Development of a 6 Degree-of-Freedom Model for an Autonomous Underwater Vehicle

Modeling and Controlling the Trajectory of a New AUV

Ilana Stern [a,b]

[a] *Applied Physics Labratory, University of Washington, Seattle, WA.*
[b] *The George Washington University, Washington, DC*

August, 2024

Abstract

This paper outlines the development and product of a six degree of freedom Python model for the tracking and control of a new Autonomous Underwater Vehicle developed at the Applied Physics Laboratory at the University of Washington. The final product is a force model which generates the acceleration of the vehicle and an application that uses PID closed-loop controls to reach target positioning and allows the vehicle to follow mission event files.

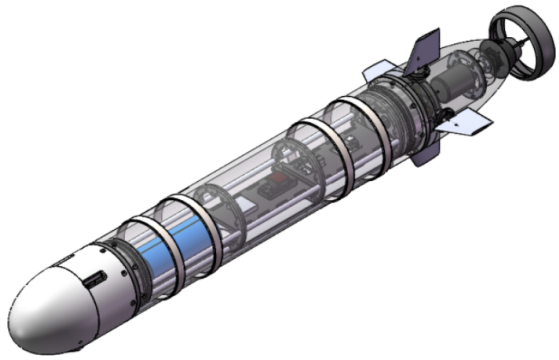

Figure 1: CAD Rendering of the Engineering Test Vehicle

1. Introduction

The first Underwater Autonomous Vehicle (AUV) was developed at the University of Washington's Applied Physics Lab (APL) in 1957 [4]. Nearly 70 years later, the Engineering Test Vehicle (ETV) was developed in the same lab. The ETV is an unmanned submersible vehicle designed to test oceanographic data collection methods. The ETV was designed and built 100% in house at the APL during Spring and Summer 2024.

The ETV is similar in shape to other commercial AUVs, featuring four steering fins at the tail of the vehicle, a propeller capable of a maximum speed of 5 m/s, and a cylindrical acrylic hull with an elliptical nose cone and a tapered tailcone, as seen in Fig.1. The ETV is around 100 pounds in weight, 8 inches in diameter, and 68 inches in length.

Currently, few models can accurately depict the flight characteristics of AUVs, and because the ETV is a brand-new vehicle, no such model exists at all. For the ETV, we are interested in understanding this vehicle's behavior before it is deployed to help develop control algorithms for planning research missions and to identify potential areas for optimization in the AUV's design, should the vehicle be modified in the future. This paper outlines the creation of an AUV trajectory model, and is created specifically for the ETV, but is general enough for use on other vehicles. The program was created in the Python coding language and accounts for the linear and angular positioning and speed of the vehicle, thus accounting for six degrees of freedom in its trajectory. The software includes a force model which solves for the acceleration of the vehicle, and applications to run open and closed-loop control. The closed-loop control application is able to guide the AUV to target positions and execute mission event files for collecting data in the field.

2. AUV Modeling

We model the AUV's location in the linear X, Y, and Z directions, as well as the angular roll (rotation around the X-axis), pitch (rotation around the Y-axis), and yaw (rotation around the Z-axis).

Almost all vectors used in calculations have the dimensions of 6×1 to represent the six degrees of freedom in the movement of the AUV, with the first three elements representing the linear components of movement and the last three components representing the angular. The velocity vector of

this AUV is depicted as

$$\text{General Velocity Vector} = \nu = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{1}$$

with u, v, w representing the linear velocities in the X, Y, and Z directions, and p, q, r representing the roll, pitch and yaw velocities.

The forces are

$$\text{General Force Vector} = \mathbf{F} = \begin{bmatrix} X \\ Y \\ Z \\ K \\ M \\ N \end{bmatrix} \tag{2}$$

with X, Y, Z representing the forces in the X, Y, and Z directions and K, M, N representing the torque around the roll, pitch, and yaw axes.

2.1. Coordinate Frames

We will work within two reference frames, the inertial Earth frame and the non-inertial body frame of the AUV. In the body frame, the origin of our coordinate system is the center of buoyancy, and all other coordinates are calculated about the center of buoyancy. In this frame, X points towards the nose of the AUV, Y points to its right side, and Z points to the 'belly' of the vehicle. In the inertial Earth frame, X points north, Y points east, and Z points downward towards the center of the Earth, and the origin is the AUV's starting position. In our model, forces are computed internally in the body-frame of the AUV, however the force model outputs acceleration in the body frame and velocity in the inertial frame.

Given an orientation of some roll (ϕ), pitch (θ), and yaw(ψ), the rotation of a linear position, velocity, or force vector from the inertial frame to body frame can be expressed with matrix multiplication shown in Fig. 3. This matrix first rotates the X-axis, then Y-axis, then Z-axis. To transform the vector back to the inertial frame, the inverse of Fig. 3 is used. When creating the model in Python, Scipy's rotation program `from_euler()` can be used. To rotate an angular velocity ω from the body frame to the inertial frame, we use matrix multiplication as shown in the following:

$$\begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \times \omega_{\text{body}} = \omega_{\text{inertial}} \tag{3}$$

Positive roll is defined as clockwise AUV rotation ("right wing down"), positive yaw is defined as an AUV rotation from the positive x-axis towards the positive y-axis, and positive pitch is defined as AUV rotation from the positive x-axis towards the negative z-axis.

2.2. Mass and Added Mass

The force model uses Newton's second law to solve for the acceleration of the AUV, allowing the modeling of its trajectory. This equation, shown in Eq. 4 requires a 6×6 mass and inertia matrix (M) and a 6×6 added mass matrix (M_a).

$$a = (M + M_a)^{-1} \times \mathbf{F}_{\text{net}} \tag{4}$$

While a typical mass matrix contains only 6 terms along the main diagonal, we must include off-diagonal terms to represent the center of gravity's offset from the center of buoyancy, as seen in Fig. 2, where x_g, y_g and z_g are the offsets of the center of gravity from the center of buoyancy [2].

Because the AUV is moving through water, we must also account for added mass. Added mass refers to the increased inertia that a body moving through a fluid has due to the fact that some liquid surrounding the body moves with the vehicle, increasing its observed mass. Our added mass matrix was created using the framework laid out in Bentes and Watts [2][6]. We note that many of these terms equal zero, but the matrix is important to include to ensure that our AUV model accelerates correctly.

3. Theory

When modeling the forces and torques that act on an AUV, we divide them into five primary categories; forces and torques from hydrostatics (buoyancy and gravity), lift and drag from fin deflection, thrust and torque from the propeller, damping due to hydrodynamic drag, and the Coriolis and centrifugal forces due to the vehicle's rotational motion. As mentioned in the previous section, each force function detailed below will produce a 6×1 vector in the form of Eq. 2.

3.1. Hydrostatic Forces

AUVs are generally made positively buoyant so that after its mission has finished the vehicle can return to the surface for retrieval. The gravitational force on the AUV is mg, where m is the mass and g is the acceleration due to gravity which acts downwards. The buoyancy force opposes gravity, and is $\rho g V \approx mg$, where ρ is the density of water and V is the AUV's displacing volume. The ETV has a gravitational and buoyant force of around

$$M = \begin{bmatrix} m & 0 & 0 & 0 & m \cdot z_g & -m \cdot y_g \\ 0 & m & 0 & -m \cdot z_g & 0 & m \cdot x_g \\ 0 & 0 & m & m \cdot y_g & -m \cdot x_g & 0 \\ 0 & -m \cdot z_g & m \cdot y_g & I_x & I_{xy} & I_{xz} \\ m \cdot z_g & 0 & -m \cdot x_g & I_{xy} & I_y & I_{yz} \\ -m \cdot y_g & m \cdot x_g & 0 & I_{xz} & I_{yz} & I_z \end{bmatrix}$$

$$M_a = \begin{bmatrix} \dot{X}_u & \dot{X}_v & \dot{X}_w & \dot{X}_p & \dot{X}_q & \dot{X}_r \\ \dot{Y}_u & \dot{Y}_v & \dot{Y}_w & \dot{Y}_p & \dot{Y}_q & \dot{Y}_r \\ \dot{Z}_u & \dot{Z}_v & \dot{Z}_w & \dot{Z}_p & \dot{Z}_q & \dot{Z}_r \\ \dot{K}_u & \dot{K}_v & \dot{K}_w & \dot{K}_p & \dot{K}_q & \dot{K}_r \\ \dot{M}_u & \dot{M}_v & \dot{M}_w & \dot{M}_p & \dot{M}_q & \dot{M}_r \\ \dot{N}_u & \dot{N}_v & \dot{N}_w & \dot{N}_p & \dot{N}_q & \dot{N}_r \end{bmatrix}$$

Figure 2: The Mass and Inertia Matrix (M) and the Added Mass Matrix (M_a)

$$\begin{bmatrix} \cos\theta\cos\phi & \cos\theta\sin\phi & -\sin\theta \\ -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \sin\psi\cos\theta \\ \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi & -\sin\psi\cos\theta + \cos\psi\sin\theta\sin\phi & \cos\psi\cos\theta \end{bmatrix} \times F_{inertial} = F_{body}$$

Figure 3: Rotating a force F from the inertial to body frame

440 N, but is likely to be ballasted by about one pound of positive force.

Before we can apply these forces to our AUV, we must rotate these forces out of the inertial earth frame into the body frame of the AUV. The matrix rotation is expressed in Fig. 3. After rotation, the gravitational and buoyant forces are referred to as $\mathbf{F}_g$ and $\mathbf{F}_b$, respectively.

The torque from gravity acting on the center of gravity is expressed as the cross product of the center of gravity (with respect to the center of buoyancy) and the body-frame gravitational force:

$$\mathbf{r}_{\mathrm{cg}} \times \mathbf{F}_{\mathrm{g}} = \begin{bmatrix} x_{\mathrm{cg}} \\ y_{\mathrm{cg}} \\ z_{\mathrm{cg}} \end{bmatrix} \times \begin{bmatrix} F_{x\mathrm{g}} \\ F_{y\mathrm{g}} \\ F_{z\mathrm{g}} \end{bmatrix} = \begin{bmatrix} \tau_{x\mathrm{g}} \\ \tau_{y\mathrm{g}} \\ \tau_{z\mathrm{g}} \end{bmatrix} \quad (5)$$

Similarly, the torque from buoyancy can be expressed by the cross product of the center of buoyancy and the body-frame buoyancy force:

$$\mathbf{r}_{\mathrm{cb}} \times \mathbf{F}_{\mathrm{b}} = \begin{bmatrix} x_{\mathrm{cb}} \\ y_{\mathrm{cb}} \\ z_{\mathrm{cb}} \end{bmatrix} \times \begin{bmatrix} F_{x\mathrm{b}} \\ F_{y\mathrm{b}} \\ F_{z\mathrm{b}} \end{bmatrix} = \begin{bmatrix} \tau_{x\mathrm{b}} \\ \tau_{y\mathrm{b}} \\ \tau_{z\mathrm{b}} \end{bmatrix} \quad (6)$$

The total force on the AUV from hydrostatics is $\mathbf{F}_g + \mathbf{F}_b = \mathbf{F}_h$. The total torque from hydrostatics is the sum of Eq. 5 and Eq. 6, yielding a 3×1 vector τ_h. To create the force and moment vector, we create a 6×1 vector comprised of $\mathbf{F}_h$ stacked on top of τ_h, which we label as $\mathbf{F}_{\mathrm{H}}$.

3.2. Fin Lift and Drag

The ETV has four identical fins: a rudder on the top and bottom of the AUV which control heading, and an elevator fin on the left and right sides of the AUV which control depth. The area of each fin is 0.03 m^2, and the airfoil of each fin is the NACA-0009 9.0% smoothed airfoil [1]. From the airfoil information, we were able to determine the relationship between fin deflection angle (α) and the related lift and drag values. α is the effective angle of attack here, as we have estimated that the slight linear slip the AUV may exhibit will not have a meaningful effect. The drag values calculated here only account for drag caused by fin deflection. Undeflected fins also cause drag, but this drag is accounted for in the damping matrix generated in Subsection 3.4, where Computational Fluid Dynamics was run for the body of the ETV with undeflected fins.

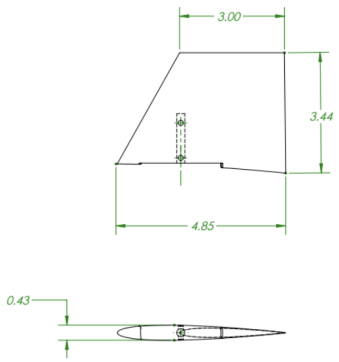

Figure 4: Front and Side View of Fins

To determine the value of the lift acting on a single fin, we first find the value of Cl, the value representing the coefficient of lift at a given α based on the airfoil. The lift force value can then be

expressed as

$$\lambda = 0.5\rho \mathbf{v}^2 \times A|\sin\alpha| \times Cl \tag{7}$$

where $\mathbf{v}$ is the speed of the AUV along the x-axis and A is the area of the fin.

The value of drag for a single fin is

$$\delta = 0.5\rho \mathbf{v}^2 \times A|\sin\alpha| \times Cd \tag{8}$$

where Cd is the coefficient of drag at a given fin deflection. The torque on each fin is found by taking the cross product of the fin's center of pressure (area) coordinates and the sum of the lift and the drag forces on that fin.

For the elevators, we defined a fin deflection of positive α to cause a positive pitch of the AUV. A positive fin deflection of the rudders causes a positive yaw of the AUV- a rotation towards the right. Regardless of the sign of α, the drag force always acts in the negative X-direction.

To find the total force and torque acting on the AUV due to the fins, we calculate a force and torque vector for each fin at its current deflection. We then sum all four vectors, resulting in a 6×1 vector in the notation of F (Eq. 2). We call this new fin force/torque vector $\mathbf{F}_{\mathrm{F}}$.

3.3. Propeller Forces

To find the force generated by the propeller, we adapted data from a similar propeller and scaled it to fit our vehicle. Forward thrust is modeled as a 2-D dependence on propeller revolutions per minute (RPMs) and through-water speed. For our propeller we believe it will generate approximately 66 N of thrust at 880 RPM and 2 m/s of through-water speed. We assume an RPM, speed-thrust dependence that follows the general form given by curves supplied by a vendor for a different thruster, shown in Fig. 5. We then scaled these curves to fit our propeller data, and created a function that determines the thrust based on current RPMs and speed of the AUV. Thrust only acts in the positive X-direction.

This is admittedly a very rough approximation, but RPM-to-thrust is a minor concern; the model is primarily focused on vehicle attitude behavior and ability to hold depth and heading.

Torque generated by the propeller is small but not negligible. Based on Prestero's AUV, we estimate that for every revolution per minute of propeller speed, a roll torque of -0.0006 Nm is generated [5]. The propellers force/torque vector is referred to as $\mathbf{F}_{\mathrm{P}}$.

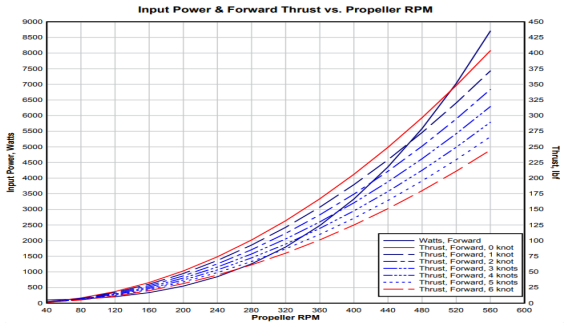

Figure 5: Prop. Thrust Model for Basis of our Thrust Force Model

3.4. Damping terms

Movement underwater is limited by water resistance, which acts in the opposite direction of movement, thus damping the velocity of the AUV. While damping can be analytically derived through approximations and generalizations, we chose to use Computational Fluid Dynamics (CFD) for increased accuracy. We used ANSYS/Fluent to calculate the forces and moments on the vehicle for water flow coming from the forward, side, top, and bottom of the ETV. The resulting force/torque values were calculated in all 6 degrees of freedom (X, Y, Z, K, M and N forces), and analysis of these forces and moments were calculated at a range of speeds. CFD were run for both the top and bottom of the AUV due to asymmetry between the top and bottom of the hull due to a protruding device used to measure the AUV's velocity. We fit each damping curve in both the linear and quadratic regimes to determine coefficients of damping.

In code, we created a matrix of all quadratic damping terms and another matrix of all linear terms. The quadratic matrix is:

$$D_q = \begin{bmatrix} X_{uu} & X_{vv} & X_{ww} & X_{pp} & X_{qq} & X_{rr} \\ Y_{uu} & Y_{vv} & Y_{ww} & Y_{pp} & Y_{qq} & Y_{rr} \\ Z_{uu} & Z_{vv} & Z_{ww} & Z_{pp} & Z_{qq} & Z_{rr} \\ K_{uu} & K_{vv} & K_{ww} & K_{pp} & K_{qq} & K_{rr} \\ M_{uu} & M_{vv} & M_{ww} & M_{pp} & M_{qq} & M_{rr} \\ N_{uu} & N_{vv} & N_{ww} & N_{pp} & N_{qq} & N_{rr} \end{bmatrix}$$

The linear damping matrix is:

$$D_l = \begin{bmatrix} X_{uL} & X_{vL} & X_{wL} & X_{pL} & X_{qL} & X_{rL} \\ Y_{uL} & Y_{vL} & Y_{wL} & Y_{pL} & Y_{qL} & Y_{rL} \\ Z_{uL} & Z_{vL} & Z_{wL} & Z_{pL} & Z_{qL} & Z_{rL} \\ K_{uL} & K_{vL} & K_{wL} & K_{pL} & K_{qL} & K_{rL} \\ M_{uL} & M_{vL} & M_{wL} & M_{pL} & M_{qL} & M_{rL} \\ N_{uL} & N_{vL} & N_{wL} & N_{pL} & N_{qL} & N_{rL} \end{bmatrix}$$

Our CFD model only accounts for damping due to movement in the u, v, and w directions, and

does not account for damping due to rotational motion. We are most interested in the damping terms along the main diagonal of the damping matrix: K_{pp} (damping in the K direction due to change in roll), M_{qq} (damping in the M direction due to change in pitch), or N_{rr} (damping in the N direction due to change in yaw). We adopted values for these terms from Prestero's 6DOF AUV model, which features a comparably shaped AUV [5]. Prestero's model only features quadratic fit for these terms, so the diagonal angular damping terms have no linear counterpart.

To compute damping values, the linear damping matrix is multiplied by the AUV's current linear and angular velocity vector, ν. Similarly, the quadratic damping matrix is multiplied by a vector of the current velocities multiplied by their absolute values. This ensures that the quadratic damping terms retain their polarity and act in the intended direction.

$$D_q \times \begin{bmatrix} u \cdot |u| \\ v \cdot |v| \\ w \cdot |w| \\ p \cdot |p| \\ q \cdot |q| \\ r \cdot |r| \end{bmatrix} + D_l \times \nu = \mathbf{F_D}$$

The resulting vector is composed of the X, Y, and Z damping forces in the first three elements and the K, M, and N damping torques in the last three elements.

3.5. Coriolis and Centrifugal forces

Because our objects movement is expressed in the non-inertial body frame of our rotating AUV, we must include the fictitious Coriolis and centrifugal forces. The general equations for the Coriolis and centrifugal force are

$$\mathbf{F}_{\text{Coriolis}} = -2m(\mathbf{\Omega} \times \mathbf{v}) \tag{9}$$

$$\mathbf{F}_{\text{Centrifugal}} = m(\mathbf{\Omega} \times (\mathbf{\Omega} \times \mathbf{r})) \tag{10}$$

where $\mathbf{\Omega}$ is the angular velocity of the rotating reference frame, $\mathbf{v}$ is the velocity of the object within the rotating reference frame, and $\mathbf{r}$ is the position vector of the object relative to the axis of rotation. Per Bentes and Fossen, these forces can be combined into matrix form as seen in Fig. 6, where x_g, y_g, and z_g are the X, Y, and Z coordinates of the center of gravity in relation to our origin [2] [3].

To find the total effect of these forces on our AUV, we must also factor in the effect of added mass. Thus, the total equation for Coriolis force is

$$(M_a \times \nu - C) \times \nu = \mathbf{F}_{\text{C}} \tag{11}$$

3.6. Total Force Calculations

The above calculations yield five 6×1 vectors. The total force/torque vector can be expressed as

$$\mathbf{F}_{\text{net}} = \mathbf{F}_{\text{H}} + \mathbf{F}_{\text{F}} + \mathbf{F}_{\text{P}} + \mathbf{F}_{\text{D}} + \mathbf{F}_{\text{C}}$$

Using Newton's famed second law ($\mathbf{F} = m \times a$), we can extract the acceleration of our AUV by premultiplying the inverse of the sum of our mass and added mass matrices with the total forces and torques, as shown in Eq. 4.

The product is a 6×1 acceleration vector, in which the first three elements are the acceleration along the X, Y, and Z axis, and the last three elements are the roll, pitch, and yaw acceleration. These accelerations are in the body frame. Eq. 4 is the core equation of this program, and the integration of acceleration allows us to model the flight characteristics as a function of time.

4. Model Dynamics and Controls

Our implementation is divided into three major components: the force model, the integrator, and two control system applications. The force model and integrator are represented in a single Python module, which is callable from an open-loop demonstrator or closed-loop controller. In addition, the specific parameters for the target vehicle are encoded outside of the force model, so this model can be easily altered for the modeling of other AUVs. Additionally, any external control system can call and use the force model due to its modular nature.

4.1. Force Model

This model computes the current forces/torques and determines the acceleration acting on the AUV by using Eq. 4 . The input of the force model is the position of the vehicle in the inertial frame and the velocities in the body frame, as well as the propeller RPM and fin deflection angles. The force model rotates the position into the body frame of the AUV, then calculates the value of $\mathbf{F}_{\text{net}}$ to find the total acceleration of a, as shown in Eq. 4 The model then rotates the velocity back into the inertial frame, and returns a vector containing the inertial velocities and body-frame accelerations.

4.2. The Integrator

A 4th order Runge-Kutta (RK4) integrator is used to determine the position and speed of the AUV along all 6 degrees of freedom from the current velocity and acceleration. At each time step, the output of the force model is fed into the RK4, where the RK4 integrates these values twice to extract the current position from velocity and the current

$$\begin{bmatrix} 0 & 0 & 0 & m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ 0 & 0 & 0 & m(y_g p + w) & -m(y_g p - w) & -m(y_g r - u) \\ 0 & 0 & 0 & m(z_g p - v) & -m(z_g r + x_g p) & m(z_g p + y_g q) \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) & 0 & I_{yz} q + I_{xx} p - I_{xz} r & I_{xz} r + I_{xy} p - I_y q \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) & -I_{yz} q - I_{xx} p + I_{xz} r & 0 & -I_{xz} r - I_{xy} q + I_x p \\ m(x_g r + v) & m(y_g r - u) & -m(z_g q + u) & I_{xz} r + I_{xy} q - I_y q & -I_{xz} r - I_{xy} q + I_x p & 0 \end{bmatrix}$$

Figure 6: Coriolis and Centrifugal Force Matrix, C

velocity from acceleration. The output of the integrator is a vector containing the inertial position values and body-frame velocity values. The RK4 method updates the solution from y_n at time t_n to y_{n+1} at time $t_{n+1} = t_n + h$ by calculating the four "slopes",

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right)$$

$$k_4 = f(t_n + h, y_n + h k_3)$$

Then updating the solution:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

The RK4 integrates once over the given step size h (generally 0.1 seconds). The RK4 is run again with new accelerations and velocities calculated by the force model after the time step, constantly looping through for the duration of the AUV's modeled trajectory, generally simulating a few minutes to hours of the vehicle's flight.

4.3. Applications

The model can be exercised using either open or closed-loop controls. Open-loop demonstrating, also called open-loop control, is a method of directing the AUV without feedback, where the instructions for the AUV are predefined before launch as a sequence of time-scripted events. In open-loop demonstrating for this model, fin angles and propeller RPM are determined before the simulations is run and the open-loop control algorithm simply reads the event file and feeds it to the force model. Our application use a `.csv` file to store controller information, but a `.json` or `.txt` could also be used. These commands are based on time duration, where a model event file may look like this:

Closed-loop control is a feedback algorithm in which an actuator response is calculated based on the error between the target and actual state. Similar to open-loop demonstrating, an event file is created which states the target depth, heading, and rpm of the vehicle at given time intervals.

t	RPM	Rudder α	Elevator α
0	0	0	0
20	880	0	0
40	880	5°	0
60	880	5°	3°

Table 1: Example Event File for Open-Loop Demonstration

t	RPM	Target Heading	Target Depth
0	600	90 °	25
20	880	180 °	35
40	880	270 °	35
60	880	360 °	45

Table 2: Example Event File for AUV Closed-Loop Control

Using a PID (Proportional, Integral, Derivative) control algorithm, the difference between the AUV's target and current states at each control step is calculated. It makes adjustments to the fin positions at some predetermined interval to control trajectory. PID controls account and correct for the speed of the vehicle and the individual tendency of the vehicle to deviate from its course.

Our PID control algorithm adjusts the fin positions for target heading, depth, and roll values. Eq. 12 shows how the elevator response is calculated based on some error Δ for the depth control, and Table 3 shows the coefficients used in the three PID control equations.

$$\alpha = K_p \Delta_{depth} + K_i \sum \Delta_{depth} + K_d w \qquad (12)$$

Heading is controlled through rudder deflections and depth is controlled through elevator deflections. Roll is controlled by cross-deflecting the fins to induce K torque. Target roll is always set to zero, and fin cross-deflection is only activated if

	Depth	Yaw	Roll
K_p	0.6	0.3	0.25
K_i	0.005	0.01	0
K_d	0.4	0.2	1

Table 3: PID Coefficients for Depth, Yaw, and Roll control

roll is greater than 1°. This control system features a deadband of 2 meters of depth error and 2° of yaw error, and if the AUV is within this deadband, the fins will rest at a trim value and will not move until the deadband is exited. While both open and closed loop control allows for adjustments in RPMs, we do not close the loop on target speed. This is because we can generally characterize the relationship between RPMs and speed empirically, and this will hold unless the model changes significantly.

The control system iterates through the force model and integrator once per second, whereas the time step of the integrator is generally run at ten iterations per second. The one-second time step is a reasonable rate at which actual fin deflections could be effected. This model uses a nested loop design, where the control algorithm is *not* called for every iteration of the integrator, and the model can update the current position and velocities of the vehicles without updating the actuator states. A sample loop showing the interactions between the force model, the integrator, and the controls is shown in Fig. 7.

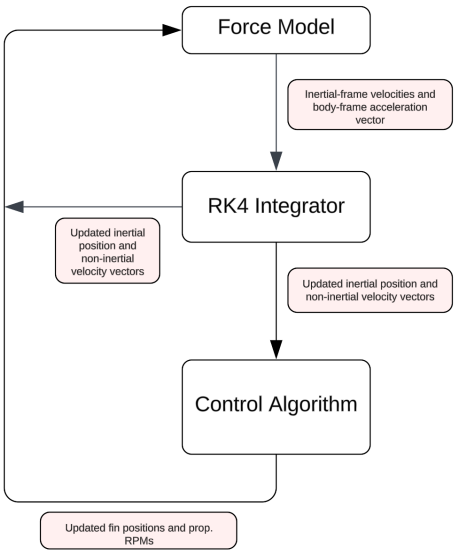

Figure 7: Closed-loop Control Diagram

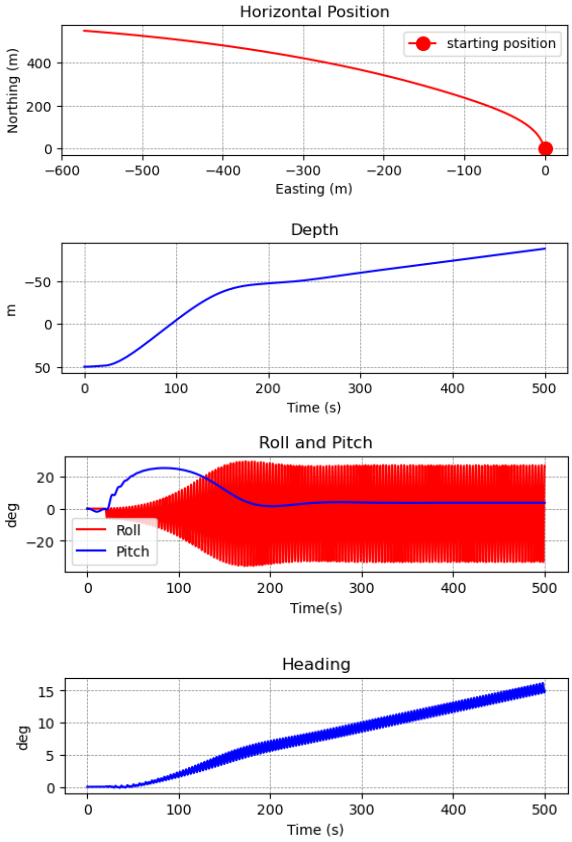

Figure 8: The Linear and Angular Trajectory of AUV at 2 m/s during Open-Loop Control

5. Results

The resulting model suggests a vehicle which may display some instability, but can be controlled.

The model displays a heading bias that causes the vehicle to change heading by roughly 15° in the positive direction (turning to starboard) when the AUV is moving at around 2 m/s with zero fin deflection over a 500 second run time during open-loop controlling (Fig. 8). The AUV climbs 150 meters towards the surface over this duration, likely due to the positive buoyancy of the ETV. We observe intense roll oscillations between 25° and -25°, but roll can be almost completely eliminated with closed-loop control. The cause of the roll and heading instability is likely due to the unequal mass distribution between the head and tail of the vehicle, as well as due to the cross-flow drag terms due to radial asymmetry on the hull.

This instability suggested by the model is likely exaggerated. However, at the time of writing, no flight data in the field has been gathered yet, and until actual flight data is available we will not be certain. Given that the ultimate goal of this work is to develop a robust control system, a model which overestimates instability is in fact a useful

tool - if we can demonstrate control of the more "energetic" modeled vehicle we are much more confident that we will be able to control the vehicle itself.

We are able to control the vehicle's heading, roll, and depth with the PID algorithm. In Fig. 9 and Fig. 10, we ran closed loop controls for a 'lawn-mower' path for about two hours of elapsed time, with the AUV switching between a target heading 90°, 0°, and -90°, and a target depth of 25 m. While we do observe some oscillations in depth, these oscillations only arise after a change in target heading due to the PID controlling settling to the new target positioning. These oscillations are damped after a few minutes of run time at a target position.

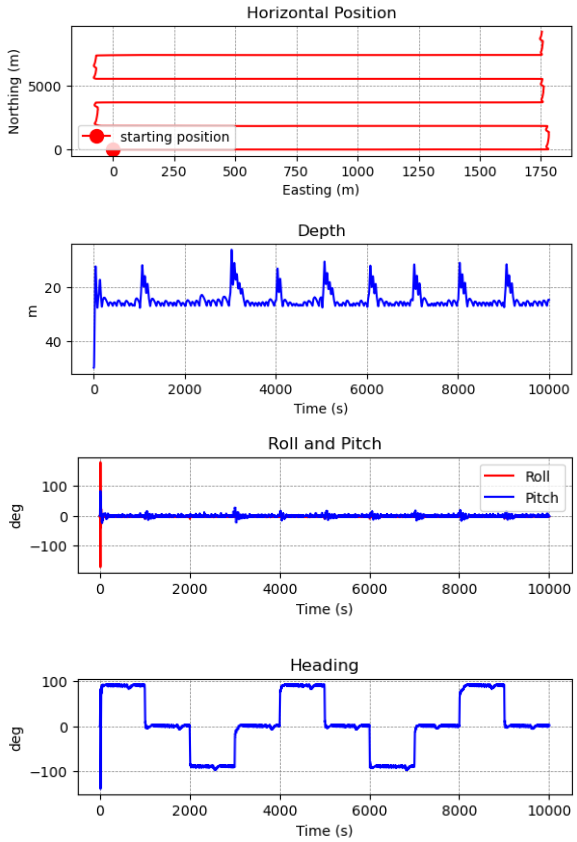

Figure 9: Linear and Angular Trajectory from Closed-Loop Control

These results show that we can control the trajectory of the AUV, and the vehicle will follow event files with acceptable accuracy. By testing various target headings and depths, we observed that the AUV will consistently reach its target states in a timely fashion and stay at those states until a new target state is commanded.

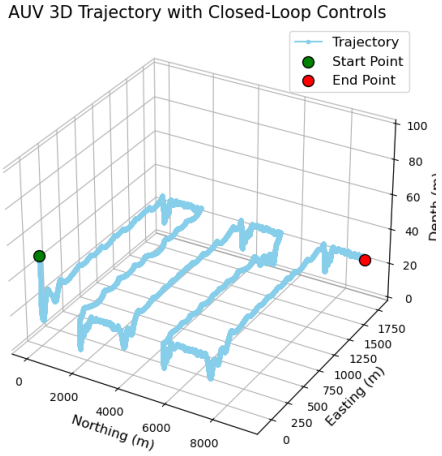

Figure 10: 3D Trajectory from Closed-Loop Control

Acknowledgements

Thanks to Kevin Manganini, for developing the CAD model and supplying all mass, buoyancy, and moments of inertia parameters as well as propeller thrust estimates. Vern Miller generated damping parameters by running a computational fluid dynamics (CFD) model on the vehicle shape. The biggest thank you goes to Peter Brodsky, who mentored and guided me throughout this process from start to finish, and for whom this model would not be possible without.

References

[1]Airfoil Tools, *Airfoil naca 0009sm-il details*, http://airfoiltools.com/airfoil/details?airfoil=n0009sm-il, Accessed: 2024-08-01, 2024.

[2]C. A. Bentes, ≪Modeling of an autonomous underwater vehicle≫, in (2016).

[3]T. Fossen and O. E. Fjellstad, ≪Nonlinear modelling of marine vehicles in 6 degrees of freedom≫, Mathematical Modelling of Systems **1**, 10.1080/13873959508837004 (1995) 10.1080/13873959508837004.

[4]Marine Technology News, *Autonomous underwater vehicles*, https://www.marinetechnologynews.com/articles/marinetechnology/autonomous-underwater-vehicles-100055, Accessed: 2024-08-01, 2024.

[5]T. Prestero, ≪Development of a six-degree of freedom simulation model for the remus autonomous underwater vehicle≫, in , Vol. 1 (Feb. 2001), 450–455 vol.1, ISBN: 0-933957-28-9, 10.1109/OCEANS.2001.968766.

[6]G. D. Watt, *Estimates for the added mass of a multi-component, deeply submerged vehicle*, Tech. Rep. Available online: http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA203234 (DTIC Document, 1988), pp. 23, 50, 52, 53.