Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Erik Rowe
Simen Hustad
Petter Øvereng Juliebø
Elias Olsen Almenningen

Implementation of a quaternion-based PD controller in ROS2 for a generic underwater vehicle with six degrees of freedom

Bachelor's thesis in Electrical Engineering
Supervisor: Christian Fredrik Sætre
May 2022

NTNU
Norwegian University of Science and Technology

Erik Rowe
Simen Hustad
Petter Øvereng Juliebø
Elias Olsen Almenningen

Implementation of a quaternion-based PD controller in ROS2 for a generic underwater vehicle with six degrees of freedom

Bachelor's thesis in Electrical Engineering
Supervisor: Christian Fredrik Sætre
May 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Kunnskap for en bedre verden

Oppgavetittel (norsk og engelsk):
Implementation of a quaternion-based PD controller in ROS2 for a generic underwater vehicle with six degrees of freedom

Implementasjon av en kvaternion-basert PD regulator i ROS2 for et generelt undervannsfartøy med seks frihetsgrader

Forfattere:	**Prosjektnummer:**	E2205
Erik Rowe	**Innleveringsdato:**	20.05.22
Simen Hustad	**Gradering:**	[x] åpen
Petter Øvereng Juliebø		[ ] lukket
Elias Olsen Almenningen		

Studium:	Elektroingeniør - BIELEKTRO
Studieretning:	Automatisering og robotikk
Veileder internt:	Christian Fredrik Sætre
Institutt:	Institutt for teknisk kybernetikk
Oppdragsgiver:	Institutt for teknisk kybernetikk
Kontaktperson:	Damiano Varagnolo

Sammendrag:
Denne bachelor rapporten omhandler implimentasjonen av en ulineær PD regulator for et generelt undervannsfartøy med 6 frihetsgrader i ROS2. Rapporten vil dekke metodene og løsningen brukt for å utvikle en fungerende regulator node i ROS2.

En forsksningsgruppe ved Intitutt for teknisk kybernetikk, ved NTNU, har hatt et ønske om å få utviklet et fleksibelt system i ROS2 for å tilrettelegge for sensorbruk og data innsamling, og på lengre sikt tilrettelegge for autonom multi agent kontroll. Vår oppgave har derfor vært å utvikle et kontrollsystem som kan gjennomføre slike oppgaver.

I den teoretiske delen forklares nyttig kunnskap om undervannsfartøy, samt ROS2 funksjonaliteter. Det er også en beskrivelse av de ulike undervannsfartøyene brukt for testing, samt spesielle hensyn tatt for hver av dem.

Det har blitt implimentert en dynamisk posisjons kontroller i et ROS2 miljø. Rapporten beskriver hvordan gruppen utvidet kontrolleren til å inkludere spesifikke funksjoner, samtidig som fleksibiliteten til det generelle oppsettet ble ivaretatt. Kontrolleren er utviklet med den hensikt at den skal kunne bli brukt på alle slags undervannsfartøy. Det vil si at den skal kunne fungere på både ROV'er og AUV'er, og både med og uten tilbakekoblet regulering.

For å finne regulator parametere for det virkelige systemet har det også blitt utviklet en simulator. Rapporten inneholder derfor en simulator analyse, med tilhørende matematiske prinsipper som legger grunnlaget for slik simulering.

For å sikre at kontrolleren er generell, har det blitt utført tester på forskjellige BlueROV2 Heavy fartøy, samt på Beluga Mk.2, et undervannsfartøy utviklet av student organisasjonen Vortex NTNU. Disse testene viser at kontrollsystemet fungerer utmerket i kontrollerte omgivelser, og spesielt når det gjelder styring med joystick. Flere resultater er fremhevet i rapporten, inkludert en sammenligning mellom virkelig og simulert system.

Stikkord norsk:	**Stikkord engelsk:**
ROS2, Kvaternioner, 6 frihetsgrader, UUV, ROV, AUV, simulering	ROS2, Quaternions, 6 degrees of freedom, UUV, ROV, AUV, simulation

Summary

This thesis describes the implementation of a non-linear PD controller for a generic underwater vehicle with six degrees of freedom in ROS2. The thesis covers the methods and solutions used to develop a functioning controller node in ROS2.

A research group under the Department of Engineering Cybernetics at NTNU has a desire to develop a flexible system in ROS2, capable of aiding in the collection of sensor data and ultimately, enable automatic multi agent control. Our task was to develop a control system that may perform such operations.

In the theoretical section, we explain useful knowledge regarding underwater vehicles as well as ROS2 functionality. The section also includes a description of the various vehicles used for testing and the specific considerations for each of them.

We have implemented a dynamic positional controller as a module in a ROS2 environment. The thesis explains how the group expanded the controller to include specific features, while keeping the flexibility of a generic setup. The controller is created with the idea that it can be used on any UUV system, hence the controller can be used by ROVs and AUVs alike, with or without feedback control.

The group also created a simulator in order to help with finding control parameters for the real system. The thesis therefore includes a simulation study explaining how mathematical principles were used to facilitate this.

To ensure the controller is generic, tests have been conducted on different BlueROV2 Heavys, as well as on the Beluga Mk.2 AUV, a vehicle developed by the student organisation Vortex NTNU. These tests have proved that the control system handles admirably in controlled environments, especially when it comes to functionality regarding joystick control. Some results are highlighted in the report, including comparisons with the simulated environment.

Abstract

In this bachelor project a group of four students at NTNU have developed a control system, implemented in ROS2, able to control position and attitude of underwater vehicles.

All group members study Electrical Engineering with Automation and Robotics at NTNU. Motivation behind the project itself originates from ITK at NTNU, and their vision of a multi agent underwater fleet. Based on this vision, a foundation needs to be made for conducting research with the underwater vehicles, whom require a control system to operate.

We would like to give special thanks to the following people for their contribution and council for the duration of the project:

- Damiano Varagnolo - Providing the task and challenging us along the way
- Christian Fredrik Sætre - Supervisor
- Vortex NTNU - Providing the Beluga Mk.2 for testing purposes

All produced code is available as open source on GitHub:
https://github.com/ErikRowe/Bachelor-E2205

Data from testing is available on OneDrive: Bachelor E2205

List of Figures

List of Tables

Contents

Terminology

- **ITK** - The Department of Engineering Cybernetics, at NTNU
- **CAS** - Control Augmentation System: Control system that operates on the users behalf
- **UUV** - Unmanned underwater vehicle
- **ROV** - Remotely Operated Vehicle. Commonly referred to as underwater vehicles.
- **AUV** - Autonomous underwater vehicle
- **ROS/ROS2** - Robot Operating System: Set of software libraries and tools used for building robot applications
- **6DOF** - Six degrees of freedom refers to the ability to completely define an objects position and orientation in three dimensional space.
- **BlueROV2** - BlueRobotics underwater ROV
- **BlueROV2 Heavy** - BlueROV2 with extra thrusters, giving six DoF
- **CB** - Centre of buoyancy
- **CG** - Centre of gravity
- **STATE** - Measurements of position, attitude, velocities, etc.

Chapter 1

Introduction

1.1 Background

Underwater vehicles (UUV) are today used in a variety of marine operations, such as industrial, military and research applications. Specific tasks may be inspections, seafloor mapping and surveillance, as well as military specific missions like mine hunting and sabotage. They perform tasks that earlier have been done by humans, as well as task humans can't do. They are in general highly maneuverable, as well as cost efficient, reliable and delivers high quality services.

Based on their level of autonomy, UUVs may be devided into two categories, either remotely operated vehicles (ROV) or autonomous underwater vehicles (AUV). AUVs level of autonomy is continuously increasing and have today abilities to conduct autonomous missions with no operator input and following preplanned routes, as well as re-assessing underway according to data collected by on-board sensors. (*Autonomous Underwater Vehicle, HUGIN* 2022). However, commerically available AUVs are limited to only individual work, and have no abilities to conduct cooperative tasks in a multitude.(Varagnolo 2020). Due to the lack of a complete solution regarding this issue, ITK is therefore conducting a research project with the *object to develop knowledge that enables fleets of AUVs to operate collectively, cooperatively, adaptively, and in a leaderless fashion, and through this enable more efficient reliable monitoring and utilization of sea resources and infrastructures* (ibid.) As a part of the project, ITK is using a modified BlueROV2- Heavy as a testbed for development, containing advanced sensor systems.

As ROVs, similar to AUVs, are highly maneuverable underwater vehicles with multiple thrusters, they can be used in versatile applications, and therefore often have to carry additional hardware and tools. The added mass can make the vehicle unbalanced and difficult to control by a human operator. In such cases, the ROVs must have a control

augmentation system (CAS). That is, a control system that takes the commands from the operator and maneuvers the vehicle by controlling its thrusters (Varagnolo 2022).

In the BlueROV2's case, it has a default CAS distributed by ArduSub which works well for general hobby purposes. However, there has been a desire by ITK to create a CAS in the ROS2 environment, because ROS2 is specifically made for robotic systems.

1.2 Problem

The specific BlueROV2 Heavy we were to develop a system for was equipped with a CAS and teleoperation functionality. However, the teleoperation was practically unusable and the CAS was not complete.

Our task was primarily to develop a teleoperation system, but to increase complexity, and to get a better learning experience we decided to also develop a CAS with dynamic positioning. Later on we decided to make the system applicable beyond just the BlueROV2 Heavy, as well as have all code produced open-source. Our system was to be implemented in a ROS2 network running on a RasberryPi.

As a base, the CAS should be able to control orientation and depth, so that it may be used to collect valuable sensor data for research purposes. To increase complexity, functionality and learning experience, the group also decided to include positional control in the xy-plane.

Before trying out the CAS on the physical system, there was a desire to produce a simulated environment where a controller could be tested and tuned.

Neither ROS nor C++ are topics included in the groups study program, so a part of the task would also be to familiarize and learn to use these environments.

1.3 Report Structure

The report has been divided into six chapters. The structure is chosen to facilitate for an easy and intuitive reading, and is based on the natural progress throughout the project.

- Chapter one, which you are currently reading, contains the background and issue for the project.
- Chapter two is about the fundamental theory we have considered as favourable knowledge to have before continuing reading the report. It contains the basics regarding quaternion mathematics, coordinate frames and underwater vehicles, as well as ROS2.

- Chapter three conveys the methods we have used to solve the problem introduced in chapter one. Choices and solutions will be motivated when introduced throughout the chapter. We present how the ROS2 controller node has been developed as well as a simulation study conducted to get a rough overview of system responses.
- Chapter four contains response tests conducted on three different underwater vehicles: Standard BlueROV2 Heavy, Modified BlueROV2 Heavy and Vortex Beluga Mk.2. Simulation responses are also included in the relevant graphs.
- Chapter five contains discussion regarding response tests, choices made along the way and the final product. An evaluation of our learning experience with comparisons to the preliminary project is also found here.
- Chapter six concludes the report and the work we have done, as well as provides suggestions on possible expansions.

Chapter 2

Theory

In this chapter we present theory that is considered relevant knowledge to have before continuing. It contains fundamental theory regarding quaternions, coordinate frames, underwater vehicles and the implemented controller. In addition it will give a short introduction to ROS2.

2.1 Fundemental theory

2.1.1 Quaternion mathematics

Quaternions, invented by mathematician Hamilton, is a generalization of two-dimensional complex numbers, into three dimensions. It is a type of algebraic non commutative math, with the purpose of describing three-dimensional problems in mechanics. By adding a fourth dimension one allows to retain the normal rules of algebra, making it have associative- and non commutative-properties, for multiplication.(*quaternion | mathematics | Britannica* 2022)

Quaternions are generally represented in the form:

$$\boldsymbol{q} = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T \tag{2.1}$$
$$\eta \in \mathbb{R}$$

Unit quaternions, which is quaternions on normalized form, is a convenient mathematical notation for representing spatial attitude/orientations and rotations of elements in three dimensional space (*Quaternions and spatial rotation* 2022). The unit quaternion is normalized in the same way a vector is, by dividing its length:

$$\boldsymbol{q_u} = \frac{\boldsymbol{q}}{|\boldsymbol{q}|} \tag{2.2}$$

The elements of unit quaternions are called Euler parameters and satisfy: (T. I. Fossen 2021)

$$\eta + \epsilon^T \epsilon = \eta^2 + {\epsilon_1}^2 + {\epsilon_2}^2 + {\epsilon_3}^2 = 1 \tag{2.3}$$

Euler angles are often used to represent orientation, as they are very easy to visualize and use compared to quaternions. However, the greatest problem when using Euler angles is the phenomenon called Gimbal lock. Gimbal lock is a form of singularity where you lose one degree of freedom in three dimensional space, and may cause unpredictable behaviour when used in calculations.

Using quaternions has several advantages over Euler angles, including calculation speed and stability. However, the greatest drawback of quaternion representation is its complexity and lack of intuitive visualization.

2.1.2 Coordinate frames

Coordinate frames refers to how different objects are oriented and positioned in relation to eachother. When controlling a moving vehicle, it is essential to have the control object in relation to some constant anchor. This anchor is what we call the *world frame*, or inertial reference frame, $\{\boldsymbol{i}\}$.

The body frame $\{\boldsymbol{b}\}$ remains constant when seen from a moving vehicles perspective, even though the body frame is moving in relation to the world frame. A common way to represent both the world frame and body frame of sea-going vehicles is the *NED* representation.

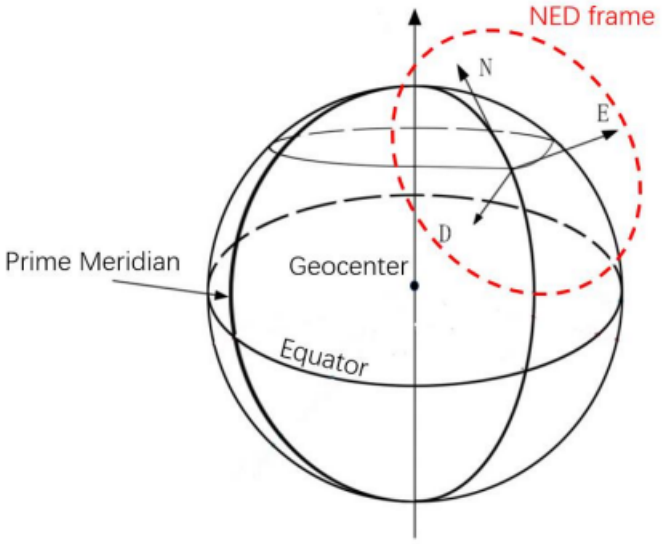


Figure 2.1: North-East-Down frame on the earth (Xu 2021, p. 14)

Figure 2.1 shows how the NED-frame placed on the earths surface, with the "Down" axis pointing towards the centre of the earth.

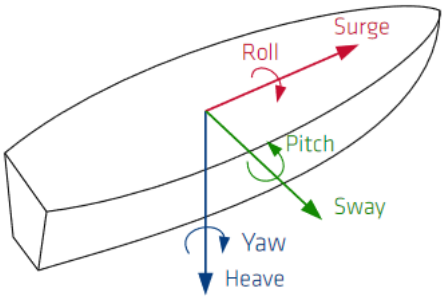


Figure 2.2: Ship motion convention (NED)(*Ship motion conventions* 2022)

The ship motion convention (*Ship motion conventions* 2022) describes standardized movement directions in relation to a vehicles body frame. When applying attitude and position setpoint of a vehicle, it is important to know what frame the attached sensors operate in to get predictable results.

2.1.3 Underwater vehicles

Underwater vehicles, also known as Unmanned Underwater Vehicles (UUVs), are a generic expression to describe both remotely operated vehicles (ROVs) and autonomous operated vehicles (AUVs). Whereas an ROV is operated by a human operator, through a physical connection between operator and vehicle, the AUV operates without operator intervention (US Department of Commerce 2022).

Different categories of ROVs

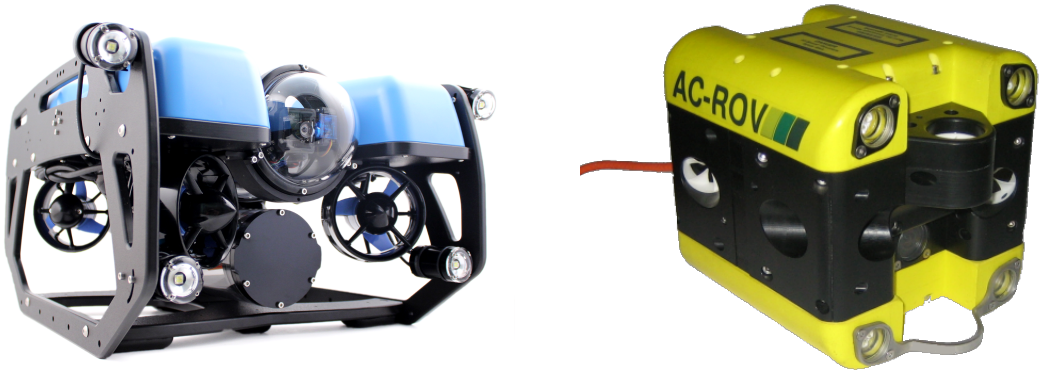
ROVs are divided into subcategories depending on their area of use: Inspection-Class ROVs and Intervention-Class ROVs. The Intervention-Class ROV, also known as work-class, can again be subdivided into standard- and heavy-duty work-class ROVs. A standard work-class ROV can weigh from 100 to 1500 kgs, and dive to 3000 meters. A heavy-duty work-class ROV are more robust, weighing up to 5000 kg and capable of diving to 5000 meters. Depending on their tasks, they can be equipped with hydraulic systems, advanced sensors, and accurate navigation instruments. Usual work can be cleaning, drilling, drilling support and construction. Because of their weight and complexity, the necessary equipment and crew required to deploy and operate them usually leads to high operation costs.

The Inspection-Class ROV's are much smaller, again being subdivided into medium sized and micro- or handheld sized. Medium sized ROVs can weigh from 30 up to 120 kg, and a micro from a couple of kg up to 30. Because of their light weight, they can often be deployed and recovered by hand, reducing cost. Their work usually contains underwater mapping and surveying, visual inspection, cleaning, latching or recovery of smaller items.

Geometrical properties of ROVs

The Inspection-Class ROV comes in many shapes and sizes but the most common configuration is the Open frame design. The open frame design aids the natural stability of the ROV, which is related to the distance between the CB and CG. By having the CB directly above the CG, the ROV will naturally right itself to its upright position. Another benefit of the open frame design is the added ease of access to mount auxiliary equipment, such as sensors and extra thrusters. In the "closed" frame ROV there is no clear difference between the frame, buoyancy and thruster system. This is clearly seen in figure 2.3b.

The most common propulsion method for Inspection-Class ROVs are thrusters. The level of control on the motion of an ROV depends on the number and configuration of these thrusters. Factors such as size, power, required DoF all influence configuration.



(a) Open frame ROV (BlueROV2) (*BlueROV2* 2022)

(b) "Closed" frame ROV (AC-ROV 100) (*AC-ROV 100* 2022)

Figure 2.3: Different design types for ROVs

ROVs have in general at least three thrusters, making control in surge, sway and heave possible. To enable control in 6DOF at least six thrusters are needed (Capocci et al. 2017).

Dynamics and mathematical framwork

When analyzing the dynamics of an underwater vehicle, it can be divided into kinematics and kinetics. Kinematics treats only geometrical aspects of motion, and kinetics the forces causing the motions. The dynamic equations of motion of an UUV from Fossens Robot-inspired Model for Marine Crafts (T. I. Fossen 2021) contains the kinematic (2.4) and kinetic (2.5) equations:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{J}(\boldsymbol{q})\boldsymbol{\nu} \iff \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{\boldsymbol{q}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\boldsymbol{q}) & \boldsymbol{0}_{3x3} \\ \boldsymbol{0}_{4x3} & \frac{1}{2}\boldsymbol{U}(\boldsymbol{q}) \end{bmatrix} \begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.4}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{2.5}$$

The kinematic equation (2.4) relates movement in the body-fixed $\{\boldsymbol{b}\}$ reference frame to movement in the inertial reference frame $\{\boldsymbol{i}\}$, through the transformation matrix $\boldsymbol{J}(\boldsymbol{q})$. $\boldsymbol{R}(\boldsymbol{q})$ is the rotation matrix from $\{\boldsymbol{i}\}$ to $\{\boldsymbol{b}\}$ (A.1), $\boldsymbol{U}(\boldsymbol{q})$ is the coordinate transformation matrix (A.2). $\boldsymbol{\nu}$ is a vector describing velocities in $\{\boldsymbol{b}\}$, where $\boldsymbol{v}$ is linear velocity and $\boldsymbol{\omega}$ is angular velocity (Fjellstad and Fossen 1994).

The kinetic equations (2.5) of a UUVs motion is derived from the Newton-Euler for-

mulation (T. I. Fossen 2021). It describes the forces and moments affecting the vehicle and causing its movements, referenced in $\{\boldsymbol{b}\}$. $\boldsymbol{M}$ is the system inertia matrix, $\boldsymbol{C}(\boldsymbol{\nu})$ is the Coriolis and centripetal matrix, $\boldsymbol{D}(\boldsymbol{\nu})$ is the hydrodynamic dampening matrix and $\boldsymbol{g}(\boldsymbol{q})$ is the vector of restoring forces and moments. $\boldsymbol{\tau}$ is the vector of control forces and moments.

These matrices are derived (and simplified) in (ibid.) and are on the form:

$$\boldsymbol{M} = \begin{bmatrix} m - X_{\dot{u}} & 0 & 0 & 0 & mz_g & 0 \\ 0 & m - Y_{\dot{v}} & 0 & -mz_g & 0 & 0 \\ 0 & 0 & m - Z_{\dot{w}} & 0 & 0 & 0 \\ 0 & -mz_g & 0 & I_x - K_{\dot{p}} & 0 & 0 \\ mz_g & 0 & 0 & 0 & I_y - M_{\dot{q}} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_z - N_{\dot{r}} \end{bmatrix} \tag{2.6}$$

$$\boldsymbol{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & 0 & mw+z_{\dot{w}}w & 0 \\ 0 & 0 & 0 & -mw-z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & mv-Y_{\dot{v}}v & -mu+X_{\dot{u}}u & 0 \\ 0 & mw-z_{\dot{w}}w & -mv+Y_{\dot{v}}v & 0 & I_zr-N_{\dot{r}}r & -I_yq+M_{\dot{q}}q \\ -mw+z_{\dot{w}}w & 0 & -mu+X_{\dot{u}}u & -Izr+N_{\dot{r}}r & 0 & I_xp-K_{\dot{p}}p \\ mv-Y_{\dot{v}}v & -mu+X_{\dot{u}}u & 0 & I_yq-M_{\dot{q}}q & -I_xp+K_{\dot{p}}p & 0 \end{bmatrix} \tag{2.7}$$

$$\boldsymbol{D}(\boldsymbol{\nu}) = -\begin{bmatrix} X_u+X_{u|u|}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_v+Y_{v|v|}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_w+Z_{w|w|}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_p+K_{p|p|}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_q+M_{q|q|}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_r+N_{r|r|}|r| \end{bmatrix} \tag{2.8}$$

$$\boldsymbol{g}(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{f_B} - \boldsymbol{f_G} \\ \boldsymbol{r_B} \times \boldsymbol{f_B} - \boldsymbol{r_g} \times \boldsymbol{f_G} \end{bmatrix} \tag{2.9}$$

where

$$\boldsymbol{f_G} = \boldsymbol{R^T}(\boldsymbol{q}) \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix}, \boldsymbol{f_B} = \boldsymbol{R^T}(\boldsymbol{q}) \begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \tag{2.10}$$

As system identification and modeling is outside the scope of this project, we won't go further into the details regarding how these matrices are derived, the important part for us is that they exist and can be used for simulating different systems.

BlueROV2 Heavy

The BlueROV2 Heavy is an Inspection-Class micro ROV, modified from the BlueROV2 with two extra thrusters. They are configured such that there is four vertical and four

Parameter	Description	Unit
m	Mass	Kg
W	Weight	Newton
B	Buoyancy	Newton
$\boldsymbol{r_b}$	Distance from CB to center of vehicle frame	m
$\boldsymbol{r_g}$	Distance from CG to center of vehicle frame	m
I_x	Inertia moment around x_b	kg m^2
I_y	Inertia moment around y_b	kg m^2
I_z	Inertia moment around z_b	kg m^2
$X_{\dot{u}}$	Added mass surge	kg
$Y_{\dot{v}}$	Added mass sway	kg
$Z_{\dot{w}}$	Added mass heave	kg
$K_{\dot{p}}$	Added mass roll	m^2/rad
$M_{\dot{q}}$	Added mass pitch	m^2/rad
$N_{\dot{r}}$	Added mass yaw	m^2/rad
X_u	Linear damping surge	Ns/m
Y_v	Linear damping sway	Ns/m
Z_w	Linear damping heave	Ns/m
K_p	Linear damping roll	Ns/m
M_q	Linear damping pitch	Ns/m
N_r	Linear damping yaw	Ns/m
$X_{u\|u\|}$	Quadratic dampening surge	Ns^2/rad^2
$Y_{v\|v\|}$	Quadratic dampening sway	Ns^2/rad^2
$Z_{w\|w\|}$	Quadratic dampening heave	Ns^2/rad^2
$K_{p\|p\|}$	Quadratic dampening roll	Ns^2/rad^2
$M_{q\|q\|}$	Quadratic dampening pitch	Ns^2/rad^2
$N_{r\|r\|}$	Quadratic dampening yaw	Ns^2/rad^2

Table 2.1: Parameters in the different matrices in 2.5

horizontal thrusters, allowing for full 6DOF movement. It is rated to dive to 100 meters.

BlueROV2 Heavy Modified

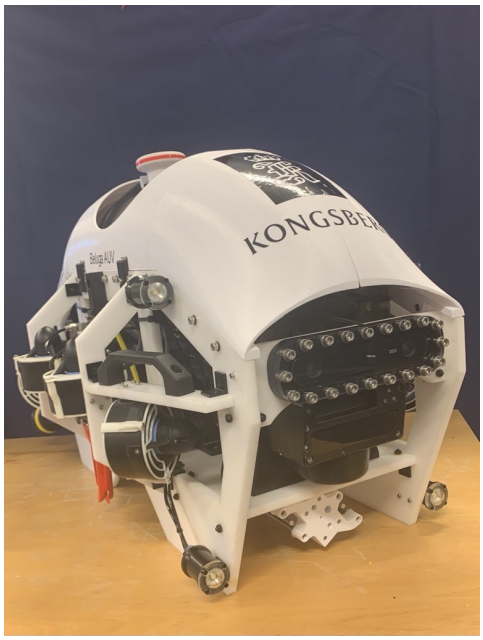
An experimental version of the BlueROV2 Heavy with additional sensors, modifying the weight and center of mass. Created internally at ITK for research.

Beluga Mk.2

Vortetx NTNU is a student organization specializing in autonomous underwater drones. Their latest project is the Beluga. It is their first drone developed with fully autonomous behaviour in mind (*Vortex NTNU* 2022)

Figure 2.4: BlueROV2 Heavy (*BlueROV2* 2022)

.



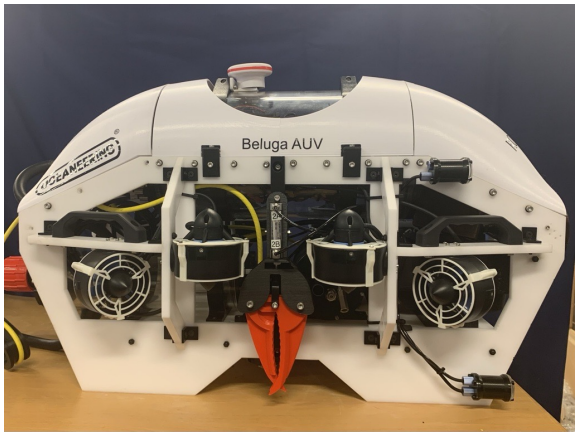


Figure 2.5: The Beluga Mk.2

2.1.4 Nonlinear PD-control law

Because maritime operations and automation is a popular and broad field, documentation and research already exists. Utilizing Fjellstad and Fossen 1994, we get a control equation in the form of a PD controller:

$$\boldsymbol{\tau} = -\boldsymbol{K_d}\boldsymbol{\nu} - \boldsymbol{K_p}(\boldsymbol{q})\boldsymbol{z} + \boldsymbol{g}(\boldsymbol{q}) \tag{2.11}$$

Where the different parameters are defined as:

$$\boldsymbol{K_p} = \begin{bmatrix} \boldsymbol{R^T}(\boldsymbol{q}) * \boldsymbol{K_x} & \boldsymbol{0_{3\times3}} \\ \boldsymbol{0_{3\times3}} & c * \boldsymbol{I_{3\times3}} \end{bmatrix}$$
$$\boldsymbol{K_x} = K_x * \boldsymbol{I_{3\times3}}$$
$$\boldsymbol{K_d} = K_d * \boldsymbol{I_{6\times6}}$$

Parameter	Meaning
K_x	Linear position proportional gain
c	Angular proportional gain
K_d	Linear and angular derivative gain

Table 2.2: PD-controller parameters

$\boldsymbol{z}$ is the error vector containing the position and attitude error:

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} - \boldsymbol{x_d} \\ sgn(\tilde{\eta})\tilde{\boldsymbol{\epsilon}} \end{bmatrix} \tag{2.12}$$

$\boldsymbol{x}$ and $\boldsymbol{x_d}$ is the current position and desired position. The error in attitude is given by

$$\tilde{\boldsymbol{q}} = \bar{\boldsymbol{q}}_{\boldsymbol{d}}\boldsymbol{q} = \begin{bmatrix} \eta_d & \boldsymbol{\epsilon}_d^T \\ -\boldsymbol{\epsilon_d} & \eta_d \boldsymbol{I_{3\times3}} - \boldsymbol{S}(\boldsymbol{\epsilon_d}) \end{bmatrix} \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{2.13}$$

where $\bar{\boldsymbol{q}}_{\boldsymbol{d}}$ is the complex conjugate of desired attitude and $\boldsymbol{q}$ is current attitude. By taking the sign of the real part and multiplying it with the complex part of the error quaternion we get the attitude error.

2.2 ROS2

ROS2 is an open-source, middleware and meta operating system used for robot systems. Its main goal is to support code reuse in researching and development of robots. The motivation to use ROS is to prevent implementation of already existing software infrastructure on new robot systems, and therefore facilitate for more efficient use of a developers time. It provides tools and libraries for obtaining, building, writing, and running code across multiple computers. It also provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. Even though, it is not defined as a operating system. (*ROS/Introduction - ROS Wiki* 2022)

The key principles we have used with ROS is the use of: nodes, messages, topics and packages. Nodes are independent programs that performs specific tasks and can publish or subscribe to topics. Topics are defined paths of communication that sends specific types of data between the nodes.

Big differences between ROS and ROS2 are the decentralized nodes, with the removal of ROS Master, and the added security.

Chapter 3

Method

This chapter focuses on our solution to the given problem. Choices made along the way will to some extent be motivated and reflected upon throughout the chapter. It will present how the ROS2 control node have been developed, with all its functionalities, and how the dynamic simulated system have been derived and implemented.

3.1 ROS2 control

For the project, the group wanted to make a modular codebase in order to work in parallel with each other efficiently. The client also expressed the need for having a modular setup to integrate with an existing ROS2 system. Hence, the group decided in a ROS2 node, containing exchangeable modules in order to easily integrate with the clients system, as well as make it possible to use on any other system for an UUV with 6DOF. In this part of the report we will explain the general flow of the ROS2 node, as well as elaborate on how solutions were reached.

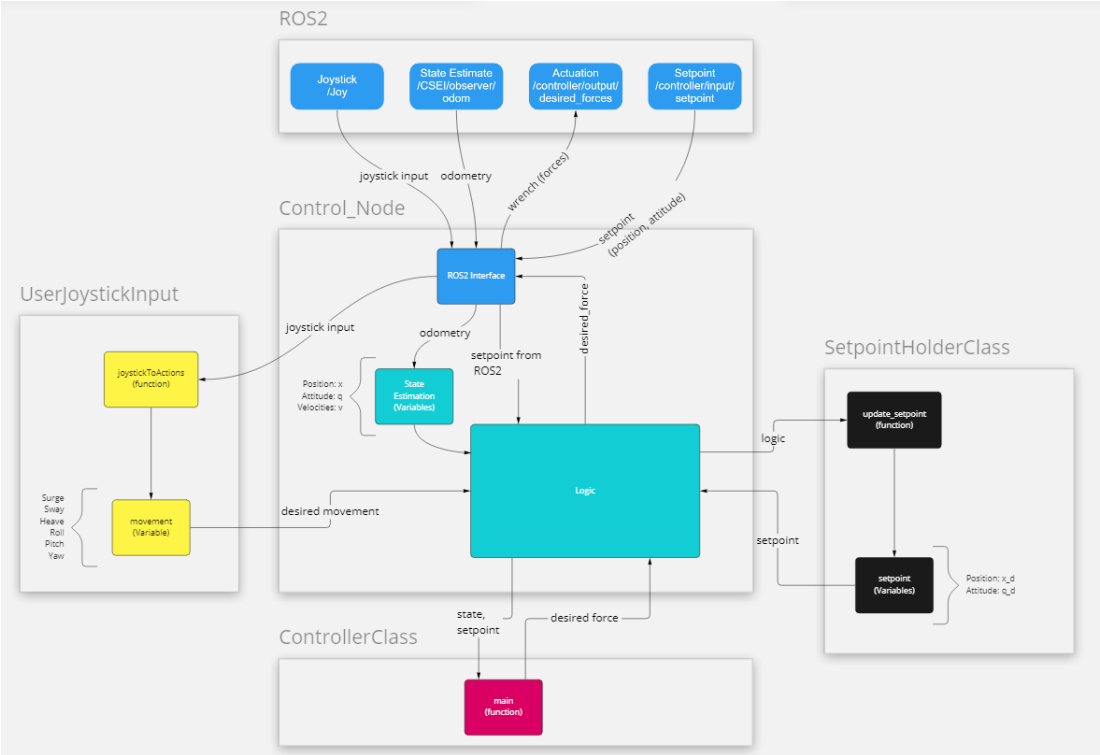


Figure 3.1: Controller pipeline map

Figure 3.1 shows an overview of ROS2 system, containing the information flowing to and from the node. The map also shows the internal informational flow of the node.

3.1.1 Control Node

The control node is the actual module that is implemented in a larger system. This module takes care of all the logic regarding control in a ROS2 system. It has been designed to be used by different systems. Essentially it will receive a message containing desired movement and state estimation, pass the information to the controller which returns a vector containing linear and angular forces to be applied upon the vehicle. The forces are denoted as $\boldsymbol{\tau}$ = **(surge, sway, heave, roll, pitch, yaw)** in the body frame $\boldsymbol{\{b\}}$. This vector is then published to the ROS2 message board, ready to be converted to thrust.

3.1.2 Operator Input

Remotely operating a vehicle using a joystick is a useful feature. Hence a teleoperation system to convert operator input to standardized movement was implemented. Joystick input is retrieved from a $sensor\_msgs/Joy$ message produced by for example an XBOX controller together with the "joy" library. This message contains information of all axes and button presses on the joystick, and represents their values between -1.0 and 1.0.

This is mapped to the corresponding action we want the ROV to perform with regards to surge, sway, heave, roll, pitch, yaw. Joystick input is mapped to ship motion convention standard to improve modularity and intuitive usability for the operator. Remapping of the button and joystick is however made easy to change. Suggested button mapping can be viewed in figure 3.2.

Figure 3.2: Xbox controller mapping

As roll, pitch and yaw are represented as euler angles, this invites possible Gimbal lock problems. This was a problem with the first iteration of joystick input we tried out in Action as setpoint, but was solved in the final method described in Action as output.

Action as setpoint

The first method of converting joystick input to desired action, surge, sway, heave, roll, pitch, yaw, involved directly interacting with the setpoint, by changing it relative to the body frame of the vehicle. For example, if an operator presses forward on the left stick, the setpoint is moved along the x axis of the body frame some distance relative to the current position. As long as the operator keeps pressing the stick, the setpoint will be continuously updated to be a relative distance along that x axis. When the input

is released the step is collapsed to whatever the current position is. The advantage of using this type of teleoperation input is that every action goes through the controller, and the controller may compensate for disturbances without interruptions. This works well on paper, but we found out that this makes teleoperation hard to use. Seeing when the setpoint collapses to the estimation value, the controller is in the middle of a step and a new step will arrive in the opposite direction. Together with delays in sensor feedback, this makes the controller overshoot and the vehicle will settle at a different position than where the joystick is released.

When inputting roll, pitch, yaw the joystick value is converted from euler angles to quaternion representation, meaning Gimbal lock situations may occur if the step is equal to 90 degrees. To prevent a 90 degree turn, we scale the action input to be smaller than $\pm\frac{\pi}{2} \approx \pm 1.57$. The relative step in a rotational direction will then never reach 90 degrees, and we avoid Gimbal lock problems. However, restricting step size also restricts reaction time and speed for rotational input, which is a undesirable attribute when using a joystick.

Action as output

The second method was to map input directly to output. As previously mentioned, the joystick input is mapped to standardized movement surge, sway, heave, roll, pitch, yaw in the body frame. This corresponds to the same format as the controller output. Therefore it is a natural conversion from joystick input to override the controller output. An advantage of this solution is that manual control responds much more quickly, and may operate regardless of controller parameters. A disadvantage of this method is that it bypasses the controller which may cause problematic situations in strong currents or other turbulence. To handle this problem, logic was implemented for the manual control to only be applied in the inputted direction, while the controller remains active the remaining directions. This proved to work very well. Setpoint updates happen when releasing the joystick, which saves the current position and orientation as the setpoint. Contrary to the previous solution with joystick to setpoint changes, this made maneuvering the ROV with joystick very comfortable and smooth.

3.1.3 Configurability

Since not all UUV's are created equal, where some are autonomous and some are not, or some have feedback possibilities and some do not, the node includes a configuration file to let the user configure it to their need. For example, if a user has access to state estimation, the node can use this as feedback for the controller. The control equation (2.11) allows for decoupling position and angular movement, implied by the diagonal structure of the $\boldsymbol{K_p}$ matrix. Having a decoupled controller enables control of position

and attitude separately, and the group has implemented functionality to take advantage of this. Keep in mind that this does not mean that the dynamics of the system is decoupled, only the elements being controlled.

Basically the node can be used in a variety of ways based on the operators preference, and the following sections will explain what those ways are.

Pure teleoperation

When using our package purely for teleoperations, the controller logic is skipped and the desired movement from the XBOX controller is directly transformed into $\boldsymbol{\tau}$ in the body frame and published to the message board. The standardized inputs for the XBOX controller can be viewed in figure 3.2 above. By using this map, if the user wants the vehicle to go forward, the left stick should be moved to the forward position.

Attitude-control

The controller can be enabled without controlling the linear movement. It is designed like this to work on UUV's with only angular feedback, since angular estimation is common whilst positional estimation requires more sensors and filtering. By enabling this feature the node will retrieve information from an external state estimate through a *nav_msgs/Odometry* message and use the attitude and angular velocities in the controller. Essentially, what this means for equation 2.11 is setting the positional effects as zero:

$$\boldsymbol{z} = \begin{bmatrix} 0 & 0 & 0 & sgn(\tilde{\eta})\tilde{\varepsilon_1} & sgn(\tilde{\eta})\tilde{\varepsilon_2} & sgn(\tilde{\eta})\tilde{\varepsilon_3} \end{bmatrix}^T$$
$$\boldsymbol{\nu} = \begin{bmatrix} 0 & 0 & 0 & u & v & w \end{bmatrix}^T$$
$$\boldsymbol{q} = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T$$

Linear-Z-control

Sensors providing data to create position estimates in the xy-plane underwater may be expensive and can vary greatly in quality of measurements. However, depth information may be created by much cheaper components and less work. Therefore, xy-plane and z-axis controls have been split to function as standalone, but mergeable components. This can facilitate the use of a depth/altitude hold function by manipulating the Setpoint from topic function.

For the equation2.11 this functionality means:

$$\boldsymbol{z} = \begin{bmatrix} 0 & 0 & z_d & sgn(\tilde{\eta})\tilde{\varepsilon_1} & sgn(\tilde{\eta})\tilde{\varepsilon_2} & sgn(\tilde{\eta})\tilde{\varepsilon_3} \end{bmatrix}^T$$
$$\boldsymbol{\nu} = \begin{bmatrix} 0 & 0 & z & u & v & w \end{bmatrix}^T$$
$$\boldsymbol{q} = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T$$

Note that attitude estimation is assumed available at all times when using the controller.

Linear-XY-control

When an operator wants full control of a UUV, this feature can be enabled. As functionality goes, this feature has the same base as Linear-Z-control. However the equation in 2.11 will have the following adjustments:

$$\boldsymbol{z} = \begin{bmatrix} x_d & y_d & z_d & sgn(\tilde{\eta})\tilde{\varepsilon_1} & sgn(\tilde{\eta})\tilde{\varepsilon_2} & sgn(\tilde{\eta})\tilde{\varepsilon_3} \end{bmatrix}^T$$
$$\boldsymbol{\nu} = \begin{bmatrix} x & y & z & u & v & w \end{bmatrix}^T$$
$$\boldsymbol{q} = \begin{bmatrix} \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \end{bmatrix}^T$$

Setpoint from topic

This feature adds the possibility to define the setpoint of the controller from a topic on the ROS2 message board, instead of being defined by joystick release. The message containg the setpoint is a *geometry_msgs/Pose* message. The pose is a combination of position (x, y, z) and a quaternion attitude (x, y, z, w). Having this feature active however does not mean disregarding the joystick completely, as teleoperation may be used to move the vehicle. However, the setpoint is not changed by joystick release and the vehicle will be moved to the setpoint provided by ROS2 topic. Reading setpoint from a topic is a necessary feature for pathfollowing, as the setpoint may be provided by computer generated programs.

3.1.4 Data Logging

As soon as the group had a rudimentary controller for testing, the need for data logging became important. Functionality was implemented to write out desired data to a .xlsx file on the system running the control node. The data-points logged can be viewed in table 3.1 together with a description of the data.

$\boldsymbol{\tau}$	vector containing angular and linear forces from the controller
$\boldsymbol{z}$	current error vector
$\boldsymbol{q}$	current attitude
$\boldsymbol{q_d}$	desired attitude
$\boldsymbol{x}$	current position
$\boldsymbol{x_d}$	desired attitude
$\boldsymbol{\nu}$	velocity vector
axis input	current input from the xbox controller

Table 3.1: Logged Data

This feature works passively on the control system. However, when the group wanted to plot step-responses, the .xlsx file became too tedious to read when the system had been running for a longer period of time. Therefore the group created functionality to start logging when pushing a button (**B**), and stop when pushing the same button again. To aid in data collection, as well as being a useful feature, we also added a button press (**A**) to reset the attitude setpoint to $\boldsymbol{q} = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$. Effectively this allows us to easily reset the attitude to a fixed orientation before making step responses. If the attitude estimation is set to have $\boldsymbol{q} = \begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$ as neutral pitch and roll, this functionality will also make operating with a joystick more convenient.

3.1.5 Development

When it comes to choosing a language as a development platform, it is important to keep in mind where the product will be used. The project problem describes the control of an underwater vehicle integrated in a ROS2 system, specifically running on a RasberryPi 4B.

Different types of hardware provides different levels of processing power and memory capabilities. A RasberryPi, although a common hardware component for smaller ROVs and other embedded systems, offers limited processing power and memory compared to any modern personal computer. Hence, using a fast and memory efficient development language makes the system generally more applicable.

Initially the contenders were C++ and Python, but seeing as C++ is faster, more efficient and less prone to runtime errors compared to Python (*Difference between Python and C++* 2020), the group decided on C++. As a compiled language, C++ is less exposed to data type and bad input problems. However, a drawback to C++ is its complexity and developement time compared to Python. None of the group members had previously worked with C++ and this led to the group learning C++ while simultaneously working with it. A very worthwhile exercise nonetheless, since C++ competence is a useful skill to acquire.

3.2 Simulation study

Performing simulations to get rough baselines for the control parameters and system response, makes tuning and physical testing faster and more efficient. Simulation can have different meanings, from virtual environments to system response tests. We looked for any existing solutions to use, but ultimately decided to simulate the non-linear dynamic system using python ourselves. Although this type of simulation may provide very useful results, it requires good mathematical modeling of the system dynamics.

3.2.1 Mathematical principle

The total kinetic model of an underwater vehicle is given by equation 2.5. By solving this equation for $\dot{\boldsymbol{\nu}}$ we get:

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{g}(\boldsymbol{q})\right) \tag{3.1}$$

By inserting the equation for the PD-controller (2.11) for $\boldsymbol{\tau}$, we get an ordinary differential equation (ODE) representing the system kinetics. Combining this with equation 2.4, we get a set of ODE's representing the systems dynamics:

$$\begin{bmatrix} \dot{\boldsymbol{\xi}} \\ \dot{\boldsymbol{\nu}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}(\boldsymbol{q})\boldsymbol{\nu} \\ \boldsymbol{M}^{-1}(-\boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} - \boldsymbol{K_p}(\boldsymbol{q})\boldsymbol{z} - \boldsymbol{K_d}\boldsymbol{\nu}) \end{bmatrix} \tag{3.2}$$

Solving these equations using Python and Scipys odeint we get a 13x1 vector containing the states of the system at every desired timestep:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{q} \\ \boldsymbol{\nu} \end{bmatrix} = \begin{bmatrix} x & y & z & \eta & \epsilon_1 & \epsilon_2 & \epsilon_3 \\ & & & & u & v & w & p & q & r \end{bmatrix}^T \tag{3.3}$$

where $\boldsymbol{x}$ is the position, $\boldsymbol{q}$ is the orientation (in quaternions), and $\boldsymbol{\nu}$ is linear- and angular velocities.

In addition to the ODE's, odeint also needs initial conditions and time points for when to return solutions. The simulation is therefore set up to take initial conditions for $\boldsymbol{x}$, $\boldsymbol{q}$ and $\boldsymbol{\nu}$ as well as desired endpoints for the aforementioned. The time points are also automatically spaced to emulate the same sampling time as the controller runs on the real system.

3.2.2 Dynamics

Using modeling parameters derived by Wu 2022, we have everything we need to simulate the BlueROV2 Heavys dynamics. Since some of our tests will be done on a modified BlueROV2 Heavy, the physical responses may differ from the simulated ones. The parameter values used can be found in tables A.1, A.2 and A.3.

To get some sort of verification on how the simulation corresponds to the real world, we decided to compare control parameters at the point where the system becomes unstable. Depending on how different the responses are, this test will give some indication on how well the simulation represents the real world situation.

The BlueROV2 Heavy is weighed down to make it neutrally buoyant. The placements of these weights affect the CG and inertia moments. The tether will also affect the vehicle by adding extra drag and will try to rotate it such that the tether and connection point are perpendicular to each other, resulting in disturbances in yaw.

It is worth noting that the simulation environment we created is adapted to a BlueROV2 Heavy, but may be applicable to other systems by simply changing the content of the different dynamic matrices.

3.2.3 Results

To get a baseline for parameters the following step responses were simulated:

	45°	90°	135°	180°
Roll	x	x		
Pitch	x	x		
Yaw	x	x	x	x

Table 3.2: Simulated step responses

As most UUVs are created with the intent for inspecting and precise work, they do not really require overly acrobatic movements. This was the mindset when deciding which steps to simulate. Rotations in yaw are useful in most circumstances, and pitch are useful when inspecting objects. Roll is not used frequently, but it is useful to know that parameters that are good for yawing and pitching are also good for rolling. Values for position may be utilized in the simulation, but since the state estimate was not available on the vehicle when the initial testing was performed, the linear gain was set to zero to simulate the real world as much as possible.

After going through multiple simulations with different parameters for $\boldsymbol{c}$ and $\boldsymbol{K_d}$, it was concluded that $\boldsymbol{c = 20}$ and $\boldsymbol{K_d = 5}$ gave a satisfactory result for most of the steps. The

deciding factors were the angular accelerations and settling times. These parameters gave a snappy initial response, and about a second of settling time.

Chapter 4

Results

This chapter contains plots of the step-responses belonging to the vehicles used in the project, the pathfollowing of the modified BlueROV2 Heavy, and the realism tests for the simulator.

4.1 Response tests

The following figures show some of the step responses performed on the underwater vehicles. These have been selected specifically to give the best overview of how the controller performs in regards to different setpoints. For both BlueROV2 Heavys, the simulated step-responses are placed in the same graphs as the measured responses. The gains used for the step responses are shown in table 4.1.

	Parameters		
ROV	Kx	c	K_d
BlueROV2 Heavy	0	20	5
BlueROV2 Heavy Modified	0	20	5
Vortex Beluga Mk.2	0	20	5

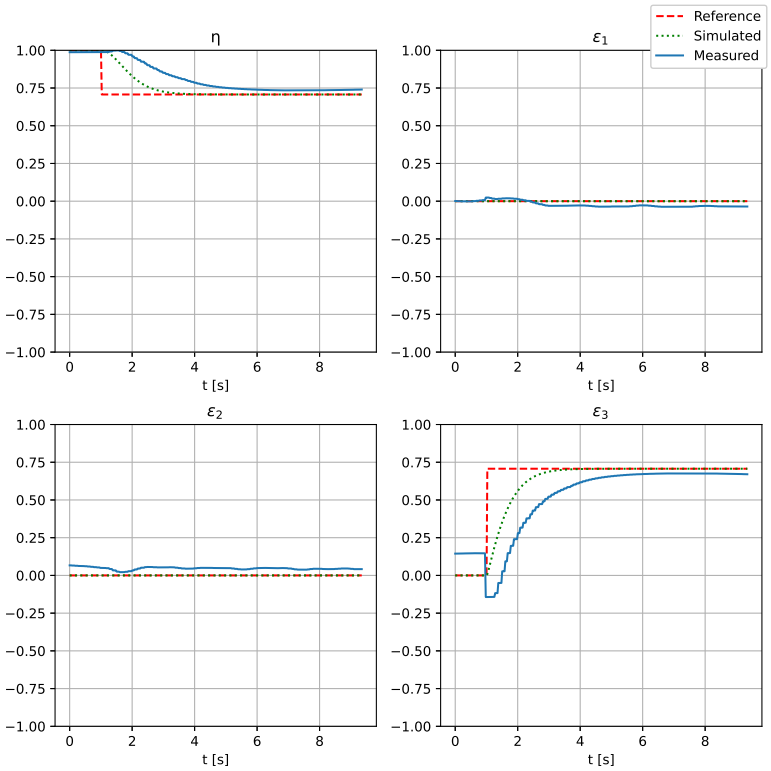
Table 4.1: Step-response parameters

Some figures, especially the ones containing roll and pitch, may display some irregularities. These irregularities are manual interventions to prevent the vehicle from crashing into the walls of the testing environment, which was a small water tank, and are regarded as sources of error. This is due to the lack of position control and the fact that when the vehicle pitches or rolls, it naturally starts to drift.

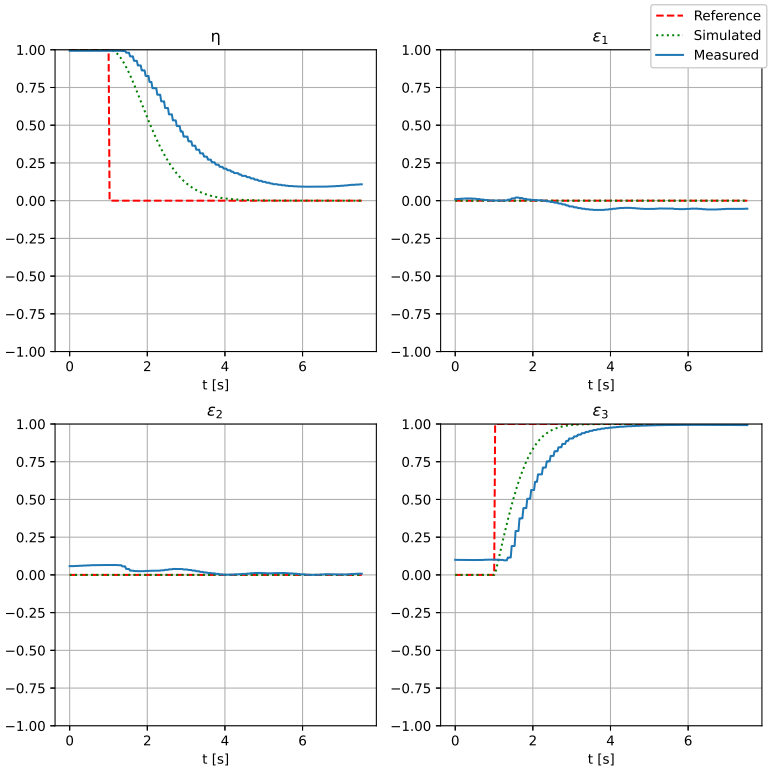
Another discrepancy, especially in the modified BlueROV2 Heavys figures, one can see that the measured values are moving in the opposite direction of the setpoint. This is because of the quaternions. Since quaternions are of a periodic nature, the same

physical orientation may be expressed in two different ways mathematically. Every step response was checked for correct physical representation during testing.

4.1.1 Standard BlueROV2 Heavy

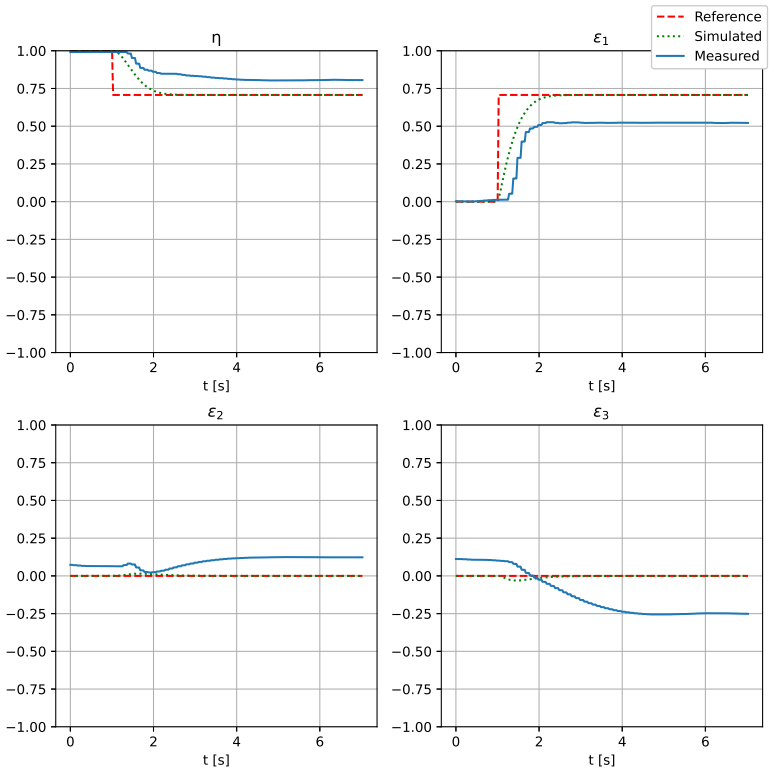


(a) 90°yaw

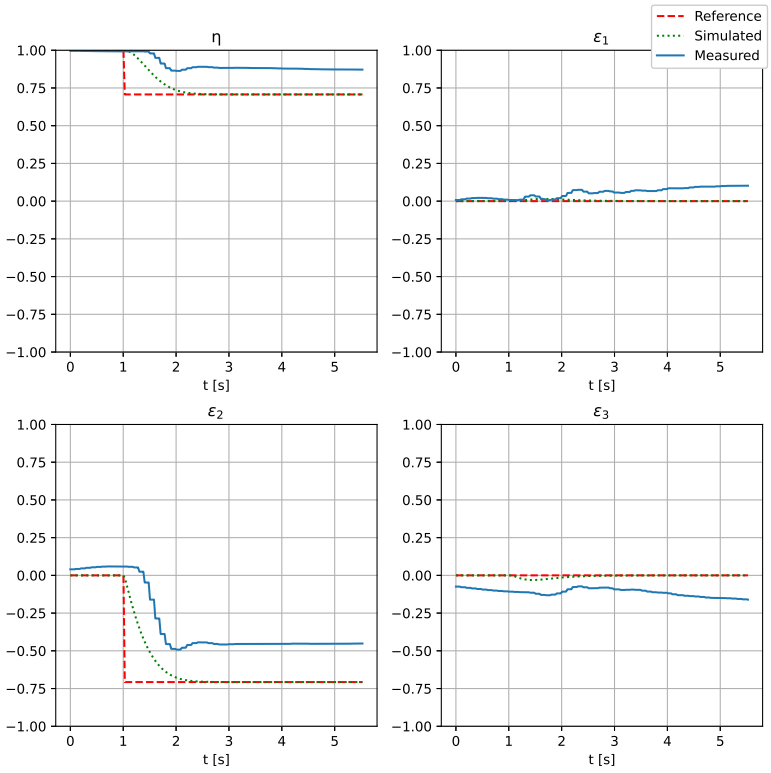


(b) 180°yaw

Figure 4.1: Step responses in yaw



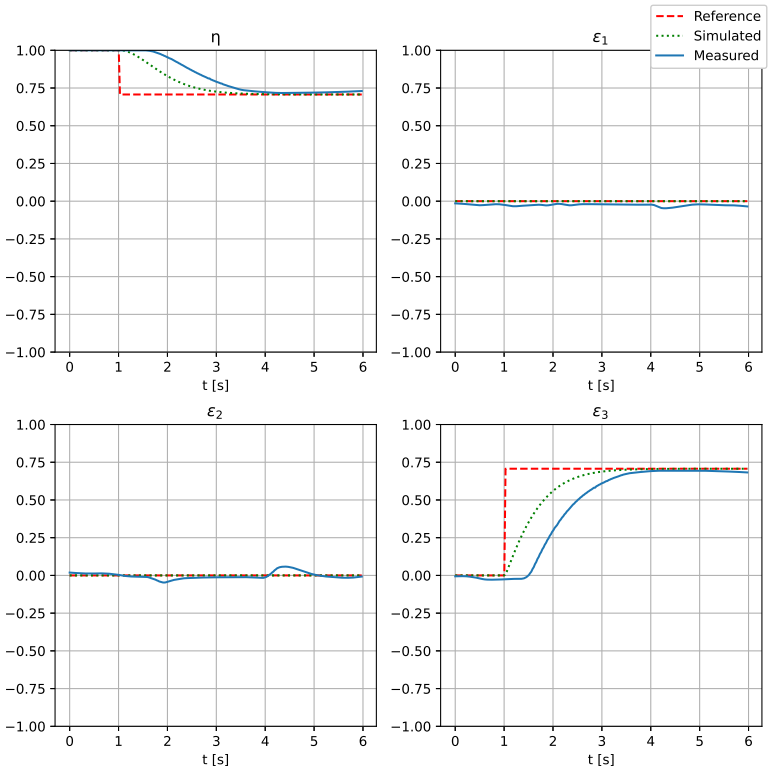
(a) 90°roll



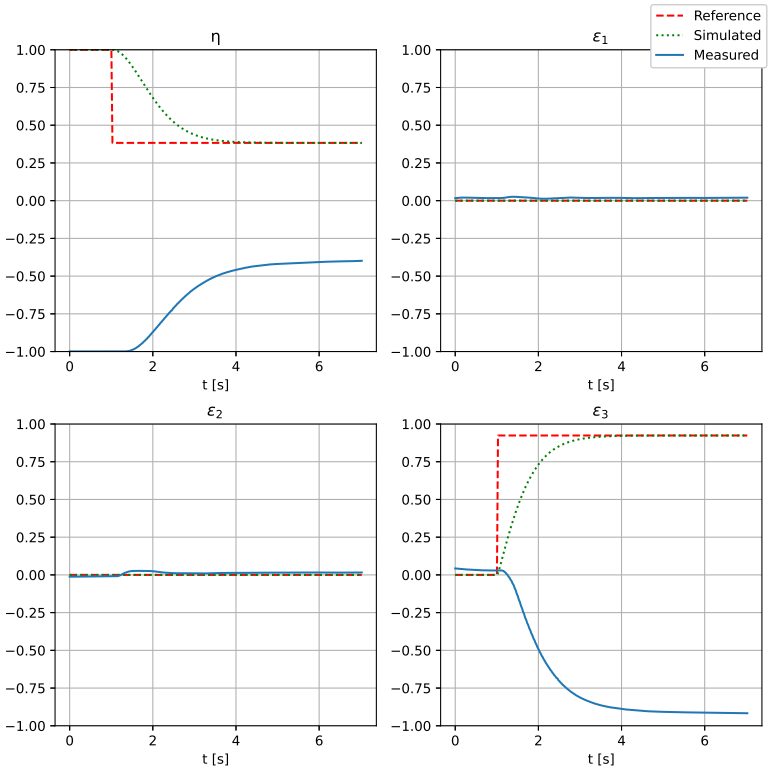
(b) 90°pitch

Figure 4.2: Step responses in roll and pitch

4.1.2 Modified BlueROV2 Heavy

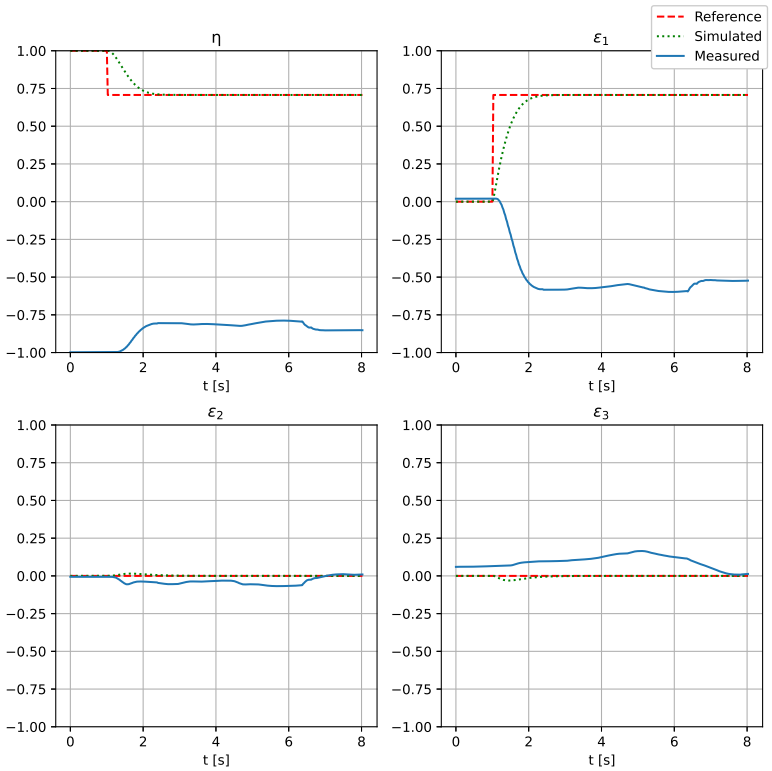


(a) 90°yaw

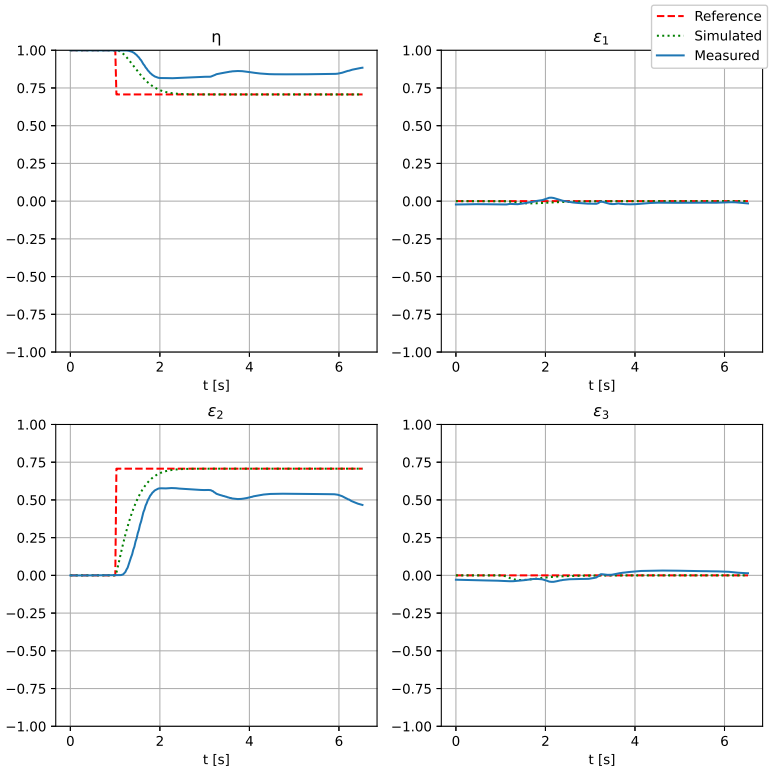


(b) 135°yaw

Figure 4.3: Step responses in yaw



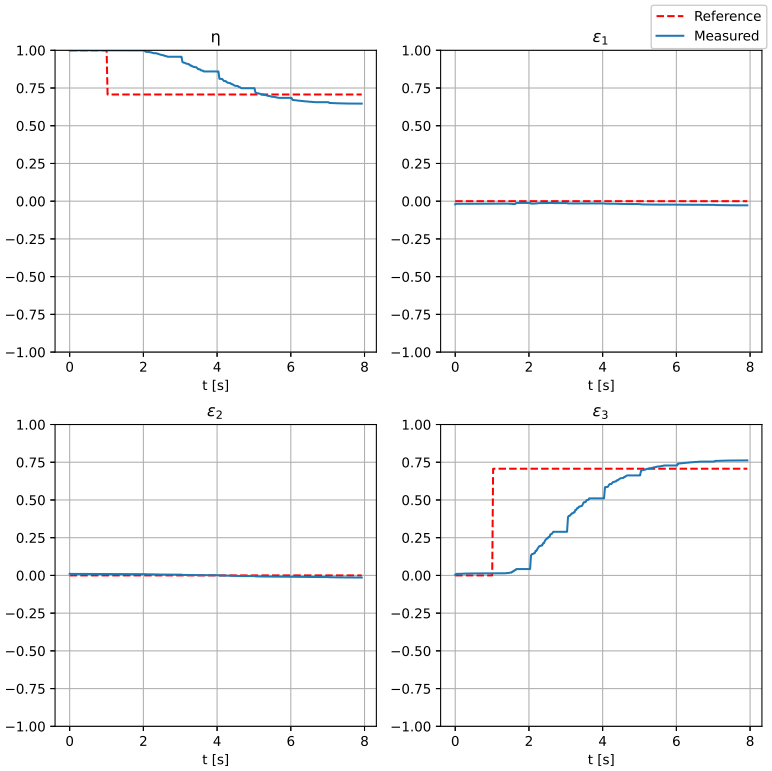
(a) 90°roll



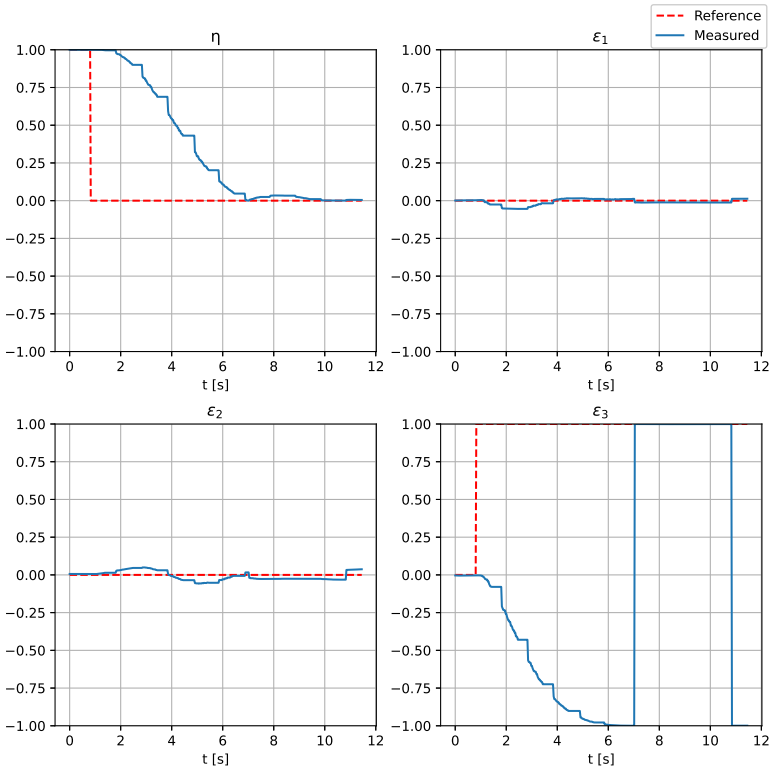
(b) 90°pitch

Figure 4.4: Step responses in roll and pitch

4.1.3 Vortex Beluga Mk.2

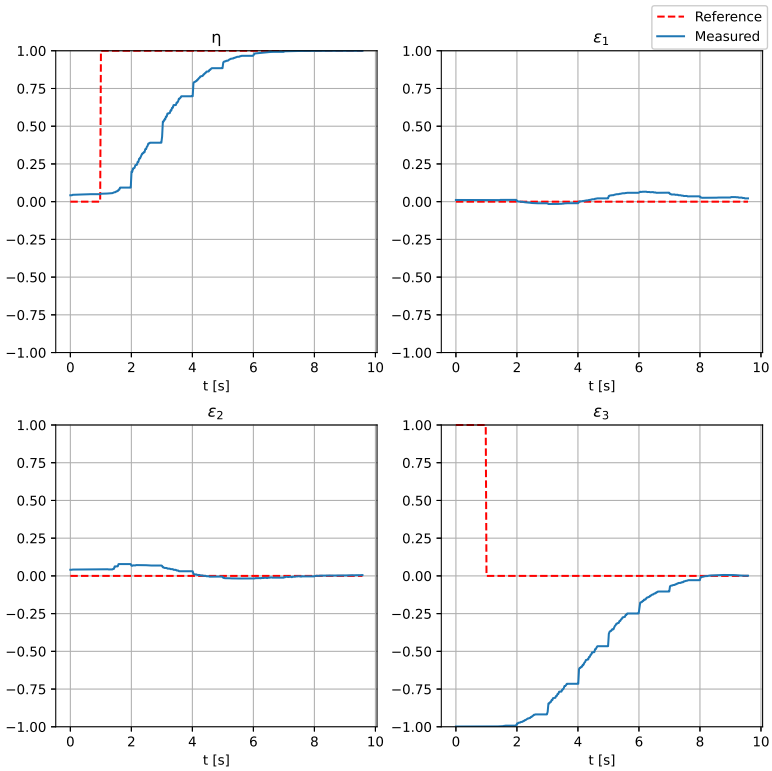


(a) 90°yaw

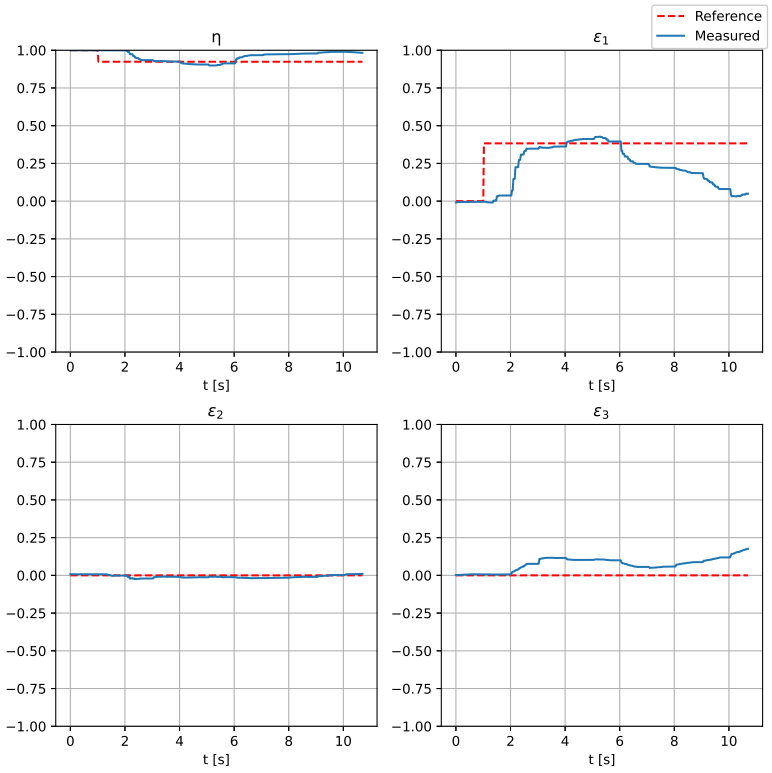


(b) 180°yaw

Figure 4.5: Step responses in yaw



(a) 45°roll

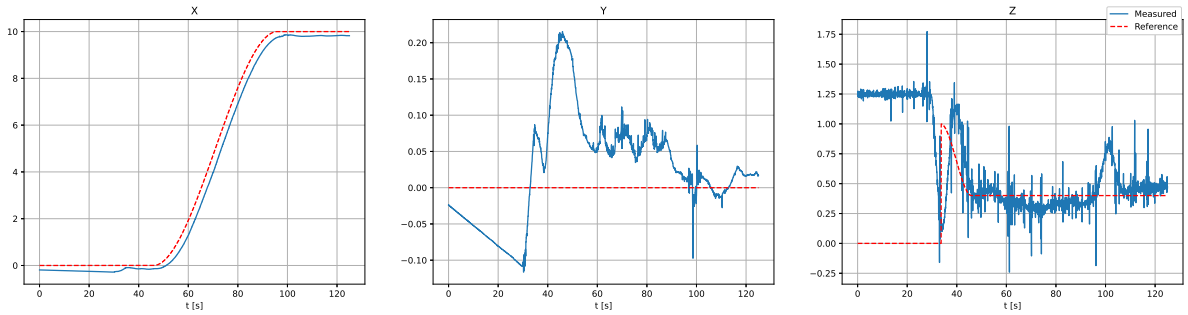


(b) 45°pitch

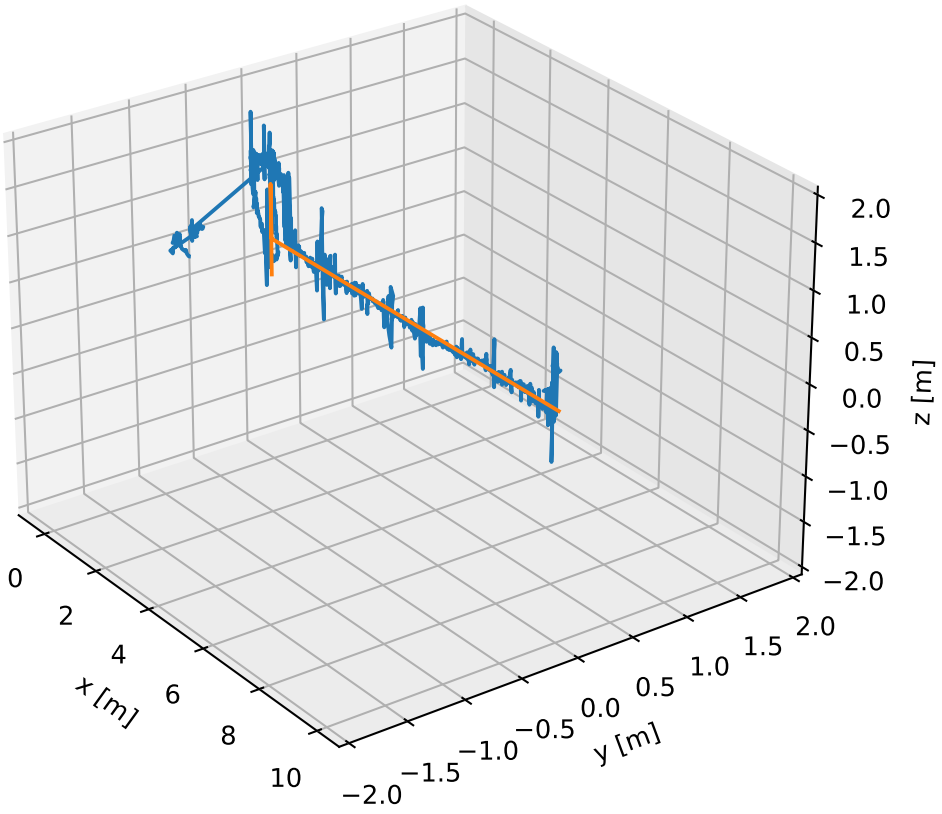
Figure 4.6: Step responses in roll and pitch

4.2 Path-following

At a late stage of the project, a state estimate containing position became available to the group. This state estimate was however at a prototype stage, and we found that changes in attitude reference values caused the x- and y-values to drift. The test was performed by raising the vehicle to a depth of 0.4 meters and then surging 10 metres. Although figure 4.7 shows how the measurements in x and y are quite noisy, the vehicle managed to smoothly move to the desired endpoint.



(a) 2D



(b) 3D

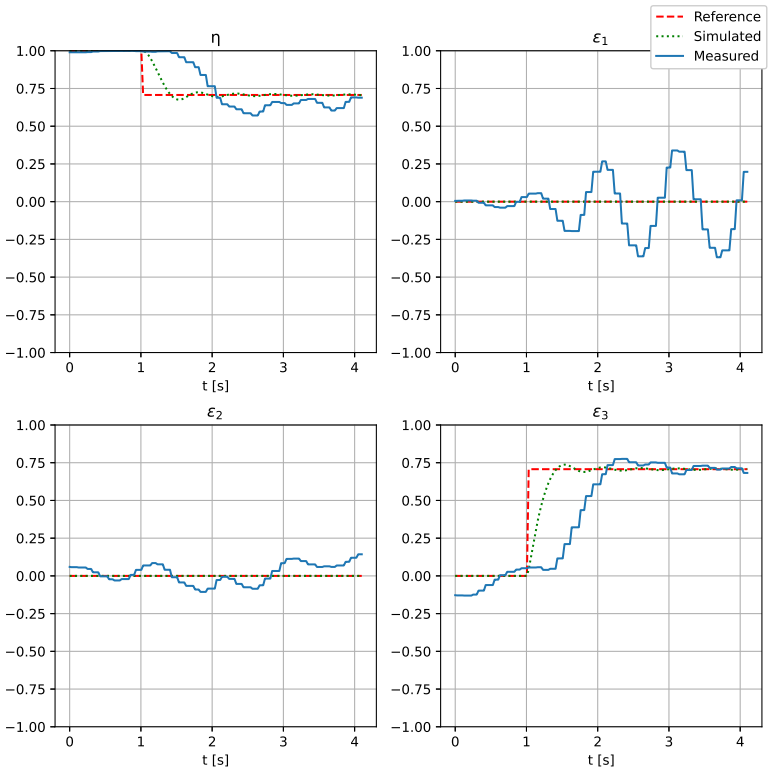
Figure 4.7: Trajectory following in 2D and 3D

4.3 Realism test

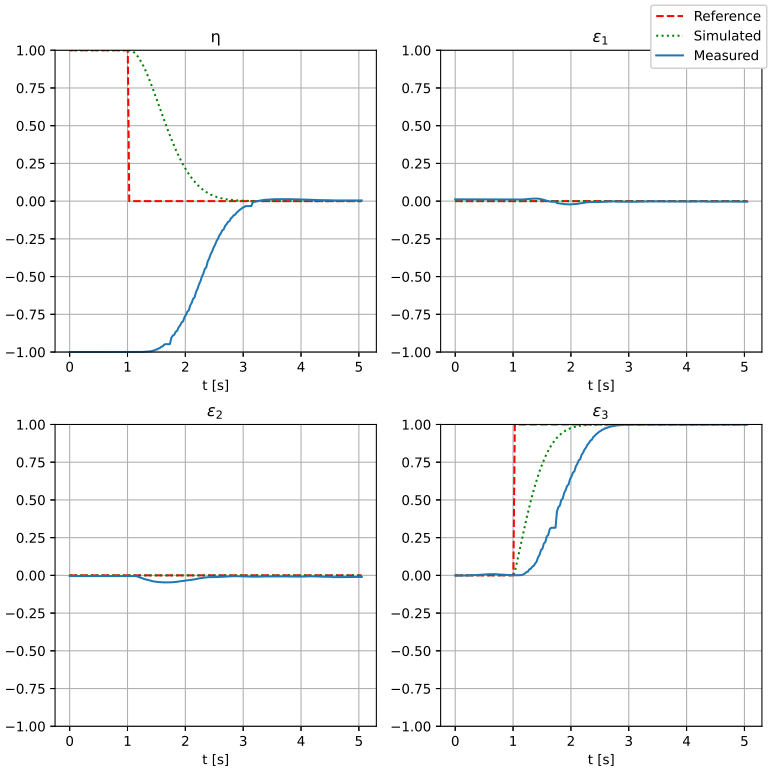
The reason we conducted the realism test was to compare the control in the real world to control in a simulated environment. As mentioned in 3.2.2, step-responses with both ”good” and ”bad” parameters were tested. Due to time limitations, only ”dull” and ”extreme” steps were taken, and they were only tested on the standard BlueROV2 Heavy. The different gains for each step-response can be found in table 4.2:

BlueROV2 Heavy	Parameters		
Step	Kx	c	K_d
90°yaw	0	80	0
180°yaw	0	40	5
180°roll	0	50	5
180°pitch	0	10	10

Table 4.2: Gains for the different step responses

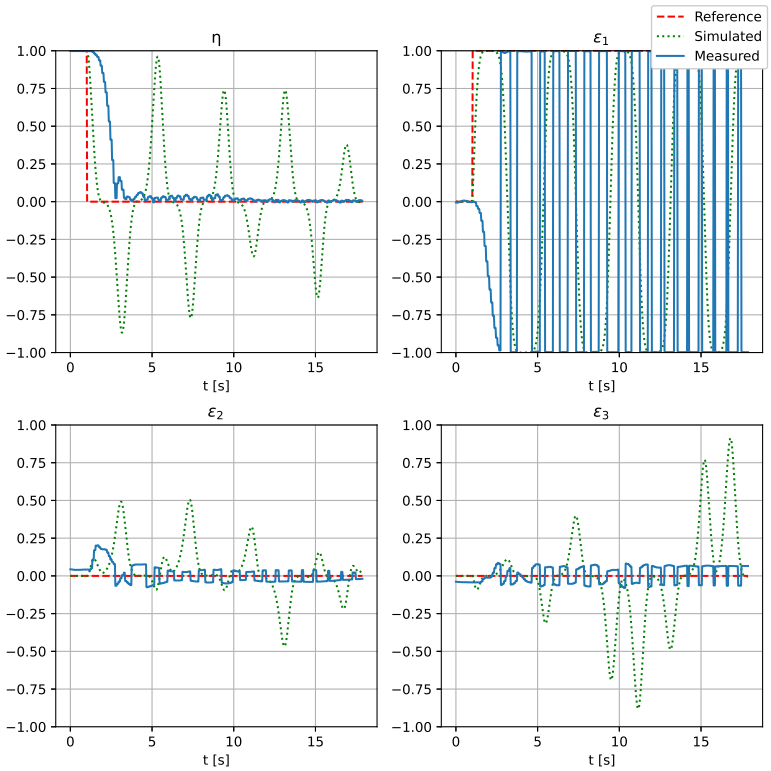


(a) 90°yaw

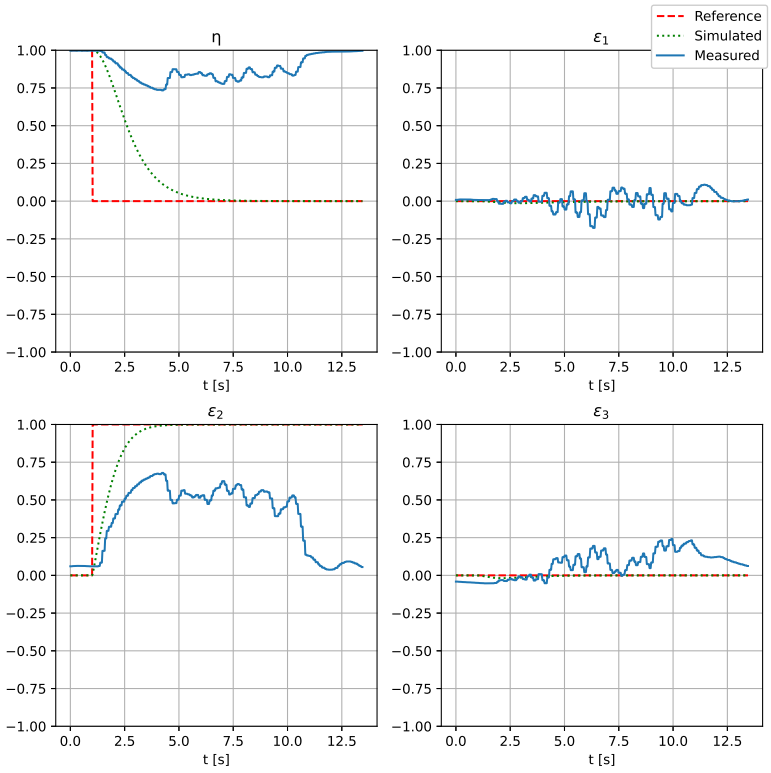


(b) 180°yaw

Figure 4.8: Step responses in yaw



(a) 180°roll



(b) 180°pitch

Figure 4.9: Step responses in roll and pitch

Chapter 5

Discussion

In the discussion chapter we intend to evaluate the results gathered through response tests, as well as strengths, weaknesses and application of the final product. We will also take a look at knowledge and experiences we have gathered and compare it with expectations from the preliminary project.

5.1 Results

5.1.1 PD Controller

Dynamic positioning is a vessels ability to maintain position and heading automatically, using its own thrusters and sensors. Fjellstad and Fossen (Fjellstad and Fossen 1994) derives such a dynamic positioning system using the nonlinear dynamics of an underwater vehicle. The resulting controller is in the form of a PD-controller with the control equation as seen in equation 2.11.

Early on in the project we decided to base our system on the dynamic positioning controller derived by (ibid.). There are several reasons for this, but the first and most important being the fact that this controller used quaternions to represent attitude. As well as being requested by our client, quaternions do not suffer from Gimbal lock and provide fast calculation speeds for a processor. Furthermore, we were informed that we would have a state estimation available for both attitude and position, which meant we would be able to utilize the dynamic positioning aspect. We later learned that the position estimation was not complete, nor perfect, but the control system is still able to handle attitude as a separate feature and therefore is still very applicable.

Since the control equation is derived from the nonlinear dynamics of an underwater vehicle, this PD controller is specialized to perform in nonlinear environments around such a vehicle. However, seeing as the controller does not have an integrator part it

may suffer from steady-state errors. When using joystick, the setpoint is chosen to be wherever the vehicle is when the joystick is released. Since the vehicle should remain as is when the setpoint is created, the controllers mission mostly becomes counteracting disturbances like drift and currents. In such cases, an integrator would make sure the vehicle returns to the setpoint even in strong currents. However, since a new setpoint is set when releasing the joystick, any windup created by an integrator part would cause it to move after joystick release.

When only using setpoint control, without a joystick, such as when using a trajectory planner, we suspect the controller will work just as well with or without an integrator. This is because a trajectory planner frequently sends small steps and, depending on step size, a small steady-state error from the trajectory is negligible. The system may still be vulnerable to drift problems caused by currents nonetheless.

5.1.2 Alternative control solutions

Another way we could have gone about the control system would be to create several controllers, each controlling its own degree of freedom. ArduSub (*ArduSub Homepage* 2022) is an example of this, were the rate of change in any direction is used as the control element. This solution requires less sensors, and does not require the same amount of information regarding the physical properties to tune well. However, if enough sensor information is available the PD controller functions better than the ArduSub solution when it comes to autonomy. The ArduSub solution may handle pure teleoperation better, as more logic is required around the PD-controller to make the vehicle stop when the joystick is released.

A completely different approach would be creating a control system based on a LQR (Linear-Quadratic Regulator). LQR may provide a more optimal control system and generally be more robust than a PID or PD controller. It will however, require a lot of trial and error in order to get the optimal cost function parameters. As LQR is not a part of our study program, we decided to prioritize PD and PID control and first develop a functioning system before expanding to LQR. Further on, LQR was completely removed in favor of system optimization. Trying out LQR instead of the nonlinear PD controller would definitely be an interesting project nonetheless.

5.1.3 Controller behavior

By analyzing the step-responses on the BlueRov2 Heavy, while taking into account the physical observations and general handling of the UUV, we consider it to be performing in a satisfactory way. Seeing as how the positional estimate needed for the positional control was not available at the time the step-responses were recorded, this performance

should therefore mainly be attributed to attitude control. Since a prototype positional control became available, we were able to perform a pathfollowing test using the PD-controller. As seen in figure 4.7b and 4.7, the UUV converged on a setpoints in a path given by a path-generator. The positional estimate did have some limitations however. It was noted that when applying a change in attitude, the current positional estimate diverged from the previous estimate. This was because the positional estimate was at a prototype stage and in no way complete.

From figures 4.1a, 4.1b, 4.3a, 4.3b we see that the controller is performing well for general yaw movements. Theese step-responses visualize the step-responses for the two BlueRov2 Heavy vehicles. The controller does not overshoot the setpoints in any of the measured step-responses. We also noted that the steady-state error was negligible for these cases. This performance is most likely due to the lack of gravitational forces affecting the pure yaw movement. Generally speaking, the hydrodynamic damping forces of the water will also help the system to keep from overshooting. For the Beluga Mk.2 the step-responses were somewhat different. Figures 4.5a, 4.5b show that the control system did in fact overshoot a small amount in some cases. As the Beluga is a lot heavier than the other two vehicles, this is mainly attributed to the tuning of the controller, but also due to a communication bottleneck between the Beluga system (ROS1) and the control code (ROS2). The message containing the state-estimate of the Beluga was sampled at a rate that led the controller to hold a reference value for longer than it should.

When analyzing the step-responses for the roll and pitch movements, we can generally see a greater steady-state error as well as overshooting in some cases. Figures 4.2a, 4.2b, 4.4a, 4.4b show the step-responses for the two BlueRov2 Heavy vehicles. We can see that the responses have the aforementioned steady-state error in the quaternion elements affected by the setpoint, as well as the element meant to stay the same. This is likely due to drift and measurement errors described in 4.1 as well as being affected by gravitational forces. This will lead to a lack of gain from the controller, preventing it from reaching the desired attitude setpoint. Due to the Belugas buoyancy and mass distribution, it was only possible to test pitch and roll up to 45 degrees as seen in figures 4.6a, 4.6b. However it handled this well, leading to less stationary error.

We mention that adding an integrator to the controller may fix the steady-state error. But, due to the fact that the PD-controller is derived form the dynamic equation, this means that there are no guarantees that the PID will maintain any of its stability characteristics. When implementing this feature it is therefore necessary to conduct an extensive stability analysis to ensure predictable behavior.

From a observational viewpoint, by seeing the handling characteristics of all the vehicles, they behave in a generally predictable way. While one can clearly see the challenges mentioned above in regards to steady-state errors, we are are generally satisfied with how the controller works.

5.1.4 Simulation and reality

From the realism test in the simulation study, we can clearly see differences between simulated and actual responses. In most cases, the simulation response is stable and without steady-state errors while the physical response remains unstable. The reason for these differences is the fact that the simulation study is based on the modelling of a BlueROV2 Heavy that may differ from the one the group used. Changes on where mass is distributed and thruster force limitations are factors that make the physical system different from the simulations. It is likely that the desired force outputted from the controller does not equal the actual force provided by the thrusters.

Arguably, the most important control lies in yaw. If control gains are kept in a reasonable area, responses in yaw control seem to perform excellent. In these cases the simulation uses shorter time to reach its reference value than its physical counterpart, although this is not by much.

With the modelling used in the simulation, it seems to be roughly usable for control parameters. However, this is not conclusive. Parameters should be fine-tuned on the physical system itself.

5.1.5 General applicability

As a proof of concept for the general applicability of our product, in collaboration with Vortex (*Vortex NTNU* 2022), we were able to test our control system on the Beluga Mk.2 UUV. The Beluga runs on ROS, while our system runs on ROS2. Even so, by running the ROS1 bridge (*Bridge communication between ROS 1 and ROS 2* 2022), we were able to communicate with the Beluga seamlessly with our ROS2 system. Without changing any controller parameters we were able to control the Beluga with ease, and the control would most likely be even better if we tuned the controller to the Beluga system. It is worth noting the Beluga already had state estimation and actuator drivers compatible with the controller output when we used it.

5.2 Product application

We are confident in claiming that the final product is satisfactory and well functioning. This does not mean it is flawless nor optimized, but rather it functions all-round as

expected and has many different use cases. However, the area we believe it shines the brightest is for researching purposes. With the ability to dynamically customize controller behaviour based on specific tasks, the control system works excellent to aid in the collection of sensor data, photographing sub-sea elements and just exploring.

Ease of use and simplicity has been a focus from the beginning. By making our package open source and readable, we improve the chances that our product will be used outside NTNU and ITK. Popular usage may lead to both improvement of the system as well as help others with their research. However, since the controller is tuned to a BlueROV2 Heavy, paremeter tuning is required for other systems which requires some level of knowledge regarding control theory.

By itself, our product does not discriminate between ROV's and AUV's. That being said, we have made specific features like teleoperation that is only used in ROVs. Our control system has the ability to operate with or without teleoperation and operator input, since state estimates and setpoints may be given by ROS2 topics. This allows for a trajectory publisher to feed a path the controller may follow. If our product is to be used in an AUV, a framework with trajectory generation and collision checking needs to be built around it since we do not provide those features as the project stands now.

5.3 Goals and experiences

In the preliminary project the overarching goal of the bachelor thesis was defined as "create a stabilizing control element for both position and attitude". As the final product stands and given the results we have achieved, we can claim that this is largely fulfilled. The goal was divided into three subcategories: effect-, result-, and process goals, which describe, what we wanted to achieve, which products we ended up delivering and the effect the project had on the participants.

5.3.1 Achieved goals and the product

We have developed a CAS which allows for easy and intuitive teleoperation as well as attitude and position control. With an intuitive, and if necessary easy to modify, button mapping giving the user full control of the UUV in 6DOF. Attitude and position control is made possible with a nonlinear PD-controller, making operations with special requirements in attitude and position easier. We can therefore say that the effect goals are fulfilled.

A working modular ROS2 node containing the controller as well as all controller functions has been developed. Parameter files for change in controller parameters, allows for real time interaction with the controller parameters, making tuning on different systems

easy. Our C++ code has been designed with focus on readability.

A hypothesis regarding battery usage and the response of the system was discussed. Since there was no battery data available, it was abandoned. The product delivered still lives up to what was planned in the preliminary project.

5.3.2 Project experience

Through the duration of this project we have acquired experience and knowledge in multiple fields. The following list highlights what we deem to be the most important experiences regarding project planning and administration.

- How project organisation is a vital part in a project development and implementation. Through the preliminary project we got experience in how to plan and organise a project. By defining the process, effect and result goals, and creating work packages we have learnt the importance of a strong foundation. This is key for project success. Organisation throughout the project, with the use of Gant diagrams and timesheets, is also important for documentation of progress and milestones. By having regular meetings with both the supervisor and client we have also acquired experience in organising meetings in a formal way.
- How to use research and knowledge acquisition as a way to gather and filter information. This has been of great importance during our project, as it is the preferred and most instructive method for us to acquire previously unpossessed knowledge. The ability to extract and filter usable information from various sources is a key feature for any engineer.
- How to work in TEAM. Over the course of this thesis we have gained a lot of experience in working together as a group. The importance of having a common goal, being able to rely on each other, as well as complement each other, in order to succeed as a TEAM.
- How to approach, to plan and develop a system. We have gained experience in how to use engineering principles and procedures to plan and develop general engineering systems, and more specific control systems.

5.3.3 Technical experience

As control engineering students, the bachelor thesis has been a very valuable source of experience and knowledge regarding practical work and technical understanding. The group has been given the opportunity to familiarize itself with several highly relevant topics within control theory, as well as useful skills like C++ and ROS2. A list containing some highlights is found below:

- Since the main target of the thesis was to create a controller implemented in a larger ROS2 system, being able to understand and develop in ROS2 has been essential to the completion of the project. In addition, since C++ was the desired code language to use with ROS2, we also had to acquire skills regarding its programming syntax. Getting to know the functionalities of ROS2 and how to both get information from, and to send information to the ROS2 environment has been crucial to actually making the controller work with the real system. Furthermore, we have learnt the importance of modularity in systems to facilitate easy to use and comprehensible features. This has also enabled the group to easily make changes and extensions.
- How quaternions are very useful representations for orientation and rotation, due to its mathematical properties and the fact that it avoids Gimbal locks.
- How to implement position and attitude control for a 6DOF underwater vehicle. We have familiarized ourselves with a dynamic non-linear PD controller with 6DOF, derived by (Fjellstad and Fossen 1994), and gained experience in how to implement it in ROS2.
- How to develop a dynamic simulated system for an underwater vehicle, based on the dynamic equation of motion, derived by Fjellstad and Fossen, which describes the total dynamic model of an underwater vehicle.

Chapter 6

Conclusion

6.1 Summary of conclusions and recommendations

We have produced a functional dynamic positioning controller developed as a ROS2 package, compatible with any underwater vehicle with 6DOF. This packages is able to be implemented in any existing or new ROS2 network. However, the dynamic positioning controller requires state estimation in regards to position and attitude in order to function optimally. If no such estimates are available, one may use our package for teleoperations regardless.

Even though the controller is generally compatible with any system, drivers are required to convert controller output to thruster actuation. Implementing our system on an AUV also requires external logic to calculate trajectories and response to environmental changes.

Throughout the bachelor project our group has gained valuable experience and knowledge in regards to ROS2, teamwork, research, programming and much more.

6.2 Further work

As the duration of our bachelor thesis has only been for a limited time period, we have had to prioritize work in accordance with its utility and relevance. Our decision from the start was to prioritise a working control system, and thereafter implement additional features if our time frame allowed it. Based on the chapter regarding discussion (5), we propose these features as future work that may expand the functionality of the product we have created:

- Tune control parameters for position control.
- Implement standard operations to further increase usability.

- Perform stability analyzes of the controller to see if implementing an integrator would be suitable.

Bibliography

AC-ROV 100 (2022). en-US. URL: https://rts.as/product/ac-rov-100/ (visited on 05/07/2022).

ArduSub Homepage (2022). URL: https://www.ardusub.com/ (visited on 05/12/2022).

Autonomous Underwater Vehicle, HUGIN (2022). no. URL: https://www.kongsberg.com/no/maritime/products/marine-robotics/autonomous-underwater-vehicles/AUV-hugin/ (visited on 05/13/2022).

BlueROV2 (2022). en-US. URL: https://bluerobotics.com/store/rov/bluerov2/ (visited on 05/07/2022).

Bridge communication between ROS 1 and ROS 2 (May 2022). original-date: 2015-06-17T18:36:47Z. URL: https://github.com/ros2/ros1_bridge (visited on 05/17/2022).

Capocci, Romano et al. (Mar. 2017). "Inspection-Class Remotely Operated Vehicles—A Review". en. In: *Journal of Marine Science and Engineering* 5.1. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 13. ISSN: 2077-1312. DOI: 10.3390/jmse5010013. URL: https://www.mdpi.com/2077-1312/5/1/13 (visited on 05/07/2022).

Difference between Python and C++ (Mar. 2020). en-us. Section: C++. URL: https://www.geeksforgeeks.org/difference-between-python-and-c/ (visited on 05/09/2022).

Fjellstad and Fossen (Aug. 1994). "Quaternion feedback regulation of underwater vehicles". In: *1994 Proceedings of IEEE International Conference on Control and Applications*, 857–862 vol.2. DOI: 10.1109/CCA.1994.381209.

Fossen, Thor I. (2021). *Handbook of marine craft hydrodynamics and motion control =: Vademecum de navium motu contra aquas et de motu gubernando*. Second edition. Hoboken, NJ: Wiley. ISBN: 978-1-119-57505-4.

quaternion | mathematics | Britannica (2022). en. URL: https://www.britannica.com/science/quaternion (visited on 05/03/2022).

Quaternions and spatial rotation (May 2022). en. Page Version ID: 1085786949. URL: https://en.wikipedia.org/w/index.php?title=Quaternions_and_spatial_rotation&oldid=1085786949 (visited on 05/03/2022).

ROS/Introduction - ROS Wiki (2022). URL: http://wiki.ros.org/ROS/Introduction (visited on 04/28/2022).

Ship motion conventions (2022). URL: https://support.sbg-systems.com/sc/kb/latest/underlying-maths-conventions/ship-motion-conventions (visited on 05/09/2022).

US Department of Commerce, National Oceanic and Atmospheric Administration (2022). *What is the difference between an AUV and a ROV?* EN-US. URL: https://oceanservice.noaa.gov/facts/auv-rov.html (visited on 05/05/2022).

Varagnolo, Damiano (2020). *Autonomous Underwater Fleets*. eng. (Visited on 05/13/2022).

– (May 2022). *Remote control of an underwater vehicle*. eng/no.

Vortex NTNU (2022). en. URL: https://www.vortexntnu.no (visited on 05/17/2022).

Wu, Chu-Jou (2022). *6-DoF Modelling and Control of a Remotely Operated Vehicle*. en. Electronic Thesis or Dissertation. Publisher: Flinders University. College of Science and Engineering. URL: https://theses.flinders.edu.au/view/27aa0064-9de2-441c-8a17-655405d5fc2e/1 (visited on 03/31/2022).

Xu, Yang (2021). "Modelling of a Remotely Operated Vehicle and Tuning for Its Robust and Optimal Dynamic Positioning Control". eng. In: Accepted: 2021-09-29T16:28:36Z Publisher: uis. URL: https://uis.brage.unit.no/uis-xmlui/handle/11250/2786261 (visited on 04/28/2022).

Appendix A

Tables and matrices

$$\mathbf{R(q)} = \begin{bmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2(\epsilon_1\epsilon_2 - \eta\epsilon_3) & 2(\epsilon_1\epsilon_3 + \eta\epsilon_2) \\ 2(\epsilon_1\epsilon_2 + \eta\epsilon_3) & \eta^2 + \epsilon_2^2 - \epsilon_1^2 - \epsilon_3^2 & 2(\epsilon_2\epsilon_3 - \eta\epsilon_1) \\ 2(\epsilon_1\epsilon_3 - \eta\epsilon_2) & 2(\epsilon_2\epsilon_3 + \eta\epsilon_1) & \eta^2 + \epsilon_3^2 - \epsilon_1^2 - \epsilon_2^2 \end{bmatrix} \tag{A.1}$$

$\boldsymbol{R(q)}$ is the rotation matrix from $\boldsymbol{\{i\}}$ to $\boldsymbol{\{b\}}$ (ibid.).

$$\boldsymbol{U(q)} = \begin{bmatrix} \boldsymbol{-\epsilon^T} \\ \boldsymbol{\eta I_{3x3} + S(\epsilon)} \end{bmatrix} \tag{A.2}$$

$\boldsymbol{U(q)}$ is the coordinate transformation matrix (ibid.).

$$\boldsymbol{S(a)} \triangleq \begin{bmatrix} 0 & -a_3 & +a_2 \\ +a_3 & 0 & -a_1 \\ -a_2 & +a_1 & 0 \end{bmatrix} \tag{A.3}$$

$\boldsymbol{S}$ is the skew-symmetric matrix operator used for calculating vector cross products.

Parameter	Value
m	11.5 (kg)
W	112.8 (N)
B	114.8 (N)
$\boldsymbol{r_b}$	$[0, 0, 0]^T$ (m)
$\boldsymbol{r_g}$	$[0, 0, 0.2]^T$ (m)
I_x	0.16 (kg m^2)
I_y	0.16 (kg m^2)
I_z	0.16 (kg m^2)

Table A.1: A Priori information for parameters in rigid body dynamics and restoring forces (Wu 2022, p. 48)

DoF	Added Mass	Value
Surge	$X_{\dot{u}}$	-5.5 (kg)
Sway	$Y_{\dot{v}}$	-12.7 (kg)
Heave	$Z_{\dot{w}}$	-14.57 (kg)
Roll	$K_{\dot{p}}$	-0.12 (kg m^2/rad)
Pitch	$M_{\dot{q}}$	-0.12 (kg m^2/rad)
Yaw	$N_{\dot{r}}$	-0.12 (kg m^2/rad)

Table A.2: Determined added mass parameters (Wu 2022, p. 48)

DoF	Linear Damping	Value	Quadratic Dampening	Value
Surge	X_u	-4.03 (Ns/m)	$X_{u\|u\|}$	-18.18 (Ns^2/m^2)
Sway	Y_v	-6.22 (Ns/m)	$Y_{v\|v\|}$	-21.66 (Ns^2/m^2)
Heave	Z_w	-5.18 (Ns/m)	$Z_{w\|w\|}$	-36.99 (Ns^2/m^2)
Roll	K_p	-0.07 (Ns/rad)	$K_{p\|p\|}$	-1.55 (Ns^2/rad^2)
Pitch	M_q	-0.07 (Ns/rad)	$M_{q\|q\|}$	-1.55 (Ns^2/rad^2)
Yaw	N_r	-0.07 (Ns/rad)	$N_{r\|r\|}$	-1.55 (Ns^2/rad^2)

Table A.3: Determined linear and quadratic damping parameters (Wu 2022, p. 48)

Implementation of a quaternion-based PD controller in ROS2 for a generic underwater vehicle with six degrees of freedom

Background

Commercially available AUVs are today limited to only individual work and have no abilities to conduct cooperative tasks. ITK is therefore conducting a research project with the objective to develop knowledge that enables fleets of AUVs to operate collectively, adaptively, and in a leaderless fashion.

As a part of the research project, a modified BlueROV2 Heavy is used as a testbed. The current ROVs control system is lacking several features, making it hard to operate. To improve the usability and make it easier to handle, there has been a desire by ITK to develop a new control system in ROS2.

Task

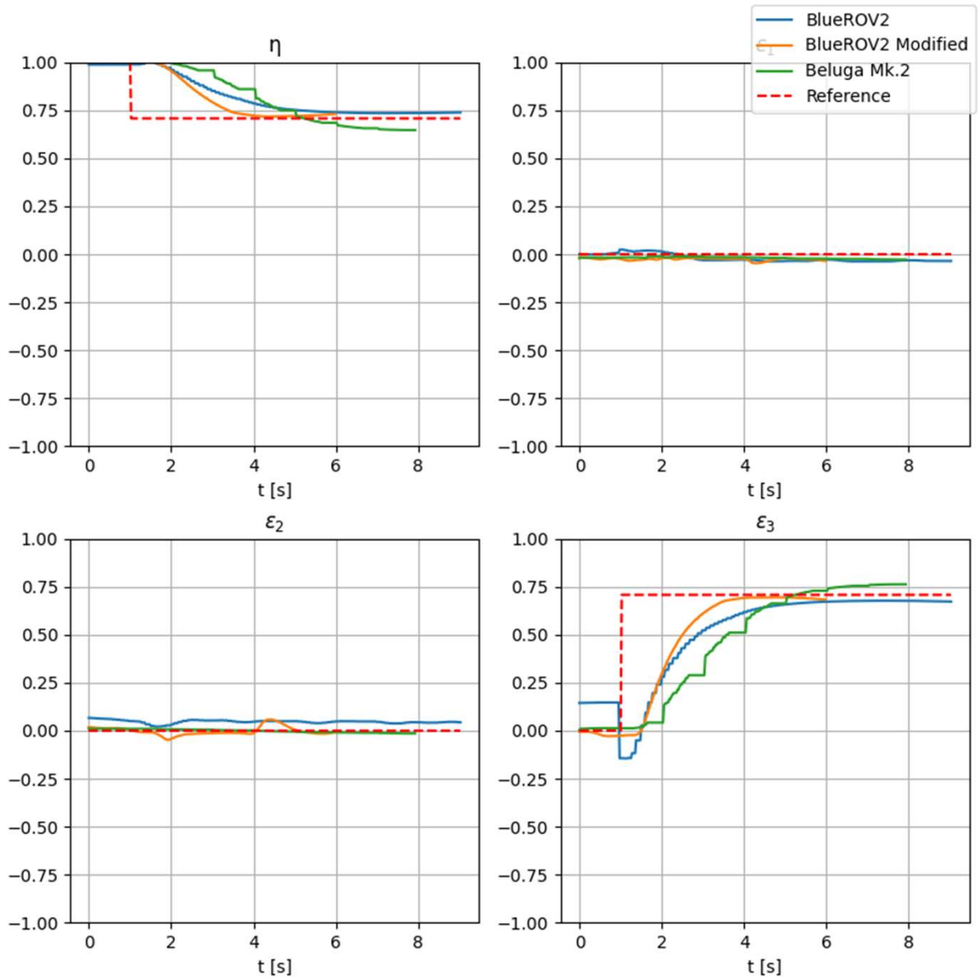
Implement a control scheme for a BlueROV2 Heavy in ROS2, with teleoperation functionality.

Method

By utilizing the control law derived by Fjellstad and Fossen[1] and implementing it in a ROS2 node, a dynamic positioning control system has been developed.

Result

Multiple response tests have been conducted on a total of three different UUVs. The plots displayed below show the step-responses of all the UUVs with a 90° yaw movement, represented in unit quaternions.

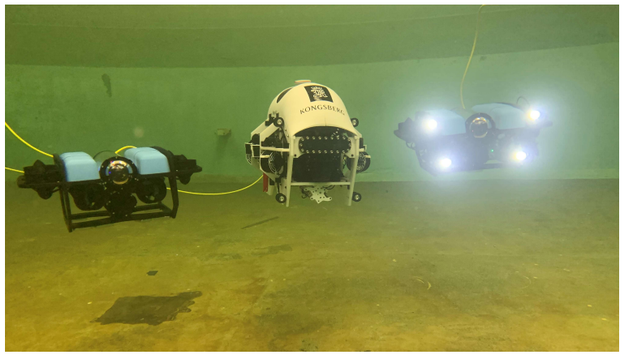


Step responses on three different UUVs

Conclusion

A dynamic positional controller has been produced and implemented in ROS2. Taking advantage of the standard messages in ROS2, the controller can be configured to fit both AUVs and ROVs. A state estimate for position and attitude is required for optimal control. Without such an estimate, the controller can be used for teleoperations alone.

The UUVs used in our thesis

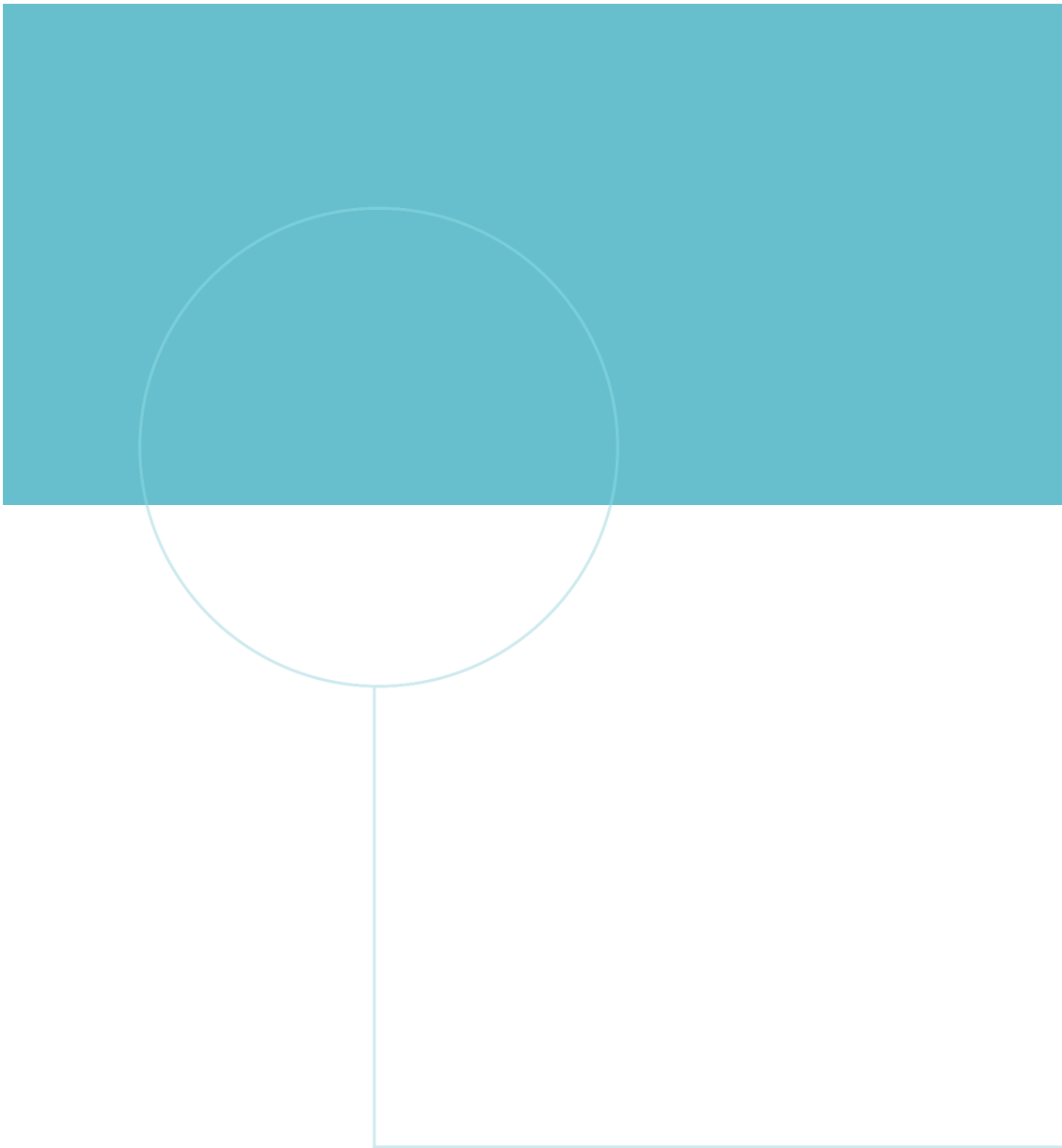


From left to right: BlueROV2 Heavy, Vortex Beluga Mk.2 and BlueROV2 Heavy modified at ITK

1. Fjellstad and Fossen (Aug. 1994). "Quaternion feedback regulation of underwater vehicles". In: 1994 Proceedings of IEEE International Conference on Control and Applications, 857–862 vol.2. Available at: https://ieeexplore.ieee.org/document/381209

Department of Engineering Cybernetics, ITK
Erik Rowe, Simen Hustad, Petter Øvereng Juliebø, Elias Olsen Almenningen

NTNU | Norwegian University of Science and Technology



NTNU

Norwegian University of
Science and Technology