Marine and Maritime Intelligent Robotics Program

Marine Mechatronics

Submitted by

Group 02

Name	ID
Md Ether Deowan	22303698
Md Shamin Yeasher Yousha	22303716
Shahriar Hassan	22303704
Tihan Mahmud Hossain	22303703
Akshat Sinha	22303714
Krittapat Onthuam	22303709

Submitted to

Vincent Hugel
Professor, University of Toulon
Vincent Creuze
Professor, University of Montpellier

March 28, 2024

Contents

List of Figures

Implementation of a Proportional controller on the BlueROV

1 Introduction

In this lab project, we dive into the challenge of controlling an underwater vehicle, specifically the BlueROV, using a proportional controller. The simplicity and effectiveness of PID controllers make them a go-to choice for managing such complex systems, but getting them just right involves a few crucial steps. We start with the basics—linking thrust force to PWM signals, a fundamental aspect of vehicle control. We validate our thruster models and adjustments for the ROV's buoyancy, ensuring our controller can accurately compensate for these forces.

Then, we introduce a proportional controller focused on depth, refining it with buoyancy adjustments and then expanding our control strategy with a PI controller to see how well it handles disturbances. The project doesn't stop at depth control; we also estimate vehicle heave from depth measurements and explore the addition of floatability compensation to a PID controller for even finer control.

Our goal is to enhance the BlueROV's handling and deepen our understanding of underwater vehicle control, making this work a valuable step forward in marine robotics.

1.1 BlueROV2 Robot

BlueROV2 is a remotely operated underwater vehicle manufactured by Blue Robotics, a specialized company in ocean technology. Blue Robotics have introduced a series of BlueROV that can perform various underwater task, for example, observation, research, exploration, inspection, and other specialized operation. These vehicles are widely used for their affordability, accessibility, and modularity.

In our lab, we have used BlueROV2 (6-thruster vectored configuration) for performing the given task. It has 5 degree of freedom. It has an open-source electronics and software facility.

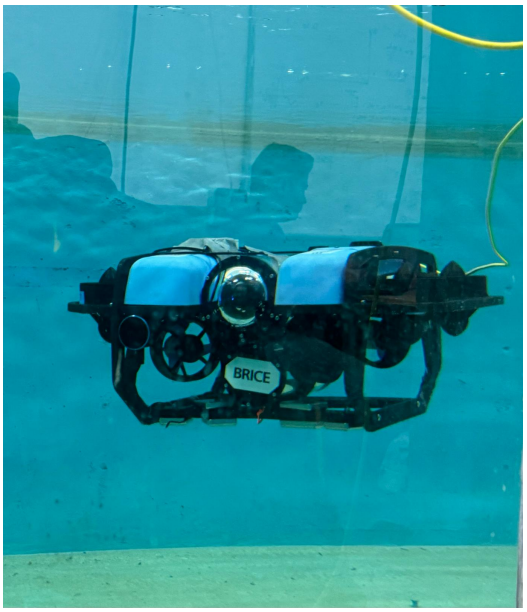


Figure 2: BlueROV2

1.2 PID Controller

A PID controller stands for Proportional-Integral-Derivative controller. It has a feedback control mechanism that enables us to regulate the output of a system based on the difference between the desired output and the actual output. It continuously calculates the error value and applies necessary corrections based on proportional, integral, and derivative terms.

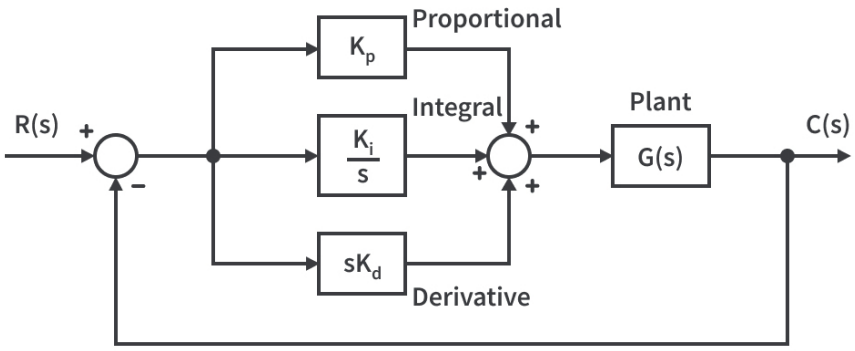
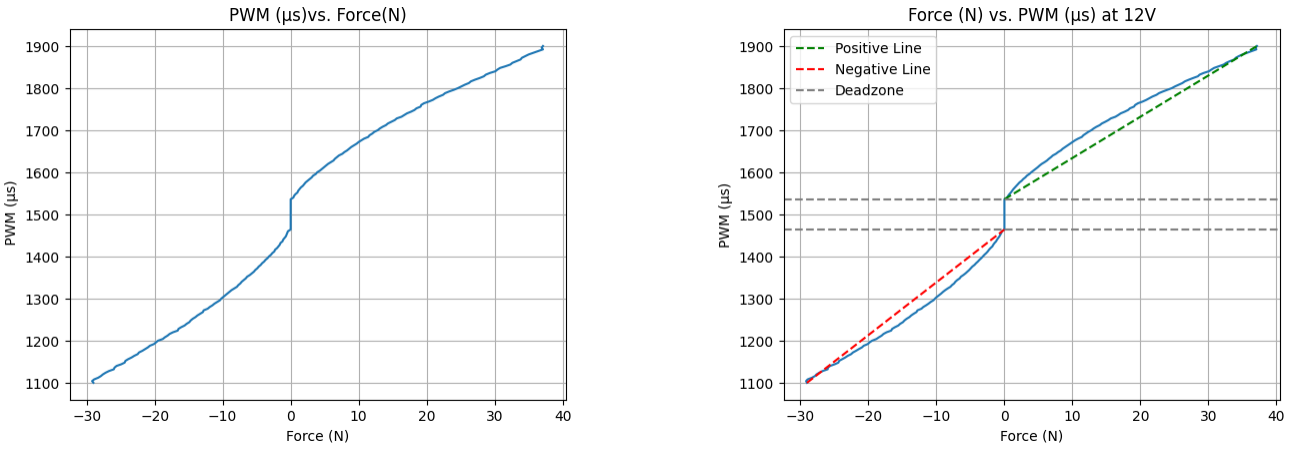


Figure 3: Block diagram of PID controller

In this experiment, we tried to implement a PID controller with our BlueROV. In general, PID controllers might be used for different purposes, for example, depth control, stabilization, heading control, etc. We tried to implement depth control with our BlueROV in this experiment.

2 Computing a Thruster's Control Input Depending on the Desired Thrust



(a) Force plot

(b) Linear plot

Figure 4: Linear Approximation of PWM-Thrust Relationship for Non-linear Curves

3 Experimental Validation of the Thruster's Model

It's essential to gauge its buoyancy level to implement any control commands on the robot. This involves determining the precise thrust needed from the thrusts to attain neutral buoyancy underwater, where the robot neither sinks nor floats but maintains a balanced position.

This involves calculating the floatability. To calculate that, we have followed the following steps:

- We have used a 1.5-litre bottle. We filled the bottle with water and put that on top of the robot without putting any additional pressure. We keep observing the robot until it reaches a depth where it stays buoyant.
- We mark the level of water bottle for the amount of water for which the buoyancy was achieved and use that metric to calculate the floatabilty.
- Floatability Estimation: We found the water bottle was 1.1 liters empty when the buoyancy was achieved. Because we add the empty bottle and try to do the task, we need to add the 1.1 and the 1.5 to find the total force. After that, we have to divide it by 4 to find the floatability. So, the calculation will be (1.1 + 1.5) / 4: 0.65 kg.f.

The floatability:

- With empty bottle mounted on the BlueROV: 0.65 kg.f
- Without empty bottle mounted on the BlueROV: 0.275 kg.f

4 Implementing a Proportional Controller for the Depth

In this particular phase of the control process, we will be implementing a proportional controller to manage and maintain the depth of our system. We are using a proportional controller here:

$$f_z = K_p \cdot (z_{\text{des}} - z)$$

Having implemented the proportional controller within the BlueROV's control code, the subsequent step was to test its performance through a step input test. The desired depth z_{des} was set to a constant value of 0.5 meters. The results of this test are displayed in the provided plots, which showcase the depth and thruster PWM (Pulse Width Modulation) values over time.

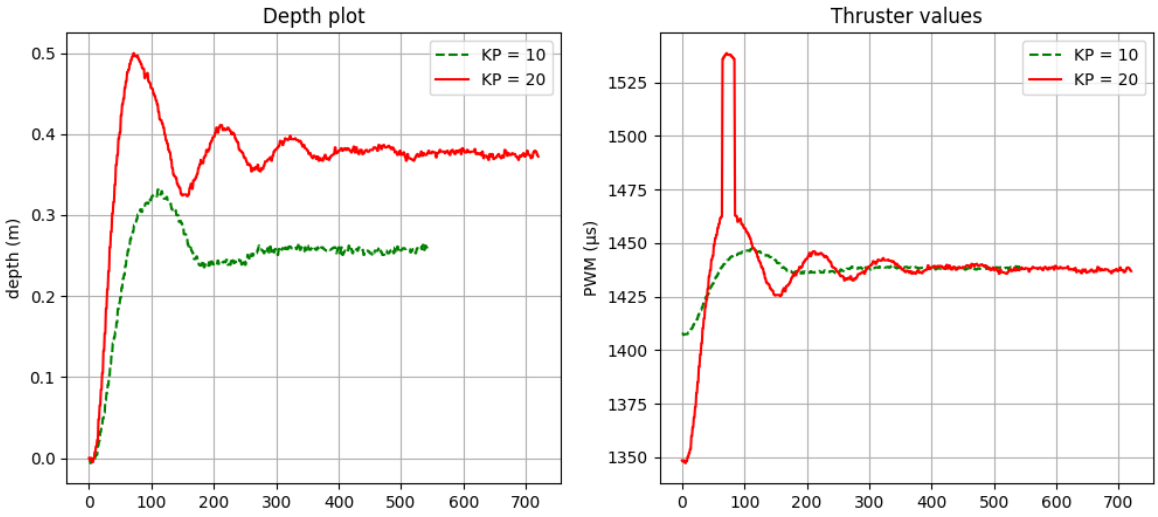


Figure 5: Proportional Controller for the Desire Depth

In the figure of Depth plot, we can see that with higher K_p (K_p=20) the oscillation is higher but it gives less error than the K_p with lesser value (K_p=10).

The Depth Plot reveals that upon initiation, the ROV's descent towards the target depth was rapid, before it reached an oscillatory state around the desired value. This oscillation indicates an overcompensation by the thrusters, a typical characteristic of a system governed by a proportional controller without damping effects from an integral or derivative component.

The Thruster PWM Plot provides insight into the control signals sent to the thrusters. It's observed that the PWM values for heave (which control depth) fluctuate as the system attempts to stabilize, reflecting the oscillations seen in the depth plot.

From these outcomes, it is apparent that while the proportional control brings the ROV close to the desired depth, there is a persistent oscillation and a residual error, a behavior well-aligned with the theoretical expectations of a P-controller. This experiment underscores the need for further tuning of the K_p value or the inclusion of integral and derivative controls to achieve a more stable and accurate depth maintenance.

5 Implementing a Depth Controller with Floatability Compensation

We incorporate floatability compensation into the BlueROV's depth control. Building on the proportional control framework from the previous steps, we introduced a constant floatability term to address the steady-state error observed in earlier tests. The modified control equation became $\tau_z = K_p(z_{des} - z) + floatability$, with the floatability term intended to counteract the buoyant force of the ROV.

The implementation of this adjusted controller into the BlueROV's control software was followed by another step input test at a constant desired depth of 0.5 meters. The aim was to evaluate the effect of the floatability compensation on the system's performance and to adjust the proportional gain K_p accordingly, which was expected to be lower than in the initial P-controller setup.

Examining the results, depicted in the updated depth and thruster PWM plots, we see a marked improvement. The depth plot displays a quicker stabilization around the desired depth, suggesting that the floatability compensation is effectively countering the buoyant forces. Consequently, the oscillations are reduced, though not entirely eliminated, indicating a more accurate steady-state than the simple P-controller implementation.

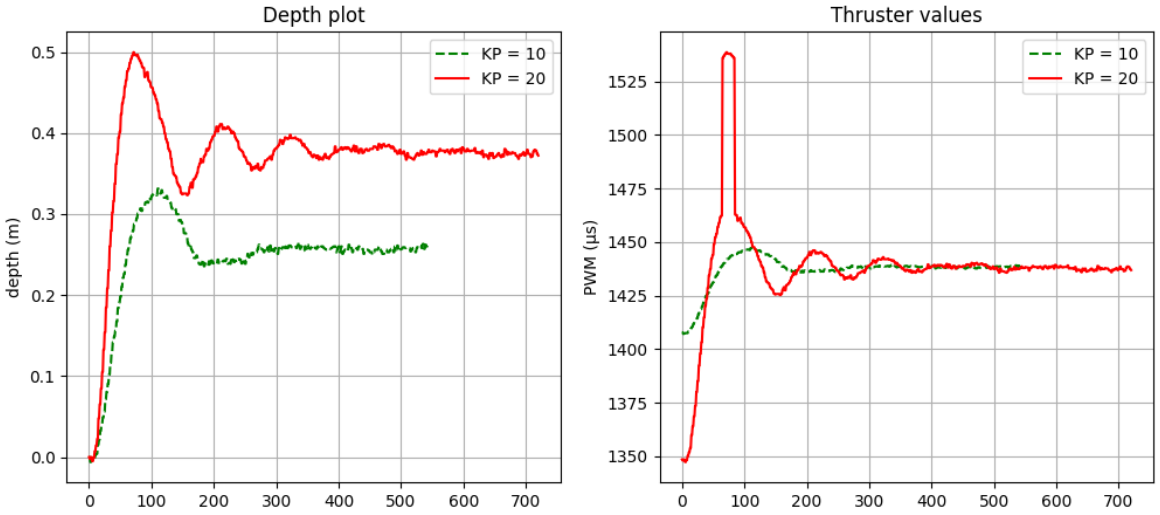


Figure 6: Proportional Controller for Floatability Compensation

In the figure of Depth plot, we can see similar result like the previous graph that with higher K_p (K_p=20) the oscillation is higher but it gives less error than the K_p with lesser value (K_p=10). The only difference that with floatability we get less error.

The thruster PWM plot corroborates this finding, showing that the thrusters reach a steady command signal more swiftly, with less fluctuation than before. This stability implies that the system is closer to equilibrium, thanks to the floatability compensation.

These experiments underscore the efficacy of including floatability in the control schema for the BlueROV, allowing for finer control over its depth and a more steady hover underwater. This addition is a significant step towards more sophisticated control algorithms that could further minimize error and enhance the ROV's navigational capabilities.

6 Creating a Trajectory Compatible with the ROV's Dynamics

The utilization of a step input as the desired trajectory causes the trajectory of the robot to exhibit overshoots or oscillations. To overcome this issue and optimize compatibility with the ROV, we have replaced the step input with a simple cubic polynomial for trajectory generation. The pseudocode given bellow

Algorithm 1 Calculate Desired Trajectories

import numpy as np
import matplotlib.pyplot as plt
Given:
$t_{\text{final}} = 20$ ▷ seconds
$z_{\text{init}} = 0$ ▷ meters
$z_{\text{final}} = 0.5$ ▷ meters
Initialize:
$t \leftarrow \text{np.linspace}(0, 25, \text{num} = 500)$ ▷ from 0 to 25 seconds for a smooth curve
$a_2 \leftarrow 3 \times \frac{(z_{\text{final}} - z_{\text{init}})}{(t_{\text{final}}^2)}$
$a_3 \leftarrow -2 \times \frac{(z_{\text{final}} - z_{\text{init}})}{(t_{\text{final}}^3)}$
Calculate:
$z_{\text{desired}}(t) \leftarrow \begin{cases} z_{\text{init}} + a_2 t^2 + a_3 t^3, & \text{if } t < t_{\text{final}} \\ z_{\text{final}}, & \text{otherwise} \end{cases}$
$z_{\text{dot\_desired}}(t) \leftarrow \begin{cases} 2a_2 t + 3a_3 t^2, & \text{if } t < t_{\text{final}} \\ 0, & \text{otherwise} \end{cases}$

The plot shows the desired depth trajectory, $z_{desired}(t)$, and its rate of change over time, $\dot{z}_{desired}(t)$, for the ROV according to the cubic trajectory generation algorithm given. The trajectory smoothly changes from the initial depth z_{init} of 0 meters to the final depth z_{final} of 0.5 meters over a span of 20 seconds without overshooting. After reaching the final depth at t_{final}, the depth remains constant and the rate of change drops to zero, indicating that the ROV has stopped descending.

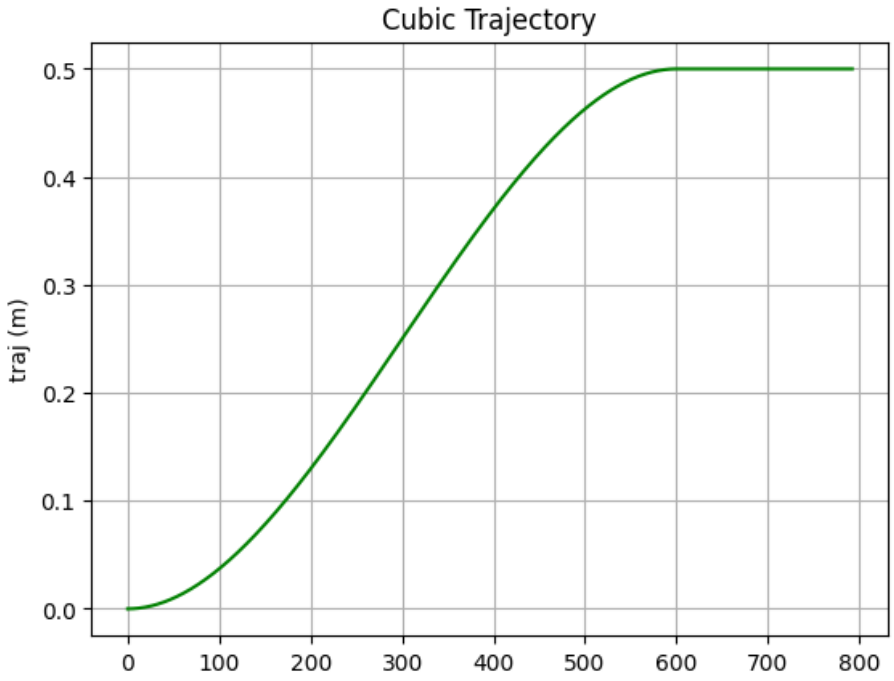


Figure 7: Cubic Trajectory for Desire Depth

7 Implementing the Polynomial Trajectory

The integration of the polynomial trajectory into our depth control strategy marked a significant advancement in the BlueROV's handling. Adhering to the practical tip, we initiated the trajectory at the BlueROV's actual surface-floating depth to avoid the abrupt activation of the thrusters. This refined approach incorporated the proportional (P) controller with gravity/buoyancy compensation that we had previously designed.

During the testing phase, we executed the controller with the polynomial trajectory, aiming to achieve a smooth descent and ascent along the predetermined path. The K_p value was adjusted to harmonize with the new trajectory, prioritizing a balance between responsiveness and overshoot minimization.

The experimental data, as seen in the Depth Plot, illustrates the ROV's movement towards the target depth following the smooth polynomial curve. Despite the improved trajectory, minor oscillations around the target depth were observed, indicating that while the trajectory was smoother, the P controller still introduced some degree of oscillation.

The Thruster PWM Plot demonstrates a more dynamic response than previous tests, with the thrusters adjusting their output to follow the polynomial trajectory. This dynamic adjustment is particularly noticeable in the heave component, which regulates the ROV's depth.

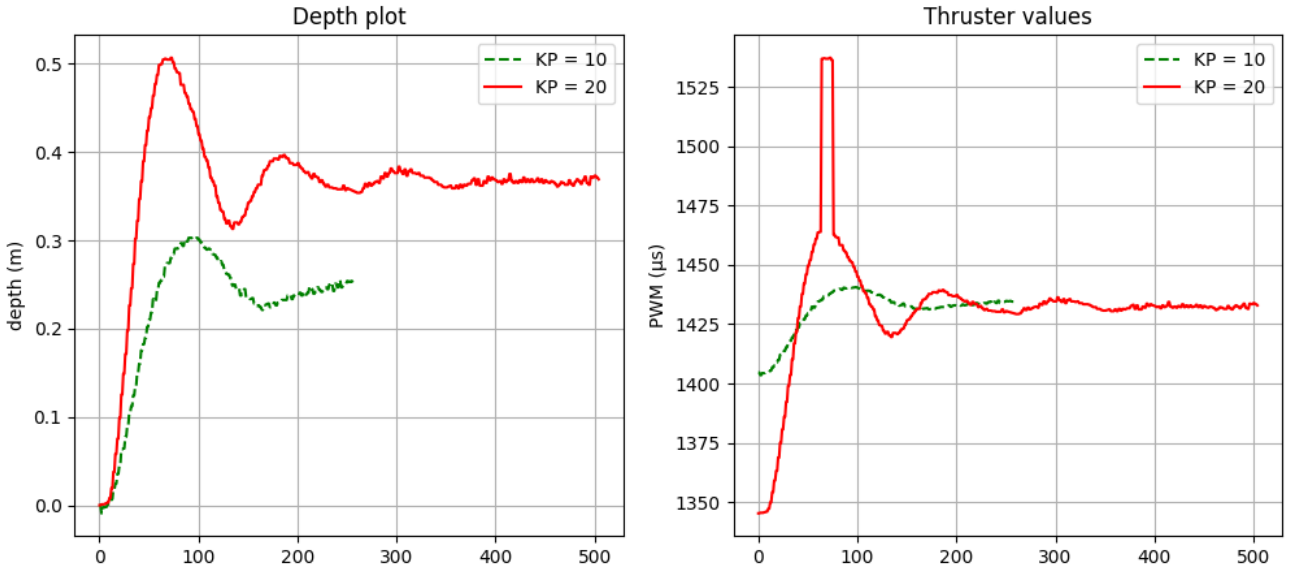


Figure 8: Proportional Controller with Trajectory Generation

This experiment with the polynomial trajectory underscores the potential for sophisticated depth control in underwater vehicles. It also highlights the necessity of meticulous K_p tuning when introducing new control variables and the possibility of further enhancements, perhaps through the integration of integral and derivative controls, to perfect the BlueROV's navigation along a desired path.

8 Proportional Integral Controller (PI) for the Depth

For addressing the steady-state error persistent in previous iterations, we have implemented a Proportional Integral (PI) controller for BlueROV's depth control system. This controller promised an

augmented approach by not only considering the current depth error but also accumulating this error over time to make necessary adjustments. We also considered the floatability while calculating the values.

$$\tau_z = K_p \tilde{z} + K_i \int_0^t \tilde{z}(t)\, dt + \text{floatability}$$

$$\text{where, } \tilde{z} = (z_{\text{destination}} - z_{\text{actual}})$$

During the trials, initial tuning of K_p was done with a softer approach than previously used with the simple P controller, followed by a cautious increment of K_i to gently eliminate the residual error without introducing oscillations.

We started with a small K_i because initially the error used to be very big. Since K_i integrates the error, it will make the system unstable if we use a large K_i. So we started with a very small value, K_i = 0.001. Then we gradually increased the value. In case when the error still didn't converge, we also tried decreasing the K_p value a little bit.

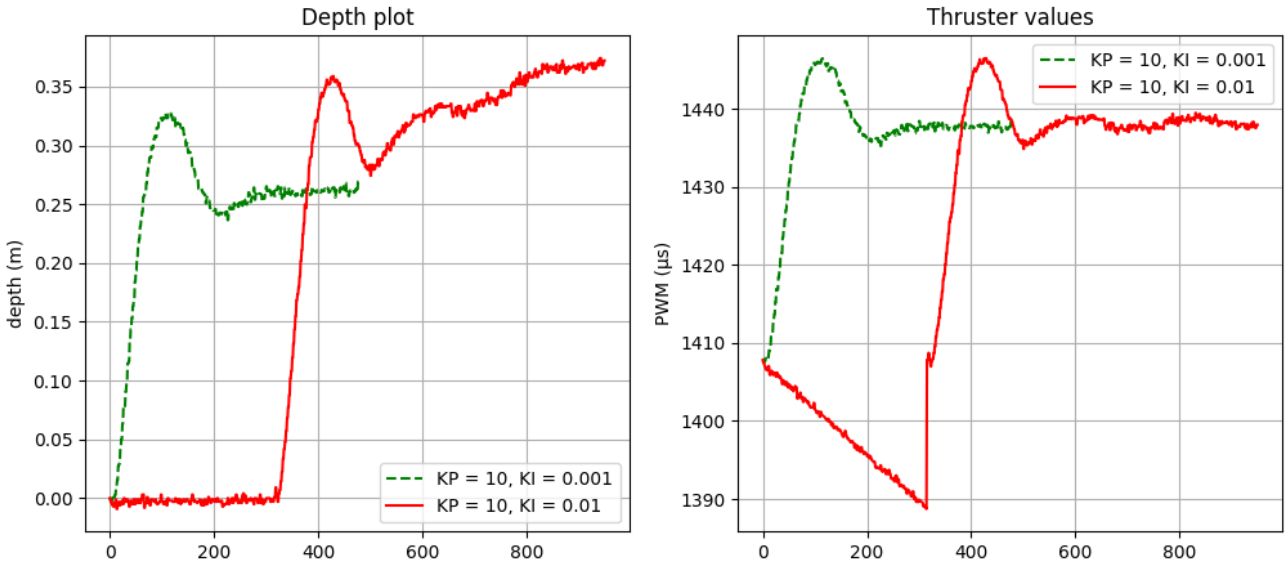


Figure 9: Proportional Integral (PI) Controller for depth and floatability compensation

Here we can see when K_i = 0.01, the Red curve on the depth plot, the error keeps getting integrated and it tries to catch up the desired value. While with less K_i value (K_i=0.001), the Green curve, it doesn't integrate the error that much and thus it has higher error than the previous error. (We had an initial error at Red curve of the 'Thruster values' graph)

However, the resulting Depth Plot shows a marked improvement in reducing the steady-state error, especially evident when the BlueROV reached the target depth. However, a small overshoot followed

by minor oscillations is visible, suggesting a need for further fine-tuning of the K_i gain to achieve a more damped response.

The Thruster PWM Plot also indicates a more refined operation, with the heave component exhibiting a smoother transition and less aggressive corrections compared to earlier tests. This demonstrates the PI controller's ability to adjust the thrust with a more nuanced understanding of the ROV's positional history, leading to a steadier maintenance of the desired depth.

This experiment confirms the value of adding an integral component to the control system, showcasing its potential to mitigate steady-state errors and enhance the control accuracy for the BlueROV's depth maintenance, ultimately bringing us closer to achieving precise and reliable underwater navigation.

9 Is the PI Robust Toward External Disturbances?

9.1 Testing the Robustness of the PI Controller Against Constant External Disturbances [Constant]

For this part, we will test the robustness using an empty bottle and mounting that on the top of the BlueROV.

For fairness, we have kept the values the same as the previous ones and tested the BlueROV the same way. After plotting the desired depth trajectory below:

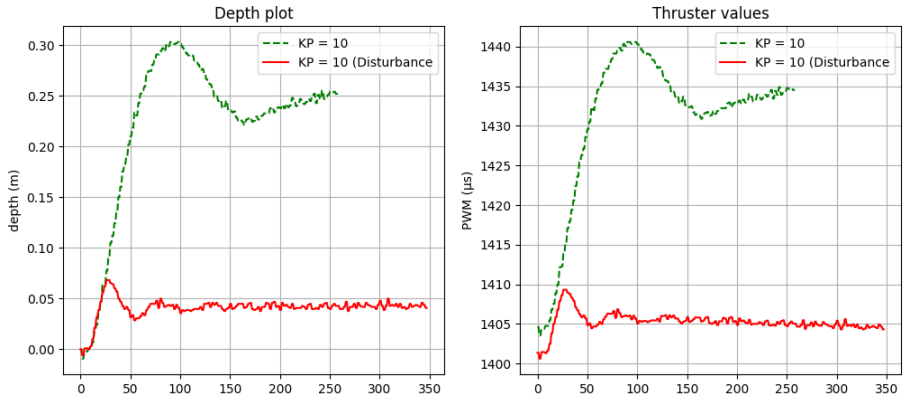


Figure 10: Robustness of the PI Controller Against Constant External Disturbances

From the result, we can see that, in general, the desired depth trajectory is as expected. On the other hand, the depth trajectory from the constant disturbance will only delay the expected steady state. But the achieved result will be the same.

9.2 Testing the Robustness of the PI Controller Against Dynamic External Disturbances [Dynamic]

For evaluating the robustness of our PI controller, we introduced an external disturbance to the system—a 1.5-liter plastic bottle attached to the top of the BlueROV—to simulate an unexpected change

in buoyancy. The gains previously optimized were left unchanged to test the controller's ability to cope with this new variable.

Upon attaching the bottle and initiating the depth control sequence, we observed the BlueROV's response and captured the depth trajectory alongside the thruster values. The resulting data is plotted in the figure, showing the BlueROV's ability to reach and maintain the desired depth, despite the disturbance.

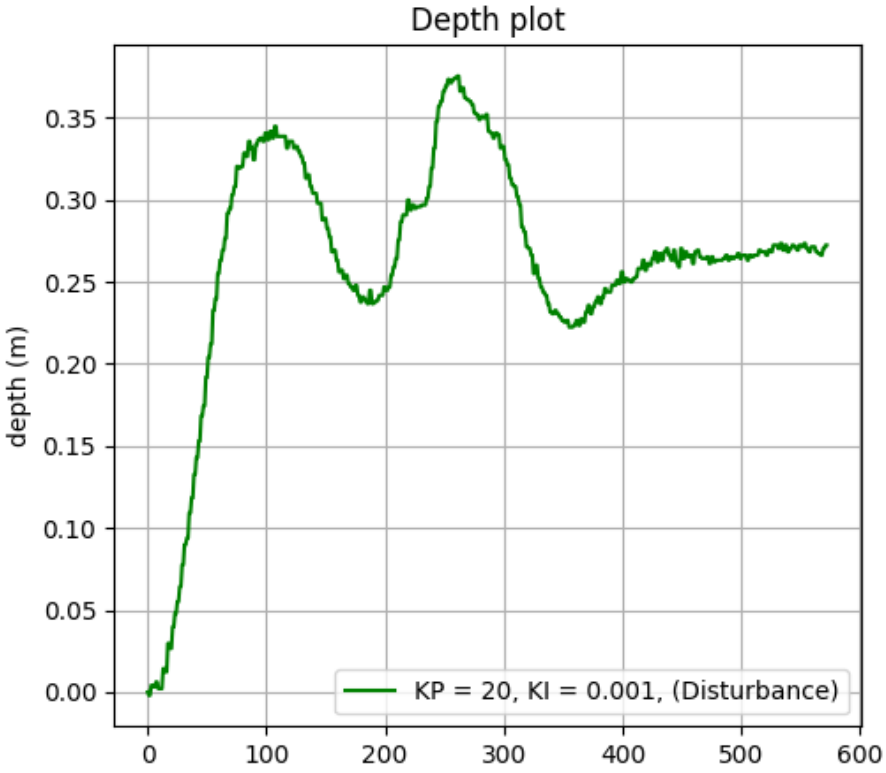


Figure 11: Robustness of the PI Controller Against Variable External Disturbances

The comparison of the new depth trajectory against the nominal test without disturbance revealed the PI controller's response characteristics. As anticipated, the depth trajectory exhibited a deviation from the nominal curve immediately after the disturbance was introduced. However, the BlueROV did eventually stabilize at the target depth, indicating that while the PI controller is not entirely immune to external disturbances, it can recover and maintain control.

This experiment underscores the importance of robust control strategies in unpredictable marine environments. Although the PI controller displayed a certain level of resilience, the observed deviation confirms the need for even more sophisticated control systems—such as adaptive or robust controllers—to handle disturbances more effectively, maintaining performance without the need for retuning.

10 Estimating the Heave from the Depth Measurements with an Alpha-Beta Filter

For estimating the vertical speed (heave) of our BlueROV without a dedicated sensor, we implemented the alpha-beta filter, an efficient computational method to filter noisy data and estimate the rate of

change. In accordance with the provided guidelines, the filter's parameters were set to $\alpha = 0.1$ and $\beta = 0.005$, with adjustments to be made as needed, and the implementation was crafted around the ROV's depth sensor data, sampled at 58 Hz.

Algorithm 2 Estimating heave from depth measurements with an alpha-beta filter

procedure ESTIMATEHEAVE(depth, prev_depth=None, prev_heave=None)
    sample_time ← global variable
    $\alpha \leftarrow 0.45$
    $\beta \leftarrow 0.1$
    if prev_depth is None **then**
        heave ← 0
    else
        heave ← (depth − prev_depth)/sample_time
    end if
    if prev_heave is None **then**
        filtered_depth ← depth
        filtered_heave ← heave
    else
        filtered_depth ← prev_depth + sample_time × prev_heave + 0.5 × β × sample_time^2 × (heave + prev_heave)
        filtered_heave ← prev_heave + α × sample_time × (heave + prev_heave)
    end if
    return filtered_heave
end procedure

Upon deploying the filter during a test dive, we carefully monitored the output to assess its performance. The results were promising, as the alpha-beta filter effectively smoothened the depth signal and provided a heave estimate that closely followed the BlueROV's movements, despite the inherent noise in the raw depth data.

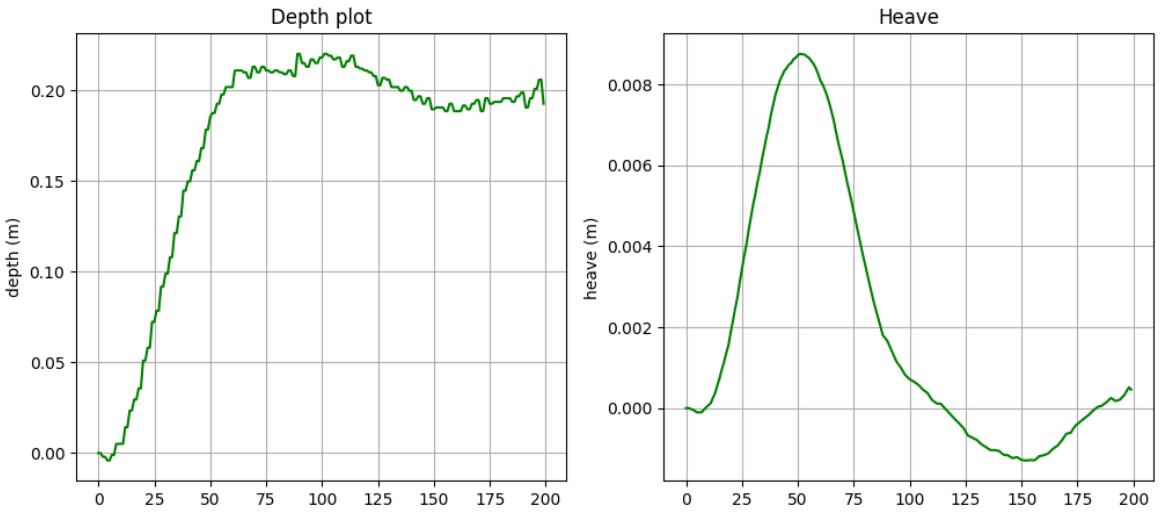


Figure 12: Estimation of ROV Heave Using an Alpha-Beta Filter

In the above plot we can clearly see the result of the velocity according to our desired depth trajectory. It can be observed that around the 50 seconds time the desired trajectory reaches almost the

steady state. And around this time, the velocity reaches its peak point and starts to go decrease. This dynamics between the velocity and the desired depth trajectory ensures a smooth steady state for the BlueROV.

The filter's performance validated our implementation approach, confirming the suitability of the chosen α and β values for our sensor setup. This successful application of the alpha-beta filter stands as a testament to the versatility of such algorithms in enhancing sensor data usability, particularly in the field of marine robotics where direct measurements are not always feasible.

11 Implementing a PID with Floatability Compensation

Taking our BlueROV control to the next level, we implemented a full PID controller, integrating the heave rate derived from our alpha-beta filter to account for the vertical speed of the ROV. The PID formula,

$$\tau_{PID} = K_p z + K_i \int z(t)dt + K_d \dot{z} + floatability$$

We used this formula to get the desired depth trajectory from Step 5 was followed, and floatability compensation was included for robust buoyancy management.

The addition of the derivative term aimed to anticipate the ROV's behavior, providing a corrective action preemptively to reduce overshooting and improving the response time. We conducted tests to compare the performance of the newly implemented PID controller with the previous PI controller under the same conditions, including a sudden manual disturbance applied to the ROV to simulate an external force.

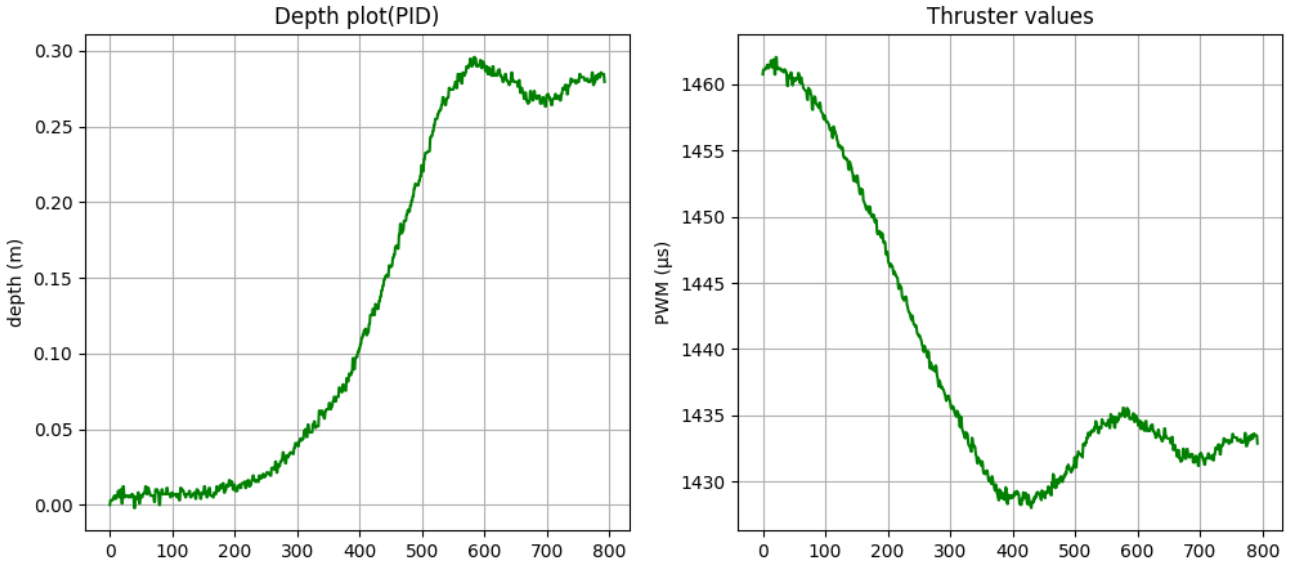


Figure 13: PID Controllers with Floatability Compensation

The comparison revealed the PID controller's enhanced ability to counteract the effects of the disturbance, swiftly returning to the desired depth trajectory with minimal overshoot, compared to the PI controller. While the PI controller effectively reduced the steady-state error, the PID controller

demonstrated superior handling of sudden changes, thanks to the predictive power of the derivative term.

Heave Analysis: In our robot, there is no sensor able to measure the vertical speed of the ROV. We also analyzed the heave motion using the derivative of the position of the BlueROV along z-axis. Since, in real life the ROV doesn't stay exactly at the same place all the time, rather it goes up and down for noise. So if we calculate the Heave it will go up and down showing noise. That's why we have used "Alpha-Beta filter" for smoothing the curves. It is a kind of Kalman filter that smoothen out the derivatives taking the minimum value and removing the noise.

12 Conclusion

Throughout this comprehensive exploration of underwater vehicle control via the BlueROV, we have systematically navigated through the implementation of various controllers, each building upon the insights gained from the last. From initial proportional control to sophisticated PID algorithms, we've fine-tuned the ROV's response to depth changes, compensated for buoyancy, and addressed the challenges of yaw control. The practical tests have demonstrated the relative strengths and weaknesses of each approach, highlighting the necessity for integral and derivative components in managing steady-state errors and disturbances. Our foray into trajectory generation has further revealed the complexity and necessity of adaptive control strategies in marine robotics. Notably, the implementation of an alpha-beta filter has shown the power of estimation in systems where direct measurement is not feasible, a common scenario in underwater environments. By experimenting with the robustness of these control systems against external disturbances, we have laid the groundwork for future research, where more adaptive and resilient control mechanisms can be explored. The journey of the BlueROV, from simple P to advanced PID control, underscores the enduring pursuit of precision and stability in the dynamic and often unpredictable underwater domain.