NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

MODELING AND IMPLEMENTATION OF A SPHERICAL UUV PLATFORM CONTROL SYSTEM FOR LITTORAL AND CONFINED-SPACE OPERATIONS

by

Matthew J. Coleman

September 2023

Thesis Advisor: James Calusdian
Second Reader: Ross A. Eldred

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
MODELING AND IMPLEMENTATION OF A SPHERICAL UUV PLATFORM CONTROL SYSTEM FOR LITTORAL AND CONFINED-SPACE OPERATIONS

5. FUNDING NUMBERS

6. AUTHOR(S) Matthew J. Coleman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Unmanned underwater vehicles (UUVs) offer a unique opportunity to safely explore areas too dangerous for human divers. The purpose of this research is to model and implement a robust closed-loop control method for the Wreck Interior Exploration Vehicle (WIEVLE), a novel UUV conceptualized, built, and tested exclusively on the Naval Postgraduate School campus. This work builds on previous open-loop testing and experimentation by a capstone Systems Engineering team of students in 2020. Expanding on their work, a MATLAB/Simulink simulation was created to test the proposed control scheme. To validate computer modeling results, the Pixhawk drone autopilot feasibility was tested as a commercial off the shelf solution for control. Due to the constraints of the autopilot, it was deemed infeasible for use in this application. We suggest as future work another approach that was unable to be implemented prior to release.

14. SUBJECT TERMS
AUV, UUV, WIEVLE, modeling, control, spherical, underwater, vehicle, autonomous undersea vehicle, unmanned underwater vehicle, Wreck Interior Exploration Vehicle

15. NUMBER OF PAGES
93

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

MODELING AND IMPLEMENTATION OF A SPHERICAL UUV PLATFORM CONTROL SYSTEM FOR LITTORAL AND CONFINED-SPACE OPERATIONS

Matthew J. Coleman
Lieutenant, United States Navy
BE, State University of New York, Maritime College, 2017

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2023

Approved by: James Calusdian
Advisor

Ross A. Eldred
Second Reader

Douglas J. Fouts
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Unmanned underwater vehicles (UUVs) offer a unique opportunity to safely explore areas too dangerous for human divers. The purpose of this research is to model and implement a robust closed-loop control method for the Wreck Interior Exploration Vehicle (WIEVLE), a novel UUV conceptualized, built, and tested exclusively on the Naval Postgraduate School campus. This work builds on previous open-loop testing and experimentation by a capstone Systems Engineering team of students in 2020. Expanding on their work, a MATLAB/Simulink simulation was created to test the proposed control scheme. To validate computer modeling results, the Pixhawk drone autopilot feasibility was tested as a commercial off the shelf solution for control. Due to the constraints of the autopilot, it was deemed infeasible for use in this application. We suggest as future work another approach that was unable to be implemented prior to release.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AUV	autonomous undersea vehicle
C	geometric center
CB	center of buoyancy
CdF	drag force coefficient
CdM	drag moment coefficient
CG	center of gravity
CL	closed loop
COTS	commercial off the shelf
DOF	degrees of freedom
ESC	electronic speed controller
GPS	global positioning system
I2C	inter-integrated circuit
INS	inertial navigation system
OL	open loop
PID	proportional, integral, derivative
RCVR	receiver
REMUS	remote environment monitoring units
ROS	robot operating system
TX	transmitter
UAV	unmanned aerial vehicle
UUV	unmanned undersea vehicle
WIEVLE	Wreck Interior Exploration Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This thesis would not have been possible without the unwavering support of my wife, Meredith. Her unfiltered feedback, both solicited and not, was a great sounding board throughout the research and composition of this document. To my mother, Cheryl, the legacy you left behind inspires me to keep asking the tough questions. I am grateful for my father, Thomas, whose 27 years of service in the Navy inspired my own career.

To the faculty and staff of the Naval Postgraduate School, thank you for the enriching experience that has been my master's degree. Dr. James Calusdian was instrumental in the technical aspects of this research. Thank you for the support and guidance through the tough parts. Mr. Ross Eldred, thank you for allowing me to be a part of your project. Dr. Roberto Cristi's course on linear systems introduced me to controls. His enthusiasm for education in the field inspired me to pursue the controls sequence of courses. Finally, many thanks to Dr. Xiaoping Yun for challenging me in his controls courses and greatly elevating my understanding of controls.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Since as early as 2000, the United States Navy has been working toward integrating unmanned undersea vehicle (UUV) technology into battlespace dominance [1]. While UUVs can and will play a role in tactical operations, they are also a proven tool for many other uses including salvage, deep sea exploration, search and rescue, and ocean research. As technology has advanced in the field of UUVs, there has been a steady trend toward intelligent vehicles that are able to autonomously make decisions to achieve their overall objective. As [1] suggests, the limits of unmanned technology, namely UUVs, have not yet been limited by imagination, but only budgeting and coordination. Despite these challenges, the Wreck Interior Exploration Vehicle (WIEVLE) was conceptualized in 2015 by LCDR Ross Eldred, U.S. Navy Reserve. The novel 12 in free-flooding spherical UUV was first intended to search confined-space shipwrecks in tangle prone environments, too dangerous for human divers or conventional submarine shaped UUVs [2].

A. BACKGROUND

To advance LCDR Eldred's research, a team of system engineering students built the first prototype WIEVLE and tested it in 2020. From [3], the hydrostatics of the WIEVLE were proven stable through the clever use of ballasting on the vehicles frame. Afterward, the vehicle was subjected to a tethered control testing regimen and was able to rotate and translate underwater while in constant communication with a topside computer. The testing identified the need for a robust, autonomous control capability [3].

B. RESEARCH OBJECTIVE

For the purposes of this research, there is a distinction made between the type of control administered to a UUV. When the UUV is directly controlled by a human, via joystick or topside computer, it will be considered open-loop (OL) control. Inversely there may be situations where a robot is provided only a position, velocity, or orientation command. In this case, when the vehicle independently controls itself and navigates to the commanded position, velocity, or orientation, it will be considered closed-loop (CL)

control. Prior to this thesis, the WIEVLE is considered to have achieved OL control. For the vehicle to achieve autonomy and shift from a UUV to an Autonomous Undersea Vehicle (AUV), a CL scheme was required to be implemented.

Using the previous research as a reference, this thesis seeks to prove the WIEVLE is a controllable rigid body and develop a method of CL control for its position and orientation. This will be achieved using computer modeling, namely the MATLAB/ Simulink computing program.

Chapter II explores the modeling synthesis of the WIEVLE. This was a necessary step to develop the required equations to be instantiated in MATLAB/Simulink. Chapter III then instantiates the equations of Chapter II and simulates an OL control configuration. Chapter IV explains how two types of velocity controllers were iteratively developed and tested. Using the OL control configuration as a backbone, the velocity and eventually position and orientation are fed back to create a CL control configuration. Chapter V describes the efforts to implement the insights provided by Chapters II, III, and IV. Finally, Chapter VI draws conclusions from the research and offers recommendations on future work.

II. DYNAMIC AND KINEMATIC MODELING OF WIEVLE

This chapter will introduce a mathematical model for the WIEVLE. The model will serve as the starting point for simulation of the vehicle in MATLAB/Simulink and provide a greater understanding of the dynamics and overall stability of the WIEVLE.

A. UNIQUE ARCHITECTURE OF THE WIEVLE

The innovative design of WIEVLE provides a stable platform for maritime operations due to the position of its center-of-gravity relative to its center-of-buoyancy. In addition, the novel arrangement of the thrusters also gives the WIEVLE a unique maneuvering capability. The WIEVLE four-thruster propulsion system takes suction from below, through four Blue Robotics T-series thrusters, and ejects it at an equal angle of $45°$ through the four thruster outlets arranged at the top of the vehicle as shown in Figure 1. Figures 2 and 3 show how the thrust vectors T are oriented toward the geometric center C of the vehicle. By concentrating the four thrust vectors toward C, it can be shown that the corresponding body moments (the cross product of the position vector and the thrust force) equals zero. The fixed ballast, shown in Figure 3, ensures the vehicle center-of-gravity (CG) will never overtake its center-of-buoyancy (CB) along the Z-axis, ensuring stability.

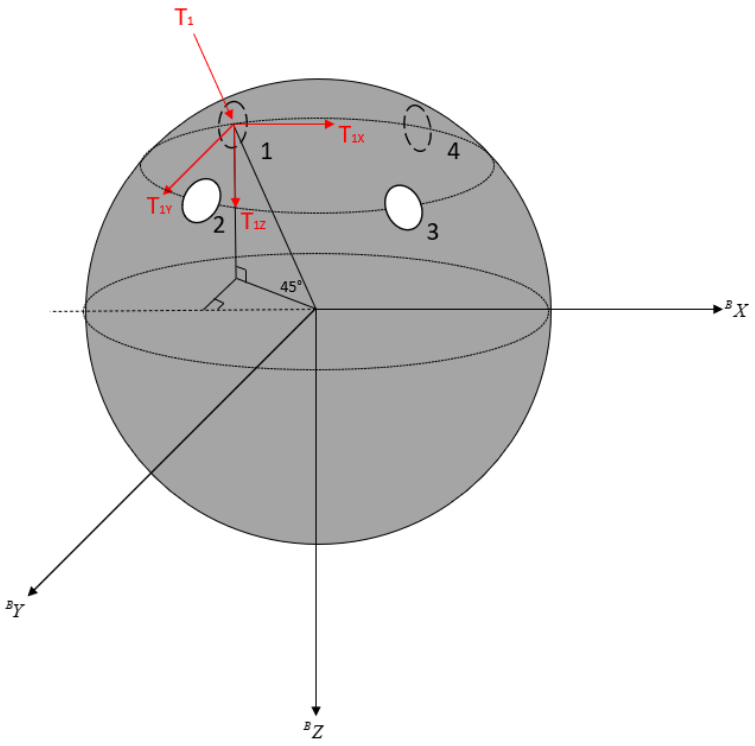


Figure 1. WIEVLE Thrust Vector

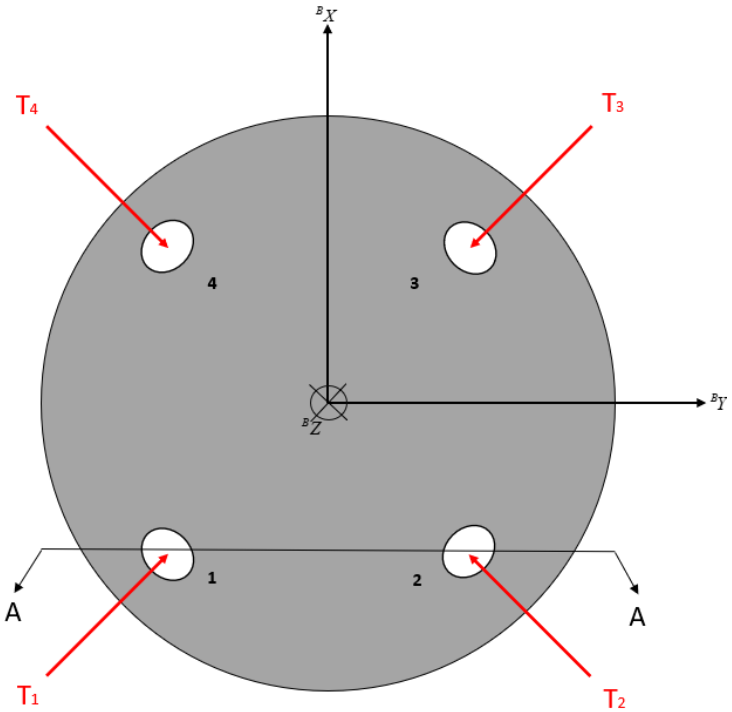


Figure 2. WIEVLE Thrust Vector Top View

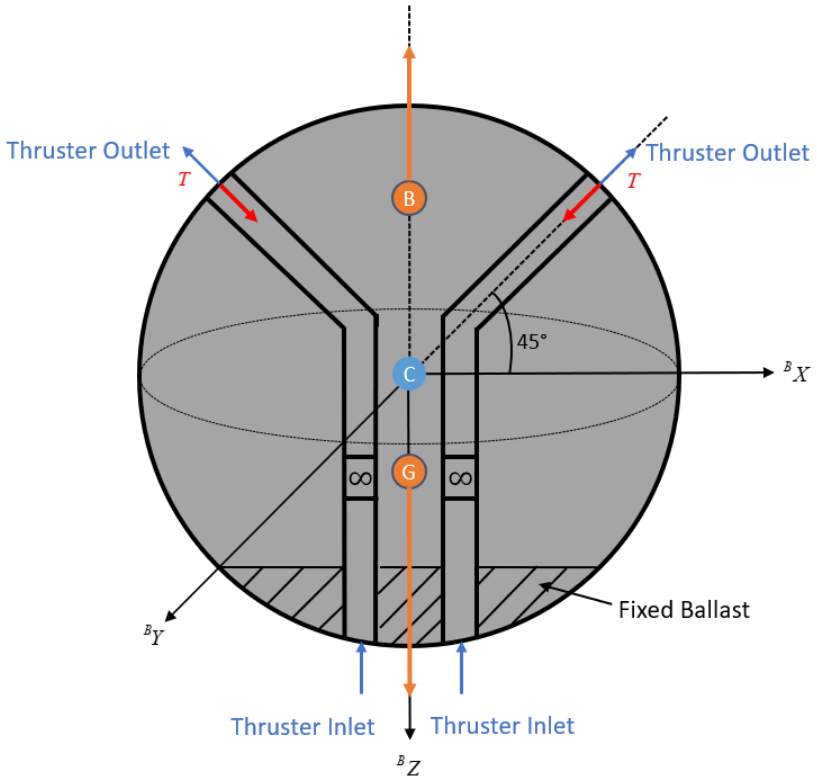


Figure 3. WIEVLE Side View, Section A-A

B. MODELING OF THE WIEVLE

Prior to beginning any physical system computer modeling, it is necessary to apply scientific equations and concepts to obtain constant values, dynamic matrices, and differential equations of motion.

1. Coordinate Frames

"A rigid body is completely described by its position and orientation with respect to a reference frame" [4, p. 23]. The two reference frames utilized for WIEVLE are the earth-fixed {E} and inertial, or body-fixed {B}, reference frames. Figure 4 shows an illustration of the relationship between the two coordinate frames. Notice, as a matter of convention, the axis of the earth-fixed frame is down. The {B} frame is also seen in Figure 1, and it is the reference frame attached to the WIEVLE. Therefore, this frame is free to move and rotate with respect to the {E} frame. Also, the standard for referring to

the translational and rotational movement along and about each axis is also provided by Figure 4.

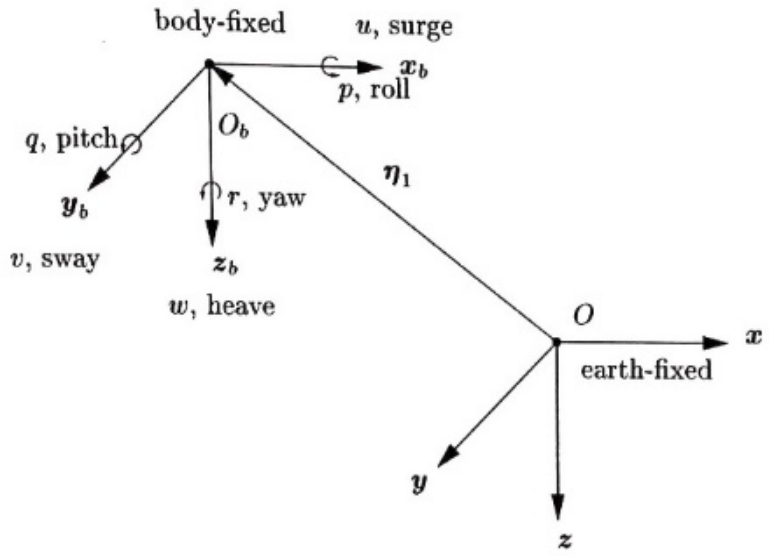


Figure 4. Earth-Fixed and Body-Fixed Coordinate Frames. Source: [4].

2. Degrees of Freedom

"The degrees of freedom (DOF) of maneuvering marine craft are the set of independent displacements and rotations that specify completely the displaced position and orientation of the craft" [5, p. 5]. Figure 4 alludes to the six degrees of freedom that are inherent to body-fixed frame of marine vehicles. Surge, sway, and heave describe the translational movement of a rigid body along its respective X, Y, and Z axes. Roll, pitch, and yaw describe the rotational movement of a rigid body about its X, Y, and Z axes. In the case of the WIEVLE, due to fixed ballasting, the controllable DOF were surge, sway, heave, and yaw, making it a 4 DOF vehicle [3].

3. State Vectors

The physical modeling of the WIEVLE was accomplished in a state-space representation using state vectors ν and η. The first state vector ν is the linear (u, v, w)

and angular (p, q, r) velocities of the WIEVLE in the {B} frame. The second essential state vector was η, which provides the position (x, y, z) and angular orientation (ϕ, θ, ψ) in the {E} frame. Equations (1) and (2) show how the linear and angular velocity, and the position and attitude of the WIEVLE rigid body were represented.

$$M\,\dot{v} + D\,v + g(\eta) = \tau \tag{1}$$

$$\dot{\eta} = J(\eta)\,v \tag{2}$$

Equations (1) and (2) express the dynamic and kinematic equations of motion of the WIEVLE rigid body as it moves through the three-dimensional {E} frame with M being the inertia matrix, consisting of mass and moments of inertia. The matrix D represents the drag terms. When D is multiplied by v it provides drag forces and moments that dampen the motion of the vehicle. The vector $g(\eta)$ has dimensions 6×1 and contains the forces and moments resulting from the buoyant and gravitational forces. On the right side of (1), τ is the vector of forces and moments resulting from the thrusters. $J(\eta)$ is the direction cosine matrix that relates the body frame variables in v to the earth frame quantities we have in the vector η. This system of differential equations, in concert with one another, mathematically represent the translational and rotational movement of the rigid body about the X, Y, and Z axes in the {B} frame. Contained in the {B} frame, the linear and angular velocities are depicted in (3). Equation (4) shows the {E} frame position and attitude of the rigid body.

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} = \begin{bmatrix} x-velocity \\ y-velocity \\ z-velocity \\ roll-velocity \\ pitch-velocity \\ yaw-velocity \end{bmatrix} \tag{3}$$

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} x-position \\ y-position \\ z-position \\ roll-angle \\ pitch-angle \\ yaw-angle \end{bmatrix} \tag{4}$$

Equations (3) and (4) completely describe the state vectors associated with the WIEVLE.

4. Hydrostatic Forces and Moments

The hydrostatic forces were buoyancy and weight. Equations (5) and (6) show how these forces were represented in the earth-fixed frame and acted on the WIEVLE in only the positive and negative Z-axis.

$${}^{E}F_{Buoyancy} = \begin{bmatrix} 0 \\ 0 \\ -\rho g V \end{bmatrix} \tag{5}$$

$${}^{E}F_{Weight} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{6}$$

Where ρ is the water density, g is gravity, V is the volume of fluid displaced by the vehicle, and m is the mass of the vehicle. These forces were then converted to the body-fixed frame through a trivial pre-multiplication operation as shown in (7)-(10). The rotation matrix utilized in these operations was obtained from [4].

$${}^{B}F_{Buoyancy} = {}^{B}_{E}R \cdot {}^{E}F_{Buoyancy} = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\theta\sin\phi \\ \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\theta\cos\phi \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ -\rho g V \end{bmatrix} \tag{7}$$

$$ {}^{B}F_{Buoyancy} = -\rho g V \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} \tag{8} $$

$$ {}^{B}F_{Weight} = {}^{B}_{E}R \cdot {}^{E}F_{Weight} = \begin{bmatrix} \cos\psi \cos\theta & \sin\psi \cos\theta & -\sin\theta \\ \cos\psi \sin\theta \sin\phi - \sin\psi \cos\phi & \sin\psi \sin\theta \sin\phi + \cos\psi \cos\phi & \cos\theta \sin\phi \\ \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi & \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi & \cos\theta \cos\phi \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{9} $$

$$ {}^{B}F_{Weight} = mg \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} \tag{10} $$

The moments were calculated in a similar fashion to the forces and represented by (11)-(14). From Eldred et al. [3], the exact location of CB and CG can be estimated, but not known exactly. This model assumes, for the purpose of simplicity, that CG and C are collocated. This could be achieved through careful ballasting. With the collocation of CG and C, the moment arm due to buoyancy was assumed to act on the top of the spherical hull of the WIEVLE. This assumption is why the radius was used for the moment due to buoyancy in (11).

$$ {}^{b}\vec{r}_{Buoyancy} = \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix} \tag{11} $$

$$ M_{Buoyancy} = {}^{b}\vec{r}_{Buoyancy} \times {}^{b}F_{Buoyancy} = \rho g V r \begin{bmatrix} -\sin\theta \\ \cos\theta \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} \tag{12} $$

$$ {}^{b}\vec{r}_{Weight} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{13} $$

$$ M_{Weight} = {}^{b}\vec{r}_{Weight} \times {}^{b}F_{Weight} = 0 \tag{14} $$

Keeping the assumption that CG and C are collocated, it is intuitive that the moment due to weight is zero, as shown in (14).

5. Drag Forces and Moments

For a sphere moving in liquid, the drag can be modeled as equation 15 [4].

$$F_{Drag} = -\frac{1}{2}\rho C_D V A_S \left(R_N\right) \tag{15}$$

With C_D being the drag coefficient, A_S is the vehicle surface area, V is the (u, v, w) directional velocities, and R_N is the Reynolds number. In the WIEVLE case, R_N was estimated to be 1.0 due to its small size and spherical shape.

The drag moment therefore was modeled assuming the moment is produced by skin drag on the surface of the vehicle. Equation (16) represents this phenomenon.

$$M_{Drag} = \frac{1}{2}\rho C_D \left(r \cdot \omega\right)^2 A_S r \tag{16}$$

C_D being the drag coefficient different than the drag force coefficient, ω is the rotational velocity rates (p, q, r), and r represents the radius of the vehicle.

6. Thruster Forces and Moments

Figure 3 depicts the assumption that each thruster force is exerted at its respective exit nozzle and angled toward the geometric center of the vehicle. As such, (17) mathematically represents the sum of the depicted thruster forces, while the other thrusters will mimic this behavior as force is applied.

$$\vec{T}_1 = \vec{T}_{1X} + \vec{T}_{1Y} + \vec{T}_{1Z} \tag{17}$$

These axes-specific force vectors are further represented by (18) and (19).

$$\vec{T}_{1X/Y} = \left|\vec{T}_1\right| \cdot \cos\left(45^\circ\right) \cdot \cos\left(45^\circ\right) = \left|\vec{T}_1\right| \cdot \frac{\sqrt{2}}{2} \cdot \frac{\sqrt{2}}{2} = \frac{1}{2} \cdot \left|\vec{T}_1\right| \tag{18}$$

$$\vec{T}_{1Z} = \left|\vec{T}_1\right| \cdot \sin\left(45^\circ\right) = \frac{\sqrt{2}}{2} \cdot \left|\vec{T}_1\right| \tag{19}$$

Using Figures 1 and 2 as a reference, the force summation for thrusters 2–4 can be easily solved using similar methods to (18) and (19). Like the thruster force vectors, the thruster moments are calculated using trigonometry. Equations (20)-(27) display the summary of the calculations.

$$\vec{T}_1 = \left|T_1\right| \cdot \left(\frac{1}{2}\hat{x} + \frac{1}{2}\hat{y} + \frac{\sqrt{2}}{2}\hat{z}\right) \tag{20}$$

$$\vec{T}_2 = \left|T_2\right| \cdot \left(\frac{1}{2}\hat{x} - \frac{1}{2}\hat{y} + \frac{\sqrt{2}}{2}\hat{z}\right) \tag{21}$$

$$\vec{T}_3 = \left|T_3\right| \cdot \left(-\frac{1}{2}\hat{x} - \frac{1}{2}\hat{y} + \frac{\sqrt{2}}{2}\hat{z}\right) \tag{22}$$

$$\vec{T}_4 = \left|T_4\right| \cdot \left(-\frac{1}{2}\hat{x} + \frac{1}{2}\hat{y} + \frac{\sqrt{2}}{2}\hat{z}\right) \tag{23}$$

$$\vec{r}_1 = r \cdot \left(-\frac{1}{2}\hat{x} - \frac{1}{2}\hat{y} - \frac{\sqrt{2}}{2}\hat{z}\right) \tag{24}$$

$$\vec{r}_2 = r \cdot \left(-\frac{1}{2}\hat{x} + \frac{1}{2}\hat{y} - \frac{\sqrt{2}}{2}\hat{z}\right) \tag{25}$$

$$\vec{r}_3 = r \cdot \left(\frac{1}{2}\hat{x} + \frac{1}{2}\hat{y} - \frac{\sqrt{2}}{2}\hat{z}\right) \tag{26}$$

$$\vec{r}_4 = r \cdot \left(\frac{1}{2}\hat{x} - \frac{1}{2}\hat{y} - \frac{\sqrt{2}}{2}\hat{z} \right) \tag{27}$$

Equation (28) shows the cross product, ensuring stability and balance.

$$\vec{r} \times \vec{T} = 0 \tag{28}$$

7. Constant Value Knowns and Assumptions

The following assumptions were either determined by this research or previous research contained in [3].

Table 1. WIEVLE Assumptions. Adapted from [3], [6].

Assumption (Equation)	Value	Units
Mass	2.26796	kg
Diameter	30	cm
Radius	15	cm
CG	[0 0 0]	m
CB	[0 0 - *radius*]	m
Volume ($\frac{4}{3} \cdot \pi \cdot r^3$)	0.141	m^3
Surface Area ($4 \cdot \pi \cdot r^2$)	0.2827	m^2
Inertia ($mass \cdot r^2$)	0.2041	$kg \cdot m^2$
Drag Force Coefficient (CdF)	0.5	-
Drag Moment Coefficient (CdM)	10	-
Density of Water (ρ)	997	kg/m^2
T100 100% FWD Thrust ($T_{FWD,100}$)	53.2030	N
T100 100 % REV Thrust ($T_{REV,100}$)	-17.8487	N

In addition to Table 1, assumptions were made regarding the known sensor data. This includes position, orientation, velocity, and velocity rates being known and available to the controller feedback loop. These assumptions were simple in the MATLAB and Simulink environment but prove difficult to implement in the physical world.

From the above calculated, observed, and computed assumptions and equations, the logical next step was to create a computer model. Simulating the model allowed the research to validate the assumptions and equations, a necessary step prior to any controller implementation. MATLAB/Simulink was chosen for modeling due to its ease of use and familiarity. The next two chapters discuss these concepts being put into practice and the resulting simulations.

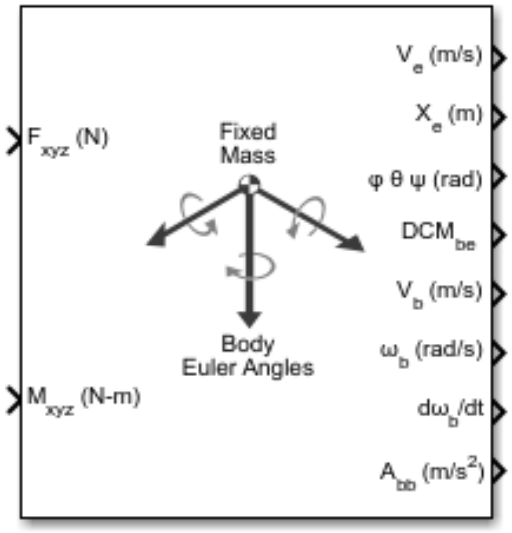
THIS PAGE INTENTIONALLY LEFT BLANK

III. OPEN LOOP COMPUTER MODELING AND RESULTS

This chapter explains the mathematical representation of WIEVLE in the MATLAB/Simulink environment. The model is controlled in an open-loop configuration using the MATLAB code contained in Appendix A and verified for accuracy and stability.

A. SIMULINK BLOCK 6 DOF (EULER ANGLES)

The 6 DOF block, in the Simulink Aerospace blockset, implements Euler angle representation of the six DOF equations of motion, with respect to the rotation of a body-fixed coordinate frame about the earth-fixed coordinate frame. Conveniently, the system of dynamic/kinematic equations for aerospace vehicles have the same form as (1) and (2), which describe the WIEVLE UUV. Figure 5 shows the DOF Simulink block. The inputs to the block include the forces $F_{xyz}\left(N\right)$ and moments $M_{xyz}\left(N-m\right)$ applied to the rigid body [7]. The outputs are of the velocity $V_e\left(m/s\right)$ and position $X_e\left(m\right)$ in {E}, the Euler rotation angles $\varphi\,\theta\,\psi\left(rad\right)$, the coordinate transformation matrix of the body between {B} and {E} DCM_{be}, the velocity $V_b\left(m/s\right)$ and angular rates $\omega_b\left(m/s\right)$ in {B}, the angular accelerations $\frac{d\omega_b}{dt}$ in {B}, and the accelerations $A_{bb}\left(m/s^2\right)$ with respect to {E} [7]. The use of the 6 DOF block, with respect to this model, allowed for real-time solutions to (1) and (2).



Source: MathWorks (2006)

Figure 5. SIMULINK 6 DOF Euler Angles Block

A constraint of the Simulink block is that it assumes the applied forces act at the CG of the body [7]. As described previously, CG and C were collocated to reconcile the uncertainty of the CG location. The implication was that the modeled vehicle would not exhibit pitch or roll, as expected, as its thrusters were turned on.

B. WIEVLE REPRESENTATION IN MATLAB/SIMULINK

Figure 6 shows WIEVLE in the SIMULINK environment. The thrust commands are delivered via the MATLAB code contained in Appendix A and sent to the Simulink thruster subsystem to produce moments and forces due to thruster action. Figures 7 and 8 show the instantiation of (20)-(28) to produce the thruster forces and moments. Simultaneously, Figures 9 and 10 show the forces and moments produced by hydrostatics and drag, instantiating (11)-(16). To account for the ability of the vehicle to rotate freely about the Z-axis without creating any unwanted X or Y translation, it was necessary to create a moment that pairs opposing thruster commands and coordinates opposing thrusters to produce a yaw action. Figure 8 shows the method chosen to accomplish the yaw action. All the forces and moments are then added and inputted to the 6 DOF (Euler Angles) block. From the 6 DOF block, the {E} frame position, orientation, and velocity

of the WIEVLE model were obtained for the timed duration of each simulation. The outputs of the 6 DOF block were then exported to the MATLAB workspace and plotted using the code contained in Appendix B. The plots were then used to ensure the model responded appropriately to the parameters of each simulation.

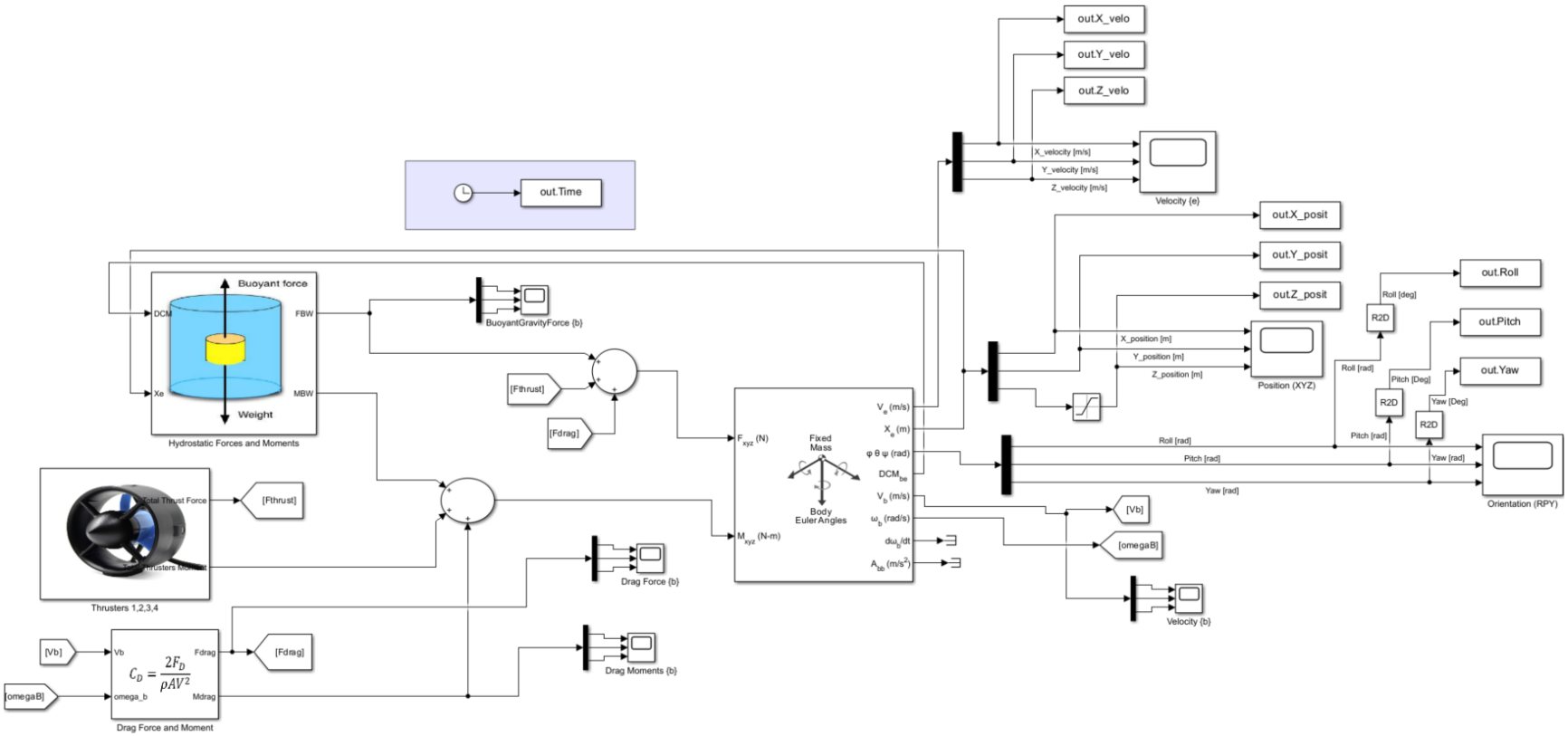


Figure 6. WIEVLE System Representation in Simulink

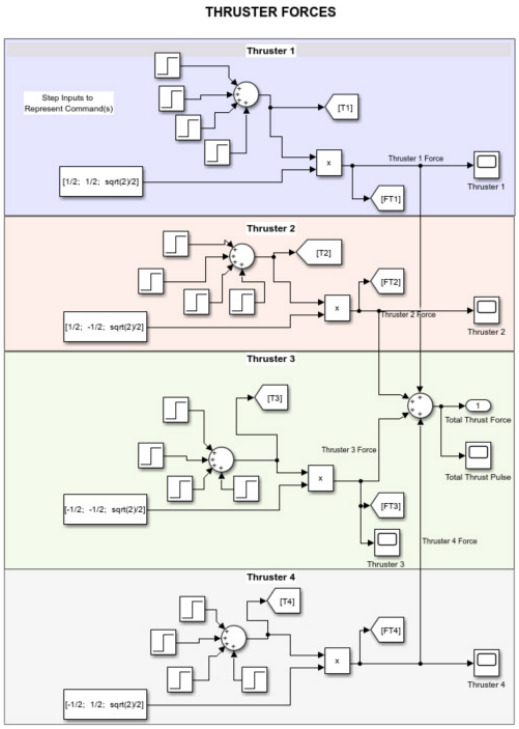


Figure 7. Thruster Forces Subsystem

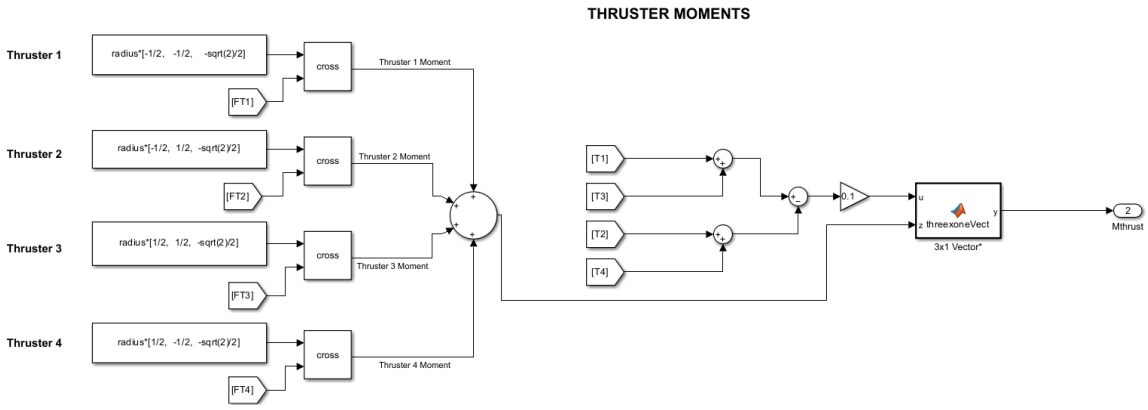


Figure 8. Thruster Moments Subsystem

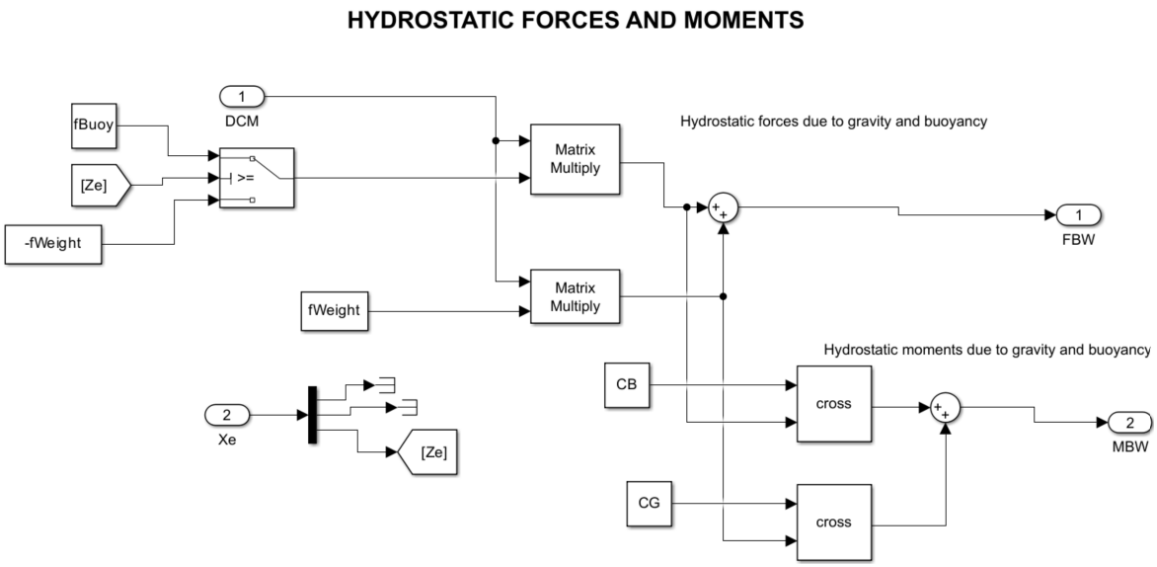


Figure 9. Hydrostatic Forces and Moments Subsystem

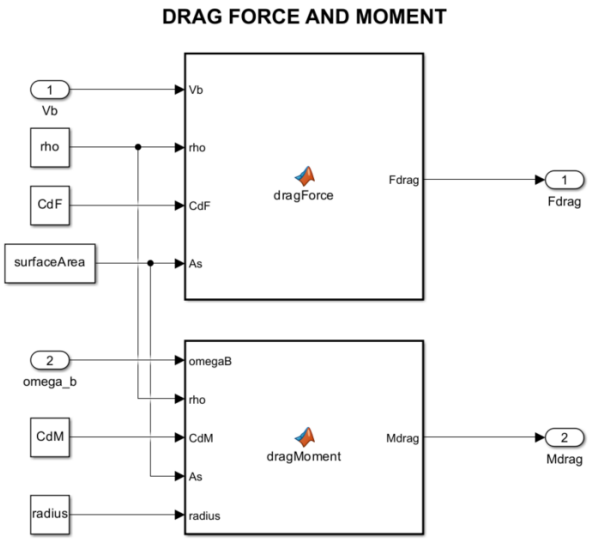
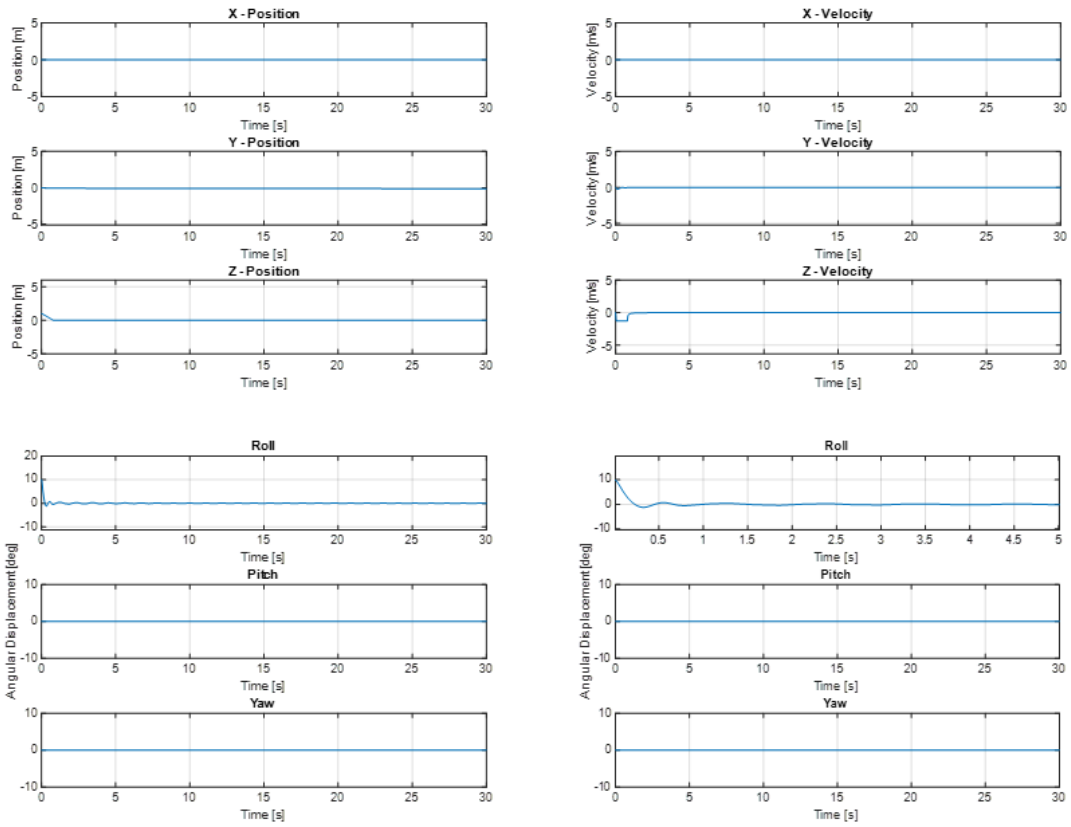


Figure 10. Drag Forces and Moments Subsystem

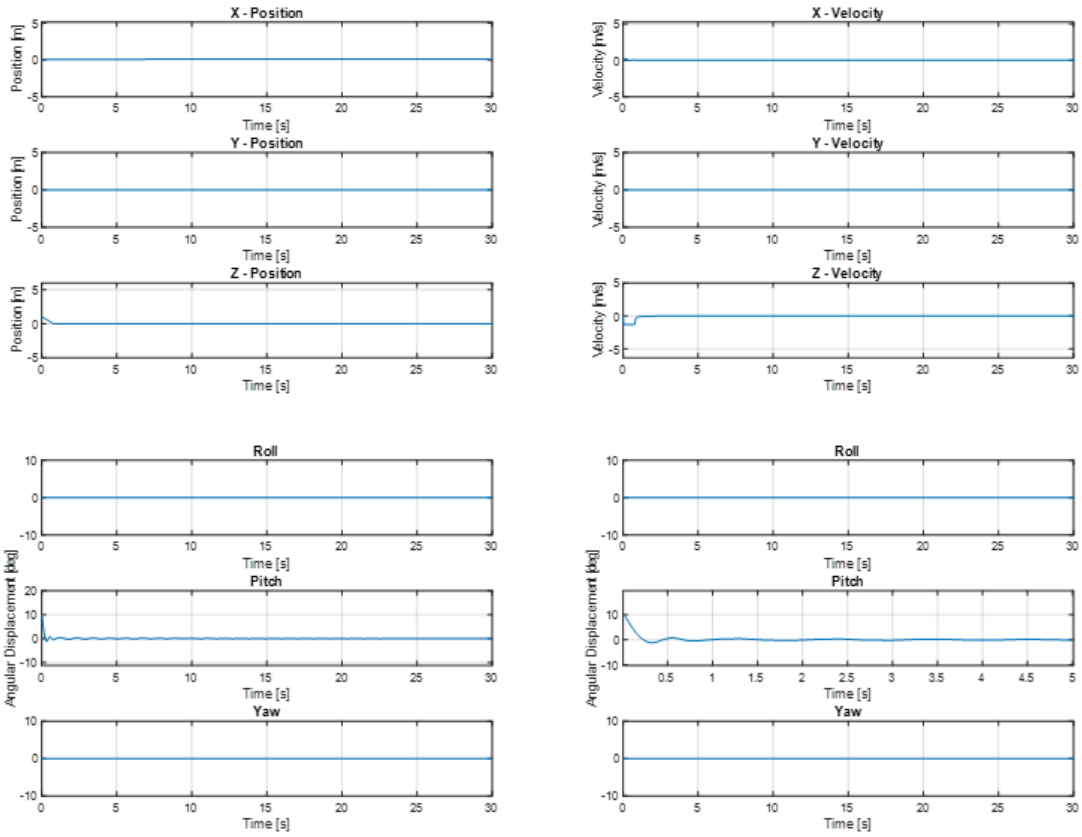
C. STATIC STABILITY TESTING

Prior to testing the model with any movement, it was necessary to test the static stability of the vehicle. Figures 11–13 show the static response of the model to an initial roll, pitch, and yaw angle. Each simulation began with an initial position of $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ m.



The bottom right plot shows the roll transient from 0–5 s.

Figure 11. Static Response to Initial Roll Angle of 10°



The bottom right plot shows the pitch transient from 0–5 s.

Figure 12. Static Response to Initial Pitch Angle of 10°

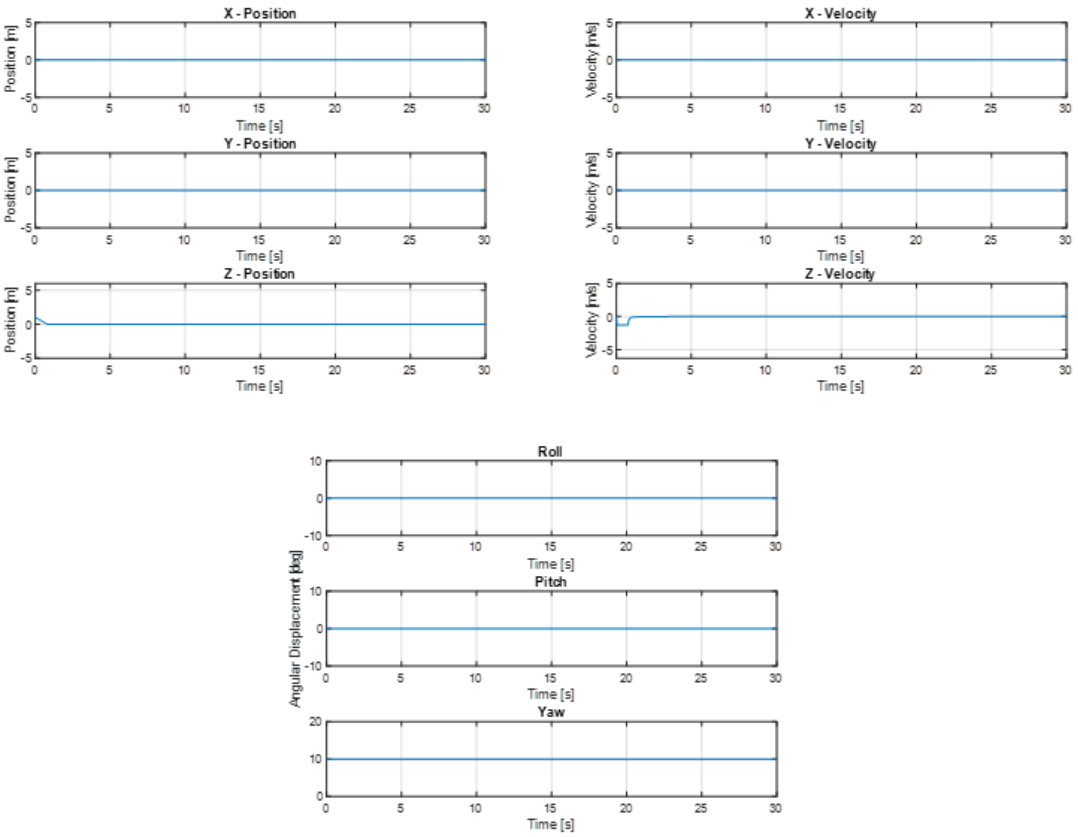


Figure 13. Static Response to Initial Yaw Angle of 10°

Chapter II asserted the notion of the CG not overtaking the CB to ensure stability of the vehicle. The observations of this section prove this notion. Each of the simulations showed the modeled vehicle returning to equilibrium after an initial displacement without the assistance of the thrusters. The pitch and roll angles settle down to 0° within 2 s. This is a desirable operational feature for the WIEVLE to have. The yaw angle is neutrally stable and remains fixed on the heading it has been programmed to at the start of the simulation. This ensures that the vehicle will not depart from its programmed heading due to any undesirable vehicle dynamics. The vehicle reached the surface of the water (0 m in Z axis) followed by the Z axis velocity oscillating in all three simulations. The model showing the vehicle rising to the surface is rooted in the slightly positive buoyancy of the WIEVLE, allowing it to rise with no counter force enacted by the thrusters. The Z axis velocity oscillation, after the vehicle has reached the surface of the water, is caused by a saturation block not allowing the model to travel further in the negative z-axis. This

saturation was necessary as the actual WIEVLE would not continue to rise into the air after reaching the surface of the water. In this case, the saturation block acted as the damping force that did not allow the vehicle to get airborne.

D. BALANCED THRUST COMMANDS

To match the natural buoyancy force attempting to push the WIEVLE toward the surface, an equal but opposite force must be created by the WIEVLE thrusters. The constant values determining the buoyancy force do not change, so determining the individual thruster commands was trivial. Equation (29) shows the derivation of an individual thruster force command to counteract the force of buoyancy. Of note, the command was multiplied by a gain of 1.1928 after it was observed the depth hold force was inadequate to counter the buoyancy force during simulation.

$$Static\ Depth\ Hold = 1.1928 \cdot \frac{|F_{BUOYANCY}|}{4} = 1.1928 \cdot \left(\frac{142.2914\ N}{4} \right) = 42.4313\ N \quad (29)$$

If the Static Depth Hold command is given to all four thrusters, the resultant force created should match the buoyancy force and allow the WIEVLE to remain at equilibrium. To create movement about any axis from this equilibrium state, balanced thruster commands are required. Balanced thruster commands are defined as "a proportional decrease in thrust for the thrusters directly opposite the pair in which thrust is increased" [3, p. 10]. Equations (30) and (31) show how this balance was calculated and achieved.

$$T_{Increase} = T_{FWD,100} - Static\ Depth\ Hold = 53.2030\ N - 42.4313\ N = 10.7717 N \quad (30)$$

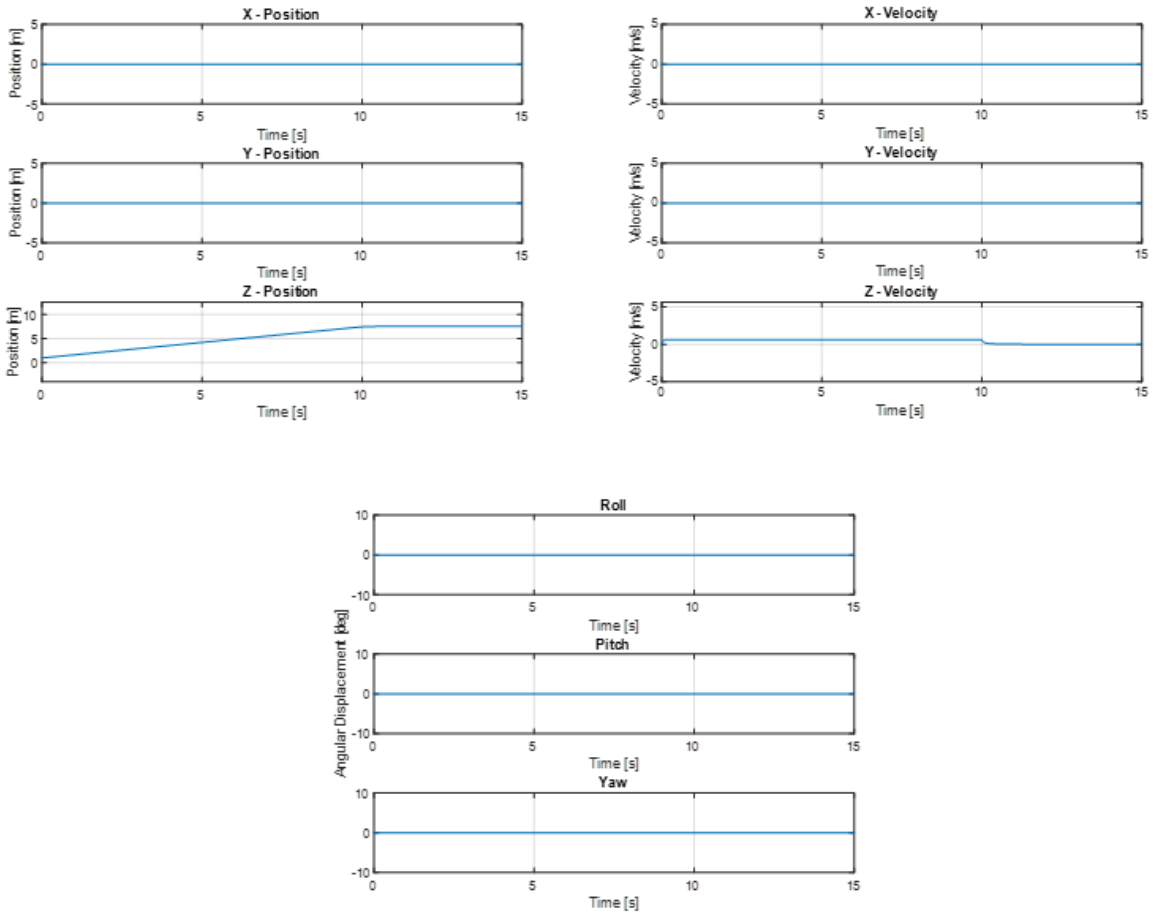
$$T_{Decrease} = Static\ Depth\ Hold - T_{Increase} = 42.4313\ N - 10.7717\ N = 31.6596\ N \quad (31)$$

Equation (30) shows that if two thruster pairs are given a $T_{FWD,100}$ command, the increase in thrust is 10.7717 N. This means the opposing two thrusters must be provided

commands of 31.6596 N to balance the thruster commands and keep the WIEVLE at equilibrium.

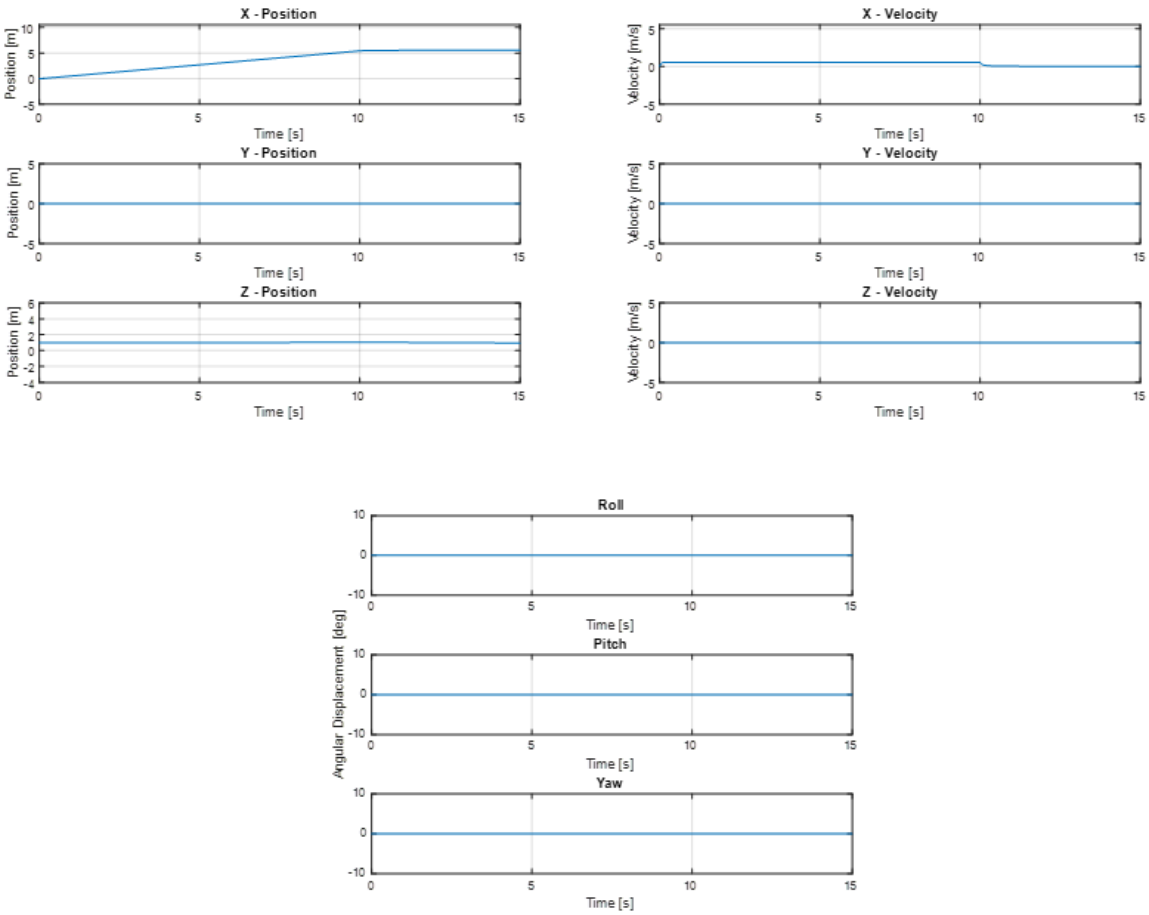
E. TRANSLATION AND ROTATION TESTING

To validate the model from a dynamic standpoint, basic translation commands were inputted to the model with the outputs observed. Figures 14–18 show the response of the model to translation commands. Each simulation began with the same initial position of $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ m with the duration of thruster pulse from 0–10 s. Each figure title indicates the paired thruster commands for each simulation. Also, the title indicates the expected outcome of the simulation. For simplicity in this subchapter, each WIEVLE thruster will be given one of two commands: *On 100%*, or *On depth hold*. The *On 100%* command refers to $T_{FWD,100}$, while the *On depth hold* command refers to $T_{Decrease}$ from Table 1.



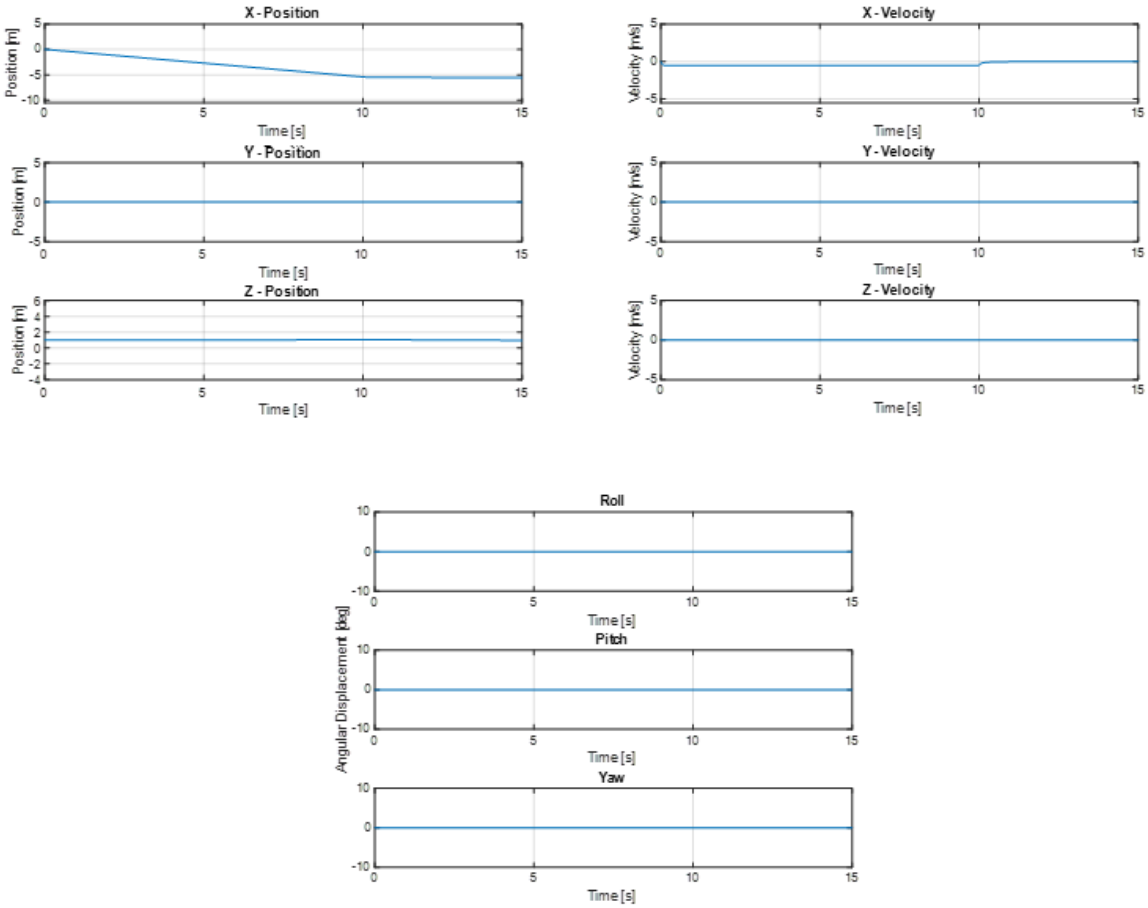
The expected result is a translation along the positive Z axis followed by a depth hold.

Figure 14. All Four Thrusters *On 100%*



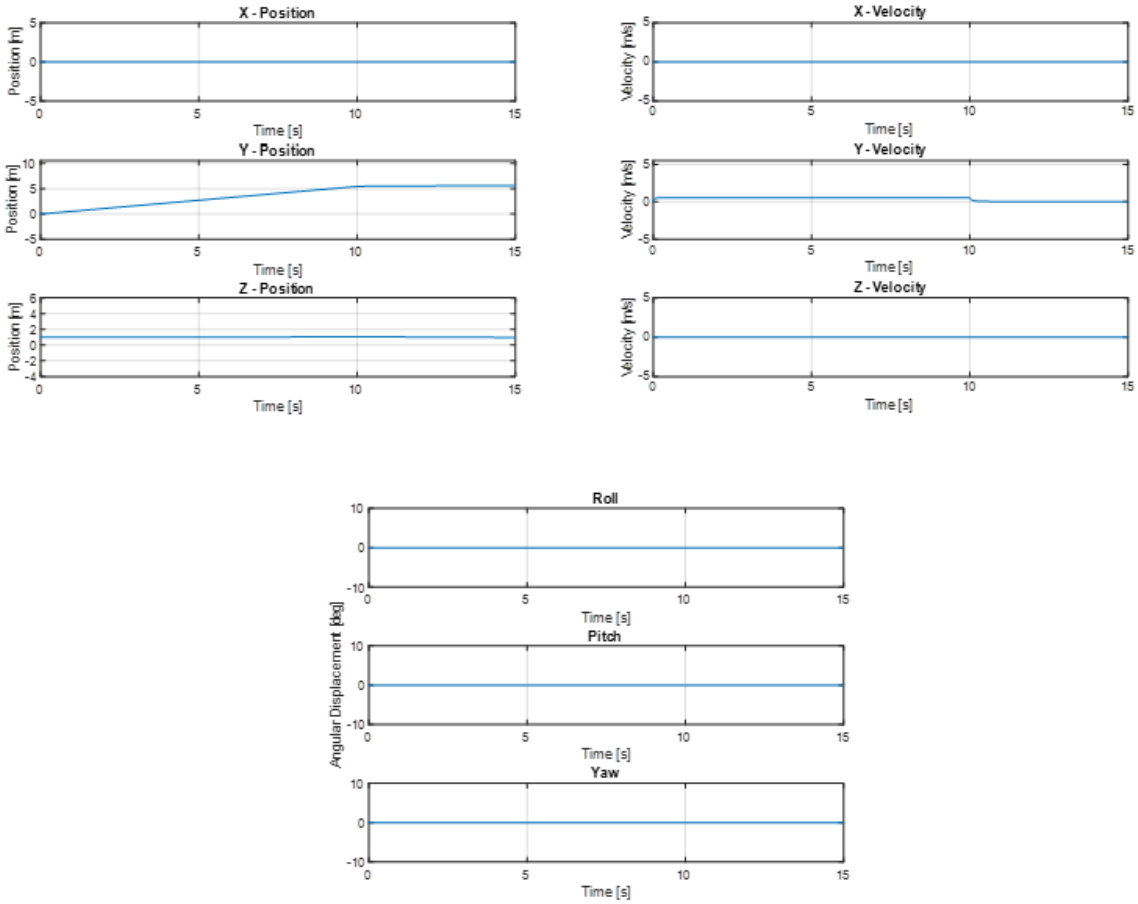
The expected result is a translation along the positive X axis at a constant depth.

Figure 15. T1 and T2 *On 100%*, T3 and T4 *On Depth Hold*



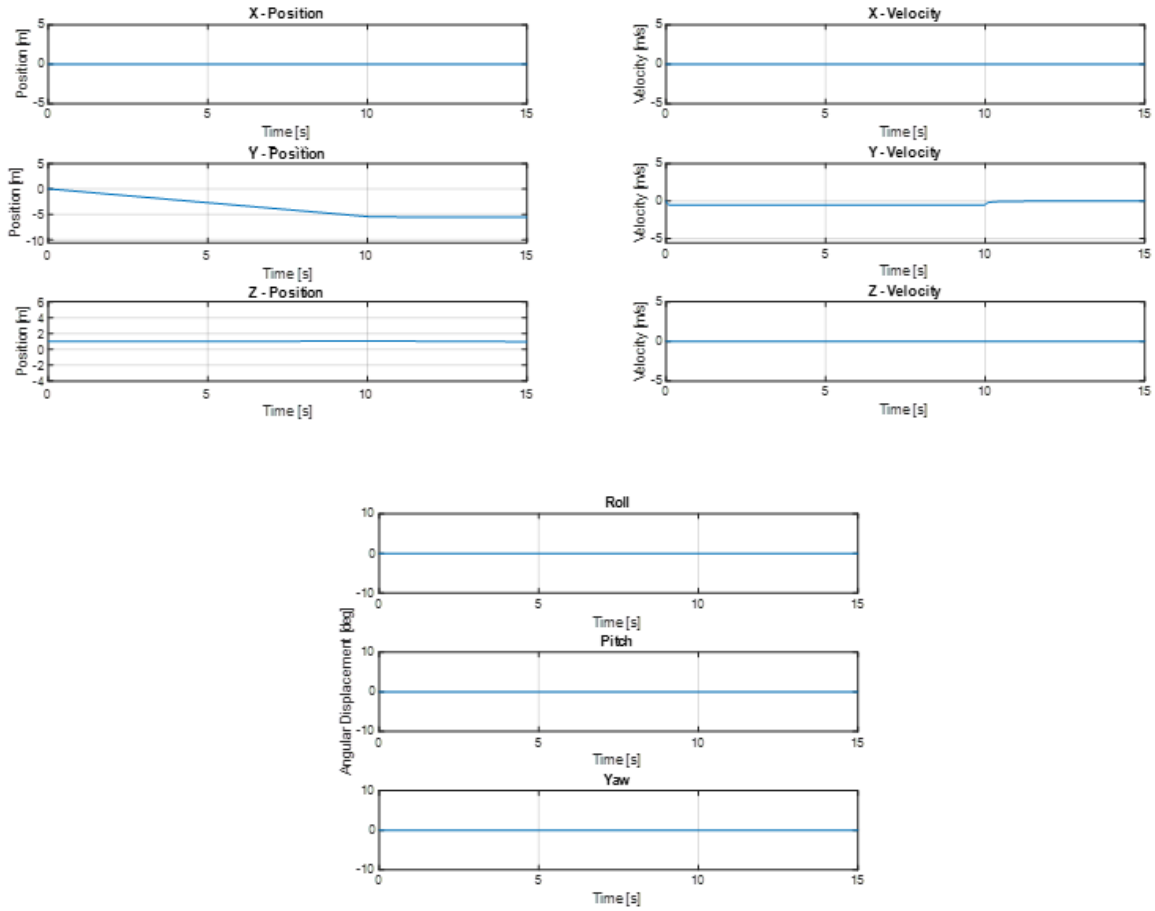
The expected result is a translation along the negative X axis at a constant depth.

Figure 16. T3 and T4 *On 100%*, T1 and T2 *On Depth Hold*



The expected result is a translation along the positive Y axis at a constant depth.

Figure 17. T1 and T4 *On 100%*, T2 and T3 *On Depth Hold*

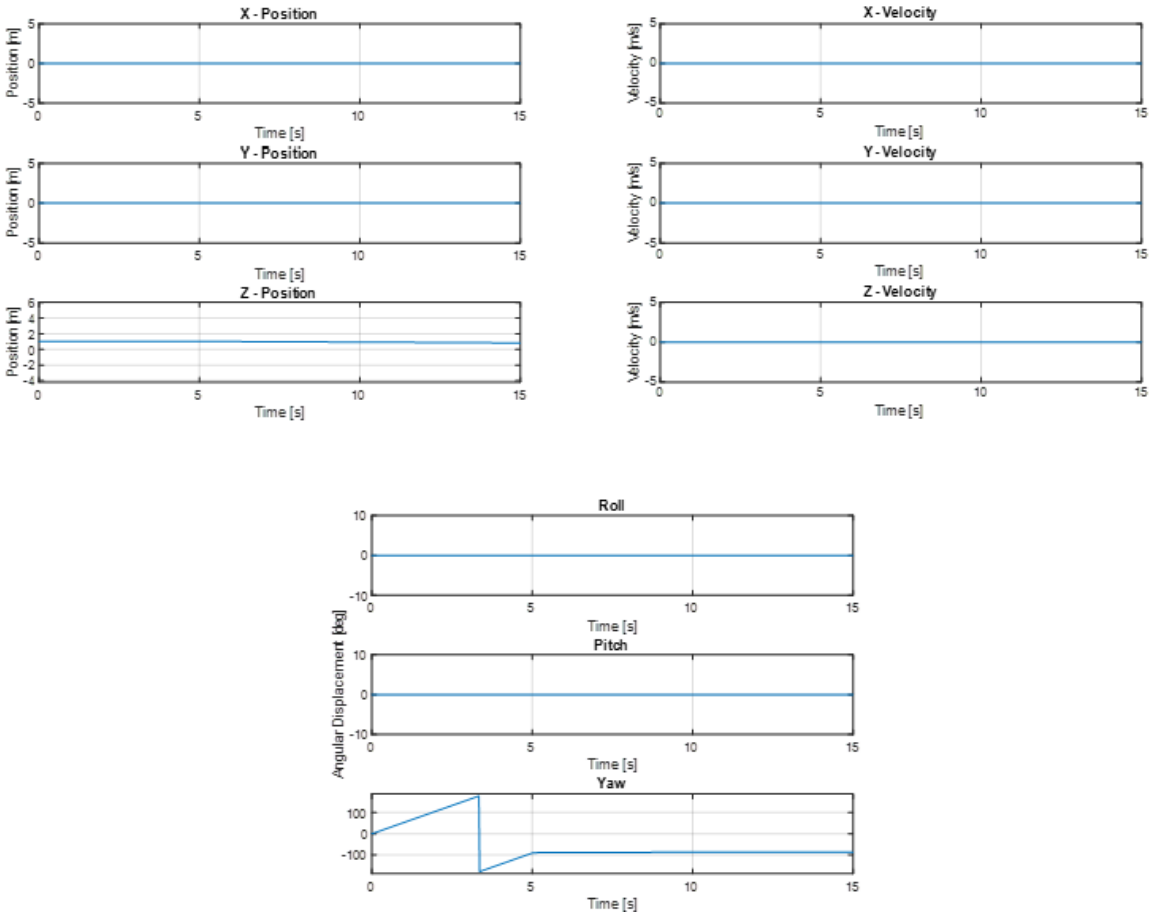


The expected result is a translation along the negative Y axis at a constant depth.

Figure 18. T2 and T3 *On 100%*, T1 and T4 *On Depth Hold*

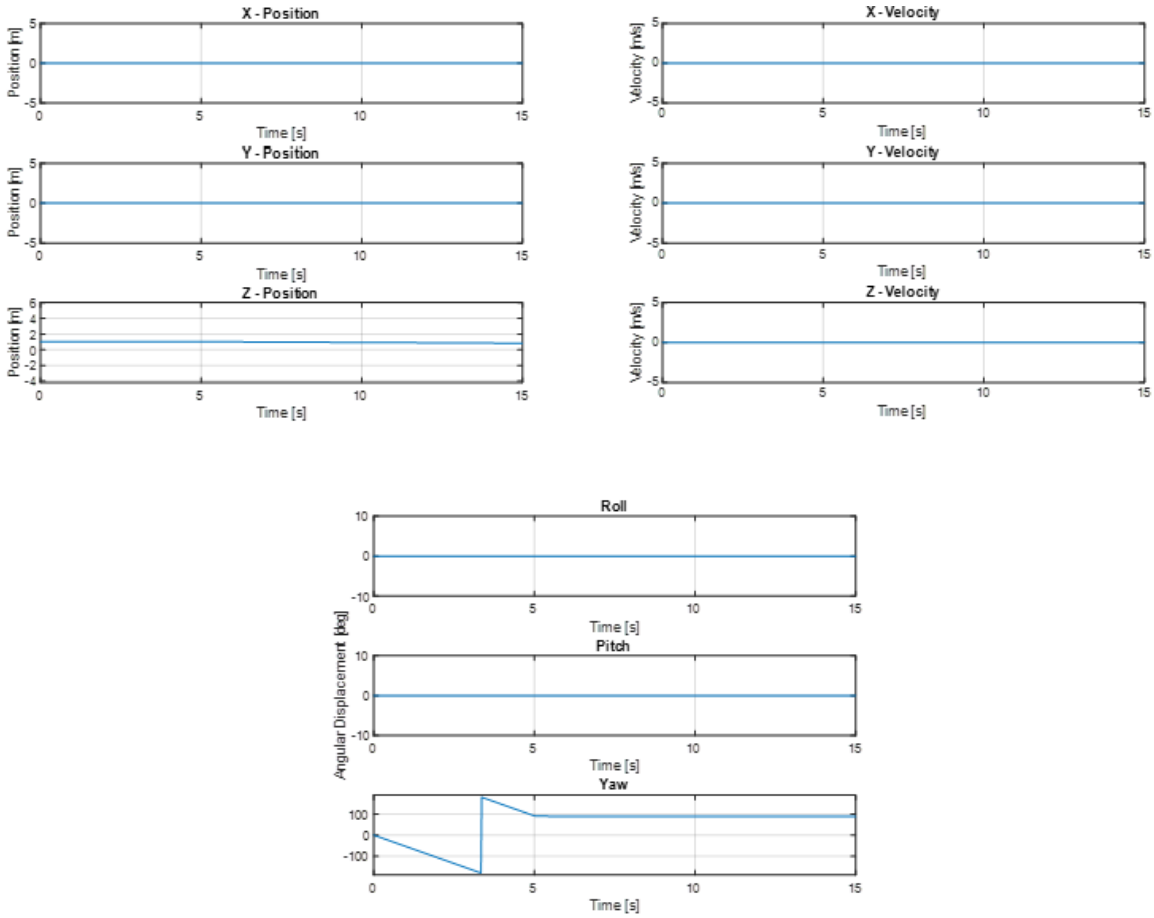
The non-translation axes velocity waveforms were observed to have noticeable oscillation. This was accounted for in the numerical error associated with the differential equation solver provided by MATLAB. All available solvers were tested with the model and no appreciable difference was observed. As the oscillations were consistent across all simulations, it was deemed acceptable for the purposes of this research.

Once translation along the x, y, and z axes were validated, the ability of the model to rotate about the z-axis was tested. Figures 19 and 20 show the model rotating about the z-axis in both the clockwise (positive) and counterclockwise (negative) directions.



The expected result is a rotation about the Z axis (positive yaw) at a constant depth.

Figure 19. T1 and T3 *On 100%*, T2 and T4 *On Depth Hold*



The expected result is a rotation about the Z axis (negative yaw) at a constant depth.

Figure 20. T2 and T4 *On 100%*, T1 and T3 *On Depth Hold*

From the exhibition of the four controllable DOF of the model, the open loop model was considered verified and valid. The logical next step in development was to close the control loop and gain autonomy. The next chapter will discuss the formulation of two types of closed loop controller models.

IV. CLOSED LOOP MODELING AND RESULTS

This chapter explains the formulation of two closed loop controllers to model the WIEVLE. The first, a velocity controller, was a necessary building block. The second position and orientation controller utilized the velocity controller backbone to achieve closed loop feedback control of the WIEVLE model.

A. CLOSED LOOP VELOCITY CONTROLLER SYNTHESIS

Appendix C contains the initialization file code associated with both CL controllers. It is the same as the OL controller without the individual thruster command variables. For the velocity controller, the desired axis-specific translational and yaw velocity command(s) were inputted through the Simulink constant block vice the MATLAB code. This was done for ease-of-use considerations, allowing the user to work only in Simulink. Further, the output plot file used was like the OL controller, with an extra plot added, and is contained in Appendix D. Figure 21 shows the control level of the velocity controller. The axis-specific and yaw velocity command(s) are provided by the constant blocks. These commands are then compared to the feedback velocity in each respective axis to create an error signal that is multiplied by the Proportional, Integral, and Derivative (PID) gains to generate a torque command, shown in Figure 22. Equations (32) and (33) depict the formulation of the torque command with respect to the X-position error signal, $e_u(t)$. For the Y and Z position errors and yaw angle error, the formulation is similar and depicted in Figure 22.

$$e_u(t) = u_{CMD} - u_{ref} \tag{32}$$

$$\tau_1 = K_{p\_u} \cdot e_u(t) + K_{i\_u} \cdot \int e_u(t)\, dt + K_{d\_u} \cdot \frac{d\, e_u(t)}{dt} \tag{33}$$

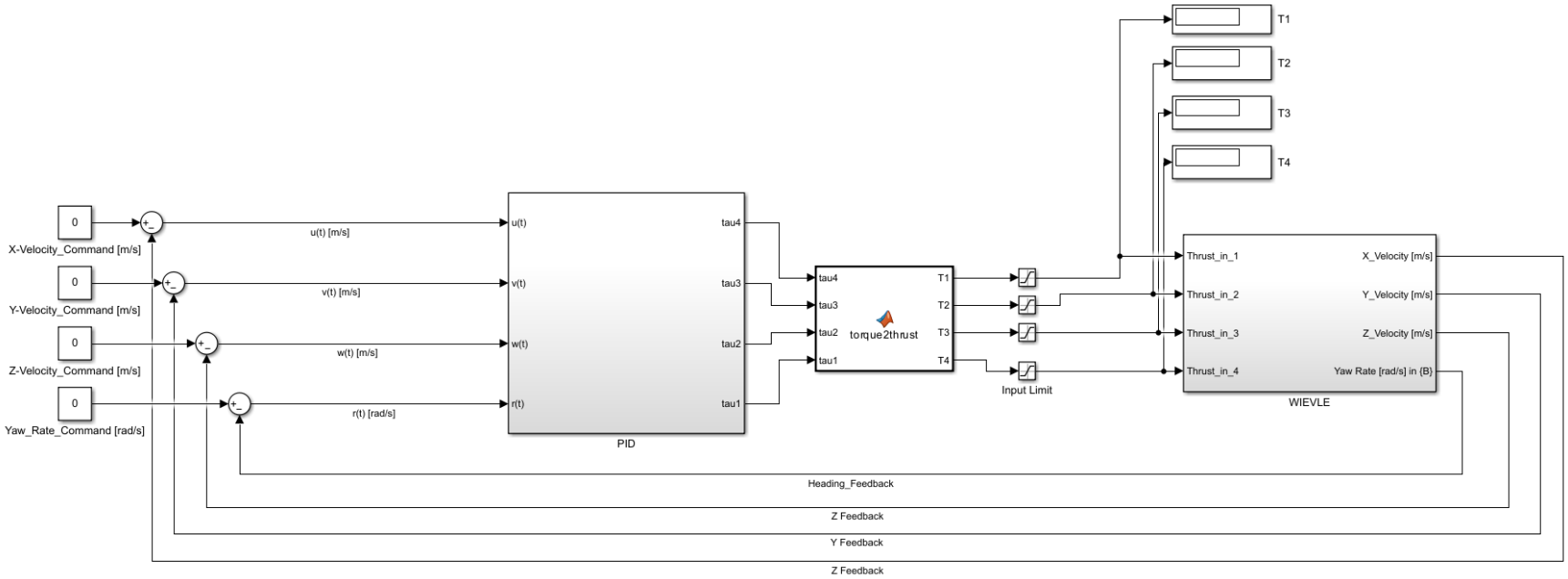


Figure 21. Velocity Controller Model Control-Level

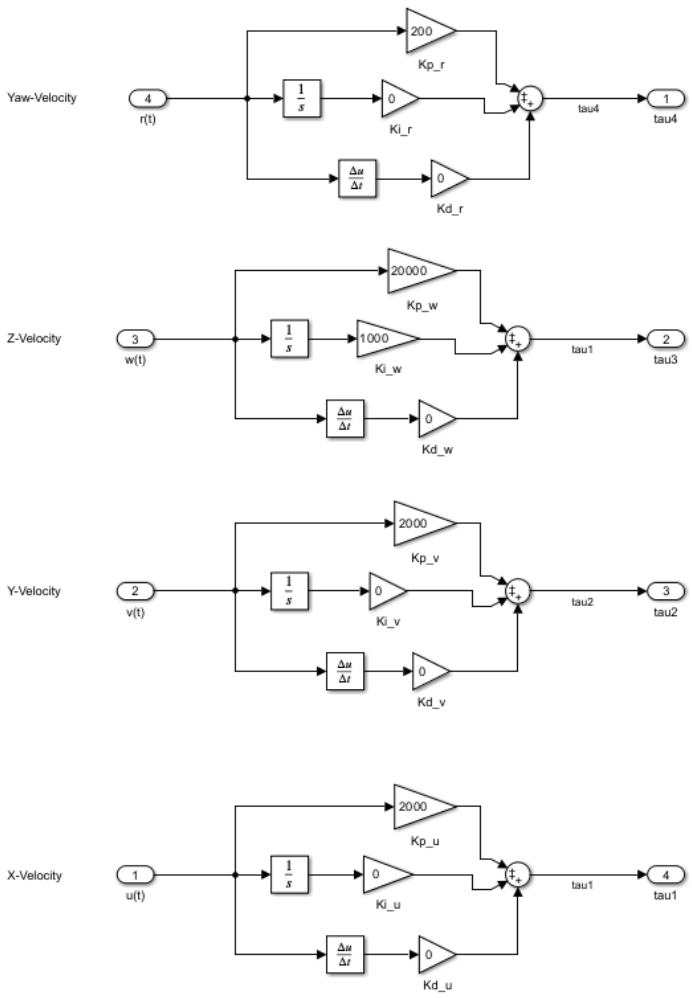


Figure 22. PID Gain Block of Velocity Controller

The compilation of these torque commands represents the control law for this feedback controller. Once the control law was formulated, the next step was to convert the torque command to useful thruster commands for the WIEVLE vehicle model. Figure 23 shows the MATLAB code contained in the *torque2thrust* block. This block takes the torque inputs and solves (34)-(37) to produce the four thruster commands. Those thruster commands are then saturated to the maximum and minimum of $T_{FWD,100}$ and $T_{REV,100}$ explained in Chapter II.

$$\tau_1 = \frac{1}{2}(T_1 + T_2) - \frac{1}{2}(T_3 + T_4) \tag{34}$$

$$\tau_2 = \frac{1}{2}(T_1 + T_4) - \frac{1}{2}(T_2 + T_3) \tag{35}$$

$$\tau_3 = \frac{\sqrt{2}}{2}(T_1 + T_2 + T_3 + T_4) \tag{36}$$

$$\tau_4 = 0.1\left[(T_1 + T_3) - (T_2 + T_4)\right] \tag{37}$$

```
function [T1,T2,T3,T4] = torque2thrust(tau4,tau3,tau2,tau1)

s = sqrt(2)/2;

A = [1/2,1/2,-1/2,-1/2;
     1/2,-1/2,-1/2,1/2;
     s,s,s,s;
     0.1,-0.1,0.1,-0.1];

T = [tau1;tau2;tau3;tau4];

t = A\T;

if t(1)<0
    t(1) = 0;
end
if t(2)<0
    t(2)=0;
end
if t(3)<0
    t(3)=0;
end
if t(4)<0
    t(4)=0;
end

T1 = t(1);
T2 = t(2);
T3 = t(3);
T4 = t(4);
```

Figure 23. *Torque2Thrust* Block MATLAB Code

In (37), the 0.1 value was chosen as an arbitrary constant gain. Of note, it was observed that occasionally the thruster commands produced by the *torque2thrust* block were inverting signs from positive to negative, and vice-versa. To prevent this, the code includes the *if* statements.

Figure 24 shows the WIEVLE subsystem of the velocity controller model. With a few key differences, it is very similar to the OL model. Figures 25 and 26 show the

thruster forces and moments. As opposed to issuing thruster commands from the MATLAB workspace, via step functions, the commands are fed directly from the controller-level in Figure 21. Other than the noted changes, the WIEVLE representation remains unchanged between the OL to CL models.

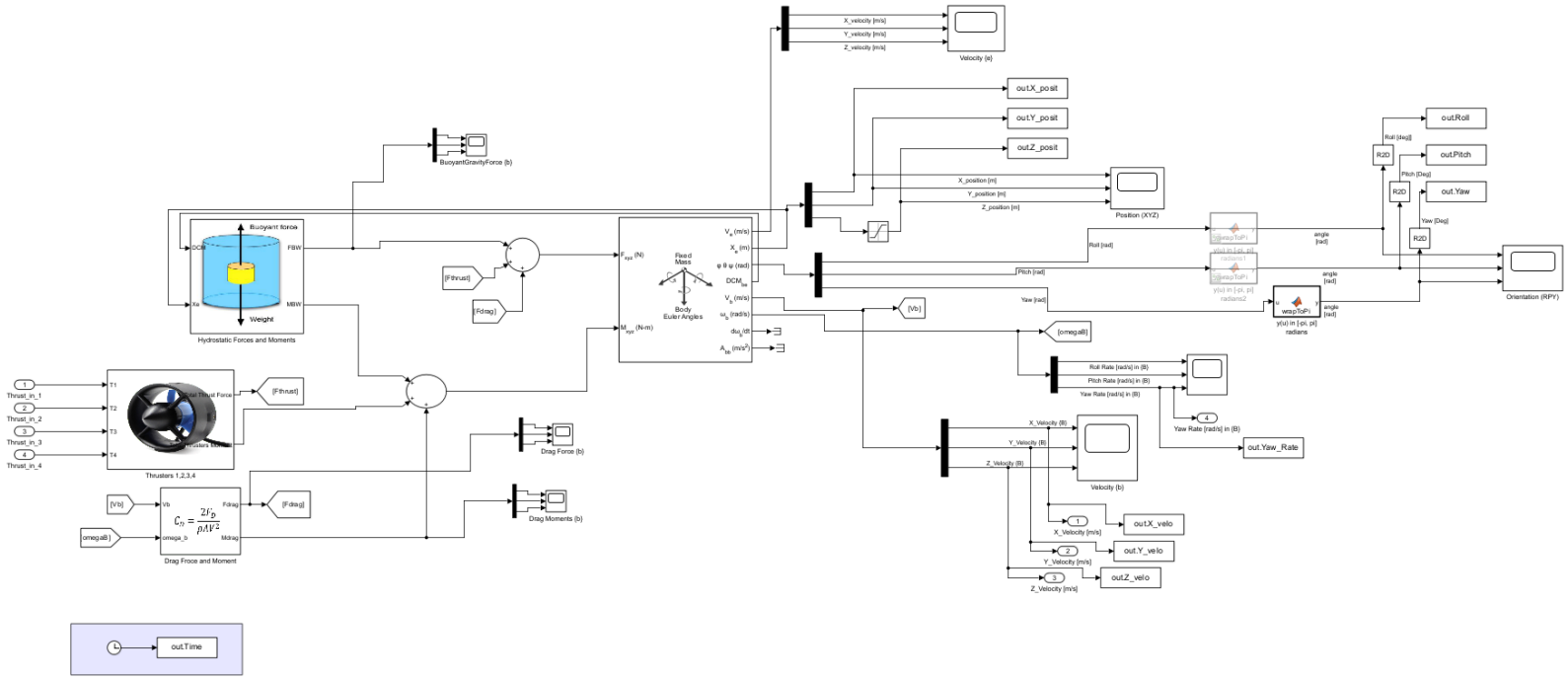


Figure 24. Velocity Controller WIEVLE Subsystem

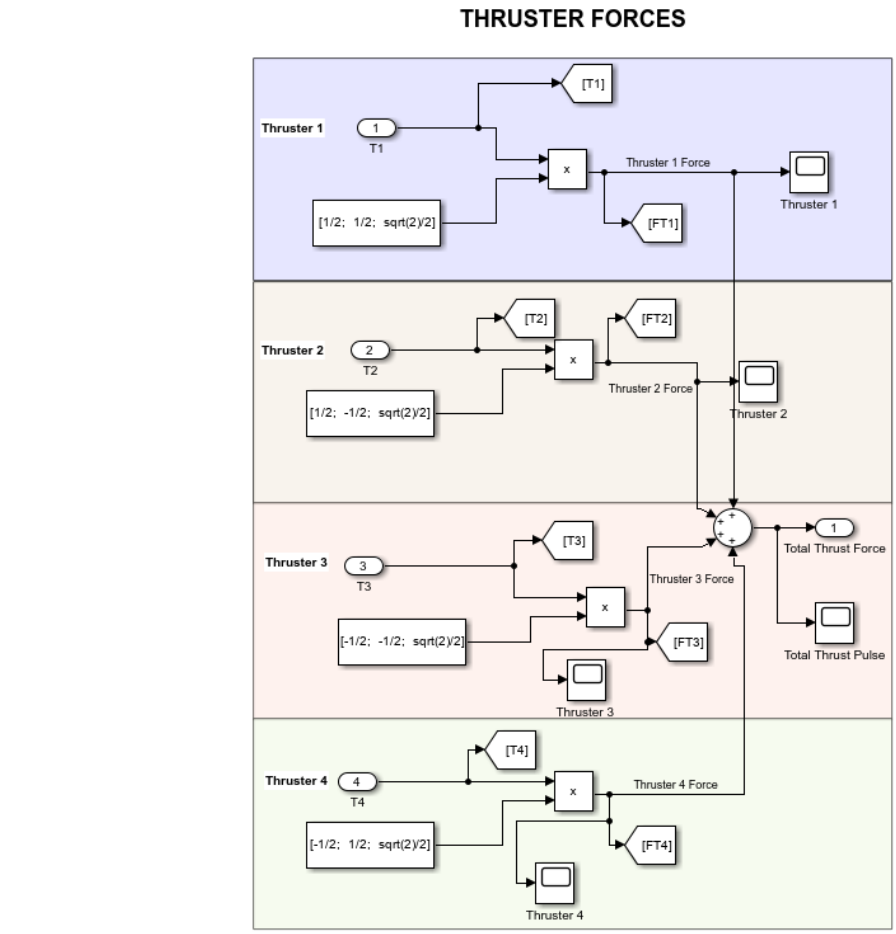


Figure 25. Velocity Controller Thruster Forces Subsystem

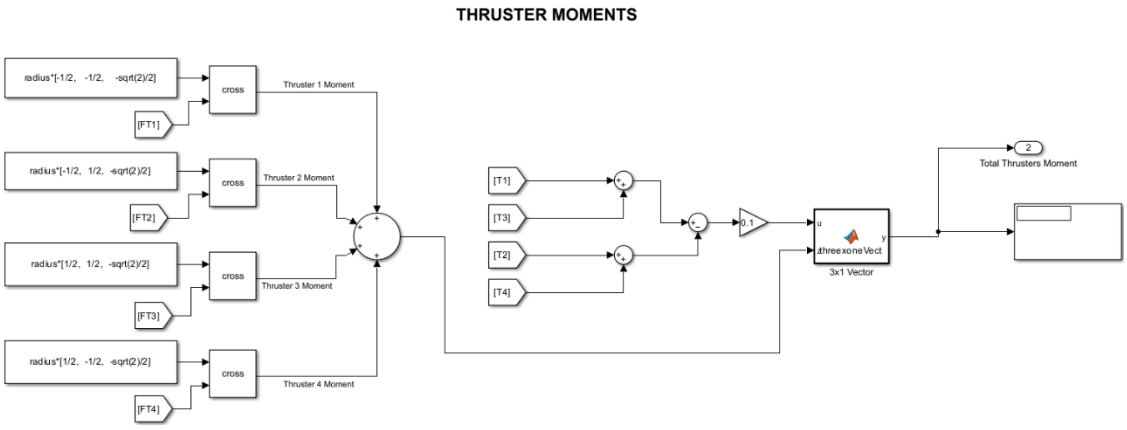
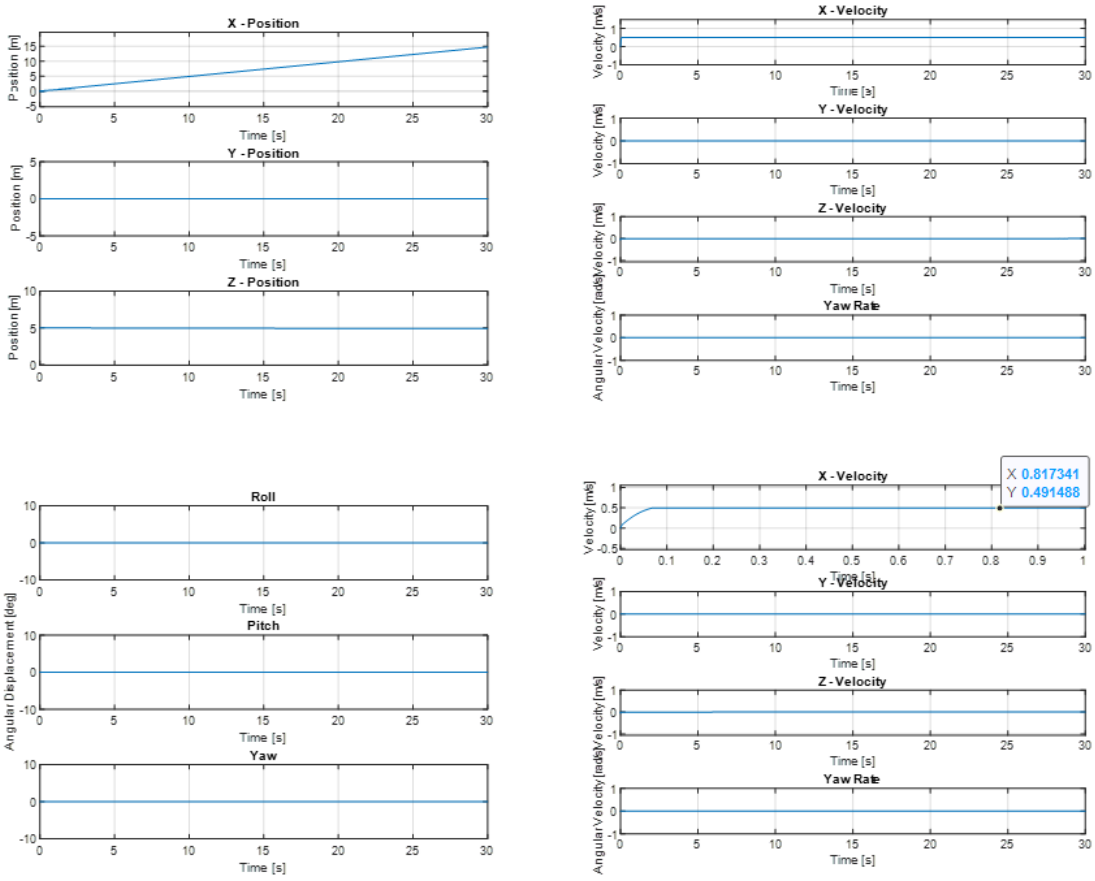


Figure 26. Velocity Controller Thruster Moments Subsystem

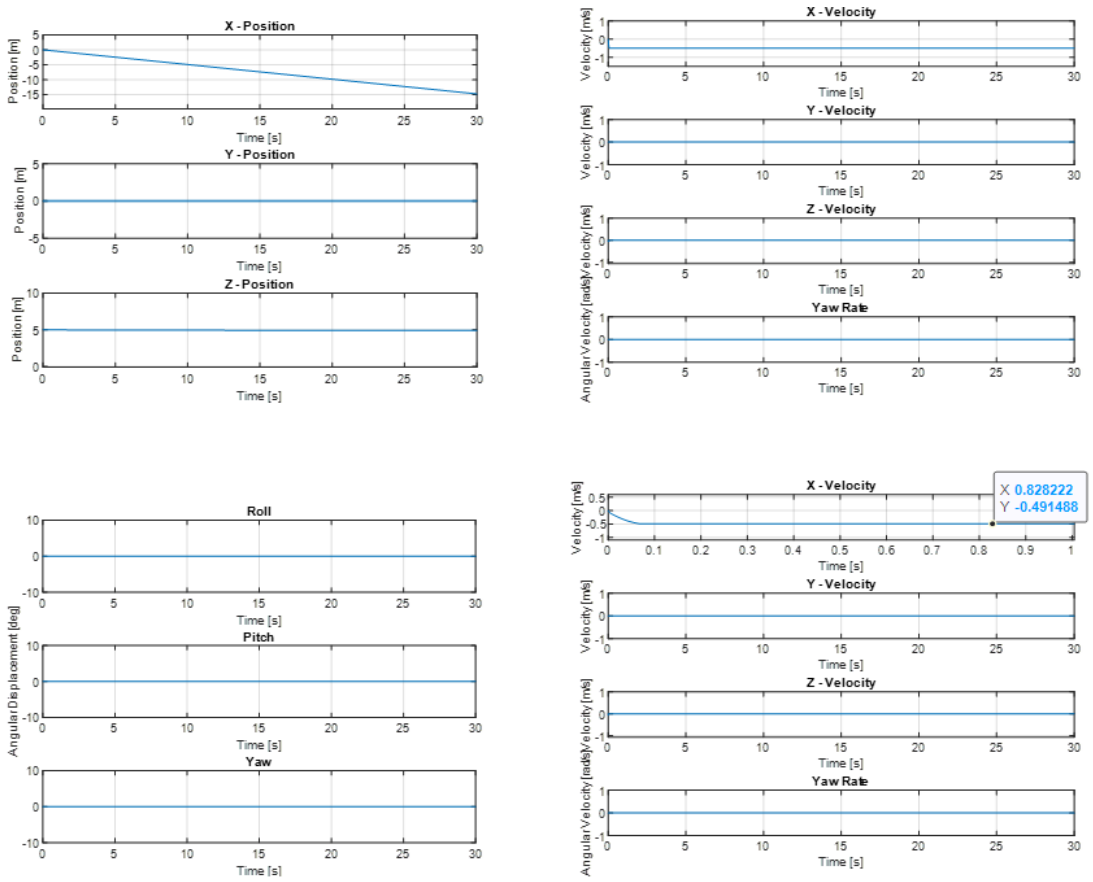
B. CLOSED LOOP VELOCITY CONTROLLER VALIDATION

Once the velocity controller was created in the Simulink environment, the logical next step was to validate the model and tune the PID gains to improve performance. All simulations in this subchapter start with a position of $[0 \quad 0 \quad 5]$m and zero starting velocities and angles. Figures 27–29 show the typical results of a 0.5 m/s and -0.5 m/s velocity command to each controllable DOF constant block, i.e., X, Y, and Z. Figures 30 and 31 show the results when a yaw velocity command of 0.5 rad/s and -0.5 rad/s is provided.



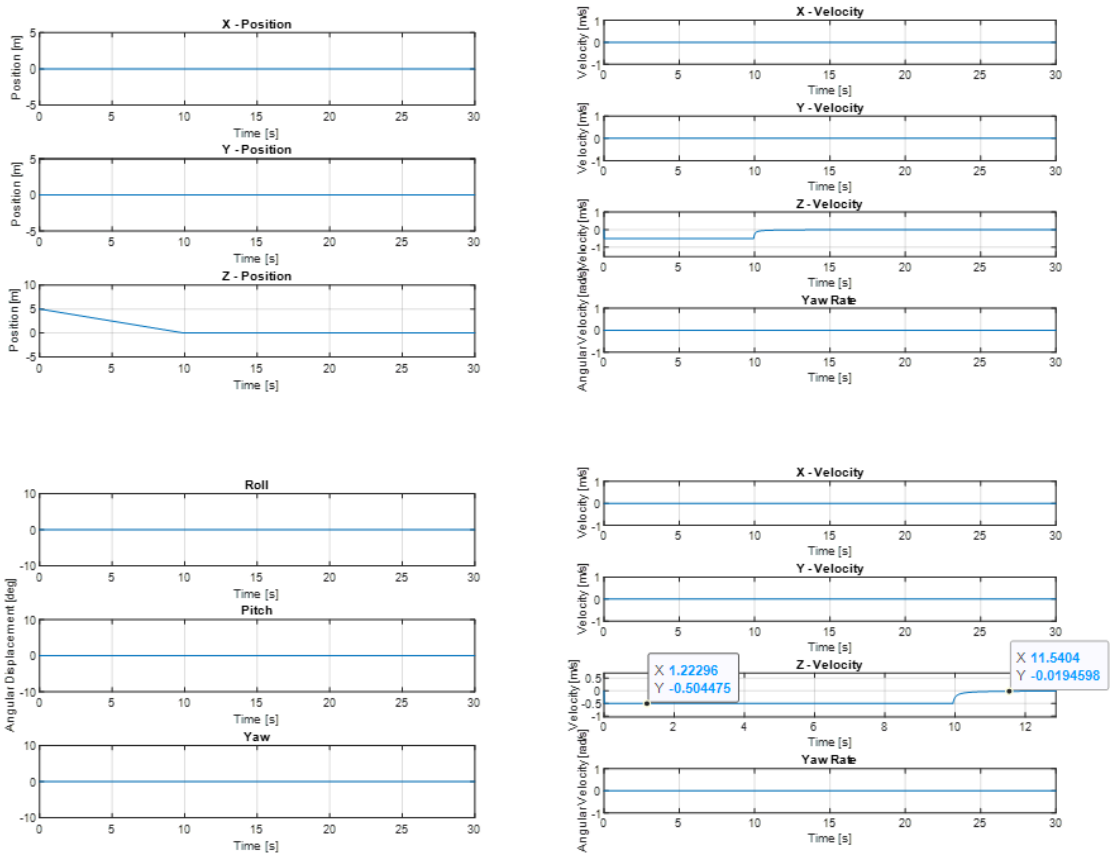
The bottom right plot shows the X-velocity transient from 0–1 s with an annotation indicating the commanded rate was achieved.

Figure 27. Closed Loop Velocity Controller Positive X Command



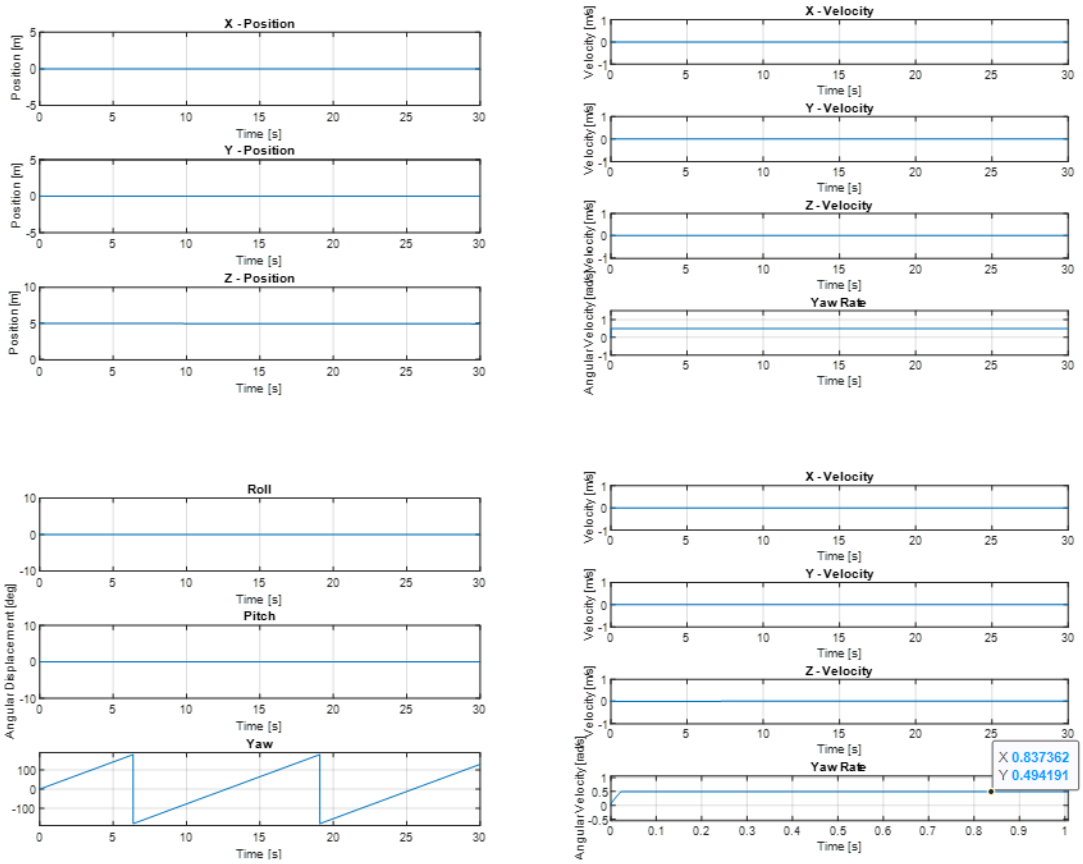
The bottom right plot shows the X-velocity transient from 0–1 s with an annotation indicating the commanded rate was achieved.

Figure 28. Closed Loop Velocity Controller Negative X Command



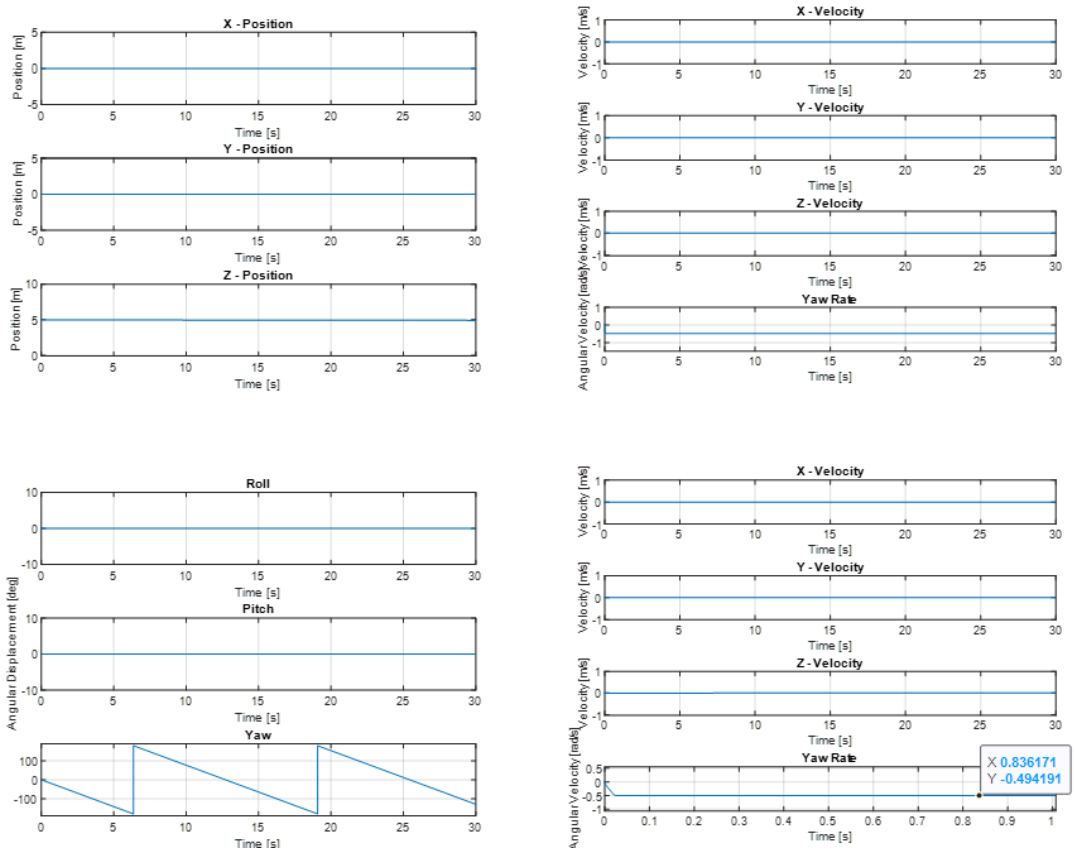
The bottom right plot shows the Z-velocity transient from 0–12 s with an annotation indicating the commanded rate was achieved.

Figure 29. Closed Loop Velocity Controller Negative Z Command



The bottom right plot shows the yaw-velocity transient from 0–1 s with an annotation indicating the commanded rate was achieved.

Figure 30. Closed Loop Velocity Controller Positive Yaw Command



The bottom right plot shows the yaw-velocity transient from 0–1 s with an annotation indicating the commanded rate was achieved.

Figure 31. Closed Loop Velocity Controller Negative Yaw Command

The results of the CL velocity controller were satisfactory and as expected. Through iteration, the PID gains were identified to tune the transient response of each DOF velocity to an acceptable degree. In each simulation, the rise times and steady state error responses were the main parameters of interest. The goal was to decrease the steady state error to less than 10 % and rise time less than 0.1 s. Both goals were met after the iterative tuning of the PID gains. In the negative Z velocity command simulation, it was interesting to see the difference in the Z velocity before and after the surface saturation. In this case, the response was intuitive as the Z velocity initially meets the -0.5 m/s command and then rises back to zero as the WIEVLE model reached the surface.

C. CLOSED LOOP POSITION AND ORIENTATION CONTROLLER SYNTHESIS

Once CL velocity control was achieved, the ensuing goal was to develop a position and orientation controller. This CL controller development was integral to proving the WIEVLE model could remain stable and controllable prior to any attempts at physical implementation. Using the CL velocity controller as a backbone, Figure 32 shows the control level of the position and orientation controller. The user inputs the desired {E} frame X, Y, Z, and yaw commands into the constant block, which was then compared to the current position and orientation angles fed back from the 6 DOF (Euler Angles) Simulink block in the WIEVLE subsystem. The X, Y, and Z position error signal, ${}^{E}e$, is then converted to the {B} frame using the Direction Cosine matrix, ${}^{B}_{E}R$, which incorporates the yaw angle supplied from the 6 DOF (Euler Angles) Simulink block as depicted by (38) and (39).

$$ {}^{B}_{E}R = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{38} $$

$$ {}^{B}e = {}^{B}_{E}R \cdot {}^{E}e \tag{39} $$

Equation (38) is dynamic in that it continuously updates based on the position of the WIEVLE. The dynamism is necessary because the position of the WIEVLE model is constantly changing until it reaches the goal. As the WIEVLE nears the goal, the vehicle slows as the error signal approaches zero. Then, once the goal is reached, the error signals go to zero and it stops. This method was chosen for its simplicity and resiliency, as it will work for any translation command. From this point, the position and orientation of the WIEVLE will be referred to as its *pose* for simplicity, see (40) for example.

$$ pose = \begin{bmatrix} {}^{E}x \\ {}^{E}y \\ {}^{E}z \\ {}^{E}\psi \end{bmatrix} = \begin{bmatrix} 1\,m \\ 1\,m \\ 5\,m \\ 90° \end{bmatrix} \tag{40} $$

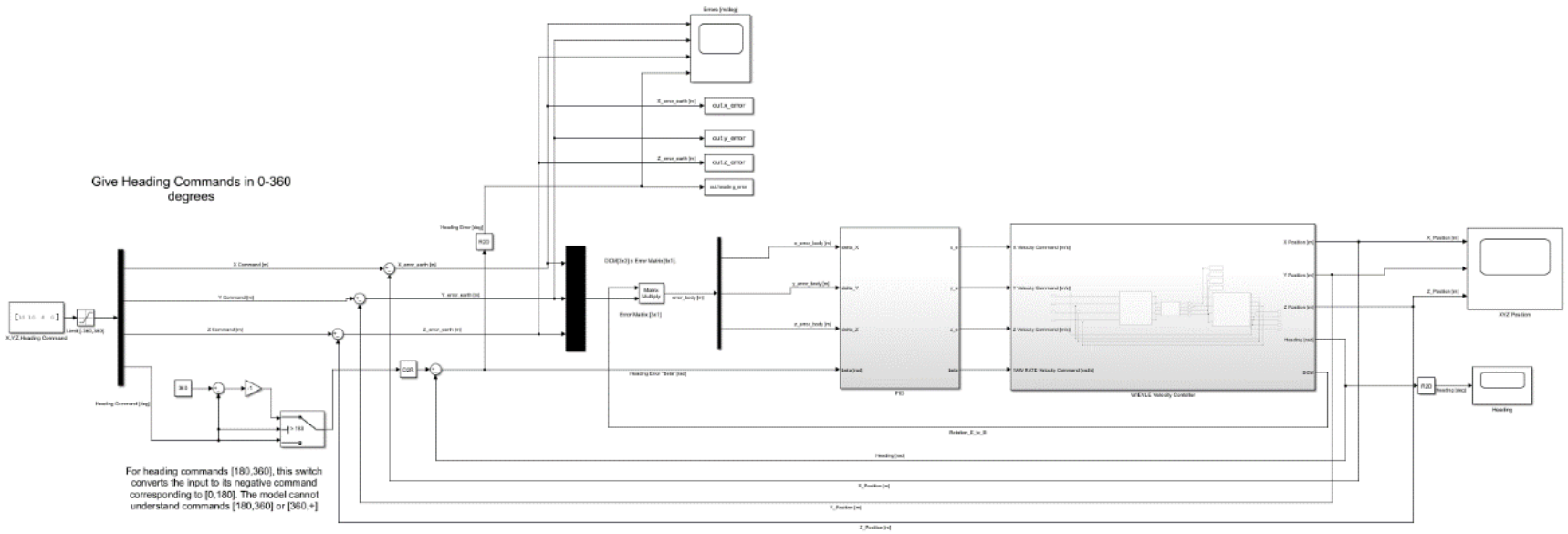


Figure 32. Pose Controller Model Control Level

Once the {B} frame error signal was obtained, it was multiplied by PID gains, shown by Figure 33, to influence the transient behavior and used as a proportional velocity command to the closed loop velocity controller backbone. After the command was sent to the velocity controller, the same control laws explained earlier in this chapter were applied and the WIEVLE model navigated to the commanded position.

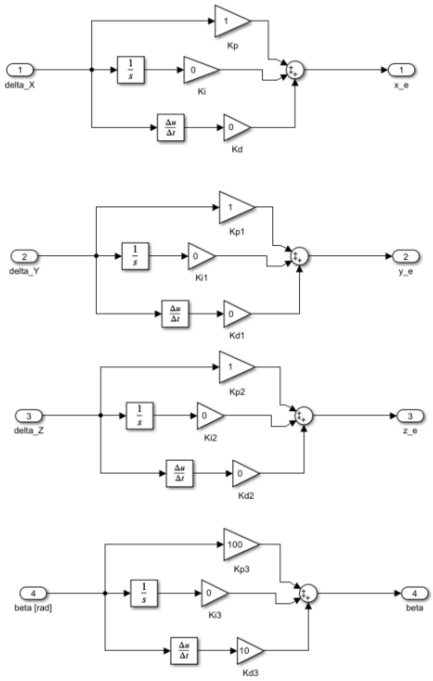


Figure 33. PID Block of Pose Controller

D. CLOSED LOOP POSITION AND ORIENTATION CONTROLLER TESTING

Once synthesis of the controller was completed, it was iteratively tested to prove resiliency and operability. During initial testing of this controller, the yaw command was increased from 0–360°. It was observed that any command greater than or less than +/-180°, respectively, would cause a singularity and force an unstable condition due to the complex math at the root of solving for the angle differential between the {B} and {E} frames. Trigonometric half-angle formulas were useful in explaining this phenomenon. Equations (41) and (42) show the sine and cosine half-angle formulas.

$$\sin\frac{\psi}{2} = sign\,(\sin\psi)\cdot\sqrt{(1-\cos\psi)/2} \tag{41}$$

$$\cos\frac{\psi}{2} = \sqrt{(1+\cos\psi)/2} \tag{42}$$

In (41), the $sign\,(\cdot)$ function returns +1 for positive arguments and -1 for negative arguments [8]. While these half-angle formulas are not expressly used in the derivation of the pose based on the yaw angle, they must be satisfied. From this, it was clear that at greater than positive or less than negative $180°$, (42) is not satisfied.

$$\cos\left(\frac{181}{2}\right) = \sqrt{(1+\cos(181))/2}$$

$$-0.0087 \neq 0.0087$$

To avoid this condition, a switch was utilized to check if the yaw command was greater than $180°$ or less than $-180°$. If either condition was met, it implements (43) and gives the corresponding angle within the valid range $-180° \le \psi \le 180°$.

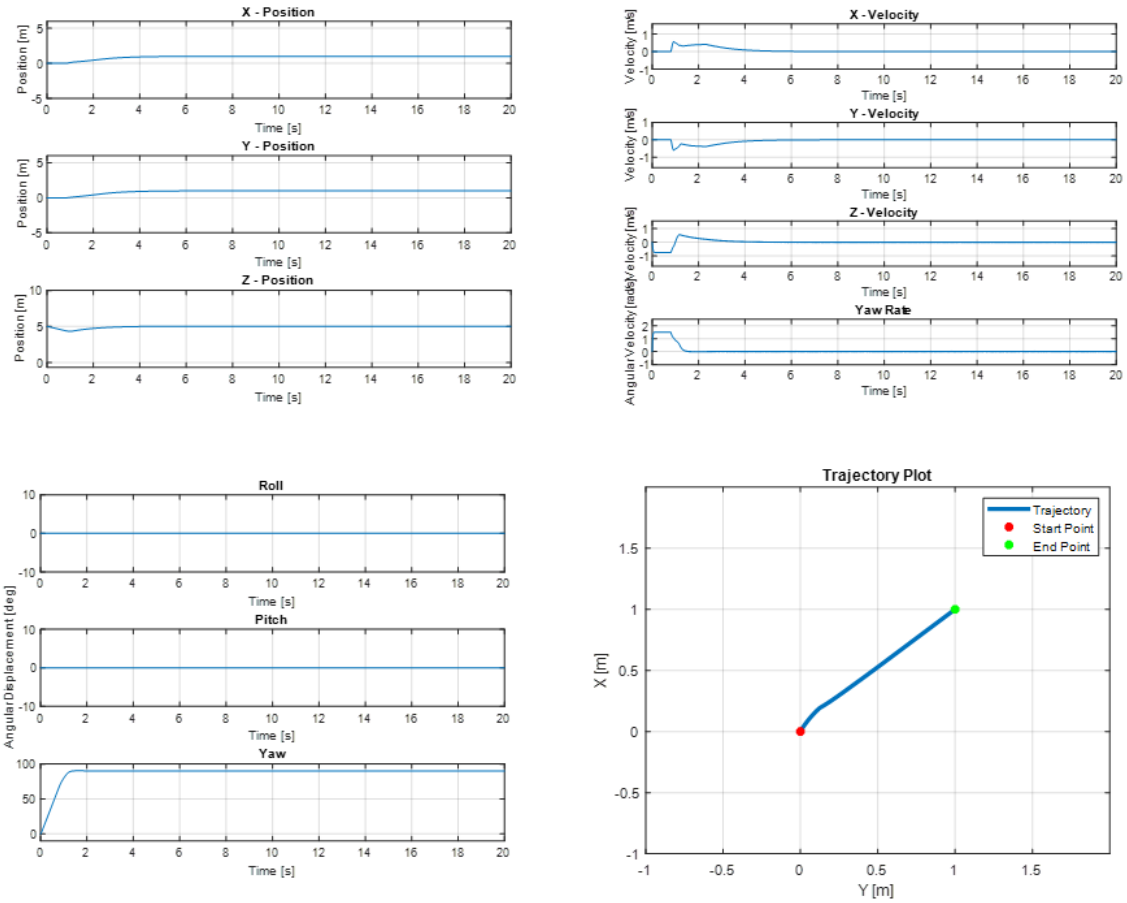
$$\psi_{command} = -\left(360 - \psi_{command>180}\right) \tag{43}$$

For example, a yaw command of $270°$ yields a command of:

$$\psi_{command} = -\,(360 - 270) = -\,90°$$

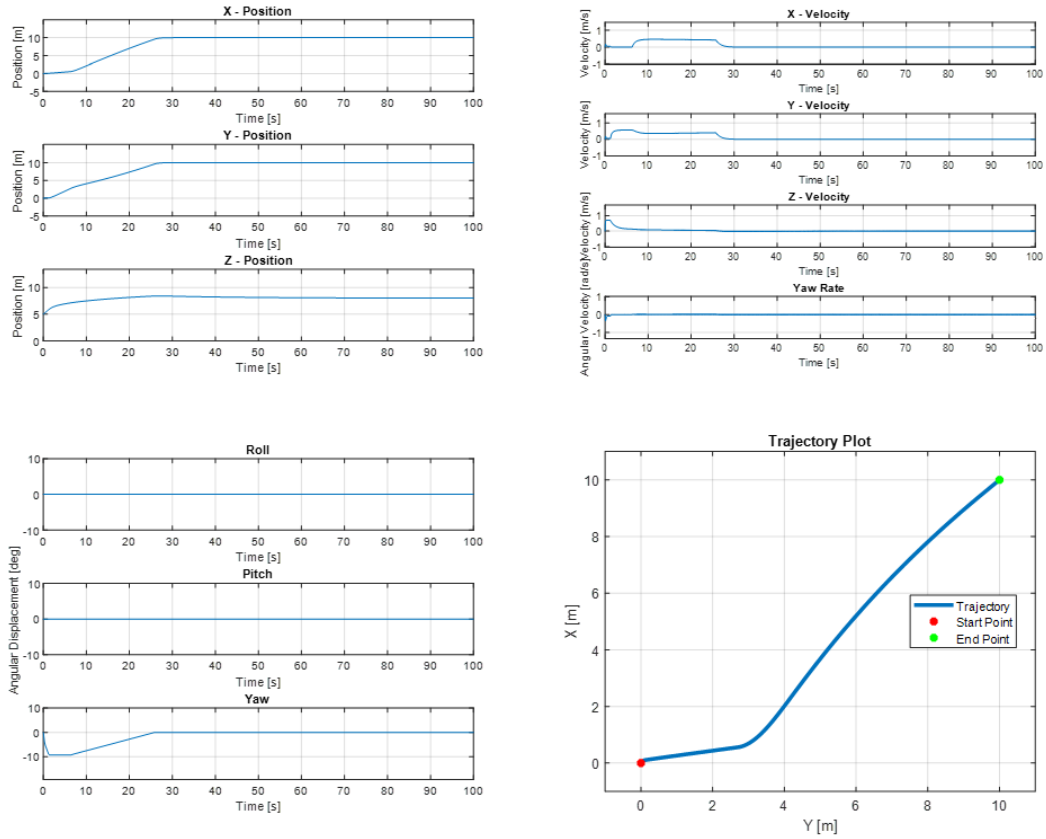
The $-90°$ corresponds to the commanded $270°$ and does not force the WIEVLE to cross the +/- $180°$ range, keeping all yaw commands from 0–$360°$ valid.

Once the Yaw command reconciliation method was proven, the next step was to test the controller. Figures 34–36 show three simulations starting at pose $\left[0\,m \quad 0\,m \quad 5\,m \quad 0°\right]^T$.



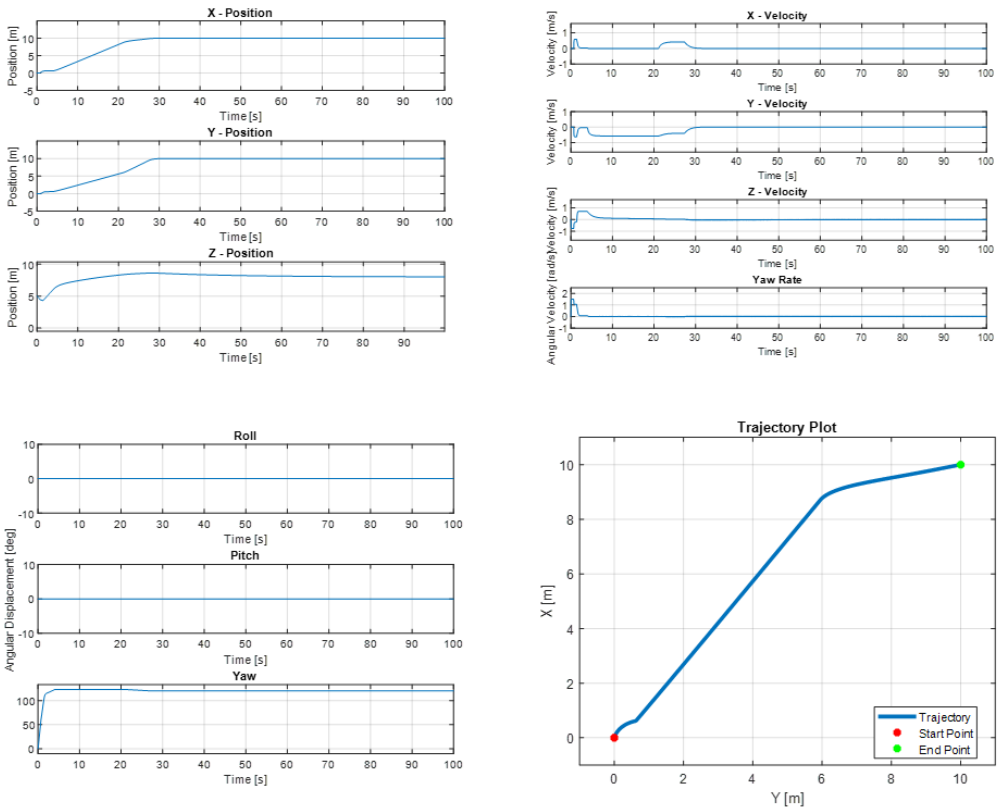
Commanded position and orientation: $\begin{bmatrix} 1\,m & 1\,m & 5\,m & 90° \end{bmatrix}^T$.

Figure 34. Closed Loop Pose Controller Simulation 1



Commanded position and orientation: $\begin{bmatrix} 10\,m & 10\,m & 8\,m & 0° \end{bmatrix}^T$.

Figure 35. Closed Loop Pose Controller Simulation 2



Commanded position and orientation: $\begin{bmatrix} 10\,m & 10\,m & 8\,m & 120° \end{bmatrix}^T$.

Figure 36. Closed Loop Pose Controller Simulation 3

Overall, the CL Pose control method employed in this model was successful in guiding the WIEVLE to the commanded pose. However, it was unexpected to note the WIEVLE *crabbing* as it was translating to the commanded position. The *crabbing* refers to the vehicle rotating to the commanded orientation while simultaneously beginning translation to the commanded position. This condition was not deemed to disturb operation of the modeled vehicle, though it was unintended and insightful to note.

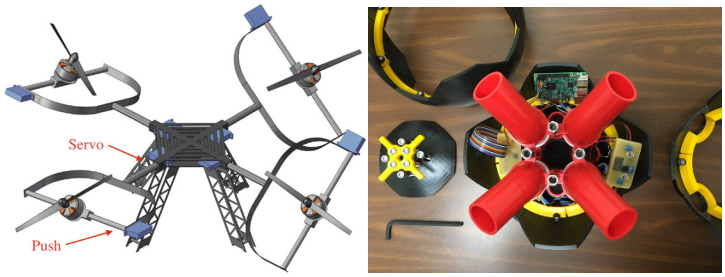
The aim of the computer modeling was to provide a working pose controller model, and it was achieved. After the computer modeling was complete, physical implementation of the pose controller was desired on an actual WIEVLE prototype. The next chapter seeks to implement the insights and validations from this chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. PIXHAWK CONTROLLER IMPLEMENTATION

The MATLAB/Simulink simulation work conducted previously established the use of a control scheme that uses a nested-loop control architecture where an inner loop is used to regulate velocities and an outer loop controls position. The cost-effective solution to complicated control problems is typically rooted in commercial off the shelf (COTS) technology. For the WIEVLE use case, it was decided to research viable COTS controllers. The goal was to implement a CL pose controller, like the controller described in Chapter IV.

When conceptualizing the control system for WIEVLE, the unique architecture of the vehicle bore resemblance to commercially available quadcopters rotated 180° along the X-axis, refer to Figure 37. The Pixhawk Mini, an unmanned aerial vehicle (UAV) autopilot manufactured by 3D Robotics, was identified as a viable COTS solution to the control problem as it also incorporated nested control loop architecture commonly used for quadcopter drone aircraft. Figure 38 shows the Pixhawk Mini and its Global Positioning System (GPS) receiver and power module.



Images compiled from two separate documents. Source: Ji et al. (2020) left. Source: Eldred and Van Bossuyt (2022) right.

Figure 37. Side-by-Side Comparison of Quadcopter to WIEVLE Construction

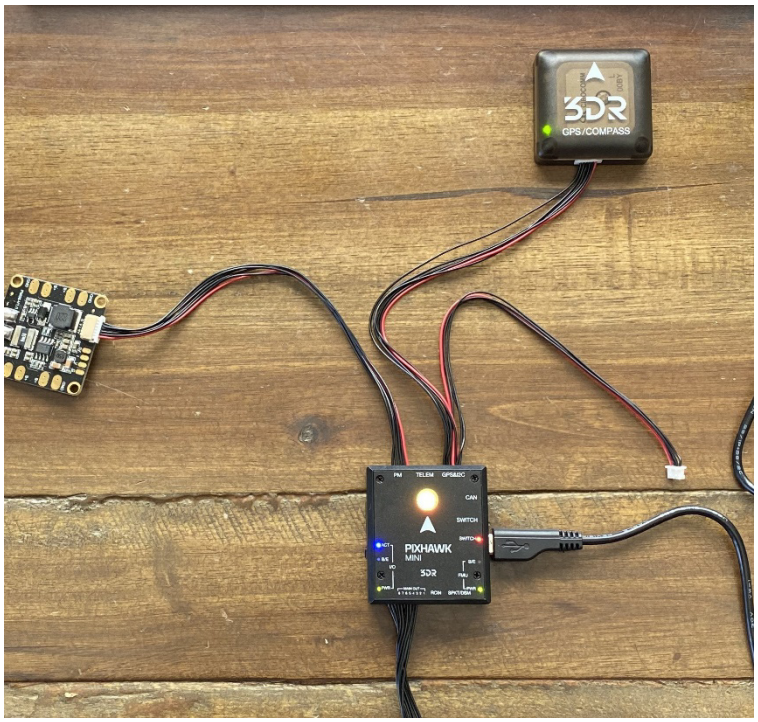


Figure 38. Pixhawk Mini Autopilot with GPS Compass and Power Module

The Pixhawk Mini inner and outer loops function similarly to the nested CL controller explained in Chapter IV by using a PID control scheme [10]. Figure 39 is a diagram of the control law within the Pixhawk Mini. $G(s)$ represents the angular rates of the vehicle, y is the body angular rate, r is the setpoint for the angular rates, e is the error between y and r, and u represents the output of the PID controller. Not depicted is the outer control loop that supplies the angular rate setpoint r based on the position and heading error signal.

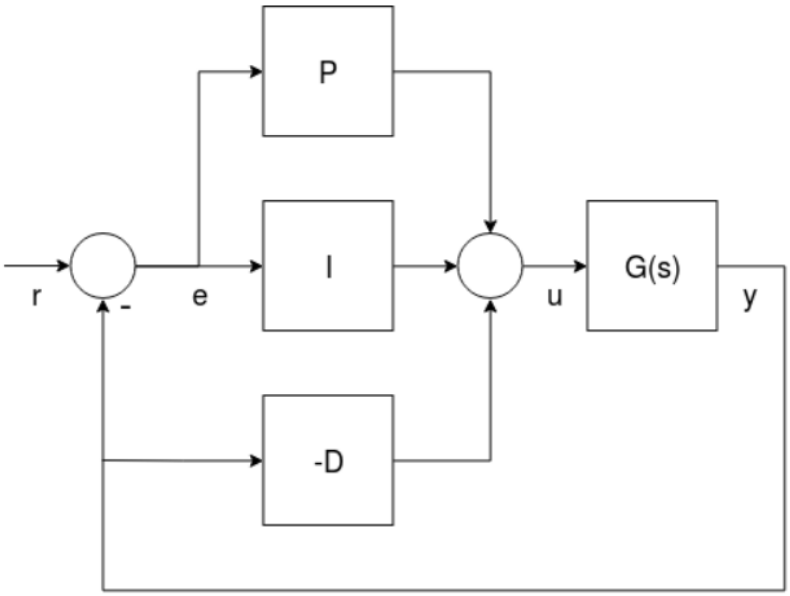


Figure 39. Pixhawk Mini Inner Control Loop. Source: [10].

With the appropriate firmware and hardware, the Pixhawk Mini can be adapted for underwater use [11]. The capability to adapt the Pixhawk Mini for underwater use and its similarities to the computer modeling control algorithm made it a viable candidate to control the WIEVLE. Using [11], the Pixhawk Mini was assembled and configured for use as a controller for the WIEVLE. Figure 40 shows the components and workflow used to initially achieve open-loop control of a WIEVLE thruster mockup.

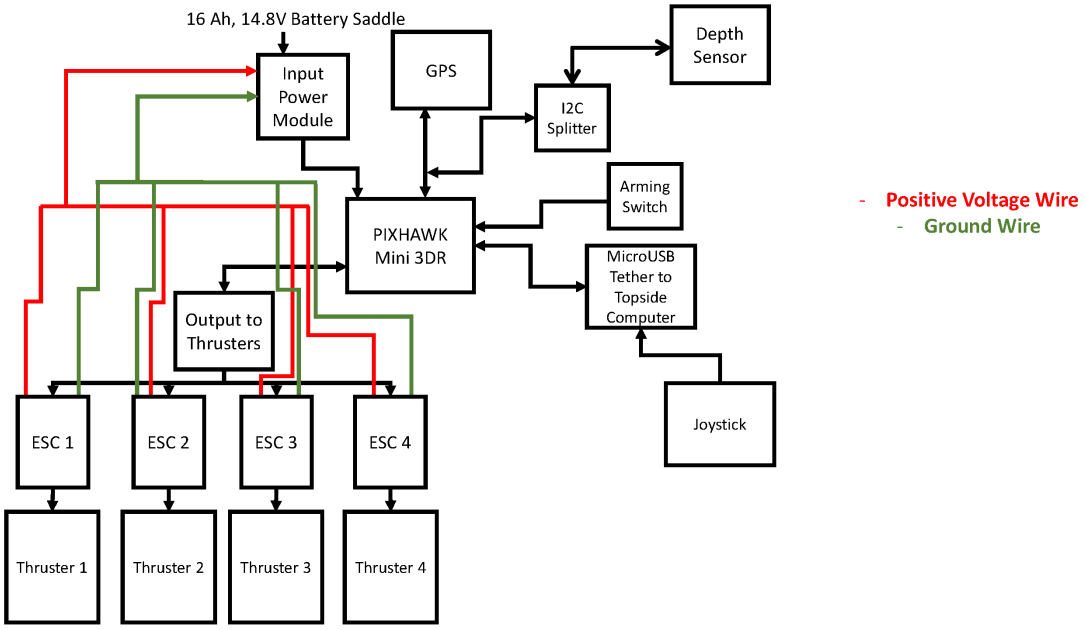


Figure 40. Pixhawk OL Tethered Controller Workflow

The GPS, inter-integrated circuit (I2C) splitter, arming switch, input power, thrusters, and tether provide feedback and inputs to the autopilot while the thruster commands are output through four electronic speed controllers (ESC). The ESCs are then directly connected to each thruster. The ESCs are crucial to thruster operation as they translate the Pixhawk two-phase command signal to a useful three-phase electrical signal to the thrusters [12]. This exercise was important for two reasons: first, the testing was a proof of concept that WIEVLE can be controlled this way and secondly, it provided visual insight as to how the rigid body of WIEVLE responds to thruster pairing commands in the physical world. Figure 41 shows a test bench mock-up of the Pixhawk controller sending thruster commands to the WIEVLE Thruster mockup in Figure 42. For example, a 100% forward command to any two opposing thrusters will cause the body to rotate, etc. This information was insightful when comparing it to the output of the computer modeling.

Figure 41. WIEVLE Open Loop Test Cart

Figure 42. WIEVLE Thruster Mockup

A. PIXHAWK UNTETHERED OPEN LOOP CONTROL

Test cart experiments proved that WIEVLE can be controlled through an ethernet tether back to a topside computer. The next focus was to attempt to integrate an untethered control variant, still utilizing the Pixhawk framework. For almost all intended mission sets of the WIEVLE, untethered operations are a necessity [3]. Figure 43 shows the concept of this untethered control using a radio transmitter (TX) and receiver (RCVR). As the figure illustrates, the use of a topside computer is unnecessary for control; the vehicle would essentially be driven around like an RC boat or car. The tether

depicted would be utilized simply for human interface with the Pixhawk onboard sensor suite, but ultimately redundant for operation. Upon further research, the feasibility of using a TX/RCVR construct was deemed infeasible [13]. states that Radio Frequency (RF) communications from air to fresh water are possible, with very poor performance [13]. states that wireless radio communications are not suited for seawater as the attenuation and propagation loss are extremely high. An RF signal of 10 kHz in freshwater, for example, experiences attenuation of about 18 dB at 0.5m and 40 dB at 100m [13]. Based on this, the TX will attempt to communicate the intended signal to WIEVLE. Once the signal breaks the surface of the water, it will almost immediately become so noisy, through attenuation, that the signal becomes infeasible for control.

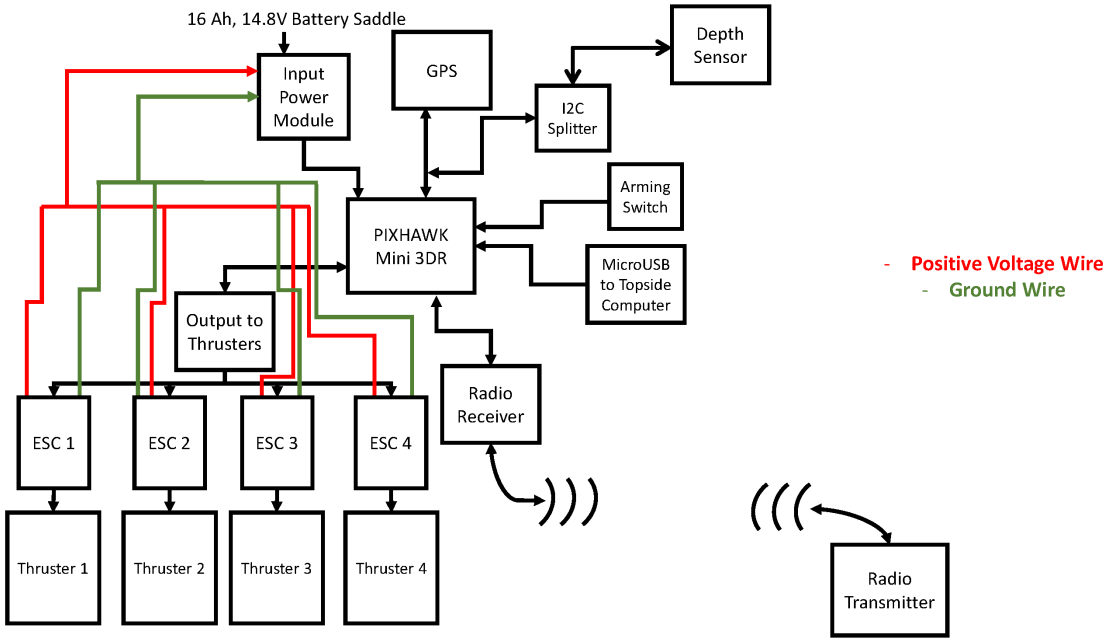


Figure 43. Pixhawk OL Untethered Controller Workflow

B. PIXHAWK UNTETHERED CLOSED LOOP CONTROL

Internal to the Pixhawk autopilot is a small Internal Measurement Unit (IMU) and Global Positioning System (GPS) receiver. The IMU and GPS work in tandem to provide a precise global position to the autopilot when in operation. With this global position

known and available, the Pixhawk would then manage the outputs to the thrusters to navigate to the desired position. A user would only need to input a desired position via the topside computer, and the autopilot would do the rest. Figure 44 shows a prospective workflow of a CL controller utilizing the Pixhawk autopilot.

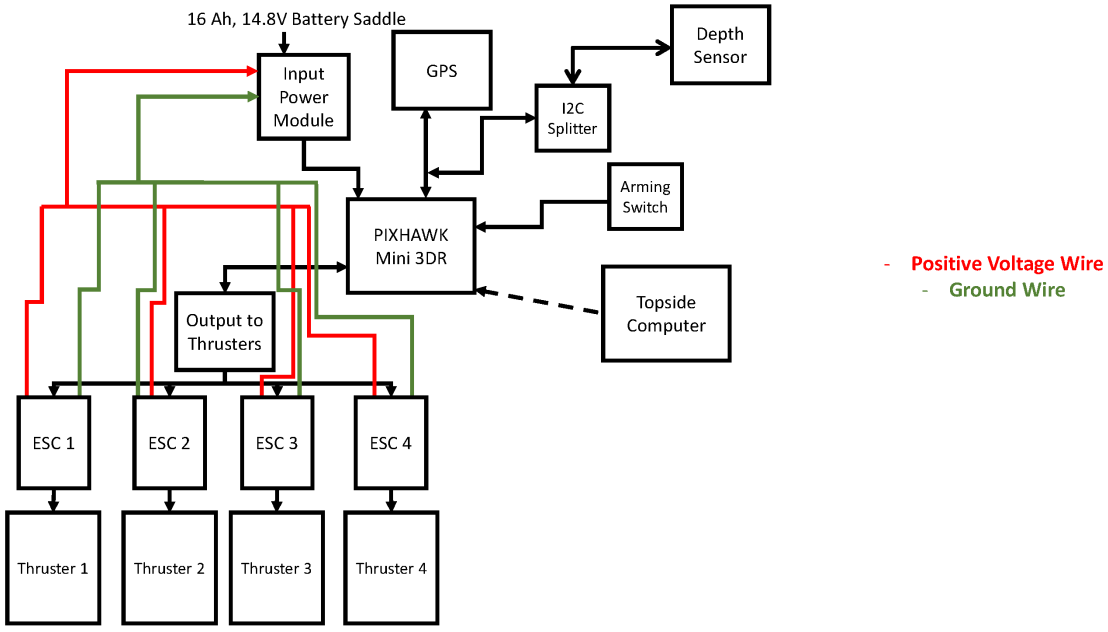


Figure 44. Pixhawk CL Untethered Controller Workflow

As with the untethered OL controller, signal attenuation occurs very shallow in the water. This attenuation would disrupt the GPS signal so significantly as to deem it unusable by the autopilot. Without the GPS signal, the IMU would be the only source of position estimations for X, Y, and Z displacement. It was believed that the IMU components were not sensitive enough to provide the quality of estimate needed to continue operation after the Pixhawk lost GPS signal.

Incidentally, this was confirmed when the test cart experiment was conducted. Through the companion software of the topside computer, the Pixhawk is easily switched from OL to CL control. Once in CL control, there was an option for *station keep*. This option used the Pixhawk IMU to compare its position to the GPS position and

accelerometers to sense movement. The idea being, if movement is sensed, the Pixhawk sends the appropriate thruster commands to counter act the movement and keep the WIEVLE in place. When *station keep* mode was enabled during the test cart experiment, the logic worked and seemingly resisted the changes introduced by moving the test cart. However, once the GPS signal flickered, the thrusters exhibited erratic and unstable behavior. Even when the GPS signal was reestablished, the thrusters remained unstable. The uncertainty of whether the Pixhawk controller could handle fluctuations in GPS connectivity was answered during the test cart experiment.

Overall, the Pixhawk experiments were useful as they verified the behaviors that were observed in the MATLAB/Simulink modeling. Though the experiments provided insight, it was decided the Pixhawk out-of-the-box navigation configuration was not a feasible option for untethered control of the WIEVLE. Further research and development can be done to upgrade the Pixhawk navigation equipment and make it a feasible solution.

The next chapter will summarize the work completed in this thesis and provide suggestions to continue the work started by this research.

VI. CONCLUSION AND FUTURE WORK

This thesis examined the introductory question of whether the WIEVLE robot could be modeled and controlled. Further, it explored the synthesis of attaining both OL and CL control of the WIEVLE. Chapter II stated the assumptions made and derived the necessary equations to create a computer model of the WIEVLE. Chapter III explained the modeling and testing of an OL controller for the WIEVLE model. Chapter IV described the synthesis and testing of two CL controllers applied to the WIEVLE model. Finally, Chapter V explained a successful experiment to implement physical OL control of a WIEVLE thruster mockup, and an unsuccessful attempt to implement untethered control.

From the evidence provided by the physical experiment and extensive computer modeling, the WIEVLE is believed to be a stable robotic platform that is suitable for UUV operations. However, as revealed by this research, stable CL control is not possible without accurate position and velocity data. The computer simulation showed good results because accurate sensor data was available for both the inner and outer control loops to operate.

In the investigation of COTS controller solutions, it was found that they relied largely on the availability of a constant GPS signal to provide the required sensor information. The autopilot controllers considered have an on-board IMU, but the quality of accelerometers and angular rate sensor performance negatively affected navigation. For the WIEVLE, satellite navigation would not be available beyond any depth useful for UUV operations because of signal attenuation.

A. FUTURE WORK

To continue the work of this thesis, two lines of effort are recommended. First, to implement the WIEVLE control system using an alternative to the Pixhawk autopilot called the Robot Operating System (ROS) architecture. Second, build the WIEVLE control system around navigation components that provide accurate position and velocity data while WIEVLE is submerged.

1. Robot Operating System and Gazebo

ROS is a useful open-source repository of software development kits for numerous robotics applications [14]. Primarily written in Python, C++, and Lisp, ROS claims language independence to implement its tools [15]. Leveraging the tools of ROS allows distributed communication between nodes of a publisher, subscriber, and service type relationship network. A master node is designated and responsible for keeping track of communications across the network. This master node monitors other nodes that are communicating via topic or service messages. Subscribers and publishers transmit and receive the topic messages which include translation commands, rotation commands, etc. Topic messages are unique in that they can be transmitted to the entire network or specific nodes. Service messages are specific peer-peer communications that request information and generate a response [15].

Given ROS has gained widespread use and popularity among contemporary roboticists, it seems to be a viable candidate to utilize for control of the WIEVLE. Due to the prevalence of MATLAB/Simulink in industry and academia alike, MathWorks, the parent development company of MATLAB/Simulink, provides tools to establish and interface with ROS networks using the Python interpreter and the ROS Toolbox. Leveraging these tools would allow the WIEVLE project to remain within the MATLAB/ Simulink environment but communicate over a ROS network to a WIEVLE robot.

Simulation software, like Gazebo, exists to test the MATLAB/Simulink to ROS connectivity paths and should be utilized prior to physical testing [16][17]. describes an extension of Gazebo to simulate robots in an underwater environment. Using the tutorials [18] and [19], a method of simulation using the MATLAB/Simulink–ROS infrastructure was attempted in conjunction with this thesis. Unfortunately, it was not completed due to time constraints and hardware shortfalls. This simulation method will be useful because it can be easily swapped for a physical WIEVLE once the ROS network infrastructure is sure.

2. Navigation Equipment

When investigating how other AUVs have solved the problem of obtaining position and velocity data regardless of GPS connection, the Remote Environment Monitoring Units (REMUS) family of AUVs is prevalent throughout the U.S. Navy. The REMUS utilizes an Inertial Navigation System (INS) for navigation [20]. An INS calculates its position, orientation, and velocity without the need for a continuous GPS signal [21]. The INS takes an initial GPS fix at the surface and estimates the position of the sensor based on the heading, velocity, and time elapsed from the fix [21]. A purpose-built WIEVLE controller that utilizes an INS would solve the issue of obtaining position fixes and velocity rates. However, it will be challenging to integrate INS equipment into WIEVLE. Due to physical space constraints and the current free flooding hull of WIEVLE, it is necessary to overcome these challenges for navigation. The computer model created during this research allows any changes to the physical parameters of the WIEVLE (i.e., weight, CG, CB, diameter, etc.) to be accounted for and simulated.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. OPEN LOOP MODEL PARAMETER CODE

```
% LT Matthew Coleman and Dr. James Calusdian. Naval Postgraduate School - 2022
% Declare parameters- Version A.  This script loads the Workspace with the various
% constants we need for our Simulink model.  Run this script first before
% launching the Simulink model.  Use Version A of the Simulink model.


clc
clear all
close all

mass = 2.26796;  % mass [kg] or [5 lbs]
g = 9.81;        % gravity [m/s^2]
dia = 0.3;      % diameter = 30 cm or 0.3 m. ~ 12 in.
radius = dia/2;
CG = [0;0;0]; % assume CG is concentrated at the geometric center of the UUV.
CB = [0;0;-radius]; % assume bouyant force is at the very top of the UUV along negative z-axis
volume = 4/3*pi * radius^3;   % volume of UUV for bouyant force [m^3]
surfaceArea = 4*pi*radius^2;  % surface area for drag calculation
CdF = 0.5;                   % drag force coefficient
CdM = 10;                    % drag moment coefficient, was 0.5
rho = 1026;         % density of water [kg/m^3]
I = mass*radius^2;  % treat as a pendulum for all inertia quantities
inertiaMatrix = [[mass       0           0           0           mass*CG(3)        -mass*CG(2)]
                 [0        mass          0      -mass*CG(3)          0             mass*CG(1)]
                 [0         0           mass     mass*CG(2)     -mass*CG(1)             0]
                 [0    -mass*CG(3)    mass*CG(2)    I               -I                -I]
                 [mass*CG(3)   0     -mass*CG(1)   -I                I                -I]
                 [-mass*CG(2)  mass*CG(1)     0    -I               -I                 I]];


% inverse of inertia matrix
MRB_inv = inv(inertiaMatrix);
fWeight = [0;0; mass*g];    % weight in Inertia frame {i}
fBuoy = [0;0;-rho*g*volume];  % buoyant force in inertia frame {i} [N]

forwardthrust100 = 5.425*9.807; %Thrust value for the selected thrusters (kgf to N conversion)
reversethrust100 = -1.82*9.807;
zerothrust = 0;
staticdepthHold = 1.1928*(abs(fBuoy(3)/(forwardthrust100*4))*forwardthrust100); % static depth hold command in [N].
balanceFWD = forwardthrust100-staticdepthHold;
depthHold = staticdepthHold - balanceFWD; % depth hold command in [N].

% first command from stepstart to stepend below
Thrust1 = 0;
Thrust2 = 0;
Thrust3 = 0;
Thrust4 = 0;
step1start = 0;
step1end = 10;
% second command from stepend to step2end
Thrust_1_2 = 0;
Thrust_2_2 = 0;
Thrust_3_2 = 0;
Thrust_4_2 = 0;
step2start = step1end;
step2end = 15;
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. OPEN LOOP MODEL PLOT CODE

```
% LT Matthew Coleman and Dr. James Calusdian
% Output Plots - Version A. This script imports the {E} frame position,
% orientation, and velocity data from SIMULINK and produces the resulting
% time domain plots.

X_posit = out.X_posit;
Y_posit = out.Y_posit;
Z_posit = out.Z_posit;
X_velo = out.X_velo;
Y_velo = out.Y_velo;
Z_velo = out.Z_velo;
roll = out.Roll;
pitch = out.Pitch;
yaw = out.Yaw;
time = out.Time;

figure(1) %Position Comparison
subplot(3,1,1)
plot(time,X_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('X - Position')
ylim([min(X_posit)-5 max(X_posit)+5])
subplot(3,1,2)
plot(time,Y_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('Y - Position')
ylim([min(Y_posit)-5 max(Y_posit)+5])
subplot(3,1,3)
plot(time,Z_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('Z - Position')
ylim([min(Z_posit)-5 max(Z_posit)+5])
figure(2) %Velocity Comparison
subplot(3,1,1)
plot(time,X_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('X - Velocity')
ylim([min(X_velo)-5 max(X_velo)+5])
subplot(3,1,2)
plot(time,Y_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('Y - Velocity')
ylim([min(Y_velo)-5 max(Y_velo)+5])
subplot(3,1,3)
plot(time,Z_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('Z - Velocity')
ylim([min(Z_velo)-5 max(Z_velo)+5])

figure(3) %Angular Displacement Comparison
subplot(3,1,1)
plot(time,roll)
grid on
xlabel('Time [s]')
title('Roll')
ylim([min(roll)-10 max(roll)+10])
subplot(3,1,2)
plot(time,pitch)
grid on
xlabel('Time [s]')
ylabel('Angular Displacement [deg]')
title('Pitch')
ylim([min(pitch)-10 max(pitch)+10])
subplot(3,1,3)
plot(time,yaw)
grid on
xlabel('Time [s]')
title('Yaw')
ylim([min(yaw)-10 max(yaw)+10])
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. CLOSED LOOP MODEL PARAMETER CODE

```
% Dr. James Calusdian and LT Matthew Coleman. Naval Postgraduate School - 2022.
% Declare parameters- Version A2. For use with Closed Loop WIEVLE model. No
% need to run prior to SIMULINK, as the matlab script is an initializing
% function for the model.


clc
clear all
close all

mass = 2.26796;  % mass [kg] or [5 lbs]
g = 9.81;        % gravity [m/s^2]
dia = 0.3;      % diameter = 30 cm or 0.3 m. ~ 12 in.
radius = dia/2;
CG = [0;0;0]; % assume CG is concentrated at the geometric center of the UUV.
CB = [0;0;-radius]; % assume bouyant force is at the very top of the UUV along negative z-axis
volume = 4/3*pi * radius^3;   % volume of UUV for bouyant force [m^3]
surfaceArea = 4*pi*radius^2;  % surface area for drag calculation
CdF = 0.5;                   % drag force coefficient
CdM = 10;                   % drag moment coefficient, was 0.5
rho = 997;         % density of water [kg/m^3]
I = mass*radius^2;  % treat as a pendulum for all inertia quantities

inertiaMatrix = [[mass       0          0          0          mass*CG(3)        -mass*CG(2) ]
                 [0        mass          0      -mass*CG(3)       0              mass*CG(1)]
                 [0        0           mass      mass*CG(2)     -mass*CG(1)          0]
                 [0    -mass*CG(3)    mass*CG(2)    I             -I                -I]
                 [mass*CG(3)   0     -mass*CG(1)   -I              I                -I]
                 [-mass*CG(2)  mass*CG(1)     0    -I             -I                I] ];


% inverse of inertia matrix
MRB_inv = inv(inertiaMatrix);

fWeight = [0;0; mass*g];    % weight in Inertia frame {i}
fBuoy = [0;  0;  -rho*g*volume];  % buoyant force in inertia frame {i} [N]
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. CLOSED LOOP MODEL PLOT CODE

```
% Output Plots - Version B. For use with Closed Loop WIEVLE model. No
% need to run prior to SIMULINK, as the matlab script is a stop
% function for the model.
% Dr. James Calusdian and LT Matthew Coleman - Naval Postgraduate School
% 2022.

% Grab the Output conditions of the simulation from the workspace output
% structure.
X_posit = out.X_posit;
Y_posit = out.Y_posit;
Z_posit = out.Z_posit;
X_velo = out.X_velo;
Y_velo = out.Y_velo;
Z_velo = out.Z_velo;
Yaw_Rate = out.Yaw_Rate;
roll = out.Roll;
pitch = out.Pitch;
yaw = out.Yaw;
time = out.Time;

figure(1) %Position Comparison
subplot(3,1,1)
plot(time,X_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('X - Position')
ylim([min(X_posit)-5 max(X_posit)+5])
subplot(3,1,2)
plot(time,Y_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('Y - Position')
ylim([min(Y_posit)-5 max(Y_posit)+5])
subplot(3,1,3)
plot(time,Z_posit)
grid on
xlabel('Time [s]')
ylabel('Position [m]')
title('Z - Position')
ylim([min(Z_posit)-5 max(Z_posit)+5])
figure(2) %Velocity Comparison
subplot(4,1,1)
plot(time,X_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('X - Velocity')
ylim([min(X_velo)-1 max(X_velo)+1])
% xlim([-0.0025 0.25])
subplot(4,1,2)
plot(time,Y_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('Y - Velocity')
ylim([min(Y_velo)-1 max(Y_velo)+1])
% xlim([-0.0025 0.25])
subplot(4,1,3)
plot(time,Z_velo)
grid on
xlabel('Time [s]')
ylabel('Velocity [m/s]')
title('Z - Velocity')
ylim([min(Z_velo)-1 max(Z_velo)+1])
% xlim([-0.0025 0.25])
subplot(4,1,4)
plot(time,Yaw_Rate)
grid on
xlabel('Time [s]')
ylabel('Angular Velocity [rad/s]')
title('Yaw Rate')
ylim([min(Yaw_Rate)-1 max(Yaw_Rate)+1])
xlim([-0.0025 0.25])

figure(3) %Angular Displacement Comparison
subplot(3,1,1)
plot(time,roll)
grid on
xlabel('Time [s]')
title('Roll')
ylim([min(roll)-10 max(roll)+10])
subplot(3,1,2)
plot(time,pitch)
grid on
xlabel('Time [s]')
ylabel('Angular Displacement [deg]')
title('Pitch')
ylim([min(pitch)-10 max(pitch)+10])
subplot(3,1,3)
plot(time,yaw)
grid on
xlabel('Time [s]')
title('Yaw')
ylim([min(yaw)-10 max(yaw)+10])


%Trajectory Plot
figure(4)
plot(Y_posit,X_posit,'Linewidth',3)
hold on
plot(Y_posit(1),X_posit(1),'r*','LineWidth',3)
plot(Y_posit(end),X_posit(end),'g*','LineWidth',3)
hold off
legend('Trajectory','Start Point','End Point')
ylim([min(X_posit)-5 max(X_posit)+5])
xlim([min(Y_posit)-5 max(Y_posit)+5])
grid
title('Trajectory Plot')
ylabel('X [m]')
xlabel('Y [m]')
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] B. Fletcher, "UUV master plan: A vision for Navy UUV development," in *OCEANS 2000 MTS/IEEE Providence*, Providence, RI, USA, 2000. Available: https://doi.org/10.1109/OCEANS.2000.881235

[2] R. A. Eldred, "Autonomous underwater vehicle architecture synthesis for shipwreck interior exploration," M.S. thesis, Dept. of Sys. Eng., NPS, Monterey, CA, USA, 2015. Available: https://calhoun.nps.edu/handle/10945/47940

[3] R. Eldred, J. Lussier, and A. Pollman, "Design and testing of a spherical autonomous underwater vehicle for shipwreck interior exploration," *JMSE*, vol. 9, no. 3, p. 320, Mar. 2021. Available: https://doi.org/10.3390/jmse9030320

[4] G. Antonelli, *Underwater Robots*, 3rd ed. New York, NY, USA: Springer International Publishing, 2014. Available: https://link.springer.com/book/10.1007/978-3-319-02877-4

[5] T. Fossen I., *Handbook of Marine Craft Hydrodynamics and Motion Control*, 2nd ed. Hoboken, NJ, USA: John Wiley and Sons, 2021.

[6] K. Fedorenko and C. D'Souza, "System identification of blue robotics thrusters video," Mathworks. Accessed: May 20, 2023. Available: https://www.mathworks.com/videos/matlab-and-simulink-robotics-arena-from-data-to-model-1518156121608.html

[7] Mathworks, "Implement euler angle representation of six-degrees-of-freedom equations of motion – Simulink." Accessed: May 15, 2023. Available: https://www.mathworks.com/help/aeroblks/6dofeulerangles.html

[8] X. Yun, E. R. Bachmann, and R. B. McGhee, "A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements," *IEEE Trans. on Inst. and Mea.*, vol. 57, no. 3, pp. 638–650, Mar. 2008. Available: https://doi.org/10.1109/TIM.2007.911646

[9] R. Ji, J. Ma, and S. Sam Ge, "Modeling and control of a tilting quadcopter," *IEEE Trans on. Aerosp. Electron. Syst.*, vol. 56, no. 4, pp. 2823–2834, Aug. 2020. Available: https://doi.org/10.1109/TAES.2019.2955525

[10] Px4, "Multicopter PID tuning guide (manual/advanced) | PX4 user guide." Accessed: Jul. 25, 2023. Available: https://docs.px4.io/main/en/config_mc/pid_tuning_guide_multicopter.html#rate-controller

[11] Ardusub, "Overview – gitbook." Accessed: Jul. 25, 2023. Available: https://www.ardusub.com/

[12] Blue Robotics, “Basic ESC.” Accessed: Jul. 26, 2023. Available: https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/

[13] S. Jiang and S. Georgakopoulos, “Electromagnetic wave propagation into fresh water,” *JEMAA*, vol. 03, no. 07, pp. 261–266, Jul. 2011. Available: https://doi.org/10.4236/jemaa.2011.37042

[14] ROS, “ROS: Home.” Accessed: Jul. 25, 2023. Available: https://ros.org/

[15] ROS, “ROS/introduction – ROS wiki.” Accessed: Jul. 26, 2023. Available: https://wiki.ros.org/ROS/Introduction

[16] J. Britto, A. Conceição, S. Joyeux and J. Albiez, “Improvements in dynamics simulation for underwater vehicles deployed in Gazebo,” in *OCEANS 2017 – Anchorage*, Anchorage, AK, USA. 2017. Available: https://ieeexplore.ieee.org/document/8232205

[17] M. M. M. Manhães, S. A. Scherer, M. Voss, L. R. Douat, and T. Rauschenbach, “UUV simulator: a Gazebo-based package for underwater intervention and multi-robot simulation,” in *OCEANS 2016 MTS/IEEE Monterey*, Monterey, CA, USA, 2016. Available: https://doi.org/10.1109/OCEANS.2016.7761080

[18] Mathworks, “Get started with Gazebo and simulated TurtleBot – MATLAB & Simulink.” Accessed: Jul. 26, 2023. Available: https://www.mathworks.com/help/ros/ug/get-started-with-gazebo-and-a-simulated-turtlebot.html

[19] Mathworks, “Move a TurtleBot robot using ROS actions – MATLAB & Simulink.” Accessed: Jul. 26, 2023. Available: https://www.mathworks.com/help/ros/ug/move-a-turtlebot-robot-using-ros-actions.html

[20] Huntington Ingalls Industries, “Remus 100 UUV data sheet.” Accessed: Jul. 26, 2023. Available: https://hii.com/what-we-do/capabilities/unmanned-systems/remus-uuvs/

[21] A. Bitar, “4 interesting facts about inertial navigation systems,” Honeywell Aerospace. Accessed: Jul. 28, 2023. Available: https://aerospace.honeywell.com/us/en/about-us/blogs/four-facts-about-inertial-navigation-systems

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California