5-2023

Modeling and Control Strategies for a Two-Wheel Balancing Mobile Robot

John Alan Moritz
University of Arkansas, Fayetteville

Modeling and Control Strategies for a Two-Wheel Balancing Mobile Robot

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mechanical Engineering

by

John Moritz
University of Arkansas
Bachelor of Science in Mechanical Engineering, 2020

May 2023
University of Arkansas

This thesis is approved or recommendation to the Graduate Council.

_____
Uche Wejinya PhD
Thesis Director

_____
Mark Arnold PhD
Committee Member

_____
Adam Huang PhD
Committee Member

Abstract

The problem of balancing and autonomously navigating a two-wheel mobile robot is an increasingly active area of research, due to its potential applications in last-mile delivery, pedestrian transportation, warehouse automation, parts supply, agriculture, surveillance, and monitoring. This thesis investigates the design and control of a two-wheel balancing mobile robot using three different control strategies: Proportional Integral Derivative (PID) controllers, Sliding Mode Control, and Deep Q-Learning methodology. The mobile robot is modeled using a dynamic and kinematic model, and its motion is simulated in a custom MATLAB/Simulink environment.

The first part of the thesis focuses on developing a dynamic and kinematic model for the mobile robot. The robot dynamics is derived using the classical Euler-Lagrange method, where motion can be described using potential and kinetic energies of the bodies. Non-holonomic constraints are included in the model to achieve desired motion, such as non-drifting of the mobile robot. These non-holonomic constraints are included using the method of Lagrange multipliers. Navigation for the robot is developed using artificial potential field path planning to generate a map of velocity vectors that are used for the set points for linear velocity and yaw rate.

The second part of the thesis focuses on developing and evaluating three different control strategies for the mobile robot: PID controllers, Hierarchical Sliding Mode Control, and Deep-Q-Learning. The performances of the different control strategies are evaluated and compared based on various metrics, such as stability, robustness to mass variations and disturbances, and tracking accuracy. The implementation and evaluation of these strategies are modeled tested in a MATLAB/SIMULINK virtual environment.

Acknowledgements

Firstly, I would like to thank my graduate supervisor, Dr. Uche Wejinya, for his guidance, support, and patience throughout my graduate study. His expertise in robotics and dedication to mentorship has shaped me into a fervent learner, in addition to numerous other engineering students at the University of Arkansas.

I would also like to thank Mishek Musa, a fellow member of the Micro-Nano Systems Engineering and Robotics Laboratory, for his valuable advice and perspective on robotics. His guidance has been instrumental in helping me navigate through my research work. Additionally, his friendship has been a cherished aspect of my graduate experience.

To my endlessly supportive parents, I'm forever grateful for your unyielding patience and love. Your support in helping me face every challenge and seize every opportunity that comes my way is immeasurable.

Finally, I want to express my deepest gratitude to my fiancée, Hannah. Your unwavering understanding and encouragement have been a constant source of inspiration, without which my graduate studies would have been considerably more arduous.

Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Background

The global market for mobile robotics is poised for significant expansion, with projections estimating an increase from $9.34 billion in 2019 to a staggering $64.08 billion by 2026. This impressive growth trajectory represents a compound annual growth rate (CAGR) of 12.9% [1]. One of the key drivers of this growing market has been the COVID-19 pandemic, which laid bare the vulnerabilities inherent in many industries and underscored the necessity for automation in numerous processes. As the pandemic prompted widespread lockdowns and mass quarantines to curb the transmission of the virus, industries heavily reliant on human labor faced unprecedented disruptions. Many were either forced to shut down completely or experienced frequent operational interruptions due to personnel exposures. In response to these challenges, substantial investments poured into the research and development of autonomous systems, with a particular emphasis on autonomous mobile robots [2]. Several other factors have also contributed to the rapid growth of the mobile robotics market. Among them are the declining costs of computer hardware (before the COVID-19 global pandemic), which has made the technology more accessible and affordable for a broader range of applications. Additionally, a resurgence in space exploration has spurred innovation and interest in mobile robotic systems designed for extraterrestrial environments. Furthermore, the need to perform hazardous tasks safely and efficiently has led to the development and deployment of mobile robots in industries such as mining, disaster relief, and defense [3].

Autonomous mobile robots are experiencing widespread adoption across a diverse range of sectors, including healthcare, military and defense, last-mile delivery, and transportation. One particularly intriguing type of mobile robot is the two-wheel, balancing robot (TWBR), which bears similarities to the classic Inverted Pendulum system. The Inverted Pendulum system has

long been a subject of extensive research due to its inherently unstable and nonlinear behavior. Unlike the Inverted Pendulum with its single degree of freedom, the TWBR boasts four degrees of freedom relative to a fixed global frame: x, y, pitch, and yaw [4]. The enhanced maneuverability of the TWBR makes it an optimal choice for navigating narrow spaces, such as hallways, sidewalks, and warehouses. This adaptability has led to its growing popularity in various applications, including last-mile delivery, pedestrian transportation, and warehouse automation. The unique properties of the TWBR have also sparked an upsurge in research interest, particularly in the development of control strategies for the system. Several similarly underactuated, unstable systems have already been successfully integrated into a range of technologies outside of academia. Notable examples include SpaceX's Falcon 9 landing function, the Segway PT personal transporter, and Boston Dynamics' Handle robot [5 - 7] . Consequently, the study and development of novel control strategies for the TWBR are of significant importance in further advancing the field of mobile robotics and its real-world applications.

1.2 System Configuration and Variables Definition

Variables

- x Position of the chassis in the $\hat{x}$ direction of a specified frame
- y Position of the chassis in the $\hat{y}$ direction of a specified frame
- z Position of the chassis in the $\hat{y}$ direction of a specified frame
- $\hat{x}$ x axis in a specified frame
- $\hat{y}$ y axis in a specified frame
- $\hat{z}$ z axis in a specified frame
- α Pitch angular position of the body center of mass about the local frame y-axis
- θ Yaw angular position of the robot in the about the global frame z-axis
- m Mass of the body

- M — Mass of each wheel
- f — Gravitational acceleration
- L — Distance from the concentric wheel axis to the body center of mass
- (I_x, I_y, I_z) — Mass Moment of Inertia of the body about the chassis frame's $\hat{x}, \hat{y}, \hat{z}$ axes
- (K, J, K) — Mass Moment of Inertia of each wheel, about the chassis frame's $\hat{x}, \hat{y}, \hat{z}$ axes
- d — Distance between the wheel centers of mass
- r — Wheel radius
- $\vec{P}$ — Position vector
- $\vec{\omega}$ — Rotational velocity vector
- $\vec{v}$ — Translational velocity vector
- γ — Wheel angular position
- R — Instantaneous radius of curvature
- L — Lagrangian Function
- T — Kinetic Energy Function
- V — Potential Energy Function
- F — Force
- λ — Lagrange multipliers
- μ — Coefficient of friction of the wheels
- u — Input Force
- ϕ — Input Torque
- q — Generalized Coordinates
- Q — Generalized Forces
- C — Generalized Constraint Forces
- a — Non-Holonomic Constraint Function
- $\boldsymbol{M}$ — Inertial Matrix
- $\boldsymbol{G}$ — Gravitational Matrix
- $\boldsymbol{C}$ — Coriolis Matrix
- $\boldsymbol{D}$ — Damping Matrix

- $\boldsymbol{B}$ Control Gain Matrix
- $\boldsymbol{\tau}$ Control Input Tensor

Subscripts

- L Left Wheel
- R Right Wheel
- w Wheel
- c Chassis
- b Wheel
- G Global Frame
- C Chassis Frame
- B Body Frame
- trans Translational
- rot Rotational
- $\hat{x}$ x axis in a specified frame
- $\hat{y}$ y axis in a specified frame
- $\hat{z}$ z axis in a specified frame

The two-wheel balancing robot represents a complex system that presents unique design challenges due to its underactuated nature. With limited direct control over all degrees of freedom (DOF), the system must leverage its inherent dynamics to maintain balance. Specifically, in the robot's chassis frame, the TWBR has three degrees of freedom: forward position (x), pitch angle (α), and yaw angle (θ). It is important to note that the pitch angle is highly coupled with the x position state variable, implying that changes in pitch will influence position, and vice versa. To better understand the system's design and dynamics, a computer-aided design (CAD) assembly of the two-wheel balancing robot was developed using SOLIDWORKS 2021, as depicted in Fig. 1. This enables tangible visualization of the system and extraction of essential mass properties, such as mass, mass moment of inertia, and center of mass, which will play a crucial role in the model

development process in subsequent sections of the thesis. The system comprises of two wheels actuated by DC motors. Unlike most "two-wheel" actuated systems, there is no caster wheel or additional supporting structure to prop the robot up. As a result, the system must rely solely on balancing its center of mass above the axle to maintain stability. This necessitates careful consideration of the robot's weight distribution, as well as precise control of the motors to achieve the required balance.

Table 1. TWBR Mass and Geometric Properties

m	6.575 kg
M	0.2121 kg
g	9.81 m/s^2
L	0.3895 m
$(I_{\hat{x}}, I_{\hat{y}}, I_{\hat{z}})$	$(1.128 \times 10^{-1}, 1.248 \times 10^{-1}, 4.641 \times 10^{-3})\ kg \cdot m^2$
(K, J, K)	$(5.229 \times 10^{-3}, 2.651 \times 10^{-4}, 5.229 \times 10^{-3})\ kg \cdot m^2$
d	0.15 m
r	0.11

The mass properties obtained from SOLIDWORKS are provided in Table 1, where m represents the mass of the body, M denotes the mass of each wheel, L signifies the normal distance from the body to the chassis, $I_{\hat{x}}, I_{\hat{y}}$, and $I_{\hat{z}}$ are the mass moments of inertia of the body about the local $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes respectively, and K, J, and K are the mass moments of inertia of the wheels about the chassis local $\hat{x}$, $\hat{y}$, and $\hat{z}$ axes respectively.

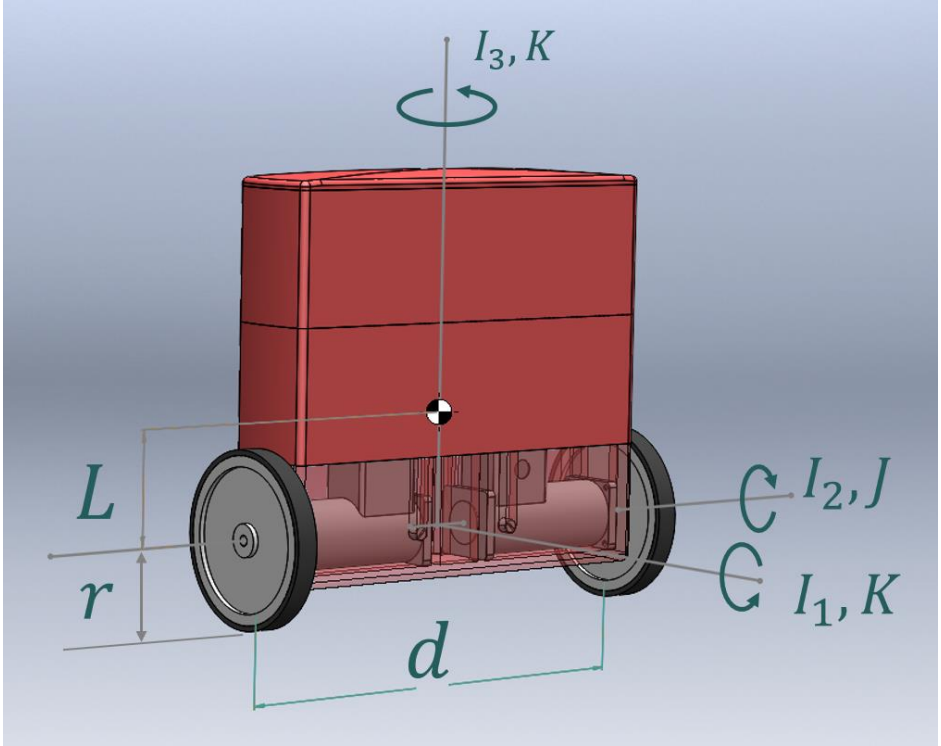

Figure 1. CAD Model of Two-Wheel Balancing Robot

1.3 Objective

The main objectives of this research are to develop a dynamic model for the two-wheel balancing mobile robot that accurately describes its motion and trajectories and to design and evaluate control strategies that can effectively stabilize the robot to achieve the desired motion. The evaluated control systems include PID, Sliding Mode Control, and Deep Q-Learning methodology. Through this research, these questions were identified and addressed:

- Can we apply these controllers to the TWBR for successful autonomous navigation?
- What are the advantages and disadvantages that each controller has over the others?
- Are these controllers well-suited for this application?
- What can be done to improve the performance of these controllers?

1.4 Approach

To answer the questions posed, we begin by developing a model for our system that using system its kinematics and dynamics. Obtaining the equations of motion for the Two-Wheeled Balancing Robot (TWBR) is complex, so we utilize the MATLAB Symbolic Math toolbox to perform the intricate algebraic substitutions and solve for the equations of motion in their highest order for the state variables. The equations of motion are then implemented in Simulink and integrated to obtain the variables of interest, validate the model, and make predictions. We then create a simplified version of the path planning algorithm using Artificial Potential Fields to establish set points for the controllers to follow. After that, we develop control architectures for PID, Sliding Mode Control, and Deep Q-learning methodology. These controllers are tested and evaluated to determine their effectiveness in controlling the TWBR.

1.5 Literature Review

1.5.1 System Modeling

There are multiple approaches to modeling the system. These approaches include but not limited to Newtonian mechanics, Kane's mechanics, or Lagrange mechanics. Using Newton's laws is a classical approach to dynamic modeling based on the concept of defining all external forces acting on the system. This approach is well suited for simple systems, but due to forces being represented as vector quantities and the necessity of including reaction forces, dynamic modeling complex systems using Newtonian mechanics can quickly become very involved. The Lagrange formulation uses the concept of kinetic and potential energies to derive the equations of motion. Lagrangian mechanics relies fundamentally on a quantity called the Lagrangian function, which is the difference between kinetic and potential energies. Due to its less-involved formulation, Lagrangian mechanics is well suited for constrained multibody systems, such as ours

[8]. However, for systems such as wheeled-mobile robots, non-holonomic constraints must be considered in order to accurately describe the motion. Non-holonomic systems have constraints on their motion. For example, for the two-wheel drive mobile robot, the robot cannot travel laterally in the axis normal to its wheel axis. This is what makes it considered a non-holonomic system. In order to account for non-holonomic constraints, we must utilize Lagrange multipliers, which represent reaction forces, such as friction (non-conservative force), for our system [9]. Therefore, we derive the Dynamic model of the TWBR using Euler-Lagrange approach. There are existing publications detailing different types of controllers used for TWBR with a few different methods of deriving its dynamic model. Many designs include a third castor wheel to keep it stabilized, eliminating a degree of freedom for the system. This significantly simplifies the dynamic model as well as the control model for the robot. The existence of a castor wheel can impede the robot's maneuverability in very narrow or cluttered spaces. The TWBR described by this research has no castor wheel and is therefore highly unstable.

1.5.2 Path Planning

Artificial potential fields will be used for path planning due to their simplicity, low computational cost, and effectiveness for generating trajectory set points for mobile systems. There are problems with Artificial Potential Field algorithms that we don't address here, since the scope of this project is focused more on dynamic modeling and controller design rather than developing a robust path planning algorithm. However, solutions to the problems with Artificial Potential Fields are well researched and will be discussed in this section along with the general idea being the algorithm itself.

The Artificial Potential Field algorithm consists of assigning repulsive potential fields to obstacles and an attractive potential field to the target. The potential field map can be represented

as a vector field where the magnitude of the vector is interpreted as velocity magnitude and the desired yaw angle can be extracted from the vector angle (relative to the global frame). One major issue with Artificial Potential Fields is the problem of local minima. Some obstacle geometries such as the one shown in Fig. 1 can generate a local minimum that essentially traps the robot. There are many other methods for this, such as one proposed by Lee & Park, where they place virtual obstacles at or near local minima to alter the potential field [10]. Iswanto et al. propose a modified Artificial Potential Field algorithm for a Quadrotor unmanned aircraft where the potential force at locations of local minima can be increased until the aircraft is repulsed to escape [11]. Lee et al. proposes a solution to local minima by placing an attractive potential field that draws the robot out of the local minimum. Their method is dubbed New Point-APF (NP-APF) [12]. Bounini et al. propose a modified APF method that utilizes the well-known A-Star search algorithm by adding a repulsive potential field to the local minima and searching for a feasible path using the A-Star technique [13].

1.5.3 Controllers

There are many controller options for this application of the TWBR. Matthew et al. propose a controller design an optimized PID controller design of the TWBR. However, their controller is designed for pitch stability only, and it neglects the other control variables needed for autonomous navigation [14]. Jang et al. show a gain scheduling approach to the design of fuzzy logic controllers, where the feedback gains are changed based on the Takagi-Sugeno fuzzy controller, which uses nested "if-then" statements to generate the weights [15]. They simulate their controller method for the cart-pole system as well as the Ball-Beam problem, both highly unstable, nonlinear systems. Deep Q-Learning was originally developed by Mnih et al. in their manuscript "Playing Atari with Deep Reinforcement Learning". The authors introduce the use of experience replay,

which randomly samples previous experiences when updating the neural network. This allowed for a much more stable and efficient learning process. They use Convolutional Neural Networks to estimate the quality function. This makes it highly suited for high dimensional observation spaces with a discrete, limited action space [16]. Carlucho et al. develop an adaptive PID controller for mobile robots, that uses a DQN agent trained to change gain values to achieve desired results. This is similar to gain scheduling; however, the gains are adaptively scheduled by an intelligent agent [17].

CHAPTER 2: TWO-WHEEL BALANCING ROBOT SYSTEM MODEL

2.1 Coordinate Frames

Before modeling, it is crucial to define and understand the three coordinate frames associated with the system. For the robot to navigate through a workspace, or even to understand its dynamics, its position and orientation in space must be known relative to some reference frame. For the purpose of our model, the system is defined as consisting of three rigid bodies: Left wheel, Right wheel, and the body, denoted by the subscript L, R, and b, respectively. There are three coordinate frames associated with the robot: the fixed global frame denoted by the subscript G, the chassis frame denoted by C, and the robot body frame denoted by B. The global frame is fixed in space and is used as a reference frame for describing the position and orientation of the robot and its environment. The chassis frame is attached to the robot's base and is used to help describe the orientation of the robot. The robot body frame is attached to the robot's base and is also used to describe the orientation of the robot's body. By understanding these coordinate frames, the TWBR's kinematics and dynamics can be modeled, and we can plan and execute trajectories that allow the robot to perform tasks in its environment. The rotation from frame C to B by angle α is given by the rotational matrix.

$$ {}^{B}_{C}R(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{1} $$

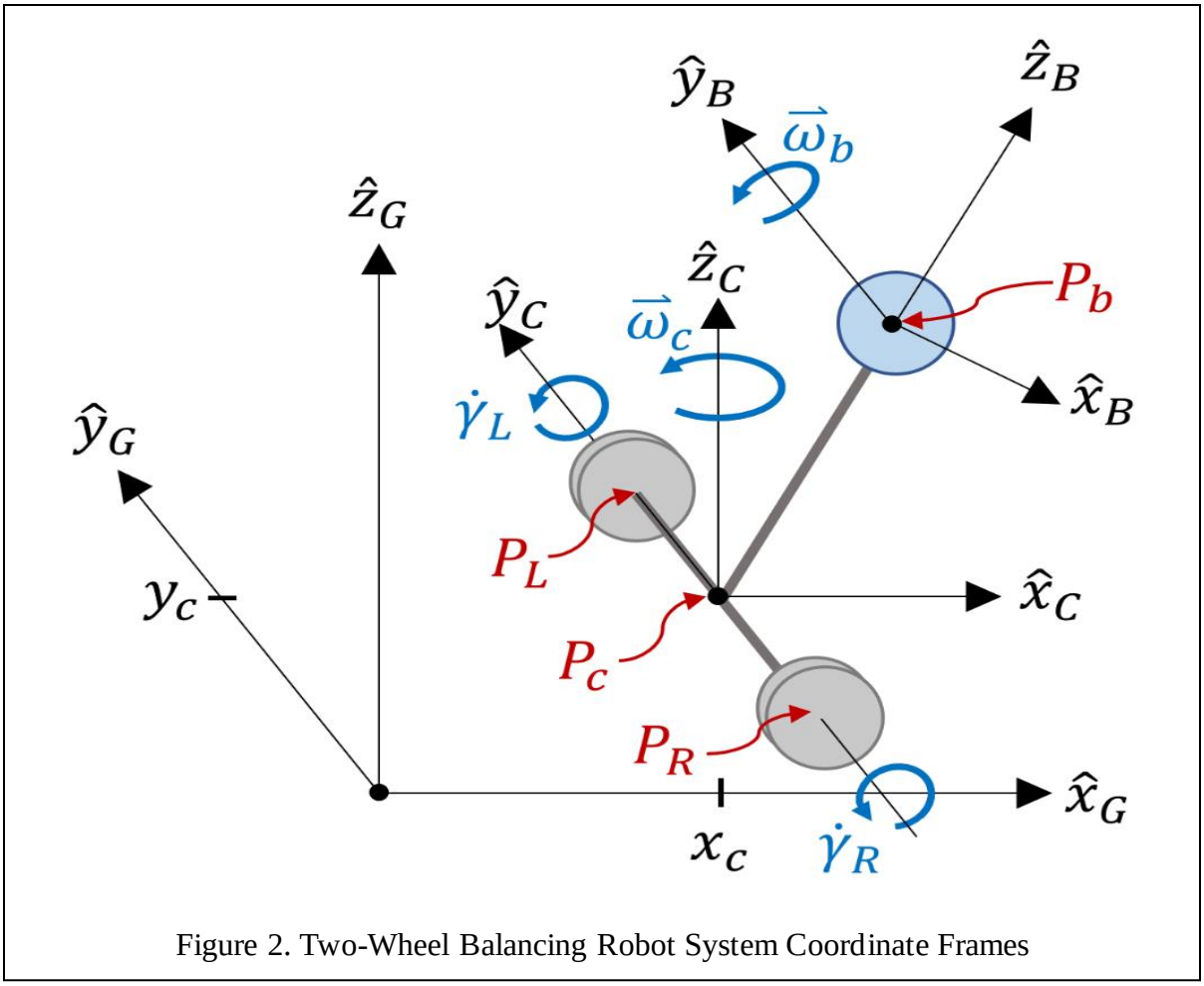

Figure 2. Two-Wheel Balancing Robot System Coordinate Frames

Figure 2 shows the three coordinate systems associated with the robot: the fixed global frame, chassis frame, and the body frame denote. The figure provides a visual representation of the orientation and relationship between these frames. $\dot{\gamma}$ is used to denote the individual wheel

velocities, P is used to denote the position vectors of each rigid body, and $\vec{\omega}$ denotes the rotational vectors, which are described in equation (2)

$$\vec{\omega}_c = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta} \end{bmatrix}; \vec{\omega}_b = \begin{bmatrix} 0 \\ \dot{\alpha} \\ 0 \end{bmatrix} \tag{2}$$

These rotational vectors describe motion relating to the pitch rate, $\dot{\alpha}$, and the yaw rate, $\dot{\theta}$. By understanding how these frames are oriented and related to each other, the robot's movement can accurately be modeled and plan trajectories that allow it to perform tasks in its environment. [18]

2.2 Kinematic Constraints

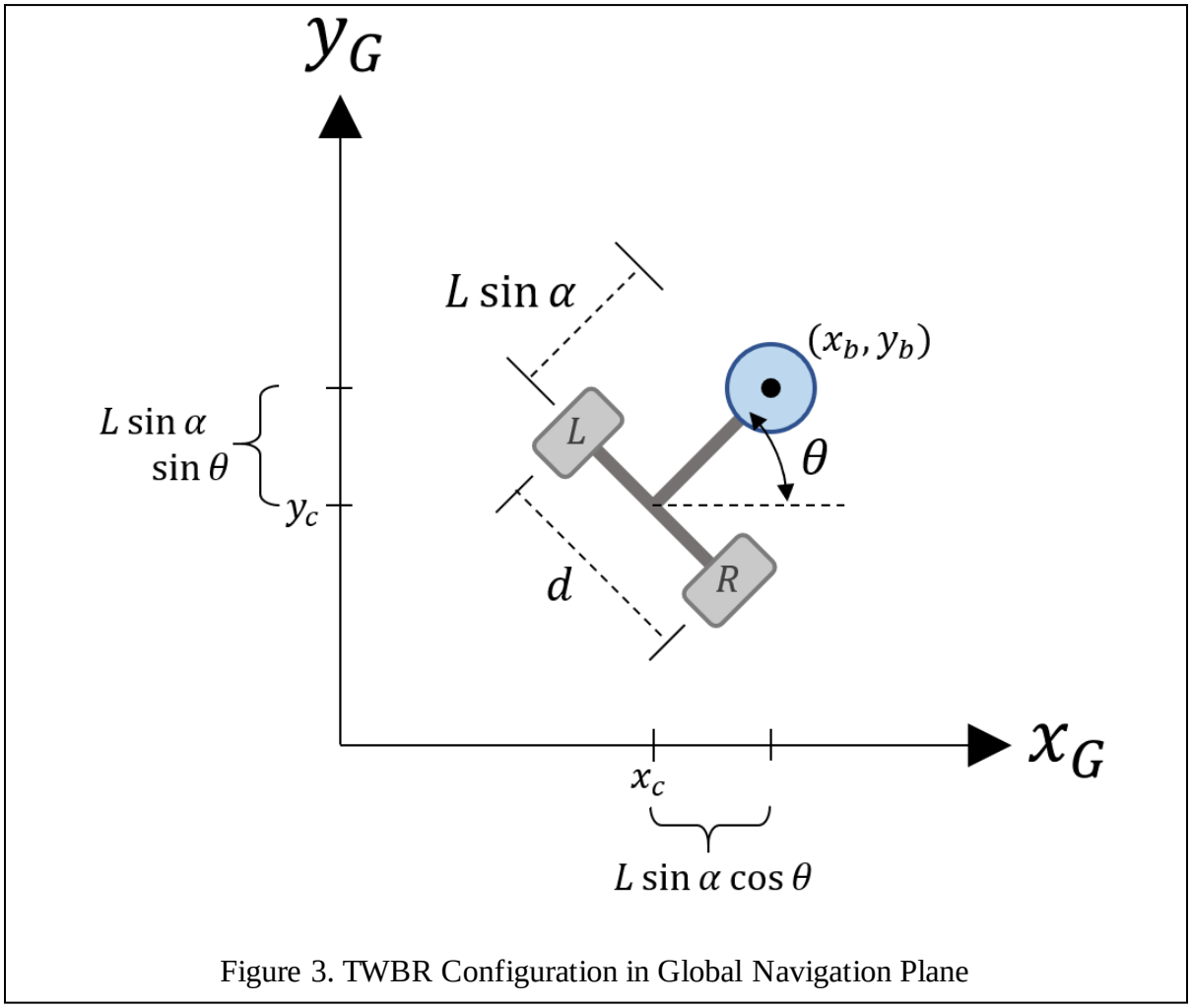

Figure 3. TWBR Configuration in Global Navigation Plane

The TWBR is subject to both Holonomic and non-Holonomic constraints. Holonomic constraints are geometric constraints on configuration, which reduce the total number of controllable degrees of freedom [19]. Non-holonomic constraints constrain the motion of a system. There are three holonomic or geometric constraints relating to the left wheel, right wheel, and the body to the chassis center coordinate. As shown in Fig. 3, the center of mass of the three rigid bodies are geometrically constrained to the chassis center by the following holonomic conditions, with respect to the global frame:

$$p_b = \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} x_C + L\sin\alpha\cos\theta \\ y_C + L\sin\alpha\sin\theta \\ L\cos\alpha \end{bmatrix} \tag{3}$$

$$p_L = \begin{bmatrix} x_L \\ y_L \\ z_L \end{bmatrix} = \begin{bmatrix} x_C - \frac{d}{2}\sin\theta \\ y_C + \frac{d}{2}\cos\theta \\ 0 \end{bmatrix} \tag{4}$$

$$p_R = \begin{bmatrix} x_R \\ y_R \\ z_R \end{bmatrix} = \begin{bmatrix} x_C + \frac{d}{2}\sin\theta \\ y_C - \frac{d}{2}\cos\theta \\ 0 \end{bmatrix} \tag{5}$$

The two non-holonomic constraints are “pure rolling” of the wheels, and “non-drifting”. A pure rolling constraint occurs when a wheel rolls without slipping or skidding along a surface. The pure rolling constraints can be represented using the following equations.

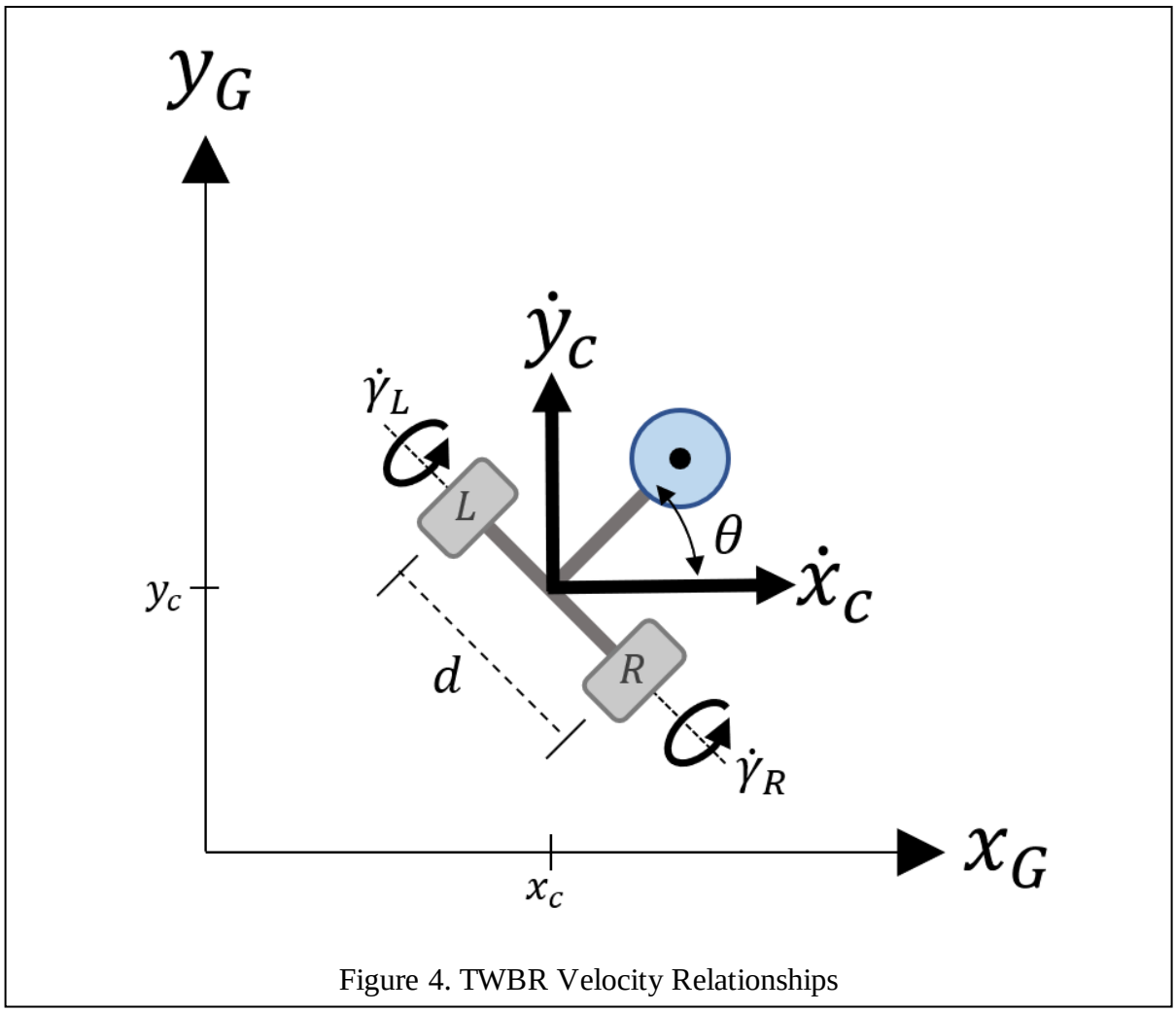

Figure 4. TWBR Velocity Relationships

$$\dot{x}_L \cos\theta + \dot{y}_L \sin\theta = r\dot{\gamma}_L \tag{6}$$

$$\dot{x}_R \cos\theta + \dot{y}_R \sin\theta = r\dot{\gamma}_R \tag{7}$$

Equations (6) and (7) can be altered by substituting in equations (4) and (5) to result in the following relationships.

$$\dot{x}_c \cos\theta + \dot{y}_c \sin\theta - \left(\frac{d}{2}\right)\dot{\theta} - r\dot{\gamma}_L = 0 \tag{8}$$

$$\dot{x}_c \cos\theta + \dot{y}_c \sin\theta + \left(\frac{d}{2}\right)\dot{\theta} - r\dot{\gamma}_R = 0 \tag{9}$$

By referring to Fig. 4, we can see how equations (6) and (7) can be related to the chassis center. The mapping of the chassis velocity from the global frame to the chassis frame can be summarized by the following relationship:

$$\dot{x}_c \cos\theta - \dot{y}_c \sin\theta = \dot{x} \tag{10}$$

A non-drifting constraint refers to the constraint that the robot's wheels must not slide or skid along their local y-axis, which is perpendicular to the direction of forward motion. The non-lateral drifting constraint is represented by zero-velocity in the direction perpendicular to the heading velocity, and it is mathematically related using equation (11).

$$\dot{x}_c \sin\theta - \dot{y}_c \cos\theta = 0 \tag{11}$$

2.3 Navigational Kinematics

For mobile robotics, inverse kinematics is the most appropriate since the motor inputs necessary to achieve a desired motion and configuration can be found. We start by relating our linear and rotational velocities to wheel angular velocities. Consider the case where the left wheel has a slower linear velocity than the right wheel. In this case, the robot will be rotating about a point called the instantaneous center of curvature (ICC). The robot's linear velocity, v_c, is described by the following equation:

$$v_c = \dot{\theta} R \tag{12}$$

Where R is the distance from the ICC to the midpoint between the two wheels and $\dot{\theta}$ is the angular velocity of the robot turning. Equation (12) can be changed to describe the wheel velocities both in terms of their individual rotational velocities about y_c, and in terms of their rotational yaw velocity about the ICC as follows.

$$\vec{v}_L = \dot{\gamma}_L r = \dot{\theta}(R + d) \tag{13}$$

$$\vec{v}_R = \dot{\gamma}_R r = \dot{\theta}(R - d) \tag{14}$$

Solving equation (12) for R and substituting it into equations (13) and (14) will result in the following relationships.

$$\dot{\gamma}_L = \frac{\dot{x} + \dot{\theta} d}{r} \tag{15}$$

$$\dot{\gamma}_R = \frac{\dot{x} - \dot{\theta} d}{r} \tag{16}$$

Here, we have the inverse kinematic equations that describe our individual wheel velocities written in terms of linear velocity and angular yaw velocity.

2.4 Euler-Lagrange Formulation

The Euler-Lagrange method is used to determine the equations of motion using the Lagrangian function.

$$L = T - V \tag{17}$$

L is the Lagrangian function, T is the kinetic energy, and V is the potential energy. We begin by describing the kinetic and potential energies of each of the three rigid bodies we have defined. The Kinetic energy of our system is categorized by two types, translational and rotational, which is given by the subscripts $trans$ and rot respectively [20].

$$T_{trans} = \frac{1}{2} m((\vec{v}_b)^T \times \vec{v}_b) + \frac{1}{2} M((\vec{v}_L)^T \times \vec{v}_L) + \frac{1}{2} M((\vec{v}_R)^T \times \vec{v}_R) \tag{18}$$

$$T_{rot} = \frac{1}{2} (\vec{\omega}_b)^T \vec{I}_b \vec{\omega}_b + \frac{1}{2} (\vec{\omega}_L)^T \vec{I}_w \vec{\omega}_L + \frac{1}{2} (\vec{\omega}_R)^T \vec{I}_w \vec{\omega}_R \tag{19}$$

Here, m is the mass of the body, M is the mass of each wheel, and the inertia matrices are represented in equations (20) and (21)

$$\vec{I}_b = diag\{I_{\hat{x}}, I_{\hat{y}}, I_{\hat{z}}\} \tag{20}$$

$$\vec{I}_w = diag\{K, J, K\} \tag{21}$$

By taking the time derivative of equations (3) - (5), we find the velocity vectors of our rigid bodies, with respect to the fixed global frame, as:

$$\vec{v}_b = \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \end{bmatrix} = \begin{bmatrix} \dot{x}_C + \dot{\alpha} L \cos\alpha \cos\theta - \dot{\theta} L \sin\alpha \sin\theta \\ \dot{y}_C + \dot{\alpha} L \cos\alpha \sin\theta + \dot{\theta} L \sin\alpha \cos\theta \\ -\dot{\alpha} L \sin\alpha \end{bmatrix} \tag{22}$$

$$\vec{v}_L = \begin{bmatrix} \dot{x}_L \\ \dot{y}_L \\ \dot{z}_L \end{bmatrix} = \begin{bmatrix} \dot{x}_C - \dot{\theta} \frac{d}{2} \cos\theta \\ \dot{y}_C - \dot{\theta} \frac{d}{2} \sin\theta \\ 0 \end{bmatrix} \tag{23}$$

$$\vec{v}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{z}_R \end{bmatrix} = \begin{bmatrix} \dot{x}_C + \dot{\theta} \frac{d}{2} \cos\theta \\ \dot{y}_C + \dot{\theta} \frac{d}{2} \sin\theta \\ 0 \end{bmatrix} \tag{24}$$

The angular velocities of the three rigid bodies with respect to their moving frames C or B are described in (25)-(27).

$${}^{C}\vec{\omega}_b = {}^{B}_{C}R(\alpha) \times \vec{\omega}_c + {}^{B}\vec{\omega}_b = \begin{bmatrix} -\dot{\theta} \sin\alpha \\ \dot{\alpha} \\ \dot{\theta} \cos\alpha \end{bmatrix} \tag{25}$$

$$\vec{\omega}_L = \vec{\omega}_c + \begin{bmatrix} 0 \\ \dot{\gamma}_L \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\gamma}_L \\ \dot{\theta} \end{bmatrix} \tag{26}$$

$$\vec{\omega}_R = \vec{\omega}_c + \begin{bmatrix} 0 \\ \dot{\gamma}_R \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ \dot{\gamma}_R \\ \dot{\theta} \end{bmatrix} \tag{27}$$

By substituting equations (20) through (27) into equations (18) and (19), we find that our energies are given by:

$$T_{trans} = \left(M + \frac{m}{2}\right)\dot{x}_c^2 + \left(M + \frac{m}{2}\right)\dot{y}_c^2 + M\dot{\theta}^2\frac{d^2}{4} + \frac{m}{2}\dot{\alpha}^2L^2 + \frac{m}{2}\dot{\theta}L^2(s^2\alpha) + m\dot{x}_c\dot{\alpha}L(c\alpha)(s\theta) -$$

$$m\dot{x}_c\dot{\theta}L(s\alpha)(s\theta) + m\dot{y}_c\dot{\alpha}L(c\alpha)(s\theta) + m\dot{y}_c\dot{\theta}L(s\alpha)(c\theta) \quad (28)$$

$$T_{rot} = \frac{I_{\hat{x}}}{2}\dot{\theta}^2(s^2\alpha) + \frac{I_{\hat{z}}}{2}\dot{\theta}^2(c^2\alpha) + \frac{I_{\hat{y}}}{2}\dot{\alpha}^2 + \frac{J}{2}(\dot{\gamma}_L^2 + \dot{\gamma}_R^2) + \dot{\theta}^2K \quad (29)$$

$$V = mgL(c\alpha) \quad (30)$$

For convenience, we substitute $(c\alpha)$, $(s\alpha)$, $(c^2\alpha)$, and $(s^2\alpha)$, for $\cos(\alpha)$, $\sin(\alpha)$, $\cos^2(\alpha)$, and $\sin^2(\alpha)$, respectively. By substituting our Kinetic and Potential energies into equation (17), we can formulate our Lagrangian function as follows:

$$L = \left(M + \frac{m}{2}\right)\dot{x}_c^2 + \left(M + \frac{m}{2}\right)\dot{y}_c^2 + M\dot{\theta}^2\frac{d^2}{4} + \frac{m}{2}\dot{\alpha}^2L^2 + \frac{m}{2}\dot{\theta}L^2(s^2\alpha) + m\dot{x}_c\dot{\alpha}L(c\alpha)(s\theta) -$$

$$m\dot{x}_c\dot{\theta}L(s\alpha)(s\theta) + m\dot{y}_c\dot{\alpha}L(c\alpha)(s\theta) + m\dot{y}_c\dot{\theta}L(s\alpha)(c\theta) + \frac{I_{\hat{x}}}{2}\dot{\theta}^2(s^2\alpha) + \frac{I_{\hat{z}}}{2}\dot{\theta}^2(c^2\alpha) + \frac{I_{\hat{y}}}{2}\dot{\alpha}^2 +$$

$$\frac{J}{2}(\dot{\gamma}_L^2 + \dot{\gamma}_R^2) + \dot{\theta}^2K - mgL(c\alpha) \quad (31)$$

With the newly formulated Lagrangian function, we see that it is a function of six generalized coordinates q_i, which is given by:

$$q_1 = x_c$$

$$q_2 = y_c$$

$$q_3 = \alpha$$

$$q_4 = \theta$$

$$q_5 = \dot{\gamma}_L$$

$$q_6 = \dot{\gamma}_R$$

The Euler-Lagrange method does not inherently account for non-holonomic constraints and non-conservative forces, such as friction [21]. Therefore, generalized forces and the Lagrange multiplier method is employed to include these as follows.

$$\frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathrm{L}}{\partial \dot{\mathrm{q}}_i}\right) - \frac{\partial \mathrm{L}}{\partial \mathrm{q}_i} = \mathrm{Q}_i + \mathrm{C}_i \tag{32}$$

Here, Q_i represents the generalized forces such as inputs and non-conservative forces, and C_i represents the generalized constraint forces. For $i = 1, \ldots, 6$, we find Q_i to be describe by the following:

$$Q_1 = 0$$

$$Q_2 = 0$$

$$Q_3 = -(Q_5 + Q_6)$$

$$Q_4 = 0;$$

$$Q_5 = F_L - \mu(\dot{\gamma}_L - \dot{\alpha})$$

$$Q_6 = F_R - \mu(\dot{\gamma}_R - \dot{\alpha}) \tag{33}$$

The wheel friction is represented by μ. The generalized constraint forces are given by:

$$C_i = \sum_{j=1} \lambda_j a_{j,i} \tag{34}$$

Here, λ_j represents the Lagrange multipliers, and $a_{j,i}$ represents the constraint functions. These Lagrange multipliers help the generalized constraint matrix represent the reaction forces that would be meticulously model, had Newtonian mechanics method being utilized [18]. The constraint functions are as follows.

$$a_{1,i} = \frac{\partial}{\partial \dot{q}_i}\left(\dot{x}_c \cos\theta + \dot{y}_c \sin\theta - \frac{d}{2}\dot{\theta} - r\dot{\gamma}_L\right) \tag{35}$$

$$a_{2,i} = \frac{\partial}{\partial \dot{q}_i}\left(\dot{x}_c \cos\theta + \dot{y}_c \sin\theta + \frac{d}{2}\dot{\theta} - r\dot{\gamma}_R\right) \tag{36}$$

$$a_{3,i} = \frac{\partial}{\partial \dot{q}_i}(\dot{x}_c \sin\theta + \dot{y}_c \cos\theta) \tag{37}$$

Expanding the Euler-Lagrange equations of motion with respect to the generalized coordinates give the following equations of motion.

$$\ddot{x}_c(2M+m) + \ddot{\alpha}mL(c\alpha)(s\theta) - \ddot{\theta}mL(s\alpha)(s\theta) - \dot{\alpha}^2 mL(s\alpha)(s\theta) - \dot{\theta}^2 mL(s\alpha)(c\theta) + \dot{\alpha}\dot{\theta}mL(c\alpha)(c\theta) - \dot{\alpha}\dot{\theta}mL(c\alpha)(s\theta) = \lambda_1(c\theta) + \lambda_2(c\theta) + \lambda_3(s\theta) \tag{38}$$

$$\ddot{y}_c(2M+m) + \ddot{\alpha}mL(c\alpha)(s\theta) + \ddot{\theta}mL(s\alpha)(s\theta) - \dot{\alpha}^2 mL(s\alpha)(s\theta) + \dot{\theta}^2 mL(s\alpha)(c\theta) + \dot{\alpha}\dot{\theta}mL(s\alpha)(c\theta) + \dot{\alpha}\dot{\theta}mL(c\alpha)(s\theta) = \lambda_1(s\theta) + \lambda_2(s\theta) + \lambda_3(c\theta) \tag{39}$$

$$\ddot{\alpha}\left(mL^2 + I_{\hat{y}}\right) + \ddot{x}_c mL(c\alpha)(s\theta) + \ddot{y}_c mL(c\alpha)(s\theta) + \dot{x}_c\dot{\theta}mL(c\alpha)(c\theta + s\theta) - \dot{\theta}\frac{mL^2}{2}(s2a) - \dot{\theta}^2\frac{I_{\hat{x}}}{2}(s2a) + \dot{\theta}^2 I_{\hat{z}}(s\alpha)(c\alpha) - mgL(s\alpha) = -(F_L + F_R) + \mu(\dot{\gamma}_L + \dot{\gamma}_R) - 2\dot{\theta}\mu \tag{40}$$

$$\ddot{\theta}M\frac{d^2}{2} + 2K + I_{\hat{x}}(s^2\alpha) + I_{\hat{z}}(c^2\alpha) - \ddot{x}_c mL(s\alpha)(s\theta) + \ddot{y}_c mL(s\alpha)(c\theta) + \dot{\alpha}\frac{mL^2}{2}(s2\alpha) - \dot{x}_c\dot{\alpha}mL(c\alpha)(s\theta + c\theta) = -\lambda_1\frac{d}{2} + \lambda_2\frac{d}{2} \tag{41}$$

$$J\ddot{\gamma}_L = F_L - \mu\left(\dot{\gamma}_L - \dot{\theta}\right) - \lambda_1 r \tag{42}$$

$$J\ddot{\gamma}_R = F_R - \mu\left(\dot{\gamma}_R - \dot{\theta}\right) - \lambda_2 r \tag{43}$$

The first two Lagrange multiplier terms λ_1 and λ_2 can be solved for by using equations (42) and (43) to substitute into equations (38), (39), and (41). We can also solve for λ_3 using equation (38). Lastly, by applying the relationships in equations (10), (15), and (16), results in equations of motion

with respect to three generalized coordinates, x, α, and θ. We use the substitution of $u = (F_L + F_R)$ and $\phi = \frac{d(F_R - F_L)}{2}$ for simpler representation.

$$\left(2M + m + \frac{2J}{r^2}\right)\ddot{x} - mL\left(\dot{\theta}^2 + \dot{\alpha}^2\right)(s\alpha) + (c\alpha)(mL)\ddot{\alpha} + \frac{2\mu}{r}\left(\frac{\dot{x}}{r} - \dot{\alpha}\right) = \frac{u}{r} \quad (44)$$

$$\left(I_{\hat{y}} + mL^2\right)\ddot{\alpha} + (c\alpha)(mL)\ddot{x} + (c\alpha)(s\alpha)(I_{\hat{z}} - I_{\hat{x}} - mL^2)\dot{\theta}^2 - (s\alpha)mLg - 2\mu\left(\frac{\dot{x}}{r} - \dot{\alpha}\right) = -u \quad (45)$$

$$\left(I_{\hat{z}} + 3K + \frac{Md^2}{2} + \frac{Jd^2}{2r^2} + (I_{\hat{x}} + mL^2 - I_{\hat{z}})(s\alpha)^2\right)\ddot{\theta} + (mL\dot{x} + 2(c\alpha)(I_{\hat{x}} + mL^2 - I_{\hat{z}})\dot{\alpha})\dot{\theta}(s\alpha) + \frac{\mu\dot{\theta}d^2}{2r^2} = \frac{\phi}{(2r)} \quad (46)$$

These equations of motion can be partially linearized. Notice, that this will not fully linearize our equations of motion, but it will reduce the complexity of the model, and therefore it will reduce the computational cost that it takes to run simulation. We can linearize our equations by using small-angle approximation ($\alpha \approx 0$), where the following approximations are assumed.

$$\sin(\alpha) \approx \alpha \quad (47)$$

$$\cos(\alpha) \approx 1 \quad (48)$$

This linearization relies on the fact that we know we want the pitch to be at or very near the equilibrium point, $\alpha = 0$, at all times. This is because $\alpha = 0$, is a point of marginal stability for the system [21]. By substituting this approximation into equations (44-46) and solving these equations for $\ddot{x}, \ddot{\alpha}$, and $\ddot{\theta}$ in terms of first order time derivatives, we obtain these new equations of motion.

$$\left(2M + m + \frac{2J}{r^2}\right)\ddot{x} - mL\left(\dot{\theta}^2 + \dot{\alpha}^2\right)\alpha + (mL)\ddot{\alpha} + \frac{2\mu}{r}\left(\frac{\dot{x}}{r} - \dot{\alpha}\right) = \frac{u}{r} \quad (49)$$

$$\left(I_{\hat{y}} + mL^2\right)\ddot{\alpha} + (mL)\ddot{x} + \alpha(I_{\hat{z}} - I_{\hat{x}} - mL^2)\dot{\theta}^2 - \alpha mLg - 2\mu\left(\frac{\dot{x}}{r} - \dot{\alpha}\right) = -u \quad (50)$$

$$\left(I_{\hat{z}} + 3K + \frac{Md^2}{2} + \frac{Jd^2}{2r^2} + (I_{\hat{x}} + mL^2 - I_{\hat{z}})\alpha^2\right)\ddot{\theta} + \left(mL\dot{x} + 2\left(I_{\hat{x}} + mL^2 - I_{\hat{y}}\right)\dot{\alpha}\right)\dot{\theta}\alpha + \frac{c\dot{\theta}d^2}{2r^2} = \frac{\phi}{(2r)} \tag{51}$$

Now, we have our partially linearized equations of motion in terms of our three generalized coordinates. We can arrange these equations into the following matrix form [22]:

$$\boldsymbol{M}\ddot{q} + (\boldsymbol{C} + \boldsymbol{D})\dot{q} + \boldsymbol{G} = \boldsymbol{B}\boldsymbol{\tau} \tag{52}$$

where M is the inertia matrix, C is the Coriolis matrix, D is the damping matrix, G is the gravitational matrix, B is the control gain matrix, and τ is the control input. The matrices are given as follows:

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & m_{33} \end{bmatrix}$$

$$\boldsymbol{G} = \begin{bmatrix} 0 \\ g_2 \\ 0 \end{bmatrix}$$

$$\boldsymbol{C} = \begin{bmatrix} 0 & c_{12} & c_{13} \\ 0 & 0 & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

$$\boldsymbol{D} = \begin{bmatrix} d_{11} & d_{12} & 0 \\ d_{21} & d_{22} & 0 \\ 0 & 0 & d_{33} \end{bmatrix}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{1}{r} & 0 \\ -1 & 0 \\ 0 & \frac{1}{r} \end{bmatrix}$$

$$\boldsymbol{\tau} = \begin{bmatrix} u \\ \phi \end{bmatrix}$$

The entries for the previous matrices are given as follows:

$$m_{11} = m + 2M + \frac{2J}{r^2}; \qquad m_{12} = mL(c\alpha); \qquad m_{21} = mL(c\alpha);$$

$$m_{33} = I_{\hat{z}} + 2K + \frac{\left(M + \frac{J}{r^2}\right)d^2}{2} + (mL^2 + I_{\hat{x}} - I_{\hat{z}})(s\alpha)^2;$$

$$c_{12} = -mL\dot{\alpha}(s\alpha); \qquad c_{13} = mL\dot{\theta}(s\alpha);$$

$$c_{23} = -(mL^2 + I_{\hat{x}} - I_{\hat{z}})\dot{\theta}(c\alpha)(s\alpha); \qquad c_{31} = mL\dot{\theta}(s\alpha);$$

$$c_{32} = (mL^2 + I_{\hat{x}} - I_{\hat{z}})\dot{\theta}(s\alpha)(c\alpha); \qquad c_{33} = (mL^2 + I_{\hat{x}} - I_{\hat{z}})\dot{\alpha}(s\alpha)(c\alpha);$$

$$d_{11} = \frac{2\mu}{r^2}; \qquad d_{12} = -\frac{2\mu}{r} \qquad g_2 = -mLg(s\alpha)$$

The 2nd order state variables can be solved for like so

$$\ddot{q} = \boldsymbol{M}^{-1}\left(-(\boldsymbol{C} + \boldsymbol{D})\dot{q} - \boldsymbol{G}(q) + \boldsymbol{B}\boldsymbol{\tau}\right) \tag{53}$$

CHAPTER 3: PATH PLANNING

Variables

- U Potential Field
- ∇ Gradient
- Δ Step, or change in the variable it precedes
- ρ Threshold distance
- k Gain
- e Error
- π The constant

Subscripts

- d Desired
- n The n'th step
- g Goal
- a Attractive

Artificial Potential Field (APF) path planning is a method of global navigation, where the configuration space is ascribed an artificial potential field that draws the robot toward the potential minimum. APF's consist of two components: attractive fields and repulsive fields. The attractive field can be thought of as a "well' where the global minimum is the navigation goal. The repulsive fields are ascribed to obstacles within the configuration space. Figure 5 shows an example of a velocity vector field that maps the workspace for a mobile robot. The goal of the robot is to navigate towards the red circle, which represents the target location. However, there are obstacles in the way represented by the black circles. The vector field is designed such that the robot is repelled by the obstacles and attracted towards the target location. It can be seen that around the black circles, the velocity vectors are pointing away from the obstacles, and around the red circle, the vectors are

pointing towards it. This repulsive and attractive behavior of the vector field allows the robot to plan its path and avoid collisions with obstacles while navigating towards the target.

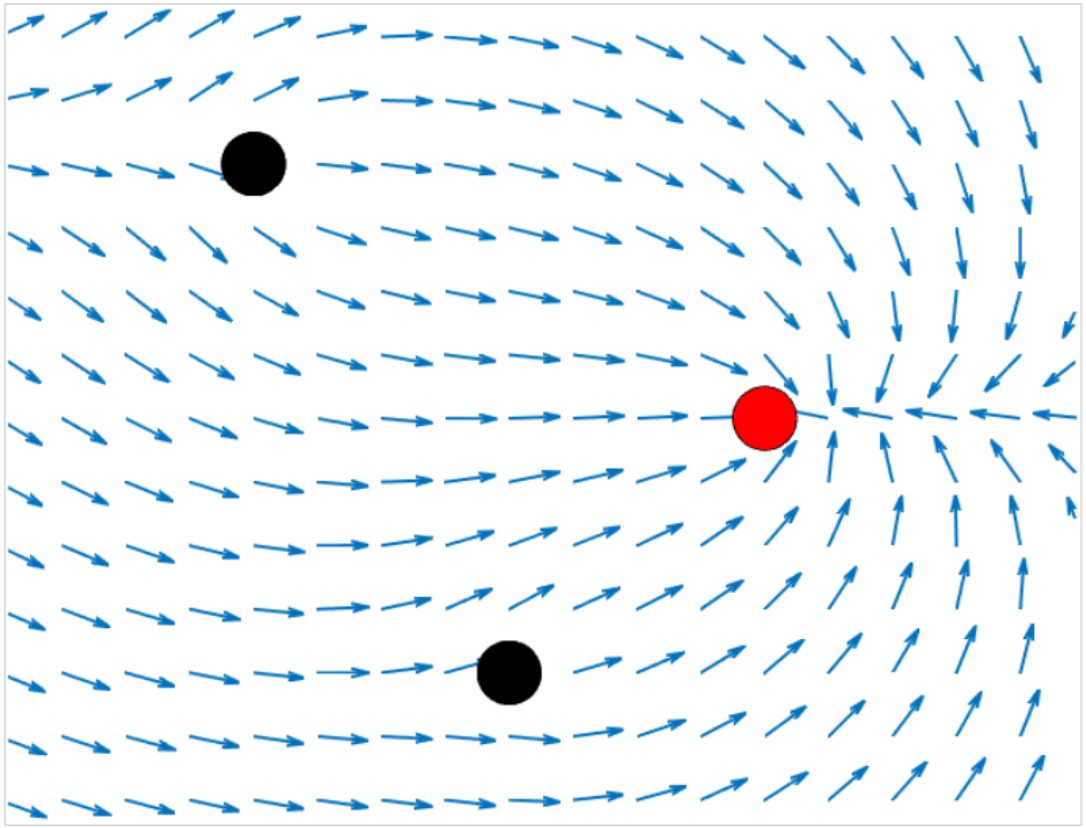

Figure 5. Vector Field with Repulsive Obstacles and Attractive Goal

The attractive field in potential field-based mobile robot navigation is split into two types: Conical and Paraboloidal. These potential fields are scalar fields whose gradient is interpreted as the velocity vector. Both conical and paraboloidal potential fields can be visualized in Fig. 6. In conical vector fields, the gradient magnitude is constant for any given point in the field, which is interpreted as constant velocity. In contrast, in paraboloidal vector fields, the magnitude of the gradient converges to zero closer to the goal (also known as a soft landing), but its magnitude increases infinitely further from the goal. However, for many mobile robot applications, the

desired velocity behavior is to have an upper bound and converge to zero at the goal. To achieve this behavior, we combine these two fields in a piecewise function, while maintaining continuity at a threshold distance from the goal. Specifically, we use a linear combination of the two fields with a smooth transition from the conical field to the paraboloidal field. The resulting attractive field has a maximum velocity that decreases smoothly as the robot approaches the goal, and converges to zero at the goal, producing a soft-landing behavior. This combined attractive field, along with the repulsive field created around obstacles, allows the mobile robot to navigate safely and efficiently to its goal in complex environments.

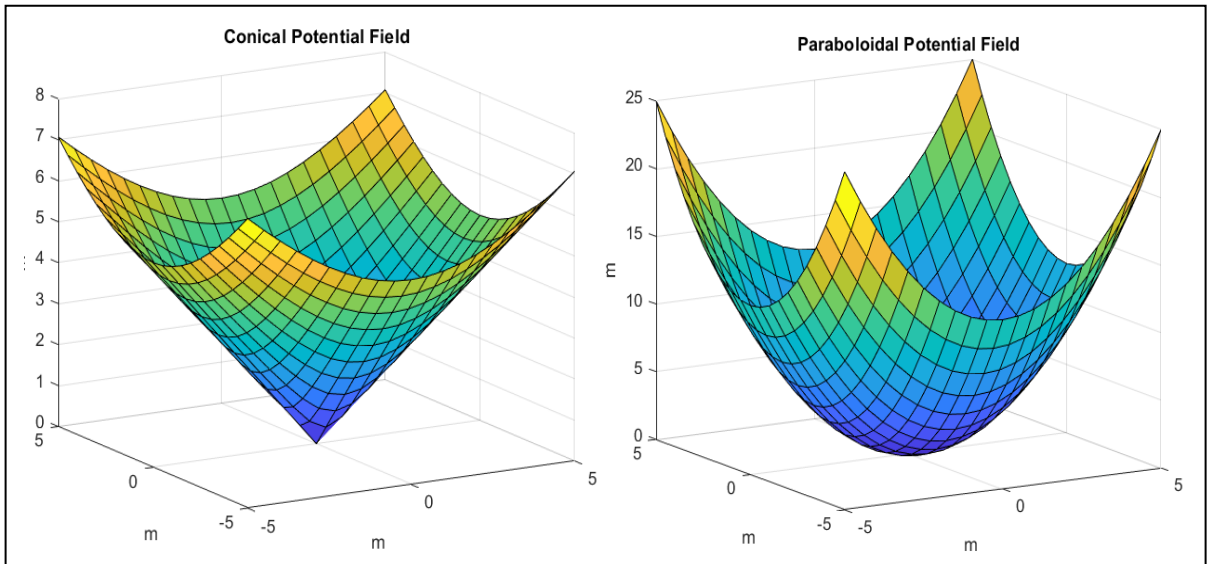

Figure 6. Conical and Paraboloidal Artificial Potential Fields

We define our attractive potential field using the following piecewise function [23].

$$U_a(x,y) = \begin{cases} 0.5k_1\|e(x,y)\|^2 & \|e(x,y)\| \le \rho \\ k_2\|e(x,y)\| & \|e(x,y)\| > \rho \end{cases}, \tag{54}$$

where, ρ is the threshold distance from the target, k_a and k_b are the gain values, and $e(x,y)$ is the distance error given by.

$$e(x,y) = \begin{bmatrix} x_g - x_c \\ y_g - y_c \end{bmatrix} \tag{55}$$

In order to maintain continuity at ρ, we must satisfy the condition defined in (56)

$$\lim_{\|e(x,y)\| \to \rho^-} U_a(x,y) = \lim_{\|e(x,y)\| \to \rho^+} U_a(x,y) \tag{56}$$

This results in equation (57) which describes a relationship that must be maintained.

$$k_1 = 0.5\rho k_2 \tag{57}$$

Here, we take the negative gradient of the attractive potential field, to result in the vector field [24]–[26]

$$F_a(x,y) = -\nabla U_a(x,y) = \begin{cases} k_1 e(x,y) & \|e(x,y)\| \le \rho \\ k_2 \frac{e(x,y)}{\|e(x,y)\|}, & \|e(x,y)\| > \rho \end{cases} \tag{58}$$

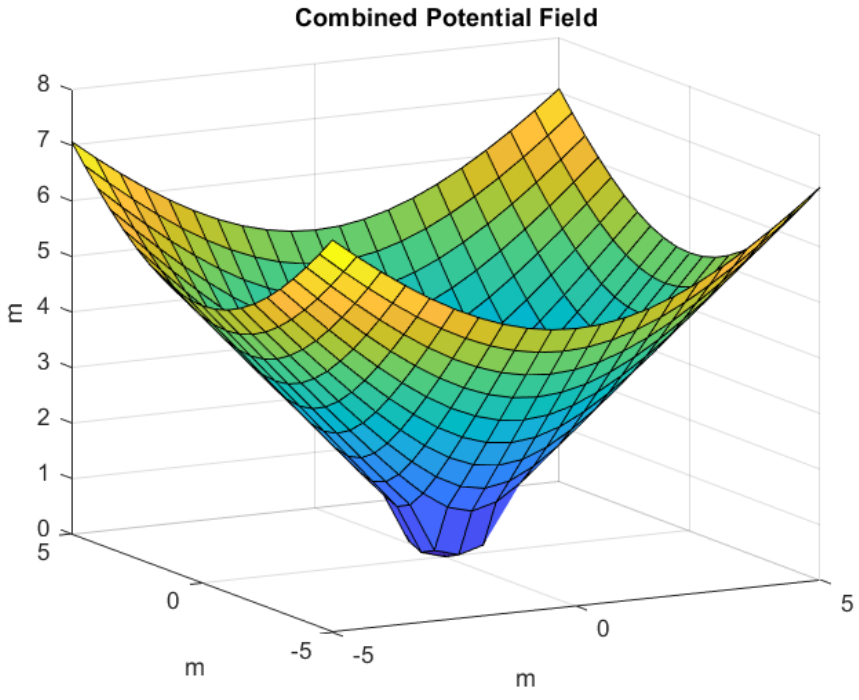

Figure 7. Combined Potential Field

The attractive vector field is interpreted as the Chassis velocity vector with respect to the global coordinate frame. It is much easier to directly control velocity rather than to directly control position, using a PID controller. This is because if there is a large distance error, the controller will use a very large input to drive the error to zero.

$$F_a(q) = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{59}$$

With respect to the chassis frame, the Chassis set-point velocity is given as

$$^{C}v_x = \|F_a(x,y)\|_2 \tag{60}$$

Figure 7 shows the combination of conical and paraboloidal Artificial Potential Fields, with continuity maintained at $e(x,y) = \rho$. The set point yaw angle is given by the following equation [27].

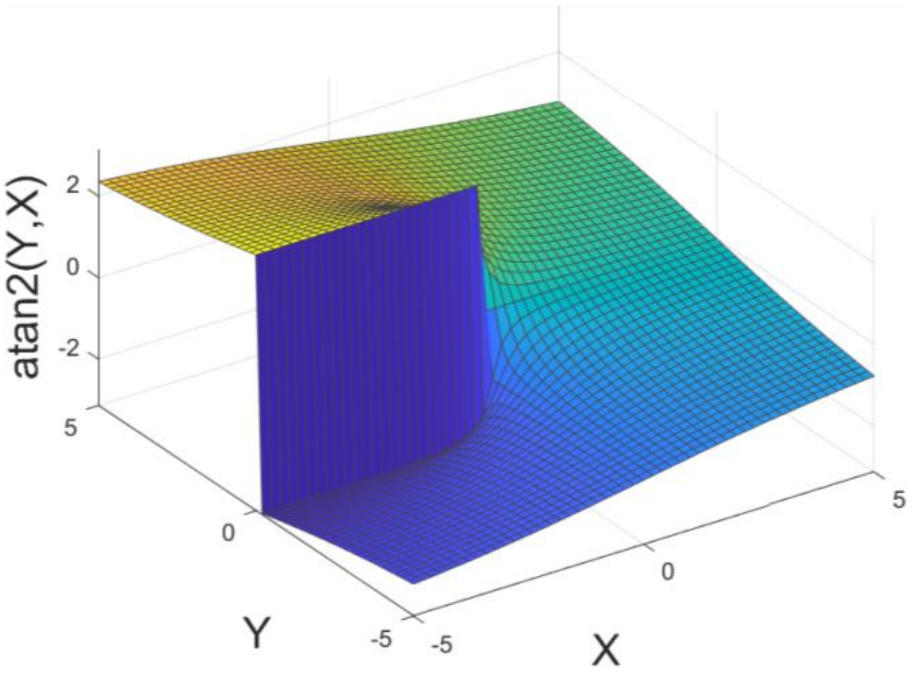

Figure 8. atan2(Y, X) Surface Map

$$\theta = \text{atan2}(v_y, v_x) \tag{61}$$

However, the function atan2(y, x) has a discontinuity occurring at the interface of the II and III quadrants, or $\theta = \pm\pi$. If the robot is pursuing a moving target that requires it to turn more than $\pm\pi$ radians, we must implement a conditional statement that "winds up" the desired yaw angle to more than $\pm\pi$ radians. Figure 8 shows the mapping of ATAN2(Y, X) and its discontinuity at $(X, Y) = (-|x|, 0)$. The windup condition relies on the direction of the angular velocity to determine whether we add or subtract π from our θ_{des}. For this condition, a step delay is used to find the direction of the desired angular velocity.

$$\theta_d(\Delta t) = \theta_d(t_{n+1}) - \theta_d(t_n) \tag{62}$$

t_n is the time value at step n. At the discontinuity, $\theta_{des} = N\pi$, where $N \in \mathbb{Z}$, the following condition will be invoked.

$$\theta_{des} = \begin{cases} \theta_d + 2\pi \\ \theta_d - 2\pi \end{cases} \quad if \quad \begin{matrix} \theta_d(\Delta t) \geq 0 \\ \theta_d(\Delta t) < 0 \end{matrix} \tag{63}$$

This method of using APF to determine velocity set-points has drawbacks. For one thing, if the robot is pursuing a moving target with the goal of a "soft-landing" the robot will never reach the target and, instead, chase the target until the end of the simulation. To prevent this, one could choose to only use the conical potential field, which if the target is moving slower than the TWBR, then the TWBR would reach the target with a "hard-landing". The chosen parameters for the Artificial Potential Field Algorithms are given in table 2.

Table 2. APF Parameters

ρ_x	0.5 m
$(k_1, k_2)_{\dot{x}}$	$(3, 1.5)$
ρ_θ	$5\pi/180$ rad
$(k_1, k_2)_{\dot{\theta}}$	$(9, \quad \pi/4)$

The threshold distance from the target, ρ_x, is set to be 0.5 m. This is so that the controller has enough distance/time to slow down when it crosses the distance threshold. The threshold angular distance is much smaller, $\rho_\theta = \frac{5\pi}{180}$. This is because the heading angle needs to be precise. If the yaw error doesn't close fast enough, the TWBR may travel past the target. The k_b values can be interpreted as the maximum desired speed. So, for velocity, the maximum is $1.5\,\frac{m}{s}$, and for yaw rate, the maximum is $\frac{\pi}{4}\,\frac{rad}{s}$, or $45\,\frac{deg}{s}$.

CHAPTER 4: PID CONTROL

Subscripts

- P Proportional term
- I Integral term
- D Derivative term

4.1 PID System Architecture

The PID controller is one of the most widely used controllers in linear systems, it is based on the concept of feedback control. PID is a such a popular technique due to its simple design and implementation for countless applications [28]. The simplicity of design comes from the need to tune only three parameters. The controller utilizes feedback error, also known as the difference between the set point and the measure process variable, to determine the value for the control input. There are many methods to tune a PID controller for linear systems such as the Root Locus method, where gain values can be chosen to meet stability criterion. However, these methods cannot be applied to nonlinear systems (such as the TWBR) because the design process requires the analysis to be done in the 's' domain. For this reason, the PID controller gains for this research are found through trial and error. The PID controller uses three different control actions to adjust the control input, these are the Proportional, Integral, and Derivative control actions, whose gains are denoted by k_p, k_i, and k_d, respectively. The Proportional control action is based on the current error it aims to reduce by applying a control input that is proportional to the error. However, if the proportional gain is set to be too large, the system will begin to oscillate. To prevent this, an integral control can be included. The Integral control action is based on the accumulated error over time, it aims to eliminate any residual error that remains after the proportional control action has been applied. The problem with adding integral control is that it increases the percent overshoot, but some overshoot is to be expected and can be tolerated. The Derivative control action is based on the rate

of change of the error and aims to anticipate future errors and takes preventative action. Adding derivative control will help the system error converge faster, but it also introduces significant noise to the system. Each of these gain values determine the strength of their respective control action, however, care must be taken so that the negative tradeoffs of increasing these gain values does not cause instability to the system [29] – [30].

The control signal u(t) is the output of the PID controller, it is a combination of the three control actions. The error is the difference between the set point and the measured process variable, and the PID controller uses the error, e(t), along with the gains k_p, k_i, and k_d to determine the control signal. The PID controller, equation (64) continuously adjusts the control signal in response to the error until the error is minimized, and the process variable reaches the set point.

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{64}$$

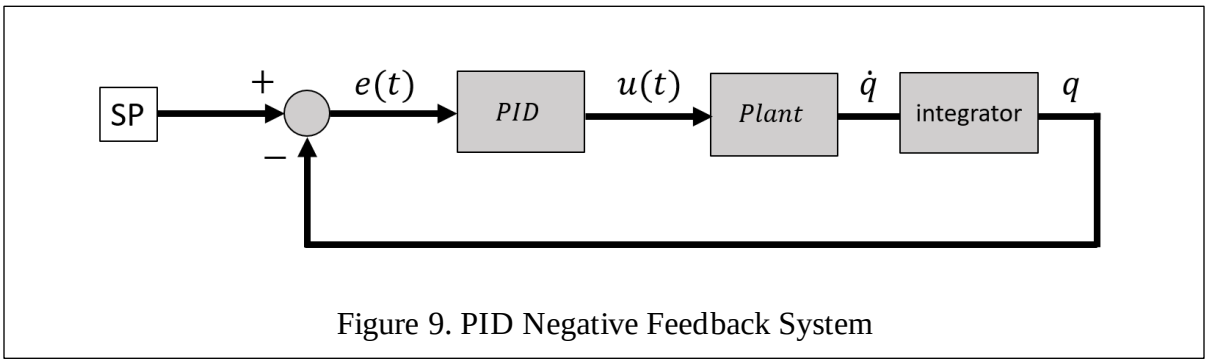

Figure 9. PID Negative Feedback System

Figure 9 shows a basic negative feedback control loop using a PID controller for a single-input single output (SIMO) system. Using this, we can create a block diagram representing a multi-input multi output (MIMO) system, which has the three control variables.

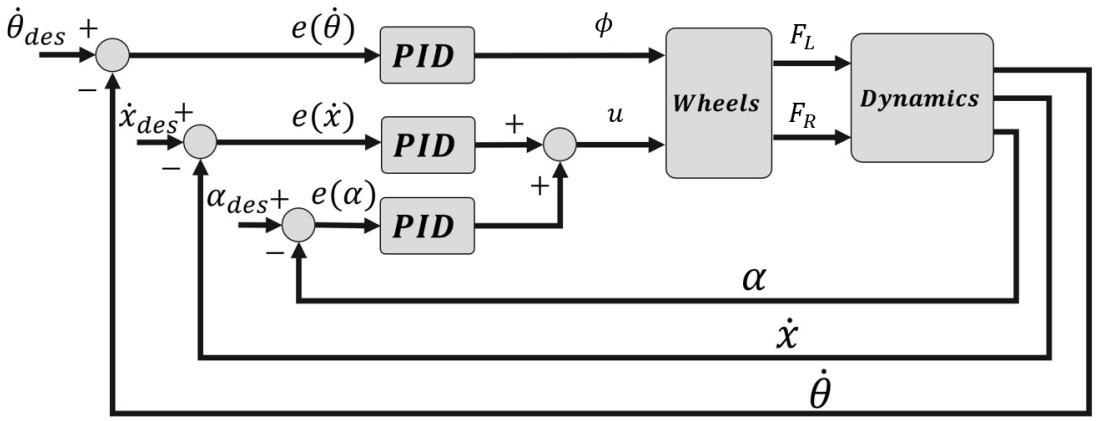

Figure 10. TWBR PID System Architecture

Figure 10 shows the control architecture in block diagram form for the three control variables, $\alpha, \dot{x},$ and $\dot{\theta}$, where the set points for $\dot{x}$ and $\dot{\theta}$ are determined by the Artificial Potential Fields.

4.2 PID Tuning

After conducting a series of simulations, the suitable gain values for the three PID controllers were determined as shown in Table 3.

Table 3. PID Gain Values

$(K_P K_I K_D)_\alpha$	(250, 2000, 31.7)
$(K_P K_I K_D)_{\dot{x}}$	(10, 250, 25)
$(K_P K_I K_D)_{\dot{\theta}}$	(5, 1, 1)

The PID controllers were tuned in the order of pitch, velocity, and yaw rate. Pitch is the most crucial variable to control because without stability, the other variables cannot meet their desired values. The PID values for pitch error was found to be 5000, 2000, 31.7, respectively. These values were chosen based on their ability to provide a stable and responsive control of the system. In particular, the high proportional and integral gains for the pitch controller were selected because the pitch angle is a highly sensitive parameter that requires a rapid and precise response to any deviation from the desired value in addition to the ability to quickly reject disturbances. The large

proportional gain allows the system to quickly correct for any error in the pitch angle, while the integral gain provides additional control to ensure that the system remains stable over time. The derivative gain acts like a damping feature on the system to reduce oscillation and overshoot caused by the other gains.

4.3 PID Simulation

To rigorously assess the robustness of the controllers, they were subjected to disturbances and parameter changes, which enables us to identify their strengths and weaknesses. We will conduct a series of simulations across various scenarios and environments to comprehensively evaluate the performance of each controller. These scenarios may encompass changes in the target position, different types and magnitudes of disturbances, and variations in system parameters such as mass. By carefully analyzing the simulation results, we can assess each controller's ability to maintain stability and achieve the desired performance under diverse conditions. This process of subjecting the controllers to different scenarios and disturbances helps to identify their limitations and areas of excellence, ultimately guiding in further optimization and the development of more effective control strategies.

For the TWBR, the highest priority is maintaining stability in its pitch angle. If the pitch angle is not adequately controlled, it can result in a loss of stability for the TWBR, rendering any other tasks it is designed to perform—such as reaching a target or maintaining a certain altitude—unachievable without proper pitch angle control. One of the primary tests of a controller's stability is subjecting the system to an impulse disturbance. An impulse disturbance is a sudden, brief force applied to the system to alter the output. In the context of the TWBR, an example of an impulse disturbance might be rolling over a small rock or a crack in the sidewalk. It is crucial that the controller is capable of withstanding and rejecting these types of disturbances.

In the first test, the impulse disturbance applied to the base of the TWBR is increased from 0 N to 20 N over the course of a 10-second simulation. The impulse alternates direction and occurs every 1 second, and it lasts for 5% of the impulse width. We evaluated the PID control architecture's ability to maintain the robot's stability under impulse disturbances.

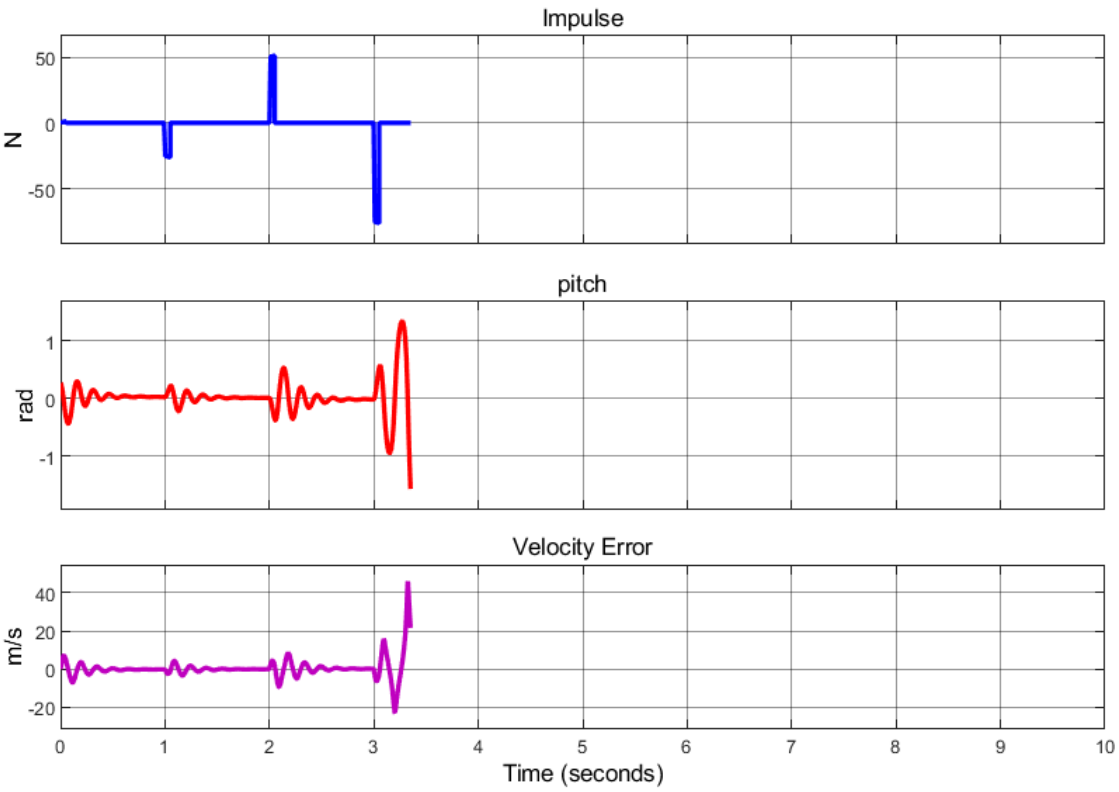

Figure 11. Impulse Disturbances on PID system

As illustrated in Fig. 11, the controller effectively rejected disturbances up to 50N. with both pitch and velocity responses settling within 0.5 seconds. However, some oscillations were observed in the error tracking for both pitch and velocity responses. However, it failed to reject the 75N disturbance at t=3s. Notably, a large oscillation occurred at the beginning of the simulation, prior to the introduction of a major impulse disturbances. This observation suggests that the initial

conditions of the TWBR contributed to the strong controller response. This oscillatory nature of the PID controller was a major cause of the failure of the robot.

The subsequent disturbance introduced to the TWBR is a step disturbance, which more closely resembles the robot encountering a steep incline or a constant gust of wind. This test is more rigorous due to the persistent application of the disturbance. For the step disturbance test, we applied a 5 N force at $t = 1$. As with the impulse disturbance test, we observed oscillations before the disturbance was even applied, attributable to the initial conditions. Upon the application of the step disturbance, minor oscillations were observed; however, the reference variables were rapidly stabilized.

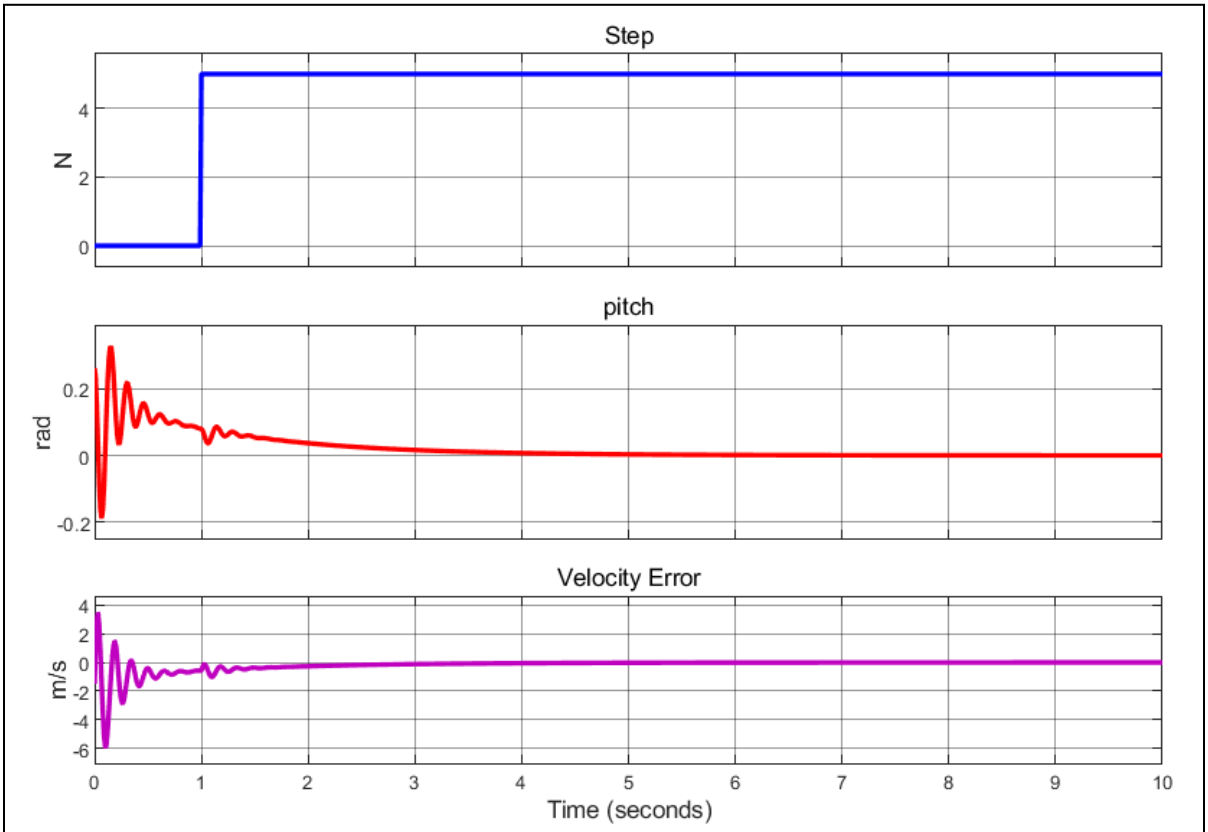

Figure 12. Step Disturbances on PID system

The results of the step disturbance test are presented in Fig. 12, which illustrates the step force, pitch angle, and velocity error. The figure demonstrates that the PID control architecture is effective in maintaining the robot's stability in the presence of a constant external force. Although the initial oscillations were present, the controller quickly adapted to the disturbance and brought the system back to a stable state. The pitch angle and velocity error responses show that the controller managed to minimize the impact of the disturbance and return to the desired reference variables in a short time. This test further validates the robustness of the PID control architecture, indicating its ability to handle a variety of disturbances and maintain stability under challenging conditions. The quick stabilization of the reference variables following the step disturbance application highlights the controller's efficacy in mitigating the effects of external forces and ensuring the TWBR's optimal performance.

In this test, we introduced a ramp disturbance, which increased linearly from 0 to 20 N over a 10-second period, or $F_d = 2t$. This type of disturbance represents a gradual change in the external forces acting on the TWBR, simulating scenarios such as a steadily increasing wind or a gradual slope. As shown in Fig. 13, for this range of disturbance, the controller exhibited no issues in stabilizing the reference variables. As observed in the previous tests, initial oscillations were present due to the initial conditions, but they were quickly mitigated by the controller. No signs of instability or additional oscillations were detected throughout the test, indicating the controller's effectiveness in managing ramp disturbances.

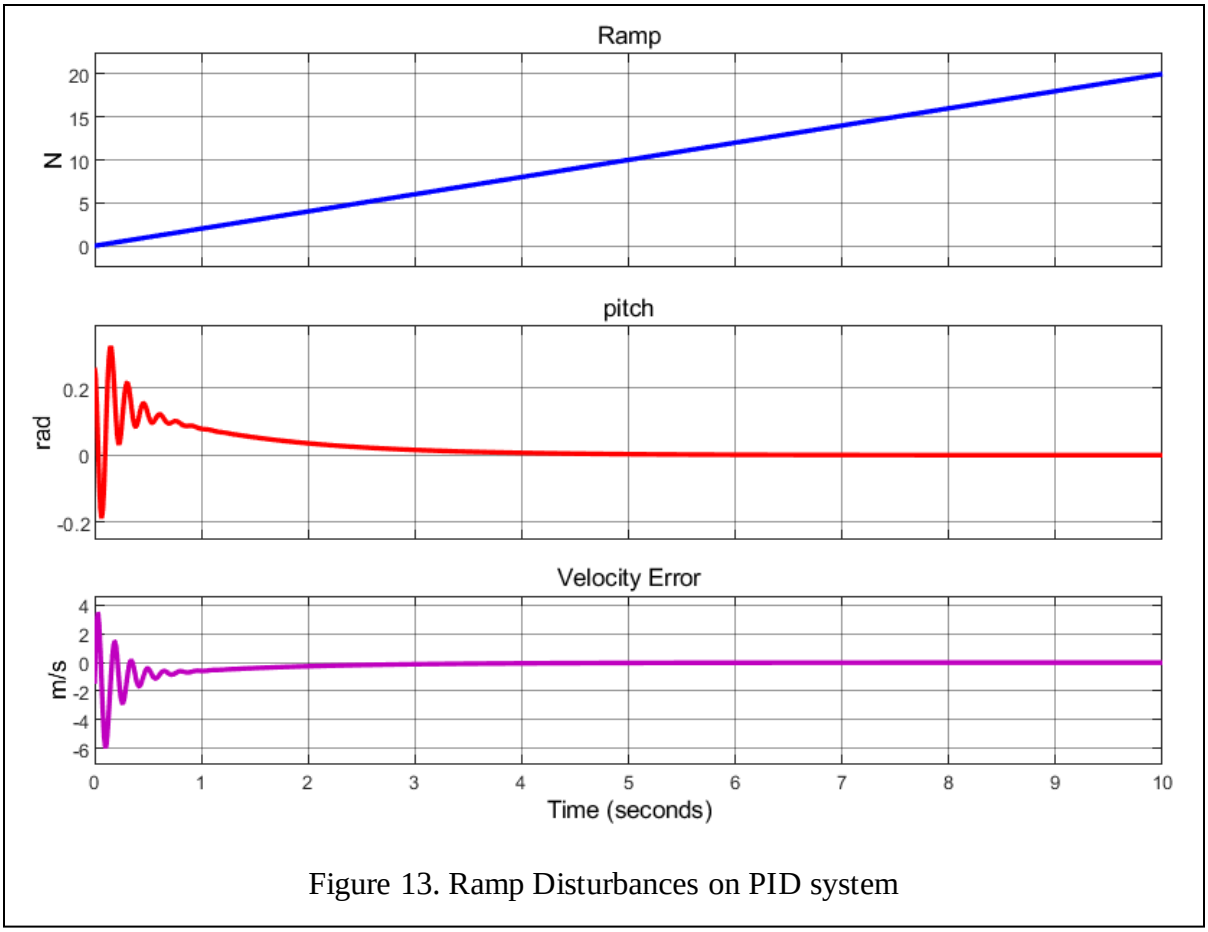

Figure 13. Ramp Disturbances on PID system

The results of the ramp disturbance test further demonstrate the robustness and adaptability of the PID control architecture in maintaining the TWBR's stability under varying types of disturbances. The ability to handle a gradual increase in external forces showcases the controller's capacity to adjust to dynamic environments and ensure optimal performance.

The successful outcomes of the step, and ramp disturbance tests provide strong evidence of the PID control architecture's versatility and effectiveness in maintaining the TWBR's stability under a wide range of disturbances. However, the failure of the controller to reject the impulse disturbances was not acceptable. These tests offer valuable insights into the controller's performance, allowing for future optimization and development of even more effective control strategies for the TWBR.

CHAPTER 5: DEEP Q-LEARNING

Variables

- R Cumulative Reward
- r Immediate reward
- Q Quality Function
- γ Discount Factor
- E Expected
- L Loss
- θ Parameters of Neural Network
- relu Rectified Linear Unit function

Subscripts

- t Time
- s State
- a Action

Superscripts

- ‘ The next time step
- t Time

5.1 Background

Reinforcement learning is a method of machine learning that allows a computer agent to learn how to complete a task by interacting with its environment over the course of many episodes. Unlike supervised learning, where the agent is provided with labeled examples of correct inputs and outputs, the agent in reinforcement learning receives feedback from the environment in the form of rewards or penalties. The environment provides the agent with observations and rewards, and the agent learns by trying different actions and receiving feedback from the environment. The goal of the agent is to maximize its expected reward by making a series of decisions that lead to successful completion of the task. The future rewards are discounted at each time step in order to make current decisions that lead to better future rewards. This cumulative return at each time step, t, is given as follows [31]:

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \tag{65}$$

T is the time step where the episode is terminated, γ is the discount factor, and r_t is the reward at time step, t. The intuition behind discounted future rewards is that current actions are more important than future actions and should thus be valued more. The discount factor, γ, is set to a value in the range $0 \leq \gamma < 1$. This affects how much emphasis is put on expected rewards occurring in the future. When γ is higher, it places more value on actions that lead to long term rewards, and when it is lower, it values actions that lead to short term reward.

DQN requires a discrete action space, due to the nature of having a finite number of outputs in the Deep Neural Network used approximate the expected Q-values of each possible action. The DQN agent is iteratively updated to converge to the optimal action-value function [deep mind paper]. This iteratively determined Q-Value function is found by using the Bellman equation, which is given as follows:

$$Q_t(s,a) \leftarrow Q(s,a) + \alpha\left(R(s,a) + \gamma\, argmax_a Q(s',a) - Q_{t-1}(s,a)\right) \qquad (66)$$

where $Q_t(s,a)$ is the Q-Value for a state-action pair at time step t, α is the learning rate, $R(s,a)$ is the reward for a state-action pair, and $argmax_a Q(s',a)$ is the maximum expected reward for being in the new state s'. $R(s,a) + \gamma\, argmax_a Q(s',a)$ term is called the target network, or Q_{target}, which is used to update the Q-network. The learning rate α is a hyperparameter that affects how much the DQN agent is updated during training.[32] Too high of a learning rate value will cause instability during training and can the agent from converging to an optimal policy. However, having too low of a learning rate value will cause the agent to train more slowly, and possibly converge to a suboptimal policy.

This determined Q-value function is used like a 'label' for which the Artificial Neural Network is trained to estimate based on actions taken. The DQN agent is trained by minimizing the loss

function, which is based on the mean squared error of the expected values, and it is described as follows [16],[31]:

$$L_i(\theta_i) = E_{(s,a,r,s')\sim U(D)}\left[\left(target_{DQN} - Q(s,a,\theta_i)\right)^2\right] \quad (67)$$

Where $(s,a,r,s') \sim U(D)$ denotes that the batch of transitions is chosen uniformly and randomly from the experience pool. Experience replay buffer is a technique that saves experiences or transitions into an experience pool, which is then randomly sampled from and reused to update the network. The number of transitions that are sampled is called the experience buffer length, which is a hyperparameter that is tuned to improve sample efficiency and stability during training. [32]

5.2 Hyperparameter Tuning

Hyperparameter tuning is a critical step in machine learning that involves adjusting the fixed parameters of a model to achieve optimal performance. Hyperparameters play a crucial role in the training process as they affect various aspects of the model, such as training stability, speed, and convergence to an optimal solution. Without proper tuning, issues such as overfitting, underfitting, slow convergence, and training instability can arise, leading to poor model performance. In this thesis, several hyperparameters are explored, including the learning rate, discount factor, number of hidden layers, number of neurons per layer, reward shaping, and observation selection. Each of these hyperparameters can significantly impact the performance of the model and should be carefully tuned to achieve the best results. For example, the learning rate determines how much the model's parameters are adjusted during each iteration of training. A high learning rate may result in instability during training, while a low learning rate may lead to slow convergence and suboptimal performance. Similarly, the discount factor determines how much importance is given to future rewards versus immediate rewards. Setting the discount factor too high can lead to a hasty

policy that only focuses on short-term rewards, while setting it too low can lead to a policy that is too conservative and doesn't explore enough.

5.3 Deep Neural Networks

Deep neural networks (DNN), or simply neural networks, are powerful and flexible machine learning models that can learn highly complex patterns in data. They are based on the structure and function of the biological neural networks in the human brain. Neural networks consist of nodes, also called neurons, which are connected by weighted paths. Each neuron receives one or more input signals, multiplies each input by its corresponding weight, and then sums up the weighted inputs. The resulting sum is then passed through a nonlinear activation function, producing the output of the neuron, as shown in Fig. 14.

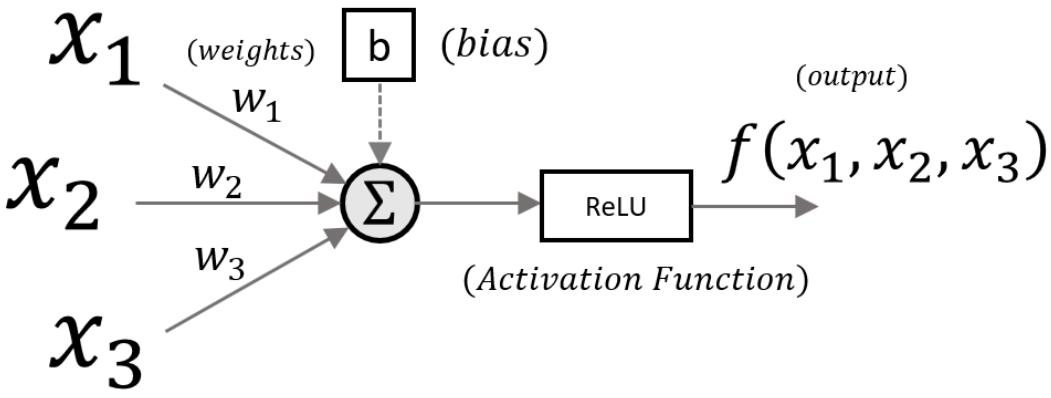

Figure 14. Neuron Function

Examples of activation functions are Sigmoid, Rectified Linear Unit (ReLU), Tahh, Softmax, etc. relu is a standard and often default activation function used for Deep Neural Networks [33]. It is a piecewise function that takes the input from a path and for negative inputs it returns zero and for positive outputs it returns the input. The ReLU Piecewise function can be given using the following form [33].

$$relu(x) = \begin{cases} 0 & x < 0 \\ x & x \geq 0 \end{cases} = \max(0, x) \tag{68}$$

In Fig. 15, we visualize the Rectified Linear Unit (relu) activation function. The blue line in Fig. 15 represents the function output when the input value (x) is less than zero. In this region, the relu function output is zero, effectively thresholding negative inputs. The orange portion of the line represents the function output when x is greater than or equal to zero. In this region, the relu function output is equal to the input value itself. This creates a piecewise-linear activation function that introduces non-linearity in neural network models and enables them to learn complex relationships between inputs and outputs. The relu activation function is widely used in deep neural networks and has been shown to improve the model's training efficiency and overall performance.

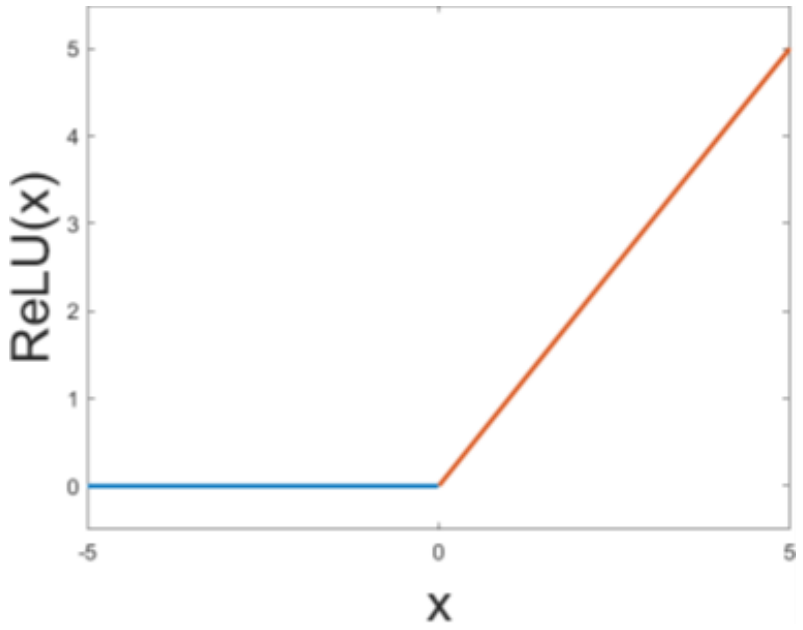

Figure 15. relu Function

The intuition behind introducing an activation function is that it can express nonlinearity to the output of the neuron. Without the activation function, the neural network would be a linear function of the inputs, which severely limit its function approximation capabilities. By introducing activation functions, the neural network can model real-world, complex relationships such as the

dynamics for our TWBR. To visualize how DNN approximate function, we will show examples of approximations using different number of neurons. The function that we are approximating is a parabola $f(x) = x^2$.

Figure 13 shows an approximation using only 2, 3, 4, and 10 neurons in a deep layer (any layers between input and output layers).

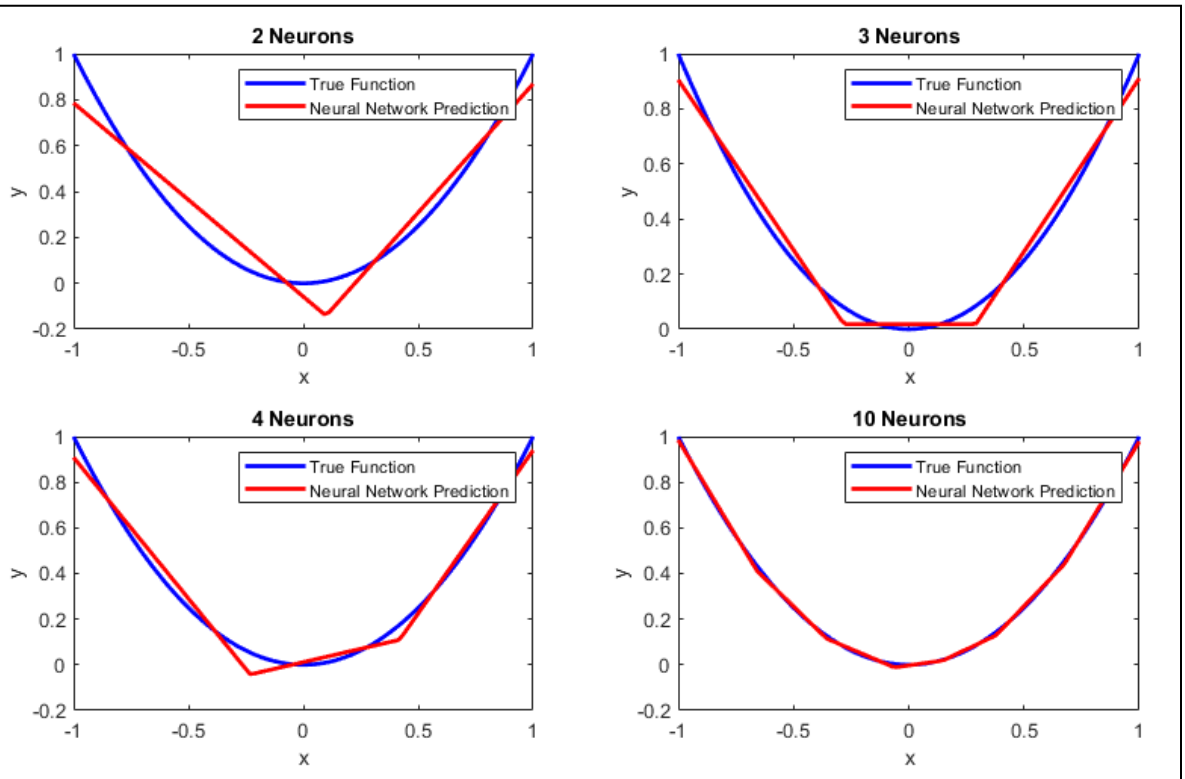

Figure 16. Function approximation using a one-layered Deep Neural Network

We observe in Fig. 16 how the number of neurons in a neural network affects its ability to approximate a given function. Each neuron in the network corresponds to a basis function or a feature that can be used to approximate the function in different regions of the input space. When we have only a few neurons, the network is limited to using a small number of basis functions to approximate the function, while a larger number of neurons allows for more complex combinations of basis functions [34]. It's important to note that the number of neurons required to accurately

approximate a function depends on the complexity of the function and the amount of noise in the data. In the case shown in Fig. 16, the parabola function can be well approximated with just a few neurons, while in other cases such as the relationship between observations and desirable actions in the TWBR, a more complex function may require many more neurons. Additionally, adding more neurons not only increases the accuracy of the approximation but also increases the capacity of the model to learn more complex features and patterns in the data. However, this comes at the cost of increased computational resources and may require more data for training. By carefully selecting the number of neurons and other hyperparameters, we can balance these factors and obtain the best possible approximation of a given function.

5.4 Reward Function Shaping

Reward shaping is a crucial aspect of training a reinforcement learning (RL) agent to achieve success in a particular task. In RL, the agent learns through trial and error, and the reward signal it receives from the environment plays a vital role in guiding its behavior. The reward function defines the goal of the task, and it is used to evaluate the agent's performance at each step of the learning process. If the reward function is poorly defined, the agent may learn a suboptimal policy that fails to achieve the desired goal. This is because the agent will optimize its behavior to maximize the reward signal it receives, regardless of whether the policy it learns is actually useful for achieving the task at hand. For example, if we reward the TWBR for keeping its pitch at zero, it may learn to spin in circles forever. We need to further define the reward function to reflect what successful completion of a task is. Tightly defining the reward function is also useful for faster training. For instance, we can reward stability of the TWBR by giving a reward for every time step the agent survives, or we can also penalize the agent at every timestep for the magnitude of pitch error it has. The agent may eventually figure out through the first method that it survives better

when its pitch is zero, but defining the stable operating point through error penalties will help the robot converge to the solution faster. On the other hand, if the reward function is too tightly bounded, the agent may be constrained in exploring its environment, and it may fail to discover optimal policies. In such cases, the agent may learn a policy that exploits the limited environment it is exposed to, leading to suboptimal performance. Therefore, it is essential to carefully design the reward function such that it balances between providing sufficient guidance for the agent to learn the optimal policy, while allowing the agent to explore the environment to discover better policies.

For the TWBR, we started with a reward function, shown in equation 69, that penalizes the agent for the magnitude of error it has at each time-step. There is also an early termination condition that penalizes the agent for exceeding an angle of $\pm 10^o$. This is meant to expedite the training process by not wasting time on a failed simulation.

$$r_t = \gamma^t \left(-0.4\alpha^2 - 0.2e_{\dot{x}}^2 - 0.2e_{\dot{\theta}}^2 - 10(flag)\right) \tag{69}$$

The early termination condition is $flag$ which given the value of 1 when the pitch exceeds the bounds and a value of 0 otherwise. The intuition behind this is that the agent will minimize the error by driving the error to zero. However, due to the early stopping condition, the agent learned to intentionally terminate each episode as quickly as possible. It did this because the way the reward function is defined, the agent only accumulates penalties, and there was no positive reward for surviving. So, while the agent can minimize its penalty at each time step by minimizing error, it found that it was better to quickly terminate the episode despite the termination penalty. To overcome this, it was decided that the agent needed to generally receive some positive reward at every timestep. To do this, it needs to be ensured that the sum of the penalties at each time step are smaller than the constant gained for surviving at each time step. Therefore, it is important that the

errors are normalized, and weighted appropriately. Equation 70 shows the decided reward function.

$$r_t = \gamma^t \left(1 - 0.4 \left(\frac{\alpha}{0.3491\ rad} \right)^2 - 0.2 \left(\frac{e_{\dot{x}}}{1.5\frac{m}{s}} \right)^2 - 0.2 \left(\frac{e_{\dot{\theta}}}{\frac{\pi rad}{4\ s}} \right)^2 - 10(flag) \right) \qquad (70)$$

The constants dividing the reward observations are essentially normalizing the values between -1 and 1. This ensures that each of these three terms are less or equal to 1. The gain values in front of these terms are their weights that emphasis the importance of each term and ensure that their sum does not exceed -1. This makes it so the reward is always some positive value, but it is maximized by minimizing the error. Therefore, the TWBR is incentivized to make decisions that prolong its survival in the environment while driving the errors to zero.

5.5 Training the DQN Agent

In this section, we will discuss the observations and analysis of the training and performance of the DQN agent for the two-wheel balancing robot. The agent was trained with a step size of $\Delta t = 0.02\ seconds$, allowing for a fine control of the robot's movements. It was noticed over many training attempts that too small of a step size led too poorer training results. This is likely since longer step sizes display larger stronger between actions and resulting states, allowing the agent to more appropriately learn how to interact with the environment. However, too large of a step size, also resulted in poor results due to discrete actions having a large effect on the state. Over the course of training, the agent's performance was evaluated in 20-second episodes, with a maximum cumulative reward of $R_T = 1000$.

To ensure that the agent was adequately trained, a training criterion was set such that the agent must achieve a 3-episode average reward of 850. This criterion was designed to prevent an undertrained agent with a suboptimal policy from prematurely exiting the training phase due to an

isolated exceptional performance, and it is shown in Fig. 17 by the dark blue line. The light blue line represents the individual episodic reward. The agent was trained to pursue a stationary target. These targets were randomly generated to lie anywhere on a circle with a radius of 10, centered on the TWBR's initial position. This randomness is meant to prevent the agent from overfitting and being more generalized for autonomously navigating to any point it could be given.

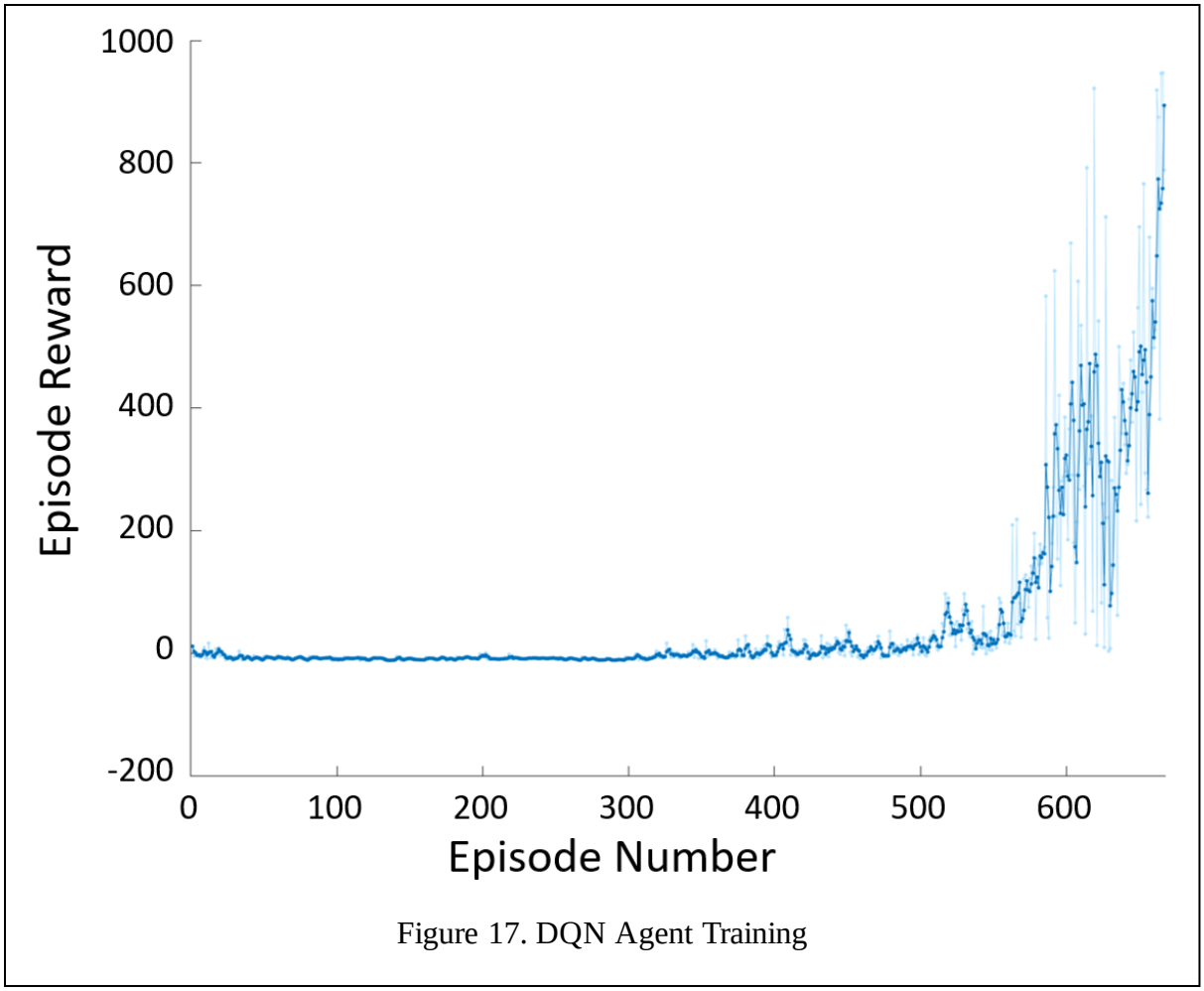

Figure 17. DQN Agent Training

We see from training that the agent doesn't see any noticeable progress until around episode 320, and the agent starts seeing large rewards at around episode 550. In Fig. 18, we observe how the DQN agent performs in a 20 second simulation with the task of navigating to a stationary target. We see the plots of pitch, velocity error, and yaw rate error.

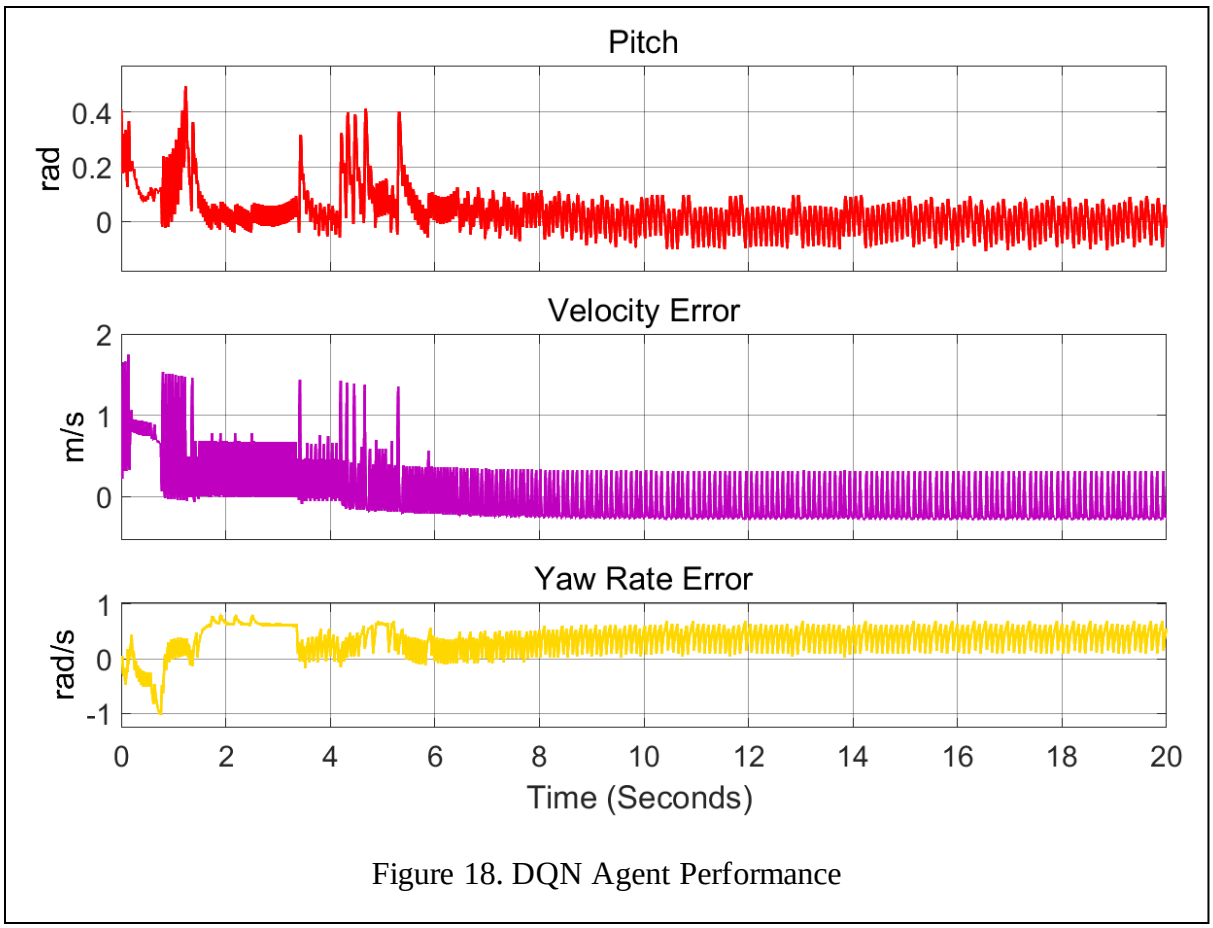

Figure 18. DQN Agent Performance

The maximum pitch error reached approximately 0.5 radians, which is acceptable as it quickly converges back to zero. This demonstrates the agent's effectiveness in maintaining the robot's pitch stability. Regarding the velocity error, it converges to zero, indicating that the agent is successful in achieving the target velocity. However, due to the nature of the discrete action space, the error exhibits choppiness, fluctuating between -0.5 m/s and 0.5 m/s. Despite this, the overall performance remains satisfactory, as velocity is the least critical control variable among the three. The yaw rate error does not converge to zero, which is a big issue for navigation. Instead, it maintains a steady-state error of approximately 0.33 rad/s. This level of error is not acceptable, as accurate yaw tracking is essential for proper navigation.

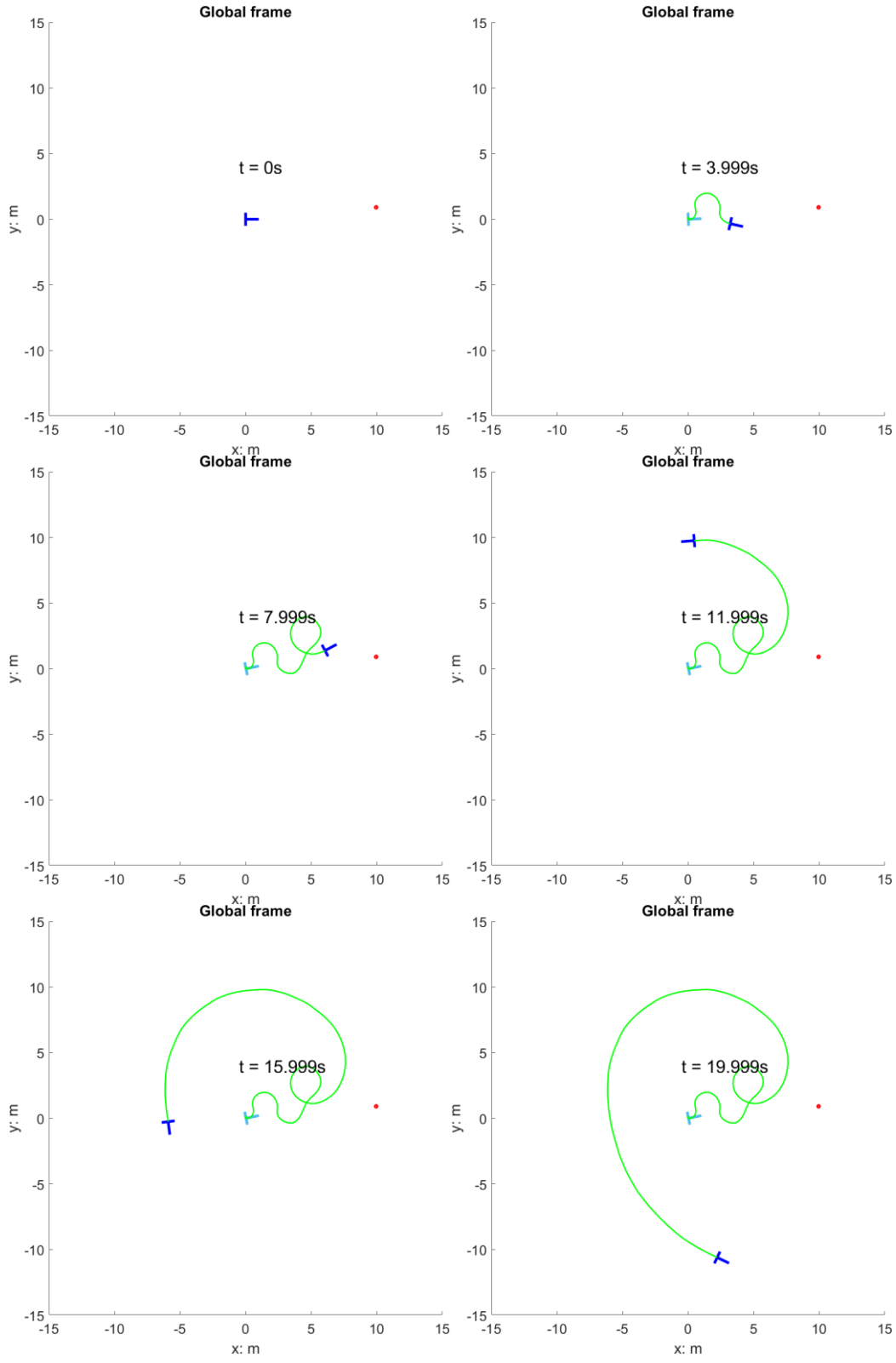

Figure 19. DQN Moving Target Pursuit

The impact of this error can be more easily visualized in Fig. 19, which displays six frames of the robot navigating within the global XY frame. The two-wheel balancing robot (TWBR) exhibits a wandering behavior and struggles to navigate towards the target. This analysis suggests that further refinements to the DQN agent's training process and exploration of additional techniques, such as continuous control methods, may be necessary to enhance the agent's ability to minimize yaw rate errors and improve overall navigation performance.

Having analyzed the performance of the DQN agent in attempting to navigate the two-wheel balancing robot under regular conditions, we will now transition to examining its stability under various disturbances. This stability analysis is critical to assess the agent's robustness and resilience when subjected to unforeseen or challenging scenarios in real-world applications. To evaluate the DQN agent's ability to maintain stability, we will subject it to impulse, step, and ramp disturbances. These disturbances represent a diverse range of challenges that the agent may encounter in practice, and their inclusion in the analysis provides a comprehensive assessment of the agent's performance under different conditions.

First, the DQN agent's ability to reject disturbances was evaluated by subjecting it to impulse disturbances. Specifically, the impulses alternate and increase from 0 to 220N and have an impulse width of 5% of the period, which is set to 1 second. Figure 20 illustrates the response of the pitch and velocity following each impulse disturbance applied to the DQN agent. The agent effectively stabilized pitch in response to the first three impulses, demonstrating its robustness in handling disturbances up to 75N. However, the velocity did not converge and the agent failed at $t = 4s$, where an impulse of 100N was applied

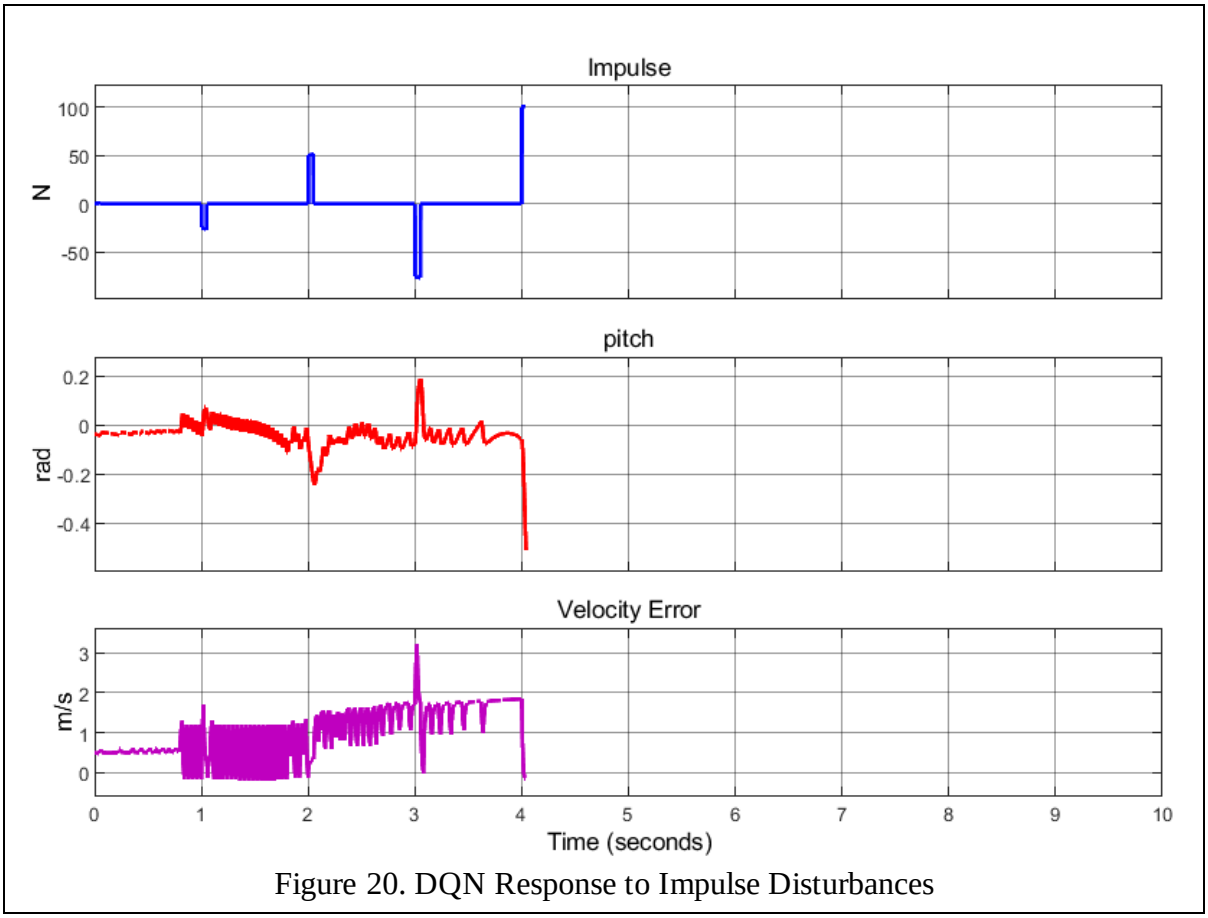

Figure 20. DQN Response to Impulse Disturbances

The outcomes of the impulse disturbance tests highlight the necessity to enhance the DQN agent's effectiveness in rejecting impulse disturbances and maintaining the robot's stability. While the agent demonstrates success in stabilization for disturbances lower than 100N, there is room for improvement in its velocity tracking capabilities. To address this issue, further refinements to the agent's training process may be required, potentially incorporating a more diverse set of disturbances and focusing on velocity tracking during training. By doing so, the agent can become more robust and adaptable, providing a more reliable and effective control solution for the two-wheel balancing robot in various challenging situations.

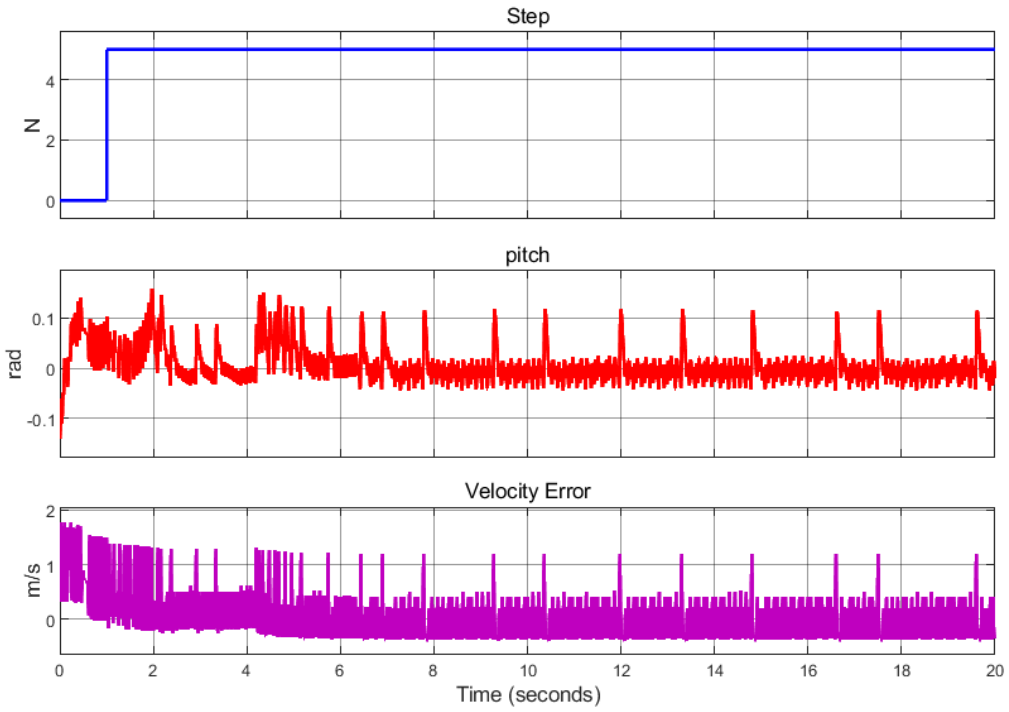

Figure 21. DQN Response to 5N Step Disturbance

The next disturbance is the 5N step disturbance, as shown in Fig. 21. When subjected to a 5N step disturbance beginning at $t = 1s$, the DQN agent is able to maintain the robot's stability, although frequent spiking is observed in both the pitch and velocity. These spikes slightly exceed 0.1 rad and are quickly compensated for, ensuring the robot's stability throughout the disturbance. This observed behavior is likely a result of the agent having been trained primarily with impulse disturbances, making it respond to any disturbance in the same manner it would to a quick, short disturbance. To improve the agent's performance under step disturbances, it is recommended to expose it to various types of disturbances during the training process. By incorporating a diverse range of disturbances, the agent will become more versatile and adaptive, enhancing its ability to handle real-world challenges. Despite the observed spiking, the agent successfully maintains pitch

and survives the entire 20-second period, demonstrating its resilience under the influence of the step disturbance.

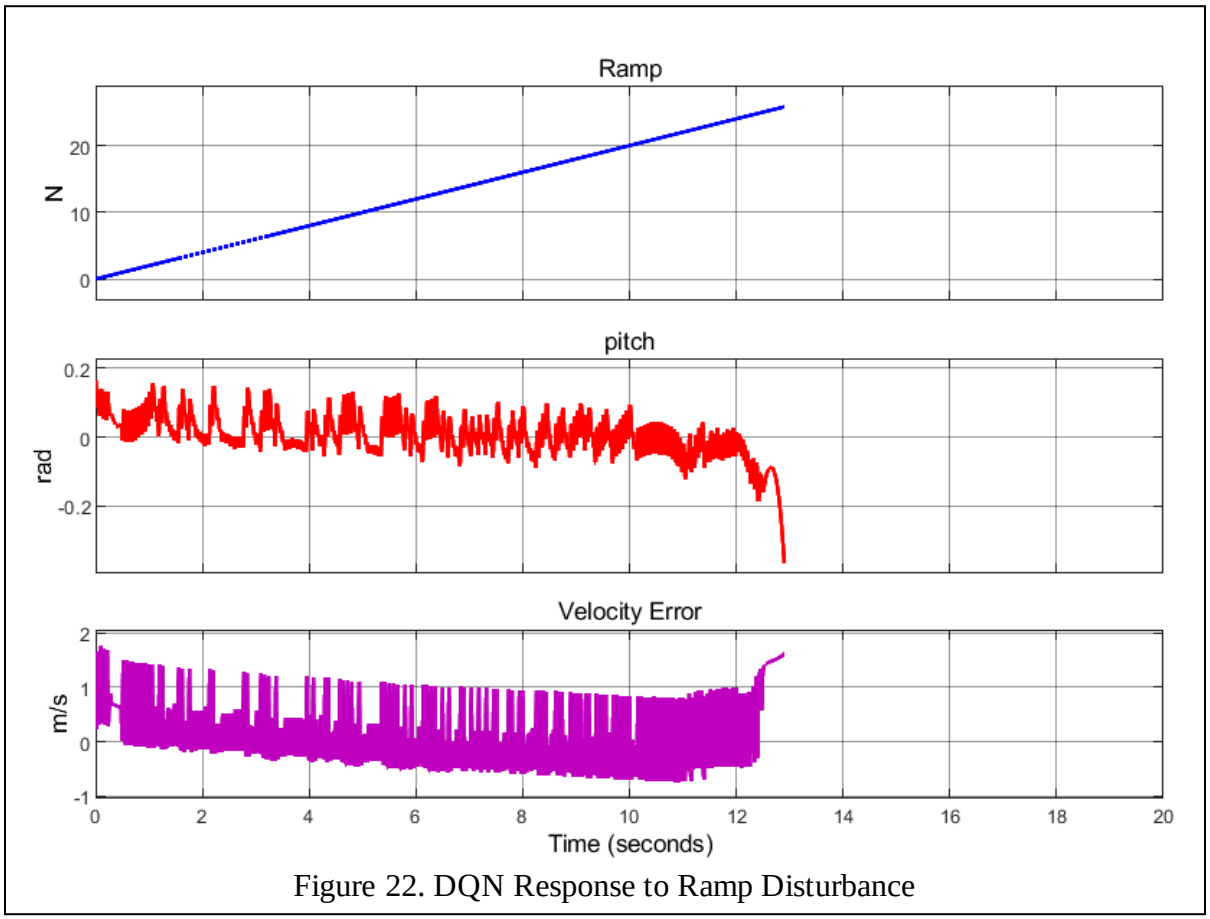

Figure 22. DQN Response to Ramp Disturbance

Next, we further investigate the stability of the DQN agent by subjecting it to a ramp disturbance, in Fig. 22. The agent successfully maintains stability until $t = 12s$ when the disturbance reaches about 20 N. This shows that the agent has difficulties compensating for this ramped disturbance, and it indicates a need to include more diverse disturbances during training.

While the agent demonstrates success in stabilizing pitch and achieving the target velocity, it struggles to accurately track the yaw rate, which is crucial for effective navigation. To address the issue from the failed yaw rate tracking, further refinements to the DQN agent's training process may be required, or alternative control methods could be explored to enhance the agent's ability to

minimize yaw rate errors and improve overall navigation performance. This analysis highlights the need for additional improvements and exploration of alternative techniques to enhance the agent's performance in all three control variables.

Variables

- f Exact Dynamics
- b Exact Control Gain
- $\hat{f}$ Estimated Dynamics
- $\hat{b}$ Estimated Control Gain
- F Dynamics Bounding function
- V Lyapunov Candidate
- β Control gain margin
- λ Convergence speed parameter
- k Switching Control Law gain
- s Sliding surface
- η Eta parameter
- ∞ Infinity

Subscripts

- min Minimum
- max Maximum
- eq Equivalent
- sw Switching

6.1 First Order Sliding Mode Control

Sliding Mode Control is a nonlinear, robust control method that is designed to provide stable and reliable performance in the presence of modeling inaccuracies, parameter variations, disturbances, and other uncertainties. The driving idea behind sliding mode control is that it simplifies the control problem of tracking errors in second or higher order systems by transforming it into a more manageable stability problem for a first-order system [2]. By designing a sliding surface that separates the system into two regions, the controller drives the system towards the surface, and

maintains it on the surface by switching between control modes. This approach provides robustness against modeling uncertainties and disturbances, making it a popular control strategy in various fields such as robotics, power electronics, autonomous vehicles, and aerospace. Sliding mode control is a powerful control method, but it comes with a tradeoff: near perfect performance at the expense of a large control effort. This is largely due to a phenomenon in Sliding Mode Control referred to as chattering. Chattering is characterized as rapid oscillations of the system output and the sliding surface around the target value or equilibrium point. This chattering means that the system is constantly using a large amount of control even when operating at the stable point. While chattering can be undesirable due to its impact on system stability and performance, some amount of chattering can be tolerated and may even be necessary to achieve good tracking performance in some systems.

In order to begin the design of a Sliding Mode controller, we must represent our 2nd order system as it is in equation (71) [2-5].

$$\ddot{x} = f + bu \tag{71}$$

The terms f is the system dynamics and b is the control input gain. The system dynamics are not precisely known; however, it is of known bounds. The estimated system is given by equation (72), with bounds as defined in equation (73) [35]–[41].

$$\ddot{x} = \hat{f} + \hat{b}u \tag{72}$$

$$\left|f - \hat{f}\right| \leq F \tag{73}$$

The bounds on the control gain are given in equations (74) and (75). The term β, as described in equations (76) and (77) are used for later equations [2-5].

$$0 < b_{min} \leq b \leq b_{max} \tag{74}$$

$$\hat{b} = \sqrt{b_{min} b_{max}} \tag{75}$$

$$\beta = \sqrt{\frac{b_{max}}{b_{min}}} \tag{76}$$

$$\beta^{-1} \leq \frac{\hat{b}}{b} \leq \beta \tag{77}$$

For a 2nd order system such as ours is for the TWBR, we can define the sliding surface using equation (78) [2-5].

$$s = \dot{e} + \lambda e \tag{78}$$

The intuition behind sliding mode is that driving the function to the sliding surface and keeping it there implies that we have driven the error to zero and we are keeping it there. The error is defined here as $e = x - x_d$, which should be noted as the negative of the error defined for our previous PID negative feedback loop. The time constant, λ, is a positive constant that determines the convergence rate of the error. A larger time constant value will cause the controller to force error to decay more rapidly. So now we wish to define a control law that will force the sliding surface to zero. To do this we choose the Lyapunov Candidate shown in equation (79).

$$V = \frac{1}{2}s^2 \tag{79}$$

In order to have asymptotic stability about the isolated equilibrium point $s = 0$, we must ensure the following criteria are met [2]:

(a) $\dot{V} < 0$ for $s \neq 0$

(b) $\lim_{|s|\to\infty} V = \infty$

Condition (b) is satisfied by equation (79). To satisfy condition (a) we need a control law that will satisfy equation (80).

$$\dot{V} = s\dot{s} = -k|s| \tag{80}$$

By dividing by the sliding variable s, we arrive at equation (81).

$$\dot{s} = -ksgn(s) \tag{81}$$

By taking the time derivative of equation (78) and substituting in equation (72) we see that $\dot{s}$ can also be represented as

$$\dot{s} = \ddot{e} + \lambda\dot{e} = \hat{f} + \hat{b}u - \ddot{x}_d + \lambda\dot{e} \tag{82}$$

Solving equations (81) and (82) for the input u gives the following control law shown in equation (83).

$$u = \hat{b}^{-1}\left(\ \ddot{x}_d - \hat{f} + -\lambda\dot{e} - ksign(s)\right) \tag{83}$$

However, we must still find k such that it satisfies condition (a). We begin by revisiting equation (80).

$$\dot{V} = s\dot{s} = s(\ddot{e} + \lambda\dot{e}) = s(\ddot{x} - \ddot{x}_d + \lambda\dot{e}) \tag{84}$$

Then we substitute equation (71) and the definition of u from equation (83) to get equation (85).

$$\dot{V} = s\left(f + b\hat{b}^{-1}\left(\ddot{x}_d - \lambda\dot{e} - \hat{f} - ksign(s)\right) - \ddot{x}_d + \lambda\dot{e}\right) \tag{85}$$

Then we can substitute in the estimated dynamics $\hat{f}$ from equation (72) to form equation (86) [35].

$$\dot{V} = s\left(\left(f - \hat{f}\right) + \left(b\hat{b}^{-1} - 1\right)\hat{u} - b\hat{b}^{-1}ksign(s)\right) \leq |s|\left(\left|f - \hat{f}\right| + \left|b\hat{b}^{-1} - 1\right|\left|u_{eq}\right| - \left|b\hat{b}^{-1}\right|k\right) \tag{86}$$

For s to be asymptotically stable, we must show that the k is negative for all cases except at $s = 0$. This is to satisfy that $\dot{V}$ is, in fact, negative definite and therefore satisfies condition (a). We can do by showing that the terms multiplying $|s|$ in equation (86) is less or equal to some small negative number [2].

$$\left|f - \hat{f}\right| + \left|b\hat{b}^{-1} - 1\right|\left|u_{eq}\right| - \left|b\hat{b}^{-1}\right|k \leq -\eta \tag{87}$$

Which gives us the following definition for k in equation (88) [2]

$$k \geq b\hat{b}^{-1}\left(\left|f - \hat{f}\right| + \eta\right) + \left|b\hat{b}^{-1} - 1\right|\left|\ddot{x}_{des} - \lambda\dot{e} - \hat{f}\right| \geq \beta(F + \eta) + (\beta - 1)\left|\ddot{x}_{des} - \lambda\dot{e} - \hat{f}\right| \tag{88}$$

This value of for k can be a constant, and it completes the requirements for a control law u to ensure that we can drive the sliding variable s to zero, and thus drive the tracking error e to zero and keep it there.

6.2 Sliding Mode Control for the TWBR

The uncertain parameter we wish to design for is the body mass, m. This is because for applications such as personal transportation, warehouse automation, and last-mile delivery, mass is never precisely known. Therefore, the mass and uncertainty it causes in the dynamics can be bound. We chose that the mass minimum and maximum are 1kg and 10kg respectively. We can see that the gain bounds are given in equations (89) and (90).

$$b_{max} = b|_{m_{max}} \tag{89}$$

$$b_{min} = b|_{m_{min}} \tag{90}$$

The bounds on the dynamics are given by equations (91) and (92)

$$f_{max} = f|_{m_{max}} \tag{91}$$

$$f_{min} = f|_{m_{min}} \tag{92}$$

The values for λ and η are chosen to be small constants as they were defined previously.

6.3 Hierarchical Sliding Mode Control

Arranging multiple Sliding Mode controllers in a scheme similar to what we've done with the PID before poses a challenge due to the extreme precision required by SMC. This precision can become problematic in underactuated systems with highly coupled state variables, like the TWBR. For example, changing the robot's velocity in the TWBR requires changing its pitch first. If one SMC is set to keep the pitch at zero, the other SMC responsible for velocity must compete to control its variable, resulting in high instability in the control system.

To address this issue, we can use a controller called "hierarchical sliding mode control". This type of controller extends the sliding mode controller by organizing multiple sliding surfaces into a hierarchical tree or control scheme. This structure enables the controllers to operate with less precision, which is crucial for underactuated systems like the TWBR. There are several versions of Hierarchical Sliding Mode control (HSMC), including Incremental HSMC and Adaptive HSMC. In this context, we will design an Aggregated HSMC.

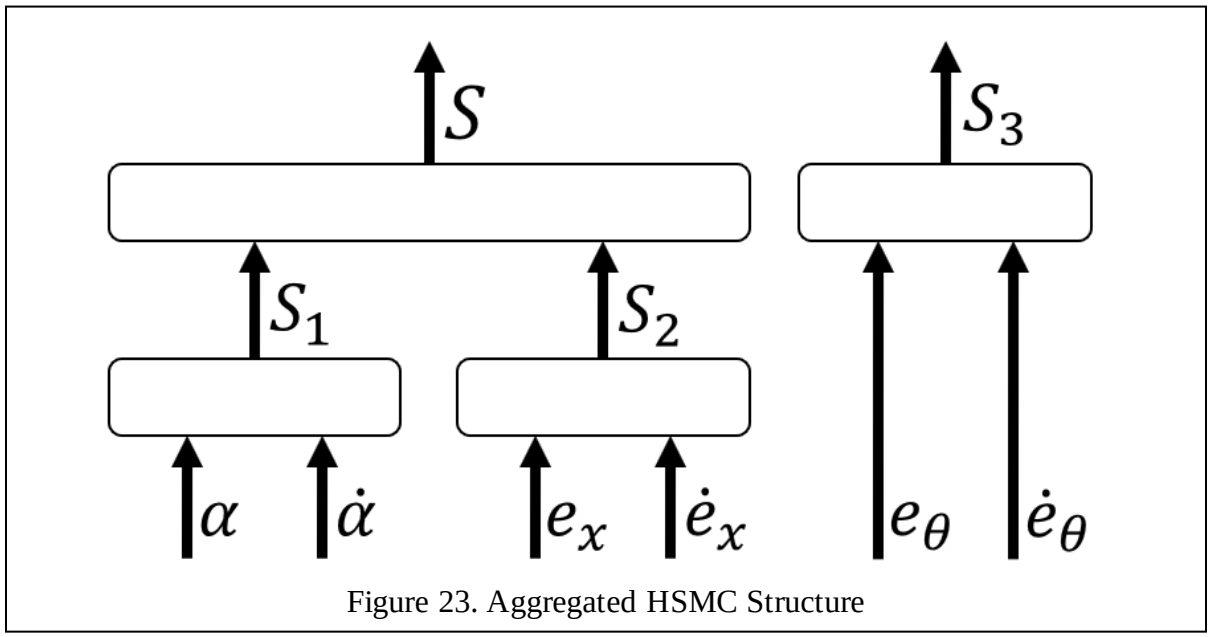

Figure 23. Aggregated HSMC Structure

As shown in Fig. 23, the design of the Aggregated HSMC starts with defining the sliding surfaces for the three variables we wish to control. Those sliding surfaces are fed into a parent sliding surface. We have yaw under a different sliding surface because it is a function of a separate control input. The equations for these low-level sliding surfaces are given by equations (93) through (96) [4 - 9] .

$$s_1 = \lambda_1 \alpha + \dot{\alpha} \tag{93}$$

$$s_2 = \lambda_2 e_x + \dot{e}_x \tag{94}$$

$$s_3 = \lambda_3 e_\theta + \dot{e}_\theta \tag{95}$$

The parent sliding surface that aggregates the sliding surfaces s_1 and s_2 is given by equation 96.

$$S = s_1 a + s_2 \tag{96}$$

The parameter, a, in general, can be a positive or negative parameter. It can be time varying or constant [8]. The aggregated HSMC control law for the input, u, is defined by equation (97), where it is the sum of the equivalent control for the two lower level sliding surfaces given in equations (93) and (94). The control law, ϕ, for the sliding surface s_3 is given in equation (98).

$$u = u_{eq1} + u_{eq2} + u_{sw} \tag{97}$$

$$\phi = \phi_{eq} + \phi_{sw} \tag{98}$$

The equivalent control forces are given by equations (99) through (101) [8].

$$u_{eq1} = \frac{\lambda_1 \dot{\alpha} + f_1}{b_1} \tag{99}$$

$$u_{eq2} = \frac{\lambda_2 \dot{e}_x + f_2}{b_2} \tag{100}$$

$$\phi_{eq} = \frac{-f_3 + \ddot{\theta}_d - \lambda \dot{e}_\theta}{b_3} \tag{101}$$

The switching control law for both inputs are given by equations (102) and (103)

$$u_{sw} = -\frac{a b_1 u_{eq2} + b_2 u_{eq1} + k_1 S + \eta_1 sign(S)}{a b_1 + b_2} \tag{102}$$

$$\phi_{sw} = \frac{-k_2 sign(s_3)}{b_3} \tag{103}$$

6.4 HSMC Simulation

In this section, we introduce the simulation results for the proposed HSMC controller for the TWBR. After fine-tuning the parameters, we conducted a series of simulations to evaluate the controller's performance under various conditions and disturbances. These simulations allowed us to assess the controller's effectiveness in maintaining stability and responsiveness, while also highlighting any potential limitations or areas for improvement. The detailed outcomes of the

simulations, along with relevant performance metrics, are discussed further in the subsequent sections.

Due pitch stability being the most critical function of this system, the parameter tuning process began with a focus on pitch-related parameters. After identifying and stabilizing pitch parameters, we proceeded to refine those parameters influencing velocity. This involved iterative adjustments to optimally balance between the two sets of parameters for the HSMC controller. The resulting calibrated parameters for the TWBR aggregated HSMC are presented in Table 4.

Table 4. HSMC Parameters

λ_1	1.8
λ_2	5.5
λ_3	60
a	2.225
η_1	0.8
η_2	0.1
k_1	-0.3
k_2	2

After initial tuning of the HSMC parameters, we evaluated the HSMC's ability to achieve the desired reference tracking for pitch, velocity, and yaw rate. The first test involved having the HSMC navigate the robot to a stationary target located at $(x, y) = (-10, 10)$. This first test did not subject the system to disturbances, as we will see in further simulations. Figure 24 shows that the HSMC was effective in driving the error to zero within a reasonable amount of time. The robot's pitch remained within 0.04 radians or 2.3 degrees, demonstrating excellent tracking

performance. The velocity error was reduced to zero within four seconds, and the yaw rate error was reduced within two seconds, indicating that the robot maintained a stable heading and velocity tracking. This observation justified the yaw rate error tracking. Further validation of the results can be shown in Fig. 25, where we visualize the TWBR path in the global XY plane. We see that the robot is capable of navigating to the target. However, we also see that the robot takes a curved path. This indicates suboptimal yaw error tracking. We also notice at the end of the simulation, when the TWBR is closer to the target, the yaw starts to diverge. This can be explained by the fact that as the TWBR grows closer to the target, sensitivity of yaw to the input increases. This divergence occurs when the TWBR is less than $0.5\ m$ from the target, which helps validate this occurrence. These results demonstrate the system's fundamental capabilities autonomously navigate in a simulation environment, and further tests will evaluate the system's performance under more challenging conditions and disturbances.

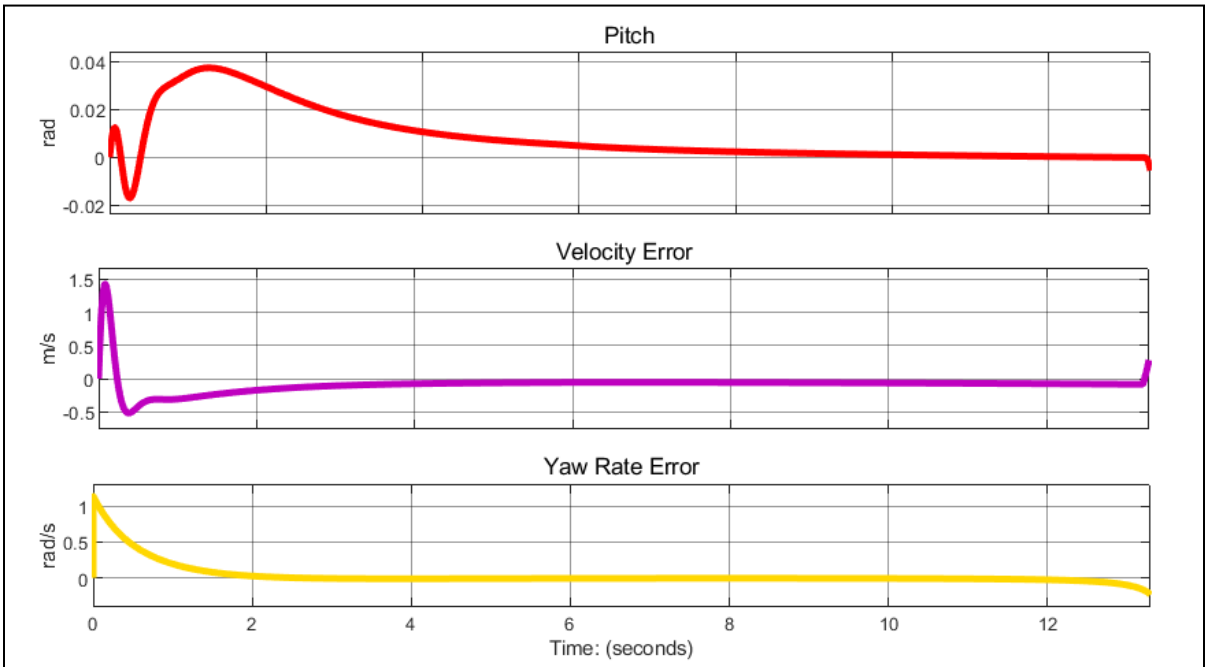

Figure 24. HSMC Undisturbed Reference Tracking

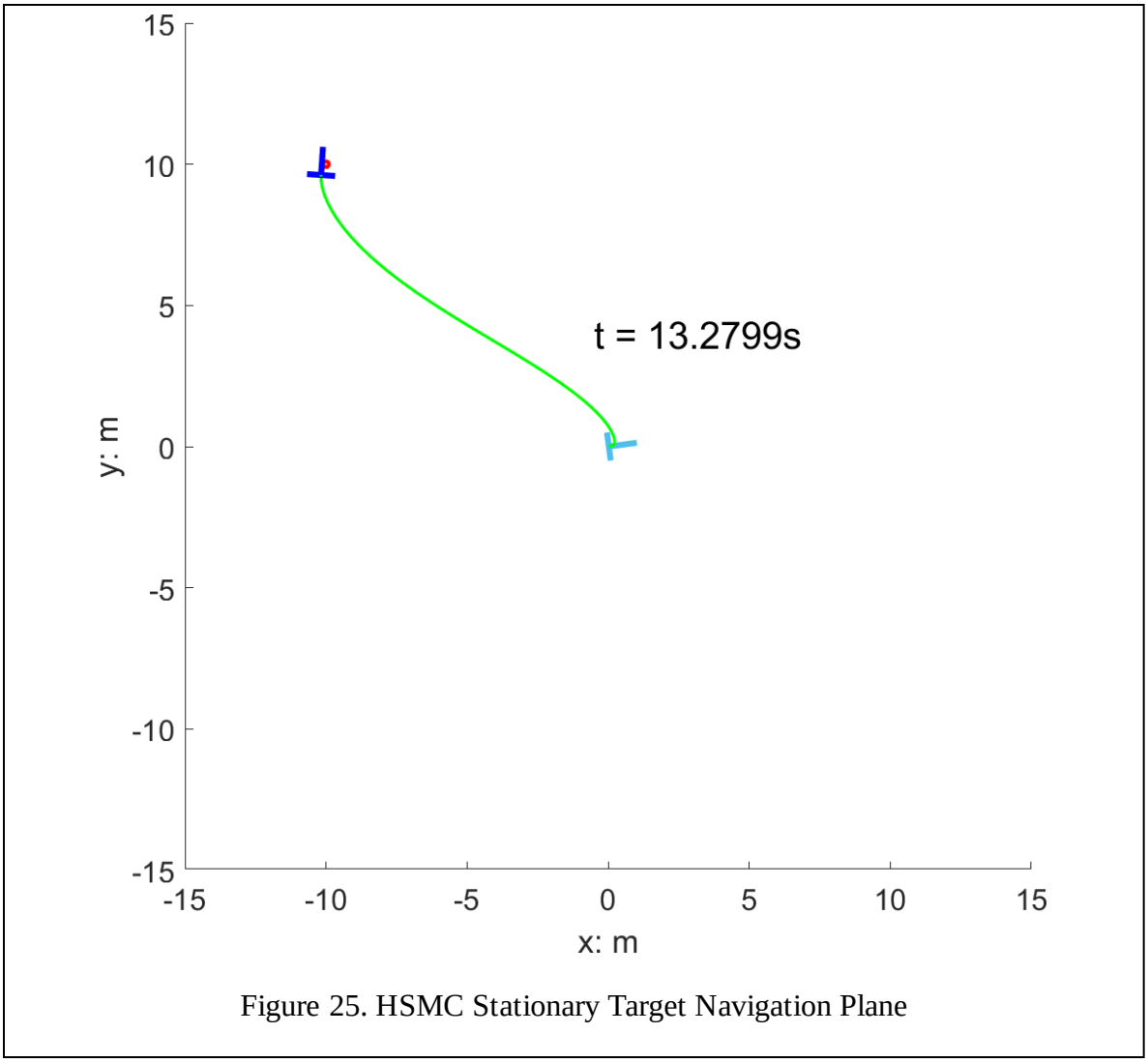

Figure 25. HSMC Stationary Target Navigation Plane

Next, we will evaluate the HSMC's ability to reject disturbances by subjecting it to impulse disturbances. The purpose of this test is to demonstrate the HSMC's ability to maintain stability and pursue the reference point even when the robot is subjected to external disturbances. These impulse disturbances can be likened to the TWBR rolling over a rock, bump, or crack on the ground. Such disturbances can significantly impact the robot's balance and stability, and it is crucial for the controller to react appropriately to maintain the robot's balance. Specifically, the impulses will be alternating directions and increase in magnitude from 0 to 220N with an impulse width of 5% of the period, which is set to 1 second.

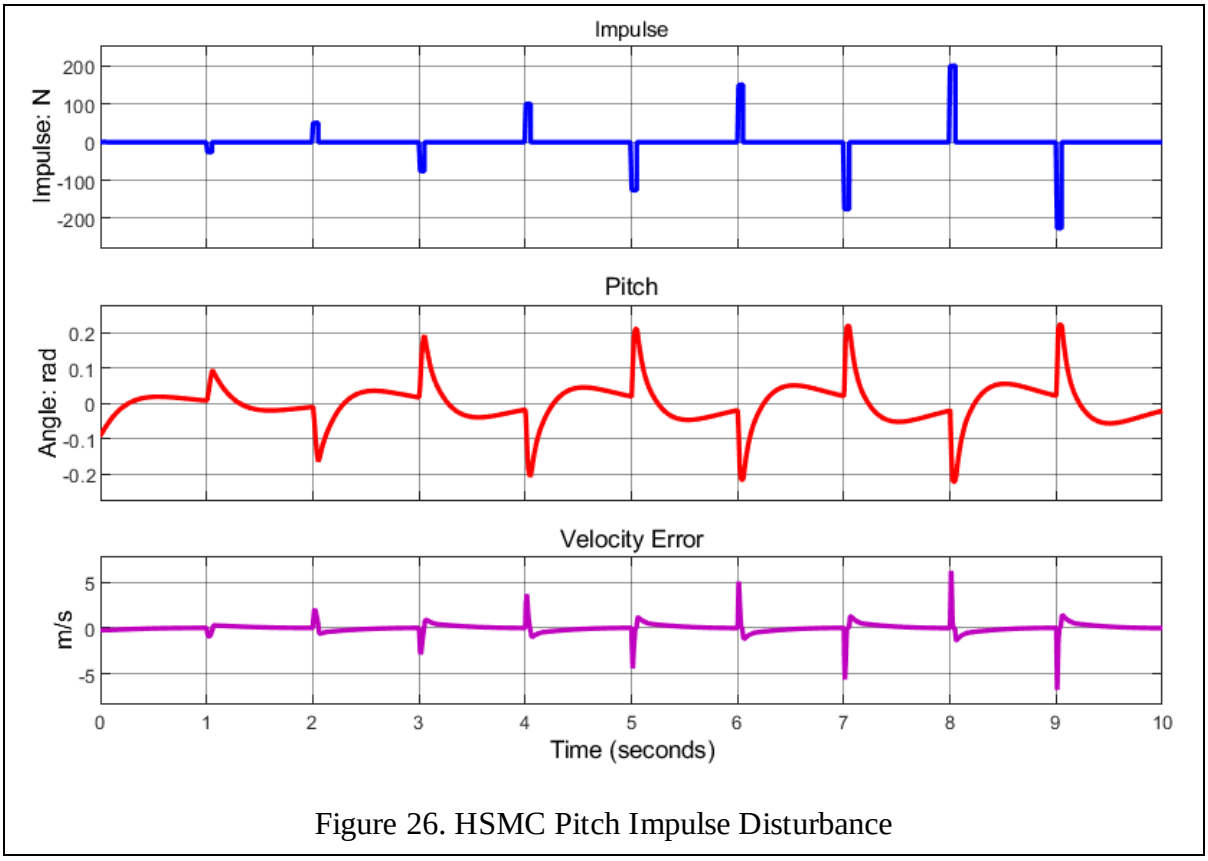

Figure 26. HSMC Pitch Impulse Disturbance

Figure 26 shows the response of the pitch and velocity after every impulse disturbance. The HSMC was effective in stabilizing both pitch and velocity with minimal oscillations in response to the impulses. The maximum pitch error observed was 0.6 radians or 34 degrees, which occurred at t=8 seconds in response to the maximum impulse disturbance. The pitch angle consistently settled within 1 second of the disturbance, indicating the controller's ability to detect and respond to the disturbance quickly. Similarly, the velocity error settled within 0.5 seconds of the disturbance, indicating that the controller could maintain the robot's speed despite the disturbance. These impulse disturbance tests validate the HSMC's effectiveness in rejecting impulse disturbances and maintaining the robot's stability. The minimal oscillations observed in the response of both pitch and velocity indicate the effectiveness of the controller's feedback loop in correcting for the disturbances quickly. Additionally, the short settling times of both pitch and velocity after the

disturbance indicate the controller's ability to react quickly to disturbances and restore the robot's stability.

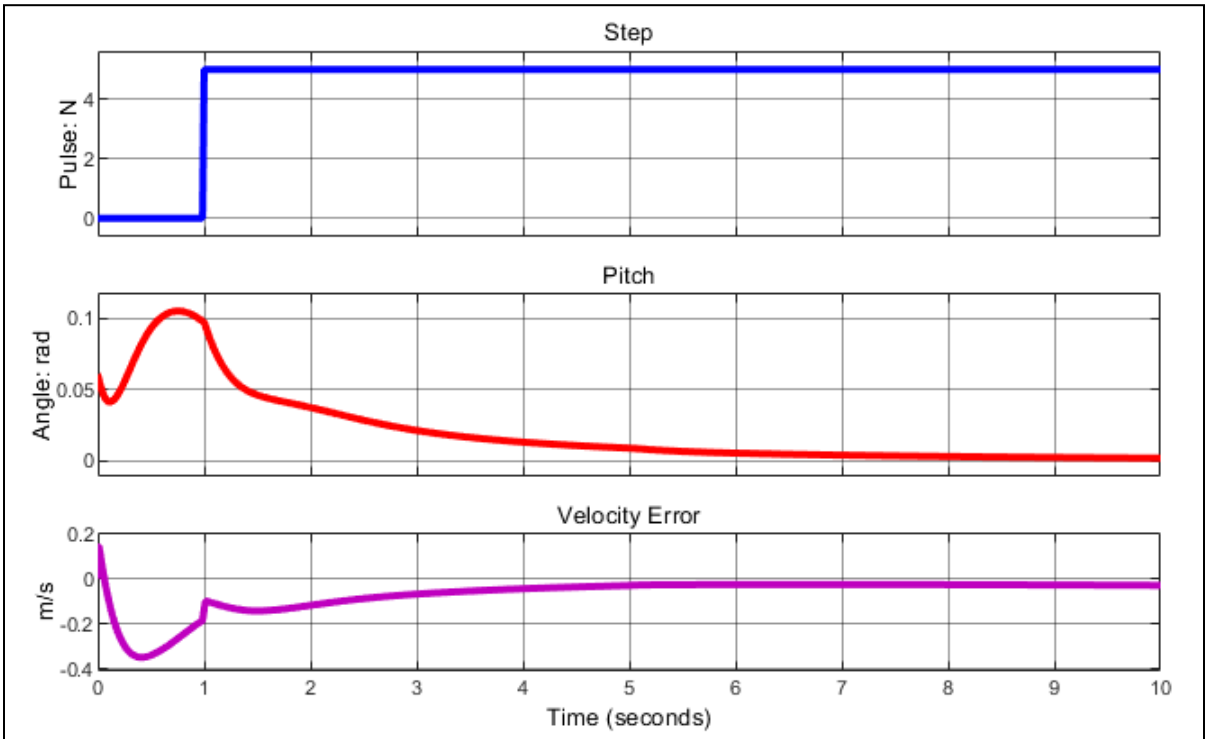

Figure 27. HSMC Pitch Step Disturbance

The subsequent test involved subjecting the TWBR to a step disturbance with a magnitude of 5N, initiated at t=1 second. As illustrated in Fig. 27, the HSMC demonstrated its effectiveness in managing the error caused by the step disturbance. The controller rapidly converged to the desired state, ensuring the TWBR maintained stability throughout the disturbance period. Upon the application of the step disturbance, the HSMC quickly responded to counteract its impact on the system. The pitch and velocity errors were minimized in a relatively short time, highlighting the controller's ability to adapt to sudden changes in the system dynamics. Additionally, the HSMC exhibited minimal overshoot and oscillation during this test, further underscoring its potential to maintain the stability of the TWBR under step disturbances. In summary, the HSMC proved to be an effective control strategy in handling step disturbances, showcasing its adaptability and

responsiveness. The controller's rapid convergence to the desired state and minimal oscillation indicates its suitability for applications where the TWBR may encounter abrupt changes in external conditions or system dynamics.

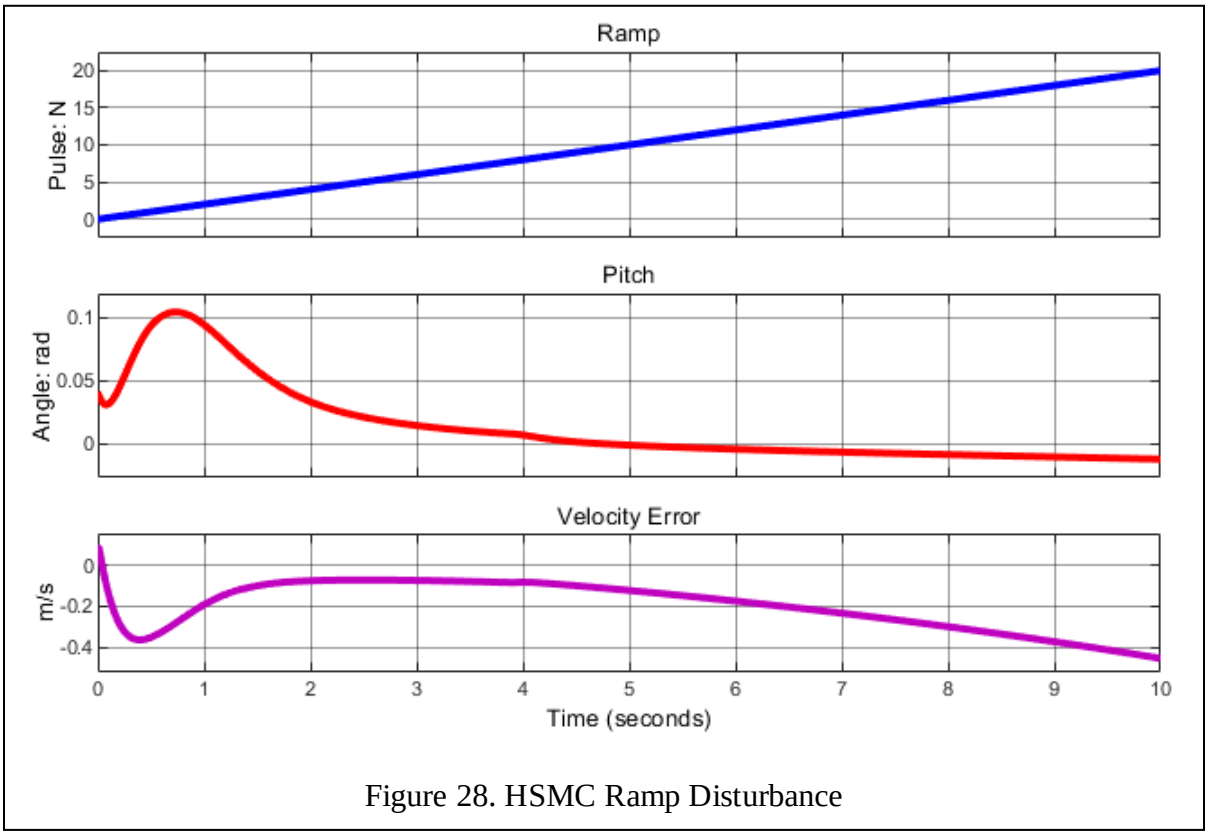

Figure 28. HSMC Ramp Disturbance

The ramp disturbance, in Fig. 28, which increased linearly from 0 to 10 N over a 10-second period, $F_d = 2t$, was applied to evaluate the HSMC's ability to handle gradual changes in external forces acting on the TWBR. The results revealed that the velocity error tracked with minor steady-state error until t=4s, after which it began to diverge. This indicates that the HSMC was initially able to maintain a stable velocity response; however, as the disturbance increased over time, the controller struggled to maintain the desired velocity, causing the error to diverge. Despite the divergence in the velocity error, the pitch angle tracked relatively well throughout the test, demonstrating the controller's ability to maintain stability in the pitch angle under the ramp disturbance. These findings suggest that while the HSMC shows promise in handling a certain range of the ramp

disturbance, but it faces challenges in ensuring optimal velocity tracking as the disturbance progressively increases. This observation highlights the need for further investigation and refinement of the HSMC to improve its performance in dealing with dynamic environments, particularly in relation to velocity control.

The next objective of the HSMC to pursue a moving target while being subject to impulse disturbances. As shown in Fig. 29, the initial location of the TWBR is given by a light blue "T" shape, the current position of TWBR is given by a dark blue "T" shape, the current target location is given by a red circle, and the paths of the TWBR and target are given by green and pink curves respectively. We see here that the target moves counterclockwise in a circle around the origin. The robot is able to pursue the target well.

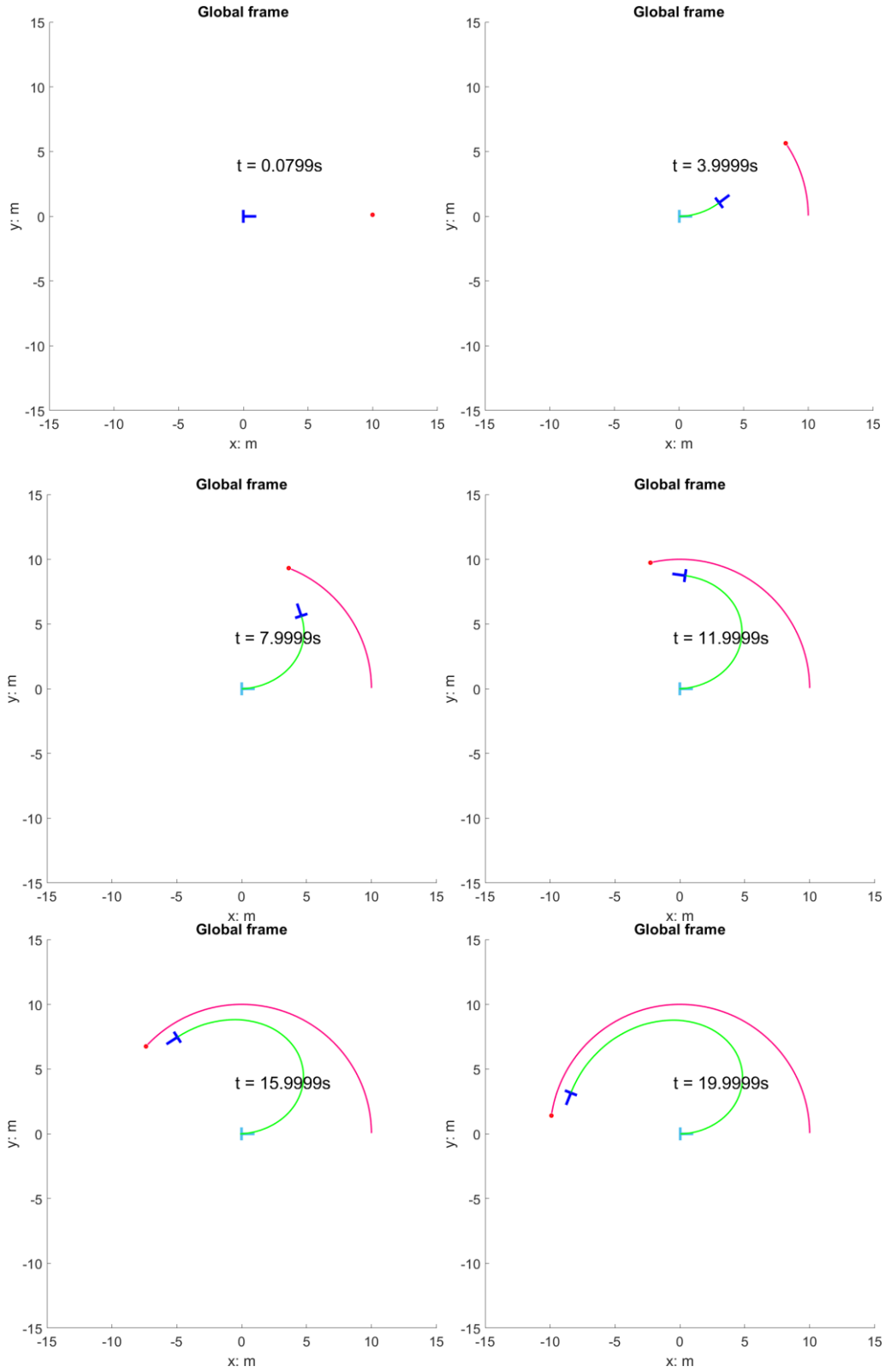

Figure 29. HSMC Moving Target Pursuit

CHAPTER 7: CONCLUSION

In this study, we developed the equations of motion representing the Two-Wheeled Balancing Robot (TWBR) and investigated its performance under various control architectures. We began by identifying the holonomic and non-holonomic conditions constraining our system and formulated the equations of motion using the Lagrangian Mechanics approach. To represent the system in terms of three generalized coordinates - position (x), pitch (α), and yaw (θ) - we applied relationships between the global and chassis frames. Following the development of the dynamic model, we developed a path planning algorithm called Artificial Potential Fields to determine set points for the variables of interest, namely velocity ($\dot{x}$) and yaw rate ($\dot{\theta}$). Subsequently, we described and designed three different control architectures with the goal of driving the state variables to their desired values. These 3 controllers of varying performance were able to stabilize the pitch and velocity variables, which plays a critical role in enabling the TWBR to autonomously navigate to stationary and moving targets while maintaining pitch stability and rejecting disturbances. These three control architectures were PID, DQN, and Sliding Mode Control. The PID system architecture consisted of three PID controllers that were successfully deployed to control the pitch, yaw rate, and forward velocity. In contrast, the sliding mode control architecture was developed by creating a hierarchical system for the pitch and velocity sliding surfaces and by applying a traditional first order sliding mode controller for the yaw setpoint.

To evaluate the performance of the control architectures, the robot models were subjected to multiple simulations, each designed to test different aspects of the controllers. These tests included impulse, step, and ramp disturbances to assess the robustness and adaptability of each controller in maintaining the TWBR's stability under varying conditions. In this model in a simulation environment, the PID was fully capable of rejecting all disturbances that we tested for.

The results obtained from these tests provided valuable insights into the strengths and limitations of each control architecture, guiding further optimization and development of more effective control strategies for the TWBR.

In the PID controller, we observed a significant ability to reject the step and ramp disturbances subjected to the system. Despite its robust performance in maintaining stability under various conditions, the PID controller introduced a significant amount of oscillation to the system, which lead to the failure of the controller at a disturbance o 75N. This rapid oscillation could be detrimental in a real-world system where we cannot rely on the non-holonomic constraints to hold true. However, as the observed oscillation might raise concerns, for smaller disturbances, these oscillations were admissible in the context of stabilizing this model that we developed.

The DQN agent showed promising results in stabilizing the pitch and velocity errors, but it was not successful at accurately stabilizing the yaw rate. This resulted in an inability to navigate to the desired target location. This is likely in part due to the very large action space. It is known that the DQN algorithm is very well suited for problems with a large observation space but has problems when the action space is too large. A further analysis of the action space's effect on training should be conducted to improve the error tracking of the yaw rate error. Despite this problem navigating, the agent showed a strong ability to stabilize the pitch and velocity under step disturbances. It was not capable of rejecting the impulse disturbances, which indicates the need to introduce larger disturbances during training. It failed at rejecting the ramp disturbance, but this could be further improved by including more diverse types of disturbances to the agent during the training phase.

The HSMC demonstrated promising results in rejecting impulse and step disturbances. The control scheme efficiently managed these disturbances while exhibiting minimal oscillation, highlighting its potential for certain real-world applications. However, we encountered challenges when the

controller was subjected to ramp disturbances. As the ramp disturbance increased to around 10N, the error began to diverge, indicating limitations in the HSMC's ability to cope with continuously increasing disturbances. This could be further improved by developing function for the gain k in the design of the Hierarchical SMC controlling pitch and velocity. In further work we can define a continuous function that bounds this gain, and still satisfies the condition defined in equation 88. This would allow the gain to be non-stationary and, in some form, 'adapt' the control input to stabilize the pitch and velocity more robustly.

In conclusion, this work has done significant work in modeling of the TWBR and characterizing the performances of PID, DQN, and HSMC controllers for this application. Each controller demonstrated their strengths and weaknesses in handling various types of disturbances. The PID controller excelled in disturbance rejection but struggled with oscillation, while the HSMC performed well in rejecting impulse and step disturbances but faced challenges with ramp disturbances. The DQN agent showed valuable results in pitch and velocity control and demonstrated a powerful ability to develop a unique policy to suit its environment. However, the DQN method is highly sensitive to reward shaping and hyperparameter tuning, which takes a very long time to optimize without GPU accelerated computing. These observations serve as valuable insights for further optimization and development of more effective control strategies for the two-wheel balancing robot, ultimately enhancing its performance and adaptability in a wide range of applications and environments. The insights gained from the tests and simulations contribute to the advancement of control systems for the TWBR, enhancing its potential for reliable and efficient operation in a wide range of applications and environments.

CHAPTER 8: FUTURE WORK

Future work will focus on optimizing the training process for the Deep Q-Learning agent used in this study. One approach that will be explored is the use of GPU-accelerated computing to reduce the time required to train complex models. By utilizing GPUs, a wider range of hyperparameters can be explored and more thorough evaluations of their impact on the agent's performance can be conducted. Overall, the goal is to achieve better performance and faster convergence of the agent, leading to more efficient and effective control in the target application. An optimized Deep Q-Learning agent will be evaluated extensively to validate its effectiveness and robustness. Another improvement to the DQN control architecture would be to train two agents: one to stabilize pitch and velocity, and the other one to stabilize the yaw rate. This would distribute the complexity of the tasks for the DQN agents, and potentially lead to better results in navigation, and the robustness to a wider range of disturbances. Further work will also focus on the development of a proof-of-concept to implement these controllers and further evaluate their capabilities for the real-world environment.

REFERENCES

[1] M. Cardona, A. Palma, and J. Manzanares, "COVID-19 pandemic impact on mobile robotics market," in *2020 IEEE ANDESCON, ANDESCON 2020*, Oct. 2020. doi: https://doi.org/10.1109/ANDESCON50619.2020.9272052.

[2] J.-J. E. (Jean-Jacques E.) Slotine and W. Li, *Applied nonlinear control*. Prentice Hall, 1991, p. 459.

[3] Vo, Tran Van Thien, Nguyen Ngoc Son, and Mai Thang Long, "Adaptive neural sliding mode control for two wheel self balancing robot," *International Journal of Dynamics and Control*, vol. 10, no. 3, pp. 771–784, Jun. 2022, doi: https://doi.org/10.1007/s40435-021-00832-1.

[4] Y. Zhang, L. He, B. Yan, J. Chen, and C. Wu, "Hierarchical sliding mode control for the trajectory tracking of a tendon-driven manipulator," *Journal of Mechanisms and Robotics*, vol. 15, no. 6, Dec. 2023, doi: https://doi.org/10.1115/1.4056577.

[5] K.-S. Huang, Y.-H. Lin, K.-B. Lin, B.-K. Lee, and C.-K. Hwang, "CASCADE SLIDING MODE CONTROL OF A SPHERICAL WHEEL ROBOT DRIVEN BY OMNI WHEELS."

[6] M. Idrees, S. Ullah, and S. Muhammad, "Sliding mode control design for stabilization of underactuated mechanical systems," *Advances in Mechanical Engineering*, vol. 11, no. 5, May 2019, doi: https://doi.org/10.1177/1687814019842712.

[7] L. Pupek and R. Dubay, "Velocity and position trajectory tracking through sliding mode control of two-wheeled self-balancing mobile robot," in *12th Annual IEEE International Systems Conference, SysCon 2018 - Proceedings*, May 2018, pp. 1–5. doi: https://doi.org/10.1109/SYSCON.2018.8369548.

[8] J. Yi, W. Wang, D. Zhao, and X. Liu, "Cascade sliding-mode controller for large-scale underactuated systems," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2005, pp. 301–306. doi: https://doi.org/10.1109/IROS.2005.1545463.

[9] D. Qian and J. Yi, "Hierarchical sliding mode control for under-actuated cranes design, analysis and simulation."

[10] M. A. Kon and Leszek Plaskota, "Neural networks, radial basis functions, and complexity."

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. The MIT Press, 2016.

[12] Y. Wei, X. Nie, M. Hiraga, Kazuhiro Ohkura, and Z. Car, "Developing end-to-end control policies for robotic swarms using deep q-learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 23, no. 5, pp. 920–927, 2019, doi: https://doi.org/10.20965/jaciii.2019.p0920.

[13] R. Dorf and R. Bishop, *Modern control systems*, Thirteenth. Pearson, 2016.

[14] G. F. Franklin, J David Powell, and Abbas Emami-Naeini, *Feedback control of dynamic systems sixth edition*, Sixth. Pearson, 2010.

[15] P. Badoni, "Robotics controller: A literature survey," INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, Oct. 2015. Accessed: Jan. 22, 2023. [Online]. Available: https://www.ijert.org/research/robotics-controller-a-literature-survey-IJERTV4IS100533.pdf

[16] S. E. N and Che, “POTENTIAL FIELD METHODS AND THEIR INHERENT APPROACHES FOR PATH PLANNING,” vol. 11, no. 18, 2016, Available: www.arpnjournals.com

[17] Volodymyr Mnih *et al.*, “Playing atari with deep reinforcement learning,” Dec. 2013. http://arxiv.org/abs/1312.5602

[18] C. R. McInnes, “Velocity field path-planning for single and multiple unmanned aerial vehicles,” *The Aeronautical Journal*, vol. 107, no. 1073, pp. 419–426, 2003, doi: https://doi.org/10.1017/s0001924000013348.

[19] X. Yuan, “Research on the limitations of UAV path planning based on artificial potential field method,” *2022 9th International Forum on Electrical Engineering and Automation (IFEEA)*, 2022, doi: https://doi.org/10.1109/ifeea57288.2022.10037827.

[20] Y. Chen and T. Li, “Collision avoidance of unmanned ships based on artificial potential field,” *2017 Chinese Automation Congress (CAC)*, 2017, doi: https://doi.org/10.1109/cac.2017.8243561.

[21] Q. Zhu, Y. Yan, and Z. Xing, “Robot path planning based on artificial potential field approach with simulated annealing,” *Sixth International Conference on Intelligent Systems Design and Applications*, 2006, doi: https://doi.org/10.1109/isda.2006.253908.

[22] D. J. Inman, *Engineering vibration*. Prentice Hall, 1994.

[23] R. C. Hibbeler, *Engineering mechanics: Dynamics*. Prentice Hall, 2013.

[24] D. Jang and S. Ma, “Lecture 20: (soft robotics part 1),” 2020. Available: https://pages.github.berkeley.edu/EECS-106/sp22-site/assets/scribe_notes/scribe_lec_11B.pdf

[25] B. Siciliano and E. Al, *Robotics: modelling, planning and control*. Springer, 2010.

[26] I. Carlucho, Mariano De Paula, and G. G. Acosta, “Double Q-PID algorithm for mobile robot control,” *Expert Systems with Applications*, vol. 137, pp. 292–307, 2019, doi: https://doi.org/10.1016/j.eswa.2019.06.066.

[27] J.-S. .R Jang and N. Gulley, “Gain scheduling based fuzzy controller design,” *NAFIPS/IFIS/NASA ’94. Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference. The Industrial Fuzzy Control and Intelligent Systems Conference, and the NASA Joint Technology Wo*, 1995, doi: https://doi.org/10.1109/ijcf.1994.375142.

[28] A. Mathew, R Ananthu, P Binsy, A. Vahid, C. Thomas, and S Sidharthan, “Design and control of a two-wheel self-balancing robot,” *IOP Conference Series: Materials Science and Engineering*, vol. 1114, no. 1, p. 012058, 2021, doi: https://doi.org/10.1088/1757-899x/1114/1/012058.

[29] D. Lee, J. Jeong, Yong Hwi Kim, and Jin Bae Park, “An improved artificial potential field method with a new point of attractive force for a mobile robot,” *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, 2017, doi: https://doi.org/10.1109/icrae.2017.8291354.

[30] Min Cheol Lee and Min Gyu Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, 4AD, doi: https://doi.org/10.1109/aim.2003.1225434.

[31] Farid Bounini, D. Gingras, Herve Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, doi: https://doi.org/10.1109/ivs.2017.7995717.

[32] Iswanto Iswanto, Alfian Ma'arif, Oyas Wahyunggoro, and A. Imam, "Artificial potential field algorithm implementation for quadrotor path planning," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019, doi: https://doi.org/10.14569/ijacsa.2019.0100876.

[33] H. Goldstein, J. Safko, and C. P, *Classical mechanics*, Third. Pearson, 2014, pp. 12–50.

[34] A. M. Shafei and H Mirzaeinejad, "A general formulation for managing trajectory tracking in non-holonomic moving manipulators with rotary-sliding joints," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 3–4, pp. 729–746, 2020, doi: https://doi.org/10.1007/s10846-019-01143-6.

[35] Y. Zhou, Z. Wang, and K. Chung, "Turning motion control design of a two-wheeled inverted pendulum using curvature tracking and optimal control theory," *Journal of Optimization Theory and Applications*, vol. 181, no. 2, pp. 634–652, 2019, doi: https://doi.org/10.1007/s10957-019-01472-4.