MODELING AND CONTROL OF A FULLY ACTUATED UNMANNED SURFACE VEHICLE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA KILINÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

APRIL 2023

Approval of the thesis:

MODELING AND CONTROL OF A FULLY ACTUATED UNMANNED SURFACE VEHICLE

submitted by **MUSTAFA KILINÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** ________________

Assoc. Prof. Dr. Mustafa Mert Ankaralı
Supervisor, **Electrical and Electronics Engineering, METU** ________________

Examining Committee Members:

Prof. Dr. Afşar Saranlı
Electrical and Electronics Engineering, METU ________________

Assoc. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU ________________

Prof. Dr. Klaus Werner Schmidt
Electrical and Electronics Engineering, METU ________________

Assoc. Prof. Dr. Emre Özkan
Electrical and Electronics Engineering, METU ________________

Prof. Dr. Hüseyin Demircioğlu
Electrical and Electronics Engineering, Hacettepe University ________________

Date:20.04.2023

ABSTRACT

MODELING AND CONTROL OF A FULLY ACTUATED UNMANNED SURFACE VEHICLE

Kılınç, Mustafa
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Mustafa Mert Ankaralı

April 2023, 79 pages

This thesis presents a mathematical model and control strategies for a fully actuated unmanned surface vehicle (USV). We aim to control the body velocity of the robot with a model-based time-varying linear quadratic regulator (TV-LQR) and operate the control algorithm from the ground station of the robot with low-frequency remote communication signals. As TV-LQR computes the optimal control policy by considering a mathematical model, the 3 degrees of freedom (3DoF) decoupled model is derived from the Newton-Euler equations and Fossen's low-speed USV assumptions. MATLAB's constrained nonlinear optimization function estimates the parameters of this model by using experimental data. The k-fold cross-validation method is implemented to see the distributions of the parameters between randomly chosen validation datasets. The conventional PI control algorithm is chosen as a baseline method for comparison with our controller approach. The main contributions of this study are developing a mathematical model of a novel USV and performance analysis of the model-based TV-LQR controller in actual experiments. Results show that fully actuated velocity control with TV-LQR and quadratic programming based control allocation improves the robot's velocity tracking performances in each DoF. Also, quadratic programming allocates the robot's actuator inputs according to the thrusters' physical limits and optimizes power consumption during the control allocation.

Keywords: Unmanned Surface Vehicle, Mathematical Modelling, System Identification, Velocity Control, TV-LQR, PI, Control Allocation

ÖZ

TAM TAHRİKLİ BİR İNSANSIZ YÜZEY ARACININ MODELLENMESİ VE KONTROLÜ

Kılınç, Mustafa
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Mustafa Mert Ankaralı

Nisan 2023 , 79 sayfa

Bu tez, tam tahrikli yeni bir insansız su üstü aracı (İSA) için matematiksel bir model ve kontrol stratejileri sunmaktadır. Biz, model tabanlı zamanla değişen ikinci dereceden düzenleyici (TV-LQR) ile robotun kendi hızını kontrol etmeyi ve düşük frekanslı uzaktan iletişim sinyalleri ile kontrol algoritmasını kontrol algoritmasını yer istasyonundan çalıştırmayı amaçladık. TV-LQR, matematiksel bir modeli göz önünde bulundurarak optimum kontrol algoritmasını hesapladığı için 3 serbestlik derecesinde ayrıştırılmış model, Newton-Euler denklemlerinden ve Fossen'in düşük hızlı İSA varsayımlarından türetilmiştir. MATLAB'ın kısıtlı doğrusal olmayan optimizasyon fonksiyonu deneysel verileri kullanarak bu modelin parametrelerini tahmin etmiştir. Rastgele seçilen doğrulama veri setleri arasındaki parametre dağılımlarını görmek için k-katlı çapraz doğrulama yöntemi uygulanmıştır. Kendi kontrolcü yaklaşımımızı karşılaştırmak için geleneksel PI kontrol algoritması dayanak yöntem olarak seçilmiştir. Bu çalışmanın ana katkıları, yeni bir İSA'nın matematiksel modelinin geliştirilmesi ve model tabanlı TV-LQR kontrolcüsünün gerçek deneylerdeki performans analizidir. Sonuçlar, TV-LQR ve ikinci dereceden programlama tabanlı kontrol ataması ile tam tahrikli hız kontrolünün robotun hız takip performansını iyileştirdiğini göstermiştir. Ayrıca ikinci dereceden programlama, robotun çalıştırıcı girdilerini iticilerin fiziksel limitlerine göre tahsis eder ve itici tahsisi sırasında güç tüketimini optimize eder.

Anahtar Kelimeler: İnsansız Yüzey Aracı, Matematiksel Modelleme, Sistem Tanılama, Hız Kontrolü, TV-LQR, PI, Kontrol Ataması

Scientia Dux Vitae Certissimus

ACKNOWLEDGMENTS

I would like to express my gratitude to my academic advisor, Assoc. Prof. Dr. Mustafa Mert Ankaralı, who supported me throughout my thesis work. He has made lasting contributions to my academic life with the interest he has shown during my graduate education.

I am grateful to Oğuz Özdemir, with whom we carried out our thesis work in the same period, for the fun times we had together. During this time, we always tried to motivate each other.

My friends Atakan Durmaz and Ferhat Gölbol contributed significantly to the experimental robot's preparation process and test activities.

Turkish Scientific and Technological Research Council (TÜBİTAK) provided the necessary financial resources to manufacture the experimental robot, enabling us to validate our work in real life. The TÜBİTAK 1001 program has contributed partially to this thesis through projects 118E195 and 122E249.

I would like to thank ROKETSAN, where I have worked since my graduate education, for their convenience.

I would like to convey my respects to my first scout leader, Ali Emul, who has been an example in my decisions regarding my life during my thesis work.

I could not have undertaken this journey without my fiancee, Deniz Bakırcı, who continuously increased my motivation in my thesis work. She has been my strongest supporter in pursuing the values I believe in.

Finally, words cannot express my gratitude to my family, who raised me with great dedication. I want to express my gratitude to my first teachers, my mother, Hanife Şahin, and my father, Yakup Kılınç, who showed me the right path throughout my education life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
3DOF	3 Degrees of Freedom
ESC	Electronic Speed Control
GPS	Global Positioning System
IMO	International Maritime Organization
IMU	Inertial Measurement Unit
MAVlink	Micro Air Vehicle Link
METU	Middle East Technical University
MIMO	Multi Input Multi Output
MSE	Mean Squared Error
NED	North-East-Down
PSO	Particle Swarm Optimization
ROS	Robot Operating System
RTK	Real Time Kinematics
RTT	Round Trip Time
SISO	Single Input Single Output
SITL	Software in the Loop
TV-LQR	Time Varying Linear Quadratic Regulator
USV	Uncrewed Surface Vehicle

CHAPTER 1

INTRODUCTION

1.1 Motivation and Background

The civil and military industries require research in Unmanned Surface Vehicles (USVs) due to their potential to accomplish security missions, path following, velocity tracking, object detection, and obstacle avoidance tasks in marine areas. USVs can utilize navigation and guidance algorithms to perform autonomous tasks in the port area. As a result, USVs require high control capabilities to be effective in these application areas.

As there is a growing interest in specialized applications of USVs, it is necessary to establish regulations for their autonomous operations. To this end, levels indicating the autonomy of uncrewed autonomous vehicles have been determined, and regulations have been developed to classify them [2]. The International Maritime Organization (IMO) has identified four levels of ship automation: entirely autonomous, fully remotely controlled, partially remote and partially crew-controlled, and partially autonomous [3]. The existence of international standards for crewless autonomous sea vehicles underscores the significance of work in this field, which can lead to the development of more systematic USV technologies. Thus, it is imperative to address these regulatory issues to promote the effective and safe operation of USVs in marine areas.

The closed-loop control algorithm can operate in the ground station and the robot's embedded hardware. The disadvantage of controlling from a host computer is that it works at low frequencies, and communication delays can affect the robot's performance. However, by completing the control loop using a ground station, more computationally demanding algorithms can be implemented, enabling the execution of sophisticated control algorithms without affecting the robot's performance. We aim to control the robot from the ground station by compensating for the round trip time of the robot's telemetry messages and motor commands.

At low frequencies, USVs control should prioritize velocity control over position control. This is because the transfer function with position output at low frequencies has a phase angle of -90 degrees and decreasing gain. In comparison, the transfer function with velocity output has a constant gain and a phase angle of 0 degrees. As a result, a velocity controller is better suited for delays that may occur in low-frequency controllers and is more robust and smooth than a position controller. Furthermore, using velocity control, the USV can achieve its control objectives with less energy consumption. A study by Zelenak highlights the advantages of velocity control for low-frequency controllers and its potential benefits for USV control [4].

The PI controller can operate without a dynamic model of the robot, while the TV-LQR controller requires the linearization of the robot's dynamic model. The dynamic model information enhances the robustness of the TV-LQR controller and enables tuning the robot's feedback gains efficiently. The robot's dynamics can be obtained through an optimization algorithm applied to a specific mathematical model. This study aims to obtain the best robot model by comparing optimization methods. The mathematical model derived from this study can be utilized in various model-based motion controllers.

Fully actuated systems have equal or more actuators than the robot's degrees of freedom. When the number of actuators exceeds the degrees of freedom, the control allocation matrix becomes non-square, causing the inverse of the matrix to be undefined. Typically, the pseudo-inverse method is used in such cases. However, this method does not consider the robot's constraints and energy efficiency. Alternatively, quadratic programming can address the control allocation problem and enhance robot performance by accounting for the constraints of robot thrusters and energy consumption [5]. The present study aims to improve the robot's performance by considering these factors in control allocation.

1.2 Problem Statement

This study examines the control and modeling of an innovative uncrewed surface robot. The primary challenge lies in determining the dynamic and kinematic properties of the robot, which is remotely controlled wirelessly. Increasing the number of parameters in the dynamic model complicates system identification. Optimizing with fewer parameters may result in a dynamic model that does not accurately represent the robot's dynamic characteristics. Another issue involves developing a control algorithm tailored to the robot to follow a reference velocity accurately. Achieving velocity tracking with a low-frequency control loop presents a significant challenge. Factors influencing trajectory tracking performance include the observation of the robot's state vector at a constant frequency, consistent and noise-free GPS data measurements, and the environmental noise generated by water currents. This study compares the performance of model-based and classical PI controllers in trajectory tracking. Another challenge is allocating control signals across three axes to four motor inputs, considering both motor limits and energy consumption. The constraint-unaware nature of the pseudo-inverse method may hinder the robot's controller from operating at the suitable motor inputs.

1.3 Literature Review

USVs have become a prominent research topic, with advancements in both hardware and simulation environments. Uncrewed sea vehicle technologies began with Nikola Tesla's work on a floating robot in 1898 [6], and numerous robot models have since addressed the challenges in USV technology. USV models can be classified as underactuated or fully actuated based on their number of motors and positioning. An underactuated robot has a rank of less than the robot's DoF number in its transformation matrix, which converts motor forces into the robot's body axes [7], [8], [9]. At the same time, fully actuated systems are easier to control and enable the robot to perform high-resistance maneuvers on each degree of freedom [10], [11].

In Unmanned Surface Vehicles, communication, and computational systems facilitate the robot's interaction with its environment and ground station. Tethered communication, Wi-Fi, and Radio Frequency can be alternatives to transfer data between the uncrewed vehicles and ground station [12], [13], [14]. A study by Ge et al. [15] highlights that Wi-Fi is a fast and cost-effective method for short-range robots, enabling continuous transmission of images or videos. Meanwhile, the computational system of a USV typically employs two distinct hardware structures: microcontrollers serve as autopilots and translate communication signals into meaningful servo commands such as the Pixhawk [16], or Ardupilot Mega flight controllers [17], and single-board computers used for the algorithms requiring high processing power and onboard execution such as image processing methods. The onboard computation system and ground station communicate using a specific middleware, such as the Micro Air Vehicle Link (MAVlink) protocol [18], selected for its compatibility with the PyMAVlink python library and Robot Operating System (ROS) software packages [19]. In summary, advances in wireless communication and computational systems have driven the development of more efficient and capable USVs, enabling further communication with ground stations and practical onboard computation.

In order to establish a mathematical model for a robotic system, it is essential to identify a suitable representation for the variables constituting the robot's state space. The derivation of the kinetic and dynamic equations for a crewless surface vehicle necessitates the employment of the SNAME notation to represent the position, velocity, force, angle, angular velocity, force, and momentum vectors across three axes [20]. The axes defining the robot's motion on a two-dimensional maritime surface plane consist of surge, sway, and yaw motion. These notations facilitate the modeling of both the kinematic and kinetic aspects of the robot. The kinematic model serves to convert the robot's motion from a global reference frame, such as the North-East-Down (NED) coordinate frame, Earth-Centered Earth-Fixed (ECEF), and Earth-Centered Inertial Frame (ECI), to the robot's body frame [21]. The sideslip angle effect can be added to the kinematic model to obtain a more realistic robot model [9]. However, the development of advanced control algorithms necessitates the implementation of a dynamic model.

Utilizing Newton's Second Law and the Newton-Euler formulation, it becomes possible to examine the relationship between a robot's acceleration and the forces affecting the system dynamics through a differential equation. Derived from these principles are the motion equations for underwater robots with a 6 DoF vector representation [22]. This study employs the equations in a reduced form, incorporating only 3 DoF. These equations show that Coriolis, hydrodynamic damping, gravitational, and buoyancy forces impact a robot floating in the water. This dynamic equation and its numerous simplified versions are applied in various USV studies, depending on specific needs [9], [23], [24].

System identification is a method that estimates a dynamic model's mathematical representation and constructs the predicted model for the robot. It solves linear and nonlinear black and gray box models [25]. Modeling the USV with Artificial Neural Networks (ANN) without theoretical knowledge can exemplify the black-box problem [26]. Estimating parameters in the Newton-Euler model of USV is a gray-box problem. In research from Sonnenburg and Woolsey, the dynamic model of the USV is identified by parameter estimation [9].

As outlined in the study [27], USV technology research can be categorized under guidance, navigation, and control headings. Control of the USV has four different applications; set-point regulation, trajectory tracking, path following, and path maneuvering. Velocity control methods try to minimize the error between the actual and desired velocity of the robot. Model-based optimal control and conventional PID control methods can solve the velocity control problem by tracking the desired velocity trajectories.

PID controllers are still used in the control of many industrial robots today. The PID velocity controller looks at the error signal in velocity tracking, the derivative of the error, and the accumulated error amount as decoupled without needing a dynamic system model [28]. Since each error type is multiplied by a separate coefficient, three coefficients must be tuned in a Single-Input Single-Output (SISO) system. In order to tune these coefficients, the step responses of the system can be analyzed by performing experimental studies, or the Ziegler-Nichols type theoretical methods can choose coefficients [29], [30]. At the same time, coefficients can be scheduled by applying fuzzy gain scheduling [31], or PID gains can be generated with neural networks [32]. PID controller is preferred for controlling chemical processes [30]. Many motor and uncrewed vehicles can be controlled with PID controllers or their variations, such as PI and P controllers [33].

Optimal control methods can be applied to the USVs as well as model-free methods. A nonlinear optimal control policy can be solved by nonlinear programming tools [34]. As Tedrake studied TV-LQR-based optimal control for the perching maneuvers [35], it is possible to implement TV-LQR for the USVs to obtain robust trajectory tracking performances. In reinforcement learning algorithms, a robot can be viewed as an agent, and the deep deterministic policy gradient method can be employed for its control [36]. Model predictive control is another alternative for optimal control methods. With additional Lypunav function constraints, Lyapunov-based MPC can be obtained with improved trajectory tracking performance [37].

Our contributions are as follows:

- A mathematical model of a novel uncrewed surface vehicle is derived, and the model parameters are estimated with nonlinear constrained optimization. Distributions of the parameters are evaluated with the k-fold cross-validation technique.
- Reference trajectory tracking with conventional PI controller and model-based TV-LQR controller are compared. Software in the Loop (SITL) and the actual robot tests are considered with some performance metrics.
- Control allocation of the fully actuated USV is improved with quadratic programming. Constraints-aware control allocation is applied as an alternative to the pseudo-inverse method.

1.4 Experimental Setup

In this section, we delineate the distinctive design of an uncrewed surface vehicle, emphasizing its functional mechanical characteristics. We elucidate the technical characteristics of this fully actuated robotic system, which features 3 degrees of freedom motion and remote wireless control capabilities. Particular attention is given to the mechanical design elements and selecting electronic components, such as sensors, actuators, control, and communication units. Furthermore, we explicate the software elements that facilitate the robot's remote operation. Finally, some experiments are held to understand measurement noise, sampling rate, and communication latency.

1.4.1 Mechanical Design

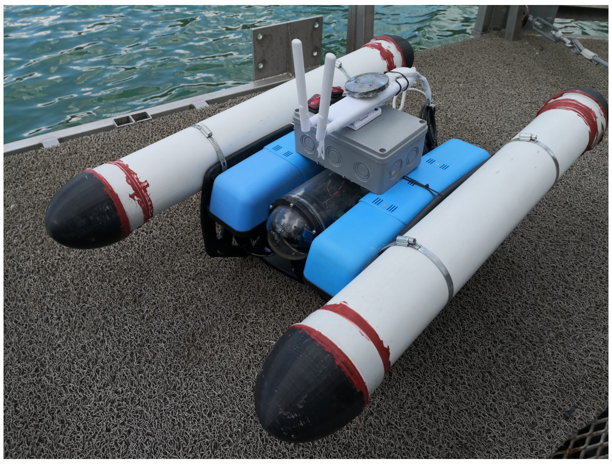


Figure 1.1: Real experimental setup: USV robot

The USV has a watertight structure. It uses a black frame made of water-resistant High-Density Polyethylene material. The robot is a modified version of the BlueRov2 underwater vehicle [12]. BlueRov 2 is a suitable platform for underwater vehicle developments [38] [39] [40]. However, its hardware and mechanical structure must be modified to operate this robot as a surface vehicle. Mechanical changes aim to keep the robot on the surface of the water. As seen in Figure 1.1, the USV is supported by Catamaran type floats. Diving motors are replaced with additional floating sponges to decrease the density of the robot. Figure 1.2 shows the remaining motors' configuration. T200 thrusters are placed on this frame with 45-degree angles as in figure 1.2. Four motors are positioned to create force in the surge, sway, and yaw axis. The battery unit used for wireless communication is housed in an IP65 standard enclosure. The total mass of the robot is 12.40 kg. The width and length measurements are 54 cm and 88 cm, respectively. On the upper surface of the robot, a GPS antenna and a Wi-Fi module are placed in a visible area.

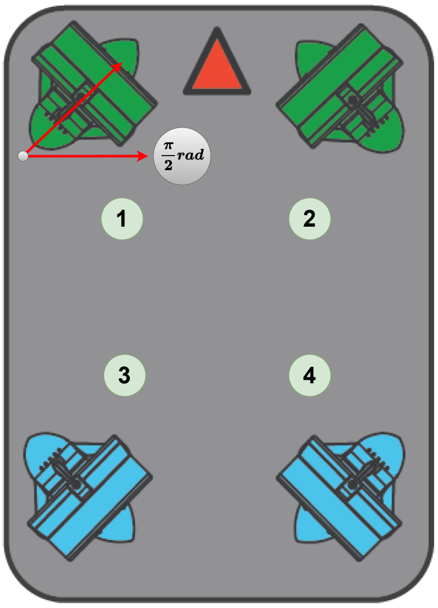


Figure 1.2: Thruster configuration of the robot. ArduSub framework controls each motor with given id numbers. Each motor's thrust vector makes $\pi/2\ rad$ angle with the sway axis of the robot. The positive surge axis direction is indicated with a red arrow.

1.4.2 Hardware Components

The hardware equipment of the robot consists of 2 classes: the ground station and the USV platform. Real-Time Kinematic (RTK) positioning sensor connection is made with a computer at the ground station to obtain a more accurate GPS signal. Real-Time Kinematic positioning represents a relative positioning technique that facilitates centimeter-level positioning accuracy for a station by incorporating satellite corrections from auxiliary stations. RTK methodologies mandate communication between the Ground station and the robot to exchange satellite correction data effectively. The QGroundControl software (fig. 1.4) automatically detects the RTK sensor plugged into the USB port of the ground station. It sends GPS correction messages obtained from the USB port to the USV by MAVLink protocol. The USV can operate manually with the remote control commands sent from the ground station computer via a joystick. Autonomous control algorithms are run on the ground computer and generate desired remote control inputs for the motor. The motor input commands on the ground computer are transmitted to the robot on the water surface via Wi-Fi signals. The motor commands sent from the ground computer to the robot are transmitted to the T200 Thrusters with Electronic speed control (ESC) motor drivers. The Inertial measurement unit (IMU) and GPS sensors on the USV are used to obtain the robot's navigation information. The body velocity estimation algorithms run on the Pixhawk module, and the estimated velocity messages are sent to the ground station via the Wi-Fi module. The hardware diagram of the robot is shown in figure 1.3.

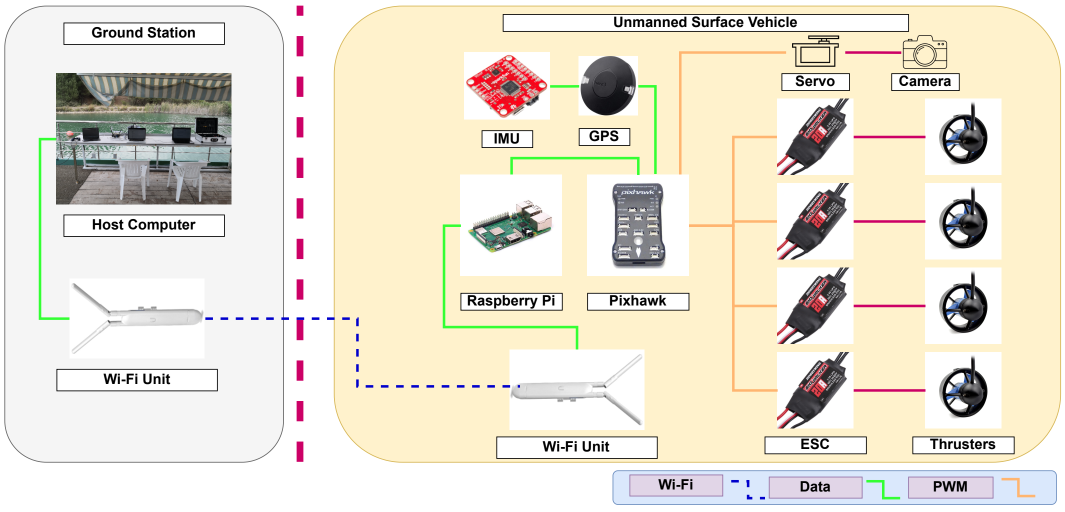


Figure 1.3: Block Diagram of the USV's hardware components. Data has two types; remote control inputs from the ground station and velocity estimation messages obtained by the Pixhawk.

In order to provide wireless communication between the robot and the ground station, Unifi UAP-AC-M Wi-Fi modules belonging to the Ubiquiti Network Company are integrated into the robot and the ground station. Message packets are sent to the IP address specified for the robot over a local network created at the ground station. More than one computer can connect to the robot and send messages at the same time. Message packets reaching the access point device on the robot with IEEE 802.11ac standard are transferred to Raspberry Pi via ethernet connection. The Pixhawk device communicates serially with the Raspberry Pi and sends the motor commands to the robot's motors according to the incoming message packets.

1.4.3 Software Components

The QGroundControl software shown in figure 1.4 can control the robot with the joystick device and communicate with the robot and the RTK sensor in the ground station. The QGroundControl software reads MAVLink messages over ethernet and displays the robot's position on the global earth map. It checks the battery and connection status of the robot with the message packets it reads. In an emergency, the robot can be disarmed by cutting off its power to the motors. With these advantages, it is used during experiments at the ground station.

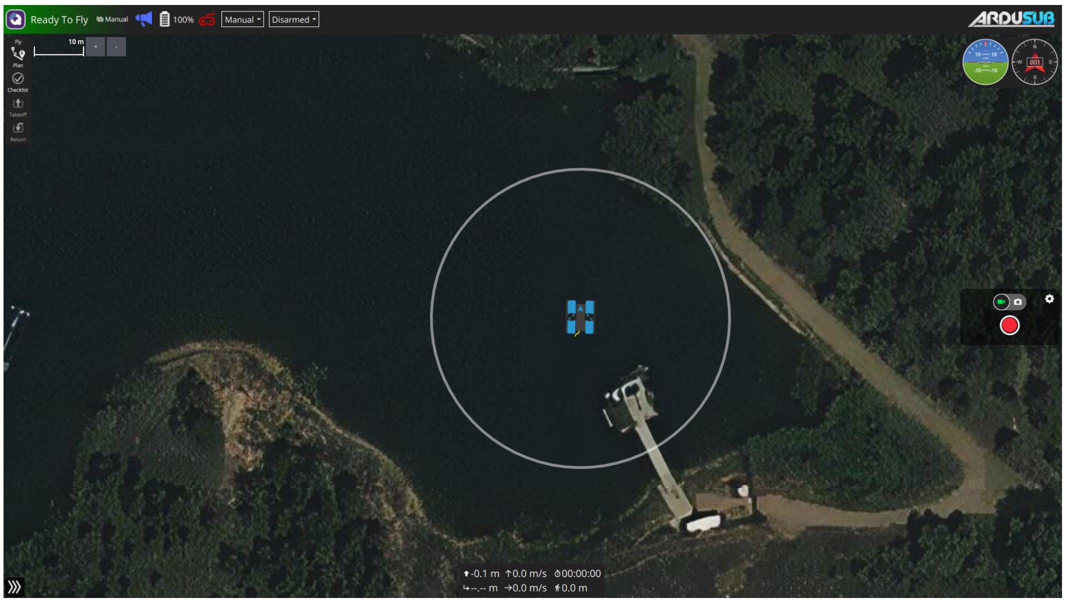


Figure 1.4: QGroundControl Software

A Conda environment has been prepared at the ground station for the robot to work effortlessly in the Python environment. Python version 3.9 is installed for the environment. Another environment module Jupyter-notebook executes modular code blocks and visualizes collected data quickly. Different controllers were tested, and experimental data were collected by running Python code blocks with the Jupyter-Notebook interface.

MAVLink protocol is used to communicate between the ground station and the robot. The advantage of this protocol is that it can work in the ROS environment with the mavros ROS package. It also supports algorithms written in Python with the PyMAVLink Python library. In this study, a Python class is prepared for sending motor commands to the robot and receiving sensor data from the robot.

When the Python class is initialized on the ground station, a connection with the robot is defined via ethernet. Then, the frequency of the MAVLink data packets "MAV_DATA_STREAM_POSITION" and "MAV_DATA_STREAM_EXTRA1" is set to 20 Hz. A state vector with six elements is obtained with the information from the "LOCAL_POSITION_NED" and "ATTITUDE" messages.

Vector operations in the Python class are performed using the Numpy library. Sympy library, which enables symbolic variable operations, is preferred to linearize the state space around a trajectory. In the control allocation part, the CVXOPT python library performs quadratic optimization. The "solve_ivp" function of the Scipy library is used to obtain numerical solutions of differential equations.

$$\begin{bmatrix} ch_1 \\ ch_2 \\ ch_3 \\ ch_4 \end{bmatrix} = \begin{bmatrix} 1.0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} x \begin{bmatrix} motor_1 \\ motor_2 \\ motor_3 \\ motor_4 \end{bmatrix} \tag{1.1}$$

When we look at the software on the USV, we can give an example of the Ardupilot autopilot software running inside Pixhawk. The firmware has been uploaded to Pixhawk by selecting the supported ArduSub configuration for underwater and surface vehicles. As one can understand, the ArduSub configuration sends inputs to the motors via remote control channels. Each channel is set to move a different degree of freedom. In order to try methods such as control allocation, it is necessary to send direct inputs to the motors rather than the body's motion axes. This situation has been solved with a tricky solution. As seen in equation 1.1, each motor input is assigned to a single channel by creating a custom frame.

1.4.4 Software in the Loop

The ArduSub software-in-the-loop (SITL) toolbox enables researchers to conduct robotic experiments on a computer. Assuming a uniform spherical object, the SITL's mathematical model derives its inertia and mass parameters from the BlueROV2 underwater vehicle [12]. However, our robot has additional mechanical components, and the added mass effect of water is more complex than that of a spherical object. Consequently, the robot exhibits distinct rigid body and added mass coefficients. While the SITL models a 6-DoF vehicle, the studied USV operates as a 3-DoF vehicle, leading to variations in system parameters and control performances. Due to the high costs associated with actual experiments, the SITL toolbox is a preparatory environment for real-world testing. An additional benefit is the direct extraction of 3-DoF dynamic model parameters from the SITL source code, eliminating the need for identification steps. This approach allows for analyzing model-based controller performances without considering parameter estimation accuracy. Notably, the source code employs a quadratic function for drag force, resulting in the absence of linear damping coefficients in the SITL vehicle model. The parameters of this mathematical model are presented below.

- m_x = 16.75 kg (Mass coefficient of the x-axis)
- m_y = 16.75 kg (Mass coefficient of the y-axis)
- I_ψ = 0.268 kgm^2 (Inertia coefficient around z-axis)
- d_{nl1} = 48.28 Ns^2/m^2 (Nonlinear damping coefficient of x-axis)
- d_{nl2} = 62.08 Ns^2/m^2 (Nonlinear damping coefficient of y-axis)
- d_{nl3} = 7.08 Ns^2/rad^2 (Nonlinear damping coefficient of yaw-axis)

The SITL toolbox acquires desired motor inputs through MAVLink messages and maps the desired PWM gains to the motor thrusts. Employing the inertia, mass, and damping coefficients solves the system dynamics and updates the robot's states. The robot's current state information can be accessed using the same MAVLink messages employed during the experiments. This implies that the same control algorithm block in the Python environment can be utilized for both the SITL and the actual robot without any code modifications.

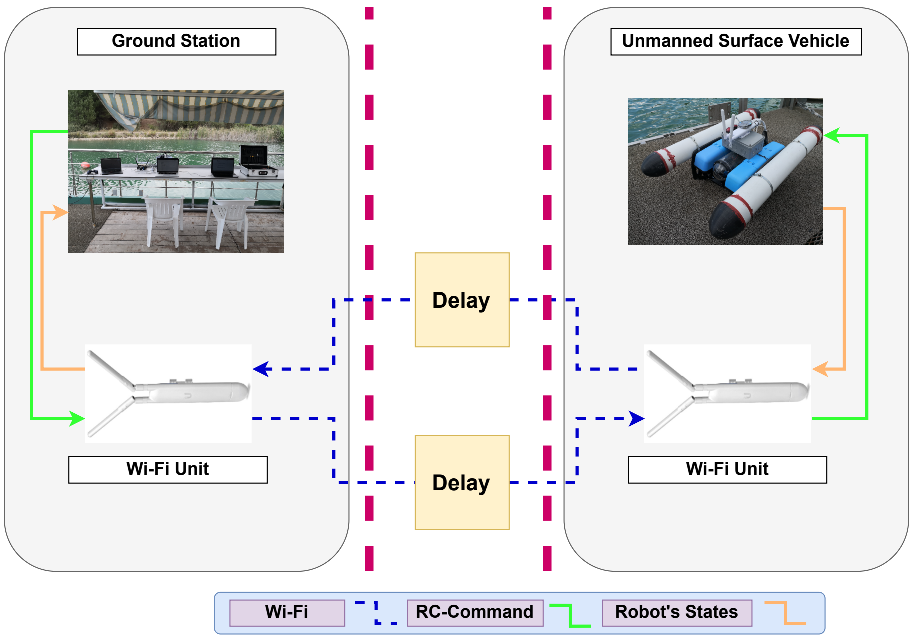


Figure 1.5: Control loop for the system. Communication delay distribution has a 9.98 ms mean value. The control algorithm runs in the ground station with a constant 20 Hz frequency.

1.4.5 Communication Latency and Rate

The robot in this study communicates with the ground station via a Wi-Fi module, which can introduce latencies. Figure 1.5 presents the communication diagram for the control loop. Latencies may occur when receiving velocity states from the ground station and transmitting motor inputs to the USV. To measure the latency of MAVLink messages, one can utilize the ’TIMESYNC’ message type. Upon sending the ’TIMESYNC’ message from the ground station, the message package stores the host computer’s current time information. When the Pixhawk returns the signal to the ground station, the time difference between the current time and the stored time parameter yields the signal’s round trip time (RTT). Half of the RTT value corresponds to the communication latency.

Figure 1.6 demonstrates that the 95% confidence value is 29.998 ms, less than the controller’s sampling time. The standard deviation of the latencies amounts to 9.36 ms. Out of 1000 samples, only 9 exhibit a latency exceeding the controller’s sampling time. These findings indicate that using this communication method, a 20 Hz

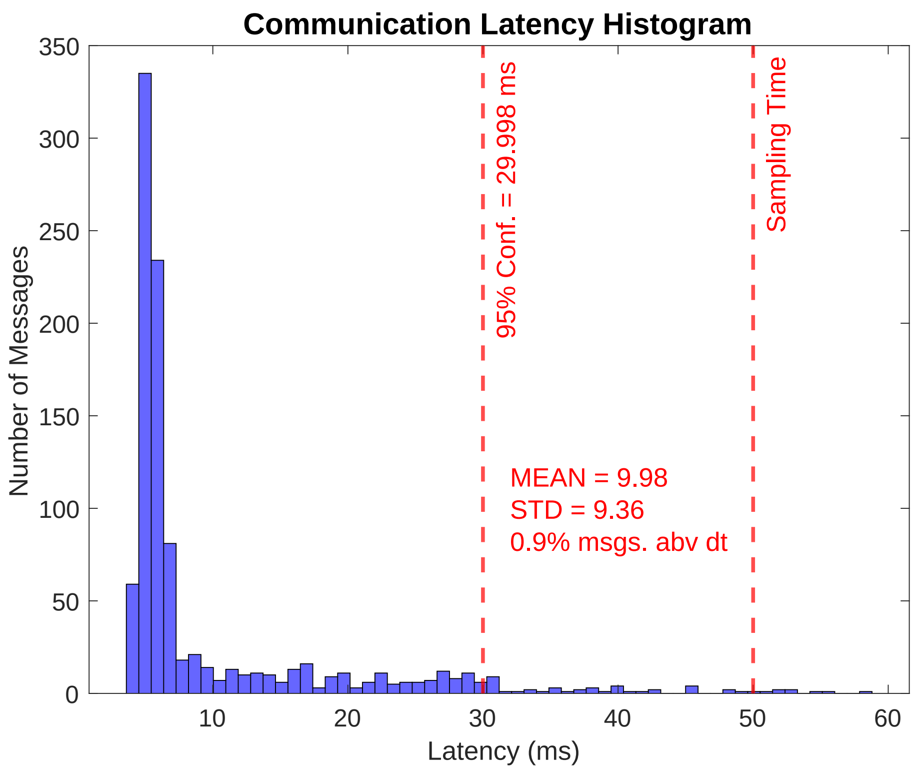
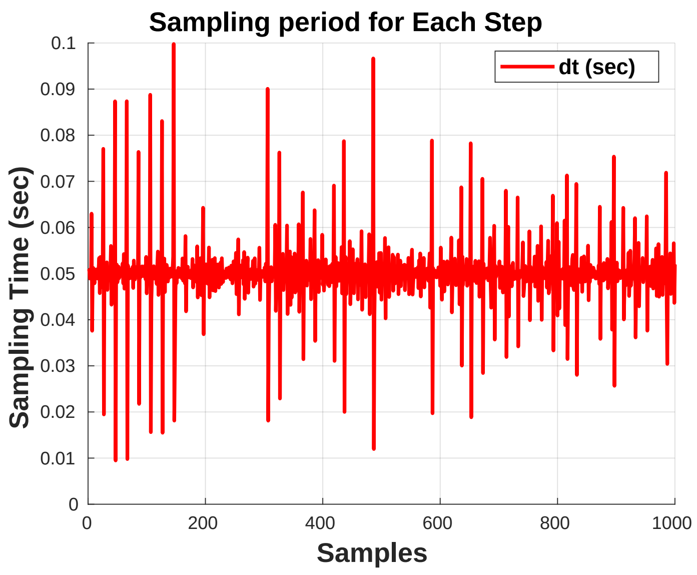


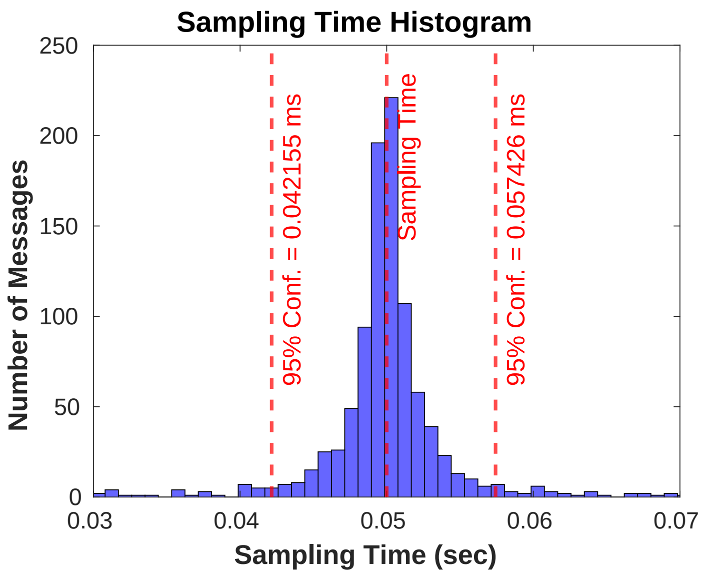
Figure 1.6: Communication Latency Histogram. The latency (RTT/2) of the 1000 message packages has 9.98 seconds mean value. 0.9% of the messages' latency values are above the sampling time of the control algorithm, which is 50 ms.

sampling rate for the control loop is reasonable. This experimental outcome justifies the sampling rate selection for the present study.

The frequency of MAVLink messages can be configured from the host computer. However, since the MavLink protocol does not operate on a real-time operating system, the sampling rate may vary due to internal operations within the Raspberry Pi. Additionally, as depicted in Figure 1.6, the Wi-Fi module can introduce latency into the system. For this experiment, the frequency of MAVLink messages is set to 20 Hz. The host computer receives messages without any wait command, and the delivery time of the messages is recorded to determine the robot's sampling rate distribution. Figure 1.7 displays the distribution of the sampling times. The host computer monitors 1,000 messages, with 900 of them falling within the range of 42 ms to 57 ms. These results demonstrate that a considerable amount of data is received within the 50 ms sampling time. To enhance the sampling rate's robustness against latencies,



(a)

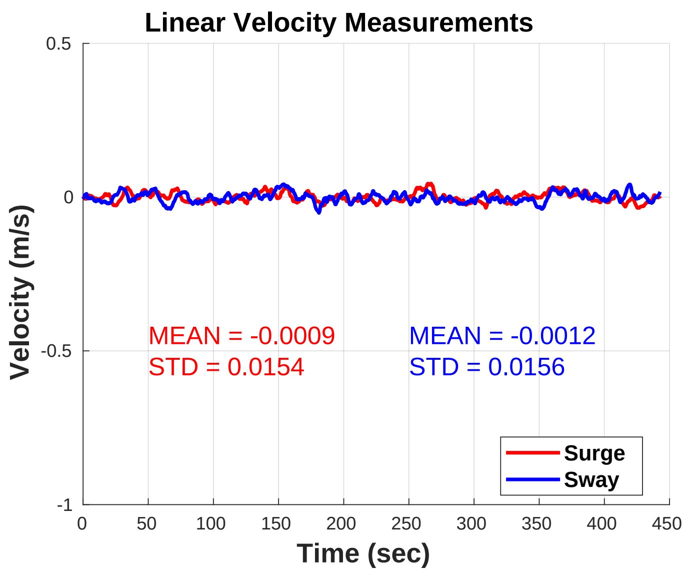


(b)

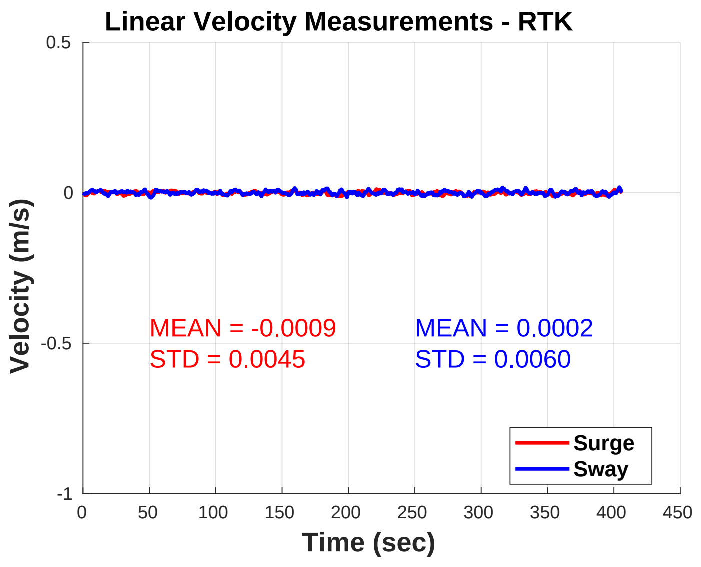
Figure 1.7: a : Sampling time changes in each step. b : Sampling time histogram. Pixhawk is set to send MAVLink messages at 20 Hz. frequency. 96% confidence parameters finds 5% outliers for both minimum and maximum sampling times. 90% of the total messages are sent in $[42, 57]$ ms time interval

the control algorithm loop in the host computer is fixed to 20 Hz, in addition to the MAVLink data stream frequency settings.

1.4.6 Velocity Measurements



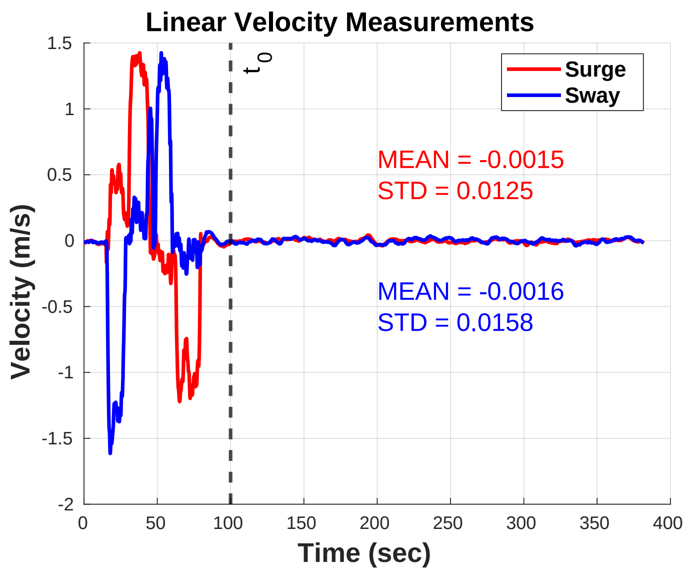
(a)



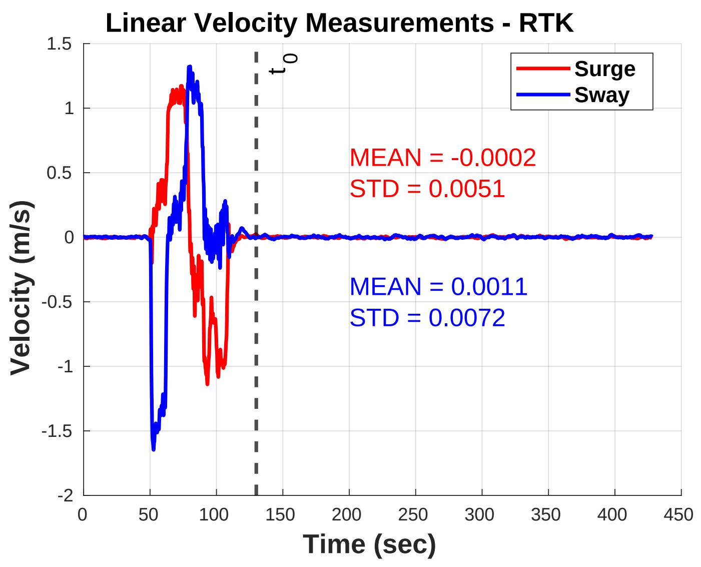
(b)

Figure 1.8: a : Linear velocity measurements of the robot during the stationary case without RTK b : Linear velocity measurements of the robot during the stationary case with RTK

In this section, the robot's measurement signals are studied. As the USV moves in 3 DoF, body frame positions and velocities for these axes are assumed to be the system's



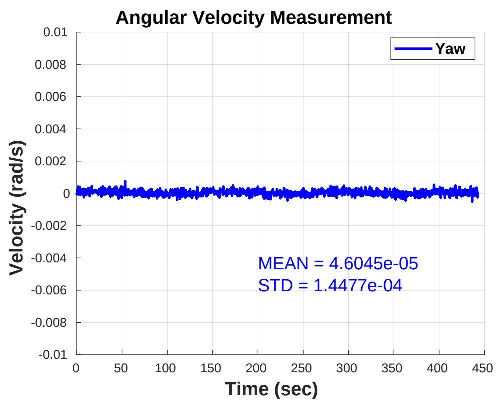
(a)



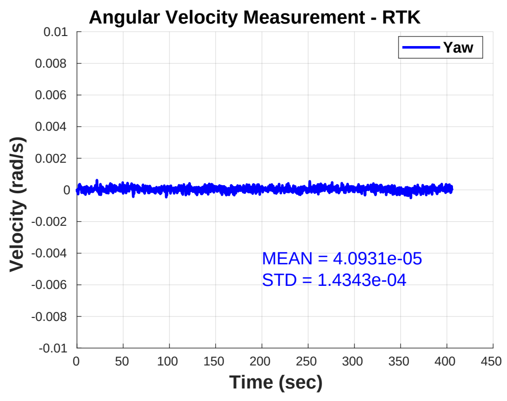
(b)

Figure 1.9: a : Linear velocity measurements of the robot after the square motion without RTK b : Linear velocity measurements of the robot after the square motion with RTK

states. ArduSub measures the states by applying Extended Kalman Filter to GPS (NEO-M8P GNSS module) and IMU (Invensense MPU 6000 3-axis accelerometer and gyroscope) measurements. Pixhawk updates each state and transfers the most

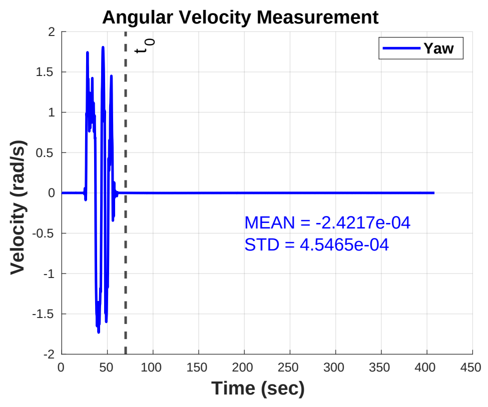


(a)

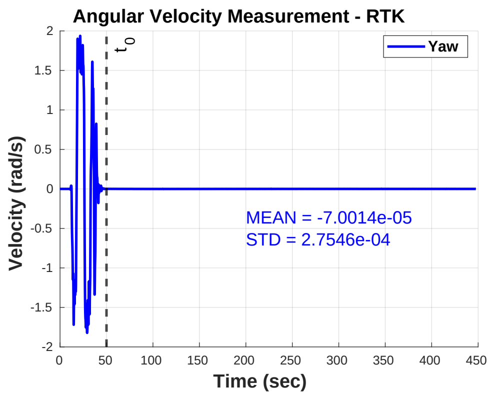


(b)

Figure 1.10: a : Angular velocity measurement of the robot during the stationary case without RTK b : Angular velocity measurement of the robot during the stationary case with RTK



(a)



(b)

Figure 1.11: a : Angular velocity measurements of the robot after the rotational motion without RTK b : Angular velocity measurements of the robot after the rotational motion with RTK

recent measurements to the ground station with the MAVLink protocol.

Measurement noise plays a crucial role in implementing stable control strategies. The characteristics of measurement noise can vary between stationary and moving cases, as the robot's motion can influence sensor measurements. Both scenarios can be analyzed by employing a ground truth sensor and examining the discrepancies between the ground truth and the actual robot's measurements. As stated in [41], measurements taken with the RTK module can serve as ground truth information. Nonetheless, we utilize the measurements from the RTK module in actual experiments. In this study, we conduct several measurement performance experiments to infer the deviations in the robot's state measurements. We consider three cases during the experiment: the robot in a stationary position, the robot moving in a rectangular trajectory, and the robot undergoing rotational motion.

As depicted in Figure 1.8, utilizing the RTK module enhances the accuracy of linear velocity measurements. Without the RTK module, the standard deviation of the surge and sway velocities are 0.0154 and 0.0156, respectively. However, with the RTK module, these values decrease to 0.0045 and 0.0060. Figures 1.8 and 1.9 illustrate the impact of the robot's excitation on the measurement unit. In Figure 1.9, t_0 represents the point at which the excitation ended. The mean and standard deviation are computed from t_0 to the final time. Figures 1.8 and 1.9 demonstrate that the robot's movement in the surge and sway axes does not contribute to accumulated measurement errors in the velocity states.

Figures 1.10 and 1.11 present the angular velocity measurements for the yaw axis when the system is stationary and when it experiences rotational motion, respectively. A slight change is observed in the mean and standard deviation. Notably, the orders of magnitude for these parameters are smaller than those in the linear velocity measurements. Furthermore, the robot's rotation does not lead to an accumulation of biased errors in yaw rate measurements. The RTK's enhancement of the yaw rate measurements is not substantial, given its role as a correction sensor for GPS. This experiment was conducted in collaboration with another student focusing on feedback motion planning methods for this USV. The impact of positional measurement noises on the robot is discussed further in his work [42].

1.5 Organization of The Thesis

Chapter 1 delineates the motivation and background of this study, specifying the problems addressed. It includes a literature review of the strategies implemented and provides a succinct overview of the technical features of the robot used in the experiments. Further, it involves an analysis of the robot's communication latency and an examination of the noise distribution for the velocity signals.

Chapter 2 develops a mathematical model of a USV with 3 DoF based on the Newton-Euler equations. It explicates the linear and nonlinear terms in the model and outlines the assumptions made to simplify the model. The process of linearizing the derived model around a reference trajectory as a time-varying system is also detailed.

Chapter 3 describes the data collection phase to acquire the parameters constituting the robot's mathematical model. It explains the system identification method applied to the collected experimental data and describes the constrained cost function used for parameter estimation. The application of the k-fold cross-validation method to procure parameter distributions for different evaluation datasets is detailed. The results from the cross-validation are presented along with a selected parameter set for the robot, complete with their evaluation costs.

Chapter 4 elucidates the velocity control methods devised for the robot to adhere to specified reference trajectories and details the mathematical formulas and implementation techniques of the PI and TV-LQR algorithms.

Chapter 5 presents the test results of the robot, based on both the SITL and actual experiments, the latter of which were conducted with the robot in the METU Yalıncak DSI lake.

Chapter 6 outlines the conclusions drawn from this study and discusses planned future endeavors.

Finally, the appendix includes TV-LQR derivations for the finite horizon continuous-time cost function.

CHAPTER 2

MATHEMATICAL MODEL

2.1 Approach

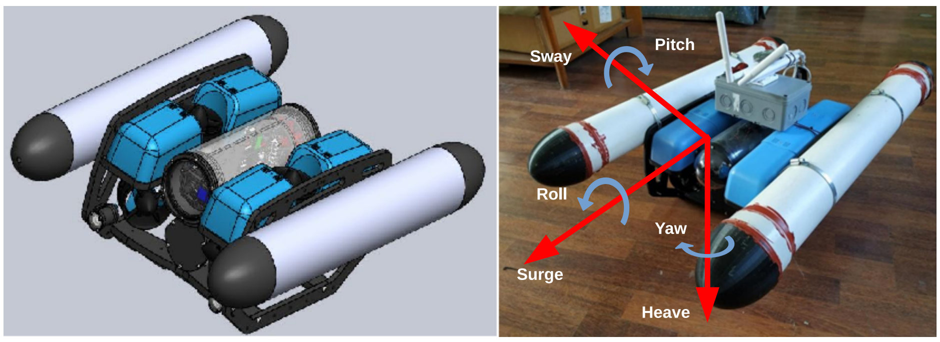


Figure 2.1: Three-dimensional representation of the robot and each body axis of the USV. 3-DoF USV model included the surge, sway, and yaw axes.

This section elucidates the mathematical modeling of the USV. As shown in Figure 2.1, despite the USV's body frame having the potential for six degrees of freedom (DoF), the robot effectively moves in three DoF: surge, sway, and yaw. The subscript 'b' denotes the body frame, while 'n' stands for the global NED frame. To promote clarity, we first define the notations used in the equations. The rotation matrix of the robot, which serves to transform between the body and NED frames, is discussed subsequently. We then derive the system's equations of motion and expound upon the physical meanings of the coefficients within these equations. We finalize the dynamic model by applying simplification methods to the robot model. Linearization of the dynamic model around a trajectory ultimately yields a time-varying state-space model. The derivation of this time-varying system is explained in this section.

When we look at environmental forces, hydrodynamic, Coriolis and centripetal, gravitational, and buoyancy forces can be given as examples [1]. Besides, the force of the airflow and the wave forces were used as disturbances. Forces and the torque created by motor inputs are modeled in the equations.

2.2 Notations

3-DoF USV robot can move in the surge, sway, and yaw direction. The surge axis (x-axis) denotes the robot's motion in a forward or backward direction. Transverse movements take place in the sway axis (y-axis). The yaw axis (ψ-axis) indicates the robot's rotation around the z-axis. These 3 DoFs can be represented in a vector form with SNAME standard symbolic representations [20]. Table 2.1 describes each axis's position, velocity, and force parameters. ψ-axis has an angle interval between $[-\pi, \pi)$.

Table 2.1: 3 DoF SNAME notations

3-DoF	Force / Torque	Velocity	Position / Angle
1-Surge	X	u	x
2-Sway	Y	v	y
3-Yaw	N	r	ψ

The vector representation of the robot's motion offers several advantages. For instance, the skew-symmetry and positive definiteness of the coefficient matrices provide insights into the robot's stability, and vector operations expedite algebraic operations. The position vector of the robot is denoted as $\boldsymbol{\eta} = [x, y, \psi]^T$, according to the notation in Table 2.1. The velocity vector can be represented as $\boldsymbol{v} = [u, v, r]^T$, and the forces and torque vector can be depicted as $\boldsymbol{\tau} = [X, Y, N]^T$. For instance, the robot's position in the body frame is symbolically represented as $\boldsymbol{\eta_b} = [x_b, y_b, \psi_b]^T$, while its position in the NED frame is given by $\boldsymbol{\eta_n} = [x_n, y_n, \psi_n]^T$.

Nine parameters are used to represent the unknown coefficients of the mathematical model. The physical meanings of these parameters are given below,

- m_x (kg) : Mass coefficient of the x-axis
- m_y (kg) : Mass coefficient of the y-axis
- I_ψ (kgm^2) : Inertia coefficient around z-axis
- d_{l1} (Ns/m) : Linear damping coefficient of x-axis
- d_{l2} (Ns/m) : Linear damping coefficient of y-axis
- d_{l3} (Ns/rad) : Linear damping coefficient of yaw-axis
- d_{nl1} (Ns^2/m^2) : Nonlinear damping coefficient of x-axis
- d_{nl2} (Ns^2/m^2) : Nonlinear damping coefficient of y-axis
- d_{nl3} (Ns^2/rad^2) : Nonlinear damping coefficient of yaw-axis

These coefficients feature prominently in the matrices of the dynamic equations. As discussed later in Chapter 3, the system identification method is employed to estimate the robot's unknown coefficients. Each degree of freedom necessitates the estimation of three parameters: mass or inertia, linear damping, and nonlinear damping coefficients.

2.3 Euler Angle Transformation

In this section, the kinematic properties of the USV are discussed. Motion vectors of the robot can be written in different reference frame forms. Dynamic equations of the robot are generally written in the body frame. However, the velocity data sent from the robot uses the NED frame. The body frame chooses an arbitrary point on the robot as the origin, and axes are defined according to the robot's body shape (Figure 2.1). NED frame covers the tangent plane on Earth's surface. The origin of the NED frame can be an arbitrary point on this surface. The direction through the Earth's north pole defines the positive direction of the surge axis. The sway axis is parallel to the equator line. The yaw axis is the rotation axis of the z-axis which increases down to the center of the Earth. The right-hand rule can find the positive direction of the yaw axis. It is necessary to transform two frames. Euler-angle transformation can rotate vectors from one frame to another according to the rotation axis. The surge and sway axes describe the translational motion of the USV, yaw axis is the rotational axis of the robot. One can see that $R_z(\psi)$ Euler angle rotation matrix is necessary for the transformation.

Operating as a planar vehicle, the robot rotates exclusively around the z-axis, resulting in the angle transformation between the NED and body frames, as illustrated below.

$$\boldsymbol{v^n} = R_b^n(\Theta)\boldsymbol{v^b}$$
$$R_b^n(\Theta) = R_z(\Psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$R_z(\psi)$ is an orthogonal matrix. Any vector can be transferred from the NED frame to the body frame by multiplying with the transpose of this rotation matrix ($R_z(\psi)^T$). Normally, the last column of the $R_z(\psi)$ converts the z-axis in 6-DoF representation. In this case, it is used to transform the yaw axis, which is identical in both frames.

2.4 Differential Equations

Equation 2.1 provides the general form of the differential equation for all velocity components of each body axis of the robot, as delineated in [1]. This equation incorporates three distinct matrix forms for the coefficients. The M matrix accounts for mass and inertia coefficients, while the $C(v)$ matrix embodies the Coriolis and centripetal terms. The $D(v)$ matrix contains both linear and nonlinear damping coefficients, addressing the water's drag forces. The vectors $\boldsymbol{g(\eta)}$ and $\boldsymbol{\tau}$ represent hydrostatic forces and motor forces, respectively.

$$M\dot{\boldsymbol{v}} + C(\boldsymbol{v})\boldsymbol{v} + D(\boldsymbol{v})\boldsymbol{v} + \boldsymbol{g(\eta)} = \boldsymbol{\tau} \tag{2.2}$$

2.4.1 Mass matrix

It is assumed that mass matrix is diagonal as given below,

$$M = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & I_\psi \end{bmatrix} \tag{2.3}$$

M is the mass and inertia matrix of the robot. The robot's rotation in the NED frame creates a hydrodynamic added mass. This M matrix is the sum of added mass and the rigid body mass of the USV. Since the magnitudes of the diagonal elements dominate the system, non-diagonal elements are neglected.

2.4.2 Coriolis and Centripetal Matrix

$C(\boldsymbol{v})$ represents Coriolis and centripetal matrix. It has two components, such as rigid body matrix $C_{RB}(\boldsymbol{v})$ and added mass matrix $C_A(\boldsymbol{v})$ (Eqn. 2.4). Let us assume that the origin of the body frame is the center of gravity. Hence the $C(\boldsymbol{v})$ matrix can be calculated using the M matrix terms as given below.

$$\begin{gathered} C(\boldsymbol{v}) = C_{RB}(\boldsymbol{v}) + C_A(\boldsymbol{v}) \\ C(\boldsymbol{v}) = \begin{bmatrix} 0 & 0 & -m_y v \\ 0 & 0 & m_x u \\ m_y v & -m_x u & 0 \end{bmatrix} \end{gathered} \tag{2.4}$$

In the context of low-speed operations, the Coriolis and centripetal effects become negligible. Given that the USV falls within the Semi-displacement vehicles category according to the Froude number, it is reasonable to disregard the $C(\boldsymbol{v})$ matrix. Notably, the $C(\boldsymbol{v})$ matrix is the sole matrix with non-zero off-diagonal elements, resulting in coupled system dynamics. By modeling the system with decoupled equations of motion, we significantly reduce the complexity inherent to system identification. Consequently, the system dynamics are modeled such that each axis' differential equation contains two parameters: the velocity of the corresponding axis and the external force acting on that axis. Furthermore, three unknown coefficients are considered: mass/inertia, linear damping, and nonlinear damping coefficients.

2.4.3 Hydrodynamic Damping Matrix

The hydrodynamic damping matrix has two components: linear damping coefficients and nonlinear damping coefficients. Wave excitation can create linear potential damping. Friction between the robot's surface and the water causes linear and quadratic damping effects. Especially at the corner points of the robot, vortex shedding may occur. In this case, vortex shedding damping exposes the robot to a nonlinear dumping effect. Considering these damping effects, one can model the damping matrix below.

$$D(\boldsymbol{v}) = \begin{bmatrix} d_{l1}u + d_{nl1}|u|u & 0 & 0 \\ 0 & d_{l2}v + d_{nl2}|v|v & 0 \\ 0 & 0 & d_{l3}r + d_{nl3}|r|r \end{bmatrix} \tag{2.5}$$

2.4.4 Hydrostatic Matrix

Hydrostatic forces contain buoyancy force and gravitational force. These forces are called restoring forces. In equilibrium, these two forces cancel each other on the z-axis (Eqn. 2.6). If the robot is exposed to a positive external force on the z-axis, the buoyancy will increase as the floating body's sinking volume increases. One can see in Equation 2.7 that a spring force is applied to the robot because the $(W - B)$ becomes a function of the displacement in the z-axis. Hence, the spring structure of mass-spring-damper models and the effect of hydrostatic coefficients are similar.

$$\begin{aligned} B &= \rho g \nabla \\ W &= mg \end{aligned} \tag{2.6}$$

In Equation 2.6, ρ is the density of the water. ∇ shows the total volume of the submerged part of the robot, and g is the gravitational acceleration. m indicates the total mass of the USV.

$$Z \simeq -\rho A_{xy} z \tag{2.7}$$

In Equation 2.7, A_{xy} represents the surface area of the robot in the xy plane. Figure 2.2 represents the metacentric stability of the hydrostatic forces on the zy plane [1]. G is the center of the mass point of the robot, and B is the center of the buoyancy force. A line passes between these two points. A new line is formed when the roll angle changes with the effect of hydrostatic force. The intersection point of these two lines is called M_T. The intersection point formed by the pitch angle change is called M_L.

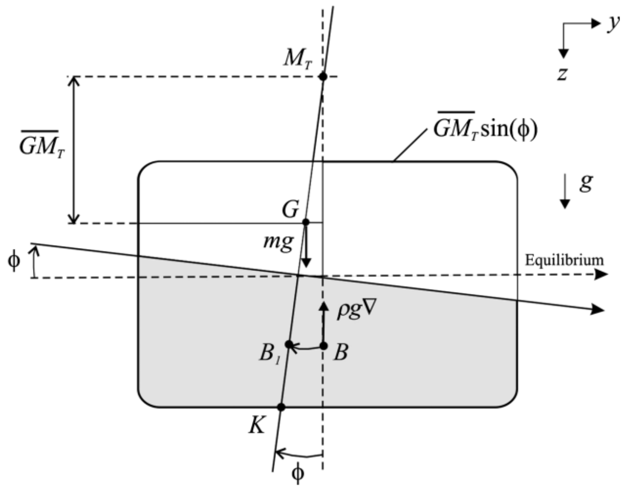


Figure 2.2: Metacentric stability of the robot in xy plane [1]

According to Fossen [1], $\boldsymbol{g}(\boldsymbol{\eta})$ can be written as given in Equation 2.8. Since the robot's velocity is small, the USV makes small angles in the roll and pitch axis. Hence the 3-DoF USV model does not have rotation around the x and y axis in the mathematical model. The $\boldsymbol{g}(\boldsymbol{\eta})$ coefficients converge to zero in this situation. $\overline{GM_T}$ is calculated by roll angle displacement, described in Figure 2.2. $\overline{GM_L}$ vector can be calculated with pitch angle displacement in xz planar coordinate system by applying the similar steps for the $\overline{GM_T}$ vector.

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W-B)sin(\theta) \\ -(W-B)cos(\theta)sin(\phi) \\ \rho g\nabla(-\overline{GM_L}cos(\theta) + \overline{GM_T})sin(\phi)sin(\theta) \end{bmatrix} \simeq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.8}$$

Upon integrating the specified modeling operations and simplifications, we can derive the decoupled equations of motion for the three axes along which the robot moves. As detailed below, the differential equations for the x-axis, y-axis, and the ψ angle serve as the cornerstone for the physical modeling of the real robot. The following system identification method relies heavily on this dynamic model.

$$\begin{aligned} m_x\dot{u} + d_{l1}u + d_{nl1}|u|u &= F_x \\ m_y\dot{v} + d_{l2}v + d_{nl2}|v|v &= F_y \\ I_\psi\dot{r} + d_{l3}r + d_{nl3}|r|r &= \tau_\psi \end{aligned} \tag{2.9}$$

2.5 Thruster Model

The motors, by default, interpret inputs as PWM signals ranging from 1200 to 1900, where any integer within this range modulates the duty cycle of the PWM signal. For the sake of simplicity, we eliminate the offset term and posit that the robot's motor inputs accept values between -400 and 400. The motor achieves maximum velocity in both forward and reverse directions with 400 and -400, respectively, while a dead-zone region exists between the $[-25, 25]$. This motor input structure translates into torque values, yielding a scaling factor between PWM gains and motor thrust value. This relationship approximates a linear function as indicated in Equation 2.10, where F_m represents the motor force and u_m signifies the motor input PWM gain.

$$F_m = 0.1023u_m \tag{2.10}$$

$$\begin{bmatrix} F_x \\ F_y \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} -0.7070 & -0.7070 & 0.7070 & 0.7070 \\ -0.7070 & 0.7070 & -0.7070 & 0.7070 \\ -0.1888 & 0.1888 & 0.1888 & -0.1888 \end{bmatrix} \begin{bmatrix} F_{m1} \\ F_{m2} \\ F_{m3} \\ F_{m4} \end{bmatrix} \tag{2.11}$$

The motors, mounted on the robot at 45-degree angles as depicted in Figure 1.2, generate thrust transformed into force and torque parameters within the robot's body axis through angle transformations. A transformation matrix (Equation 2.11) is obtained by considering the robot orientations. Given that the robot possesses more motors than its three degrees of freedom, the resulting matrix is non-square and does not have a defined inverse. This necessitates the use of control allocation to convert reference force and torque values into corresponding motor inputs for each degree of freedom.

2.6 State Space Model

The conversion of the differential equations into a state-space model provides a critical step in the system linearization process around a reference trajectory. The state vector of the USV, situated within the body frame, comprises six elements. These elements encompass the surge and sway positions and the yaw angle as positional attributes. Additionally, the surge and sway velocity, along with the yaw axis's angular rate, constitute the velocity components. As delineated in Section 2.4, the robot commands two force inputs for translational motion and one torque input for the rotational axis. Consequently, the state vector, input vector, and state equation are defined as presented in Equations 2.12 and 2.13.

$$\boldsymbol{x} = \begin{bmatrix} x & y & \psi & u & v & r \end{bmatrix}^T \text{ and } \boldsymbol{u} = \begin{bmatrix} F_x & F_y & \tau_\psi \end{bmatrix}^T \tag{2.12}$$

$$f(\boldsymbol{x},\boldsymbol{u}) = \begin{bmatrix} u \\ v \\ r \\ \frac{1}{m_x}(F_x - d_{l1}u - d_{nl1}u|u|) \\ \frac{1}{m_y}(F_y - d_{l2}v - d_{nl2}v|v|) \\ \frac{1}{I_\psi}(\tau_\psi - d_{l3}r - d_{nl3}r|r|) \end{bmatrix} \tag{2.13}$$

$$\dot{x} = f(\boldsymbol{x},\boldsymbol{u})$$

2.7 Linearization

There are two types of linearization. The system can be linearized around an arbitrary equilibrium point or a time domain reference signal. Linearizing around an equilibrium point approximates the nonlinear model as a linear time-invariant system. The linear model gives erroneous results as the robot moves away from the equilibrium point. This method ignores the time-dependent variation of the trajectory followed by the robot. The robot's trajectory is sampled discretely, and the nonlinear model must be linearized again at each sample point. When we linearize the system around a trajectory, the nonlinear model turns into a linear time-varying model. In this model, the state matrices of the system, $A(t)$ and $B(t)$, change over time. Once the reference trajectory is determined, it is sufficient to linearize the system once. Thus, the model of the robot is linearized at each time step on the reference signal. At the same time, the computational power in the loop of the robot's control algorithm is reduced.

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{2.14}$$

The nonlinear model provided above allows for deriving a linearized system model along the trajectory for each time step. By linearizing around a nominal trajectory [43], we can generate a linear time-varying state-space model for the USV. This model, in turn, facilitates the design of a TV-LQR controller, enabling the USV to follow the intended trajectory effectively.

By using the nonlinear state space model in the equations 2.12 and 2.13, one can define a new state vector $\boldsymbol{\delta_x}(t) = \boldsymbol{x}(t) - \boldsymbol{x_d}(t)$ such that the $\boldsymbol{x_d}(t)$ is the time-varying desired trajectory. A similar change of variables approach can be used for the input vector such that $\boldsymbol{\delta_u}(t) = \boldsymbol{u}(t) - \boldsymbol{u_d}(t)$. $\boldsymbol{u_d}(t)$ represents the reference input trajectory. The reference input signal is calculated using the system dynamics and the reference state vector. It is aimed to find $A(t)$ and $B(t)$ matrices of these changed input and state vectors for the equation 2.15.

$$\dot{\boldsymbol{\delta_x}}(t) = A(t)\boldsymbol{\delta_x}(t) + B(t)\boldsymbol{\delta_u}(t) \tag{2.15}$$

One can find the $A(t)$ and $B(t)$ as given in the equation 2.16 below by calculating the Jacobian matrix of the state function. Evaluating the Jacobian matrix on reference signals gives the $A(t)$ and $B(t)$ matrices.

$$\begin{aligned} A(t) &= \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}(t)=\boldsymbol{x_d}(t),\boldsymbol{u}(t)=\boldsymbol{u_d}(t)} \\ B(t) &= \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\boldsymbol{x}(t)=\boldsymbol{x_d}(t),\boldsymbol{u}(t)=\boldsymbol{u_d}(t)} \end{aligned} \tag{2.16}$$

As a result, the linear time-varying state-space model of the USV can be obtained as given in Equations 2.17 and 2.18, respectively.

$$A(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-1}{m_x}(d_{l1} + d_{nl1}\frac{u_d(t)^2}{|u_d(t)|} + d_{nl1}|u_d(t)|) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{m_y}(d_{l2} + d_{nl2}\frac{v_d(t)^2}{|v_d(t)|} + d_{nl2}|v_d(t)|) & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-1}{I_\psi}(d_{l3} + d_{nl3}\frac{r_d(t)^2}{|r_d(t)|} + d_{nl3}|r_d(t)|) \end{bmatrix}, \tag{2.17}$$

$$B(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{m_x} & 0 & 0 \\ 0 & \frac{1}{m_y} & 0 \\ 0 & 0 & \frac{1}{I_\psi} \end{bmatrix} \tag{2.18}$$

CHAPTER 3

SYSTEM IDENTIFICATION

3.1 Approach

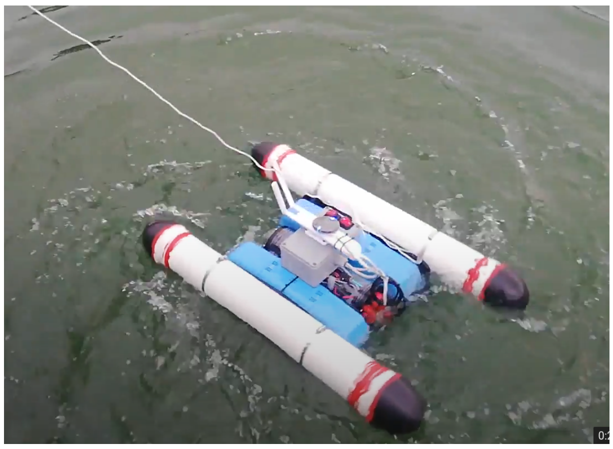


Figure 3.1: The USV's view during the experiments in METU Yalıncak Lake.

Several methods are available for identifying underwater and surface vehicles, as discussed in studies by [40], [44], and [38]. The primary objective of such identification is to facilitate the application of model-based control algorithms to the USV. Chapter 2 elaborates on how the USV embodies three independent differential equations for each axis—surge, sway, and yaw. These equations comprise three parameters: mass, linear damping, and nonlinear damping coefficients. Data gathered from the METU Yalıncak Lake (see Figure 3.2) inform the estimation of these nine parameters. For a duration of 20 seconds, constant motor inputs ranging from %62.5 to %-62.5 PWM gains are applied to each axis to ascertain the vehicle's constant input response. Subsequently, three types of sinusoidal signals ($M.sin(2\pi.0.1t)$, $M.sin(2\pi.0.075t)$, $M.sin(2\pi.0.05t)$), each bearing different frequencies, are employed to stimulate the system with varying input types. Here, 'M' signifies the gain corresponding to %62.5 of the maximum PWM gain. In addition, some closed-loop control experiments contribute data for parameter estimation. Despite the inclusion of the controller algorithm in these experiments, it is possible to estimate the USV parameters by considering the controller output as an arbitrary input signal for the robot during the closed-loop experiment. Integrating data from closed-loop, constant input, and sinusoidal experiments enhances the diversity of data available for the parameter estimation phase. The process of estimating the robot's unknown parameters involves the amalgamation of three types of experiments, selecting 30 sets of experiments. These sets are subsequently divided into ten folds to implement a k-fold cross-validation algorithm. Implementing the cross-validation algorithm ensures the separate training and validation datasets provision. The parameter estimation is facilitated through the creation of a constrained nonlinear cost function. The function is minimized using the "fmincon" algorithm, a method of nonlinearly constrained optimization [45].

3.2 System Identification Experiments

Applying constant force input to the system cancels out the derivative terms of the velocity. Linear and nonlinear damping coefficients are effective during these experiments' settling times. The mass or inertia coefficient is only effective during the transient response time of these experiments.

Figure 3.2 presents the yaw axis's open-loop constant velocity responses. Notably, the yaw axis experiments are minimally affected by lake current and wind noise, attributable to the USV's rotational motion during these tests. As constant wind and current forces exert their influence in the robot's xy-plane, the rotation around the z-axis experiences the least disturbance.

Sinusoidal signals excite the system with periodically changing thrust inputs. Figure 3.3 shows the USV's response for different sinusoidal PWM gain inputs with different frequencies. For each axis, PWM gains are applied for one direction only, and the other two motion axes' inputs are assumed to be zero.

Closed-loop experiments, integral to this study, are conducted during the robot's tra-

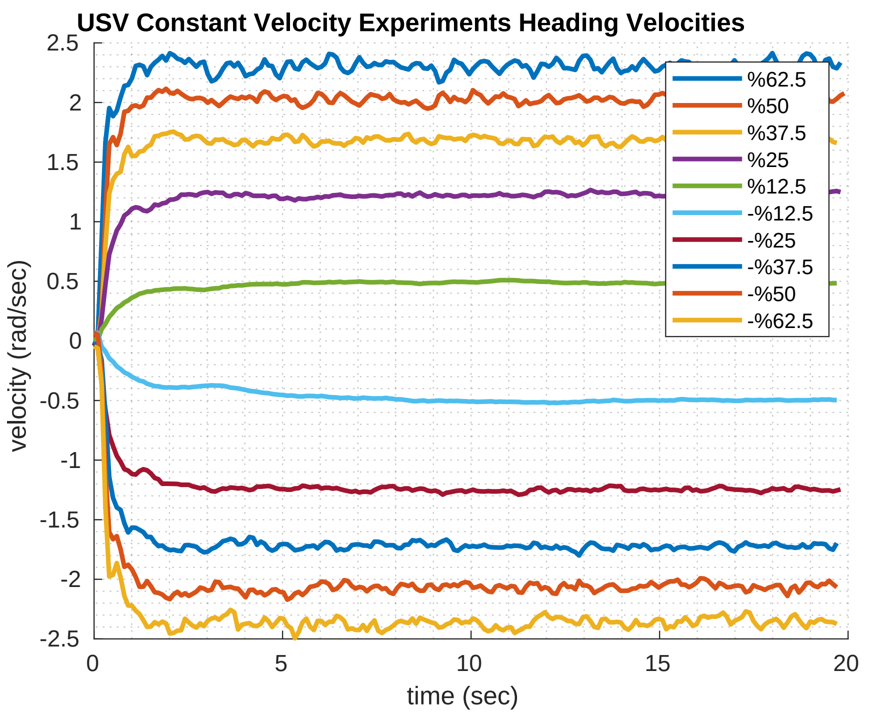


Figure 3.2: Velocity signals for the open-loop constant motor gain experiments for the heading axis. Any controller type is not applied during the constant input experiments. Labels indicate the percentage of the applied gain over maximum gain value ratio.

jectory tracking tests. These tests necessitate data collection from sinusoidal and piecewise constant velocity signal tracking experiments for both TV-LQR and PI controller algorithms. In this setup, the controller output is assumed to be an input signal for the system dynamics. Given that the control algorithms are designed to track reference velocity signals across all motion axes, the inputs for these three motion axes might not always be zero. This implies that the same experimental result could be employed to estimate the unknown system dynamics for each motion axis.

Drawing from a pool of 30 distinct experiments for each motion axis, we assemble the dataset for the optimization algorithm. We employ the k-fold cross-validation algorithm to partition the data into k groups or folds. Each group is designated as validation data once, while the remaining k-1 groups serve as training data for the optimization algorithm. This approach ensures that each parameter is estimated k times, and each estimation progression is evaluated with different experiments. In

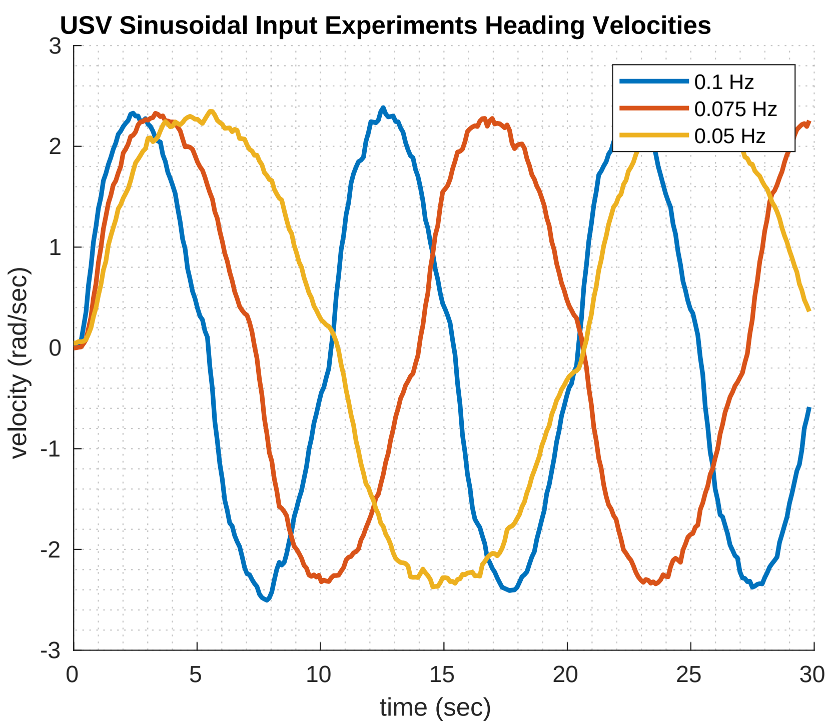


Figure 3.3: Velocity signals for the open-loop sinusoidal motor gain experiments for the heading axis. Any controller type is not applied during the this experiments. Labels indicate the frequency of the applied motor inputs. Amplitude of the motor inputs is 62.5% of the maximum gain limit of the motors.

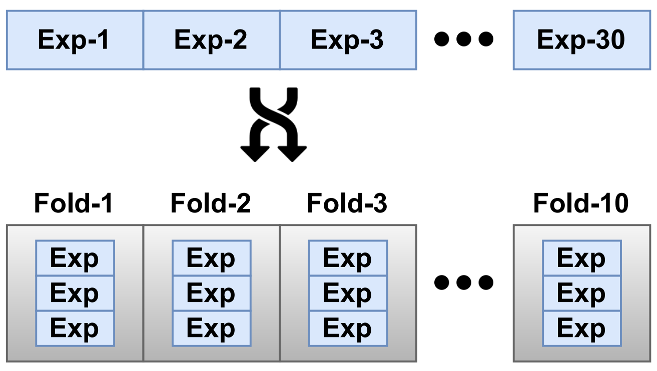


Figure 3.4: Fold generation with all experiments. 30 experiments are randomly split into 10 folds

this study, we set k at 10, and randomly divide the 30 experiments into ten folds. Each fold encompasses three randomly selected experimental data, as Figure 3.4 depicts. We fix the random function seed parameter to ensure reproducible results.

3.3 Cost Function

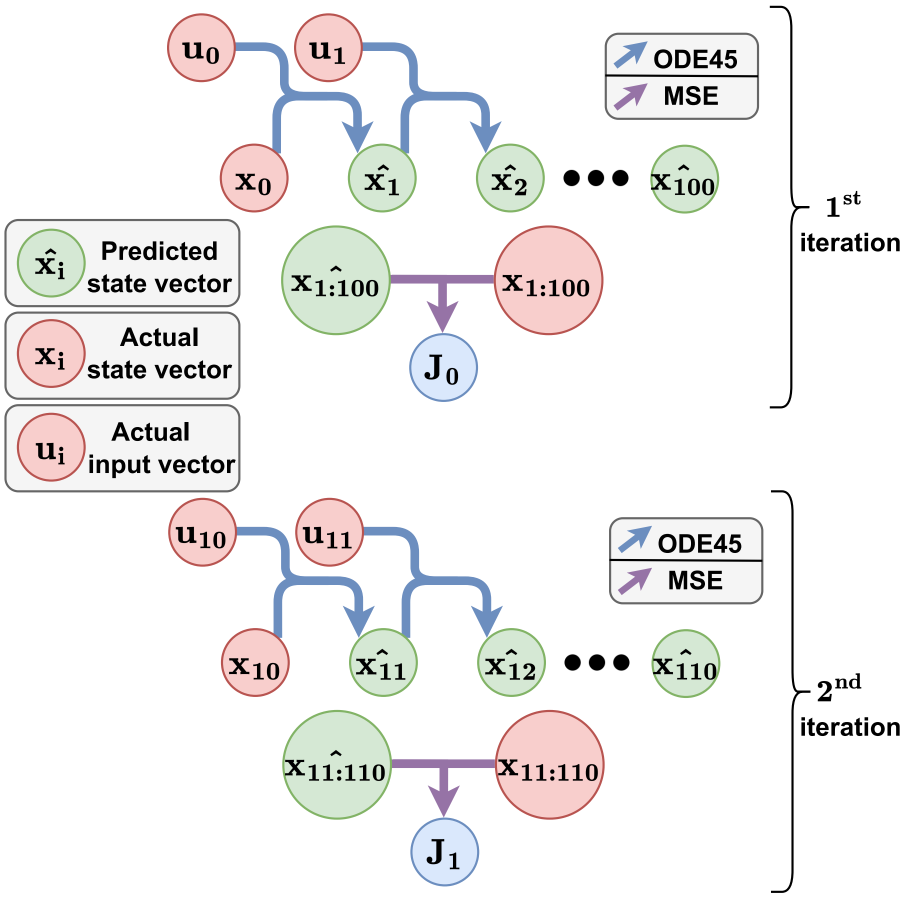


Figure 3.5: Block diagram for the cost calculation as first and second iterations. The prediction horizon is 100, and the mean squared error (MSE) calculates the error between the actual experiment and the prediction horizon.

Nonlinear optimization methods can be used to determine the parameters of the mathematical model using a cost function that compares real-world models and the estimated numerical solution of the mathematical model obtained by the ordinary differential equation solver. The general form of the cost function is given below. n is the number of estimated parameters. In this study, n is assumed to be three for each motion axis.

$$\min_{\boldsymbol{x}^*} f(\boldsymbol{x})$$
$$\boldsymbol{x} = (x_1, x_2, x_3, ..., x_n) \in \mathbb{R}^n \tag{3.1}$$

Constraints for each parameter are given below. All parameters are considered positive, as the negative model parameters are physically meaningless.

$$1000 > \boldsymbol{x} > 0 \tag{3.2}$$

Given the decoupled nature of the differential equations for each axis, we can estimate the system parameters of the surge, sway, and yaw axes independently. These parameters include the mass coefficient, linear damping coefficient, and nonlinear damping coefficients. Once estimated, these differential equations are solved via numerical calculations. We then compare the predicted and actual models using the mean squared error (MSE).

The algorithm for the cost function is outlined below. Here, τ represents the number of steps needed to form a window in all samples, serving as a prediction horizon for each selected actual data point (see Figure 3.5). The cost function solves the system dynamics for each selected sampling point, considering the corresponding sampling point $x_{actual}(j)$ as the initial point and τ as the number of time steps. x_{actual} and $x_{prediction}$ denote the actual experiment samples and the numerical solution of the predicted model, respectively. $J_{exp}(j)$ is the mean squared error between the actual and the prediction samples. x_0 represents the initial point for the ODE solver.

As shown in Figure 3.6, each prediction horizon is applied to every ten sampling points of the experiment. Increasing the selected sampling point frequency requires more computational power to evaluate the cost function. Adjusting the prediction horizon affects the noise factors—increasing it dominates the initial point noises while decreasing it dominates environmental noises.

MATLAB "fmincon" optimization function is used for the parameter estimation. This built-in function optimizes nonlinear constrained cost functions by applying Interior-Point algorithm. Interior-Point method obtains Barrier functions for the constrains of the optimization function. This Barrier functions approaches to the infinity as parameters approaches to the constrains.

Algorithm 1 Cost function

```
1:  $\tau \leftarrow$ fixed time step number
2:  for i = 1: Number of experiments do
3:      $x_{actual} \leftarrow$ Real experiment states s.t $x_{actual} \in \mathcal{R}^{2\times m}$
4:      $u_{actual} \leftarrow$ Real experiment inputs s.t $u_{actual} \in \mathcal{R}^{m}$
5:      for j 1 : 10 : $m$ do
6:          for k 1 : $\tau$ do
7:              $x_0 \leftarrow x_{actual}(j)$
8:              $x_{prediction}(k) \leftarrow ode45(x_0, u_{actual}(k))$
9:          end for
10:         $J_{exp}(j) \leftarrow mse(x_{prediction}, x_{actual}(j : j + \tau))$
11:     end for
12:     $cost_i \leftarrow mean(J_{exp})$
13: end for
14: return $mean(cost)$
```

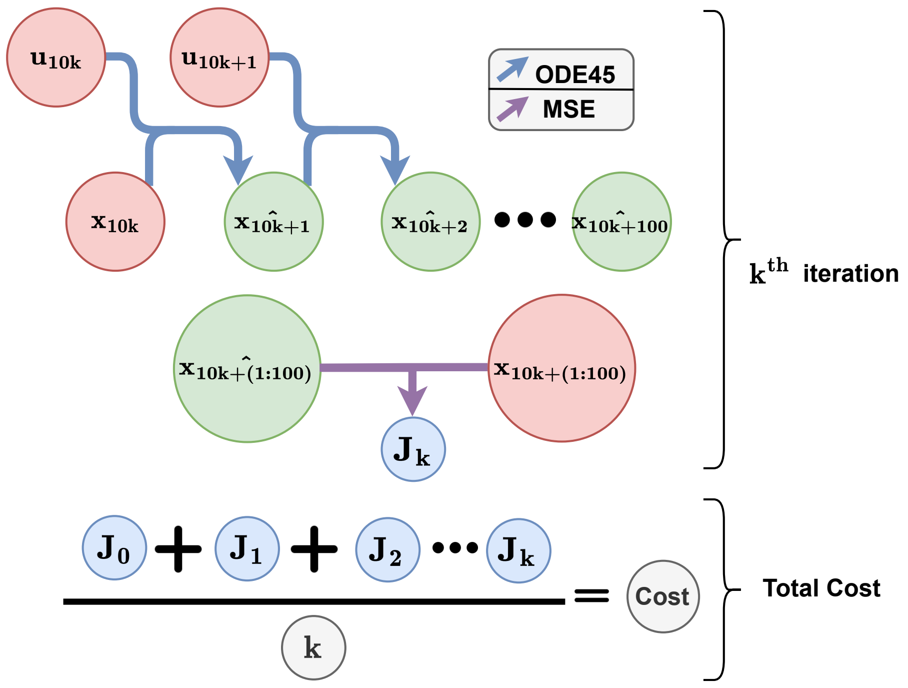


Figure 3.6: General form of the cost calculation and mean operation.

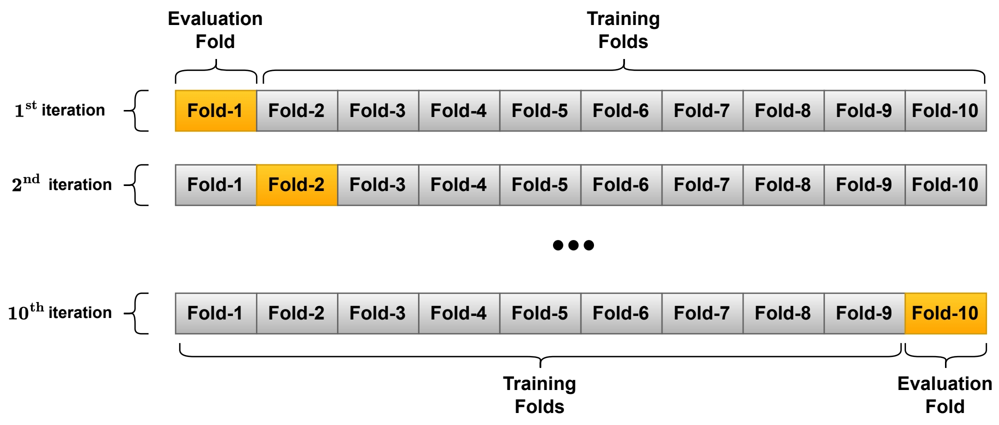
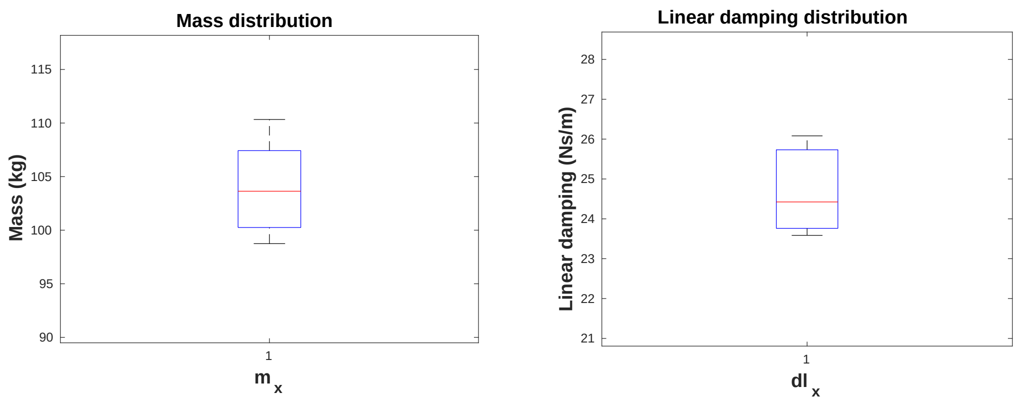


Figure 3.7: Training and the evaluation process for the k-fold cross validation

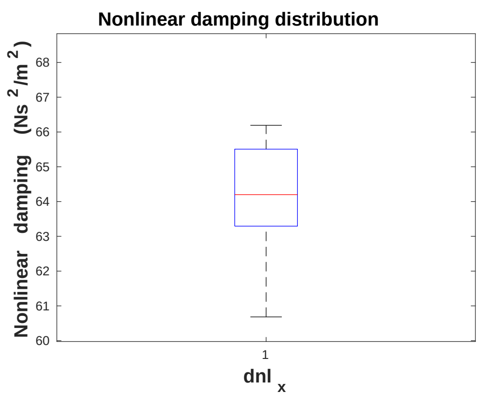


(a) (b)

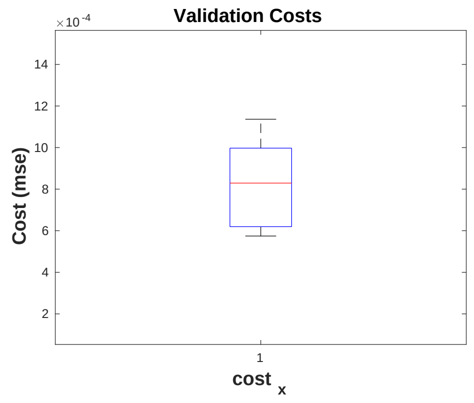
Figure 3.8: a : x component of the mass (m_x) parameter distribution by 10-fold cross-validation b : the surge axis linear damping (dl_x) parameter distribution by 10-fold cross-validation

3.4 k-fold cross-validation

Our optimization method makes gradient-based calculations to update the parameters. As it is not a stochastic global search algorithm such as Particle Swarm Optimization (PSO) [46] and Genetic algorithm, the optimal solution can stick in the local minima. We applied a k-fold cross-validation algorithm to solve this problem for our optimization procedure. K-fold cross-validation splits the identification experiments into k groups randomly. Each group's cost function is trained with the other k-1 groups,

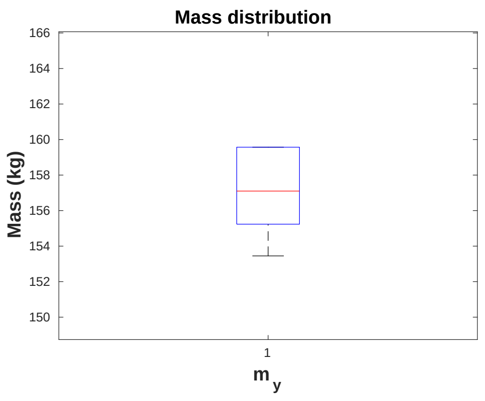


(a)

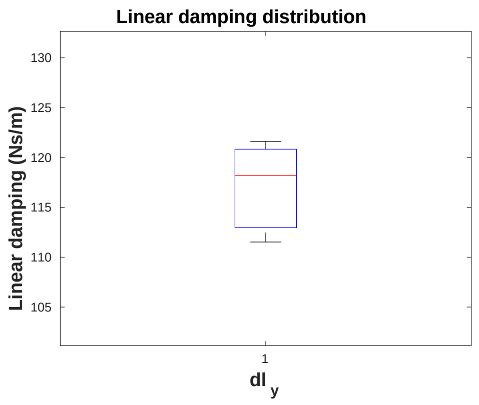


(b)

Figure 3.9: a : x component of the nonlinear damping (dnl_x) parameter distribution by 10-fold cross-validation b : 10-fold cross-validation costs distribution for the surge axis in MSE.



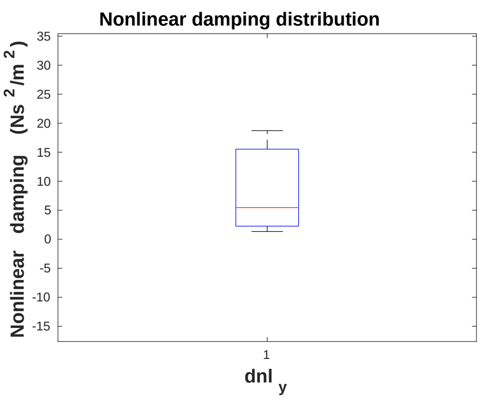
(a)



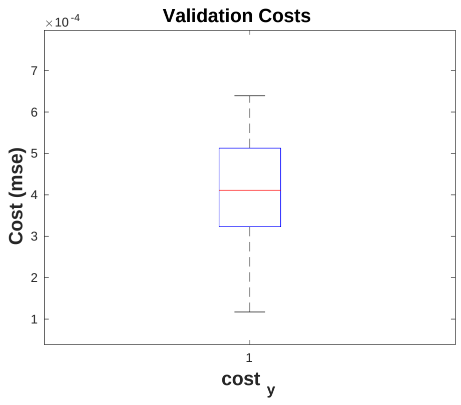
(b)

Figure 3.10: a : y component of the mass (m_y) parameter distribution by 10-fold cross-validation b : the sway axis linear damping (dl_y) parameter distribution by 10-fold cross-validation

and optimal parameters are evaluated with the corresponding group. This system makes different dataset groups for the evaluation and training phases (fig. 3.7). The cross-validation method provides a parameter distribution as the training is done for k times. This distribution can help us understand the optimization parameter deviations in randomly selected experiment groups. This study selects constant input,



(a)



(b)

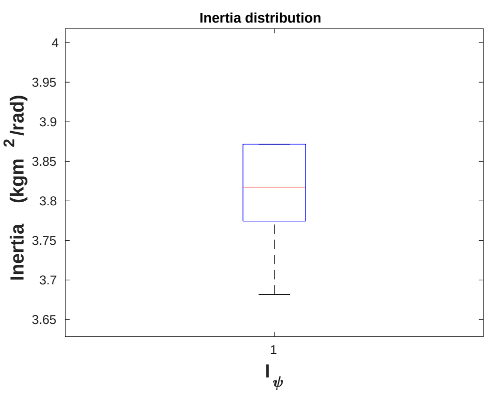
Figure 3.11: a : y component of the nonlinear damping (dnl_y) parameter distribution by 10-fold cross-validation b : 10-fold cross-validation costs distribution for the sway axis in MSE.

sinusoidal input, and closed-loop experiments for each robot axis. These experiments are divided into ten groups randomly.

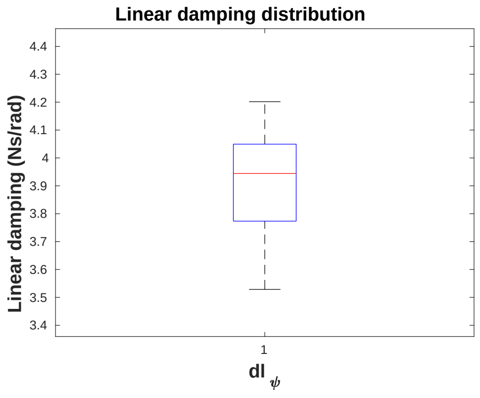
Table 3.1: Parameter Estimation results. The median value for each parameter is considered as the best parameter. Cost indicates the evaluation performances of the median values. The Max iterations value is the maximum iteration limit of the optimizer for each fold. The mass, linear damping, and nonlinear damping coefficients contain added water effects.

Parameter Estimation			
Parameter	**Surge**	**Sway**	**Yaw**
Mass	103.6370 kg	157.1003 kg	3.8175 kgm^2/rad
Linear Damping	24.4238 Ns/m	118.2121 Ns/m	3.9443 Ns/rad
Nonlinear Damp.	64.1989 Ns^2/m^2	5.4556 Ns/m	1.8720 Ns^2/rad^2
Max Iterations	400	400	400
Cost (MSE)	8.2911 10^{-4}	4.1113 10^{-4}	1.5428 10^{-3}

According to the figures 3.8, 3.9, 3.10 and 3.11, the surge and the sway axis parameter distributions are obtained, and the evaluation metrics are shown. The robot's rigid body mass is measured as 12.4 kg. The surge and the sway mass component estima-

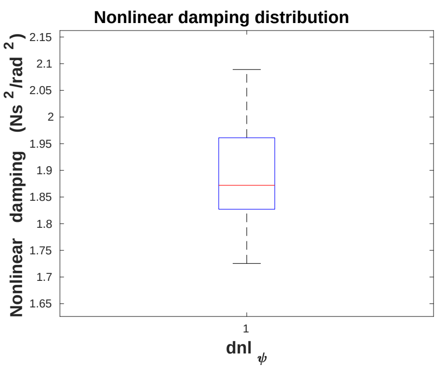


(a)

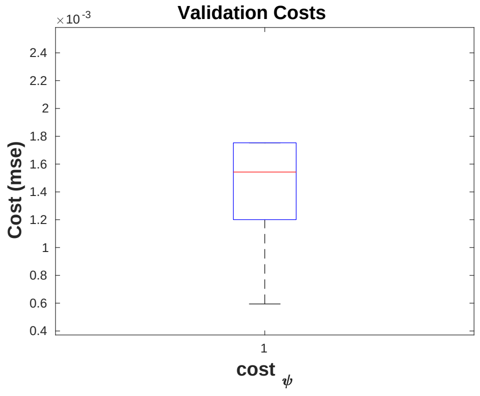


(b)

Figure 3.12: a : yaw component of the inertia (I_{ψ}) parameter distribution by 10-fold cross-validation b : the heading axis linear damping (dl_{ψ}) parameter distribution by 10-fold cross-validation



(a)



(b)

Figure 3.13: a : yaw component of the nonlinear damping (dnl_{ψ}) parameter distribution by 10-fold cross-validation b : 10-fold cross-validation costs distribution for the heading axis in MSE.

tions are 103.64 kg and 157.10 kg (Table 3.1), respectively. Water has an added mass effect, so finding more extensive mass components for both axes is reasonable. Ver, we must make a computational fluid dynamics analysis to validate the added mass effect of the water. As the catamaran-type floatings aggravate the sway motion in the water, the added mass component of the sway axis is more significant than the surge axis' added mass coefficient.

According to the figures 3.12 and 3.13, the heading axis parameter distributions and the evaluation metric deviations can be analyzed. As we know the volume of the robot, we can calculate the inertia term around the heading axis as 0.61 kgm^2 by assuming that the volume is uniform. The heading axis inertia estimation of the robot is 3.8175 kgm^2. The difference between the approximate rigid body inertia and the inertia estimation is accepted as added inertia effect of the water. We do not have a chance to validate the parameter estimation with more accurate analytical parameter calculations. As the parameter deviations of the heading axis have a reasonable magnitude, we can use these parameters to test the model-based TV-LQR controller.

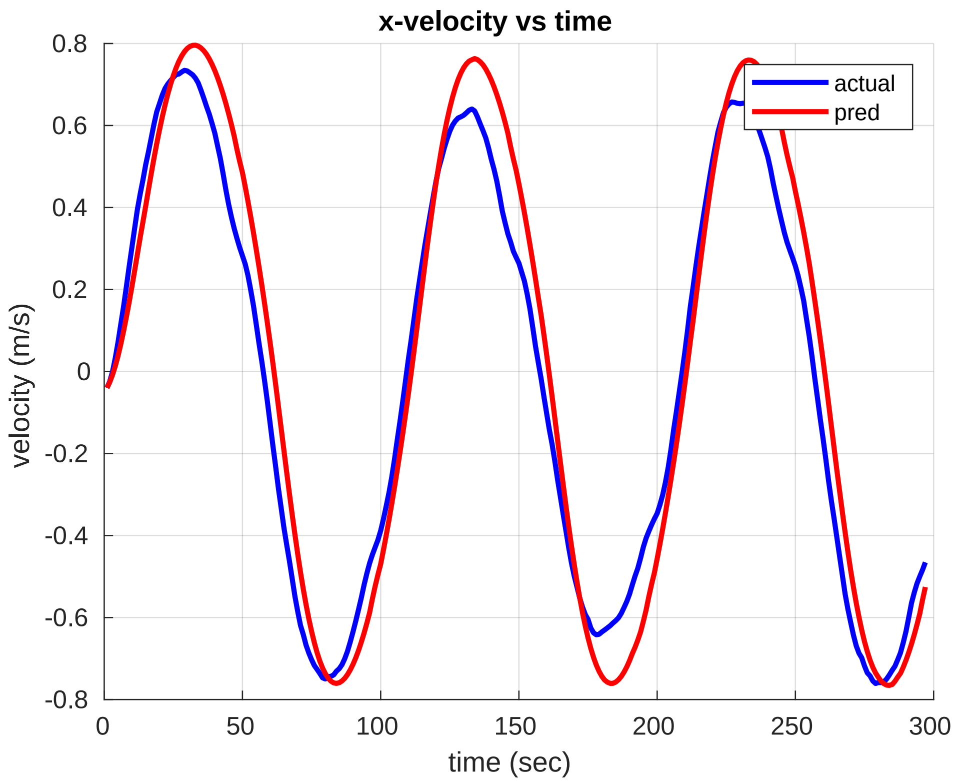


Figure 3.14: Prediction of an example sinusoidal input experiment and actual velocity measurements for the surge axis.

In figure 3.14, the dynamic model is simulated with the selected parameters. The red line indicates the numerical solution of the dynamical system with $250sin(2\pi.0.1t)$ input PWM signal for the surge axis. The blue line indicates the real robot's response to the same input signal.

CHAPTER 4

CONTROL OF THE ROBOT

4.1 PI Controller

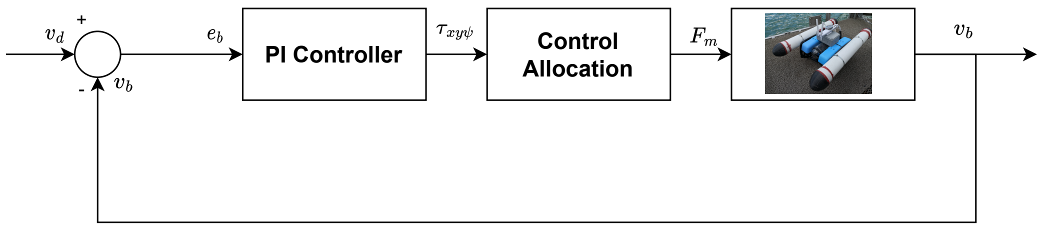


Figure 4.1: The USV PI controller block diagram

PI control is a closed-loop control technique with negative feedback. The PI controller takes the plant's output as a controlled variable and feeds it back to the controller by subtracting it from a reference signal (Figure 4.1). The reference signal and controlled variable assign the controller type. In this case, the velocity vector of the USV is used as a controlled signal and velocity control is aimed. The input of the PI, an error signal between the robot's body velocities and the reference velocities, is given in Equation 4.1. $\boldsymbol{v_b}$ and $\boldsymbol{v_d}$ indicate the body and desired velocity, respectively.

$$\boldsymbol{v_b} = \begin{bmatrix} u_b \\ v_b \\ r_b \end{bmatrix} \; \boldsymbol{v_d} = \begin{bmatrix} u_d \\ v_d \\ r_d \end{bmatrix} \\ \boldsymbol{e_b} = \boldsymbol{v_d} - \boldsymbol{v_b} \quad \text{st.} \quad \boldsymbol{e_b} \in \mathbb{R}^3 \tag{4.1}$$

The PI controller manipulates the error signal in two decoupled forms (Equation 4.2. It scales the current error signal with a constant coefficient K_P to obtain a proportional term, and the PI controller calculates the accumulated error in the experiment by taking the integral of the $\boldsymbol{e_b(t)}$. This integral term is also scaled with a constant coefficient K_I. Accumulating the error signal decreases the steady-state error. Since the USV is Multi Input Multi Output (MIMO) system, three decoupled PI controllers are designed for each axis. As the inputs of the closed-loop systems are desired velocities in 3-DoF, Manipulated variables of the robot become forces and torque values in 3-DoF. Mapping these force/torque values into motor commands is the task of the control allocation block. The control allocation block converts the three-dimensional vector into a four-dimensional motor command using Equation 2.11 and motor input constraints. The conventional PI controller needs a parameter search of the PI coefficients for better control performances. In this study, the nonlinear MIMO robot's PI tuning is done with many experimental trials, and the best parameter results are analyzed.

$$\tau_{PI} = K_p\boldsymbol{e_b(t)} + K_I \int_0^t \boldsymbol{e_b(\hat{t})} d\hat{t} \tag{4.2}$$

4.2 TV-LQR

LQR, Linear Quadratic Regulator, is a model-based optimal control method widely used in industry and scientific research. As with other optimal control problems, LQR tries to minimize a cost function of a dynamic system. It is suitable for linear systems, and the cost function is quadratic (Equation 4.4). Increasing the Q matrix maximizes tracking performances of the reference state trajectories. Decreasing the R matrix penalizes the USV's input signals less. Hence the power consumed by the motors increased. One must choose the most suitable Q and R matrices for the requirements. The solution of the cost function is a gain matrix, K, that takes the full-state feedback from the robot to control.

$$\dot{x} = A(t)\boldsymbol{x} + B(t)\boldsymbol{u} \tag{4.3}$$

The USV's linear time-varying dynamic model is obtained. Assuming that the system is completely controllable for every time t [47], one can use the time-varying model

(Equation 4.3) in LQR as TV-LQR. State feedback gain matrix becomes a time series signal in time-varying cases. Since our system is fully actuated and the rank of the transformation matrix between motor inputs and the robot's body axes is equal to the DoF number, we can assume that the USV is completely controllable.

The finite horizon continuous-time cost of the LQR is given in Equation 4.4. Finding the $\boldsymbol{u}^*(t)$ for $t \in [t_0, T]$ is the goal of this cost function [48].

$$J = \int_{t_0}^{T} [\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{u}^T R \boldsymbol{u}] dt + \boldsymbol{x}^T(T) Q_f \boldsymbol{x}(T) \tag{4.4}$$

The optimal control input and the $K(t)$ gain matrix of the TV-LQR is given in Equation 4.5. The negative sign of the control signal indicates the negative feedback of the closed loop system. The $\boldsymbol{x}(t)$ term of the input signal shows that this feedback is full-state feedback of the USV. The row number of the gain matrix is equal to the input size of the robot, and the column number indicates the state vector dimension.

$$\begin{aligned} \boldsymbol{u}^*(t) &= -R^{-1} B(t)^T P(t) \boldsymbol{x}(t) \\ K(t) &= R^{-1} B(t)^T P(t) \end{aligned} \tag{4.5}$$

$$\boldsymbol{u}(t) = -K(t)(\boldsymbol{x}(t) - \boldsymbol{x_d}(t)) + \boldsymbol{u_d}(t) \tag{4.6}$$

The control input in Equation 4.5 can be directly applied when the system has zero reference signal. Most of the time, desired trajectories are not zero. One can remember that the change of variable operation is done in the Section 2.7. This change of variable operation should be reversed as given in Equation 4.6. The $\boldsymbol{u_d}$ is desired input signal, and the $\boldsymbol{x_d}$ is desired trajectory vector for all states of the robot.

4.3 Control Allocation

Control allocation aims to map the USV's desired torque values into motor commands. Manipulated variables of both PID and TV-LQR are a vector with elements containing expected thrust values in the surge, sway, and yaw axis. The allocator distributes these thrust values to the motors. Although the system is decoupled in 3 axes, the robot can simultaneously have a desired nonzero trajectory in more than one axes. In this case, the allocator must distribute the signal in 3 axes to the motors, considering the motor constraints.

$$T^{+} = T^{T}(TT^{T})^{-1} \tag{4.7}$$

$$\boldsymbol{u}(t) = K^{-1}T^{+}\boldsymbol{\tau} \tag{4.8}$$

As is known, one of the simple and frequently used control allocation methods is the Moore-Penrose pseudo-inverse method. As mentioned, a T matrix (Equation 2.11) distributes motor commands over 3 degrees of freedom. T matrix is a transformation matrix between motor thrusts and the forces and torques for each body axis. Since T is not a square matrix, its inverse is not defined. However, the pseudo-inverse matrix of the T matrix can be calculated (Equation 4.7). This matrix is used in calculating motor inputs as in Equation 4.8. The K matrix of the equation is the matrix that scales the motor command to the torque value. There is no unique solution to distribute the thrust in 3 axes to four motors. Without constraints, the pseudo-inverse matrix can fit three torque parameters into four motor variables as a linear least squares technique. Motor inputs are clipped after the control allocation is finished. Since the constraints are not considered during the allocation, clipping the motor inputs later would result in failures when multiple nonzero reference signals are tested simultaneously.

Another method for control allocation is solving the allocation problem as an optimization problem by quadratic programming. Quadratic programming optimizes a quadratic cost function by considering the constraints.

In this study, CVXOPT [49] python library is used to implement the quadratic programming in the control allocation problem. A standard quadratic optimization problem is given in Equation 4.9. The control allocation problem should be transformed into the standard quadratic programming format. A new state variable ξ is defined to find the best motor inputs that minimize the error between the desired and predicted torque vector (Equation 4.10). An additional ξ parameter helps us modify the control allocation problem as a linear program. Quadratic cost terms minimize the squared of the motor inputs to find the least squared optimal solutions. Increasing these quadratic terms results in less power consumption but slow response time.

$$\begin{aligned} minimize \quad & \frac{1}{2}\boldsymbol{x}^T P\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} \\ s.t \quad & G\boldsymbol{x}\leq\boldsymbol{h} \\ & A\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{4.9}$$

$$\begin{aligned} minimize \quad & \begin{bmatrix}0 & 0 & 0 & 0 & -1\end{bmatrix}^T \boldsymbol{x} = -\xi \\ s.t \quad \boldsymbol{x} = & \begin{bmatrix}u_1 & u_2 & u_3 & u_4 & \xi\end{bmatrix}^T \\ -400\leq u_i\leq 400 & \quad \text{for} \quad i \in [1,2,3,4] \\ & 0\leq\xi\leq 1 \\ \begin{bmatrix} -0.7070 & -0.7070 & 0.7070 & 0.7070 & -\tau_x \\ -0.7070 & +0.7070 & -0.7070 & 0.7070 & -\tau_y \\ -0.1888 & 0.1888 & 0.1888 & -0.1888 & -\tau_\psi \end{bmatrix} \boldsymbol{x} = & \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \end{bmatrix} \end{aligned} \tag{4.10}$$

CHAPTER 5

RESULTS

5.1 Performance Metrics

5.1.1 Sinusoidal Velocity Tracking Performances

- **Tracking Error**: Tracking error measures the average error between the desired and the actual velocity signals. There are different calculation methods for tracking errors. Mean Absolute Percentage Error (MAPE) is chosen for this study. MAPE value gives the absolute error ratio between the error and desired parameter for each sample. Equation 5.1 shows the MAPE calculation. N is the number of samples. v_{ref} and v_{act} are desired and actual velocity signals for the robot. One of the disadvantageous of this method is MAPE value diverges when the reference signal sample is too small. Hence, it is unsuitable for tracking error analysis around the zero value.

$$\frac{1}{N}\sum_{i=1}^{N}\left|\frac{v_{ref}-v_{act}}{v_{ref}}\right| \times 100\% \tag{5.1}$$

- **Phase Lag**: The difference in phase between the reference and the system response is considered phase lag. It is calculated using the Fast Fourier Transform (FFT) algorithm. The FFT algorithm provides the phase of the signal in the frequency domain. As we know the frequency of the reference and the actual velocity signals, phase lag can be calculated by subtracting the reference velocity phase angle of the FFT outputs around the corresponding frequency value from the actual velocity signal's phase angle.

- **Control effort**: Integral of the squared motor thrust signals provides us a metric for energy usage of the robot during the experiments. Motor PWM gain can be transformed into thrust by multiplying with a constant scaling factor. Using the thrust (N) parameter, control effort can have a physical unit (N^2s), and some physical inferences can be made.

- **Maximum Tracking Error**: Since percentage overshoot is well-defined for the system's step response, one can consider the maximum tracking error of the robot as a measurement for the deviation of the tracking performances. Since the initial velocity of the desired velocity and the robot's initial velocity can be different, tracking error from the initial point to the settling point is not considered for the maximum tracking error.

5.1.2 Step Response Performances

- **Tracking Error**: MAPE is unsuitable for the step functions as the desired velocity might be zero for some time intervals. Normalized mean square error (NMSE) is suitable for these reference signals. The main advantage of the NMSE from mean square error is its normalization term. In this way, we can compare the performance of experiments with different amplitudes using NMSE. In equation 5.2, the NMSE formula is given for the desired and the actual velocities.

$$\text{NMSE} = \frac{1}{N} \frac{\sum_{i=1}^{N} (v_{ref} - v_{act})^2}{\sum_{i=1}^{N} v_{ref}^2} \tag{5.2}$$

- **Percentage Overshoot**: This metric is practical for transient time performance measurement. It is a ratio between the tracking error during the peak velocity of the robot and the steady-state value.

- **Control Effort**: Control effort is calculated with the same method for the sinusoidal velocity tracking experiments. The energy consumption of PI and TV-LQR controller can be deduced from this metric.

- **Settling Time**: Settling time is the time interval that the transient response of the robot is completed. Settling time is assumed to be when the tracking error remains in the 5% steady-state value band.

5.2 Sinusoidal Tracking Responses

In this section, The USV's simulation and experimental results are explained for the sinusoidal velocities. Since the USV uses Ardupilot Framework, one can simulate the dynamics of the USV with the ArduSub module of the ArduPilot's Software in the Loop (SITL) toolbox [50]. Sinusoidal experiments cover velocity tracking performances of PI and TV-LQR controllers when sinusoidal reference velocities (Eqn. 5.5) are applied to the surge and the heading axes at the same time. The sway axis velocity is tried to be fixed at the zero velocity. Sinusoidal signals also have an arbitrary constant velocity term. This term is given to make sure that robot moves in the positive direction of the heading and the surge axes. The constant velocity term provides the robot to make an elliptic motion.

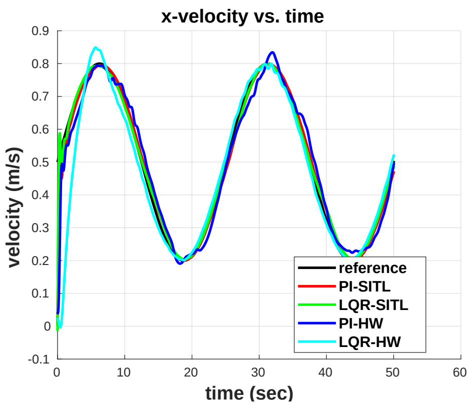
$$K_P = \begin{bmatrix} 1000 \\ 1000 \\ 10 \end{bmatrix}, \quad K_I = \begin{bmatrix} 2000 \\ 2000 \\ 200 \end{bmatrix}, \tag{5.3}$$

Assume that Q and R matrices are chosen as below,

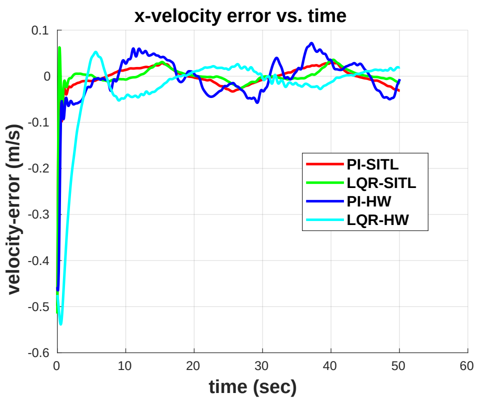
$$Q = \begin{bmatrix} 10000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}, \quad R = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \tag{5.4}$$

TV-LQR controller is computed with constant Q and R matrices as described in chapter 2. Each experiment is performed in multiple replicates, and the parameters that give the most optimal result are used. Using any optimization algorithm in parameter selection is excluded from this study's content. The robot's nonlinear effects required different parameters for each experiment. Hence, these parameters are determined by repeating each experiment a sufficient number of times. Equation 5.4 shows the example Q and R matrices configuration, which is used during the sinusoidal velocity tracking experiments of the SITL. As the system is spherical in the SITL environment, the same parameters are selected for the surge and sway axes. Since the yaw axis has a faster response than the x and y axes, small values are chosen for the yaw axis. A similar procedure is applied for the PI controller gains. Equation 5.3 gives the proportional (K_P) and the Integral (K_I) term coefficients of the PI controller for the SITL.

$$\begin{aligned} u &= 0.3sin(2\pi 0.04t) + 0.5 \\ v &= 0 \\ r &= 0.2sin(2\pi 0.04t) + 0.6 \end{aligned} \tag{5.5}$$



(a)



(b)

Figure 5.1: a : x velocity vs. time for sinusoidal reference velocities in the equation 5.5. "-SITL" suffix indicates the SITL results. PI and TV-LQR are real experiment results. b : x velocity error vs time for sinusoidal reference velocities in the equation 5.5. Difference between the robot's surge velocity and the desired surge velocity for each sampling point.

In figure 5.1 and 5.2, the robot's surge and heading velocity responses are observed with respect to time steps.The USV's motor inputs during this experiment are shown in figure 5.3. Time-varying gain of the TV-LQR. "-SITL" suffix indicates the SITL results. PI and TV-LQR are actual experiment results. Each figure is obtained from

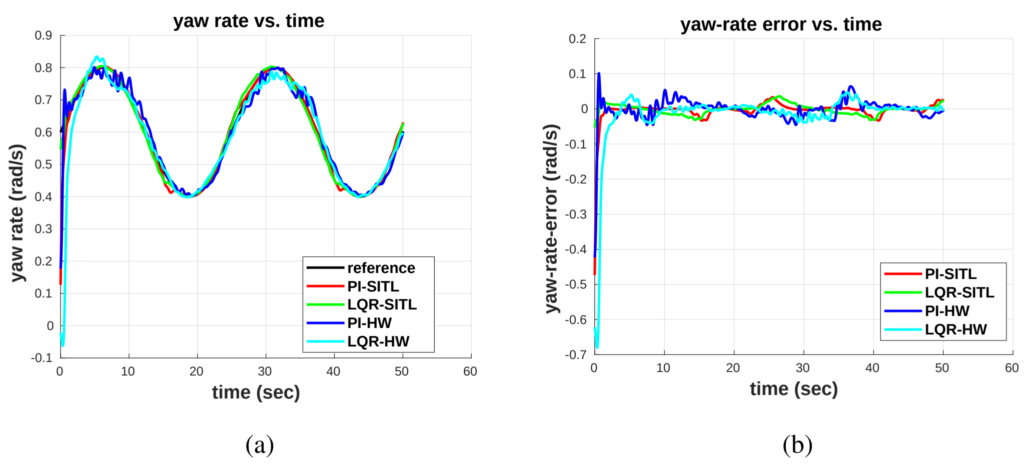


Figure 5.2: a : yaw rate response of the system with 0.04 Hz sinusoidal reference signal (Eqn. 5.5). b : yaw rate error vs time for PI and TV-LQR controllers. Difference between the robot's heading velocity and the desired heading velocity for each sampling point.

the same experiment logs. Table 5.1 and 5.1 indicate the performance metrics of the sinusoidal tracking experiment. "-S" suffix indicates the SITL results. Controller types without a suffix are the actual robot experiments.

Table 5.1: Performance metrics for the surge axis during the sinusoidal velocity tracking (Eqn. 5.5). The control effort is the total control effort applied for three axes' velocity tracking missions simultaneously. Tracking errors are not considered for the maximum calculation until the settling time of the experiments.

Surge Velocity Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (MAPE)**	**Phase Lag** (rad)	**Control effort** ($N^2 s$)	**Max. Tracking Error**
TV-LQR-S	2.6799 %	0.0420	2.4382×10^4	0.0356 m/s
PI-S	3.7340 %	0.0896	2.5651×10^4	0.0979 m/s
TV-LQR	7.2272 %	0.0564	3.1886×10^4	0.0527 m/s
PI	7.7090 %	0.1732	4.4365×10^4	0.0720 m/s

According to table 5.1, tracking error is smoothly decreased by TV-LQR in both SITL and actual robot. Since the integral term of the PI controller causes a phase lag in the

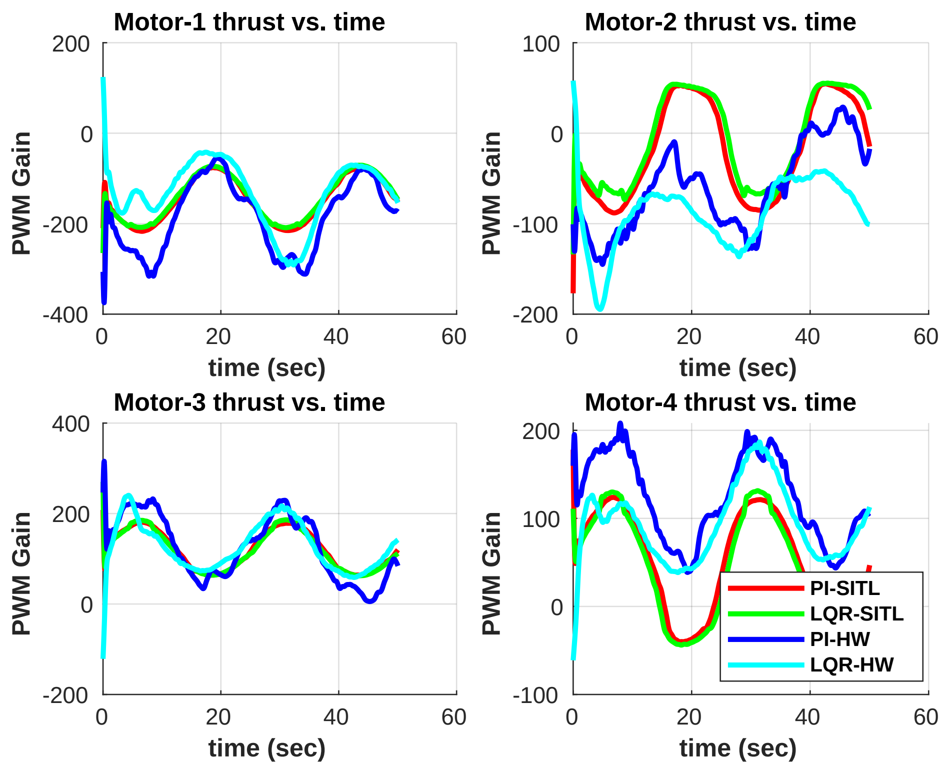


Figure 5.3: Sinusoidal velocity tracking in the x and yaw-axis (Eqn. 5.5), motor input signals for PI and TV-LQR.

Table 5.2: Performance metrics for the heading axis during the sinusoidal velocity tracking (Eqn. 5.5). Phase lag approximately converges to the zero for PI-S. Control effort is the total control effort to track three reference velocity signals in Equation 5.5

Yaw Rate Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (MAPE)**	**Phase Lag** (rad)	**Control effort** (N^2s)	**Max. Tracking Error**
TV-LQR-S	$1.7347\ \%$	-0.0769	2.4382×10^4	$0.0359\ rad/s$
PI-S	$1.7961\ \%$	≈ 0.00	2.5651×10^4	$0.0335\ rad/s$
TV-LQR	$3.9007\ \%$	0.1641	3.1886×10^4	$0.0471\ rad/s$
PI	$2.9584\ \%$	0.1026	4.4365×10^4	$0.0644\ rad/s$

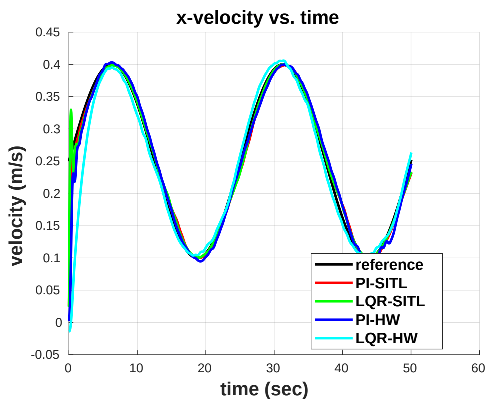
system actual robot's phase lag for the surge axis is calculated as $0.1732\ rad$, which is three times bigger than the phase lag of the TV-LQR. Since the real robot has a higher mass coefficient in the surge axis, experiments need more control efforts for the same desired velocity signals. Maximum tracking error helps find the most deviated point of the experiments after the transient response of the robot. For all experiments, the maximum tracking error is less than $0.1\ m/s$. TV-LQR outperforms the PI controller in terms of the maximum tracking error.

According to table 5.2, TV-LQR performs less tracking error than PI in the simulator experiments. However, the yaw velocity tracking of the TV-LQR controller has more tracking errors in actual experiments. While we know the exact model parameters of the SITL USV, one can see that system identification is applied for the actual robot's parameter estimation. This result shows that obtaining more accurate model parameters improves the TV-LQR performances. When the phase lag is considered TV-LQR-S experiment's phase lag is negative. Since the TV-LQR is an optimal control policy, velocity response might lead to the reference signal. The PI controller might increase the robot's output phase as it has only one pole at the origin.

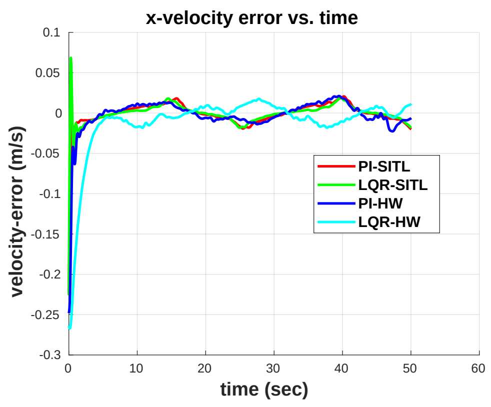
$$\begin{aligned} u &= 0.15sin(2\pi 0.04t) + 0.25 \\ v &= 0 \\ r &= 0.1sin(2\pi 0.04t) + 0.3 \end{aligned} \tag{5.6}$$

Figure 5.4 and 5.5 show low-velocity sinusoidal tracking experiments. The peak values for the surge and heading velocities are reduced by half as given in the equation 5.6. By performing this type of experiment, small velocity elliptic maneuvers of the surface vehicles are represented. As our USV has a dead zone region for the motor inputs between the PWM gain interval, $[-25, 25]$, minor disturbances might happen around this region (fig. 5.5). Figure 5.5 indicates that this problem is seen in the SITL experiments as motors 2 and 4 go into the dead zone at the same time. Since these two motors are located on the same side of the robot, they affect the symmetrical structure of the heading force components of the motors.

In table 5.3, all controller types performed less control effort than the high-speed sinusoidal tracking case (Eqn. 5.5). As the MAPE value is a normalized value with

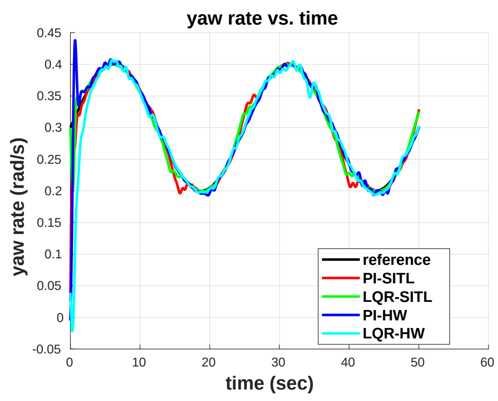


(a)

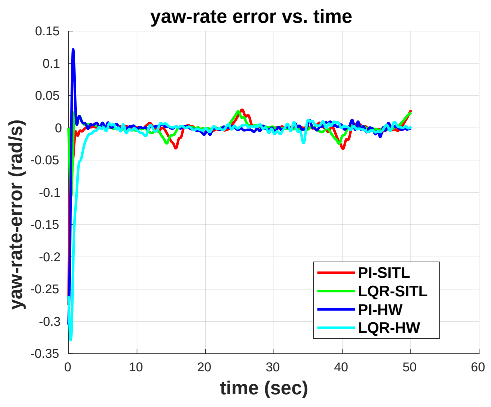


(b)

Figure 5.4: a : x velocity vs. time for sinusoidal reference velocities in the equation 5.6. LQR-SITL indicates TV-LQR controller experiment in the SITL. b : x velocity error vs time for sinusoidal reference velocities in the equation 5.6. Difference between the robot's surge velocity and the desired surge velocity for each sampling point.



(a)



(b)

Figure 5.5: a : yaw rate response of the system with 0.04 Hz sinusoidal reference signal (Eqn. 5.6). b : yaw rate error vs time for PI and TV-LQR controllers. Difference between the robot's heading velocity and the desired heading velocity for each sampling point.

respect to the reference signal, one can compare the MAPE results of the same control types between high and low-velocity experiments. As the peak velocity of the

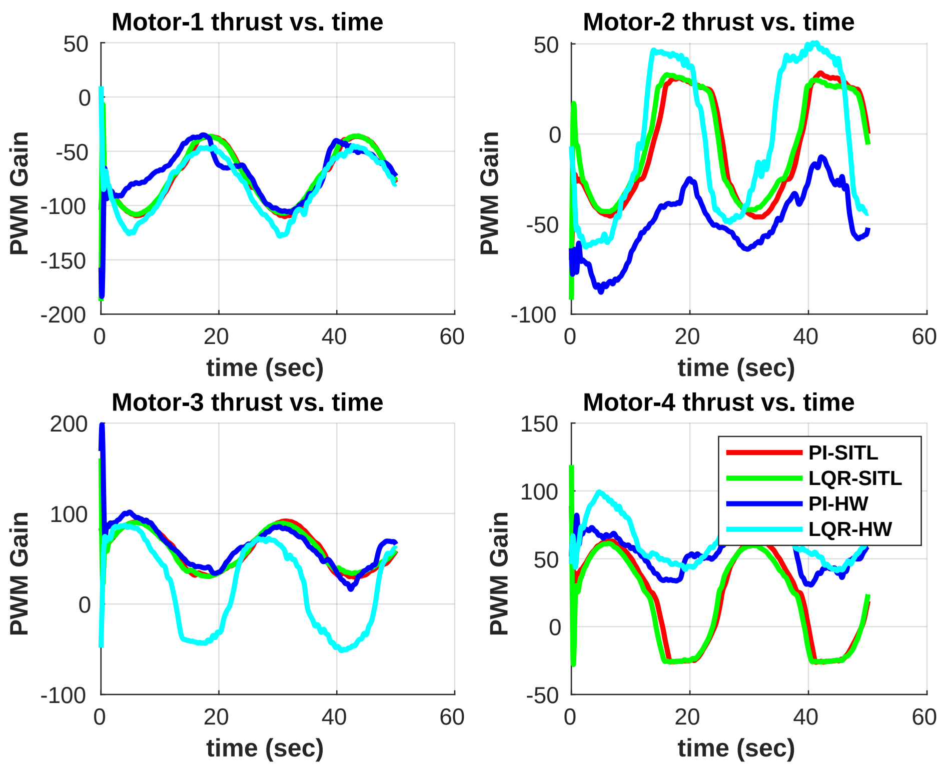


Figure 5.6: Sinusoidal velocity tracking in the x and yaw-axis (Eqn. 5.6), motor input signals for PI and TV-LQR.

Table 5.3: Performance metrics for the surge axis during the sinusoidal velocity tracking (Eqn. 5.6). The control effort is the total control effort applied for three axes' velocity tracking missions simultaneously. Tracking errors are not considered for the maximum calculation until the settling time of the experiments.

Surge Velocity Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (MAPE)**	**Phase Lag** (rad)	**Control effort** (N^2s)	**Max. Tracking Error**
TV-LQR-S	$3.2823\ \%$	0.0756	6.3283×10^3	$0.0193\ m/s$
PI-S	$3.9340\ \%$	0.0948	6.6432×10^3	$0.0206\ m/s$
TV-LQR	$6.3483\ \%$	0.0560	8.4124×10^3	$0.0183\ m/s$
PI	$5.1140\ \%$	0.1197	8.5634×10^3	$0.0227\ m/s$

Table 5.4: Performance metrics for the heading axis during the sinusoidal velocity tracking (Eqn. 5.6). Phase lag approximately converges to the zero for PI-S.

Yaw Rate Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (MAPE)**	**Phase Lag** (rad)	**Control effort** (N^2s)	**Max. Tracking Error**
TV-LQR-S	$2.0180\,\%$	-0.0555	6.3283×10^3	$0.0255\,rad/s$
PI-S	$2.3063\,\%$	≈ 0.00	6.6432×10^3	$0.0324\,rad/s$
TV-LQR	$3.4247\,\%$	0.1432	8.4124×10^3	$0.0235\,rad/s$
PI	$1.9669\,\%$	0.0241	8.5634×10^3	$0.0643\,rad/s$

experiment increases the robot's Coriolis and centripetal effects, the tracking error parameter increases. These effects are coupled in 3 DoF; however, PI and TV-LQR apply decoupled control signals for 3 DoF. PI controller has 0.1197 rad and 0.1737 rad phase lag in the low and high-velocity actual experiments, respectively. The mass over linear damping ratio gives approximate information about the phase lag of the open loop system. As the surge axis' mass over linear-damping ratio is more significant than the heading axis' ratio, the PI controller's phase lag can be noticed in the surge axis' performance metrics.

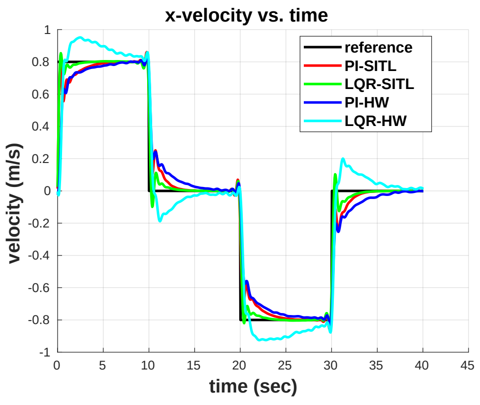
5.3 Step Responses

These experiments generate piecewise constant velocity signals to make the robot follow a rectangular shape path. The surge and sway axes are used, and the yaw is held at zero velocity. The robot's motion interval is separated into four pieces. First, zero and $0.8m/s$ velocities are applied on the sway and surge axes, respectively. Second, surge velocity becomes zero while the sway axis is $0.4m/s$. Then the negative direction of these two steps is applied to complete the rectangular shape. Equation 5.7 shows the piecewise constant velocity signals in more detail. These types of velocity responses are rare in surface vehicles. Our motivation for these experiments is to see the step responses of the robot in the surge and sway axes and perform some maneuvers that classical ship-type underactuated USVs cannot perform.

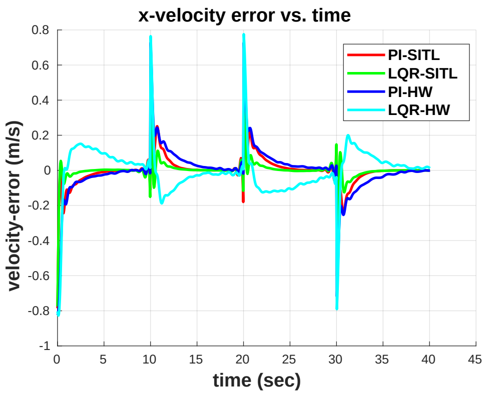
$$u = \begin{cases} 0.8, & 0 \leq t < 10 \\ 0, & 10 \leq t < 20 \\ -0.8, & 20 \leq t < 30 \\ 0, & 30 \leq t < 40 \end{cases}$$

$$v = \begin{cases} 0, & 0 \leq t < 10 \\ 0.4, & 10 \leq t < 20 \\ 0, & 30 \leq t < 40 \\ -0.4, & t \geq 30 \end{cases} \tag{5.7}$$

$$r = 0$$



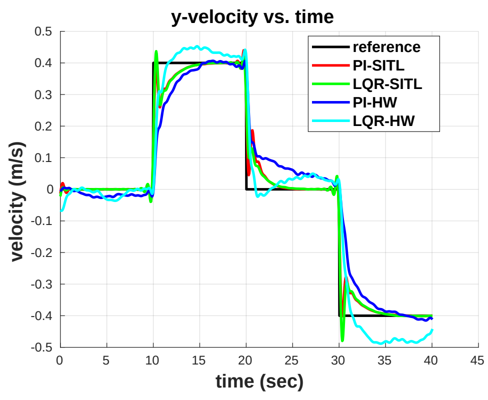
(a)



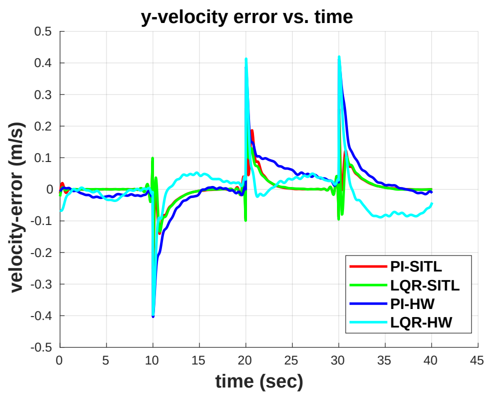
(b)

Figure 5.7: a : x velocity vs. time for the piecewise constant velocity equation 5.7. LQR-SITL indicates TV-LQR controller experiment in the SITL. b : x velocity error vs time for the piecewise constant velocities in the equation 5.7. Difference between the robot's surge velocity and the desired surge velocity for each sampling point.

Table 5.5 compares PI and TV-LQR performances. As the weight of the TV-LQR is significant, the system response is faster than the PI controller in SITL. The underdamped response for the TV-LQR and overdamped response for the PI controller can be seen in both SITL and actual experiments (Fig. 5.7 and 5.8).



(a)



(b)

Figure 5.8: a : y velocity vs. time for the piecewise constant velocity equation 5.7. LQR-SITL indicates TV-LQR controller experiment in the SITL. b : y velocity error vs time for the piecewise constant velocities in the equation 5.7. Difference between the robot's surge velocity and the desired surge velocity for each sampling point.

Table 5.5: Performance metrics for the surge axis during the step responses (Eqn. 5.7). The control effort is the total control effort applied for three axes' velocity tracking missions simultaneously. Percentage overshoot and settling time are calculated for the first part of the piecewise velocity function. Tracking error is obtained from each experiment's entire time interval.

Surge Velocity Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (NMSE)**	**Settling Time** (sec)	**Control effort** (N^2s)	**Percentage Overshoot**
TV-LQR-S	0.0142	0.9512	2.5955×10^4	7.0640 %
PI-S	0.0261	3.0538	2.4658×10^4	–
TV-LQR	0.0661	7.4092	6.6225×10^4	20.2050 %
PI	0.0576	3.4543	4.2144×10^4	–

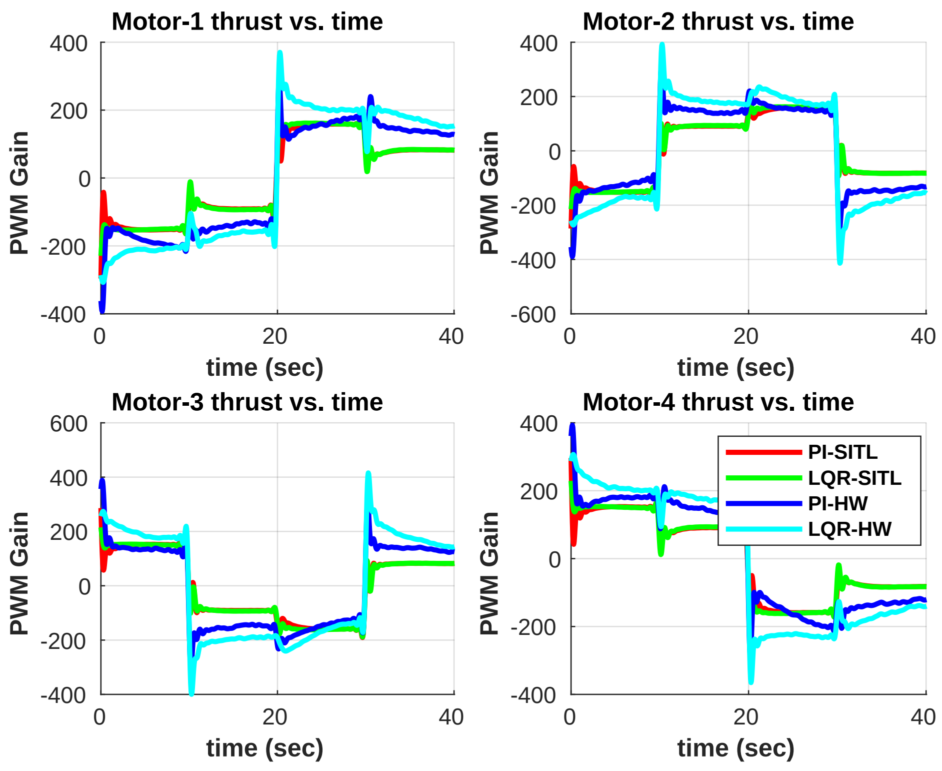


Figure 5.9: Step responses in the x and y axes (Eqn. 5.7), motor input signals for PI and TV-LQR.

Table 5.6: Performance metrics for the sway axis during the piecewise constant velocity tracking (Eqn. 5.7).

Sway Velocity Tracking Performances of PI and TV-LQR				
Controller Type	**Tracking Error (NMSE)**	**Settling Time** (sec)	**Control effort** (N^2s)	**Percentage Overshoot**
TV-LQR-S	0.0142	3.3667	2.5955×10^4	6.7462 %
PI-S	0.0017	3.3166	2.4658×10^4	–
TV-LQR	0.0053	8.4731	6.6225×10^4	12.9492 %
PI	0.0070	4.8185	4.2144×10^4	–

CHAPTER 6

CONCLUSIONS

In this study, Mathematical modeling and velocity control of a new design USV is aimed. By comparing the model-based TV-LQR and conventional PI controller, trajectory tracking performances of the fully-actuated robot are explained. A comprehensive literature review has been carried out. Autonomous vehicle technologies in the marine industry and their regulations are presented. Degrees of freedom and the Froude number of the robot are calculated to learn the vehicle group of the robot. The physical properties of the robot are explained. For the robot to move on every axis, four thrusters are placed on the robot's skeleton symmetrically. A custom robot frame has been made in Ardupilot autopilot software to directly control the robot's motors. The control allocation method is applied as quadratic programming to distribute the reference thrust values in 3 degrees of freedom as inputs to the four motors.

According to the Newton-Euler formula, a dynamic model of the robot is obtained, and some assumptions are made to make the system decoupled. Mass and damping matrices are assumed to be diagonal. Coriolis and centripetal effects are ignored. The thruster model of the robot is linear and has a small dead zone around the zero input. Six-dimensional state space model and its linearization around a reference signal derived.

System identification data is collected as constant, sinusoidal, and closed-loop experiments. A scaling factor between the torque value and the T200 Thrusters is obtained using the data generated by MathWorks [51]. The cost function for the parameter estimation of the USV dynamics is explained. The cost function takes every ten points of the experimental data as the initial condition and solves the differential equation numerically with predicted coefficients in a constant prediction horizon. We utilized a gradient-based optimization method to update our system parameters. However, this method risks getting stuck in local minima. To mitigate this, we implemented a k-fold cross-validation algorithm. This strategy provided distinct dataset groups for training and evaluation, and by repeating the process ’k’ times, we obtained a distribution of optimization parameters.

The MIMO system’s conventional PI controller and model-based TV-LQR controller are explained. TV-LQR is derived by showing that the optimal cost value has a quadratic form. By using the Hamilton-Jacobi theory matrix form of the Ricatti Differential equation is obtained. Solving the Ricatti equation with a final value boundary condition completes the solution for the optimal control signal. Control allocation methods are explained in this study. Higher Q matrices force the robot to follow the state trajectories well.

In the experimental results, we presented the sinusoidal tracking and step response performances of the USV under both PI and TV-LQR control strategies. The results indicate that the TV-LQR controller outperforms the PI controller regarding tracking error and phase lag in sinusoidal tracking experiments. The better performance of TV-LQR can be attributed to its dynamic model-based optimal control policy. The results also emphasize the importance of obtaining accurate model parameters, as the performance of the TV-LQR controller is affected by the quality of the system identification.

Furthermore, the phase lag of the PI controller was observed to be higher in the surge axis, which the mass can explain over linear damping ratio. Phase lag of the PI controller highlights the limitations of the PI controller in handling the coupled dynamics of the USV. In low-velocity sinusoidal tracking experiments, all controllers’ control effort was lower than in high-speed experiments. Less tracking error in low-velocity response is consistent with the expectation that lower velocities lead to reduced Coriolis and centripetal effects.

The USV was subjected to piecewise constant velocity signals in the step response experiments to follow a rectangular path. Although such velocity responses are not usual for surface vehicles, these experiments allowed us to examine the step response performance of the USV in the surge and sway axes and demonstrate the ability of the USV to perform maneuvers that classical ship-type underactuated USVs cannot achieve. Overall, the results presented in this study provide valuable insights into the performance of the USV under different control strategies and highlight the potential advantages of using TV-LQR controllers for improved tracking performance in various operating conditions.

As a future work, the mathematical model of this system can be used in different model-based control algorithms. TV-LQR controller can be implemented with feedback motion planning algorithms such as Tedrake's LQR-Trees method [52], and Ege's random sequential composition method [53].

REFERENCES

[1] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[2] S. International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE*, 2018.

[3] M. C. 1638, "Outcome of the regulatory scoping exercise for the use of maritime autonomous surface ships (mass)," 2021.

[4] A. Zelenak, C. Peterson, J. Thompson, and M. Pryor, "The advantages of velocity control for reactive robot motion," in *Dynamic Systems and Control Conference*, vol. 57267, p. V003T43A003, American Society of Mechanical Engineers, 2015.

[5] S. Grechi and A. Caiti, "Comparison between optimal control allocation with mixed quadratic & linear programming techniques," *IFAC-PapersOnLine*, vol. 49, no. 23, pp. 147–152, 2016.

[6] X. Bai, B. Li, X. Xu, and Y. Xiao, "A review of current research and advances in unmanned surface vehicles," *Journal of Marine Science and Application*, vol. 21, no. 2, pp. 47–58, 2022.

[7] W. Naeem, T. Xu, R. Sutton, and A. Tiano, "The design of a navigation, guidance, and control system for an unmanned surface vehicle for environmental monitoring," *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, vol. 222, no. 2, pp. 67–79, 2008.

[8] N. Gu, Z. Peng, D. Wang, Y. Shi, and T. Wang, "Antidisturbance coordinated path following control of robotic autonomous surface vehicles: Theory and experiment," *IEEE/ASME transactions on mechatronics*, vol. 24, no. 5, pp. 2386–2396, 2019.

[9] C. R. Sonnenburg and C. A. Woolsey, “Modeling, identification, and control of an unmanned surface vehicle,” *Journal of Field Robotics*, vol. 30, no. 3, pp. 371–398, 2013.

[10] M. G. Feemster and J. M. Esposito, “Comprehensive framework for tracking control and thrust allocation for a highly overactuated autonomous surface vessel,” *Journal of Field Robotics*, vol. 28, no. 1, pp. 80–100, 2011.

[11] Đ. Nađ, N. Mišković, and F. Mandić, “Navigation, guidance and control of an overactuated marine surface vehicle,” *Annual Reviews in Control*, vol. 40, pp. 172–181, 2015.

[12] B. Robotoics, “Bluerov2 datasheet,” 2021.

[13] J. Curcio, J. Leonard, and A. Patrikalakis, “Scout-a low cost autonomous surface platform for research in cooperative autonomy,” in *Proceedings of OCEANS 2005 MTS/IEEE*, pp. 725–729, IEEE, 2005.

[14] H. Lim, J. Park, D. Lee, and H. J. Kim, “Build your own quadrotor: Open-source projects on unmanned aerial vehicles,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 33–45, 2012.

[15] J. Ge, T. Li, and T. Geng, “The wireless communications for unmanned surface vehicle: An overview,” in *International Conference on Intelligent Robotics and Applications*, pp. 113–119, Springer, 2018.

[16] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, “Pixhawk: A system for autonomous flight using onboard computer vision,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 2992–2997, IEEE, 2011.

[17] S. M. Uddin, M. R. Hossain, M. S. Rabbi, M. A. Hasan, and M. S. R. Zishan, “Unmanned aerial vehicle for cleaning the high rise buildings,” in *2019 International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp. 657–661, IEEE, 2019.

[18] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, “Micro air vehicle link (mavlink) in a nutshell: A survey,” *IEEE Access*, vol. 7, pp. 87658–87680, 2019.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, p. 5, Kobe, Japan, 2009.

[20] T. SNAME, "Nomenclature for treating the motion of a submerged body through a fluid," *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin*, no. 1950, pp. 1–5, 1950.

[21] G. Cai, B. M. Chen, and T. H. Lee, "Coordinate systems and transformations," in *Unmanned rotorcraft systems*, pp. 23–34, Springer, 2011.

[22] T. I. Fossen, "Marine control systems–guidance. navigation, and control of ships, rigs and underwater vehicles," *Marine Cybernetics, Trondheim, Norway, Org. Number NO 985 195 005 MVA, www. marinecybernetics. com, ISBN: 82 92356 00 2*, 2002.

[23] J. Shin, D. J. Kwak, and Y.-i. Lee, "Adaptive path-following control for an unmanned surface vessel using an identified dynamic model," *IEEE/ASME transactions on mechatronics*, vol. 22, no. 3, pp. 1143–1153, 2017.

[24] J. Woo, J. Park, C. Yu, and N. Kim, "Dynamic model identification of unmanned surface vehicles using deep learning network," *Applied Ocean Research*, vol. 78, pp. 123–133, 2018.

[25] L. Ljung, "System identification," in *Signal analysis and prediction*, pp. 163–173, Springer, 1998.

[26] G. Rajesh and S. K. Bhattacharyya, "System identification for nonlinear maneuvering of large tankers using artificial neural network," *Applied Ocean Research*, vol. 30, no. 4, pp. 256–263, 2008.

[27] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: An overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[28] T.-J. Ren, T.-C. Chen, and C.-J. Chen, "Motion control for a two-wheeled vehicle using a self-tuning pid controller," *Control engineering practice*, vol. 16, no. 3, pp. 365–375, 2008.

[29] P. Meshram and R. G. Kanojiya, “Tuning of pid controller using ziegler-nichols method for speed control of dc motor,” in *IEEE-international conference on advances in engineering, science and management (ICAESM-2012)*, pp. 117–122, IEEE, 2012.

[30] F. G. Martins, “Tuning pid controllers using the itae criterion,” *International Journal of Engineering Education*, vol. 21, no. 5, p. 867, 2005.

[31] Z.-Y. Zhao, M. Tomizuka, and S. Isaka, “Fuzzy gain scheduling of pid controllers,” *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 5, pp. 1392–1398, 1993.

[32] C. B. Jabeur and H. Seddik, “Optimized neural networks-pid controller with wind rejection strategy for a quad-rotor,” *Journal of Robotics and Control (JRC)*, vol. 3, no. 1, pp. 62–72, 2022.

[33] R. P. Borase, D. Maghade, S. Sondkar, and S. Pawar, “A review of pid control, tuning methods and applications,” *International Journal of Dynamics and Control*, vol. 9, no. 2, pp. 818–827, 2021.

[34] D. Ma, S. Hao, W. Ma, H. Zheng, and X. Xu, “An optimal control-based path planning method for unmanned surface vehicles in complex environments,” *Ocean Engineering*, vol. 245, p. 110532, 2022.

[35] J. W. Roberts, R. Cory, and R. Tedrake, “On the controllability of fixed-wing perching,” in *2009 American Control Conference*, pp. 2018–2023, IEEE, 2009.

[36] X. Wu, S. Liu, T. Zhang, L. Yang, Y. Li, and T. Wang, “Motion control for biped robot via ddpg-based deep reinforcement learning,” in *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 40–45, IEEE, 2018.

[37] C. Shen, Y. Shi, and B. Buckham, “Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5796–5805, 2017.

[38] E. M. Einarsson and A. Lipenitis, “Model predictive control for the bluerov2 theory and implementation.”

[39] M. von Benzon, F. F. Sørensen, E. Uth, J. Jouffroy, J. Liniger, and S. Pedersen, “An open-source benchmark simulator: Control of a bluerov2 underwater robot,” *Journal of Marine Science and Engineering*, vol. 10, no. 12, p. 1898, 2022.

[40] C.-J. Wu, *6-dof modelling and control of a remotely operated vehicle*. PhD thesis, Flinders University, College of Science and Engineering., 2018.

[41] G. Reina, A. Vargas, K. Nagatani, and K. Yoshida, “Adaptive kalman filtering for gps-based mobile robot localization,” in *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pp. 1–6, IEEE, 2007.

[42] O. Özdemir, “Feedback motion planning of a novel fully actuated unmanned surface vehicle via sequential composition of random elliptical funnels,” Master’s thesis, Middle East Technical University, 2022.

[43] A. Packard, K. Poolla, and R. Horowitz, “Dynamic systems and feedback,” *Berkley, CA: Department of Mechanical Engineering, University of California*, 2002.

[44] S. S. Sandøy, “System identification and state estimation for rov udrone,” Master’s thesis, NTNU, 2016.

[45] MATLAB, *9.12.0.2009381 (R2022a)*. Natick, Massachusetts: The MathWorks Inc., 2022.

[46] R. Poli, J. Kennedy, and T. Blackwell, “Particle swarm optimization,” *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[47] B. D. Anderson and J. B. Moore, *Optimal control: linear quadratic methods*. Courier Corporation, 2007.

[48] D. E. Kirk, *Optimal control theory: an introduction*. Courier Corporation, 2004.

[49] L. Vandenberghe, “The cvxopt linear and quadratic cone program solvers,” *Online: http://cvxopt. org/documentation/coneprog. pdf*, 2010.

[50] ArduPilot, “Ardupilot.”

[51] K. Fedorenko and C. D'Souza, "System identification of blue robotics thrusters video."

[52] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[53] E. Ege and M. M. Ankarali, "Feedback motion planning of unmanned surface vehicles via random sequential composition," *Transactions of the Institute of Measurement and Control*, vol. 41, no. 12, pp. 3321–3330, 2019.

APPENDIX A

TV-LQR DERIVATIONS

The finite horizon continuous-time cost of the LQR is given in Equation A.1. Finding the $\boldsymbol{u}^*(t)$ for $t \in [t_0, T]$ is the goal of this cost function [48].

$$J = \int_{t_0}^{T} [\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{u}^T R \boldsymbol{u}] dt + \boldsymbol{x}^T(T) Q_f \boldsymbol{x}(T) \tag{A.1}$$

For the calculation of the optimal control signal, one needs to follow the path given below [47],

- If the optimal value of the cost (Eqn. A.1), $J^*(\boldsymbol{x}(t), t)$ exists with an arbitrary $\boldsymbol{u}^*(t)$, show that it is in the form of $\boldsymbol{x}^T(t)P(t)\boldsymbol{x}(t)$ where $p(t)$ is a symmetric matrix
- By using the Hamilton-Jacobi theory, show that $p(t)$ satisfies a nonlinear matrix Ricatti Differential Equation.
- Find the optimal control signal $\boldsymbol{u}^*(t)$

Let us assume that $J^*(\boldsymbol{x}(t), t)$ is continuous in the state vector, $\boldsymbol{x}(t)$. If $J^*(\boldsymbol{x}(t), t)$ has a quadratic, $\boldsymbol{x}^T(t)P(t)\boldsymbol{x}(t)$, form, it is necessary and sufficient to show that optimal cost satisfies Equations A.2 and A.3. $\boldsymbol{x_1}(t)$ and $\boldsymbol{x_2}(t)$ indicates two different reference state vector signals to show $J^*(\boldsymbol{x_1}(t), t)$ is the optimal solution for the $\boldsymbol{x_1}(t)$ and $J^*(\boldsymbol{x_2}(t), t)$ is optimal cost for the $\boldsymbol{x_2}(t)$.

$$J^*(\lambda \boldsymbol{x}(t), t) = \lambda^2 J^*(\boldsymbol{x}(t), t) \tag{A.2}$$

$$J^*(\boldsymbol{x_1}(t),t) + J^*(\boldsymbol{x_2}(t),t) = \frac{1}{2}[J^*(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t),t) + J^*(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t),t)] \tag{A.3}$$

Let us show the optimal cost value with the optimal control signal $\boldsymbol{u}^*(t)$ parameter. Since the system is linear and the cost is quadratic, multiplying $\boldsymbol{x}(t)$ and $\boldsymbol{u}(t)$ by a constant λ coefficients results in $\lambda^2 j^*(\boldsymbol{x}(t),t)$ (Eqn. A.4). Optimality of the $J^*(\lambda\boldsymbol{x}(t),t)$ garanties the inequality in Equation A.4. Multiplying the $\boldsymbol{u}(t)$ with λ^{-1} and $\boldsymbol{x}(t)$ with λ results in Equation A.5 in a similar way. Equation A.4 and A.5 satisfies the scaling property in Equation A.2.

$$J^*(\lambda\boldsymbol{x}(t),t) \le J(\lambda\boldsymbol{x}(t),\lambda\boldsymbol{u}^*(t),t) = \lambda^2 J^*(\boldsymbol{x}(t),t) \tag{A.4}$$

$$\lambda^2 J^*(\boldsymbol{x}(t),t) \le \lambda^2 J(\boldsymbol{x}(t),\lambda^{-1}\boldsymbol{u}^*(.),t) = J^*(\lambda\boldsymbol{x}(t),t) \tag{A.5}$$

One can see that Equation A.6 holds by using Equation A.2 with $\lambda = 2$. In Equation A.7 left-hand side of the equation is optimal. Hence, the cost function on the right-hand side of the inequality uses an arbitrary non-optimal control signal. As the state space model is linear, Equation A.7 can be written as Equation A.8.

$$J^*(\boldsymbol{x_1}(t),t) + J^*(\boldsymbol{x_2}(t),t) = \frac{1}{4}[J^*(2\boldsymbol{x_1},t) + J^*(2\boldsymbol{x_2},t)] \tag{A.6}$$

$$\begin{aligned}\frac{1}{4}[J^*(2\boldsymbol{x_1},t) + J^*(2\boldsymbol{x_2},t)] \le \frac{1}{4}[&J(2\boldsymbol{x_1},\boldsymbol{u}^*_{\boldsymbol{x_1}+\boldsymbol{x_2}} + \boldsymbol{u}^*_{\boldsymbol{x_1}-\boldsymbol{x_2}},t)\\ &+J(2\boldsymbol{x_2},\boldsymbol{u}^*_{\boldsymbol{x_1}+\boldsymbol{x_2}} + \boldsymbol{u}^*_{\boldsymbol{x_1}-\boldsymbol{x_2}},t)]\end{aligned} \tag{A.7}$$

$$\begin{aligned}&\frac{1}{4}[J(2\boldsymbol{x_1},\boldsymbol{u}^*_{\boldsymbol{x_1}+\boldsymbol{x_2}} + \boldsymbol{u}^*_{\boldsymbol{x_1}-\boldsymbol{x_2}},t) + J(2\boldsymbol{x_2},\boldsymbol{u}^*_{\boldsymbol{x_1}+\boldsymbol{x_2}} + \boldsymbol{u}^*_{\boldsymbol{x_1}-\boldsymbol{x_2}},t)] =\\ &\frac{1}{2}[J(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t),\boldsymbol{u}^*_{\boldsymbol{x_1}+\boldsymbol{x_2}},t) + J(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t),\boldsymbol{u}^*_{\boldsymbol{x_1}-\boldsymbol{x_2}},t)] =\\ &\qquad\frac{1}{2}[J^*(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t),t) + J^*(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t),t)]\end{aligned} \tag{A.8}$$

The right-hand side of Equation A.8 can be manipulated with $\lambda = 0.5$ by using Equation A.2. Using the system's linearity property in Equation A.9, the second necessary and sufficient condition in Equation A.3 is obtained

$$\begin{aligned}
\frac{1}{2}[J^*(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t),t)+J^*(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t),t)] = \\
2\left[J^*\left(\frac{(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t))}{2},t\right)+J^*\left(\frac{(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t))}{2},t\right)\right] \\
\leq J^*\left(\frac{(\boldsymbol{x_1}(t)+\boldsymbol{x_2}(t))}{2},\boldsymbol{u}^*_{\boldsymbol{x_1}}+\boldsymbol{u}^*_{\boldsymbol{x_2}},t\right)+J^*\left(\frac{(\boldsymbol{x_1}(t)-\boldsymbol{x_2}(t))}{2},\boldsymbol{u}^*_{\boldsymbol{x_1}}-\boldsymbol{u}^*_{\boldsymbol{x_2}},t\right) \\
= J(\boldsymbol{x_1}(t),\boldsymbol{u}^*_{\boldsymbol{x_1}},t)+J(\boldsymbol{x_2}(t),\boldsymbol{u}^*_{\boldsymbol{x_2}},t) = J^*(\boldsymbol{x_1}(t),t)+J^*(\boldsymbol{x_2}(t),t)
\end{aligned} \tag{A.9}$$

As a result of these mathematical operations, optimal cost value can be written as given in Equation A.10. If there is an optimal solution for the cost function, it can be shown as a quadratic function of $p(t)$.

$$J^*(\boldsymbol{x}(t),t) = \boldsymbol{x}^T(t)P(t)\boldsymbol{x}(t) \tag{A.10}$$

Hamilton-Jacobi Equation is given in Equation A.11 [47]. In Equation A.12, the parameters of this equation are modified according to this study. The cost function has constant Q and R matrices to regulate states and inputs, respectively. Derivative of the optimal cost value with respect to the state vector is obtained by Equation A.10. The state equation is taken as a linear time-varying system.

$$\frac{\partial J^*(\boldsymbol{x}(t),t)}{\partial t} = -min_{\boldsymbol{u}(t)}\left\{l(\boldsymbol{x}(t),\boldsymbol{u}(t),t)+\left[\frac{\partial J^*(\boldsymbol{x}(t),t)}{\partial \boldsymbol{x}}\right]f(\boldsymbol{x}(t),\boldsymbol{u}(t),t)\right\} \tag{A.11}$$

$$\begin{aligned}
l(\boldsymbol{x}(t),\boldsymbol{u}(t),t) &= \boldsymbol{u}^T(t)R\boldsymbol{u}(t)+\boldsymbol{x}^T(t)Q\boldsymbol{x}(t) \\
\left[\frac{\partial J^*(\boldsymbol{x}(t),t)}{\partial \boldsymbol{x}}\right]^T &= 2\boldsymbol{x}^T(t)p(t) \\
f(\boldsymbol{x}(t),\boldsymbol{u}(t),t) &= A(t)\boldsymbol{x}(t)+B(t)\boldsymbol{u}(t)
\end{aligned} \tag{A.12}$$

The final version of the Hamilton-Jacobi Equation can be written as given in Equation A.13 by using the time derivative of Equation A.10.

$$\boldsymbol{x}^T \dot{P(t)} \boldsymbol{x} = -min_{\boldsymbol{u}(t)} \Big[\boldsymbol{u}^T R \boldsymbol{u} + \boldsymbol{x}^T Q \boldsymbol{x} + 2\boldsymbol{x}^T P(t) + 2\boldsymbol{x}^T P(t) A(t) \boldsymbol{x} + 2\boldsymbol{x}^T P(t) B(t) \boldsymbol{u} \Big] \tag{A.13}$$

One can write the inside of the minimization operation in Equation A.13 as a quadratic form. The minimization of this quadratic form is trivial. Since the R matrix is a positive definite matrix, $(\boldsymbol{u} + R^{-1}B(t)^T P(t)\boldsymbol{x})$ term of the equation should converge to zero. Equation A.15 indicates a control signal, $\bar{\boldsymbol{u}(t)}$, that minimizes the Hamilton-Jacobi Equation. One can see that this control signal is the optimal control signal that minimizes the cost function. The control signal can be generated for the robot by calculating the $P(t)$ matrix.

$$\begin{aligned} \boldsymbol{u}^T R \boldsymbol{u} + \boldsymbol{x}^T Q \boldsymbol{x} + 2\boldsymbol{x}^T P(t) + 2\boldsymbol{x}^T P(t) A(t) \boldsymbol{x} + 2\boldsymbol{x}^T P(t) B(t) \boldsymbol{u} \\ = (\boldsymbol{u} + R^{-1}B(t)^T P(t)\boldsymbol{x}) R (\boldsymbol{u} + R^{-1}B(t)^T P(t)\boldsymbol{x}) \\ + \boldsymbol{x}^T [Q - P(t)B(t)R^{-1}B^T P(t) + P(t)A(t) + A(t)^T P(t)] \boldsymbol{x} \end{aligned} \tag{A.14}$$

$$\bar{\boldsymbol{u}(t)} = -R^{-1}B(t)^T P(t) \boldsymbol{x}(t) \tag{A.15}$$

After the control input, which is given in Equation A.15 minimizes Equation A.14, Equation A.16 is obtained and holds for all $\boldsymbol{x}(t)$. By reducing the $\boldsymbol{x}(t)$ terms, Equation A.17 gives the matrix Ricatti equation. As the matrix form of the differential Ricatti equation is obtained, it is necessary to define boundary conditions to solve the differential equation. The solution of this differential equation is the $P(t)$ symmetric matrix, which completes the optimal control signal calculation. This differential equation depends on the robot's time-varying state space matrices, the Q matrix that determines the trajectory tracking effort of the robot, and the R matrix that optimizes the robot's power consumption.

$$\boldsymbol{x}^T \dot{P(t)} \boldsymbol{x} = -\boldsymbol{x}^T [Q - P(t)B(t)R^{-1}B^T P(t) + P(t)A(t) + A(t)^T P(t)] \boldsymbol{x} \tag{A.16}$$

$$-\dot{P(t)} = Q - P(t)B(t)R^{-1}B^TP(t) + P(t)A(t) + A(t)^TP(t) \tag{A.17}$$

Since Hamilton-Jacobi Equation has a boundary condition as given in Equation A.18, the final value of the differential equation is Q_f. One can solve the differential equation numerically by using the boundary condition. As the boundary condition is a final value, the first iteration of the numerical solution method should start from the final time of the time interval $[t_0, T]$. $P(t)$ solution completes the finite horizon time-varying LQR controller derivation for the USV.

$$\begin{aligned} \boldsymbol{x}(T)^T\dot{P(T)}\boldsymbol{x}(T) &= \boldsymbol{x}(T)^TQ_f\boldsymbol{x}(T) \\ P(T) &= Q_f \end{aligned} \tag{A.18}$$

Equation A.19 provides the optimal control input and the $K(t)$ gain matrix of the TV-LQR. Notably, the negative sign associated with the control signal underscores the negative feedback inherent in the closed-loop system. Additionally, the term $\boldsymbol{x}(t)$ within the input signal signifies that the USV employs full-state feedback. The gain matrix's dimensions align with the robot's parameters, where the number of rows corresponds to the input size, and the number of columns represents the state vector dimension.

$$\begin{aligned} \boldsymbol{u}^*(t) &= -R^{-1}B(t)^TP(t)\boldsymbol{x}(t) \\ K(t) &= R^{-1}B(t)^TP(t) \end{aligned} \tag{A.19}$$