

Article

Model Predictive Control of Underwater Tethered Payload

Mark O'Connor ^{*}, Andy Simoneau [†] and Rickey Dubay [†]

Mechanical Engineering Department, University of New Brunswick, Fredericton, NB E3B 5A3, Canada; simoneau@unb.ca (A.S.); dubayr@unb.ca (R.D.)

* Correspondence: moconno3@unb.ca

[†] These authors contributed equally to this work.

Abstract

A fully automated, buoy-based deployment sensor system is being developed to acquire high-quality water column data, and requires a controller to accurately position an array of sensors at various depths. The sensor system will be potentially deployed under rough ocean conditions. Depth is measured by a pressure sensor and adjusted through a rotating drum powered by a stepper motor. The proposed controller uses a model predictive control algorithm, a type of optimal control that predicts system response to optimize control actions used to track a desired variable-depth, setpoint profile. The profile is calculated to ensure smooth motion of the system, preventing motor malfunction. A simplified system model was created and used to simulate an open-loop test and system response. Constraints were applied to the control actions to match the practical limitations of the stepper motor. The simulated results show successful tracking of both a shallow and deep profile. At this stage of testing, the effects of ocean currents are considered by using a simple disturbance that provides the effect of ocean currents. A practical prototype that can implement the model predictive controller was tested on the physical buoy-based system with good control performance.

Keywords: model predictive control; precision motion control; aquaculture



Academic Editors: Antonio Valderrabano-Gonzalez and Carlos E. Castañeda

Received: 19 August 2025

Revised: 8 September 2025

Accepted: 10 September 2025

Published: 17 September 2025

Citation: O'Connor, M.; Simoneau, A.; Dubay, R. Model Predictive Control of Underwater Tethered Payload. *Appl. Sci.* **2025**, *15*, 10122. <https://doi.org/10.3390/app151810122>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As societies continues to grow, so too does their need for food production. Many coastal regions have been developing their aquaculture facilities to meet that demand. Aquaculture focuses on the production of fish, shellfish, and aquatic plants for the goal of consumption. There are many types of aquaculture farms, ranging from salmon cages in bays and oceans, to inland farms in ponds and lakes [1,2]. With the growth in aquaculture, there is also demand to introduce Industrial Internet of Things (IIoT) and Industry 4.0 practices to assist with more optimal and environmentally sustainable practices [3–5]. Key metrics for monitoring for most aquaculture sites are the pH level, temperature, turbidity, and level of dissolved oxygen [6,7]. Through the collection of this data, aquaculture sites could be digitally twinned to provide real-time and predicted metrics of the site [8,9]. These twins will need high quality water column data, which requires accurate depth positioning of sensors. Sending a manned vessel to collect this data can be costly and time-intensive, so a buoy-based tethered sensor unit is being developed to gather this data. This work presents an optimal control algorithm to accurately position a sensor underwater, in calm water conditions and in a cross current. The ability to position sensors be used as a platform to acquire the water column data. The algorithm will be tested in simulation and through practical testing.

Figures 1 and 2 are simple diagrams highlighting the mechanical and electrical aspects of a proposed system that can use this controller.

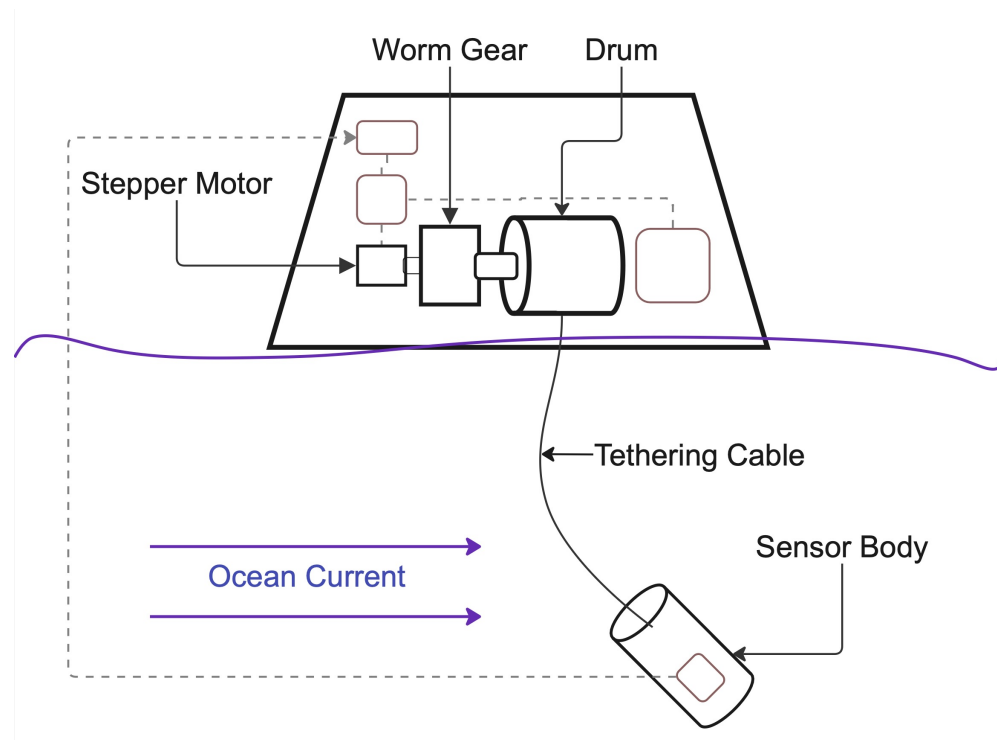


Figure 1. Mechanical components of the system.

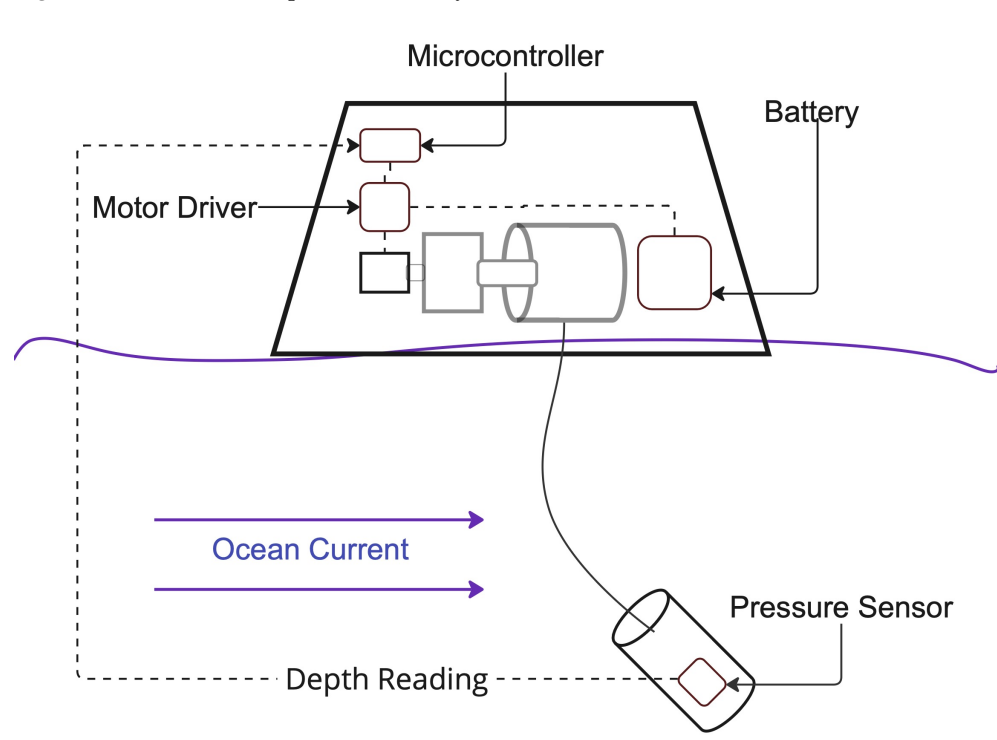


Figure 2. Electrical components of the system.

The mechanical aspect is composed of a worm gear, drum, tethering cable, and sensor body. The sensor body is considered the payload, and can comprise a variety of sensors to fit a given need. It will need to be waterproof to protect the components inside. Connected to the body is the tethering cable. The length of cable released will be the only element controlling the depth of the sensors, simplifying the system as there would be no active

depth-control elements on the sensor body, like propellers. The motor will drive the worm gear, which turns the drum and the amount of cable released. The worm gear prevents back-driving of the system, locking it in place when the motor is not moving. This forgoes the need for electric brakes.

The electrical components are the battery, motor driver, microcontroller, and the depth sensor. To provide feedback to the microcontroller, the depth of the sensor body is determined through the pressure. This data is fed into the control algorithm to determine the next control move need, which is sent to the motor through the motor driver. The microcontroller provides the ability to transmit the data locally to a nearby vessel or shore, or to the internet. To extend the capabilities of the sensor buoys, a network of buoys could be deployed in an area to extrapolate the water column data to cover a given area.

The expected use case for the system is to lower the sensors to a maximum depth, and then raise and hold at desired setpoints along the path back to the surface. The holding at depths allows different types of sensors to stabilize readings before proceeding to the next depth. A smooth setpoint trajectory can be generated that will move the sensor to points while preventing motor damage by introducing acceleration. Practical applications of this system may be subject to currents and wave motion that will pull the sensors off this trajectory, as shown in Figure 3. For sites that are in deeper, open-water environments, strong cross-currents eliminate the ability for the controller to use an encoder attached to the drum as feedback, as the amount of cable released would not be an accurate measurement of the depth of the sensor. Instead, position feedback is done through a local pressure reading from the sensor payload.

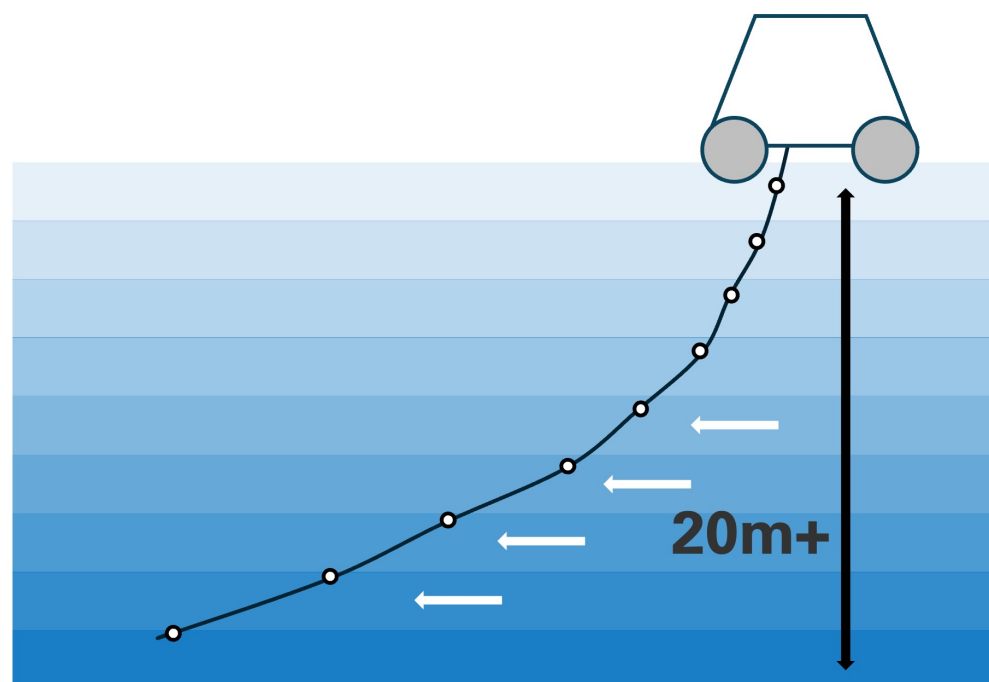


Figure 3. An example of a non-ideal use case, where the payload is pulled off the vertical axis by a current.

This work examines the depth control aspect of the sensor system, and uses a simplified model to run simulations tracking example trajectories. The control algorithm selected for these simulations is Model Predictive Control (MPC). MPC has many applications and has found success in areas like industrial control [10] and underwater positioning systems for Autonomous Underwater Vehicles [11,12].

The MPC algorithm is tested through simulation performed in MATLAB_R2024b. These simulations were performed in stages, starting simply before adding on more com-

plex elements. Upon successful simulations, a physical test bench was constructed and representative tests were able to be performed. Stability and robustness analyses and corresponding testing are left for future work.

Current smart aquaculture applications focus on highly controlled sites like onshore tanks or consistently predictable sites like shallow ponds. Static sensors are sufficient for these sites. This work aims to provide a platform to allow for the collection of high-quality water column data in a variety of deeper water environments, including those with ocean or river currents. The MPC-based controller provides flexibility through arbitrary setpoint trajectories and, as an optimal control scheme, can reduce the amount of time the motor is running, prolonging the battery life of the system.

2. Methodology

2.1. Model Predictive Control

Model Predictive Control is a type of optimal control that iteratively drives a system towards a desired setpoint through predicting a system response to a given control move. A prediction horizon N is selected as time t to $t + N$, as well as a control action horizon n_u . At time t , the optimal control actions in the control horizon are calculated and the system response is predicted from time t to $t + N$ based off the first control action. The prediction horizon is then shifted to $t + 1$, $t + N + 1$ and the process is repeated. One of the strengths of MPC is its ability to easily handle multi-input, multi-output (MIMO) systems [13]. While this work simulates a single-input, single-output (SISO) system for simplicity, by using MPC, there is a foundation for future work to easily adapt to a MIMO system that has potential to be more accurate under practical water conditions.

Calculating the optimal control action requires knowledge of the system's open-loop response. The system's response to an open-loop test over N is discretized and stored in array P as

$$P = \begin{bmatrix} P_1 & P_2 & \dots & P_n \end{bmatrix}_{1 \times N} \quad (1)$$

An example system is shown in Figure 4. The values of P can be found through simulation of a system's response, or from sampling a practical system. However, the prediction process can be sensitive to noise in the dynamic matrix. To minimize errors when using sampled data, an approximate system model can be determined and used to simulate a much less noisy open-loop test.

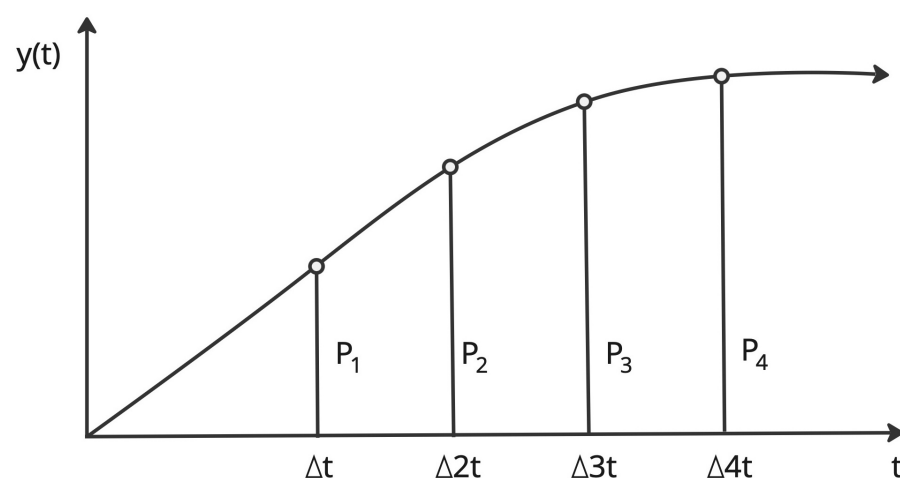


Figure 4. Example system response used to illustrate the normalized discretized values of P .

The goal of most control algorithms is to minimize the error between a desired setpoint and the system's output, and MPC takes this further by looking to minimize not only the error at the current time t , but also future errors. An example system is shown in Figure 5. In this example, there is an n_u of 3.

For each time step t , the system output y is iteratively driven towards the setpoint Y_{sp} by a control action at time t , represented by u_t . The dashed lines represent the system's output based solely on the input u_t from a starting point of P at the previous time step, P_{t-1} . This can be used to predict future values of system output denoted as $\hat{y}$. Generally, this prediction can be represented as

$$\hat{y}(t+k|t) = \hat{y}_0(k+1) + \sum_{i=1}^N P_i \Delta u(t-i+k) \quad (2)$$

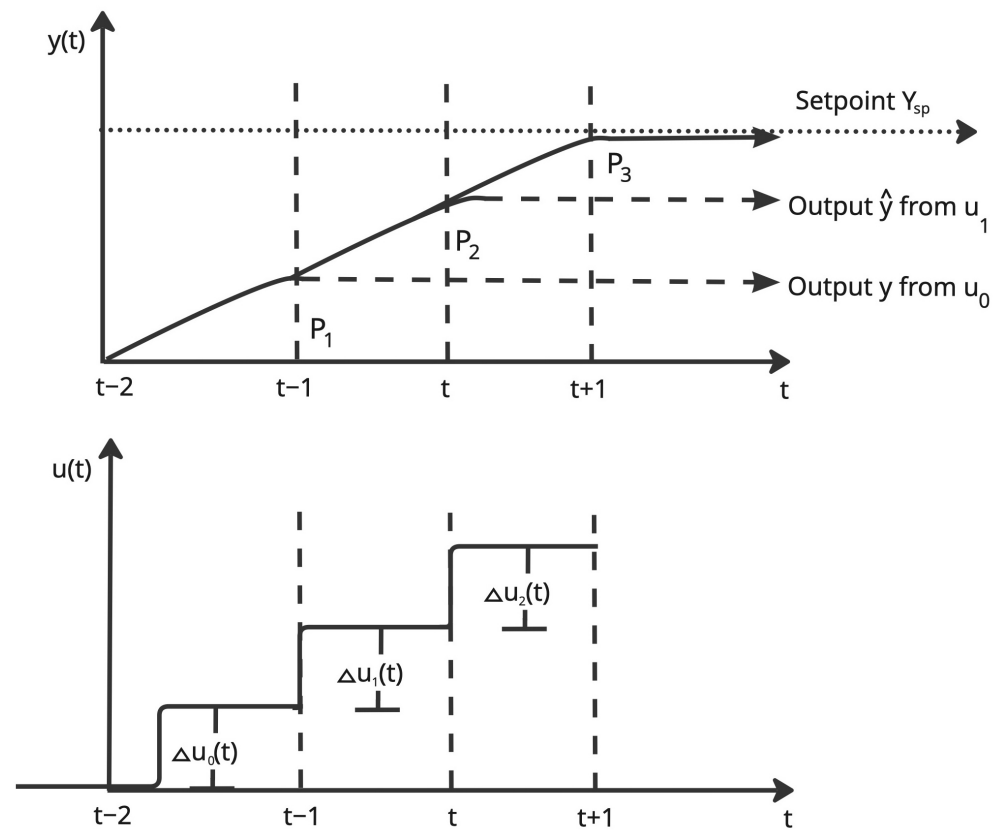


Figure 5. An example system iteratively driven towards a setpoint Y_{sp} .

The errors e_1 , e_2 , and e_3 in the example system can be represented as

$$e_1 = Y_{sp} - (\hat{y}_1 + P_1 \Delta u_0 + P_0 \Delta u_1) \quad (3)$$

$$e_2 = Y_{sp} - (\hat{y}_2 + P_2 \Delta u_0 + P_1 \Delta u_1) \quad (4)$$

$$e_3 = Y_{sp} - (\hat{y}_3 + P_3 \Delta u_0 + P_0 \Delta u_3) \quad (5)$$

and stored in a vector E . Generally, the error vector E is

$$E = \begin{bmatrix} e_1 & e_2 & \dots & e_N \end{bmatrix}_{1 \times N} \quad (6)$$

The Jacobian matrix J can be used to minimize our errors. If

$$J = \sum_{i=1}^N e_i^2 \quad (7)$$

then to minimize with respect to the control actions, the equation to solve is

$$\frac{\delta J}{\delta \Delta u} = 0 \quad (8)$$

To assist in the matrix math, a dynamic matrix A is created. This matrix is built from the elements of P . An example of a dynamic matrix for $n_u = 3$ can be shown as

$$A = \begin{bmatrix} P_1 & 0 & 0 \\ P_2 & P_1 & 0 \\ P_3 & P_2 & P_1 \\ \vdots & \vdots & \vdots \\ P_N & P_{N-1} & P_{N-2} \end{bmatrix}_{N \times n_u}$$

Solving (8) provides us with the following equation for control moves:

$$\Delta u = [(A^T A)^{-1} A^T (E)]_{1 \times n_u} \quad (9)$$

Other solutions for Δu can involve weighting matrices on the error vector and control moves.

2.2. Implementing MPC

As discussed, the initial step for this MPC implementation is to sample the process and find vector P , and from there create matrix A . For simplicity, the system's initial state is represented as zero. The setpoints should also be defined and stored in Y_{sp} . With variables initialized, the online control loop begins.

The first step is to sample the system's output, and compare it to Y_{sp} to find E . With E , (9) can be solved. A control vector u can be found as

$$u = u^- + \Delta u \quad (10)$$

where u^- is the previous control action vector. The first value of vector u is sent to the system as the control action. For the future simulations, a discretized equation of the system is used to simulate the system output at time t and stored in $y_m(t)$. The plant output prediction can then be determined as

$$\hat{y} = \hat{y} + P \Delta u(0) \quad (11)$$

The setpoint and prediction vectors Y_{sp} and $\hat{y}$ are shifted to advance them to the next time step. The process can then be repeated, returning to the sampling of the system and the calculation of E . More details on MPC can be found in [14–18]. An overview of the implemented algorithm can be shown as (Algorithm 1):

Algorithm 1: Implemented MPC algorithm.

```

for  $i = 1 : N$ 
  1. Measure  $y_m$ 
  2. Adjust prediction  $\hat{y}$ 
  3. Evaluate  $E = Ysp - \hat{y}$ 
  4. Evaluate  $\Delta u = (A^T A)^{-1} \times A \times E$ 
  5. Evaluate  $u = u^- + \Delta u$ 
  6. Send  $u(0)$  to system
  7. Find new  $\hat{y} = \hat{y} + A(:,1)\Delta u(0)$ 
  8. Advance horizons
  9. Update  $u^-$  to be  $u$ 

```

2.3. System and Modeling

The plant comprised a microcontroller (μC), motor and motor driver, a drum to release the cable, and a depth sensor attached to said cable. Initial simulations were performed with a representative model of the plant. It considers ideal equations for the stepper motor, the worm gear, and the drum. Complex aspects like the effect of releasing cable on drum inertia, or the effects of ocean conditions were not included and left for future work.

Stepper motors are a type of permanent magnet motor that rotate by a fixed angle for every input pulse received. The speed at which the motor rotates is determined by the rate the pulses are being sent [19]. Figure 6 illustrates changes in pulse rate. This method of motor control is called pulse frequency modulation (PFM). PFM was chosen over the more common pulse width modulation technique for its intuitive comparison to the steps of the stepper motor, where each pulse represents a step of the motor. Increasing the frequency of the steps increases the speed of the motor.

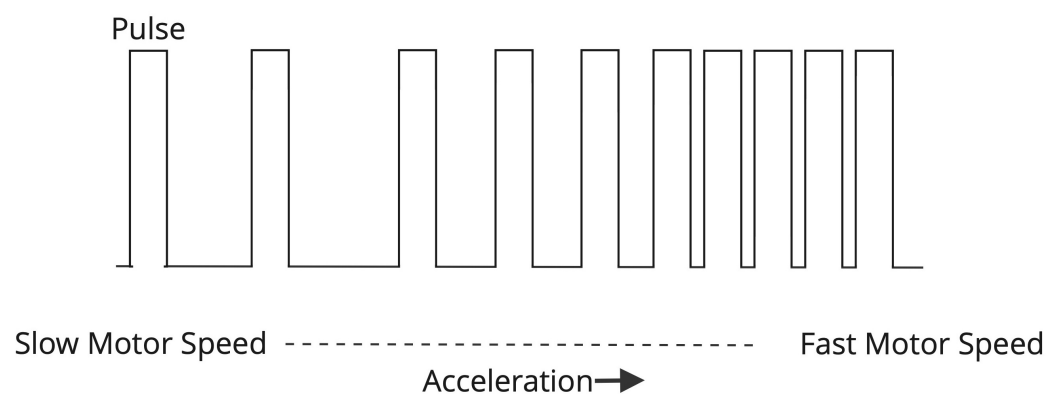


Figure 6. Illustration of pulse frequency modulation.

Figure 7 illustrates a block diagram of the system.

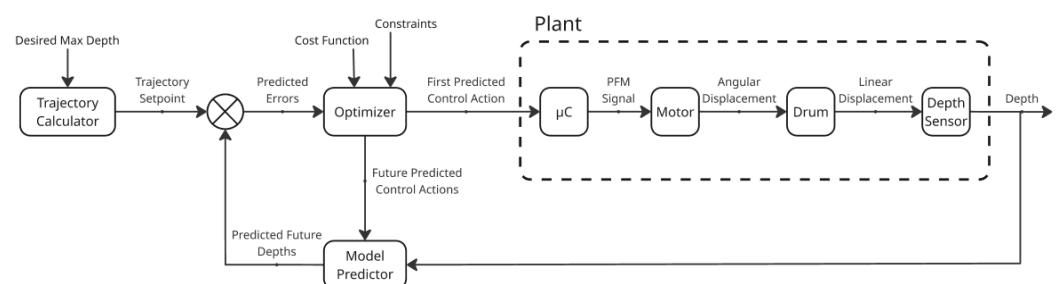


Figure 7. Block diagram of the system.

With step angle θ_s and frequency f , and assuming the system is ideal with a constant load, angular velocity of a stepper motor can be calculated as

$$\omega(\text{rad/s}) = \frac{\theta_s/\text{step}}{360} \times f(\text{Hz}) \quad (12)$$

The system includes a worm gear between the motor and drum to prevent back driving, with gear ratio g . Converting this angular velocity of the motor shaft ω to output linear velocity of the drum v is performed by

$$v = \omega r = \frac{\theta_s}{360} f \times r \quad (13)$$

The transfer function can then be found as

$$sX(s) = \frac{\theta_s}{360} r F(s) \Rightarrow \frac{X(s)}{F(s)} = \frac{\frac{\theta_s}{360} r}{s} \quad (14)$$

For the simulation to match hardware already procured, the simulation uses a step angle of 1.8° , a gear ratio of 1:10, and a drum radius of 0.10 m.

$$\frac{X(s)}{F(s)} = \frac{0.5 \times 10^{-3}}{s} \quad (15)$$

This transfer function is used to find the step response of the system required to generate a dynamic matrix A . The system transfer function was also discretized and used to simulate the response of the plant each simulated time step.

After the completion the test bench further simulations were done by approximating a transfer function from practical test data, as discussed previously. This approximation was achieved through MATLAB_R2024b's System Identification toolbox. A more accurate transfer function for the system, based on the test data, is

$$\frac{X(s)}{F(s)} = \frac{0.1 \times 10^{-3}}{s} \quad (16)$$

This transfer function is very similar to (15), albeit with a smaller gain, likely due to the omission of drag in the initial transfer function. The time constant effect is not included as it is negligible.

To avoid malfunction of the motor, smooth setpoints are tracked. MPC will use these smooth setpoints will give control actions that accelerate the motor to its operating speed, and decelerate to stop.

2.4. Trajectory Planning

The setpoint profile is calculated from a defined start point x_0 , end point x_f , and the desired end time for the motion t_f . With those defined values, and requiring velocities to be zero at the start and end points, there are four constraints to motion. A 3rd-order polynomial can be used to solve for these constraints. The constraints can be shown as

$$x(0) = x_0 \quad (17)$$

$$x(t_f) = x_{tf} \quad (18)$$

$$\dot{x}(0) = 0 \quad (19)$$

$$\dot{x}(t_f) = 0 \quad (20)$$

Position can be expressed as a 3rd-order polynomial

$$x(t) = a_0 + a_1t + a_2t^2 + a_3t^3 \quad (21)$$

resulting in equations for velocity and acceleration as

$$\dot{x}(t) = a_1 + 2a_2t + 3a_3t^2 \quad (22)$$

$$\ddot{x}(t) = 2a_2 + 6a_3t \quad (23)$$

Using the constraints, the coefficients a_1 , a_2 and a_3 can be solved for the following:

$$a_0 = x_0 \quad (24)$$

$$a_1 = 0 \quad (25)$$

$$a_2 = \frac{3}{t_f^2}(x_f - x_0) \quad (26)$$

More details can be found in [20]. Figure 8 shows an example of the output from (21), (22), and (23) to produce position, velocity, and acceleration profiles for a motion from 0 m to 10 m over a 20 s timespan.

Multistep trajectories needed for simulations are formed through a combination of these profiles over a given timeframe, as shown in Figure 9. This segmentation ensures that each step follows a smooth motion.

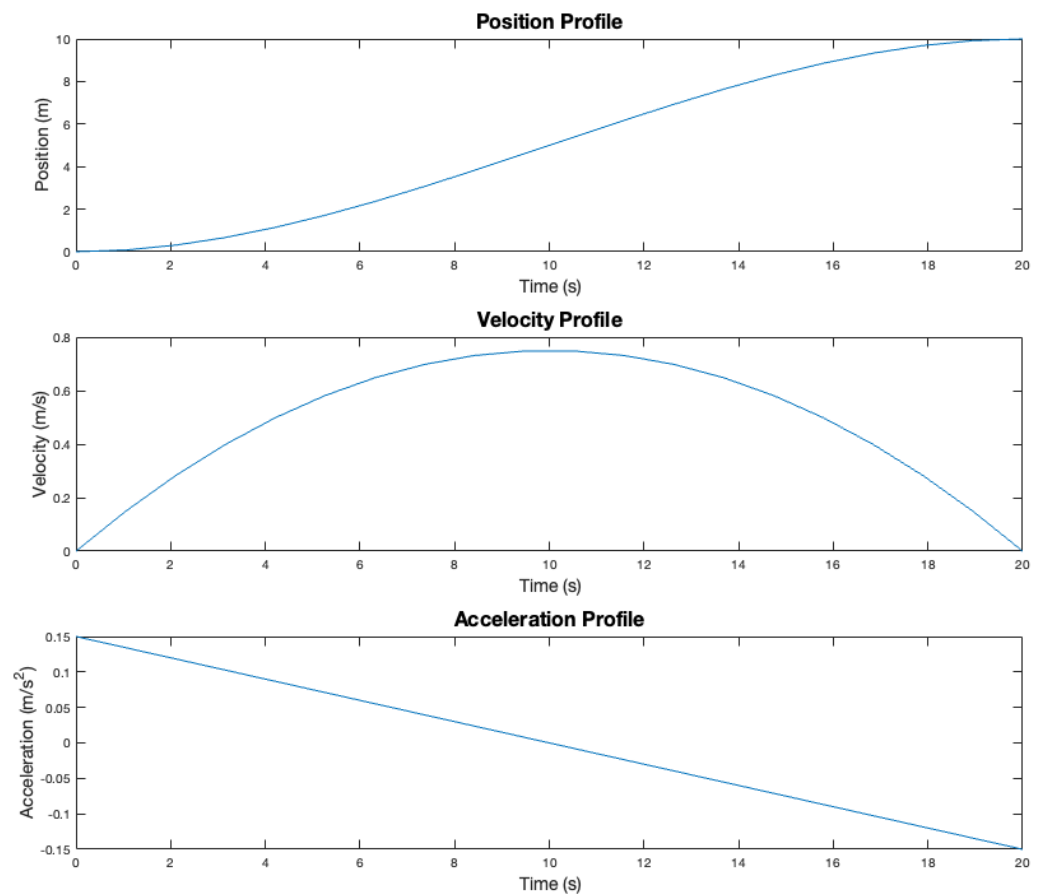


Figure 8. Example trajectory for smooth motion from 0 m to 10 m.

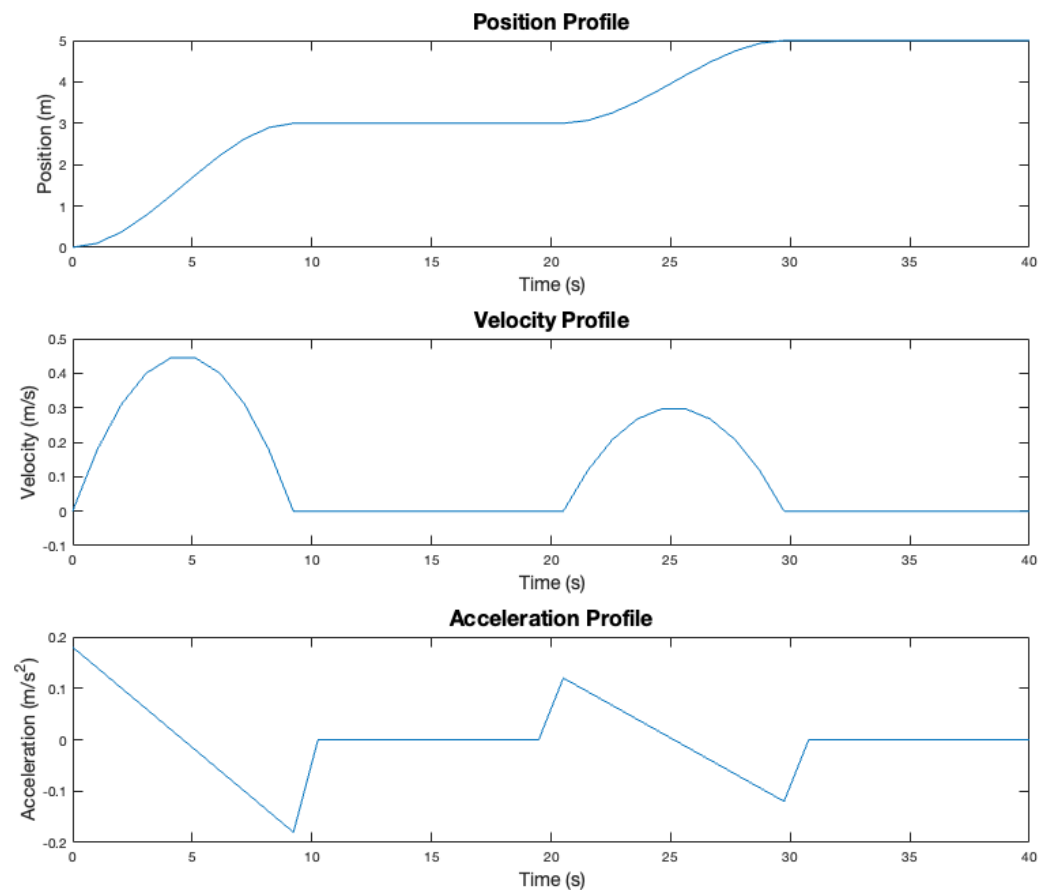


Figure 9. Example complex trajectory for smooth motion from 0 m to 5 m, with a pause and hold at 3 m.

3. Results

3.1. Simulated Results

The descent and ascent were treated as two separate trajectories and their results combined to one plot. Two simulations were performed: one with a shallower max depth of -5 m, and one with a deeper max depth of -10 m. All simulations were performed in MATLAB_R2024b. Typically, the value of N is selected so that it is when the system reaches at least 97% of steady state response. For this system, reaching the bottom of the tank in an open-loop test was treated as reaching steady state.

To better match a practical model of the system, white noise was added to the simulated outputs and control moves were constrained. The constraints were set to keep the pulse rate inputs between ± 1.5 kHz, based on initial testing with the system's stepper motor. Frequencies higher than that can cause the motor to malfunction.

The trajectory for the simulation is shown in Figure 10. The descent is a simpler path, as on the ascent, the payload is expect to be held at a certain depth to allow for sensors to stabilize before continuing to the next setpoint. The MPC algorithm tracks the setpoints well, as shown in Figure 11.

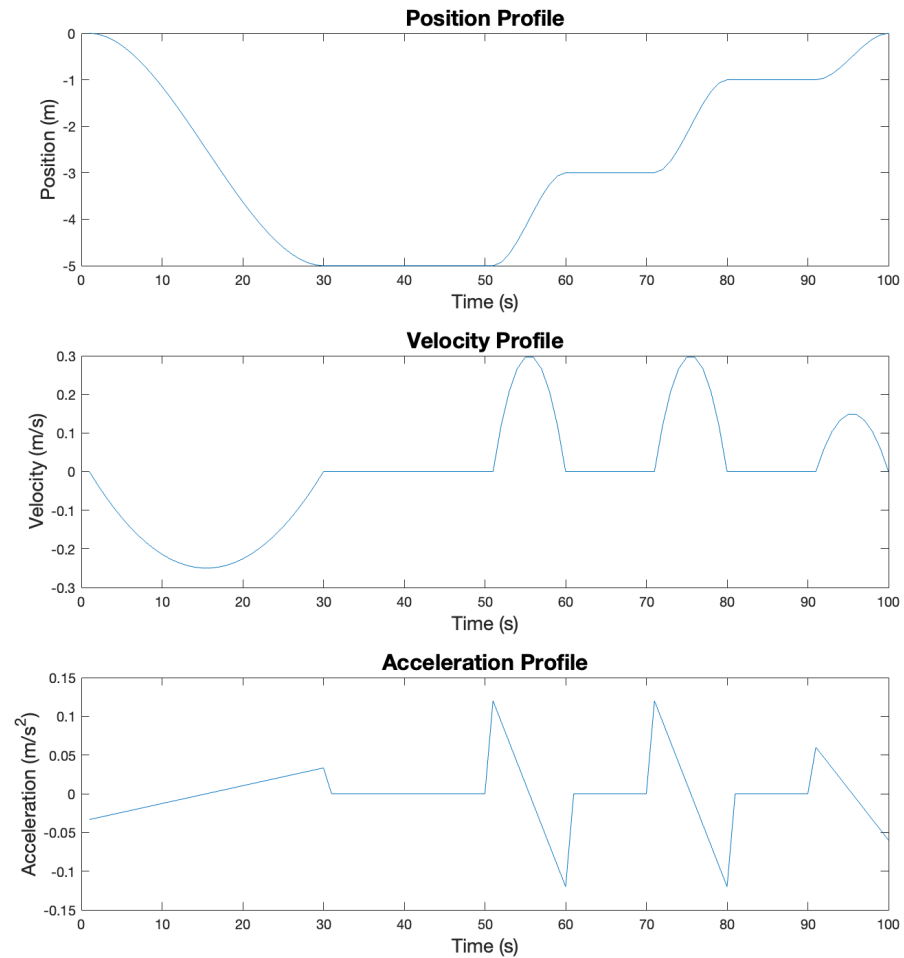


Figure 10. Example trajectory that matches the expected use case of the system.

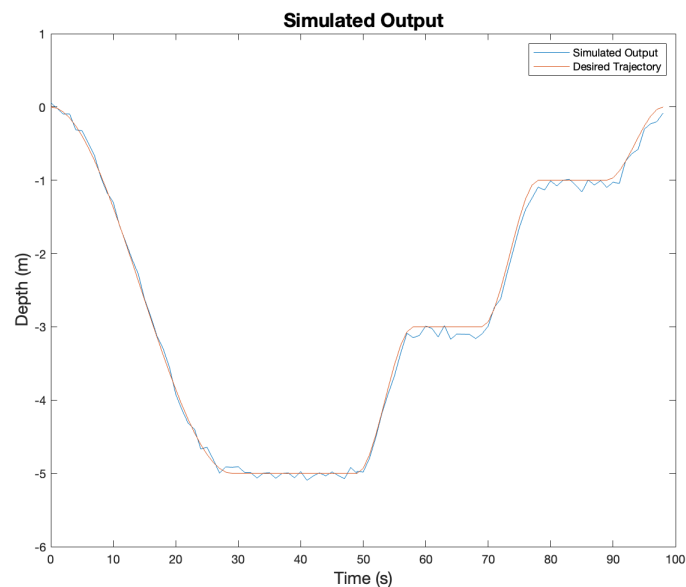


Figure 11. Simulation results of the shallow test.

The second simulation was performed with a deeper max depth of -10 m. Overall tracking is performed well, as shown in Figure 12. However, the system output falls behind the profile most notable on the spans between -6 m and -10 m. This is due to the magnitude constraint put on the system input. The optimal move would be to continue to increase the frequency; however, this would not be feasible on the practical motor. Figure 13

compares the unconstrained control moves and the constrained ones for the rising part of the trajectory.

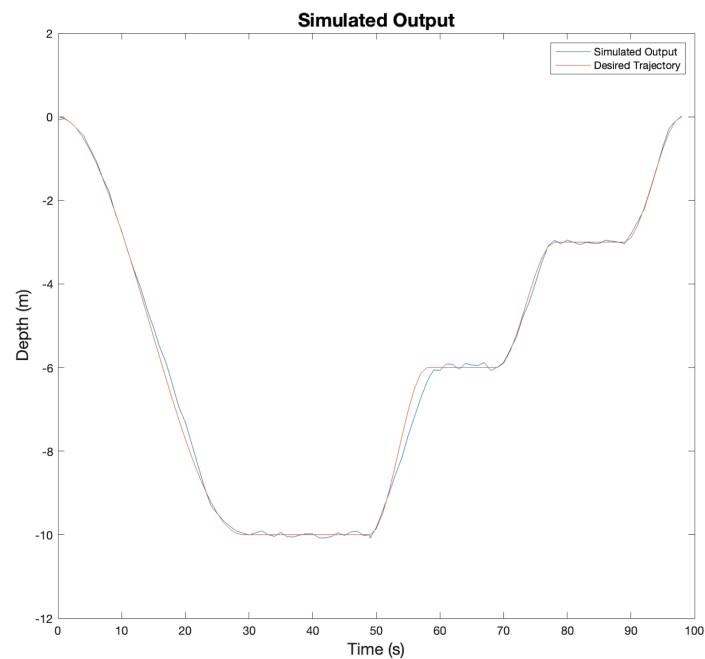


Figure 12. Simulation results from the deep test.

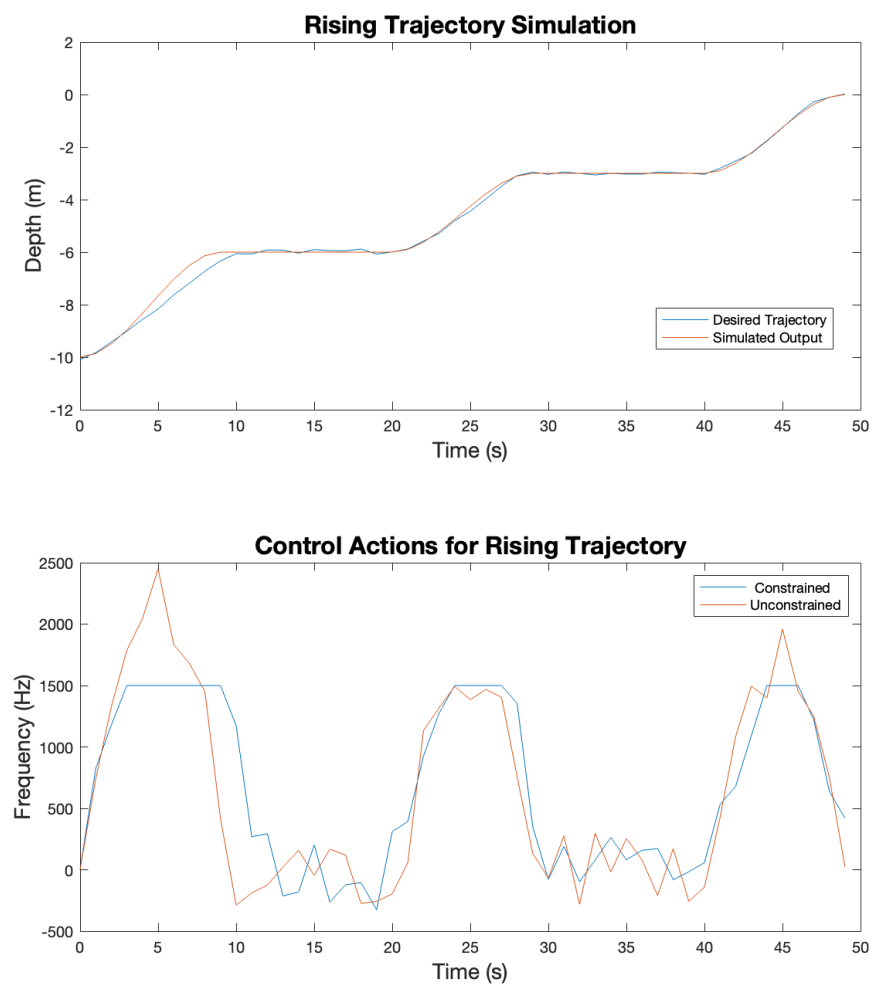


Figure 13. Control actions for the deep test, comparing the constrained and unconstrained actions.

3.2. Practical Testing

With promising simulated results, a practical test bench was constructed. While previous work was performed in MATLAB_R2024b, in order to implement the test on a microcontroller, the algorithm was converted to C++. The microcontroller used was an MKR 1500 NB, due to its availability and the built-in cellular modem, which can be used in future work to implement the desired IIoT aspect. Testing software was written to allow a user to input their desired depth for each test, and the collected data is printed in a format MATLAB_R2024b accepts to allow for quick plotting and data analysis. The created test is best shown in Figure 14.

Table 1 outlines the components used to create the test bench.

Table 1. Components used to create test bench.

Component Type	Component Selected
Stepper Motor	175-QSH5718-76-28-189-ND
Motor Driver	DRI0043-ND
Microcontroller	Arduino MKR NB 1500
Depth Sensor	MS5837-30BA
Power	Bench-top power supply

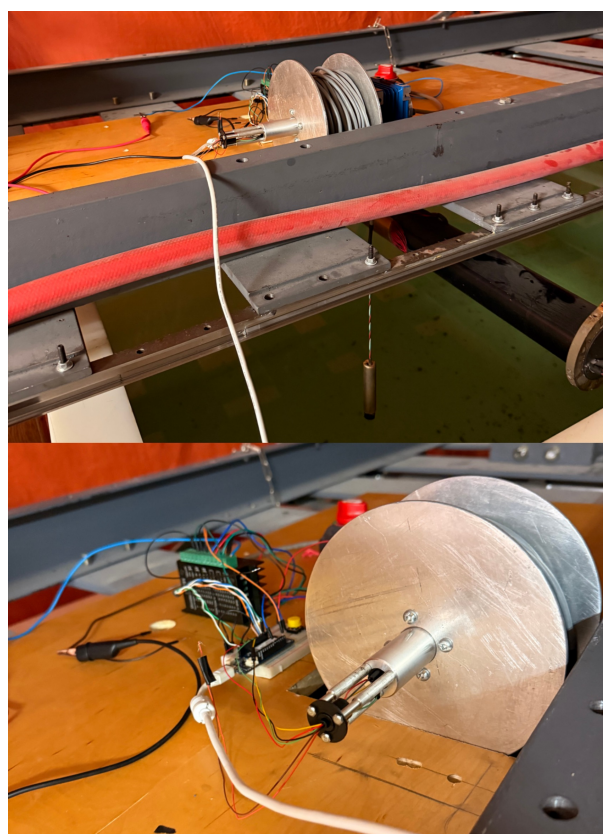


Figure 14. Photos of the test bench above the testing tank.

The previous simulations were performed with the expectation that a 10 m tank would be available for testing. However, the practical tests were conducted in a 1.4 m tank, which was more easily accessible. To generate the dynamic matrix for the 1.4 m tank, an open-loop test was conducted. As with the simulations, reaching the bottom of the tank was treated as reaching the steady state, and thus N was based off of the time steps required to reach the bottom in the open-loop test.

This data was then used to find the transfer Equation (16). A comparison of the practical step response and the simulated step response is shown in Figure 15.

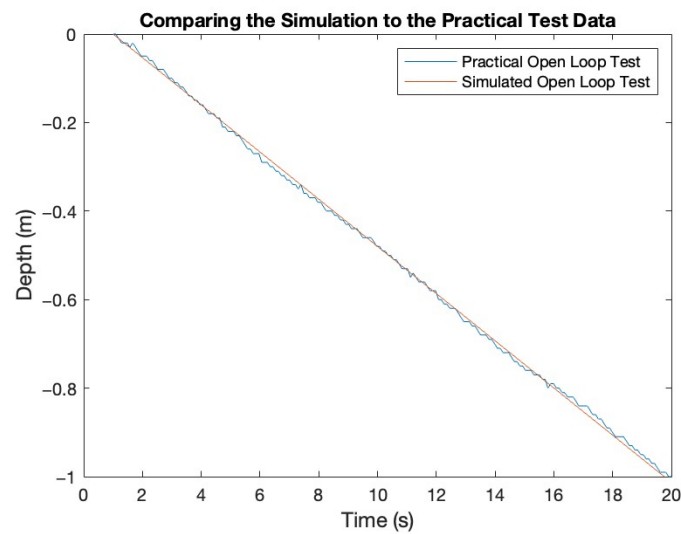


Figure 15. Comparison of practical step response data to simulated response data.

The noise seen in the practical response would propagate to the control actions if used for the creation of the dynamic matrix. A control simulation showing the effect of noise in the dynamic matrix is illustrated in Figures 16 and 17. Figure 16 is the simulation with the noisy dynamic matrix, and the effect can be seen in its inability to reach the setpoint on both descent and ascent, as well as holding at 0.6 m instead of 0.5. The control moves oscillate heavily as well. Figure 17 is the same simulation, but with a dynamic matrix that was generated from the MATLAB_R2024b System Identification Toolbox output. This simulation tracked the setpoint exactly, except where the magnitude limitation prevented it.

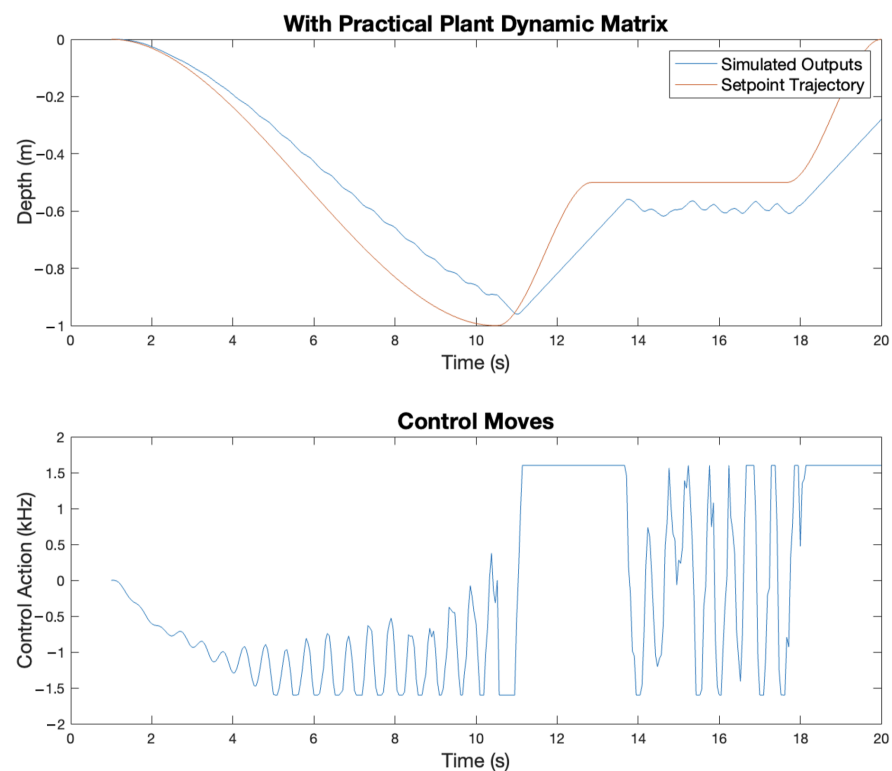


Figure 16. A simulation done with the practical open-loop response data.

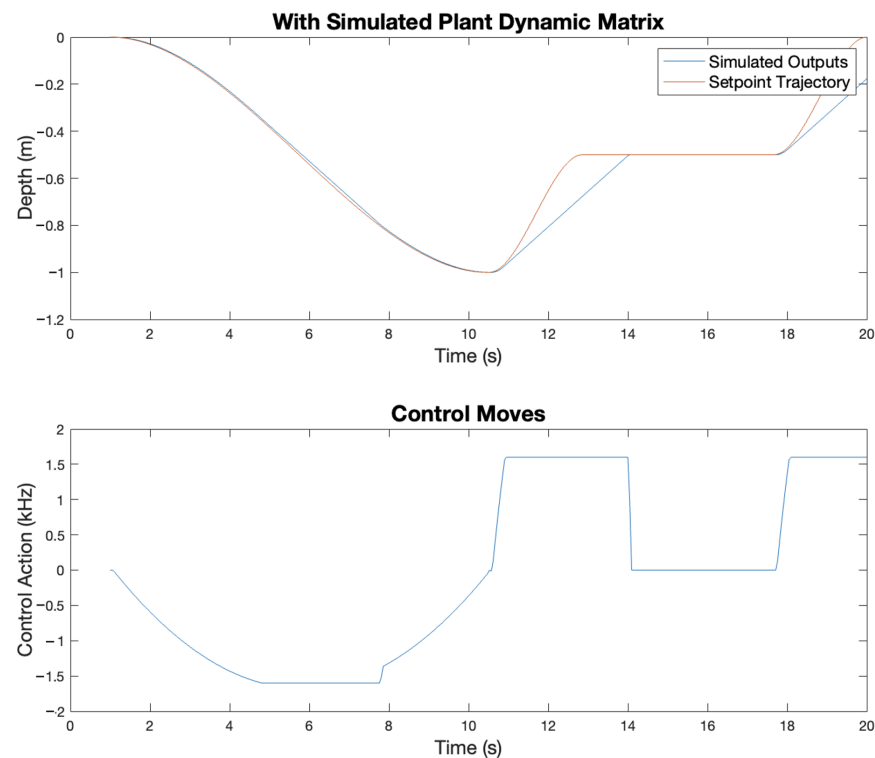


Figure 17. A simulation done with the simulated open-loop response.

With the dynamic matrix, testing in the tank could begin. All tests were performed with the same profile: descend to 0.5 m, ascend to and hold at 0.25 m, and then return to the surface. By only going to 0.5 m, when the depth sensor is lowered without the simulated ocean current effects, it does not reach speed constraints, which helps to highlight the effects of the simulated current in later tests. Ocean current was simulated by pulling on the depth sensor as it lowered, as if there was a current pushing it off its vertical axis.

The first test was conducted without any interference, and is shown in Figure 18. On the descent, the sensor's depth tracks the setpoint well, but with a slight delay. This delay is also seen on the ascent. There were small fluctuations on depth during the hold, but only of a max of ± 2 cm. Overall, this test was considered a success, as the ability to hold position accurately is the most important aspect of the controller, followed by its ability to track a smooth trajectory.

The second test has the simulated ocean current, and the results are shown in Figure 19. As the sensor was lowered, it was also pulled perpendicular to the vertical axis. This can be seen in the results, as the tracking is farther off than in the first test, and the control actions are much higher. In order to attempt to track the profile, the motor had to run faster to counteract the effect of the ocean current. The control actions get limited by the 1500 Hz cutoff at around the four-second mark, limiting the ability to track the trajectory. The simulated current slowed the ascent as well, but did not limit the systems ability to adapt and hold at 0.25 m. The delay getting to the setpoint does lower the time spent at the hold depth. This can be mitigated by holding longer than needed for a given sensor, as this is not a time critical application.

A third test was conducted to observe how the system would react to a sudden displacement instead of the slow, gradual implementation of the ocean current performed in test two. Figures 20 and 21 show the descent and ascent respectively, separated to allow for closer examination. On the descent, the displacement happened just after four seconds, and the system quickly responded by increasing the motor speed. On the ascent, the sudden displacement was done during the hold, to test the systems ability to adjust during

the most important aspect of the trajectory. At 14 s, the sensor was displaced, getting a distance of 5 cm away from the setpoint before being placed back. The controller quickly responded to the displacement, bringing the motor up to speed from standstill.

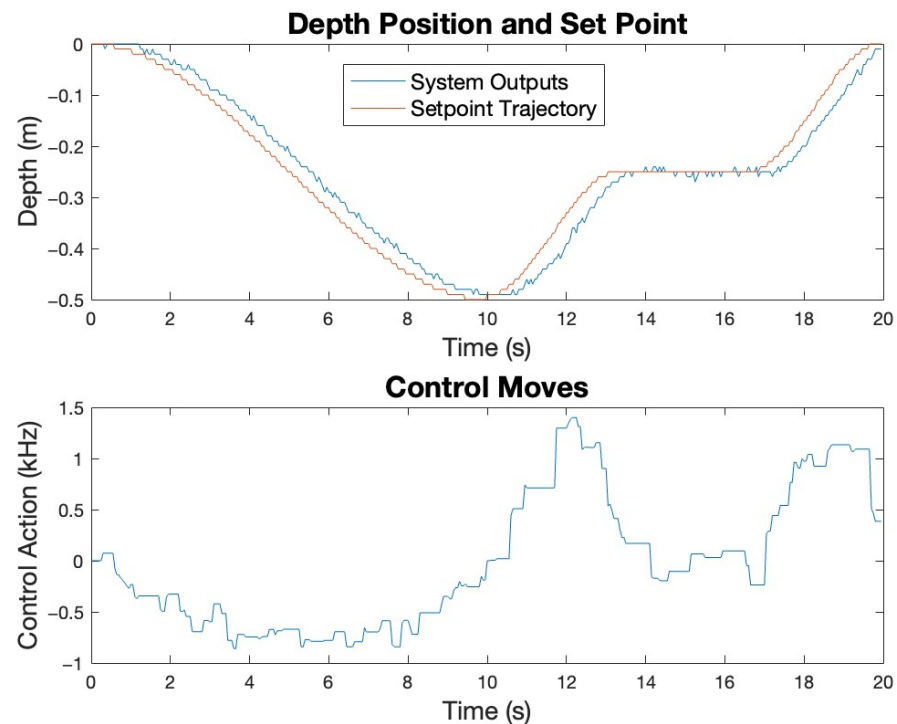


Figure 18. Results from Test 1—No disturbances.

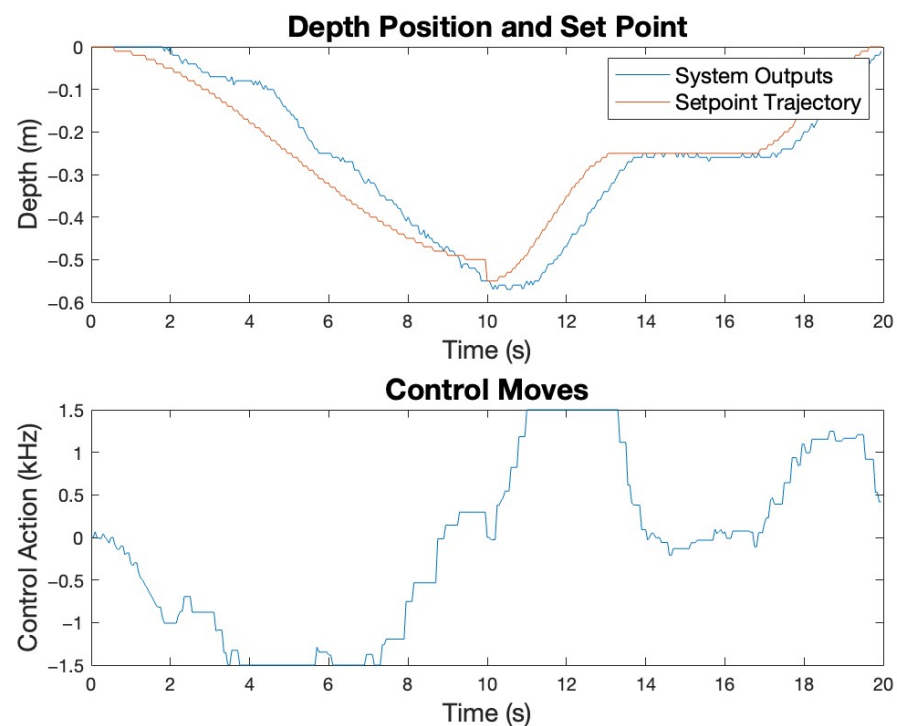


Figure 19. Results from Test 2—Ocean current disturbance from the beginning.

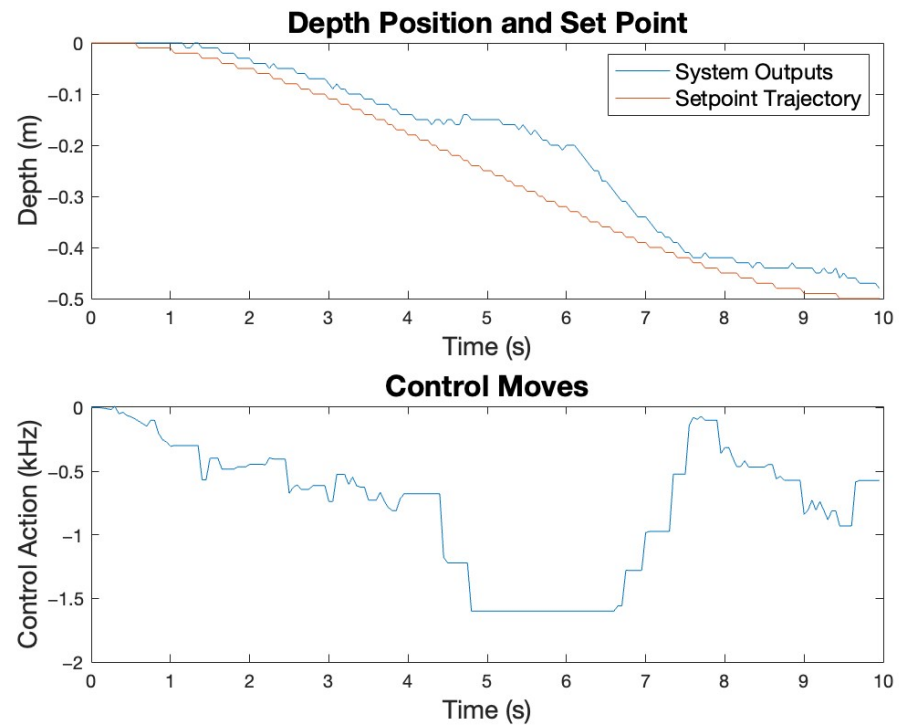


Figure 20. The descent of Test 3—Sudden Disturbance.

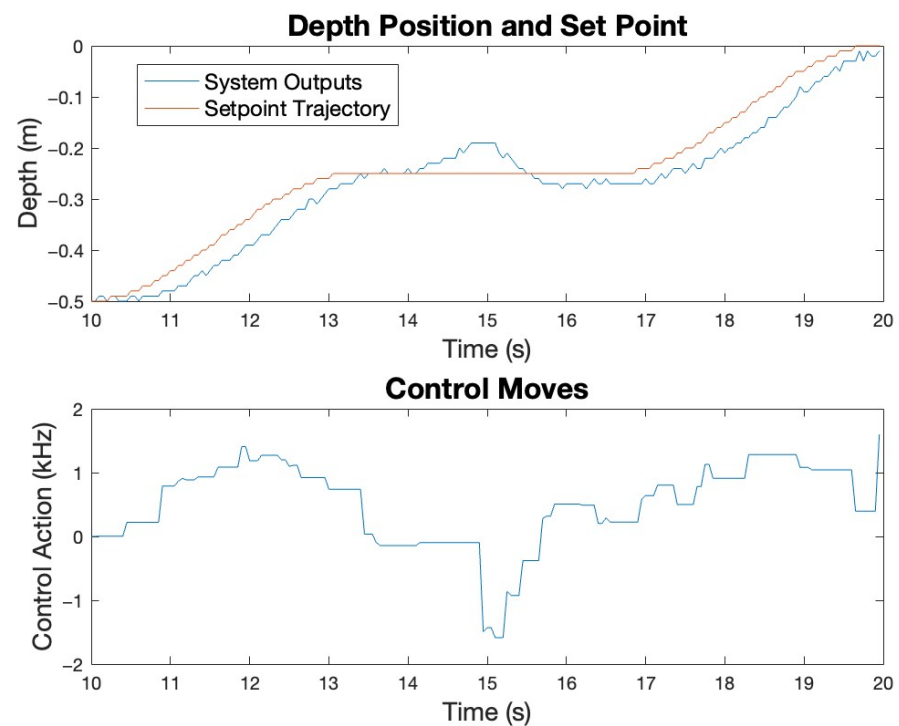


Figure 21. The ascent of Test 3—Sudden Disturbance.

The Mean Square Error (MSE) can be used to evaluate how well the system tracked the desired setpoint trajectory. MSE can be found using the following equation:

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_{m_i} - Y_{sp_i})^2 \quad (27)$$

where n is the total number of samples to compare, y_{m_i} is the measured value at index i , and Y_{sp_i} is the setpoint at index i . If the difference between the target setpoint and the measured value is small, the MSE will be small as well. A lower MSE value is indicative of better tracking. Table 2 contains the MSE values for the three practical tests.

Table 2. Mean Square Error values for practical tests.

Test Number	MSE
Test 1—No Disturbance	0.00093275
Test 2—Constant Disturbance	0.00343450
Test 3—Sudden Disturbance	0.00104725

Test 1, without the ocean current simulation, expectedly has the lowest MSE. At most, the measure value deviated away from the setpoint by ± 5 cm, during the descent. The depth-sensitive hold had a deviation of ± 2 cm. This tracking was successful. While still small, test 2 had a larger MSE compared to test 1, which was to be expected. The system was actively displaced off the vertical axis. The largest contributor to this increase are during the active ascent and descent. After reaching the hold, the system deviated just ± 2 cm, similar to test 1. Test 3's MSE was closer to test 1, as the compared to the overall trajectory the disturbance was during a small portion of the test. Low MSE values support the systems ability to accurately track the desired setpoint trajectory during these initial practical tests.

4. Discussion

4.1. Simulations

The simulations proved to be a valuable tool in fine-tuning the control algorithm on a representative system. While the initial simulations did not completely represent the future test bench, the form of the transfer function was similar. The second set of simulations used the dynamic matrix from the practical testing, which helped to identify the issues caused from the noisy dynamic matrix. When the test bench was first built, preliminary tests showed odd behavior, settling at 80% of the desired setpoint. By using the simulation, it was determined to be an issue with the dynamic matrix, and not the test bench itself. Overall, simulations were a success, and provided insight needed to conduct the practical testing.

4.2. Practical Testing

The practical testing was a success given the constraints of the testing environment. The controller was able to position the depth sensor to specified depths along a smooth trajectory, albeit with a slight delay. At this time, the delay is not a concern as the main focus is to be able to drive the system to the specified depth, and hold if required. As long as collected temperature, salinity, pH, etc., data is compared to the collected depth values, the water column data will still be accurate.

The method to simulated current was not ideal, yet provided a way to test the systems response to disturbances. By comparing the control moves from the undisturbed test in Figure 18 to any of the other tests, it is clear that the system can recognize when the sensor is being pulled off-axis. The system would likely be able to return to the desired trajectory quicker if not for the limitations on the control moves. The desired trajectory and control move limitations should be taken into account when planning routes for the sensor. Both can be adjusted, yet the goal of limiting the control moves is to protect the motor and thus care should be taken if adjusting them. Penalty matrices can be applied to the control move and errors in the objective function if needed to reduce the magnitude of the control moves.

With the ability to track the trajectory, even in the presence disturbances, these small tank tests are considered a success, and work is being done to progress the testing further under deeper conditions.

4.3. Future Work

The testing can easily be expanded on by testing in a larger tank. To more accurately represent typical conditions, the depth should be around 10 m. As more cable is released, the dynamics of the system will change as mass is removed from the drum. Future tests should consider this effect; however, the change in overall drum inertia dynamics at every time step should show up in the dynamic matrix and the system should handle the change accordingly. It would also be valuable to test with a more realistic, yet controlled current simulation. A flume tank or flow agitator could be used to push the sensor off the vertical axis.

5. Conclusions

There is a need for precise depth control of a tethered underwater sensor unit to monitor key ocean data and MPC shows potential as an algorithm for this task. Using MPC, a simulated system tracked two smooth motion trajectories. Input into the system was constrained to be below 1.5 kHz, which only affected the output while tracking the deeper of the two profiles. The effect of this constraint did not prevent the system from performing well. The successful simulations were followed by successful initial practical tests. The controller was able to track a setpoint, obtaining its feedback data from a real depth sensor in a small tank. These tests included a simple ocean current simulation, which the system responded well to. These initial practical tests can be expanded on by testing in a larger tank, and having a more realistic implementation of simulated current. If testing in a larger yet controlled tank yields promising results, open-water testing in a realistic environment can be completed.

Author Contributions: Conceptualization, M.O., R.D. and A.S.; methodology, M.O., R.D. and A.S.; software, M.O.; validation, M.O.; formal analysis, M.O.; investigation, M.O.; resources, M.O., R.D. and A.S.; data curation, M.O.; writing—original draft preparation, M.O.; writing—review and editing, M.O., R.D. and A.S.; visualization, M.O.; supervision, R.D. and A.S.; project administration, R.D. and A.S.; funding acquisition, R.D. and A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research as funded by the University of New Brunswick Graduate Research Assistantship and NSERC CREATE iMerit grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IIoT	Industrial Internet of Things
MPC	Model Predictive Control
MIMO	Multi-Input, Multi-Output
SISO	Single-Input, Single-Output
PWM	Pulse Width Modulation
MSE	Mean Square Error

References

1. Wang, C.; Li, Z.; Wang, T.; Xu, X.; Zhang, X.; Li, D. Intelligent fish farm—The future of aquaculture. *Aquac. Int.* **2021**, *29*, 2681–2711. [[CrossRef](#)] [[PubMed](#)]
2. Teja, K.B.R.; Monika, M.; Chandravathi, C.; Kodali, P. Smart Monitoring System for Pond Management and Automation in Aquaculture. In Proceedings of the 2020 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 28–30 July 2020; pp. 204–208. [[CrossRef](#)]
3. Dupont, C.; Cousin, P.; Dupont, S. IoT for Aquaculture 4.0 Smart and easy-to-deploy real-time water monitoring with IoT. In Proceedings of the 2018 Global Internet of Things Summit (GloTS), Bilbao, Spain, 4–7 June 2018; pp. 1–5. [[CrossRef](#)]
4. Azim, M.A.; Sarkar, I.; Chowdhury, M.A.H.; Rabbi, F. Towards Sustainable Aquaculture: An IoT-Driven Indoor Fish Farming System. In Proceedings of the 2024 IEEE International Conference on Power, Electrical, Electronics and Industrial Applications (PEEIACON), Rajshahi, Bangladesh, 12–13 September 2024; pp. 383–388. [[CrossRef](#)]
5. Achouch, M.; Dimitrova, M.; Ziane, K.; Sattarpanah Karganroudi, S.; Dhouib, R.; Ibrahim, H.; Adda, M. On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges. *Appl. Sci.* **2022**, *12*, 8081. [[CrossRef](#)]
6. Akhter, F.; Siddiquei, H.R.; Alahi, M.E.E.; Mukhopadhyay, S.C. Recent Advancement of the Sensors for Monitoring the Water Quality Parameters in Smart Fisheries Farming. *Computers* **2021**, *10*, 26. [[CrossRef](#)]
7. Jayandan, S.A.; Prathibanandhi, K.; Sahana, A.; Agiles, M.B.; Dhanush Moorthy, R.; Chethan, K. Smart Systems for Sustainable Aquaculture: A Focus on Water Quality. In Proceedings of the 2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 8–9 October 2024; pp. 1–4. [[CrossRef](#)]
8. Brewer, J.; Simoneau, A.; Dubay, R. Digital Twin Application to Ocean Monitoring Equipment. In Proceedings of the 2025 IEEE International systems Conference (SysCon), Montreal, QC, Canada, 7–10 April 2025; pp. 1–7. [[CrossRef](#)]
9. Singh, M.; Fuenmayor, E.; Hinchy, E.P.; Qiao, Y.; Murray, N.; Devine, D. Digital Twin: Origin to Future. *Appl. Syst. Innov.* **2021**, *4*, 36. [[CrossRef](#)]
10. Karau, F.; Leuer, M. Model Predictive Velocity Control of Electrical Drives on an Industrial-PC. In Proceedings of the 2022 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), Sorrento, Italy, 22–24 June 2022; pp. 76–81. [[CrossRef](#)]
11. Li, S.; Wang, S.; Luo, X. Depth Control of Autonomous Underwater Vehicles Based on Constrained Model Predictive Control. In Proceedings of the 2023 42nd Chinese Control Conference (CCC), Tianjin, China, 24–26 July 2023; pp. 2707–2712. [[CrossRef](#)]
12. Wang, J.; Wu, Z.; Tan, M.; Yu, J. Model Predictive Control-Based Depth Control in Gliding Motion of a Gliding Robotic Dolphin. *IEEE Trans. Syst. Man, Cybern. Syst.* **2021**, *51*, 5466–5477. [[CrossRef](#)]
13. Gehlaut, S.; Varshney, T.; Gupta, S. Performance Analysis of MPC for Level Control of Modified Quadruple Tank System. In Proceedings of the 2018 2nd International Conference on Power, Energy and Environment: Towards Smart Technology (ICEPE), Shillong, India, 1–2 June 2018; pp. 1–6. [[CrossRef](#)]
14. Wang, L. *Model Predictive Control System Design and Implementation Using MATLAB*; Springer: London, UK, 2010.
15. Camacho, E.F.; Carlos Bordons, J.M.M. *Model Predictive Control*, 3rd ed.; Springer: Cham, Switzerland, 2025.
16. Sebastian, W. Model Predictive Position Control for Permanent Magnet Synchronous Linear Motors. In Proceedings of the Innovative Small Drives and Micro-Motor Systems, 11th GMM/ETG-Symposium, Saarbrücken, Germany, 27–28 September 2017; pp. 1–6.
17. Kästner, F.; Werner, F.; Seydioglu, E.; Hübner, M. Design and Open Source Implementation of a Reconfigurable Hardware Model Predictive Controller Using Online Optimization. In Proceedings of the 2018 Conference on Design and Architectures for Signal and Image Processing (DASIP), Porto, Portugal, 10–12 October 2018; pp. 1–6. [[CrossRef](#)]

18. Huba, M.; Bistak, P.; Vrancic, D.; Sun, M. PID vs. Model-Based Control for the Double Integrator Plus Dead-Time Model: Noise Attenuation and Robustness Aspects. *Mathematics* **2025**, *13*, 664. [[CrossRef](#)]
19. Lixian, S.; Rahiman, W. A Compound Control for Hybrid Stepper Motor Based on PI and Sliding Mode Control. *IEEE Access* **2024**, *12*, 163536–163550. [[CrossRef](#)]
20. Craig, J.J. *Introduction to Robotics: Mechanics and Control*, 3rd ed.; Pearson Educacion Internacional: London, UK, 2005.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.