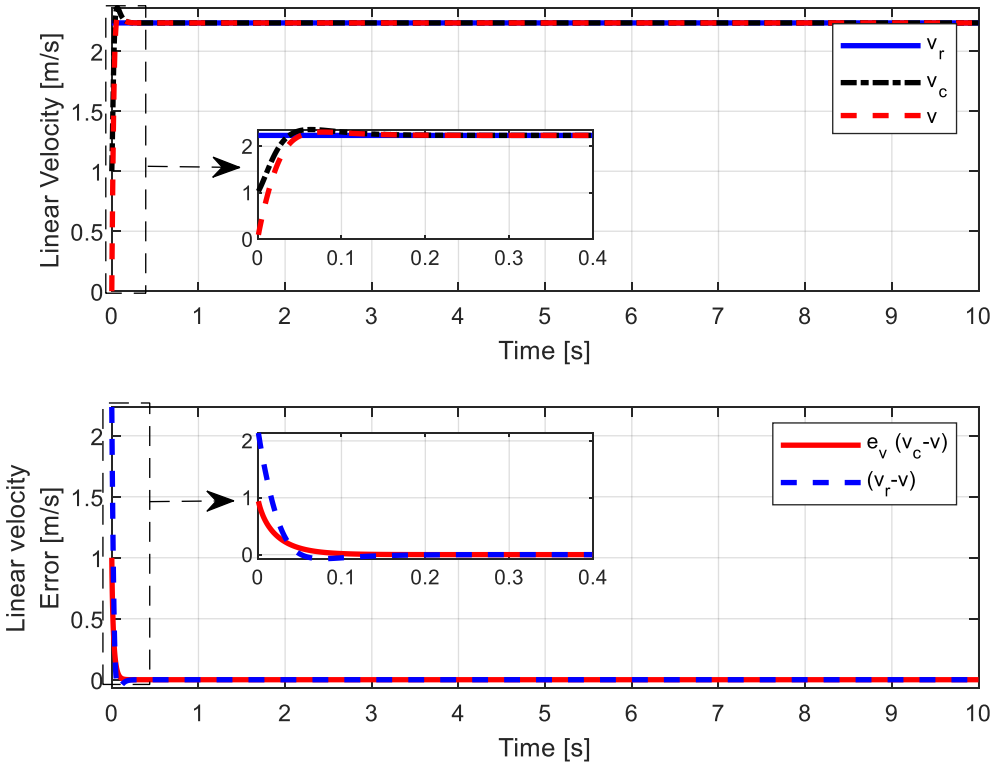

Figure 5.18: Linear velocity tracking and velocity error response (linear trajectory)

From Figure 5.18, the robot linear velocity tracking error between auxiliary velocity control and actual velocity (solid red line) has almost a 0.1 sec settling time. In addition, the robot's actual velocity reaches its reference values within a short settling time (blue dash line). As

shown in Figure 5.19, after 0.2 sec, the angular velocity of the robot converges to zero since this velocity is zero in linear trajectory tracking.

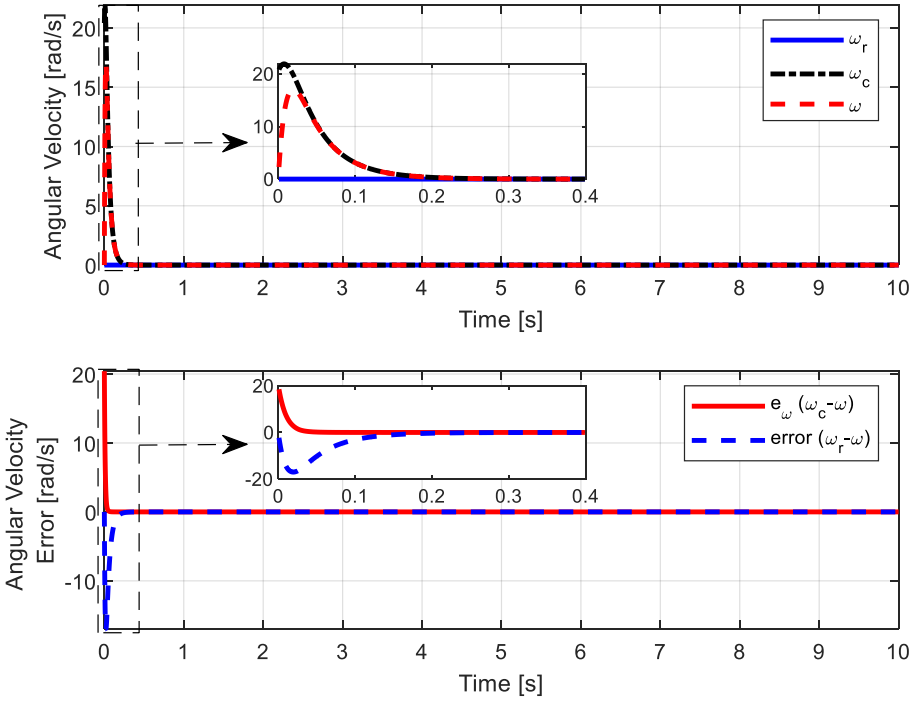

Figure 5.19: Angular velocity tracking and velocity error response (linear trajectory)

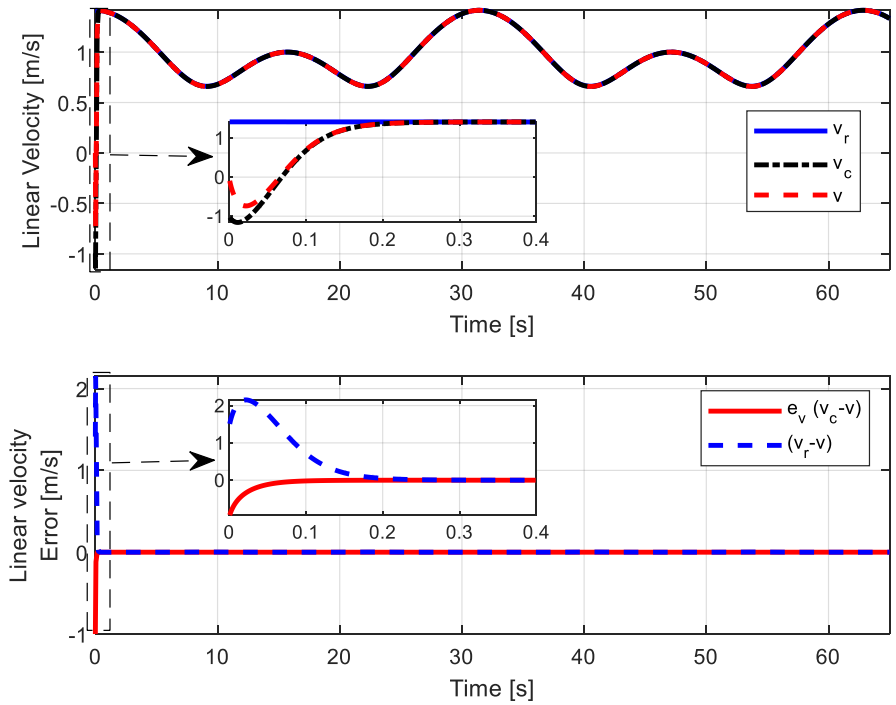

Figure 5.20: Linear velocity tracking and velocity error response (nonlinear trajectory)

Similarly, the robot's linear and angular velocity tracking of nonlinear reference trajectory is shown in Figure 5.20 and Figure 5.21 within its tracking errors. In this case, the actual robot velocity tracking to its reference control and auxiliary control velocity is quickly achieved.

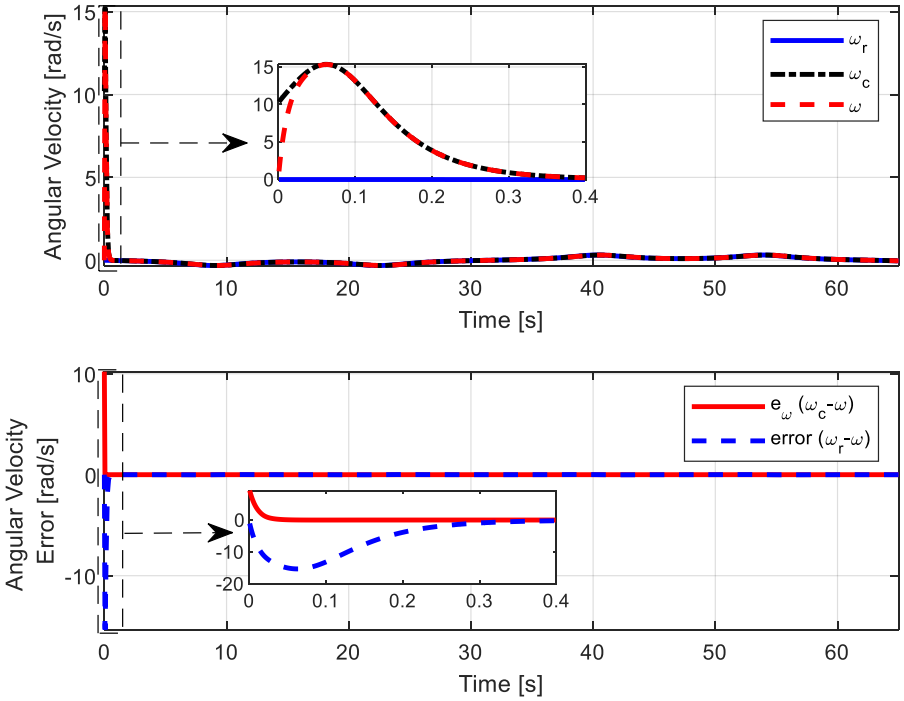

Figure 5.21: Angular velocity tracking and velocity error response (nonlinear trajectory)

Additionally, the angular velocity tracking of a robot, as shown in Figure 5.21, shows that a robot has about 0.4 sec settling time. The demonstration results show that system dynamics are stabilized and controlled with a minimum velocity error and a short settling time at the initial robot position of (0, 0, 0).

5.2.3. Trajectory Tracking Response with Unknown Disturbance

In the above demonstration, trajectory tracking control of TWMR with a linear and nonlinear trajectory is attained with minimum tracking error. A system model is considered free from external disturbances like the friction force between wheels and the ground. Also, friction occurs due to mechanical part rigidity and model uncertainty. System uncertainty arises when selecting plant parameters for a simple representation because we cannot know all the precise disturbance values. This directly impacts a system's performance in a real-time application. For this reason, the unknown disturbance torque is considered as an external disturbance that has occurred in the system to evaluate the proposed controller performance

with an unknown disturbance. The unknown and unmodeled disturbance torque in the form of $[\tau_{d1}\ \tau_{d2}]^T = [4\,sin(2t)\quad 4\,sin(2t)]^T$ is applied to both wheels of motor torque (which is 20% maximum motor torque). Figure 5.22 shows the unknown and unmodeled disturbance torque.

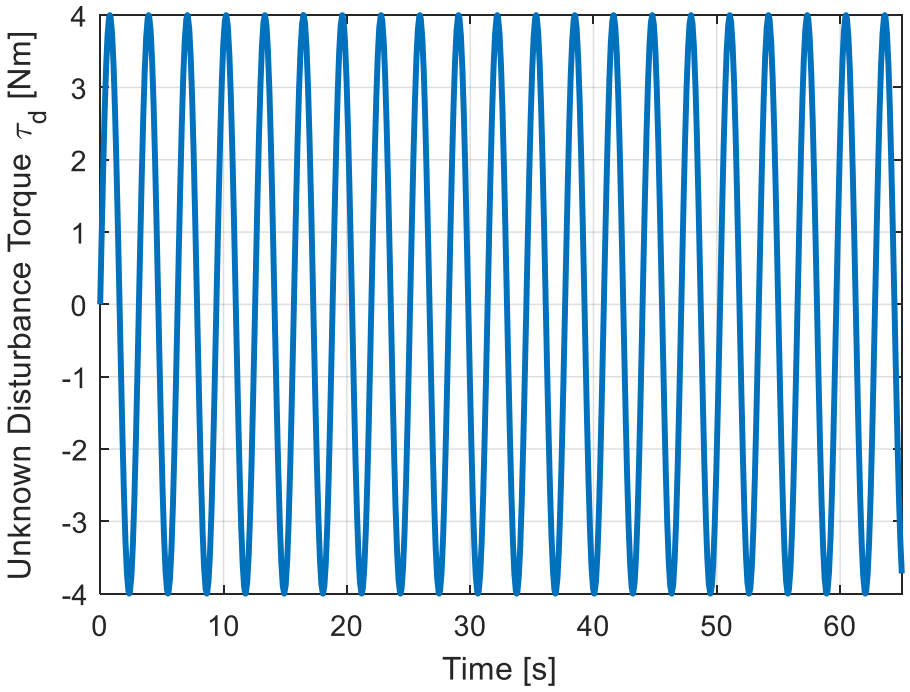

Figure 5.22: Unknown and unmodeled disturbance torque

5.2.3.1. Linear Trajectory Tracking Response with Unknown Disturbance

Figure 5.23 shows that in a position tracking in the x-direction, there is a change in the tracking error due to unknown disturbance torque. As in a zoomed-out tracking error, the unknown disturbance imposes an oscillatory response on a trajectory tracking response due to sinusoidal disturbance behavior while the reference trajectory is linear. The trajectory tracking response in the x-direction is still acceptable and reasonable regardless of a slight change in tracking error.

In addition, Figure 5.24 shows that an unknown disturbance in the system has almost zero impact on a position tracking response. The ability to reject an unknown disturbance of the proposed controller in the y position is almost perfect.

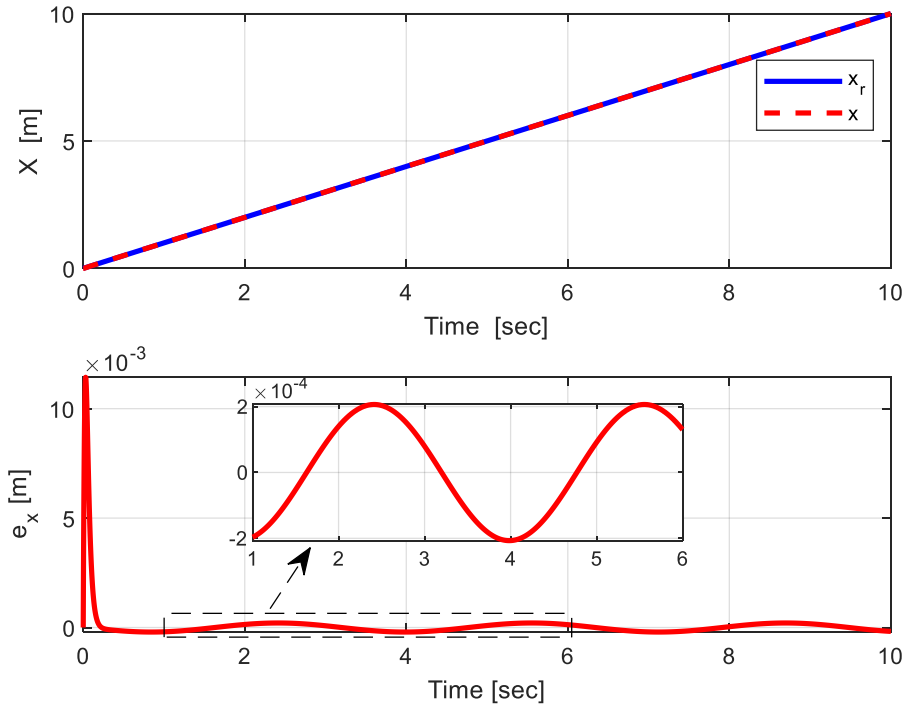

Figure 5.23: x-axis trajectory tracking with unmodeled disturbance (linear tracking)

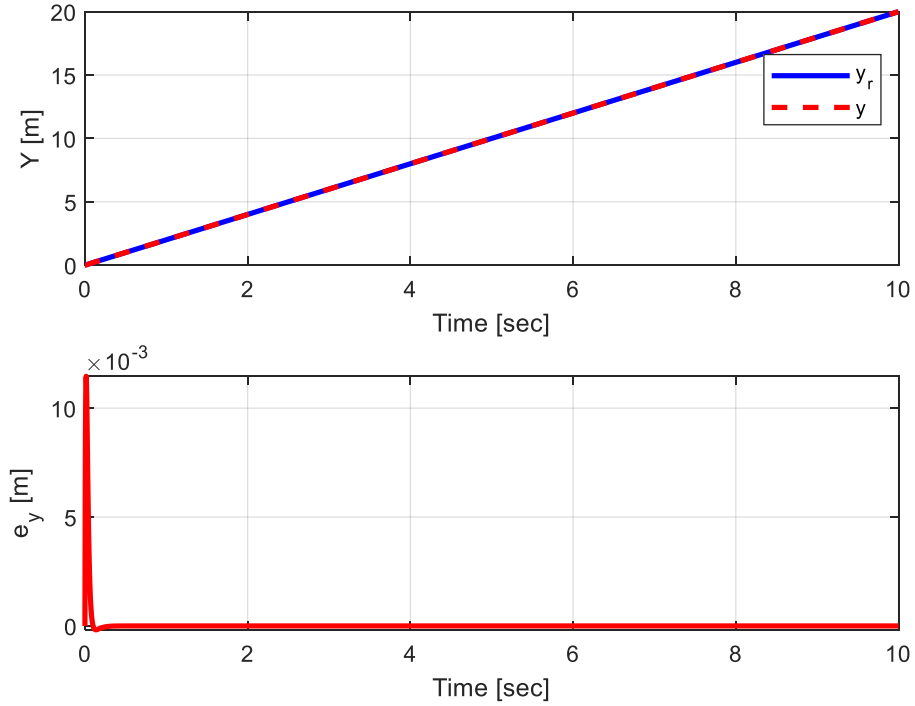

Figure 5.24: y-axis trajectory tracking with unmodelled disturbance (linear trajectory)

As shown in Figure 5.25, the robot angle tracking error is changed due to an unknown disturbance, as in a zoomed-out area. However, the effect of an unknown disturbance on the trajectory tracking performance of TWMR is minimal and almost negligible.

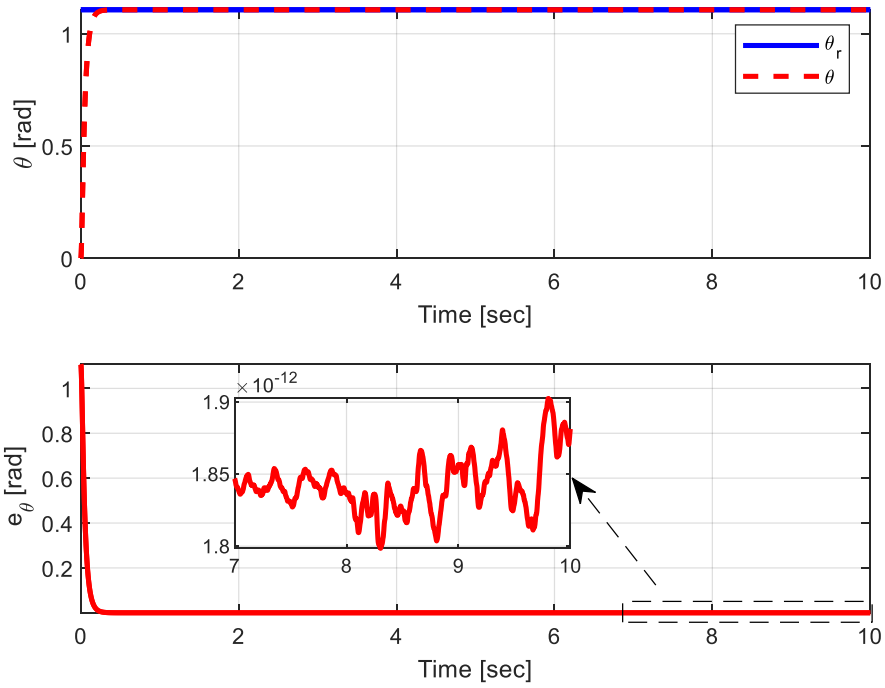

Figure 5.25: Robot angle tracking with unmodelled disturbance (Linear trajectory)

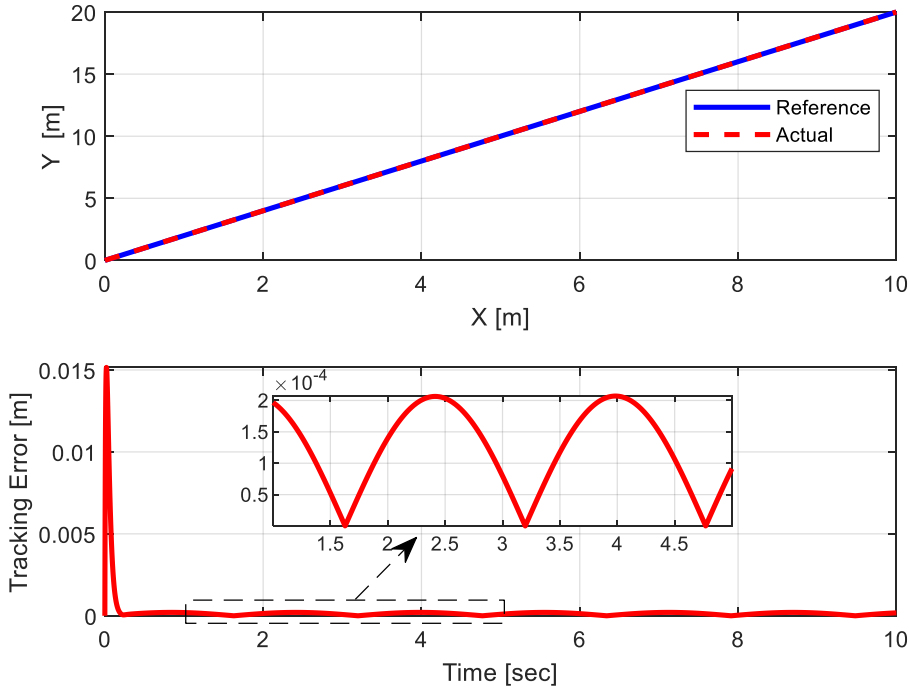

Figure 5.26: Linear trajectory and tracking error response with unmodelled disturbance

A zoomed-out area of position tracking error shows that a backstepping and NPID controller can reject an unknown disturbance effect, as shown in Figure 5.26. The proposed controller is robust to unknown disturbances and has better disturbance rejection capability.

5.2.3.2. Nonlinear Trajectory Tracking Response with Unknown Disturbance

The demonstrated response in Figure 5.27 to Figure 5.31 shows a nonlinear trajectory tracking response with unknown disturbance.

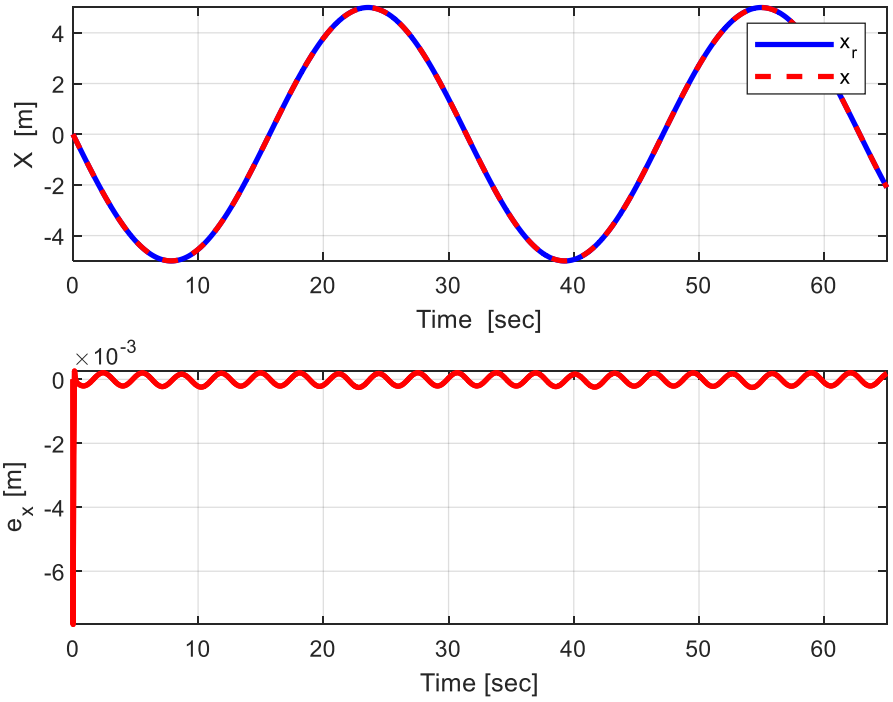

Figure 5.27: x-axis trajectory tracking with unmodeled disturbance (nonlinear trajectory)

In Figure 5.27, a change in trajectory tracking error is observed due to an unknown disturbance. The proposed controller has regained its ability to compensate for the disturbance rejection. In the presence of an unknown disturbance, a robot reasonably follows a reference trajectory.

As shown in Figure 5.28 and Figure 5.29, no tracking error change was observed in the y-axis or robot angle tracking response. The controller has better disturbance rejection in y-axis position tracking and steering angle tracking within an unknown disturbance.

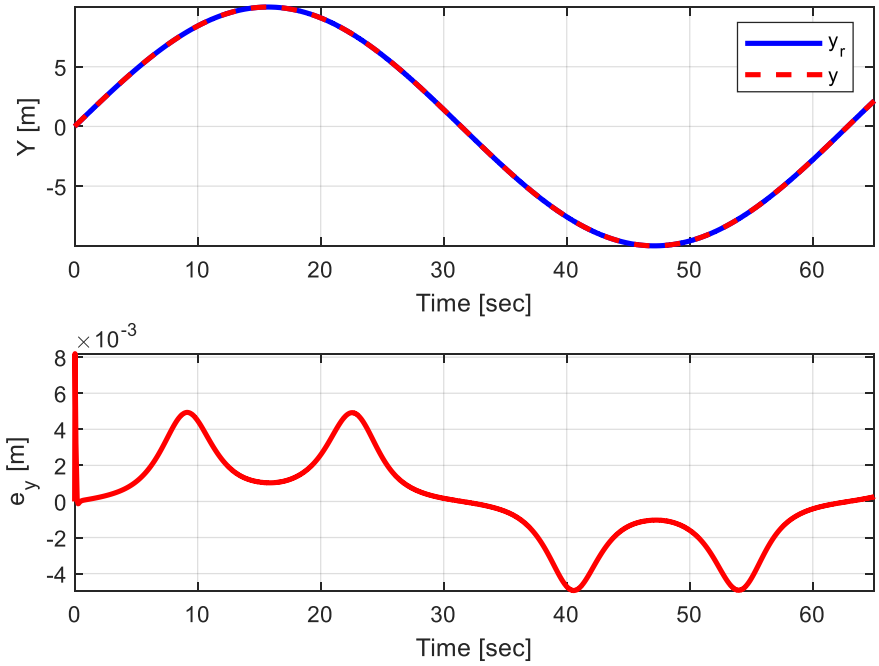

Figure 5.28: y-axis trajectory tracking with unmodeled disturbance (nonlinear trajectory)

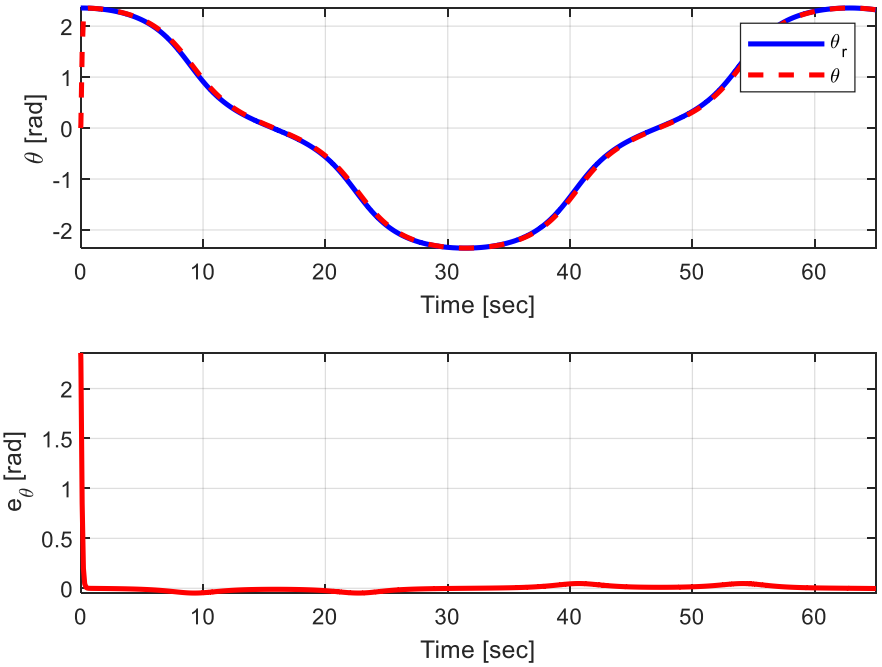

Figure 5.29: Robot angle tracking with unmodeled disturbance (nonlinear trajectory)

A nonlinear reference trajectory response with unknown disturbance is shown in Figure 5.30 and trajectory tracking error response in Figure 5.31. From Figure 5.31 it shows that the unknown disturbance imposes a change on the trajectory tracking of the robot.

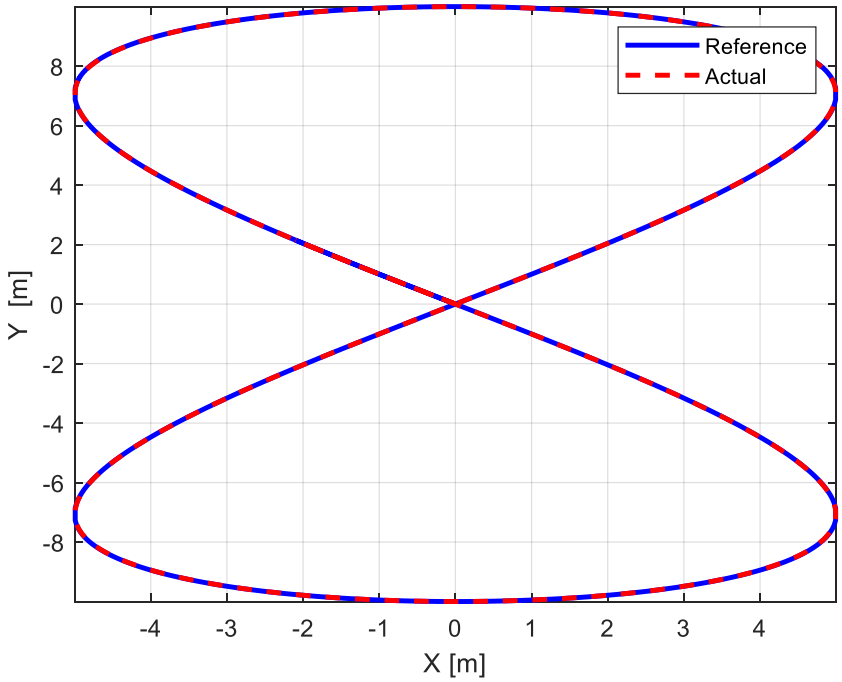

Figure 5.30: Eight-shape trajectory tracking with unmodeled disturbance

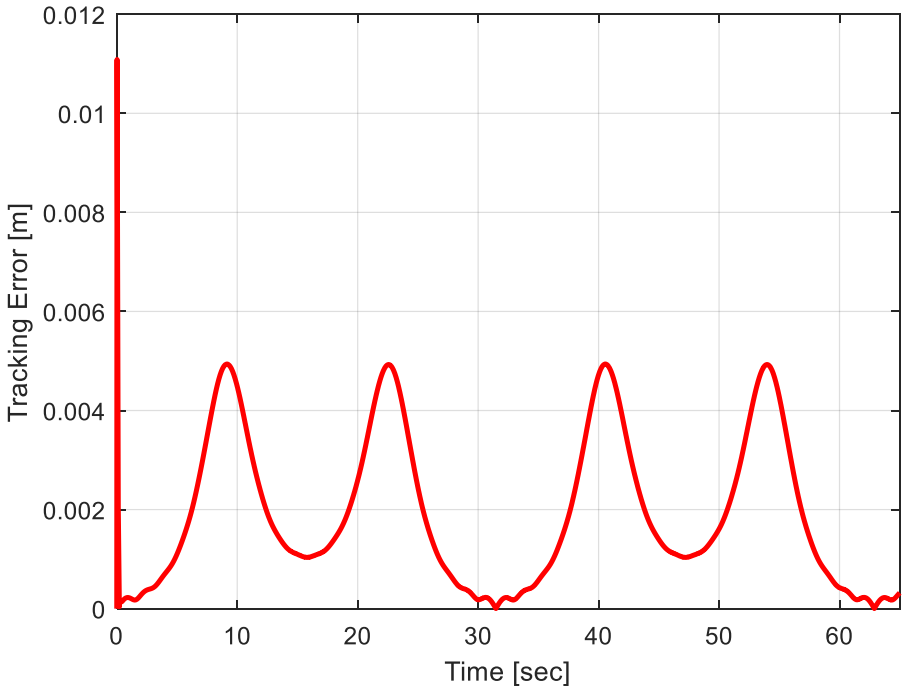

Figure 5.31: Eight-shape trajectory tracking error with unknown disturbance

To summarize the obtained result, it is clear that the proposed system has good unknown disturbance rejection capability in all scenarios. One of the advantages of the backstepping controller is that it has good disturbance rejection for low-level disturbances. The obtained

results also validated this. Even if the system has good low-level unknown disturbance rejection, it can only handle up to $\pm$ 10 sin $(2t)$ Nm, which is 50% of the maximum motor input torque.

5.3. Discussions and Comparative Analysis of the Proposed Method

The previous section discusses the proposed controller performance, tracking ability, and adaptation to initial position change with backstepping and the NPID (BSC+NPID) controller. In this section, the trajectory tracking performance of a proposed controller has been compared with that of a GA-based backstepping controller (BSC) and a GA-based backstepping plus PID (BSC+PID) controller.

To know if a particular controller performs better than other control methods for the same controlled plant, the reference tracking performance, model uncertainty and disturbance rejection ability, the transient and steady-state behavior of the response, the control signal needed to overcome an error, and the energy required should be considered. In addition, a control method's adaptability and robustness are significant concerns. For this work, the reference trajectory tracking performance and RMS values of tracking error are considered for comparative analysis.

5.3.1. Proposed Method Comparison with GA-based Backstepping Controller

This section compares the trajectory tracking response of the GA-based backstepping plus NPID controller with that of a backstepping controller, and GA optimizes the controller gains of a backstepping controller. Table 5.9 shows GA-based backstepping controller gains optimized by the GA optimization method.

Table 5.9: GA-based backstepping controller gain parameters

GA-based backstepping	K_x	K_y	K_θ	k_1	k_2	ITAE
controller gains	9.7540	79.0517	24.5095	35.1660	87.0829	0.1125

Figure 5.32 shows the linear reference trajectory tracking response and error achieved by a GA-based backstepping controller (red dash-line), the proposed controller (black dash-dot line), and the reference trajectory (solid blue line). From the tracking error response, a GA-based backstepping controller has a peak value of 0.0341 m at 0.06 sec of peak time, while a proposed controller has a peak value of 0.0152 m at 0.02 sec of peak time tracking error.

In addition, the BSC+NPID controller settles at 0.2 sec of settling time, while the backstepping controller needs almost four times that of the BSC+NPID controller (about 0.7349 sec). The proposed controller reduces the peak value by 55.43%, and the settling time will be 72.79%.

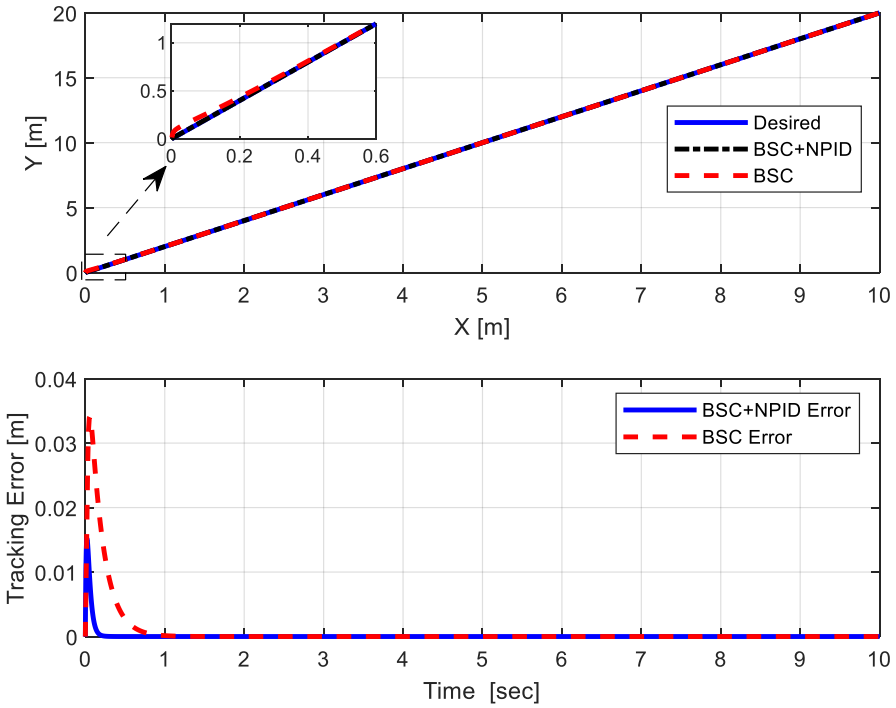

Figure 5.32: Linear trajectory tracking using a GA-based BSC and a BSC+NPID controller

Numerically, for linear trajectory tracking with [0, 0, 0] initial position, the proposed controller performs better in trajectory tracking than a backstepping controller, as shown in Table 5.10.

Table 5.10: Tracking errors of the backstepping controller and proposed controller (linear)

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0008	0.0006	0.0613	0.0010
GA-based backstepping controller	0.0009	0.0038	0.0564	0.0039
Tracking error reduced in percentage	11.11 %	84.21 %	8 % more	74.36 %

An obtained result shows that mobile robot position tracking using a proposed controller improved the tracking error compared to the backstepping controller in the x and y position tracking. However, better robot angle tracking is achieved using a backstepping controller.

In the case of nonlinear trajectory, the tracking results shown in Figure 5.33 verify that a backstepping and NPID controller tracking response is preferable to a GA-based backstepping controller. As shown in Figure 5.34, the GA-based backstepping controller of tracking error has a peak value of 0.0634 m at 0.09 sec, while the BSC+NPID controller has a peak value of 0.0111 m of position tracking at 0.024 sec of peak time. The proposed controller reduces the peak value of tracking error by 82.49%, and the peak time is 73.33%.

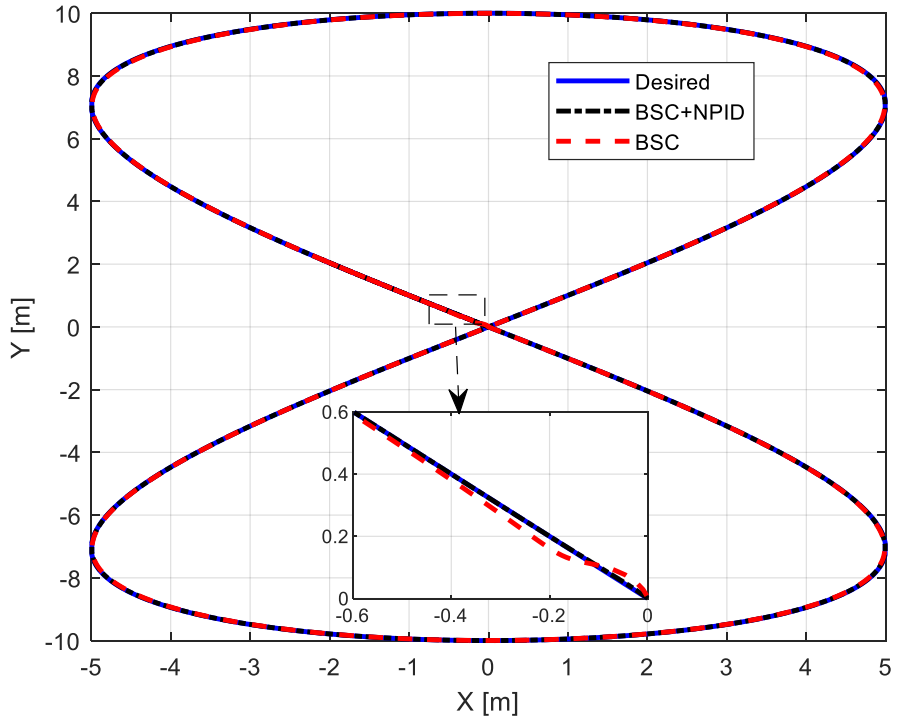

Figure 5.33: Nonlinear trajectory tracking with GA-based BSC and BSC+NPID controller

Statistically, a BSC+NPID controller has better tracking performance in the x and y positions than a GA-based backstepping controller. However, as shown in Table 5.11, better reference robot angle tracking is achieved by a backstepping controller.

Table 5.11: Eight-shape tracking errors of backstepping-only and BSC+NPID controller

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0002	0.0025	0.0799	0.0025
GA-based backstepping controller	0.0021	0.0075	0.0648	0.0078
Tracking error reduced in percentage	90.48 %	66.67 %	18.9 % more	67.95 %

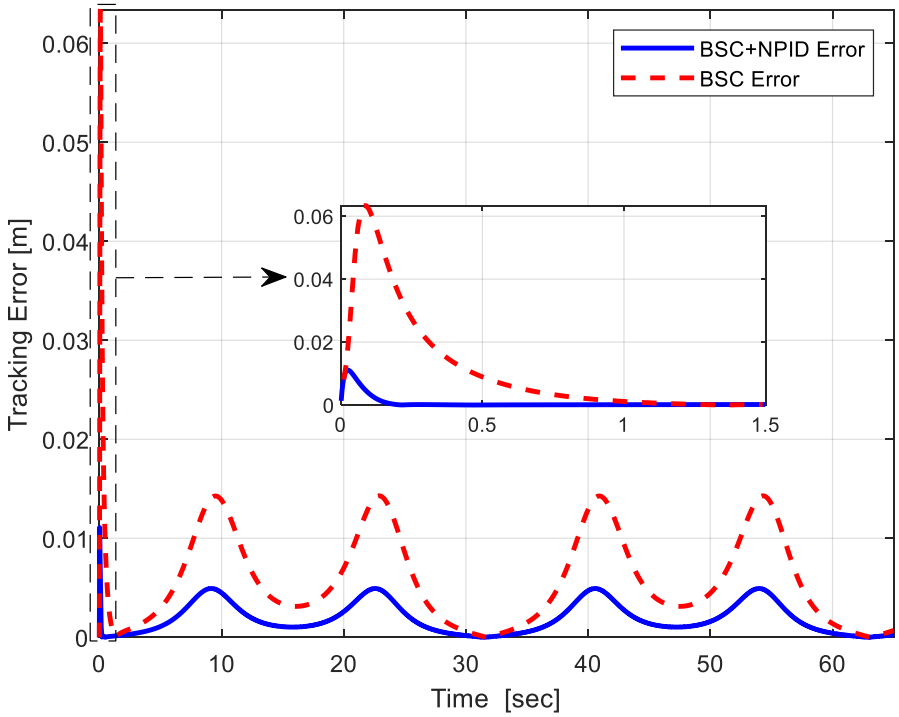

Figure 5.34: A GA-based BSC and BSC+NPID controller tracking error response

An illustrated result demonstrates that the BSC+NPID controller has better position tracking than the GA-based backstepping controller. In contrast, the GA-based backstepping controller has better robot angle tracking with [0, 0, 0] initial robot position. If the initial position changes, the proposed controller has better tracking capability in all positions.

Table 5.12: RMS values of tracking errors using the BSC controller and the BSC+NPID controller with [1, 2, pi/2]

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0328	0.0264	0.1048	0.0422
GA-based backstepping controller	0.0552	0.0391	0.1126	0.0677
Tracking error reduced in percentage	40.57 %	32.48%	6.93 %	37.67 %

An illustrated result demonstrates that the BSC+NPID controller has better position tracking than the GA-based backstepping controller. In contrast, the GA-based backstepping controller has better robot angle tracking with [0, 0, 0] initial robot position. If the initial position changes, the proposed controller has better tracking capability in all positions.

Table 5.12 shows RMS trajectory tracking errors by the GA-based BSC controller and BSC+NPID controller with the robot's initial position [1, 2, pi/2]. According to the results, backstepping and the NPID controller's trajectory tracking in the x position, y position, and steering angle tracking have less tracking error than the GA-based BSC controller. The proposed controller outperformed the GA-based BSC controller in trajectory tracking response.

5.3.2. Proposed Method Comparison with GA-based Backstepping plus PID Controller

Figure 5.35 shows the linear trajectory tracking response using a backstepping plus PID controller and a proposed controller with a tracking error. The results show that the proposed controller has a better tracking error with a settling time of 0.2 sec, while the BSC+PID controller has almost three-folded the proposed controller (0.6 sec). A BSC+PID controller also had a peak tracking error value of 0.0168 m at 0.057 sec of peak time. A proposed method improved settling time by 66.67% and the position error peak value by 9.52%.

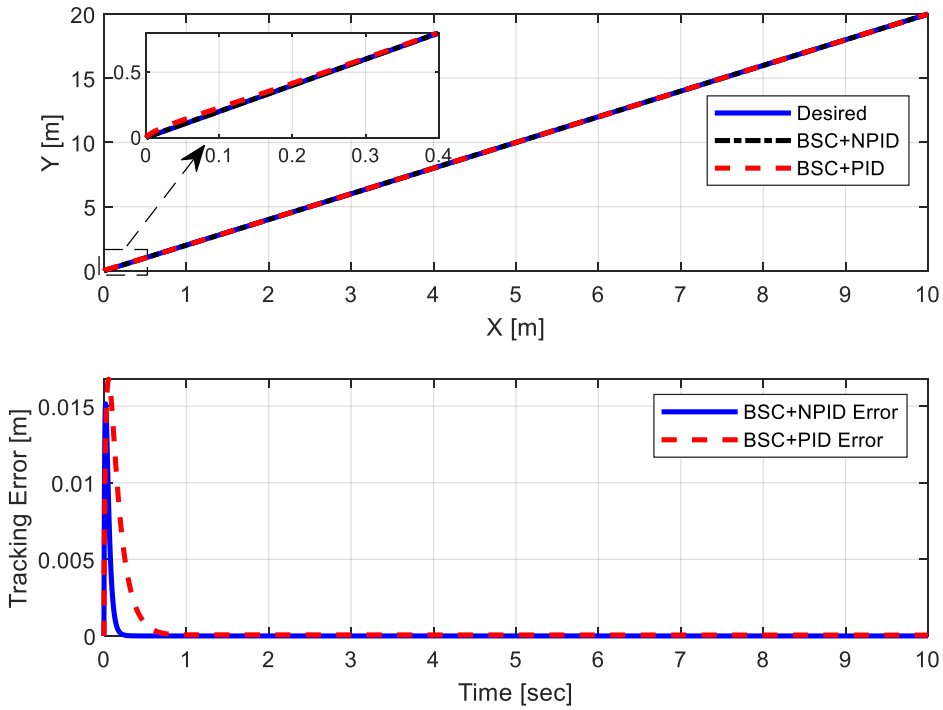

Figure 5.35: Linear trajectory tracking and tracking error response using the proposed controller and GA-based BSC+PID controller

The tracking performance shown in Table 5.13 illustrates that the proposed controller had a better tracking response in both the x and the y-direction. The GA-based BSC+PID controller also had better robot angle tracking when a robot begins its motion at the [0, 0, 0] position.

Table 5.13: An RMS value tracking errors using BSC+NPID and BSC+PID controller

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0008	0.0006	0.0613	0.0010
GA-based BSC and PID controller	0.0012	0.0016	0.0564	0.0019
Tracking error reduced in percentage	33.33 %	62.5 %	8 % more	47.36 %

Similarly, the nonlinear trajectory tracking response results with a proposed controller and GA-based BSC+PID control are shown in Figure 5.36, and the tracking error is in Figure 5.37.

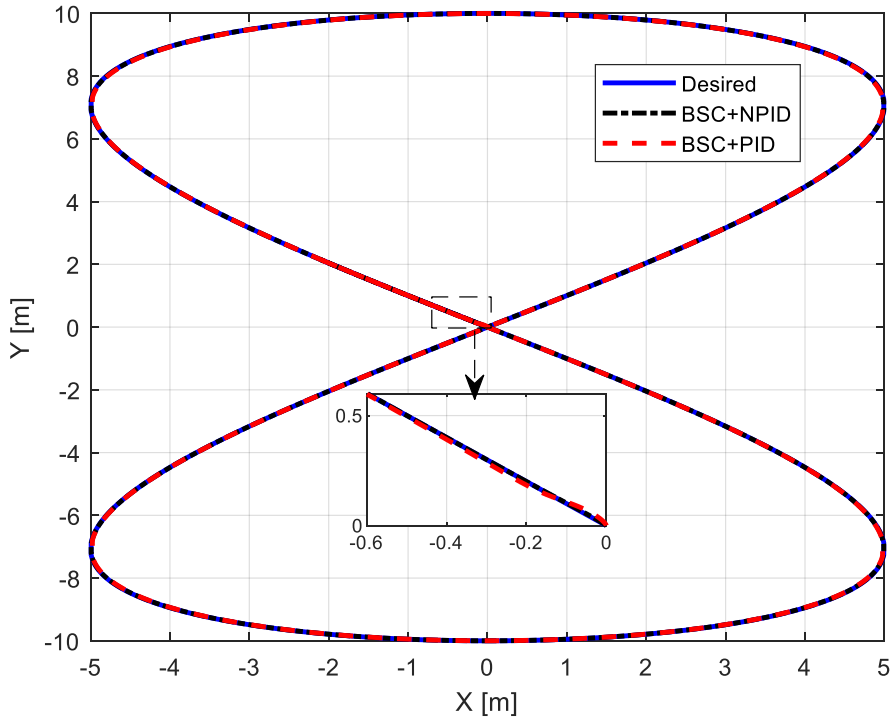

Figure 5.36: Nonlinear trajectory tracking using the proposed controller and GA-based BSC+PID controller

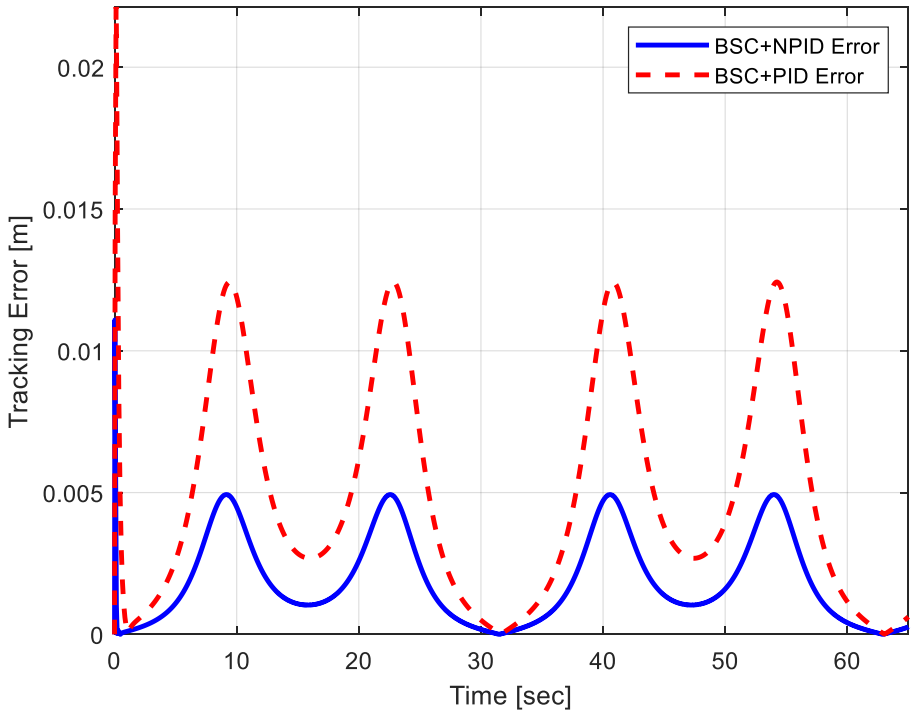

Figure 5.37: A proposed controller and a GA-based BSC+PID controller tracking error

A GA-based backstepping combined with a PID controller's tracking response is almost as good as backstepping combined with an NPID controller. A tracking error response shown in Figure 5.37 shows that a GA-based BSC+PID controller has a peak value of about 0.0221m at a 0.13 sec peak time. The proposed controller reduces the peak value of tracking error by 49.77%. The BSC+PID controller has better robot angle tracking than a GA-based backstepping controller. In contrast, the proposed controller has better tracking performance in the x and y position tracking, as shown in Table 5.14.

Table 5.14: RMS values of tracking error using the proposed controller and Ga-based BSC+PID controller with [0,0,0] initial position changes (nonlinear trajectory)

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0002	0.00025	0.0799	0.0025
GA-based BSC and PID controller	0.0009	0.0063	0.0719	0.0063
Tracking error reduced in percentage	77.78 %	60.32 %	10.01 % more	60.32 %

Figure 5.38 and Figure 5.39 show the trajectory tracking response of BSC+PID compared with a proposed controller with an initial robot position of [3,2, pi/2].

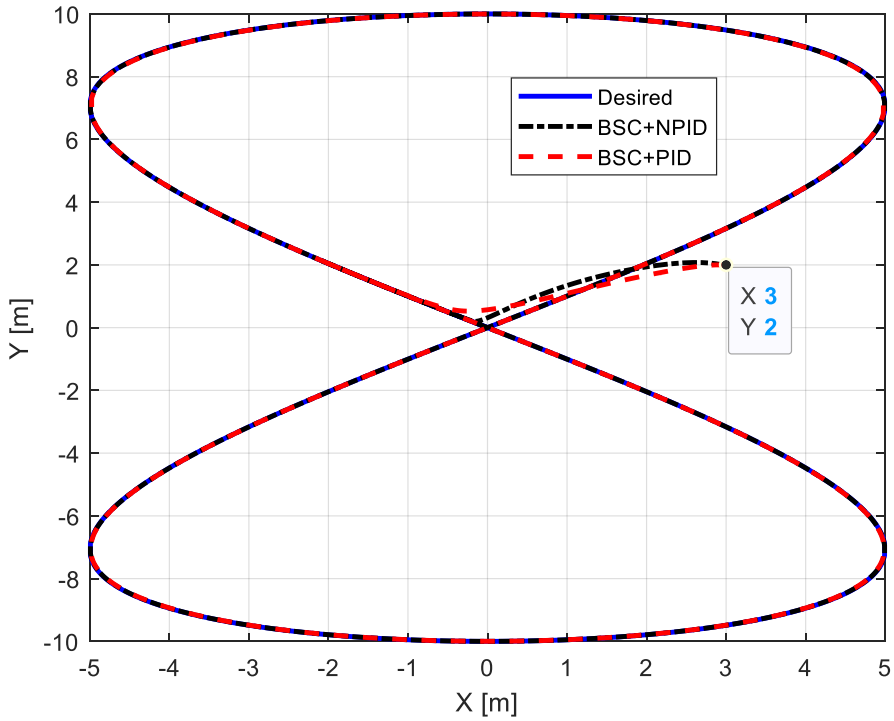

Figure 5.38: BSC+PID and BSC+NPID trajectory tracking with initial position changes

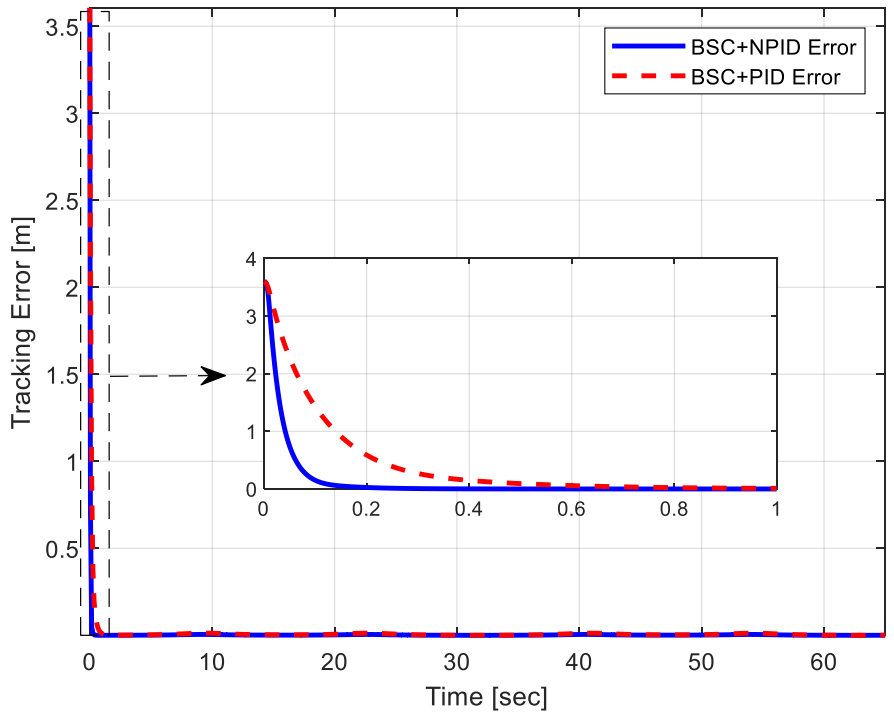

Figure 5.39: A GA-based BSC+PID and BSC+NPID controller trajectory tracking error

From Figure 5.39, a trajectory tracking error by the BSC+PID controller has a settling time of 0.5537 sec, while a proposed controller has only 0.1305 sec. A proposed controller

improves the settling time better than a BSC+PID controller by 76.43%. The numerical results in Table 5.15 also show that a BSC+NPID controller improved the RMS value of tracking errors.

Table 5.15: RMS values of tracking error using BSC+NPID and BSC+PID controller with initial position changes (nonlinear trajectory)

Tracking errors	e_x (m)	e_y (m)	e_θ (rad)	e_{pose} (m)
GA-based BSC and NPID controller	0.0484	0.0428	0.0849	0.0646
GA-based BSC and PID controller	0.0939	0.0512	0.0974	0.1069
Tracking error reduced in percentage	48.45 %	16.4 %	12.83 %	39.57 %

Generally, the trajectory tracking of the TWMR was achieved using GA-based backstepping, GA-based BSC+PID, and GA-based BSC+NPID controllers, as demonstrated. The results obtained by a proposed controller improve the trajectory tracking response's transient behavior and the tracking error's RMS value within a specified time. Consequently, the trajectory tracking of a TMWR is controlled and stabilized with an initial position change and in the presence of an unknown disturbance with a minimum trajectory tracking error.

6. CONCLUSION AND RECOMMENDATION

6.1. Conclusion

This thesis work introduced a backstepping controller combined with a nonlinear PID controller in the trajectory tracking control of a TWMR. The kinematic and dynamic modeling of a TWMR were formulated. The dynamic system models of the robot were derived using Lagrangian approaches. A backstepping plus NPID controller is designed to control a robot's trajectory tracking. The stability of a proposed controller is achieved using the Lyapunov stability analysis function.

The proposed system model is simulated on Matlab/Simulink (R2021a) software. The controller gain parameters of a backstepping plus NPID controller are tuned with the help of the GA optimization method. Also, the controller gains of the backstepping-only controller and the backstepping plus PID controller were optimized for comparison purposes.

As a demonstrated simulation result, the proposed controller achieved better reference trajectory tracking with a minimum tracking error in both scenarios. The robot follows its reference trajectory quickly if its initial position is changed. It also has better unknown disturbance rejection.

The proposed controller performance was compared with a GA-based BSC controller and a GA-based BSC+PID controller. The obtained results with [0, 0, 0] initial position show that the proposed controller reference trajectory tracking errors are more accurate than the GA-based BSC+PID controller (i.e., linear trajectory tracking error is improved by 47.36%, and nonlinear trajectory tracking error is improved by 60.23%). In addition to this, the proposed controller has better tracking performance when compared with a GA-based BSC controller (which means 74.36% in the linear case and 67.95% in the nonlinear case).

Therefore, the proposed controller is robust to initial position change and unknown disturbances. It improved the trajectory tracking response of a robot by reducing the trajectory tracking error. Thus, the trajectory tracking control of TWMR achieved using the backstepping controller combined with an NPID controller optimized by GA optimization was smooth, robust, and preferable in all scenarios.

6.2. Recommendation

This thesis work is limited to system model simulation in Matlab/Simulink software. In order to investigate the performance and capability of a proposed controller, it should be implemented in real-time. For future researchers, we recommend the following:

- Hardware implementation,
- In the dynamic system modeling, the presence of wheel slipping and skidding phenomena should be considered.
- The control law in this work did not update its parameters if the interest region (which means the desired trajectory) was changed. So, for better tracking capability for different environments, a control law based on adaptive control mechanisms and a self-taught controller is preferable and recommendable.

Acknowledgments: This research thesis was funded by Adama Science and Technology University under the grant number of ASTU/SM-R/533/22, Adama, Ethiopia.

REFERENCES

Ahmad Abu Hatab, R. D. (2013). Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Advances in Robotics & Automation*, *02*(02). https://doi.org/10.4172/2168-9695.1000107

Al-Mayyahi, A., Wang, W., & Birch, P. (2016). Design of Fractional-Order Controller for Trajectory Tracking Control of a Non-holonomic Autonomous Ground Vehicle. *Journal of Control, Automation and Electrical Systems*, *27*(1), 29–42. https://doi.org/10.1007/s40313-015-0214-2

Alam, T., Qamar, S., Dixit, A., & Benaida, M. (2021). Genetic algorithm: Reviews, implementations and applications. In *International Journal of Engineering Pedagogy* (Vol. 10, Issue 6, pp. 57–77). https://doi.org/10.3991/IJEP.V10I6.14567

Ben Jabeur, C., & Seddik, H. (2021). Design of a PID optimized neural networks and PD fuzzy logic controllers for a two-wheeled mobile robot. *Asian Journal of Control*, *23*(1), 23–41. https://doi.org/10.1002/asjc.2356

Benchouche, W., Mellah, R., & Bennouna, M. S. (2021). The Impact of the Dynamic Model in Feedback Linearization Trajectory Tracking of a Mobile Robot. *Periodica Polytechnica Electrical Engineering and Computer Science*, *65*(4), 329–343. https://doi.org/10.3311/PPee.17127

Cen, H., & Singh, B. K. (2021). Nonholonomic Wheeled Mobile Robot Trajectory Tracking Control Based on Improved Sliding Mode Variable Structure. *Wireless Communications and Mobile Computing*, *2021*, 1–9. https://doi.org/10.1155/2021/2974839

Chang, H., & Jin, T. (2013). Adaptive Tracking Controller Based on the PID for Mobile Robot Path Tracking. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 8102 LNAI* (Issue PART 1, pp. 540–549). https://doi.org/10.1007/978-3-642-40852-6_55

Coelho, P., & Nunes, U. (2003). Lie algebra application to mobile robot control: a tutorial. *Robotica*, *21*(5), S0263574703005149. https://doi.org/10.1017/S0263574703005149

Dong, J. F., Sabastian, S. E., Lim, T. M., & Li, Y. P. (2014). Autonomous Indoor Vehicles. In *Handbook of Manufacturing Engineering and Technology* (pp. 1–42). Springer London. https://doi.org/10.1007/978-1-4471-4976-7_103-1

Esmaeili, N., Alfi, A., & Khosravi, H. (2017). Balancing and Trajectory Tracking of Two-Wheeled Mobile Robot Using Backstepping Sliding Mode Control: Design and Experiments. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *87*(3–4), 601–613. https://doi.org/10.1007/s10846-017-0486-9

Fierro, R., & Lewis, F. L. (1997). Control of a nonholomic mobile robot: Backstepping kinematics into dynamics. *Journal of Robotic Systems*, *14*(3), 149–163. https://doi.org/10.1002/(SICI)1097-4563(199703)14:3<149::AID-ROB1>3.3.CO;2-N

Gao, X., Yan, L., & Gerada, C. (2021). Modeling and Analysis in Trajectory Tracking Control for Wheeled Mobile Robots with Wheel Skidding and Slipping: Disturbance Rejection Perspective. *Actuators*, *10*(9), 222. https://doi.org/10.3390/act10090222

Garbaabaa, H. A., Geda, M. G., Wase, M. G., Ranganathan, S., Jin, G.-G., & Yung-Deug, S. (2020). A Self-Tuning NPID Control Method for FOPTD Processes. *Preprints2020*, *2020040410*, 1–13. https://doi.org/(doi: 10.20944/preprints202004.0410.v1).

Hadi, N. H., & Younus, K. K. (2020). Path tracking and backstepping control for a wheeled mobile robot (WMR) in a slipping environment. *IOP Conference Series: Materials Science and Engineering*, *671*(1), 012005. https://doi.org/10.1088/1757-899X/671/1/012005

Hassan, N., & Saleem, A. (2022). Neural Network-Based Adaptive Controller for Trajectory Tracking of Wheeled Mobile Robots. *IEEE Access*, *10*(i), 13582–13597. https://doi.org/10.1109/ACCESS.2022.3146970

Hassani, I., Maalej, I., & Rekik, C. (2020). Backstepping tracking control for nonholonomic mobile robot. *2020 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, 63–68. https://doi.org/10.1109/IC_ASET49463.2020.9318221

Hwang, E.-J., Kang, H.-S., Hyun, C.-H., & Park, M. (2013). Robust Backstepping Control Based on a Lyapunov Redesign for Skid-Steered Wheeled Mobile Robots. *International Journal of Advanced Robotic Systems*, *10*(1), 26. https://doi.org/10.5772/55059

Jabeur, C. Ben. (2019). Design of RTD-PID optimized neural networks controller for non-holonomic wheeled mobile robot. *International Robotics & Automation Journal*, *5*(5), 168–177. https://doi.org/10.15406/iratj.2019.05.00191

JIN, G.-G., & SON, Y.-D. (2019). Design of a Nonlinear PID Controller and Tuning Rules for First-Order Plus Time Delay Models. *Studies in Informatics and Control*, *28*(2), 157–166. https://doi.org/10.24846/v28i2y201904

KALYONCU, M., & DEMİRBAŞ, F. (2017). DIFFERENTIAL DRIVE MOBILE ROBOT TRAJECTORY TRACKING WITH USING PID AND KINEMATIC BASED BACKSTEPPING CONTROLLER. *Selcuk University Journal of Engineering ,Science and Technology*, *5*(1), 1–15. https://doi.org/10.15317/Scitech.2017.65

Kanayama, Y., Kimura, Y., Miyazaki, F., & Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. *Proceedings., IEEE International Conference on Robotics and Automation*, *91*(91), 384–389. https://doi.org/10.1109/ROBOT.1990.126006

Klančar, G., Zdešar, A., Blažič, S., & Škrjanc, I. (2017). Control of Wheeled Mobile Systems. In *Wheeled Mobile Robotics* (pp. 61–159). Elsevier. https://doi.org/10.1016/B978-0-12-804204-5.00003-2

Mai, T. A., Dang, T. S., Duong, D. T., Le, V. C., & Banerjee, S. (2021). A combined backstepping and adaptive fuzzy PID approach for trajectory tracking of autonomous mobile robots. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, *43*(3), 156. https://doi.org/10.1007/s40430-020-02767-8

Martins, N. A., & Bertol, D. W. (2022). Wheeled Mobile Robot Control. In *Studies in Systems, Decision and Control 380* (Vol. 380). springer. https://link.springer.com/10.1007/978-3-030-77912-2

Messom, C. (2002). Genetic Algorithms for Auto-tuning Mobile Robot Motion Control. *Res. Lett. Inf. Math. Sci*, *3*(2002), 129–134. http://www.massey.ac.nz/~wwiims/research/letters/

Mohamed, M., & Hamza, M. (2019). Design PID Neural Network Controller for Trajectory Tracking of Differential Drive Mobile Robot Based on PSO. *Engineering and Technology Journal*, *37*(12A), 574–583. https://doi.org/10.30684/etj.37.12A.12

Mohareri, O., Dhaouadi, R., & Rad, A. B. (2012). Indirect adaptive tracking control of a nonholonomic mobile robot via neural networks. *Neurocomputing*, *88*, 54–66. https://doi.org/10.1016/j.neucom.2011.06.035

Moqbel Obaid, M. A., Husain, A. R., & Mohammed Al-kubati, A. A. (2016). Robust

Backstepping Tracking Control of Mobile Robot Based on Nonlinear Disturbance Observer. *International Journal of Electrical and Computer Engineering (IJECE)*, *6*(2), 901. https://doi.org/10.11591/ijece.v6i2.9594

Rabbani, M. J., & Memon, A. Y. (2021). Trajectory Tracking and Stabilization of Nonholonomic Wheeled Mobile Robot Using Recursive Integral Backstepping Control. *Electronics*, *10*(16), 1992. https://doi.org/10.3390/electronics10161992

Raptis, I. A., & Valavanis, K. P. (2011). Linear and Nonlinear Control of Small-Scale Unmanned Helicopters. In *Linear and Nonlinear Control of Small-Scale Unmanned Helicopters* (Vol. 45, Issue 1). Springer Netherlands. https://doi.org/10.1007/978-94-007-0023-9

Ren, C., Ji, J.-H., Yan, H.-Y., Zhang, H., & Yue, J.-Z. (2017). A Backstepping Control Method for Mobile Robot Path Tracking. *Proceedings of the 3rd Annual International Conference on Mechanics and Mechanical Engineering (MME 2016)*, *105*. https://doi.org/10.2991/mme-16.2017.94

Rubio, F., Valero, F., & Llopis-Albert, C. (2019). A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, *16*(2), 172988141983959. https://doi.org/10.1177/1729881419839596

Siegwart R, Nourbakhsh IR, and D. S. (2011). Introduction to Autonomous Mobile Robots. In *MIT press cambridge, MA* (2nd ed.). https://mitpress.mit.edu/books/introduction-autonomous-mobile-robots-second-edition

Smith, R. C., Antoulas, A. C., Betts, J., Delfour, M. C., Gunzburger, M. D., Helton, J. W., Krener, A. J., & Murray, R. (2016). Boundary Control of PDEs Editor-in-Chief Editorial Board Series Volumes. In *Society for Industrial and Applied Mathematics Philadelphia*. Elsevier.

Song, Y.-D. (2018). Control of Nonlinear Systems via PI, PD and PID. In *Control of Nonlinear Systems via PI, PD and PID*. CRC Press. https://doi.org/10.1201/9780429455070

Tzafestas, S. G. (2014). Introduction to Mobile Robot Control. In *Introduction to Mobile Robot Control*. Elsevier. https://doi.org/10.1016/C2013-0-01365-5

Uddin, N. (2018). Trajectory Tracking Control System Design For Autonomous Two-

Wheeled Robot. *JURNAL INFOTEL*, *10*(3), 90. https://doi.org/10.20895/infotel.v10i3.393

Vaidyanathan, A. T. A. S. (2021). Backstepping Control of Nonlinear Dynamical Systems. In *Backstepping Control of Nonlinear Dynamical Systems*. Elsevier. https://doi.org/10.1016/C2018-0-02049-6

Xu, L., Du, J., Song, B., & Cao, M. (2022). A combined backstepping and fractional-order PID controller to trajectory tracking of mobile robots. *Systems Science & Control Engineering*, *10*(1), 134–141. https://doi.org/10.1080/21642583.2022.2047125

Xu, Q., Kan, J., Chen, S., & Yan, S. (2014). Fuzzy PID Based Trajectory Tracking Control of Mobile Robot and its Simulation in Simulink. *International Journal of Control and Automation*, *7*(8), 233–244. https://doi.org/10.14257/ijca.2014.7.8.20

Yildirim, Ş., & Savaş, S. (2013). Trajectory control of a mobile robot using neural network for frail blind persons. *International Review of Applied Sciences and Engineering*, *4*(1), 27–34. https://doi.org/10.1556/irase.4.2013.1.4

Yousuf, B. M., Saboor Khan, A., & Munir Khan, S. (2021). Dynamic modeling and tracking for nonholonomic mobile robot using PID and back-stepping. *Advanced Control for Applications*, *3*(3), 1–12. https://doi.org/10.1002/adc2.71

Zaidner, G., Korotkin, S., Shteimberg, E., Ellenbogen, A., Arad, M., & Cohen, Y. (2010). Non linear PID and its application in process control. *2010 IEEE 26th Convention of Electrical and Electronics Engineers in Israel, IEEEI 2010*, 574–577. https://doi.org/10.1109/EEEI.2010.5662155

Zangina, U., Buyamin, S., Abidin, M. S. Z., Mahmud, M. S. A., & Hasan, H. S. (2020). Non-linear PID controller for trajectory tracking of a differential drive mobile robot. *Journal of Mechanical Engineering Research and Developments*, *43*(7), 255–269. http://eprints.utm.my/id/eprint/90651/1/UmarZangina2020_NonLinearPIDControllerforTrajectoryTracking.pdf

APPENDIXES

Appendix A: Matlab Functions

Appendix A1: Dynamic Model Matlab Function

```
function [vDot, omegaDot] = Dynam (Tr, Tl, v, omega, theta)
mt = 120;              % The robot mass
r   = 0.1350;          % The radius of the wheels
d   = 0.1;             % The distance between the CoM and wheels axis
L   = 0.330;           % The lateral distance of the wheels to the center
I   = 15.0656;         % inertial of motor
H = [mt 0 mt*d*sin(theta); 0 mt -mt*d*cos(theta); mt*d*sin(theta) -mt*d*cos(theta) I];
C = [0 0 mt*d*omega*cos(theta); 0 0 mt*d*omega*sin(theta); 0 0 0];
S = [cos(theta) -d*sin(theta); sin(theta) d*cos(theta);0 1];
E =[cos(theta)/r cos(theta)/r; sin(theta)/r sin(theta)/r; L/r -L/r];
Sdot =[-omega*sin(theta) -d*omega*cos(theta); omega*cos(theta) -d*omega*sin(theta); 0
0];
Hhat = S'*H*S;        Chat = S'*C*S+S'*H*Sdot;          Ehat = S'*E;
  T = [Tr; Tl];                          % input Torques
  w = [v; omega];                        % Velocity vector
vdot = (Hhat)\(Ehat*T-Chat*w);
vDot = vdot (1);      omegaDot = vdot (2);  % Accelerations Outputs
```

Appendix A2: Kinematic Model Matlab Function

```
function [xDot, yDot, thetaDot] = Kinem (v, omega, theta)
S = [cos(theta) -d*sin(theta); sin(theta) d*cos(theta); 0 1];
w = [v; omega];
qdot = S*w;
xDot = qdot (1); yDot = qdot (2); thetaDot = qdot (3);
```

Appendix A3: Nonlinear Feedback Torque Controller Matlab Function

```
function [Tr, Tl] = feedback_T (v, omega, theta, u1, u2)
H = [mt 0 mt*d*sin(theta); 0 mt -mt*d*cos(theta); mt*d*sin(theta) -mt*d*cos(theta) I];
C = [0 0 mt*d*omega*cos(theta); 0 0 mt*d*omega*sin(theta); 0 0 0];
S = [cos(theta) -d*sin(theta); sin(theta) d*cos(theta); 0 1];
```

```
E = [cos(theta)/r cos(theta)/r; sin(theta)/r sin(theta)/r; L/r -L/r];
Sdot= [-omega*sin(theta) -d*omega*cos(theta); omega*cos(theta) -d*omega*sin(theta); 0
0];
Hhat = S'*H*S;          Chat = S'*C*S + S'*H*Sdot;          Ehat = S'*E;
u = [u1; u2];
w = [v; omega];
T = (Ehat) \ (Hhat*u+ Chat*w);
Tr = T (1);      Tl = T (2);                    % output torques
```

Appendix A4: Kinematic-Based Backstepping Controller Matlab Function

```
function [vc, Omegac] = KC (Kx, Ky, Kth, vr, Omegar, ex, ey, eth)
vc = vr*cos(eth) + Kx*ex;
Omegac = Omegar + Ky*vr*ey + Kth*vr*sin(eth);  % Kinematic-based Controller
```

Appendix A5: NPID Controller Nonlinear Gain Matlab Function

```
function ke = scale_error(e)
% ec_change = 0.5;
 ec_change = 1;
% ec_change = 1.5;
ke = (exp(-e.^2/(2*ec_change^2))) *e;
```

Appendix A6: Trajectory Generation Matlab Function

```
function [xRef, yRef, tRef, vRef, omeRef] = tra(t)
  xRef = -5*sin(t/5);                    yRef = 10*sin(t/10);
 dxRef = -cos(t/5);                      yRef = cos(t/10);
ddxRef = sin(t/5)/5;                   ddyRef = -sin(t/10)/10;
  vRef = sqrt (dxRef. ^2 + dyRef. ^2);          tRef = atan2(dyRef, dxRef);
omeRef = (ddyRef. *dxRef - ddxRef. *dyRef)/ (dxRef. ^2+dyRef. ^2);
```

Appendix B: Matlab Codes

Appendix B1: Matlab Program for Backstepping Controller Optimization

```
function NPID_plus_BSC
clear; clc; close;
% rng default
options = optimoptions ('ga','PlotFcn', @gaplotbestf);
```

```
options. PopulationSize = 30;
options. MaxGenerations = 30;
nvar = 9;                                   % Number of parameters
  LB = [0 0 0 0 0 0 0 0 0];                  % Lower bound
  UB = [100 1 1 100 1 1 100 100 100];        % Upper bound
[x, fval] = ga (@EvalObj, nvar, [], [], [], [], LB, UB, [], options);
function obj = EvalObj(x)                   % Cost Function
simulink_model = 'BSC_NPID';
load_system(simulink_model);
gains= get_param (simulink_model, 'modelworkspace');
gains. assignin ('Kp1', x (1));
gains. assignin ('Ki1', x (2));
gains. assignin ('Kd1', x (3));
gains. assignin ('Kp2', x (4));
gains. assignin ('Ki2', x (5));
gains. assignin ('Kd2', x (6));
gains. assignin ('Kx', x (7));
gains. assignin ('Ky', x (8));
gains. assignin ('Kth', x (9));
simOut = sim (simulink_model, 'SaveOutput', 'on');
 t = simOut.find('t');
e1 = simOut.find('e11');
e2 = simOut.find('e22');
e3 = simOut.find('e');
e4 = simOut.find('e1');
e5 = simOut.find('e2');
 n = length(e3);
ea = t.*abs(e5) + t.*abs(e4) + t.*abs(e3) + t.*abs(e2) + t.*abs(e1);          % ITAE
obj= 0;
for i = 2: n
   stepsize = t(i)-t(i-1);
   obj = obj + 0.5*(ea(i-1) + ea(i)) *stepsize;
end
```

Appendix C: Matlab/Simulink System Block Diagrams

Appendix C1: Open-loop system simulation Simulink block

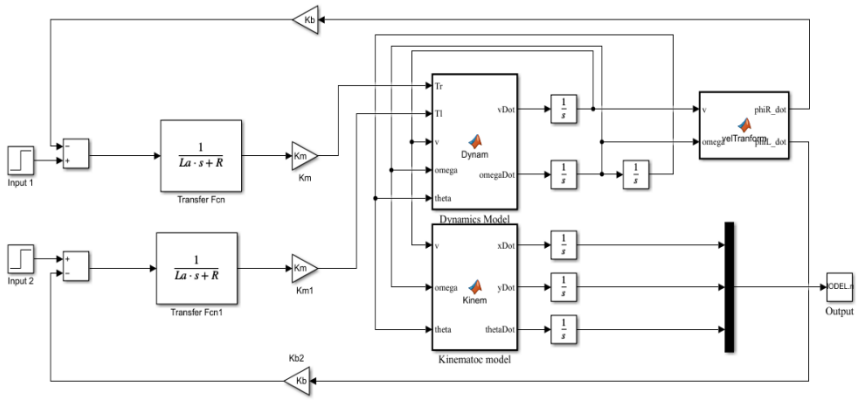

Figure C.1: Open-loop system Simulink model

Appendix C2: Overall system simulations Simulink block

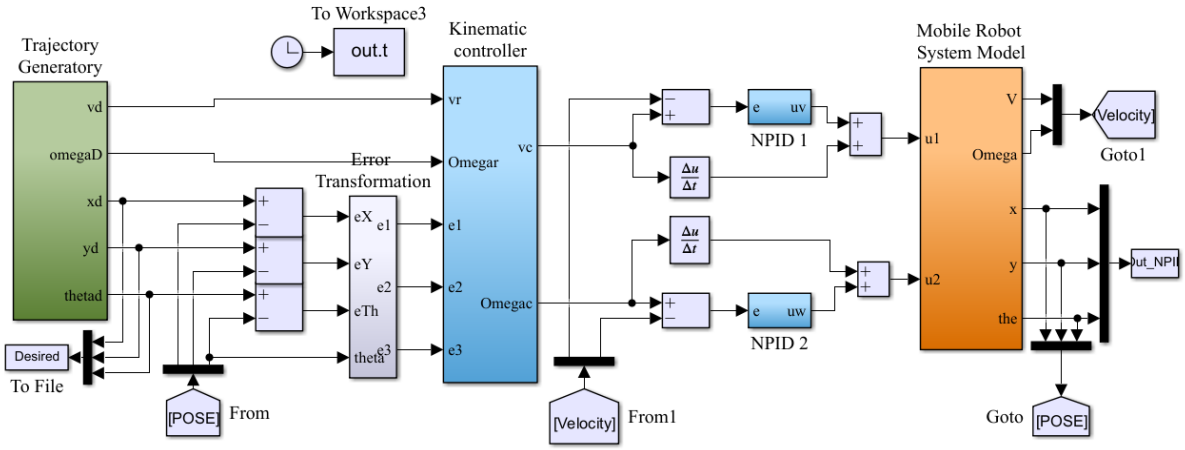

Figure C.2: Overall system Simulink block

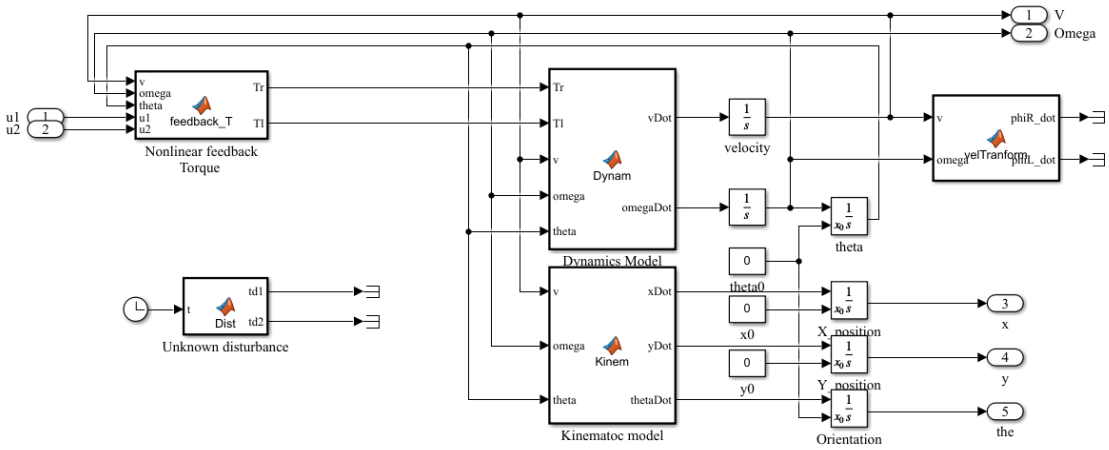

Figure C.3: Expanded mobile robot system model