Alma Mater Studiorum
Università di Bologna

SCHOOL OF ENGINEERING

-Forlì Campus-

SECOND CYCLE MASTER'S DEGREE in

AEROSPACE ENGINEERING

Class LM-20

GRADUATION THESIS

In Automatic Flight Control

Parameter Identification of the Fossen Model Applied to the UUV Blucy

Candidate:
Raffaele Borgognoni

Supervisor:
Prof. Paolo Castaldi

Co-Supervisor:
Dott. Massimiliano Menghini

Academic year 2022 · 2023

Abstract

This thesis describes the procedure followed to identify the Fossen mathematical model of the unmanned underwater vehicle Blucy developed within the SUstainable fiSHeries wIth DROnes data Processing project. The identification consists in finding the parameters that constitute the Fossen mathematical model of Blucy. These parameters describe important properties of Blucy, such as its mass, geometry and inertia, as well as its hydrostatic and hydrodynamic characteristics. In Chapter 2, the study of the mathematical model is addressed. In Chapter 3, the dynamic vectorial equations of the model are implemented into Simulink. In Chapter 4, the parameters identification is performed through a combination of different tools and methods, including detailed 3D CAD modeling, meshing and CFD simulations. In Chapter 5, the identified parameters are integrated into Simulink and tested. At the end of Chapter 5, a novel approach based on fractional order modeling is applied to test the accuracy and adaptability of a novel strategy for modeling complex systems. The results obtained from this work show the effectiveness of the proposed approach in identifying the Fossen mathematical model of Blucy.

Contents

List of Figures

List of Tables

Acronyms

ASW Anti-Submarine Warfare

AUV Autonomous Underwater Vehicle

CAD Computer-Aided Design

CB Center of Buoyancy

CFD Computational Fluid Dynamics

CG Center of Gravity

CN3 Communication/Navigation Network Nodes

DOF Degree of Freedom

DVL Doppler Velocity Log

EKF Extended Kalman Filter

FANS Favre-Averaged Navier-Stokes

FOG Fiber-Optic Gyroscope

FOV Field of View

GUI Graphical User Interface

HDPE High-Density Polyethylene

IA Influence Activities

ID Inspection/Identification

ISR Intelligence Surveillance and Reconnaissance

MBES Multibeam Echosounder

MCM Mine Countermeasures

MiniCT Miniature Conductivity-Temperature

MiniSVS Miniature Submersible Velocity Sensor

NED North-East-Down

NGC Navigation Guidance and Control

OF OpenFOAM

RANS Reynolds-Averaged Navier-Stokes

ROV Remotely Operated underwater Vehicle

SIMPLE Semi-Implicit Method for Pressure-Linked Equations

SPURV Special Purpose Underwater Research Vehicle

STL Standard Triangle Language

SUSHI DROP SUstainable fiSHeries wIth DROnes data Processing

SW SolidWorks

TCS Time Critical Strike

USBL Ultra-Short Baseline

UUV Unmanned Underwater Vehicle

Chapter 1

Introduction

Unmanned Underwater Vehicles (UUVs), also known as *underwater drones*, are submersible vehicles able to operate without human presence onboard. UUVs may be divided into two categories:

- Remotely Operated underwater Vehicles (ROVs): they are directly controlled by personnel through a tether that provides communication and, in some cases, power;
- Autonomous Underwater Vehicles (AUVs): they don't need a direct link to the ground station via a tether; in other words, they don't require the action of an operator since they are able to accomplish their mission autonomously.

The origins of the development of underwater drones dates back to the 1950s when the first UUV (classified as AUV) was created in the USA by the Applied Physics Laboratory of the University of Washington [29]. The drone was named Special Purpose Underwater Research Vehicle (SPURV), and it was designed to collect oceanographic data in the Arctic waters. Since then, the UUVs market has been rapidly increasing, especially in the last decades, when underwater drones have evolved from being extremely complex and heavy machines for academic research to representing efficient tools for solving a wide range of issues in research, military and commercial applications [5, 12].

As reported in [2, 22], nine are considered to be the main operational fields for underwater vehicles:

1. Intelligence Surveillance and Reconnaissance (ISR);
2. Mine Countermeasures (MCM);
3. Anti-Submarine Warfare (ASW);
4. Inspection/Identification (ID);
5. Oceanography/Hydrography;
6. Communication/Navigation Network Nodes (CN3);
7. Payload Deliver;
8. Influence Activities (IA);
9. Time Critical Strike (TCS).

The UUV named "Blucy", which is the primary subject of the work being discussed, lies in the fifth category.

1.1 Blucy

Blucy is the multipurpose UUV prototype initially developed within the Interreg Italy–Croatia SUstainable fiSHeries wIth DROnes data Processing (SUSHI DROP) project with the objective of supporting sustainable fisheries while promoting biodiversity protection and restoration [16, 20].

In fact, the vehicle is equipped with acoustical and optical technologies to assess the environmental status of habitats, monitor the biodiversity of marine ecosystems, and evaluate the benefits of opto-acoustic surveys in deriving single-species abundance indices for direct input into stock assessments.

To provide higher flexibility, Blucy is designed to be *hybrid*, meaning that it can operate either as a ROV or an AUV: in ROV mode, Blucy needs to be controlled by a human operator who is typically located on a surface

vessel, using a tether that provides communications to the vehicle. This mode allows for real-time control and manipulation of the vehicle. While remotely operated, Blucy is equipped with a $600\,m$ optical fiber cable winch that allows for communication with the operator. In AUV mode, on the other hand, Blucy can operate autonomously by using onboard sensors and software to navigate and carry out pre-programmed missions. This mode allows for extended operation without the need for constant human supervision, making it useful for tasks such as mapping, surveying, environmental monitoring, and, more generally, scientific research. For this purpose, Blucy is equipped with a $24\,V$ battery and a system of sensors aimed at determining the relative positioning of the drone during submarine activities, such as

Fiber-Optic Gyroscope (FOG): gyroscope that uses light waves to sense angular velocity;

Doppler Velocity Log (DVL): underwater acoustic sensor that measures the speed and direction of a moving object relative to the seabed by emitting acoustic pulses that bounce off the seabed and return to the sensor, providing information on the object's speed and direction;

Ultra-Short Baseline (USBL): type of acoustic positioning system used for underwater navigation and tracking. It works by transmitting acoustic signals from a surface vessel to an underwater transponder. The measurement of time needed for the signal to travel allows for accurate position tracking;

Miniature Submersible Velocity Sensor (MiniSVS): underwater sensor used to measure the velocity of water currents. It works by emitting acoustic signals that bounce off suspended particles in the water;

Miniature Conductivity-Temperature (MiniCT): underwater sensor used to measure the salinity and temperature of seawater. It works by measuring the electrical conductivity of seawater, which is related to its salinity, and then using this information to calculate the water's salinity level.

The information acquired by the instruments is processed by Extended Kalman Filter (EKF).

Concerning the scientific data collection, Blucy is provided with the following instruments:

PilotCAM: optical camera installed in the frontal part of the drone that provides a wide Field of View (FOV) video stream for precision navigation, inspection, visual census, and computer vision. The camera's inclination angle can be changed according to the mission's operational scenarios;

BottomCAM: optical camera installed at the bottom of the UUV with a viewpoint towards the seabed (nadir) for photogrammetric purposes. The camera's high-resolution images, LED illuminators, and photogrammetric parameters allow for precise seabed survey planning and modeling;

Multibeam Echosounder (MBES): active acoustic sensor used to produce high-quality bathymetric maps and water column information. MBES can be used for quantitative fisheries stock assessment and habitat mapping.

Furthermore, Blucy is equipped with a set of other instruments and actuators that allow it to perform its tasks; most* of them are reported in Figure 1.1 on the facing page and listed in Table 1.1 on page 16, while Blucy main dimensions are reported in Figure 1.2 on page 17.

1.2 Objectives

As previously mentioned, Blucy is designed to be a hybrid system. However, to date, Blucy has only been tested in its ROV mode, whereas it has not yet been tested in its autonomous mode of operation. To make the submersible drone autonomous, it is necessary to design autopilot controllers, but first, one needs to derive the mathematical model of Blucy.

*Some components are missing in the manual.

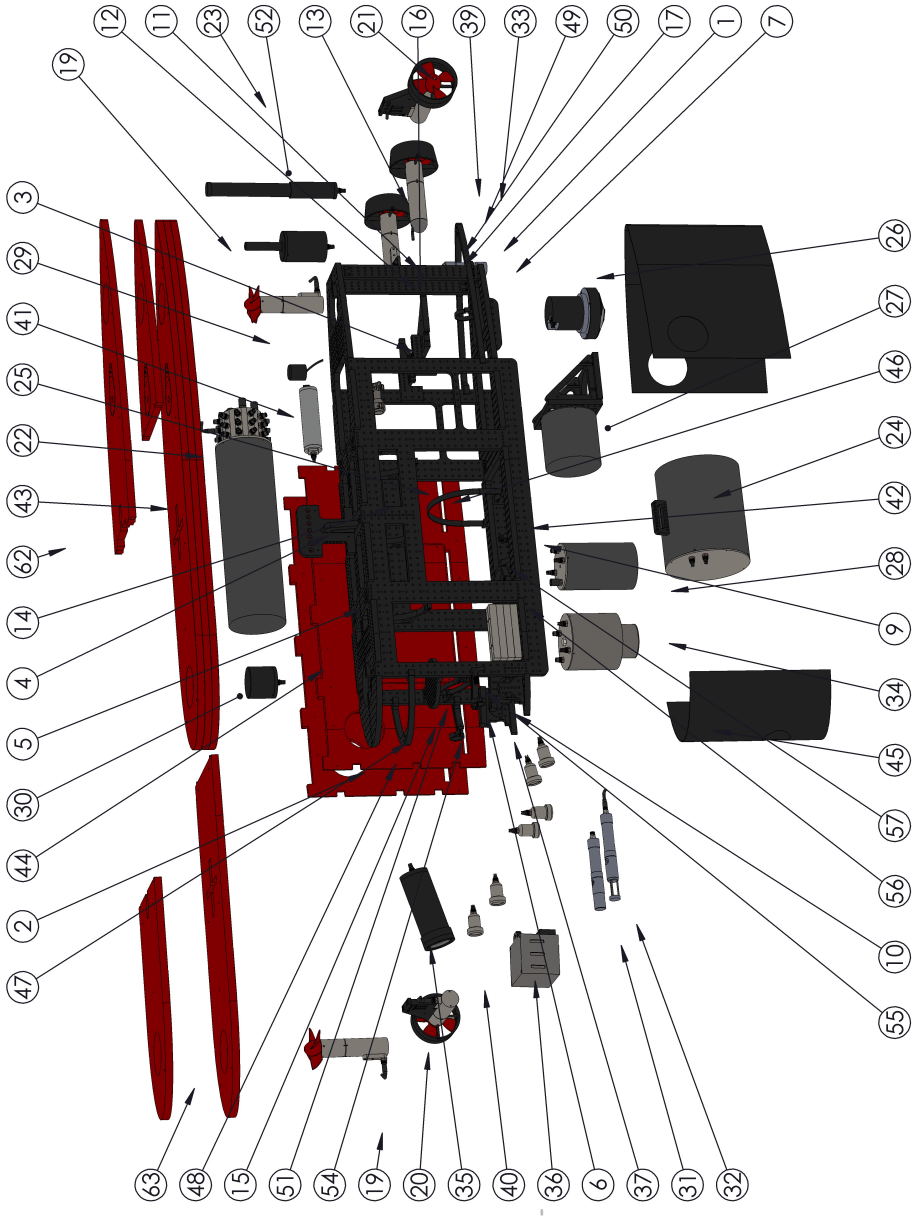


Figure 1.1: Blucy exploded-view [25]

Component	Name	Quantity
1	Structure bottom	1
2	Structure top	1
3	Structure lateral	2
4	Hook	1
5	Structure NGC canister support	2
6	Structure bow	1
7	Structure foot stern	1
9	Support transverse battery	2
10	Support multibeam	2
11	Support main thruster	2
12	Structure vertical stern	2
13	Support main thrusters	2
14	Lifting hook	1
15	Support transverse round bow	1
16	Support lateral thruster	1
19	Thrusters vertical	2
20	Thruster starboard	1
21	Thruster port	1
22	NGC canister	1
23	Groove communication system	1
24	Battery	1
25	Support Battery	1
26	DVL	1
27	FOG	1
28	SIM for SONIC	1
29	USBL transponder	1
30	Microstrain	1
31	MiniCT	1
32	MiniSVS	1
33	Altimeter	1
34	Camera	1
35	Pilot camera	1
36	Multibeam	1
37	Structure support multibeam	2
39	Support Diam. 43	2
40	Led Ageotech	6
41	Evologics S2C	1
42	Support Diam. 40	4
43	Buoyancy foam top	1
44	Buoyancy foam starboard	1
45	Hull bow	1
46	Buoyancy foam port	1
47	Hull support 1	1
48	Hull support 2	1
49	Hull support 3	1
50	Hull support 4	1
51	Pilot camera support	1
52	Radio Modem	1
54	Hull support 5	2
55	Connectors	1
56	Weights	7
57	SIM flange	1
62	Buoyancy foam top 2	1
63	Buoyancy foam top 3	1

Table 1.1: Blucy components

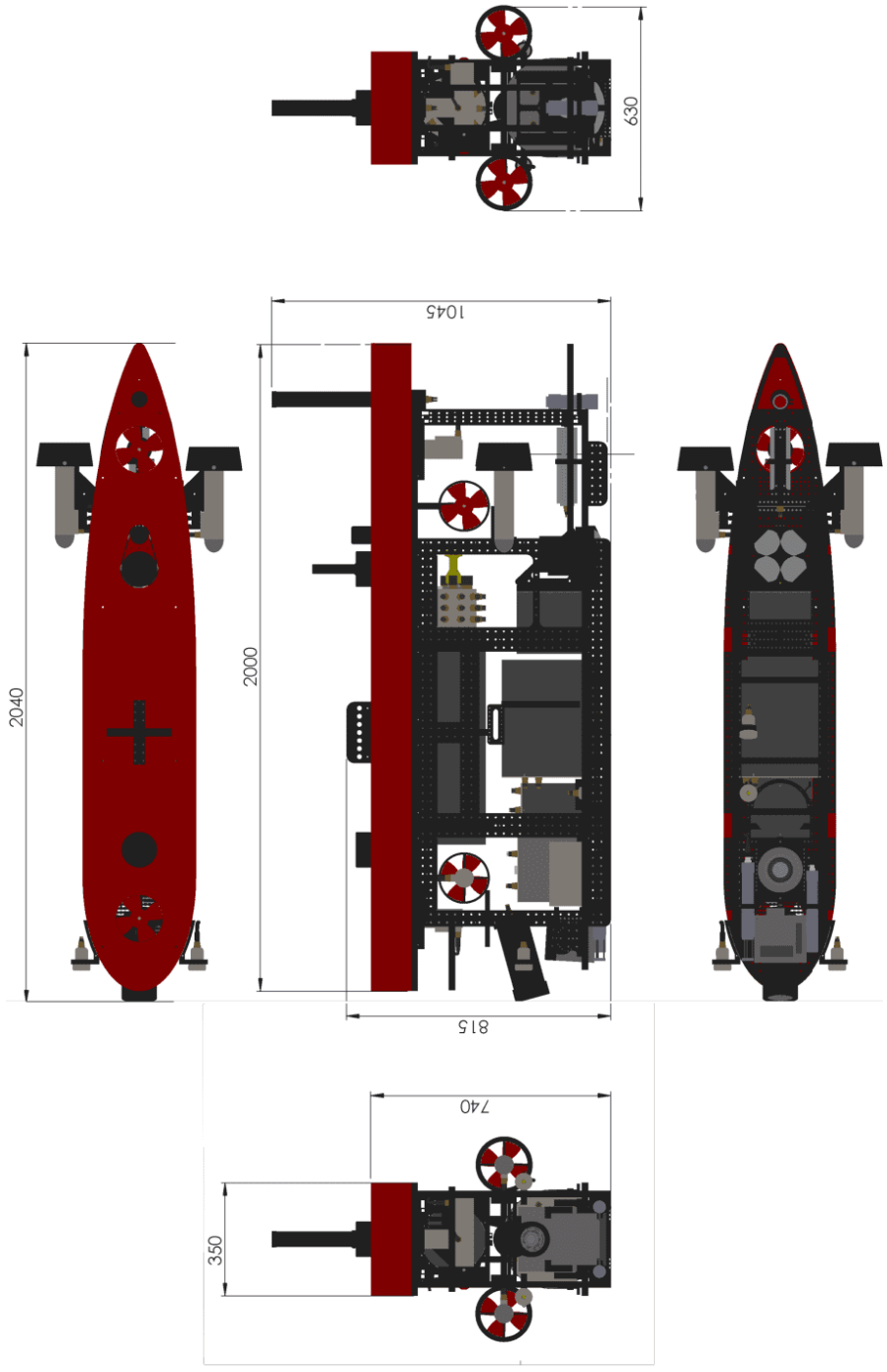


Figure 1.2: Blucy dimensions [25]

The mathematical modeling of underwater vehicles is a complex task that involves a variety of physical and mechanical parameters. While several models have been proposed, the most widely used in the field is the Fossen model [9]. This model has been extensively validated in both simulation and field experiments, and provides a reliable and accurate representation of the dynamics of underwater vehicles [3].

The model is based on six degrees of freedom: *surge*, *sway*, *heave*, *roll*, *pitch* and *yaw*. The equations of motion for each degree of freedom are expressed in terms of the mechanical properties of the vehicle, hydrodynamic coefficients, and external forces and moments acting on the vehicle (such as gravity, buoyancy, etc.). These properties need to be estimated in order to describe Blucy dynamics.

Therefore, this work aims at identifying Blucy mathematical model by estimating all the main parameters necessary to describe the dynamics of the system. Moreover, the so-obtained mathematical model will be incorporated into a simulator, so providing Blucy "virtual" model, since the simulator is essentially a digital representation of the physical vehicle.

Once the mathematical model is determined and the virtual model built, the design of autopilot controllers can proceed[†], paving the way for allowing Blucy to operate autonomously in a variety of underwater environments.

The upcoming chapters will show the procedure followed to generate the virtual model of Blucy. The main steps that will be undertaken are:

- describing the mathematical model;
- building the virtual model in the simulator software;
- presenting the methods used to identify the main parameters of Blucy and the coefficients obtained;
- presenting the first simulation results.

[†]The implementation of the controllers is not covered in the context of this work.

Chapter 2

Mathematical Modeling

As stated in the introduction, in order to make Blucy autonomous, it is necessary to derive its mathematical model, that is the set of equations of motion that describe the behavior of the drone when subjected to external and propulsive forces. Then, it is also necessary to specialize the equations by introducing Blucy characteristic parameters. In this way, the controllers capable of making the vehicle autonomous can actually be designed.

While the aforementioned parameters vary for each underwater vehicle (they, however, may be similar for vehicles with similar geometries and conditions), the equations of motion describing the *dynamics* of a small submersible are generally valid for each geometry and condition, if certain assumptions are made.

The goal of this chapter is, therefore, to underline the main characteristics of the mathematics lying behind the motion of underwater vehicles. For this purpose, the following sections mainly (but not only) focus on the material available in [9], which is considered to be the standard reference for marine craft motion and control systems design.

2.1 Coordinate Systems

To describe the motion of an object in space, it is necessary to introduce at least one system of coordinates. Considering both the literature for underwater

vehicles and the scope of this work, two reference frames will be used:

- **Body**: it is fixed to a point of the hull and moves together with the object. Its axes are defined as $\{b\} = (x_b, y_b, z_b)$ and are directed as in Figure 2.2 on the next page:
  - x_b: longitudinal axis, points toward the *prow*;
  - y_b: transverse axis, points toward the *starboard*;
  - z_b: vertical axis, points toward the *bottom*;
- **North-East-Down (NED)**: this reference frame can be considered to be inertial for most of the UUVs, as long as they navigate at low speeds and within a limited operational range (flat Earth navigation assumption), so that Newton's laws are still applicable. Its origin is fixed to a point on the water surface, and its axes are represented by $\{n\} = (x_n, y_n, z_n)$, of which the first two form a plane tangent to the Earth's reference ellipsoid. They are oriented as in Figure 2.1 on the facing page:
  - x_n points toward the magnetic north;
  - y_n points toward the east;
  - z_n points toward the earth center.

As already said, it is essential to define these two coordinate systems to describe the dynamics of marine crafts: this is because linear and angular velocities, as well as the external forces acting on the vehicle, are most conveniently expressed in terms of body coordinates, whereas it is equally convenient to describe the linear and angular positions with respect to an inertial frame. Thus, the need arises to apply vectorial transformations to pass from one frame of reference to the other. These transformations will be addressed in the following section.

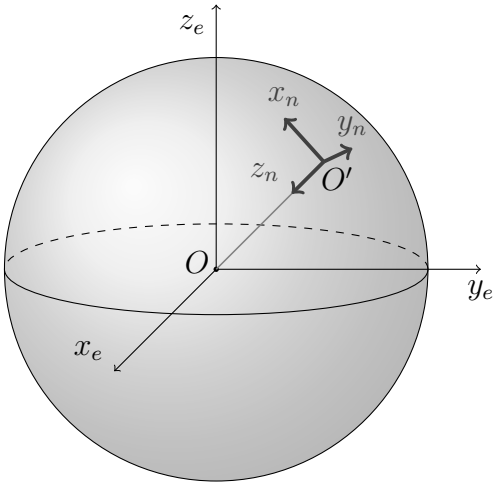


Figure 2.1: The North-East-Down frame

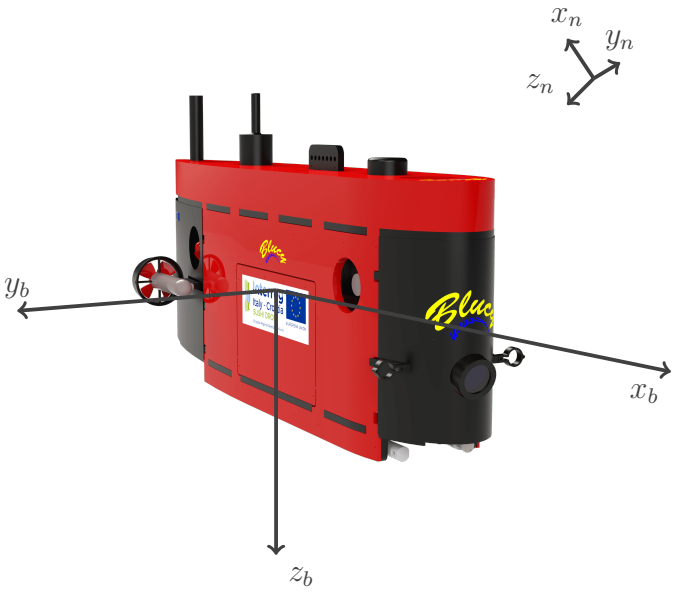


Figure 2.2: The Body-fixed frame

2.2 Kinematics

The *dynamics* of a system can be divided into *kinematics*, which is the study concerning the geometrical aspects of motion, and *kinetics*, which is the study of the forces responsible for the motion. In this section, the kinematics part will be covered.

It is clear that the real motion of a marine craft takes place in six Degrees of Freedom (6DOFs). The DOFs are represented by three displacements (or translations) and three rotations. Concerning the translations, the longitudinal motion parallel to x_b is referred to as *surge*, the sideways motion parallel to y_b is referred to as *sway*, and the vertical motion parallel to z_b is defined as *heave*; whereas the rotations about x_b, y_b, and z_b are named *roll*, *pitch*, and *yaw* respectively.

It is now useful to define the following vectors:

$$\boldsymbol{\eta} = [\, x \;\; y \;\; z \;\; \phi \;\; \theta \;\; \psi \,]^\top \tag{2.1}$$

$$\boldsymbol{\nu} = [\, u \;\; v \;\; w \;\; p \;\; q \;\; r \,]^\top \tag{2.2}$$

and stress out some aspects:

- $\boldsymbol{\eta}$ represents the position of the body frame with respect to the inertial frame in terms of linear displacements (first three entries of the vector) and Euler angles (last three elements);
- $\boldsymbol{\nu}$ represents the linear and angular velocities of the vehicle with respect to the body frame;

These considerations further emphasize the need for transformation matrices to relate vectors between different coordinate systems.

2.2.1 Linear Velocity Transformation

In guidance, navigation and control applications, it is customary to use the following rotation matrix to project the body-fixed linear velocity components into the earth-fixed linear velocity components:

$$\boldsymbol{R}_b^n(\boldsymbol{\eta}_2) = \begin{bmatrix} \mathrm{c}\psi\mathrm{c}\theta & -\mathrm{s}\psi\mathrm{c}\phi + \mathrm{c}\psi\mathrm{s}\theta\mathrm{s}\phi & \mathrm{s}\psi\mathrm{s}\phi + \mathrm{c}\psi\mathrm{c}\phi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\theta\mathrm{s}\psi & -\mathrm{c}\psi\mathrm{s}\phi + \mathrm{s}\theta\mathrm{s}\psi\mathrm{c}\phi \\ -\mathrm{s}\theta & \mathrm{c}\theta\mathrm{s}\phi & \mathrm{c}\theta\mathrm{c}\phi \end{bmatrix} \tag{2.3}$$

where $\boldsymbol{\eta}_2 = [\,\phi\ \theta\ \psi\,]^\top$, $\mathrm{s}\,\cdot = \sin(\cdot)$, $\mathrm{c}\,\cdot = \cos(\cdot)$, and the subscript/superscript notation should be read as $\boldsymbol{R}^{\text{to}}_{\text{from}}$. Analogously, to perform a vector transformation *from* $\{n\}$ *to* $\{b\}$, it is sufficient to take the transpose of the matrix, in fact:

$$\boldsymbol{R}_b^n(\boldsymbol{\eta}_2) = \boldsymbol{R}_n^b(\boldsymbol{\eta}_2)^\top \tag{2.4}$$

One can now relate the linear velocities between body and inertial frames:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R}_b^n(\boldsymbol{\eta}_2) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2.5}$$

2.2.2 Angular Velocity Transformation

In a similar way, one can also relate $\boldsymbol{\nu}_2 = [\,p\ q\ r\,]^\top$ with $\dot{\boldsymbol{\eta}}_2 = [\,\dot{\phi}\ \dot{\theta}\ \dot{\psi}\,]^\top$ through $\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}_2)$:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}_2) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.6}$$

being

$$\boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}_2) = \begin{bmatrix} 1 & \mathrm{s}\phi\mathrm{t}\theta & \mathrm{c}\phi\mathrm{t}\theta \\ 0 & \mathrm{c}\phi & -\mathrm{s}\phi \\ 0 & \mathrm{s}\phi/\mathrm{c}\theta & \mathrm{c}\phi/\mathrm{c}\theta \end{bmatrix} \tag{2.7}$$

where $\mathrm{t}\,\cdot = \tan(\cdot)$. Moreover, (2.6) can be expanded in component form to

obtain the Euler angle attitude equations:

$$\dot{\phi} = p + q\sin(\phi)\tan(\theta) + r\cos(\phi)\tan(\theta) \tag{2.8a}$$

$$\dot{\theta} = q\cos(\phi) - r\sin(\phi) \tag{2.8b}$$

$$\dot{\psi} = q\frac{\sin(\phi)}{\cos(\theta)} + r\frac{\cos(\phi)}{\cos(\theta)} \tag{2.8c}$$

From the relations (2.8), it can be seen that a pitch angle of $\theta = \pm 90^\circ$ leads the matrix $\boldsymbol{T}_{\boldsymbol{\Theta}}$ to be undefined. This condition is never encountered by surface vessels; nevertheless, underwater vehicles may face it; if that happens, two solutions can be adopted:

- use two different Euler angle representations with different singularities and switch between them at need;
- use quaternion representation.

However, experience has shown that Blucy pitch never approaches the singularity angle. Therefore, no further analyses of this matter are needed.

2.2.3 6DOF Kinematic Equations

Summarizing the results from the last two subsections, the 6DOFs kinematic equations can be written in a vectorial form as follows:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.9}$$

Expanding the latter, one obtains:

$$\begin{bmatrix} \dot{\boldsymbol{\eta}}_1 \\ \dot{\boldsymbol{\eta}}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\eta}_2) & \boldsymbol{0}_{3\times 3} \\ \boldsymbol{0}_{3\times 3} & \boldsymbol{T}_{\boldsymbol{\Theta}}(\boldsymbol{\eta}_2) \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}_1 \\ \boldsymbol{\nu}_2 \end{bmatrix}$$

2.3 Rigid-Body Kinetics

It is now necessary to analyze the motion of rigid bodies to derive the marine craft equations of motion. To this aim, the first step is to introduce the

following vectors:

$$\boldsymbol{r}_g = [\, x_g \;\; y_g \;\; z_g \,]^\top \tag{2.10}$$

$$\boldsymbol{r}_b = [\, x_b \;\; y_b \;\; z_b \,]^\top \tag{2.11}$$

$$\boldsymbol{\tau}_{RB} = [\, X \;\; Y \;\; Z \;\; K \;\; M \;\; N \,]^\top \tag{2.12}$$

where:

- $\boldsymbol{r}_g$ represents the position of the Center of Gravity (CG) of the vehicle with respect to the body frame;
- $\boldsymbol{r}_b$ represents the position of the Center of Buoyancy (CB) of the vehicle with respect to the body frame;
- $\boldsymbol{\tau}_{RB}$ is the column vector of the generalized external forces and moments acting on the craft with respect to the body frame.

Moreover, the matrix describing the inertia of the vehicle about its CG can be defined as

$$\boldsymbol{I}_g = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{yx} & I_y & -I_{yz} \\ -I_{zx} & -I_{zy} & I_z \end{bmatrix}, \qquad \boldsymbol{I}_g = \boldsymbol{I}_g^\top > 0 \tag{2.13}$$

where I_x, I_y and I_z are the moments of inertia about the x_b, y_b and z_b axes, and $I_{xy} = I_{yx}$, $I_{xz} = I_{zx}$ and $I_{yz} = I_{zy}$ are the products of inertia defined as

$$I_x = \int_V \left(y^2 + z^2\right) \rho \mathrm{dV} \qquad I_{xy} = \int_V xy\rho \mathrm{dV} = \int_V yx\rho \mathrm{dV} = I_{yx}$$
$$I_y = \int_V \left(x^2 + z^2\right) \rho \mathrm{dV} \qquad I_{xz} = \int_V xz\rho \mathrm{dV} = \int_V zx\rho \mathrm{dV} = I_{zx}$$
$$I_z = \int_V \left(x^2 + y^2\right) \rho \mathrm{dV} \qquad I_{yz} = \int_V yz\rho \mathrm{dV} = \int_V zy\rho \mathrm{dV} = I_{zy}$$

The nonlinear 6DOF rigid-body equations of motion can now be defined in terms of body-fixed coordinates through the following set of coupled

differential equations:

$$\begin{aligned}
m\left[\dot{u} - vr + wq - x_g(q^2 + r^2) + y_g(pq - \dot{r}) + z_g(pr + \dot{q})\right] &= X \\
m\left[\dot{v} - wp + ur - y_g(r^2 + p^2) + z_g(qr - \dot{p}) + x_g(qp + \dot{r})\right] &= Y \\
m\left[\dot{w} - uq + vp - z_g(p^2 + q^2) + x_g(rp - \dot{q}) + y_g(rq + \dot{p})\right] &= Z \\
I_x\dot{p} + (I_z - I_y)qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} & \\
+ m\left[y_g(\dot{w} - uq + vp) - z_g(\dot{v} - wp + ur)\right] &= K \\
I_y\dot{q} + (I_x - I_z)rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{zx} + (qp - \dot{r})I_{yz} & \\
+ m\left[z_g(\dot{u} - vr + wq) - x_g(\dot{w} - uq + vp)\right] &= M \\
I_z\dot{r} + (I_y - I_x)pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{zx} & \\
+ m\left[x_g(\dot{v} - wp + ur) - y_g(\dot{u} - vr + wq)\right] &= N
\end{aligned} \tag{2.14}$$

where m is the constant mass of the vehicle. Moreover, relations (2.14) can be expressed in a vectorial form according to [8] as follows:

$$\boldsymbol{M}_{RB}\,\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\,\boldsymbol{\nu} \;=\; \boldsymbol{\tau}_{RB} \tag{2.15}$$

where $\boldsymbol{M}_{RB}$ is the rigid-body system inertia matrix ($\boldsymbol{M}_{RB} = \boldsymbol{M}_{RB}^{\top} > 0$), and it is defined as follows:

$$\boldsymbol{M}_{RB} = \begin{bmatrix}
m & 0 & 0 & 0 & mz_g & -my_g \\
0 & m & 0 & -mz_g & 0 & mx_g \\
0 & 0 & m & my_g & -mx_g & 0 \\
0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\
mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\
-my_g & mx_g & 0 & -I_{zx} & -I_{zy} & I_z
\end{bmatrix} \tag{2.16}$$

while $\boldsymbol{C}_{RB}$ is the rigid-body Coriolis and centripetal matrix that can be written, according to [11], as:

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} \mathbf{0}_{3\times 3} & -m\boldsymbol{S}(\boldsymbol{\nu}_1) - m\boldsymbol{S}(\boldsymbol{\nu}_2)\boldsymbol{S}(\boldsymbol{r}_g) \\ -m\boldsymbol{S}(\boldsymbol{\nu}_1) + m\boldsymbol{S}(\boldsymbol{r}_g)\boldsymbol{S}(\boldsymbol{\nu}_2) & -\boldsymbol{S}(\boldsymbol{I}_b\,\boldsymbol{\nu}_2) \end{bmatrix} \tag{2.17}$$

where the operator $\boldsymbol{S}(\cdot)$ generates the skew-symmetric matrix of the vector between parentheses:

$$\boldsymbol{S}(\boldsymbol{\lambda}) = -\boldsymbol{S}^{\top}(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \qquad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_3 \\ \lambda_3 \end{bmatrix}$$

Finally, (2.17) can be expanded, according to [10], to illustrate the complexity of the 6DOFs problem:

$$\boldsymbol{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) \end{bmatrix} \tag{2.18}$$

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

2.4 Hydorstatics and Hydrodynamics

Concerning the external forces acting on the craft, they can be divided in:

- hydrostatic forces;
- hydrodynamic forces:
  - linear and nonlinear damping;
  - added mass;
- propulsion forces;
- wind forces;
- wave forces.

This work does not take into account the forces exerted by wind and waves.

2.4.1 Restoring Forces

The restoring forces are *hydrodstatic* forces. They are due to the combined effects of gravity and buoyancy acting on the vehicle. As sketched in Figure 2.3 on the next page, the buoyancy force vector points upwards, while the weight vector points downwards; and they are defined by:

$$W = mg, \quad B = \rho g \nabla$$

where ∇ represents the volume of displaced water, according to Archimedes' principle. Moreover, it is worth mentioning that, in the case of Blucy (and many other ROVs applications), the buoyancy force remains constant and slightly larger than the weight throughout the mission. This, in fact, allows the robot to

- be more easily controllable in heave (requiring less power);
- be statically stable about its roll and pitch axis;
- surface automatically in case of system failure.

As already said at the beginning of Section 2.3, the points of action of weight and buoyancy are r_g and r_b, respectively. It follows that, using Euler angles, one obtains the restoring forces vector for underwater vehicles:

$$\boldsymbol{g}(\boldsymbol{\eta}) = \begin{bmatrix} (W-B)\sin\theta \\ -(W-B)\cos\theta\sin\phi \\ -(W-B)\cos\theta\cos\phi \\ -(y_gW-y_bB)\cos\theta\cos\phi + (z_gW-z_bB)\cos\theta\sin\phi \\ (z_gW-z_bB)\cos\theta\cos\phi + (x_gW-x_bB)\cos\theta\cos\phi \\ -(x_gW-x_bB)\cos\theta\sin\phi - (y_gW-y_bB)\sin\theta \end{bmatrix} \tag{2.19}$$

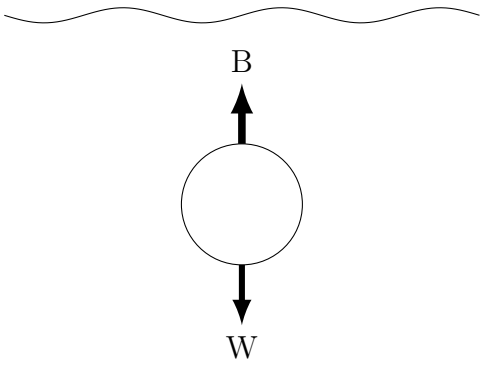


Figure 2.3: Restoring forces

2.4.2 Viscous damping

The *hydrodynamic* damping acting on submersibles is primarily due to:

- potential damping;
- skin friction;
- damping due to vortex shedding;
- lifting forces.

In general, it is difficult to separate these phenomena; therefore it is convenient to simply consider hydrodynamic damping as the sum of linear and nonlinear contributions. They are accounted for by the following matrices:

$$\boldsymbol{D}_l = -\begin{bmatrix} X_u & X_v & X_w & X_p & X_q & X_r \\ Y_u & Y_v & Y_w & Y_p & Y_q & Y_r \\ Z_u & Z_v & Z_w & Z_p & Z_q & Z_r \\ K_u & K_v & K_w & K_p & K_q & K_r \\ M_u & M_v & M_w & M_p & M_q & M_r \\ N_u & N_v & N_w & N_p & N_q & N_r \end{bmatrix} \tag{2.20}$$

and

$$\boldsymbol{D}_n(\boldsymbol{\nu}) = -\begin{bmatrix} X_{|u|u}|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{|v|v}|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & Z_{|w|w}|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{|p|p}|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & M_{|q|q}|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{|r|r}|r| \end{bmatrix} \tag{2.21}$$

To understand the meaning of this notation, one should think of the drag force acting on a moving object: taking, for instance, the longitudinal motion of Blucy, the drag force it experiences is shown in Figure 2.4 on the facing page, and it is clear that the curve can be approximated by a second-order polynomial function [14] of the form:

$$X(u) = X_u u + X_{|u|u} u^2 \tag{2.22}$$

where X_u and $X_{|u|u}$ are, respectively, the linear and quadratic hydrodynamic damping coefficients describing the force variation (parallel to the x-direction of the craft) caused by changes in longitudinal speed.

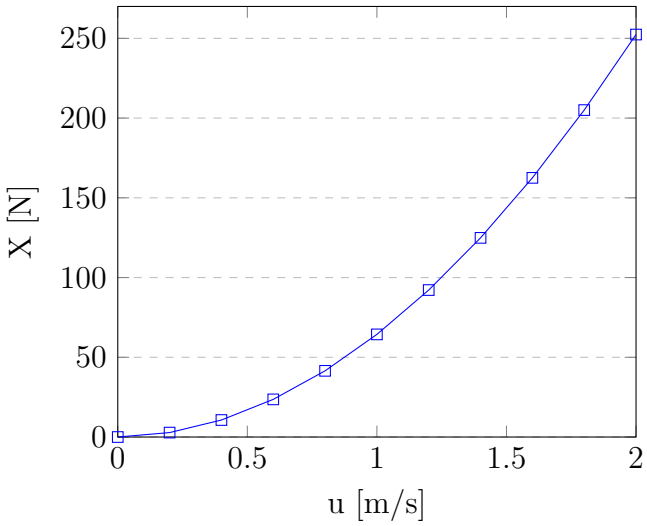


Figure 2.4: Longitudinal drag variation

2.4.3 Added mass

In fluid dynamics, the concept of *added mass* (or *virtual mass*) describes the inertia of the fluid surrounding a moving object. When the density of the medium is much lower than that of the vehicle, the effect can be neglected (this happens, for instance, to aircraft and satellites); on the other hand, when the density of the object and the density of the medium have the same order of magnitude, this phenomenon largely affects the motion of the object and needs to be taken into account. This is clearly the case for submersibles or airships. This fact is visible by simply adding the mass of the displaced fluid in Newton's second law:

$$F = (m_{\text{obj}} + m_{\text{disp}})\, a$$

where m_{obj} is the mass of the object and m_{disp} is the mass of the displaced fluid.

According to [15], the general expressions for added mass comprise 36 constant parameters. These parameters can be used to define the added mass

matrix $\boldsymbol{M}_A$ as follows:

$$\boldsymbol{M}_A = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix}, \qquad \boldsymbol{M}_A = \boldsymbol{M}_A^\top \geq 0 \tag{2.23}$$

where (taking for instance the first element) each value can be defined [13] as

$$X_{\dot{u}} = \left.\frac{\partial X}{\partial \dot{u}}\right|_{\dot{u}=0}$$

When dealing with a real fluid, all the parameters could in principle be different, whereas assuming an ideal (frictionless) medium, one has only 21 independent parameters, being $\boldsymbol{M}_{A_{ij}} = \boldsymbol{M}_{A_{ji}}$. Moreover, according to [9], expression (2.23) can be further simplified in case the AUV is only allowed to move at low speed and presents three planes of symmetry: the contribution of the off-diagonal terms in $\boldsymbol{M}_A$ can be neglected, which yields:

$$\boldsymbol{M}_A = -\text{diag}\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\} \tag{2.24}$$

Of course, engineering judgment must be used to decide whether the previous approximation is suitable for the model or not. Furthermore, for a rigid body moving through an ideal fluid, the hydrodynamic Coriolis and centripetal matrix $\boldsymbol{C}_A(\boldsymbol{\nu})$ can always be parameterized such that it is skew-symmetric:

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = -\boldsymbol{C}_A(\boldsymbol{\nu})^\top \tag{2.25}$$

One parametrization satisfying (2.25) is

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} \boldsymbol{0}_{3\times 3} & -\boldsymbol{S}(\boldsymbol{A}_{11}\boldsymbol{\nu}_1 + \boldsymbol{A}_{12}\boldsymbol{\nu}_2) \\ -\boldsymbol{S}(\boldsymbol{A}_{11}\boldsymbol{\nu}_1 + \boldsymbol{A}_{12}\boldsymbol{\nu}_2) & -\boldsymbol{S}(\boldsymbol{A}_{21}\boldsymbol{\nu}_1 + \boldsymbol{A}_{22}\boldsymbol{\nu}_2) \end{bmatrix} \tag{2.26}$$

where $\boldsymbol{A}_{ij} \in \mathbb{R}^3$ is given by

$$\boldsymbol{M}_A = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \tag{2.27}$$

Finally, (2.25) can be expressed in component form according to

$$\boldsymbol{C}_A(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & 0 & 0 & -a_3 & a_2 \\ 0 & 0 & 0 & a_3 & 0 & -a_1 \\ 0 & 0 & 0 & -a_2 & a_1 & 0 \\ 0 & -a_3 & a_2 & 0 & -b_3 & b_2 \\ a_3 & 0 & -a_1 & b_3 & 0 & -b_1 \\ -a_2 & a_1 & 0 & -b_2 & b_1 & 0 \end{bmatrix} \tag{2.28}$$

where

$$\begin{aligned} a_1 &= X_{\dot{u}}u + X_{\dot{v}}v + X_{\dot{w}}w + X_{\dot{p}}p + X_{\dot{q}}q + X_{\dot{r}}r \\ a_2 &= Y_{\dot{u}}u + Y_{\dot{v}}v + Y_{\dot{w}}w + Y_{\dot{p}}p + Y_{\dot{q}}q + Y_{\dot{r}}r \\ a_3 &= Z_{\dot{u}}u + Z_{\dot{v}}v + Z_{\dot{w}}w + Z_{\dot{p}}p + Z_{\dot{q}}q + Z_{\dot{r}}r \\ b_1 &= K_{\dot{u}}u + K_{\dot{v}}v + K_{\dot{w}}w + K_{\dot{p}}p + K_{\dot{q}}q + K_{\dot{r}}r \\ b_2 &= M_{\dot{u}}u + M_{\dot{v}}v + M_{\dot{w}}w + M_{\dot{p}}p + M_{\dot{q}}q + M_{\dot{r}}r \\ b_3 &= N_{\dot{u}}u + N_{\dot{v}}v + N_{\dot{w}}w + N_{\dot{p}}p + N_{\dot{q}}q + N_{\dot{r}}r \end{aligned} \tag{2.29}$$

2.5 Robot-Like Vectorial Model

Combining together the expressions and definitions given in the previous sections, the underwater vehicle equations of motion can be written in a vectorial setting according to [10]:

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}_{\boldsymbol{\Theta}}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.30}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}} \tag{2.31}$$

where

$$\boldsymbol{M} = \boldsymbol{M}_{RB} + \boldsymbol{M}_{A} \tag{2.32}$$

$$\boldsymbol{C} = \boldsymbol{C}_{RB} + \boldsymbol{C}_{A} \tag{2.33}$$

By manipulating (2.31), one obtains

$$\dot{\boldsymbol{\nu}} = \boldsymbol{M}^{-1}[\,\boldsymbol{\tau} - \boldsymbol{g}(\boldsymbol{\eta}) - \boldsymbol{C}(\boldsymbol{\nu})\,\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\,\boldsymbol{\nu}\,] \tag{2.34}$$

In conclusion, (2.34) represents the equation that will be implemented in the simulator to describe Blucy behavior.

Chapter 3

Simulink Model

This chapter covers the main steps followed to build the virtual model of Blucy (according to the Fossen model previously described) using MATLAB and Simulink, the MATLAB-based graphical programming environment for modeling [1] (hereafter, the software will be referred to as Matlab).

In Figure 3.1 on the next page is depicted the outermost block in which all other blocks and functions are contained. On the left of the figure, the six external control *inputs* can be seen: they are the forces and moments generated by the thrusters and directly acting on the UUV. It is important to note that this is a simplified representation that doesn't take into account the position nor the characteristics of the thrusters: it only accounts for the force and moment resultants produced by the propulsion system. Furthermore, on the right side of the block, the six *outputs* representing the state vector $\boldsymbol{\nu}$ can be found.

The level immediately below the first block is shown in Figure 3.2 on the following page. It contains two boxes: the one on the left side describes Blucy *kinematics*, while the one on the right side describes the *kinetics* of the vehicle.

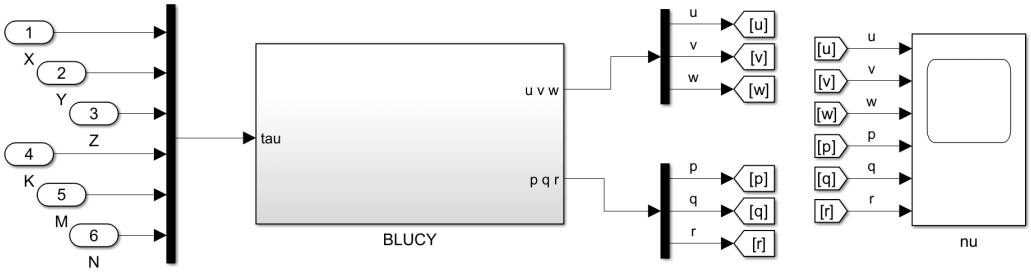


Figure 3.1: View of Blucy external block

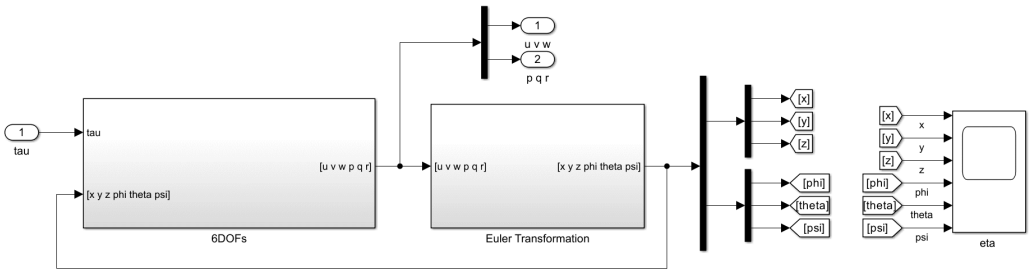


Figure 3.2: View of the two main blocks of the model

3.1 Kinematics Block

The purpose of the "Euler Transformation" block in the Simulink model is to perform the transformation described by equation (2.9). In fact, the block relates the body linear and angular velocity variables to the time derivative of the position vector of the NED frame. The schematic block diagram of the transformation is depicted in Figure 3.3 on the next page.

As already discussed in Section 2.2, both the linear and angular velocities need to be related between $\{n\}$ and $\{b\}$. The way in which this mapping is performed is visible in Figure 3.4 on the facing page.

Furthermore, the linear velocity transformation, as described by relation (2.5), is carried out by the chain of blocks shown in Figure 3.5a on page 38, where the box named "Rotation Order: XYZ" is the Simulink built-in block

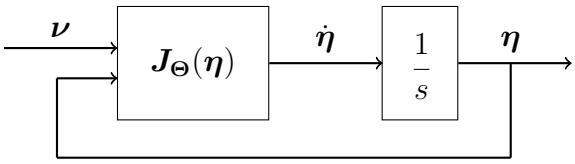


Figure 3.3: Block diagram describing Euler transformation

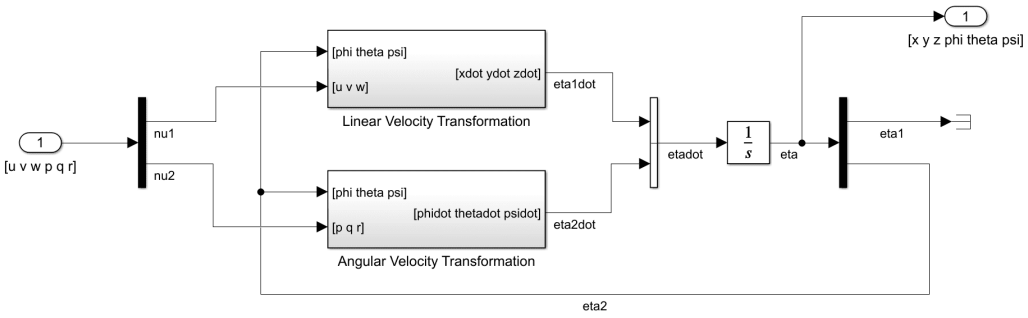
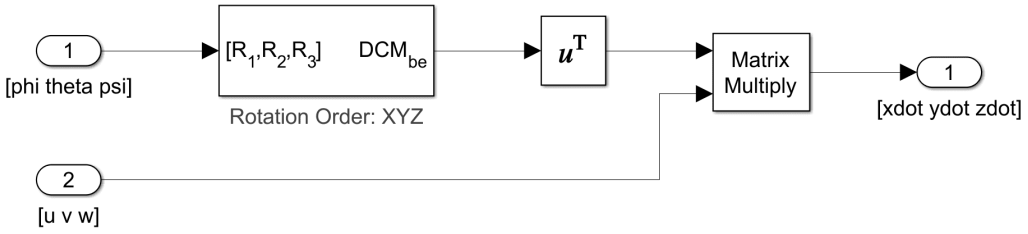


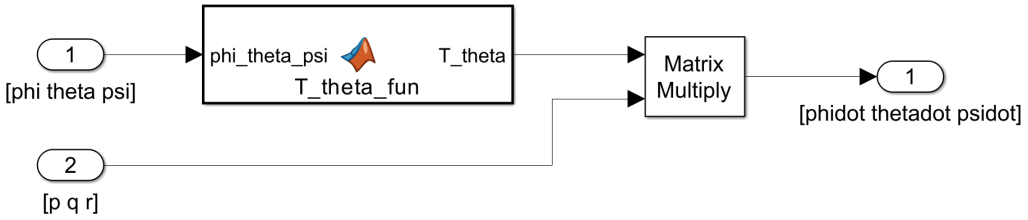
Figure 3.4: Euler transformation block

performing $\boldsymbol{R}_b^n$ (see expressions (2.3) and (2.4)). In the same way, the scheme depicted in Figure 3.5b on the following page computes the angular velocity transformation. The "T_theta_fun" block is a user-defined function that reproduces $\boldsymbol{T_\Theta}$ (recall expression (2.6)) by means of the following Matlab code:

```
function T_theta = T_theta_fun(phi_theta_psi)
phi   = phi_theta_psi(1);
theta = phi_theta_psi(2);
% 2) Pitch
T_theta_mat = [cos(theta) 0 -sin(theta);
                    0     1      0     ;
               sin(theta) 0  cos(theta)];
% 1) Roll
T_phi_mat   = [1       0        0   ;
               0  cos(phi)  sin(phi);
               0 -sin(phi)  cos(phi)];
T_theta_inverse = [ [1;0;0] T_phi_mat*[0;1;0]  ...
    T_phi_mat*T_theta_mat*[0;0;1] ];
T_theta = inv(T_theta_inverse);
end
```



(a) *Linear velocity transformation block*



(b) *Angular velocity transformation block*

Figure 3.5: Transformation block diagrams

3.2 Kinetics Block

The overall structure implemented to compute the 6DOFs equations of motion in Simulink is shown in Figure 3.6 on the next page; the sole purpose of the diagram is to reproduce relation (2.34). To understand the schematics, one can look at the arrow labels describing the logical steps of the system. Moreover, all the gain blocks containing the writing "Minv*uvec" are simply meant to multiply each block output by the inverse of the matrix $\boldsymbol{M}$, defined in (2.32).

Proceeding in the characterization of the diagram, the four light grey user-defined boxes are now described starting from the one on top.

$\boldsymbol{C}_A(\boldsymbol{\nu})$ Function Block

The content of this block is shown in Figure 3.7 on the facing page. Being the hydrodynamic Coriolis and centripetal matrix a function of the velocity

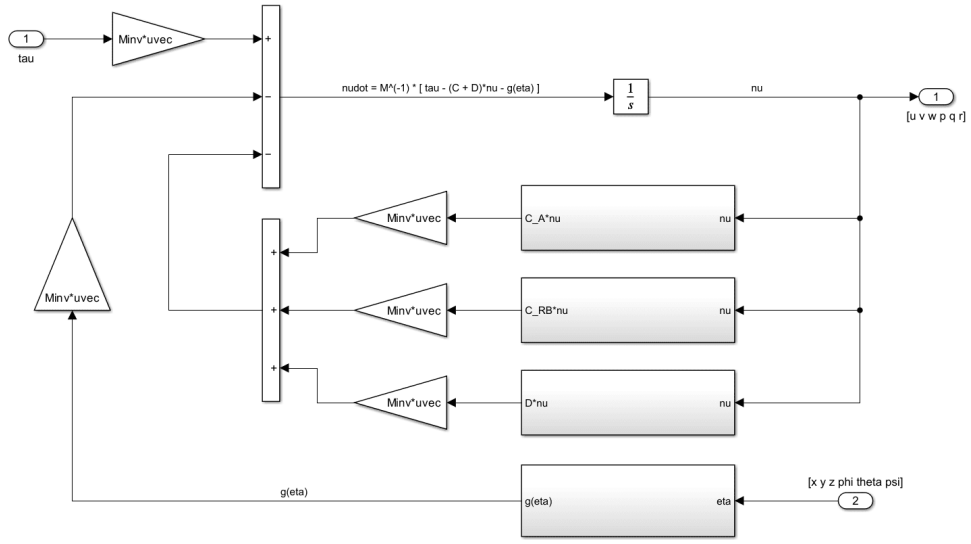


Figure 3.6: Kinetic equations block diagram

vector $\boldsymbol{\nu}$ and the added mass matrix $\boldsymbol{M}_A$, the function needs two inputs. Then, recalling expression 2.34, "C_A" is multiplied by the velocity and "C_A*nu" represents the final output. The Matlab function implemented in the "C_A_fun" box simply reproduces 2.28.

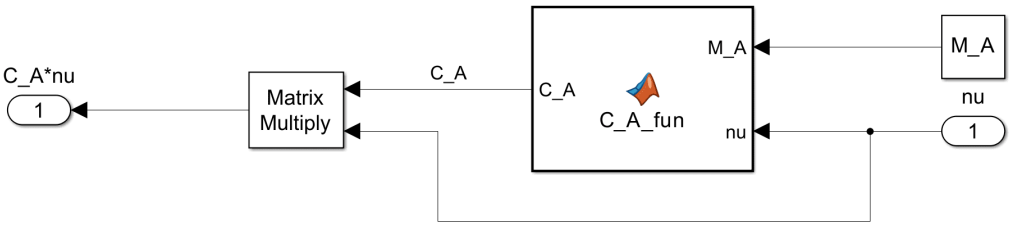


Figure 3.7: Hydrodynamic Coriolis and centripetal matrix block

```
function C_A = C_A_fun(M_A,nu)
u = nu(1); v = nu(2); w = nu(3);
p = nu(4); q = nu(5); r = nu(6);

a1 = M_A(1,1)*u+M_A(1,2)*v+M_A(1,3)*w+M_A(1,4)*p+M_A(1,5)*q+M_A(1,6)*r;
a2 = M_A(2,1)*u+M_A(2,2)*v+M_A(2,3)*w+M_A(2,4)*p+M_A(2,5)*q+M_A(2,6)*r;
a3 = M_A(3,1)*u+M_A(3,2)*v+M_A(3,3)*w+M_A(3,4)*p+M_A(3,5)*q+M_A(3,6)*r;
b1 = M_A(4,1)*u+M_A(4,2)*v+M_A(4,3)*w+M_A(4,4)*p+M_A(4,5)*q+M_A(4,6)*r;
b2 = M_A(5,1)*u+M_A(5,2)*v+M_A(5,3)*w+M_A(5,4)*p+M_A(5,5)*q+M_A(5,6)*r;
b3 = M_A(6,1)*u+M_A(6,2)*v+M_A(6,3)*w+M_A(6,4)*p+M_A(6,5)*q+M_A(6,6)*r;

C_A = [0    0    0    0   -a3   a2;
       0    0    0    a3   0   -a1;
       0    0    0   -a2   a1    0;
       0   -a3   a2   0   -b3   b2;
       a3   0   -a1   b3   0   -b1;
      -a2   a1   0   -b2   b1   0];
end
```

$\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ Function Block

The matrix of rigid-body Coriolis and centripetal forces is obtained by the scheme visible in Figure 3.8. The Matlab function inside "C_RB_fun" is aimed at reproducing the matrix $\boldsymbol{C}_{RB}(\boldsymbol{\nu})$ as it is defined in 2.17. Moreover, it can be seen that, as expected, the function needs four inputs, namely the center of gravity position, the inertia matrix, the mass and the body velocity of the vehicle.

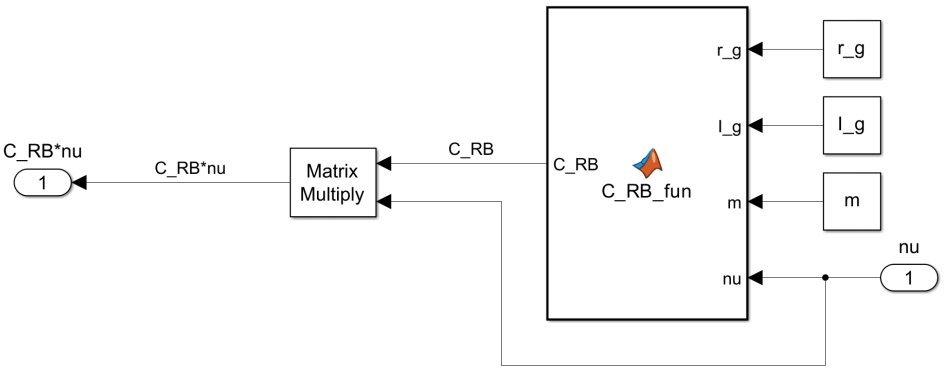


Figure 3.8: Rigid-body Coriolis and centripetal matrix block

```
function C_RB = C_RB_fun(r_g,I_g,m,nu)
nu1 = nu(1:3);
nu2 = nu(4:6);

skew_nu1 = [  0   -nu1(3)  nu1(2);
            nu1(3)   0    -nu1(1);
           -nu1(2) nu1(1)    0   ];

skew_nu2 = [  0   -nu2(3)  nu2(2);
            nu2(3)   0    -nu2(1);
           -nu2(2) nu2(1)    0   ];

skew_rg = [  0   -r_g(3)  r_g(2);
           r_g(3)   0    -r_g(1);
          -r_g(2) r_g(1)    0   ];

Ig_nu2 = I_g*nu2;
skew_Ig_nu2 = [  0   -Ig_nu2(3)  Ig_nu2(2);
               Ig_nu2(3)   0    -Ig_nu2(1);
              -Ig_nu2(2) Ig_nu2(1)    0   ];


C11 = zeros(3);
C12 = -m*skew_nu1 - m*skew_nu2*skew_rg;
C21 = -m*skew_nu1 + m*skew_rg*skew_nu2;
C22 = -skew_Ig_nu2;

C_RB = [C11 C12;
        C21 C22];
end
```

It is clear from the above Matlab code that the skew symmetric matrices are "manually" generated inside the function itself.

$\boldsymbol{D}(\boldsymbol{\nu})$ Function Block

The hydrodynamic damping, as already pointed out, is made up of its linear and nonlinear (quadratic) parts. While the linear matrix $\boldsymbol{D}_l$ is constituted by constant coefficients, the nonlinear one needs to be shaped considering the effect of the velocity, that is why, on the right-hand side of Figure 3.9 on the following page, the input "nu" is also present. The other inputs are simply the constant quadratic coefficient visible in expression (2.21) and, of course, the linear matrix. The Matlab code presented hereafter can be found in the

user-defined function “Dn_fun”.

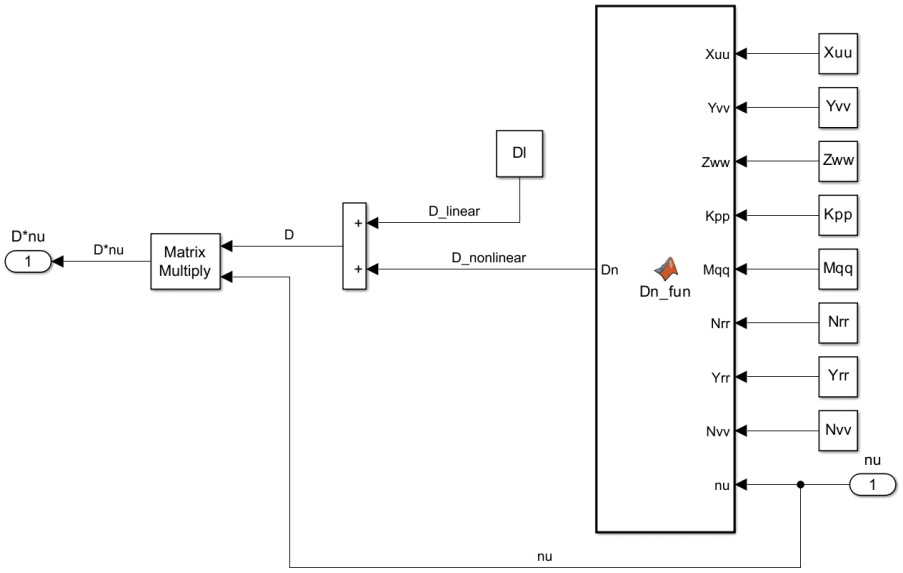


Figure 3.9: Hydrodynamic damping block

```
function Dn = Dn_fun(Xuu,Yvv,Zww,Kpp,Mqq,Nrr,Yrr,Nvv,nu)
Dn11 = Xuu*abs(nu(1));
Dn22 = Yvv*abs(nu(2));
Dn33 = Zww*abs(nu(3));
Dn44 = Kpp*abs(nu(4));
Dn55 = Mqq*abs(nu(5));
Dn66 = Nrr*abs(nu(6));

Dn = -[Dn11   0     0     0     0     0;
        0   Dn22    0     0     0     0;
        0     0   Dn33    0     0     0;
        0     0     0   Dn44    0     0;
        0     0     0     0   Dn55    0;
        0     0     0     0     0   Dn66];
end
```

$g(\eta)$ Function Block

The block implementing the restoring forces into the Simulink model is given in Figure 3.10 on the next page. It is important to note that, in contrast with the previous operators, this matrix is not a function of the body velocity, in

fact it depends on the submarine position with respect to the inertial frame of reference. Moreover, from the lines of code presented in the following, one can see that the function simply reproduces (2.19). The inputs are the position vectors of the center of gravity and the center of buoyancy, the weight and buoyancy forces of the marine craft and $\boldsymbol{\eta}$.

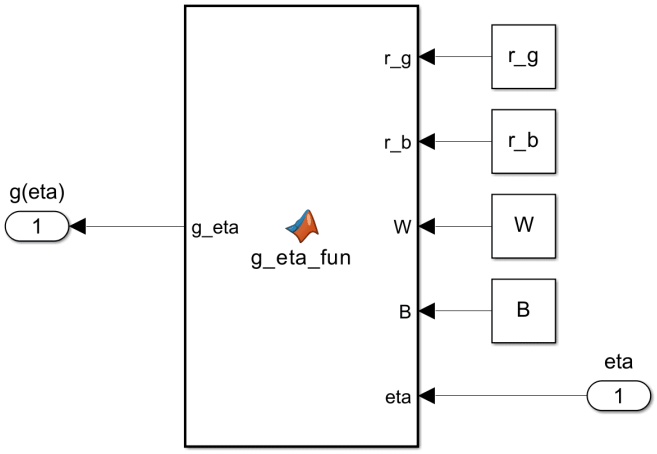


Figure 3.10: Restoring forces block

```
function g_eta = g_eta_fun(r_g,r_b,W,B,eta)
xg = r_g(1);
yg = r_g(2);
zg = r_g(3);

xb = r_b(1);
yb = r_b(2);
zb = r_b(3);

phi   = eta(4);
theta = eta(5);

g_eta=[(W-B)*sin(theta);
      -(W-B)*cos(theta)*sin(phi);
      -(W-B)*cos(theta)*cos(phi);
      -(yg*W-yb*B)*cos(theta)*cos(phi)+(zg*W-zb*B)*cos(theta)*sin(phi);
       (zg*W-zb*B)*sin(theta)+(xg*W-xb*B)*cos(theta)*cos(phi);
      -(xg*W-xb*B)*cos(theta)*sin(phi)-(yg*W-yb*B)*sin(theta)];
end
```

In conclusion, the Simulink model obtained through the outlined steps is still

general: it can be useful to describe the major part of UUVs since it simply applies the equations of motion as described by the Fossen mathematical model.

However, to specifically characterize Blucy behavior, the Simulink model needs to be fed with Blucy proper parameters, such as mass, geometry, hydrodynamic coefficients, etc.

The process of identifying the parameters of Blucy will be addressed in the following chapter.

Chapter 4

Parameters Identification

As already stated, to make Blucy autonomous it is crucial to provide the equations of motion with the vehicle characteristic parameters. It is easy to understand that the more accurate these parameters are, the better the simulator will reproduce the UUV real behavior, and the more effective will be the applied controllers. The aim of this chapter can therefore be divided in:

- introducing Blucy main parameters;
- showing the procedure followed to estimate Blucy coefficients;
- briefly discussing how the parameters influence the simulator;
- highlighting the drawbacks of the methods employed;
- presenting the results.

The first set of parameters needed to identify Blucy virtual model is represented by the vehicle main *geometrical* and *inertial* properties:

- mass and volume of the vehicle;
- position of the Center of Gravity (CG);
- position of the Center of Buoyancy (CB);
- inertia matrix.

These parameter values are typically obtained through physical measurements or Computer-Aided Design (CAD) analysis. Hence, the first step is to set up a faithful 3D CAD model of the ROV that takes into account all its main characteristics (such as detailed parts, materials, dimensions, etc.).

Moreover, the second ingredient of the mathematical model is represented by the UUV *hydrodynamic* parameters (hydrodynamic damping and added mass), their evaluation can be achieved in two different ways, which are: real experiments, and Computational Fluid Dynamics (CFD) simulations.

The second methodology was selected due to time and cost limitations — it is clear enough that the first way is generally the most expensive — along with the increasing accuracy of modern calculators. Moreover, to perform CFD analysis, some pre-processing is necessary, as shown in Figure 4.2 on the facing page.

It is noteworthy to say that the complexity of the CAD model plays an important role in the preparation time as well as in the computational cost needed for running the CFD simulations. For this reason, starting from Blucy complete CAD model, a simplified version of it was obtained using SolidWorks (SW) [26], as it is highlighted in Figure 4.1.

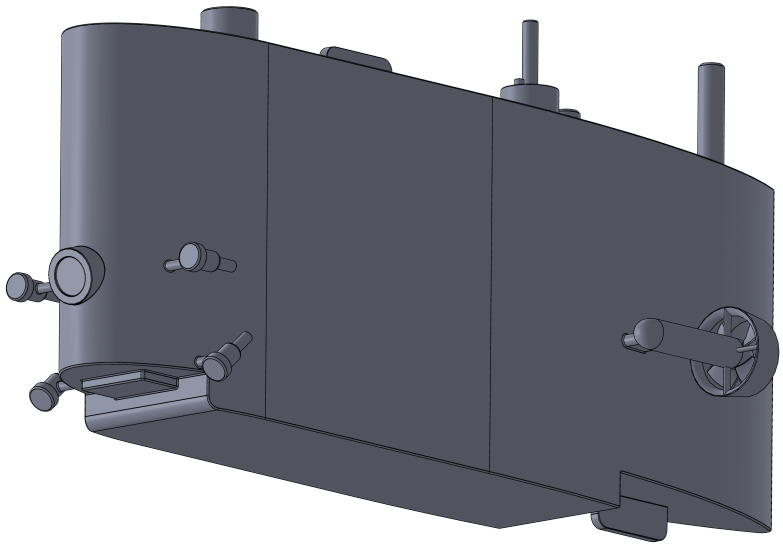


Figure 4.1: CAD model simplification

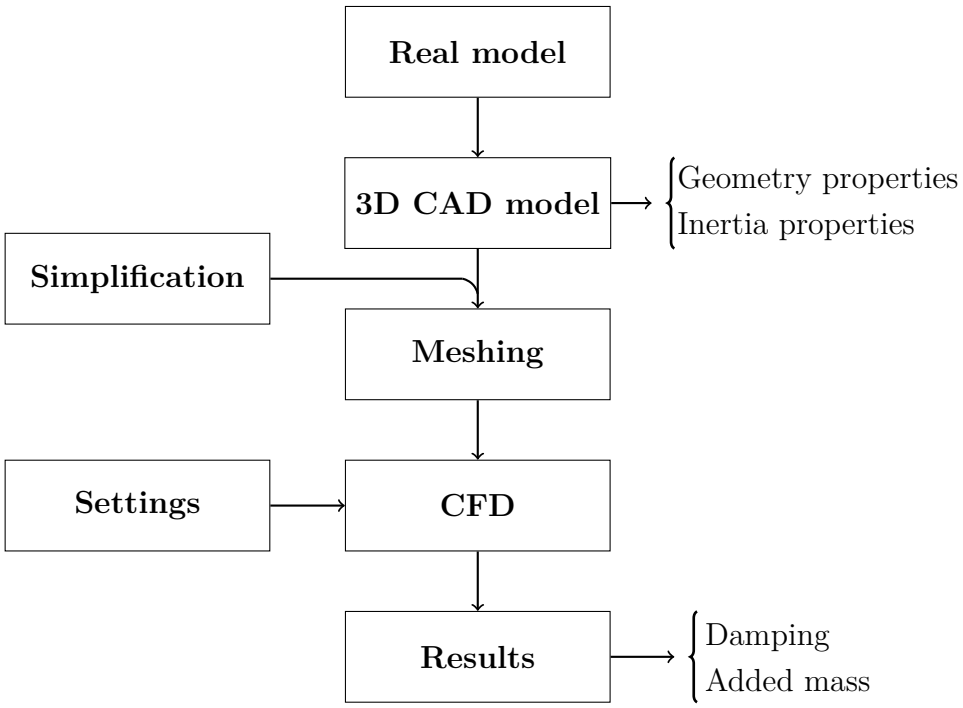


Figure 4.2: Steps for CFD set up

4.1 Geometry and Inertia Parameters Estimation

Blucy coefficients related to its shape and mass were obtained directly from the complete CAD model previously created, as well as from Blucy "Use and Maintenance" manual.

Before presenting the results obtained from Blucy CAD, it is relevant to briefly summarize the design process that was carried out during the internship work [4]. The entire procedure of shaping Blucy 3D model was performed using SolidWorks, and all the individual parts shapes and dimensions were obtained starting from:

- Blucy itself;
- the aforementioned manual of Blucy;
- research papers [19, 20].

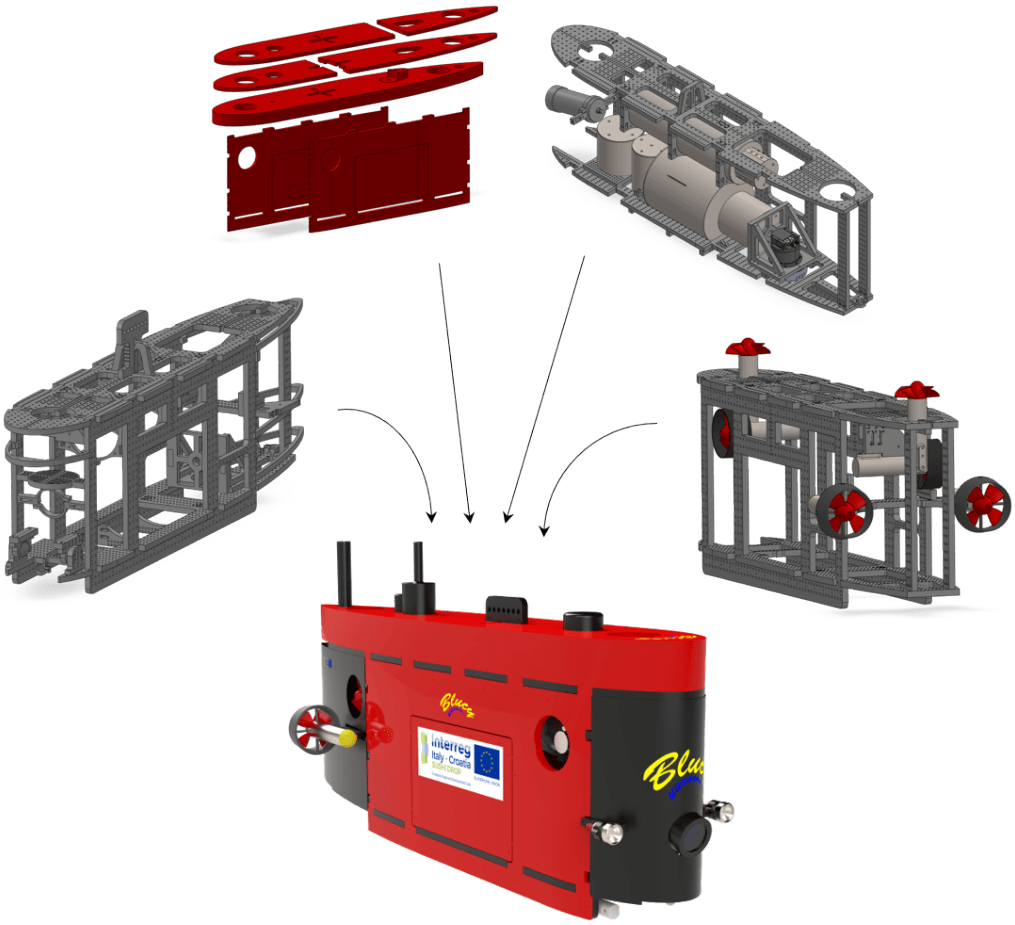


Figure 4.3: Blucy complete CAD model

Figure 4.3 on the facing page provides a glimpse into how the individual parts were assembled to create the complete system.

Canisters

The first components to be designed were Blucy various canisters, which are cylindrical-shaped pressure vessels that are used to store and transport equipment, instruments, or samples in underwater missions. They are often attached to the AUV body, and they are designed to withstand high pressures at depth. In the case of Blucy, the canisters were those stowing the main electrical components of the ROV, such as the Navigation Guidance and Control (NGC), the communication system, the payload, and the FOG.

Structure

Blucy frame is entirely made of perforated parts in High-Density Polyethylene (HDPE). These are custom components obtained through milling and drilling and are held together by nuts and bolts. It is clear enough that the structure has the task of sustaining all other parts. Another important structural component is represented by the hook, needed to haul the drone into the water.

Buoyancy Foam

Buoyancy foam is a low-density resistant material often employed in the underwater field. The purpose of the foam is to make the drone buoyancy to be slightly larger than its weight. This, in fact, allows the robot

- to require less power for hovering;
- to be statically stable about its roll axis;
- to surface automatically in case of system failure.

Thrusters

Blucy motion is guaranteed by six thrusters attached to the main frame. There are four horizontal actuators of which two control the *surge* motion (both are mounted on the stern), and two control the *yaw* angle (one is on the starboard and the other on the port side). Additionally, the two vertical thrusters fixed on the top of the robot allow for the *heave* movement of the vehicle. Each thruster consists of a cylinder containing the brushless servomotor that is magnetically connected to the propeller shaft.

4.1.1 Geometry and Inertia Results

From the detailed CAD assembly, the mass of Blucy was determined to be:

$$m_{\text{blucy}} = 216.14\,kg$$

which is in agreement with the one reported in the manual. Moreover, from the manual, it was taken the mass of the water displaced by the vehicle, which is equal to the UUV volume multiplied by the water density, resulting in:

$$m_{\text{water}} = 216.16\,kg$$

To obtain the weight and buoyancy force, it is sufficient to multiply the previous values by the gravity acceleration.

Another important result derived from the 3D model is the CG position of Blucy, which is reported in Figure 4.4 on page 52; whereas the CB position was taken from the manual. Considering Blucy CG as the center of the UUV and therefore of the body frame, its definition simplifies to:

$$\boldsymbol{r}_g = (\,0,\,0,\,0\,)\,mm$$

while the CB position with respect to CG becomes:

$$\boldsymbol{r}_b = (\,7,\,0,\,-96\,)\,mm$$

Parameter	Value $kg\,m^2$
I_{xx}	11.318
I_{yy}	50.169
I_{zz}	42.682
I_{yx}	0.015
I_{zx}	2.03
I_{zy}	0.017

Table 4.1: Mass results

making Blucy statically stable.

Concerning the inertial parameters of Blucy, they were automatically derived from SolidWorks with respect to the body axis centered in the CG; they are reported in Table 4.1. The *inertia matrix* of the UUV can now be written according to (2.13):

$$\boldsymbol{I}_g = \begin{bmatrix} 11.318 & -0.015 & -2.033 \\ -0.015 & 50.169 & -0.017 \\ -2.033 & -0.017 & 42.682 \end{bmatrix}$$

4.2 Hydrodynamic Parameters Estimation

In the process of simplifying the vehicle, some major adjustments were made: all internal components (and the gaps in between) were neglected in favor of a solid volume with a smooth and continuous external surface; on the outside of the main volume only the larger parts were left in place, furthermore the geometry of the external parts was simplified by replacing real elements with cylinders; the propellers of the external thrusters were constrained against rotation; fillets were applied at edges. These simplifications are summarized in Table 4.2 on page 53.

Moreover, two different software were employed to set up and run the CFD simulations: OpenFOAM (OF) [21] and SolidWorks Flow Simulation; their tasks were divided as follows:

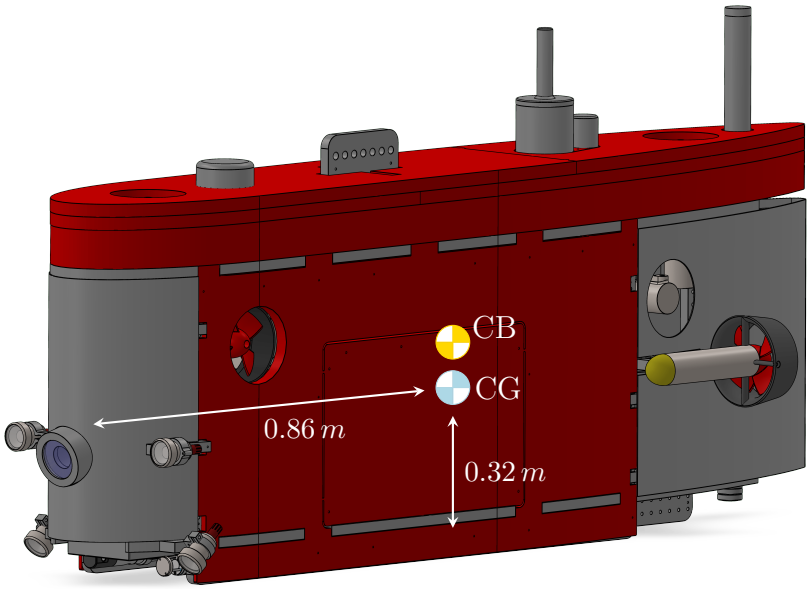


Figure 4.4: CG and CB position

OpenFOAM was used to simulate Blucy linear velocities of surge, sway and heave. If properly used, it represent a more powerful solution with respect to the SW tool for CFD; nonetheless, being an open source software without Graphical User Interface (GUI), it can be cumbersome to set up;

SW Flow Simulation was used to simulate Blucy angular velocities about the roll, pitch and yaw axis. The application is easier to set up when compared to OF, especially when dealing with parametric studies.

The tasks division was mainly due to time and hardware constraints, and the pre-processing tool selected to perform the model meshing (necessary for OF) was ANSA [27].

It is now time to proceed in analyzing the software setup.

4.2.1 Linear Motion Simulations

As already pointed out, the meshing of the CAD model was performed by means of ANSA, which is a pre-processor tool capable of preparing and

Simplifications	
Geometry	– all external holes and internal cavities were filled in order to have a continuous external surface – fillets were applied at every external edge – the original shapes of external elements, such as the mounts of the main thrusters and LEDs, were replaced by extruded cylinders – external cables were not taken into account – bottom of Blucy extremely simplified through plane surfaces
Setup	– external propellers were left rigidly in place (no rotation allowed during simulations) – the porosity of external surfaces was neglected

Table 4.2: CAD simplifications

OpenFOAM	SW Flow Simulation
Pros	
– High accuracy:it provides high accuracy results compared to many other CFD tools, which makes it suitable for advanced simulations – Flexibility: it is highly customizable and offers a wide range of modeling options, making it suitable for a variety of applications in different fields – Large quantity of documentation is readily available online from the OF community – Open source: it is free to use and the source code is available for users to modify and extend	– Very easy to set up – Very easy-to-use GUI: immediate visual feedback on CFD settings – The documentation provides extensive coverage of its features and commands – Automatic mesh generation: no need for external tools – Easy to automatically run multiple simulations with varying parameters
Cons	
– The software needs an external tool to generate the mesh: more time and effort are needed to acquire the knowledge to use another (often commercial) tool – Documentation not always easy to be found – To access all the available commands, OF needs to operate within the Ubuntu environment – Lack of user-friendly GUI	– Not so powerful for CFD: limited number of simulation parameters can be set up; under equal conditions requires a more considerable amount of RAM with respect to OF – Proprietary software – Only basic hexahedral mesh generation – It is relatively difficult to automatically run multiple simulations with varying parameters

Table 4.3: Software comparison

processing simulation models, including meshing, geometry clean-up, and model setup. ANSA is commonly used in the automotive, aerospace, and defense industries for simulations of fluid dynamics and structural mechanics. In addition, ANSA is especially useful when dealing with OpenFOAM, as it provides a convenient interface that allows the user to directly export the model into the OF case.

The steps followed for meshing the model were:

1. define the external control volume;
2. define a size-box considering the wake developed by Blucy motion;
3. define a smaller size box to enhance mesh refinement on Blucy external surface;
4. define a size box containing the propellers to avoid geometry discontinuities;
5. define the mesh geometry for surface and volume;
6. define the mesh properties.

Initially, the external control volume was shaped as a simple cuboid by defining its eight vertices coordinates with respect to the point $(0, 0, 0)$ corresponding to Blucy center of gravity. Eventually, the volume was set to be larger where the wake was expected to develop: for instance, in the case of a positive surge motion, the largest part of the volume was placed behind Blucy stern (see Figure 4.8 on page 59); this was done to improve the accuracy of the CFD analyses. For the same reason, the control volume was slightly modified for the simulations involving the sway motion and the heave one: in both cases the overall volume size was reduced due to the smaller velocities involved. In addition, the same considerations hold in the definition of the wake size boxes, drone shell and propellers; but it is, however, important to highlight some differences: while the drone and propeller refinement boxes were kept constant for the three cases, the wake box was the main varying parameter: its volume was considerably increased in the regions of wake generation. The

final overall dimensions of the volume and size boxes can be found in the table below.

Object	**Length** (m)		
	x	y	z
Control volume	32	20	21
Wake size box	6	4	5
Drone size box	2.25	0.84	1.2

Table 4.4: Volume and size boxes dimensions for surge simulations

In addition to the external dimensions of the aforementioned cuboids, it is also necessary to specify their mesh "spacing". The mesh spacings determine the resolution of the numerical mesh used in the simulation, which affects the accuracy and computational cost of the analysis. Therefore, the spacings were chosen considering the desired accuracy and computational resources available: the outer volume resolution was set to increase from $1\,m$ at the boundaries to $25\,cm$, while the size box spacings are reported in Table 4.5.

Then, the surface and volume mesh geometries were selected: concerning the surfaces, the ANSA command "Auto CFD" with the option "tetra" was used, while for the volume geometry the command "hexa interior" was selected. This selection (mostly appreciable in Figure 4.9) is due to the fact that, in any simulation case, the flow is almost unidirectional, therefore, this kind of mesh geometry results more efficient in terms of computational cost.

Finally, the mesh generation provided the number and types of elements that are summarized in Table 4.6. Afterwards, the mesh properties (i.e. the boundary conditions) were defined as reported in Table 4.7.

In order to minimize errors and computational time, it is necessary to investigate the actual behavior of Blucy prior setting up the CFD simulations. In practice, this entails estimating Blucy operational range and outlining, as far as possible, the UUV maximum linear and angular velocities. The real data of Blucy linear body velocity, obtained from the NGC, is shown in Figure 4.11 on page 62. The following considerations should be taken into account:

- the real data of Blucy has not been extensively analyzed in this work,

Name	**Max length surface** (mm)	**Max length volume** (mm)	**Growth rate volume**
Drone	10	10	1.2
Wake	35	35	1.2
Left propeller	1	1	1
Right propeller	1	1	1

Table 4.5: Properties of the size boxes

Shell	
quads	113 863
trias	1 786
total	**115 649**

Volume	
tetras	3 480 762
pentas	248 272
hexas	8 434 332
pyramids	461 425
total	**12 623 791**

Table 4.6: Type and number of mesh elements

Name	**Type**	**Num. Elem.**
Blucy	wall	1 125
Inlet	patch	630
Outlet	patch	630
Symmetry	symmetry	3 264
Auto Detect Volume	fluid	12 623 791

Table 4.7: Properties

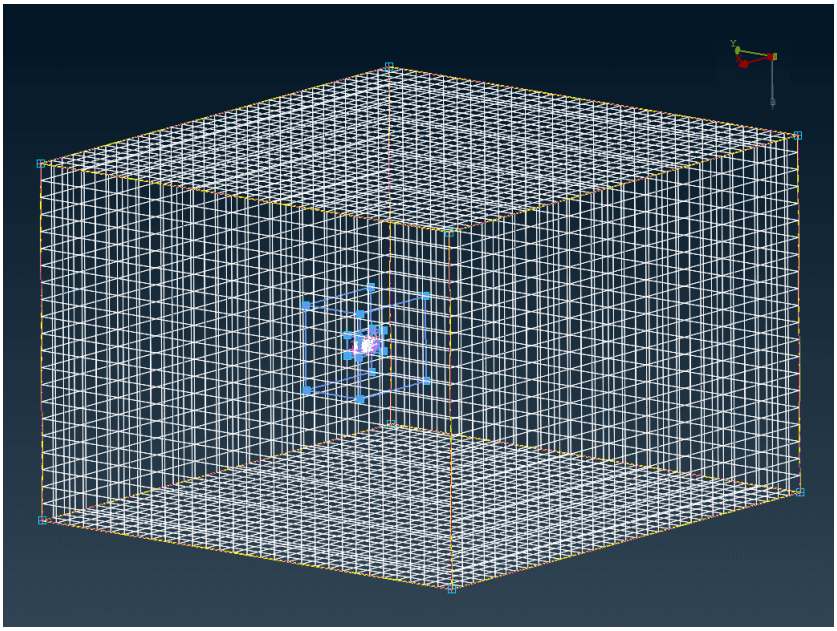


Figure 4.5: Control volume for CFD

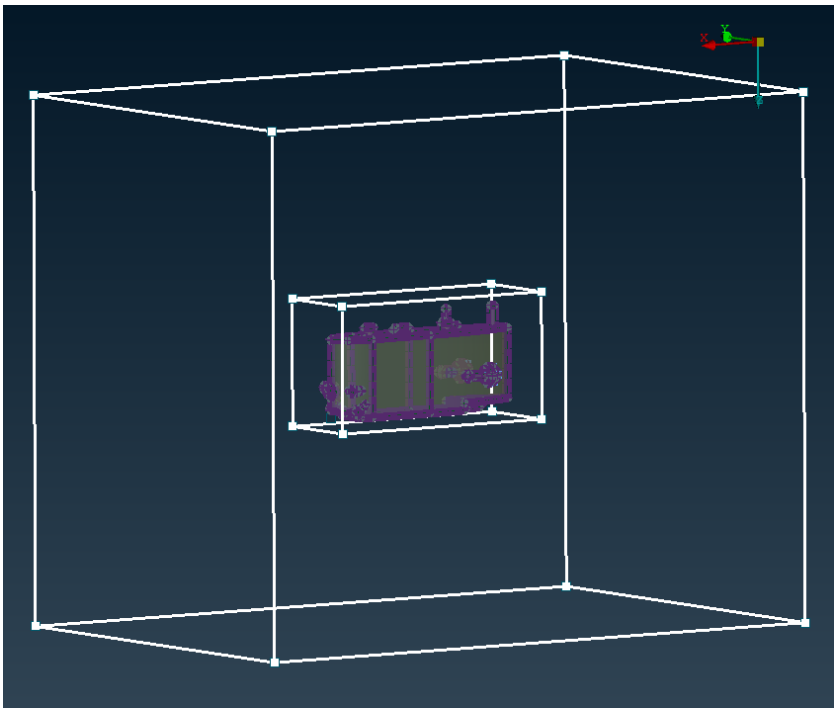


Figure 4.6: Wake and drone size boxes

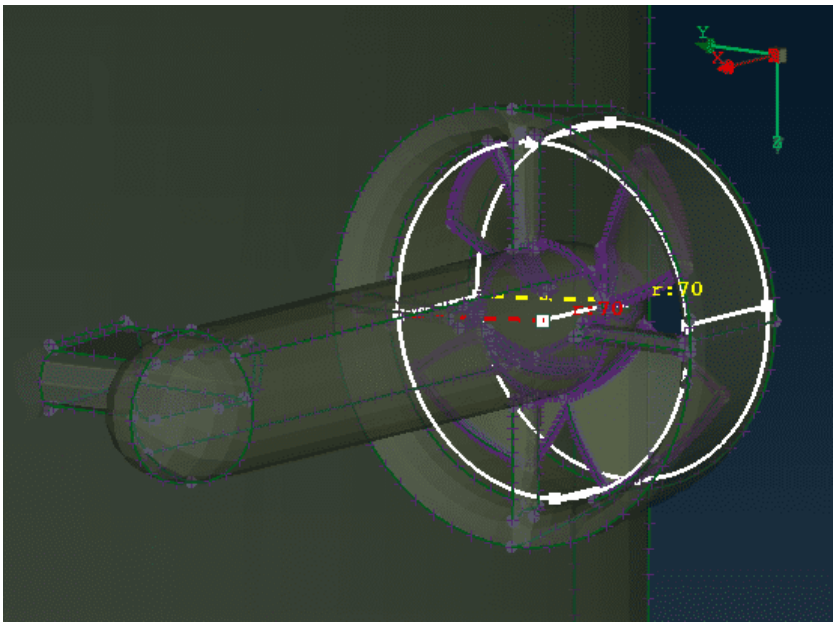


Figure 4.7: Propeller size box

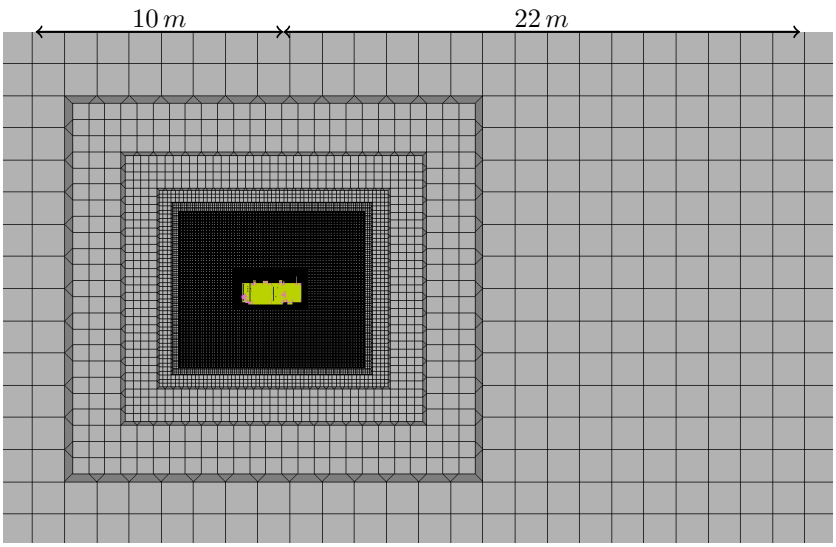


Figure 4.8: Volume mesh

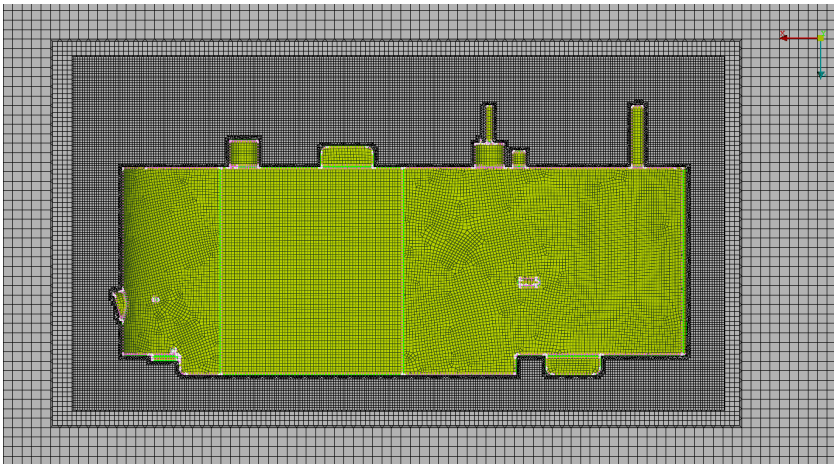


Figure 4.9: Volume mesh detail

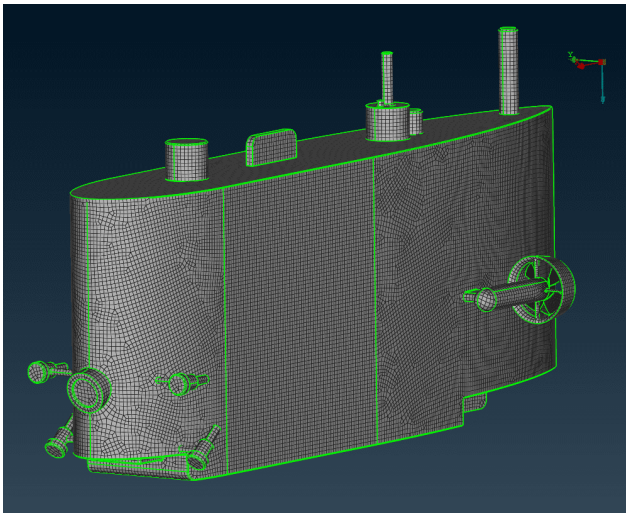


Figure 4.10: Blucy surface mesh

Motion	**Speed range** (m/s)	**Step** (m/s)	**Notes**
Surge	-2 to 2	0.2	Blucy is designed to reach its maximum speed in heave. According to the manual, the drone maximum forward speed is $2\,m/s$. However, pilot's experience reported a lower top speed of about $1\,m/s$.
Sway	-0.8 to 0.8	0.2	Blucy was not designed for speed in sway, as its lateral section is significantly larger than the other sections, causing a substantial increase in resistance when moving along the y_b axis.
Heave	-1.4 to 1.4	0.2	Blucy maximum velocity in heave can be influenced by the drone net buoyancy. However, real data shows lower velocities with respect to heave.

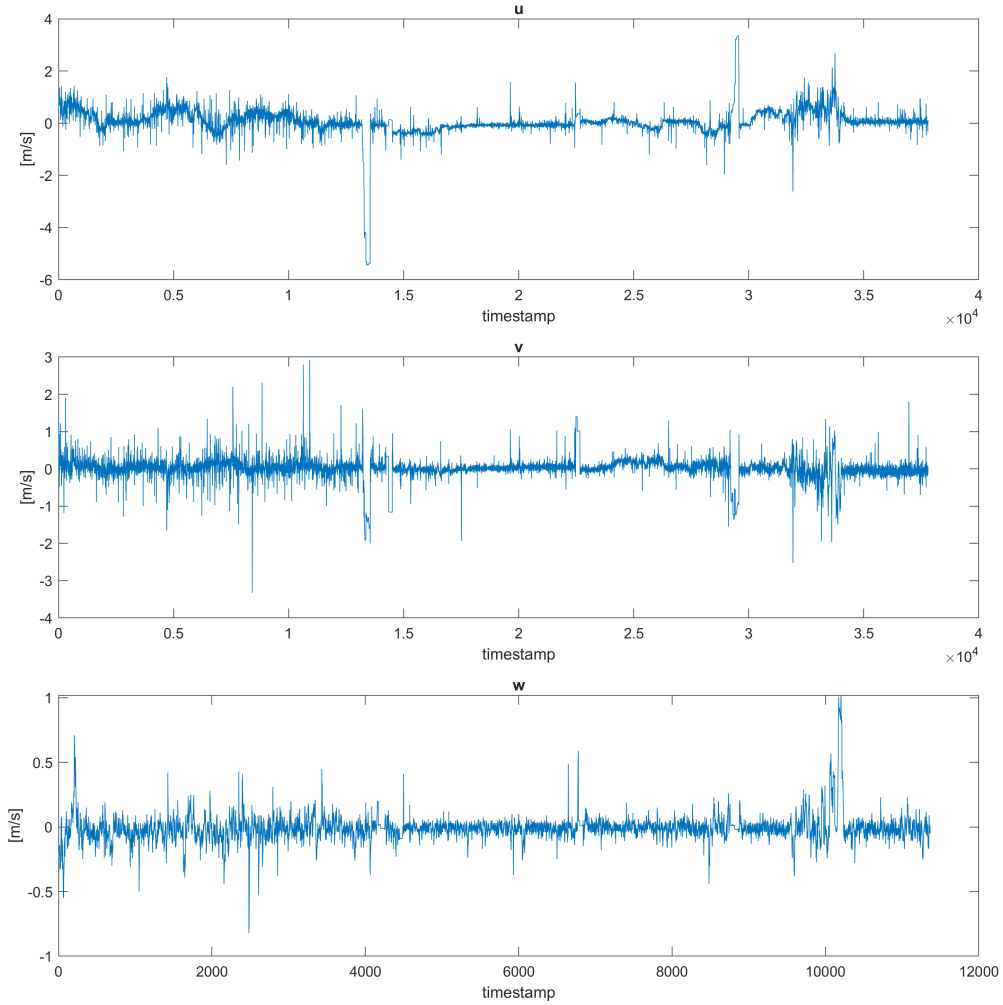
Table 4.8: Linear velocity simulations set up

which means that errors and inconsistencies may still be present. Thus, the plotted data were considered in conjunction with the pilot's experience;

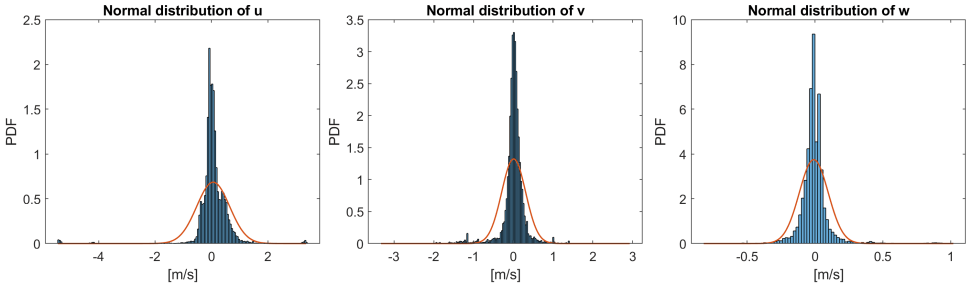
- data points that are significantly different from the mean trend were deemed outliers due to measurement noise and sea currents;
- it is important to note that not all sensors are sampled in the same manner.

Table 4.8 summarizes the simulations set up for the linear motion along the body axes.

Concerning the OpenFOAM environment, it is behind the scope of this work to give a deep insight into every aspect of the software; however, a brief introduction of the main characteristics and a concise description of the settings used should now be made. Following the scheme depicted in Figure 4.12 on page 64, one can understand that a CFD simulation in OpenFOAM is referred to as an OpenFOAM "case", which consists of



(a) *Blucy linear motion data*



(b) *PDF of Blucy linear velocities*

Figure 4.11: Visualization and statistics of Blucy linear velocities

several directories, including "0", "constant", and "system". The directory "0" contains the initial conditions of the simulation, while the directory "constant" contains information such as the geometry and boundary conditions of the simulation domain, as well as information regarding the fluid viscosity and the turbulence model employed. Finally, the "system" directory contains the solver configuration files. These configuration files are named *dictionaries*, which are text files that specify the information required by the software to run a simulation. For instance, the dictionaries contained in "system" are:

fvSchemes $\rightarrow$ defines the discretization schemes that are used to approximate the solution of the governing equations. It sets the method used to discretize the terms in the equations and determine the order of accuracy;

fvSolution $\rightarrow$ defines the solution algorithms used to solve the discretized equations. It sets the method used to iterate the solution and determine when the solution has converged. For instance, it can specify the type of linear solver and the stopping criteria for the solution process;

controlDict $\rightarrow$ defines the overall settings: the user can set the simulation start and stop time, the time step size, and the write interval for saving the solution. It can also specify the type of solver used for the simulation, as well as any additional options for that solver.

The aforementioned dictionaries, together with the dictionaries contained in "0", are reported in Appendix B (note that the initial and boundary conditions are those for the surge simulation at $1\,m/s$). It is now important to make some considerations:

- the simulations performed within this work are based on solving the steady-state Reynolds-Averaged Navier-Stokes (RANS) equations;
- simpleFoam is a solver in OpenFOAM that uses the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm to solve the RANS equations;

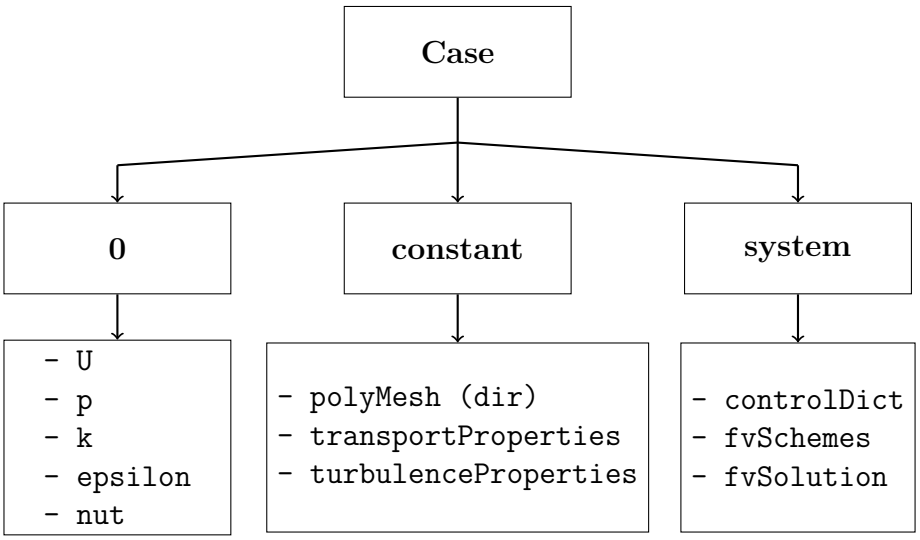


Figure 4.12: OpenFOAM hierarchy

- the turbulence model used within simpleFoam is the $k-\epsilon$ model, which involves solving two separate equations for the turbulence kinetic energy (k) and the turbulence dissipation rate (ϵ).

4.2.2 Angular Motion Simulations

The CFD simulations of Blucy angular motion were performed by means of SolidWorks Flow Simulation. The fluid density was set to 1025 kg/m^3 (salt water), and SolidWorks turbulence model was employed with turbulence intensity equal to 0.1%. In fact, SolidWorks is able to manage turbulent flows by means of the Favre-Averaged Navier-Stokes (FANS) equations, together with the transport equations for the turbulent kinetic energy and its dissipation rate: the $k-\epsilon$ formulation. Lastly, the convergence criteria were chosen to be the forces (and moments) along (and about) the three principal directions x_b, y_b and z_b.

The mesh was directly generated within the Flow Simulation tool, which, however, offers built-in meshing options that are less customizable than those of ANSA. The final mesh characteristics are reported in Table 4.9 on the facing page and depicted in Figure 4.13 on the next page.

Control volume	
x-size	$16\,m$
y-size	$16\,m$
z-size	$9\,m$
Hexa mesh	
Fluid cells	1 807 502
Fluid cells contacting solids	679 793
Maximum refinement level	5

Table 4.9: SolidWorks mesh parameters

Motion	**Speed range** (deg/s)	**Step** (deg/s)	**Notes**
Roll	0 to 10	1	Blucy may experience minor roll and pitch angular velocities due to hydrodynamic characteristics and sea currents.
Pitch	0 to 10	1	
Yaw	0 to 10	1	Blucy is designed to perform yaw maneuvers.

Table 4.10: Angular velocity simulations set up

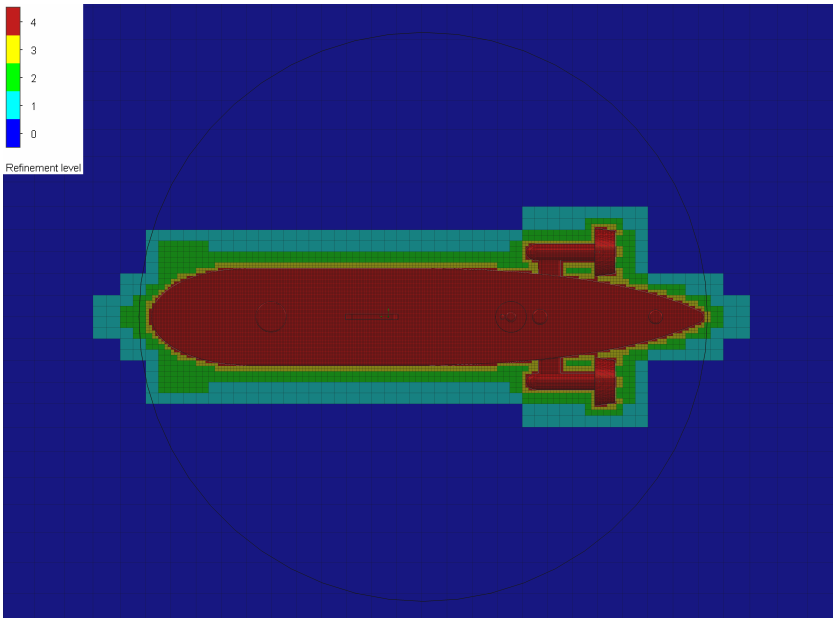


Figure 4.13: SolidWorks mesh visualization

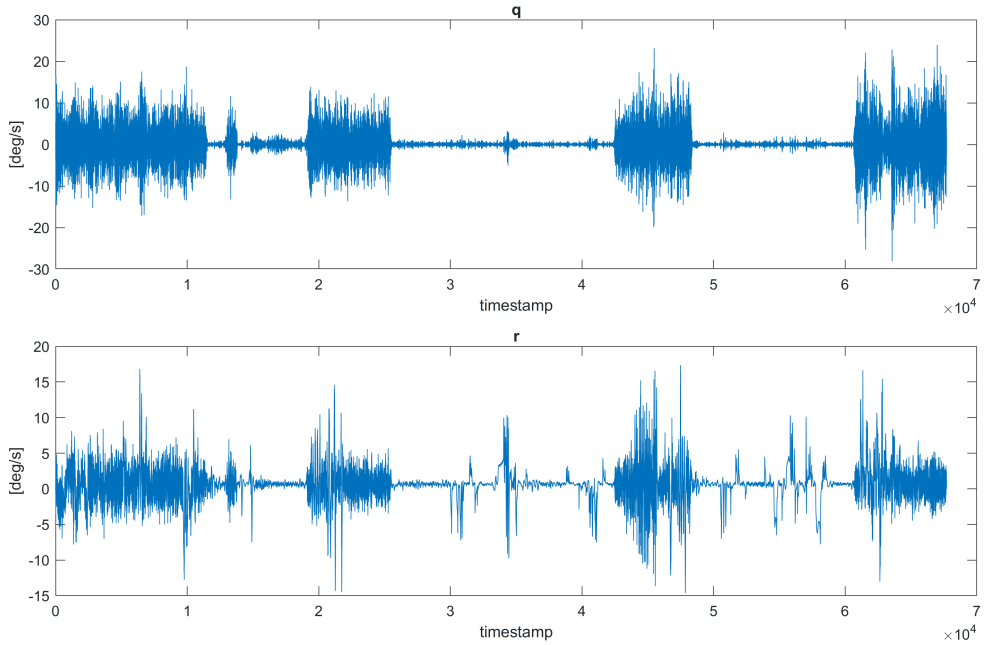
Analogously to the set up of the linear motion simulations, some considerations should now be made about the kind of angular motion that Blucy can experience in a real-world scenario. With regards to the discussion on linear velocities, Figure 4.14 on the facing page shows data on the pitch and yaw angular velocities (whereas some issues have been encountered in the roll angular velocity sampling), and their respective normal distributions, which justify the choice to simulate angular maneuvers not exceeding $10\,deg/s$. It is important to note that while Blucy is controllable in yaw, it is not designed to execute complex maneuvers about its roll and pitch axes. Nonetheless, the drone may still experience minor angular velocities due to its hydrodynamic characteristics and sea currents. Table 4.10 on the previous page summarizes these considerations.

4.2.3 CFD Results

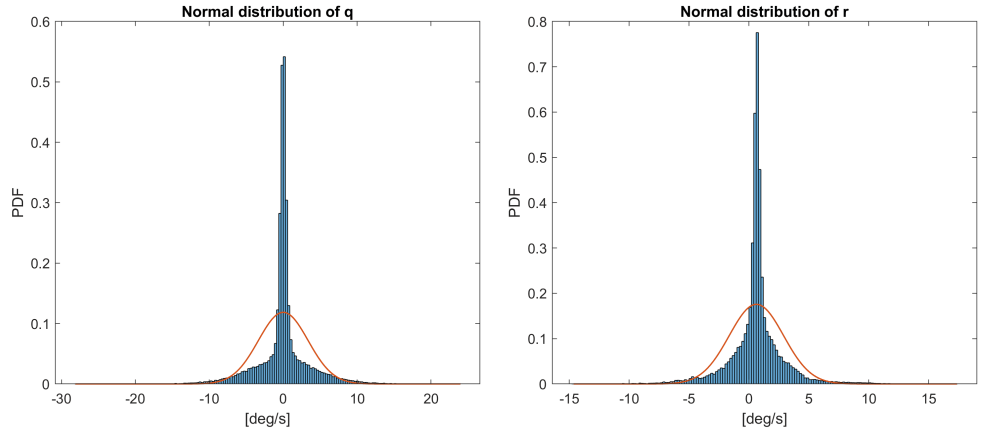
The considerations of the preceding subsections were first used to generate the different meshes for the six different simulation scenarios (surge, sway, heave, roll, pitch and yaw), then they were implemented in SolidWorks Flow Simulation as "parametric studies", and in OpenFOAM as "cases". Some considerations are now necessary:

- in SolidWorks, a parametric study is a type of simulation study that allows for automatically investigating how a design responds to changes in its parameters. In this case, the variable parameter was the angular rotation of Blucy, and the outputs were the forces and moments acting on the moving body;
- in the OpenFOAM environment, on the other hand, a "case" for each velocity was built. The code that was implemented in the dictionaries (boundary conditions, controlDict, etc.) to set up the simulations is reported in Appendix B*;
- in OpenFOAM, a "function object" was used to output the forces and moments acting on the body. A "function object" is a piece of code

*The attached code is only for the simulation of surge at $1\,m/s$.



(a) *Blucy angular motion data*



(b) *PDF of Blucy angular velocities*

Figure 4.14: Visualization and statistics of Blucy angular velocities

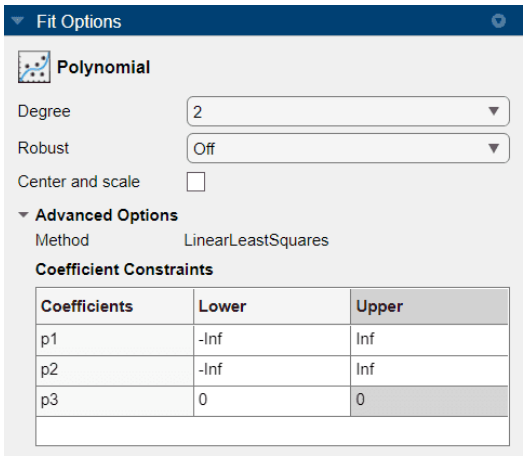


Figure 4.15: Fit options

that performs a specific operation on the simulation data at a particular time during the simulation. It is essentially a utility that can be used to extract, manipulate or post-process data generated during a simulation.

After the simulation runs, all the resulting data were collected and converted into plots of the resulting forces and moments for each different case, at each different velocity. By fitting the resulting curves, the hydrodynamic parameters can be determined. In fact, recalling that

$$X(u) = X_u\, u + X_{|u|u}\, u^2$$

X_u and $X_{|u|u}$ represent respectively the *linear* and *nonlinear damping coefficients* for the longitudinal motion. One can then employ this logic for each curve obtained from the CFD simulations.

To this aim, the Matlab tool "Curve Fitter" was employed. In Figure 4.15 the "Fit Options" can be seen. It should be noted that the fitting functions are polynomial of the second order of the form

$$f(x) = p_1 x^2 + p_2\, x + p_3$$

where p_3 was manually set to zero since no forces are applied to the body at null velocity.

Surge		**Sway**		**Heave**	
Coeff.	Value	Coeff.	Value	Coeff.	Value
X_u	-2.368	X_v	2.13	X_w	0.0837
Y_u	-0.008119	Y_v	-27.58	Y_w	0
Z_u	0.09057	Z_v	18.02	Z_w	-2.06
K_u	0.1185	K_v	5.269	K_w	-0.0066
M_u	0.04597	M_v	0	M_w	-0.1555
N_u	0.1342	N_v	-21.73	N_w	0

Table 4.11: Linear motion hydrodynamic coefficients

Roll		**Pitch**		**Yaw**	
Coeff.	Value	Coeff.	Value	Coeff.	Value
X_p	-0.0961	X_q	-0.0148	X_r	-0.7742
Y_p	-1.436	Y_q	0.0048	Y_r	5.068
Z_p	-0.1694	Z_q	-0.071	Z_r	-0.3614
K_p	0.3026	K_q	0	K_r	0.7409
M_p	0.018	M_q	-0.0626	M_r	-0.2689
N_p	0.4122	N_q	0	N_r	-7.005

Table 4.12: Angular motion hydrodynamic coefficients

Moreover, in Figure 4.16 on the next page, two examples of curve fitting are presented: on the left-hand side of the figure, the curve represents the variation of the longitudinal drag that Blucy experiences in surge at different velocities. On the right-hand side of the figure, the trend of the lateral force in sway is depicted.

Concerning the linear damping terms, they are reported in Table 4.11 and Table 4.12. It should be pointed out that the terms equal to zero are due to the fact that fitting scattered data would be meaningless. All the curves reporting the simulation results can be found in Appendix C.

Regarding the nonlinear damping coefficients, the terms of interest (see relation (2.21)) are reported in Table 4.13 on page 71.

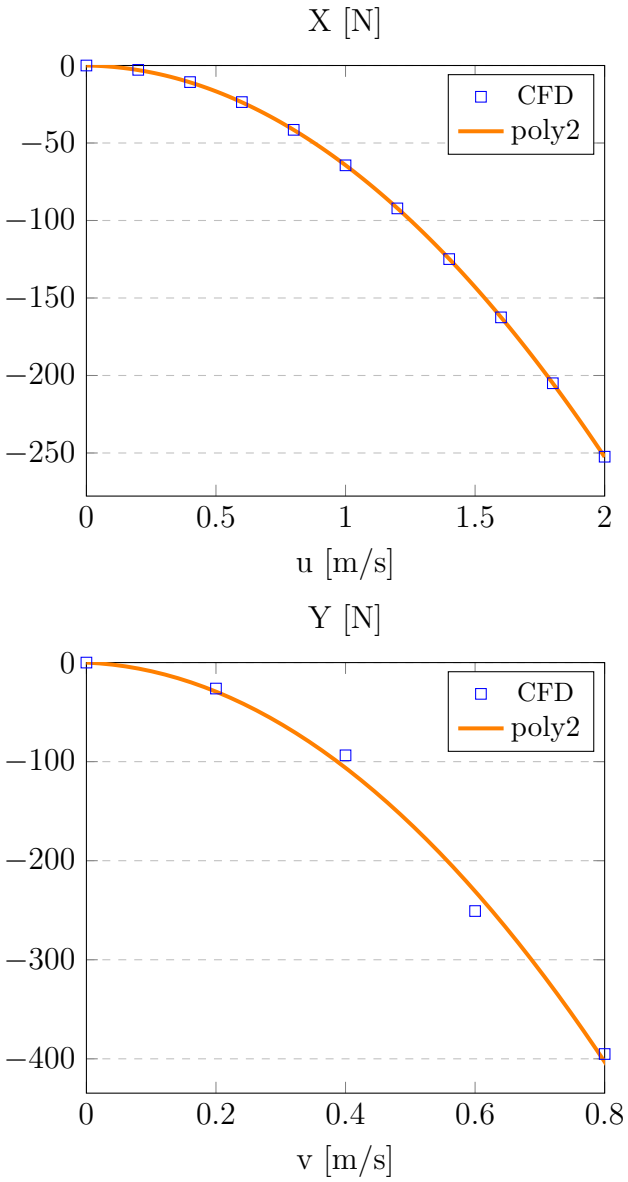


Figure 4.16: Example of fitting functions

Coeff.	Value
$X_{\|u\|u}$	-61.96
$Y_{\|v\|v}$	-595.2
$Z_{\|w\|w}$	-243;
$K_{\|p\|p}$	-23.67
$M_{\|q\|q}$	-95.52
$N_{\|r\|r}$	-240.4

Table 4.13: Nonlinear damping coefficients

The resulting matrices are

$$\boldsymbol{D}_l = -\begin{bmatrix} -2.368 & 2.13 & 0.0837 & -0.0961 & -0.0148 & -0.7742 \\ -0.0081 & -27.58 & 0 & -1.436 & 0.0048 & 5.068 \\ 0.0906 & 18.02 & -2.06 & -0.1694 & -0.071 & -0.3614 \\ 0.1185 & 5.269 & -0.0066 & 0.3026 & 0 & 0.7409 \\ 0.046 & 0 & -0.1555 & 0.018 & -0.0626 & -0.2689 \\ 0.1342 & -21.7300 & 0 & 0.4122 & 0 & -7.005 \end{bmatrix}$$

and

$$\boldsymbol{D}_n(\boldsymbol{\nu}) = -\begin{bmatrix} -61.96\,|u| & 0 & 0 & 0 & 0 & 0 \\ 0 & -595.2\,|v| & 0 & 0 & 0 & 0 \\ 0 & 0 & -243\,|w| & 0 & 0 & 0 \\ 0 & 0 & 0 & -23.67\,|p| & 0 & 0 \\ 0 & 0 & 0 & 0 & -95.52\,|q| & 0 \\ 0 & 0 & 0 & 0 & 0 & -240.4\,|r| \end{bmatrix}$$

Finally, some visualizations of pressure distribution, velocity field and streamlines around Blucy are shown in the following figures. They were generated through the open source software ParaView [17] (only to visualize OpenFOAM results), and SolidWorks.

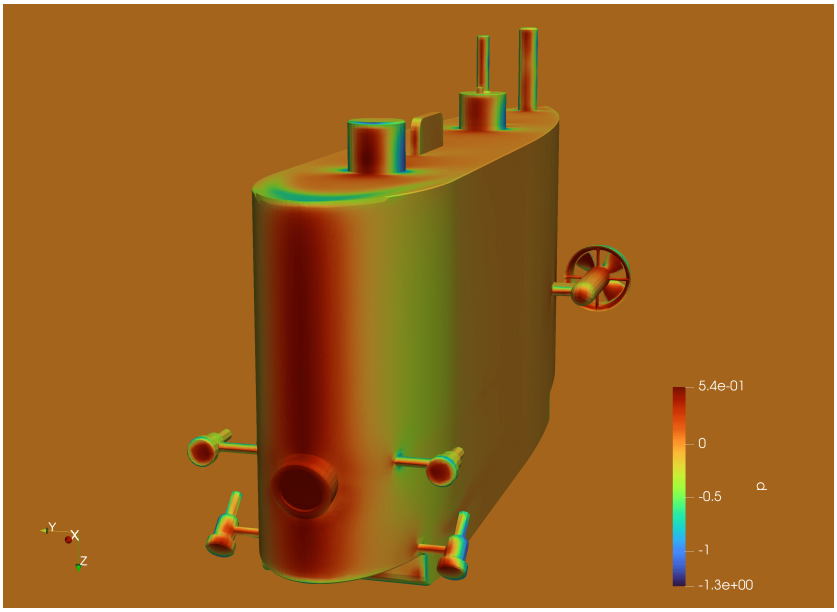


Figure 4.17: Pressure distribution in surge at $1\,m/s$

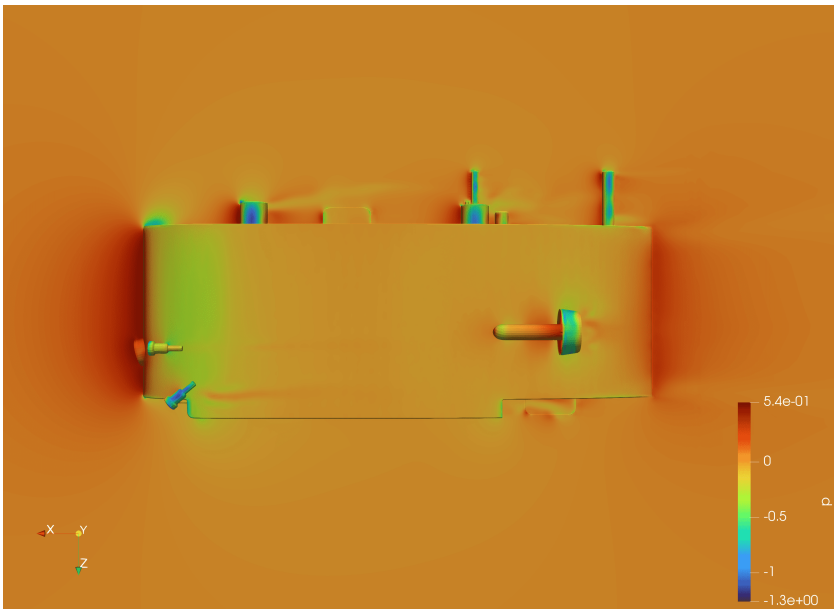


Figure 4.18: Pressure distribution on xz-plane in surge at $1\,m/s$

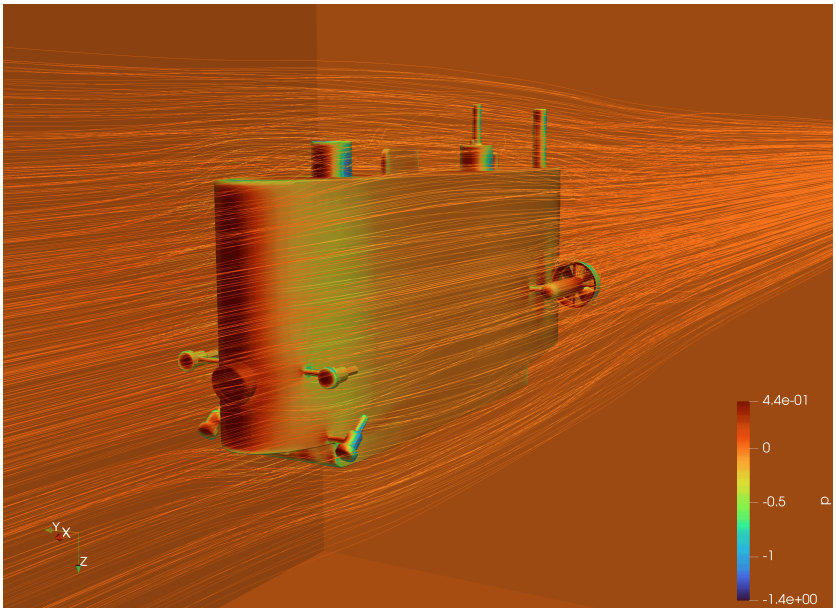


Figure 4.19: Streamlines in surge at $1\,m/s$

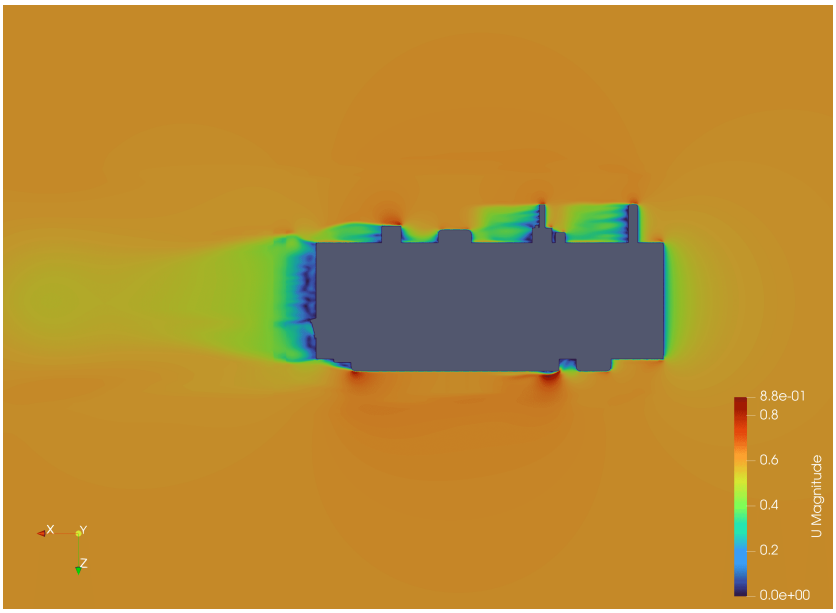


Figure 4.20: Velocity distribution on xz-plane in surge at $-0.6\,m/s$

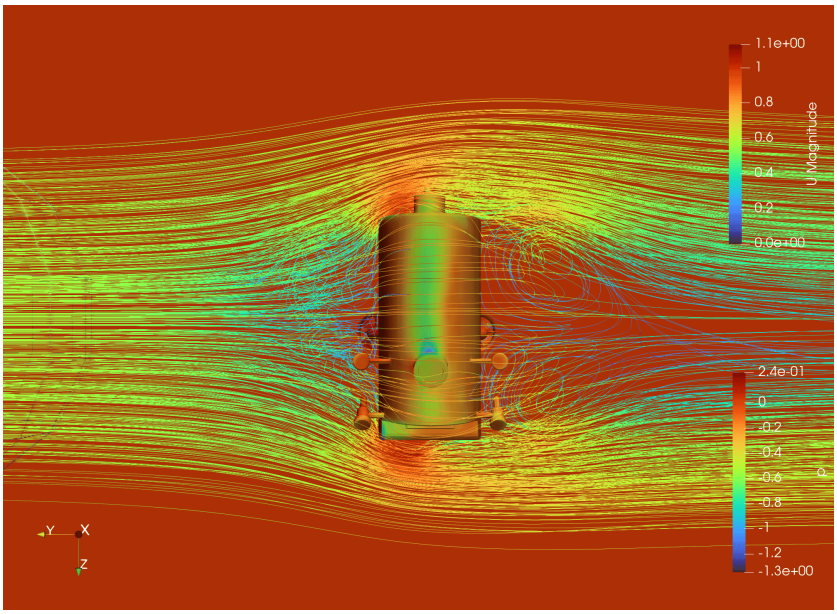


Figure 4.21: Streamlines in sway at $0.6\,m/s$

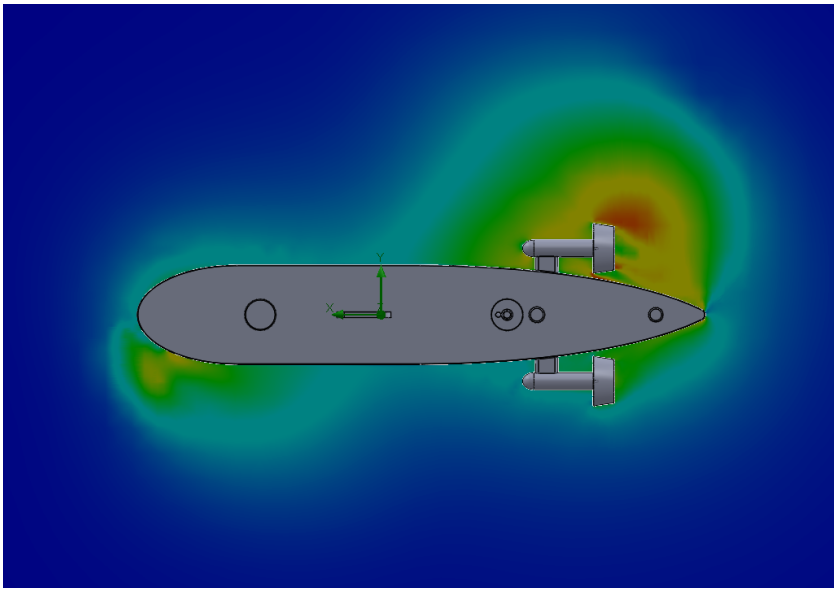


Figure 4.22: Pressure distribution on xy-plane in yaw at $8\,deg/s$

4.3 Added Mass Estimation

The added mass parameters can be estimated through experimental or numerical methods — as for the damping parameters — , and analytical methods. In this work, two techniques were used, which are:

1. the analytical Lamb's theory (as taken from [18]), based on simplifying assumptions about the geometry and motion of the AUV;

2. a numerical CAD environment for the fast computation of added mass: AMCOMP [6].

Lamb's Theory

Lamb's theory is based on an analytical solution for the potential flow around an ellipsoid, which is derived from Laplace's equation. The solution provides expressions for the added masses of the ellipsoid, based on its geometric parameters such as its semi-axes and orientation. It is important to underline the fact that approximating Blucy with an ellipsoid (see Figure 4.23 on page 77) can clearly lead to a reduction in the estimation accuracy.

Lamb's equations are here summarized:

$$A_0 = abc\int_0^\infty \frac{du}{(a^2+u)\,\Delta} \qquad B_0 = abc\int_0^\infty \frac{du}{(a^2+u)\,\Delta} \qquad C_0 = abc\int_0^\infty \frac{du}{(a^2+u)\,\Delta}$$

where a, b and c are the ellipsoid semi-axes, and

$$\Delta = \{(a^2+u)(b^2+u)(c^2+u)\}^{\frac{1}{2}}$$

By choosing a, b and c so to keep the same rations as Blucy semi-axes, and keeping the ellipsoid volume equal to the drone volume, one can use the

previous quantities (see Appendix A) to obtain the elements of $\boldsymbol{M}_A$:

$$\lambda_{11} = \frac{4}{3}\pi\rho abc\,\frac{A_0}{2-A_0} \qquad \lambda_{44} = \frac{4}{15}\frac{\pi\rho abc(b^2-c^2)^2(C_0-B_0)}{2(b^2-c^2)+(B_0-C_0)(b^2+c^2)}$$
$$\lambda_{22} = \frac{4}{3}\pi\rho abc\,\frac{B_0}{2-B_0} \qquad \lambda_{55} = \frac{4}{15}\frac{\pi\rho abc(c^2-a^2)^2(A_0-C_0)}{2(c^2-a^2)+(C_0-A_0)(c^2+a^2)}$$
$$\lambda_{33} = \frac{4}{3}\pi\rho abc\,\frac{C_0}{2-C_0} \qquad \lambda_{66} = \frac{4}{15}\frac{\pi\rho abc(a^2-b^2)^2(B_0-A_0)}{2(a^2-b^2)+(A_0-B_0)(a^2+b^2)}$$

Resulting in

$$\begin{aligned} \boldsymbol{M}_A &= -\mathrm{diag}\,\{\,\lambda_{11},\lambda_{22},\lambda_{33},\lambda_{44},\lambda_{55},\lambda_{66}\,\} \\ &= -\mathrm{diag}\,\{\,34,\,756,\,177,\,9,\,30,\,158\,\} \end{aligned} \tag{4.1}$$

AMCOMP

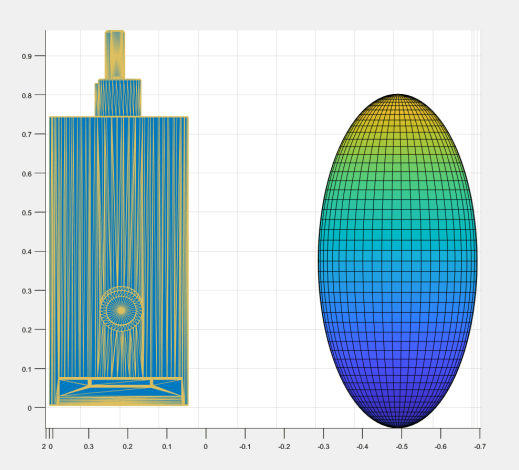
The AMCOMP tool has the ability to compute the added mass for CAD models in Standard Triangle Language (STL) file format by implementing the procedure described in [28]: the underlying idea is to calculate the kinetic energy of the fluid displaced by the moving body by means of the potential flow theory. An illustration of the AMCOMP usage is presented in Figure 4.24 on page 78.

The software gave the following matrix:

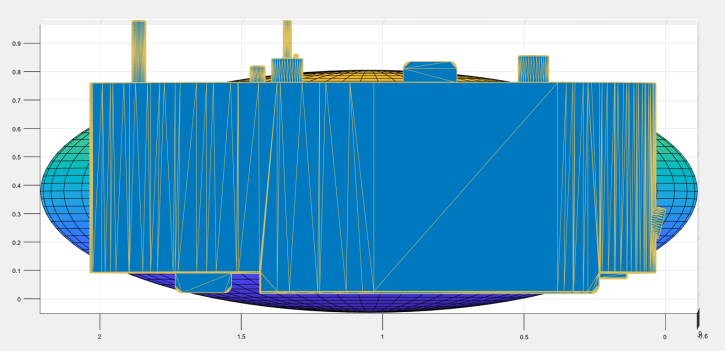
$$\boldsymbol{M}_A = -\begin{bmatrix} -56.41 & 5.8372 & -7.0623 & 2.2791 & 7.8604 & -6.5026 \\ 5.8372 & -327.41 & 4.1352 & 4.7939 & 1.5077 & 62.28 \\ -7.0623 & 4.1352 & -185.61 & 2.2847 & 6.2788 & -1.6736 \\ 2.2791 & 4.7939 & 2.2847 & 0.1137 & -0.4063 & -3.9641 \\ 7.8604 & 1.5077 & 6.2788 & -0.4063 & -39.14 & 7.3145 \\ -6.5026 & 62.28 & -1.6736 & -3.9641 & 7.3145 & -82.4 \end{bmatrix} \tag{4.2}$$

In conclusion:

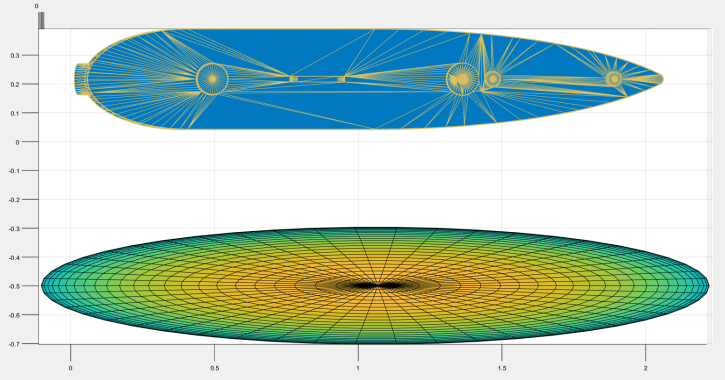
- comparing the results of Lamb's theory and AMCOMP (see Figure 4.25 on page 79), it can be seen that the diagonal elements follow a similar



(a) *Frontal view*



(b) *Lateral view*



(c) *Top view*

Figure 4.23: Blucy and ellipsoid comparison

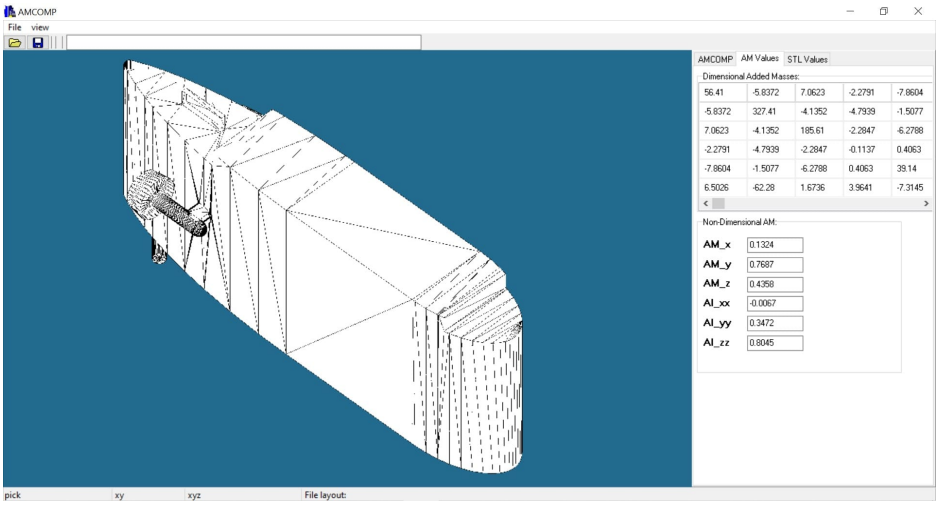


Figure 4.24: AMCOMP environment

trend, and the largest difference lies in the second element, which corresponds to $Y_{\dot{v}}$;

- as expected, a diagonal matrix is obtained from Lamb's theory: all the off-diagonal terms are null since the ellipsoid presents three planes of symmetry;
- off-diagonal terms obtained from AMCOMP, as it will be clear in the forthcoming, can trigger oscillatory effects within the simulator;
- the added mass matrix obtained from AMCOMP is not positive semidefinite: its eigenvalues are $(-0.46, 35.10, 55.14, 72.12, 186.20, 342.73)$, however, the negative element is at least two orders of magnitude smaller than the other eigenvalues, making this error negligible.

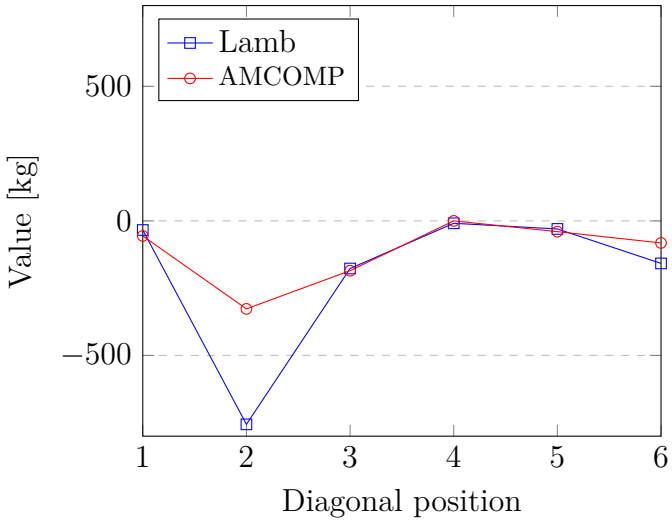


Figure 4.25: Lamb and AMCOMP comparison

Chapter 5

Testing of the Simulink Model

The Simulink model was updated with the parameters obtained in the previous chapter using a Matlab initialization script. The primary script and other necessary Matlab functions to initialize the parameters can be found in Appendix A.

Afterward, a series of tests were conducted, and the most significant results from the various simulations are presented here. The goal is to compare these results in order to better understand the behavior of the model and how the model is influenced by changes in parameters.

The work will be now divided into sections, each section will present a different test. Moreover, each section will provide a table presenting the overall settings of the test in analysis, and the plots of the velocity results (that are represented by the components of the state vector $\boldsymbol{\nu}$). The dimensions are m/s for the linear velocities and deg/s for the angular velocities.

It is also important to note that the center of the body frame is always considered coincident with the CG, therefore it will always be $r_g = (0, 0, 0)$, whereas r_b could vary.

Test I

In this test, all the parameters were left unmodified with respect to the ones derived in the previous chapter. Therefore, all the elements in the hydro-

dynamic matrices are taken into consideration. Furthermore, the buoyancy force is slightly larger than the weight force and there are no propulsive forces acting on the system ($\boldsymbol{\tau}_{\text{th}} = 0$); thus, a negative vertical velocity is expected.

All the main settings are summarized in the table below.

Parameter	**Setting**
r_b	$(\,0, 0, z_b\,)$
$B - W$	> 0
$\boldsymbol{D}_l$	complete
$\boldsymbol{M}_A$	AMCOMP
$\boldsymbol{\tau}_{\text{th}}$	$= 0$
Solver	automatic
Step size	variable
t	$100\,s$

Table 5.1: Test I settings

In Figure 5.2 on page 84 the components of $\boldsymbol{\nu}$ are plotted. Some considerations on the velocity curves should now be made:

- as expected, the heave velocity, w, has a parabolic initial behavior that stabilizes afterwards. This is simply due to the fact that the force acting on the body is constant and equal to $B - W$, this force accelerates Blucy. As Blucy moves through the fluid, it experiences a drag force that opposes its motion. This drag force increases with the velocity of Blucy, which leads to a point where the drag force becomes equal in magnitude to the net buoyant force, resulting in zero acceleration. At this point, the heave velocity stabilizes to a constant value;

- concerning the surge and sway velocities, u and v respectively, their non-zero behavior is due to the fact that the off-diagonal terms of the added mass matrix and of the linear damping matrix are different from zero, therefore the vertical motion mathematically triggers small surge and sway movements, as well as angular actions;

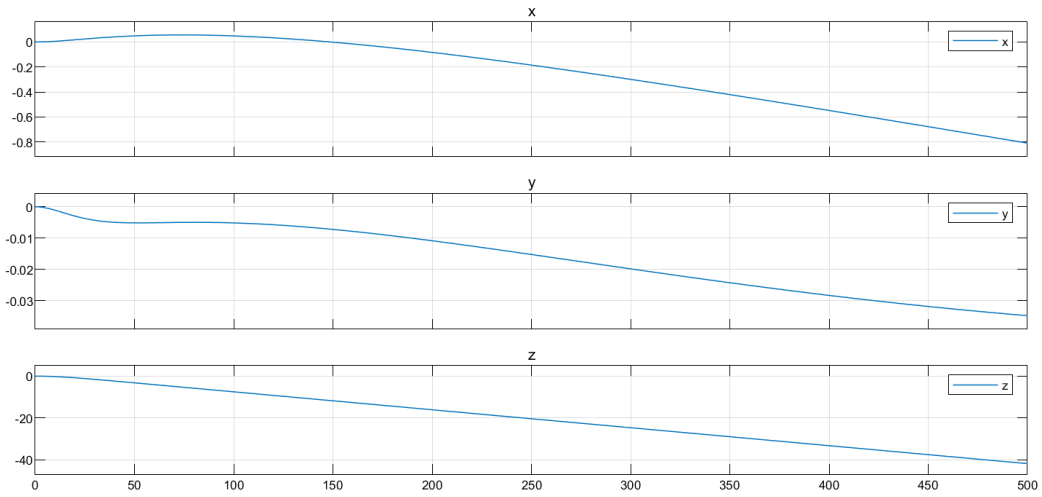
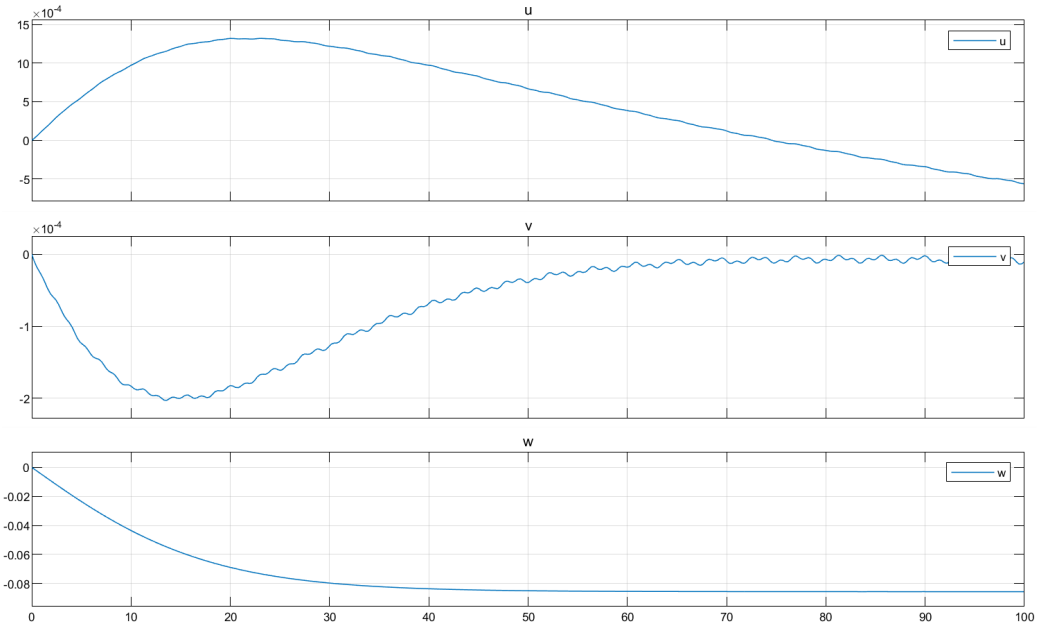


Figure 5.1: Blucy movement during an ascent of 500 s

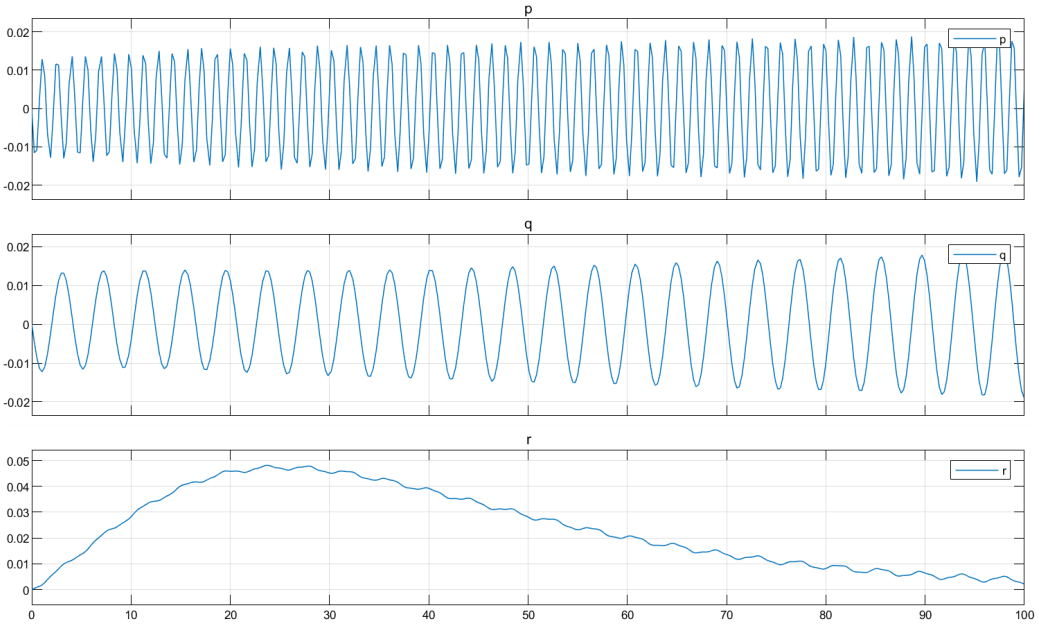
- the oscillatory behavior observed in the angular velocities is unlikely to be physically meaningful for the type of AUV being simulated. Additionally, the magnitude of the oscillation is small and has a zero mean, which is consistent with the idea that it may be a result of numerical error. Therefore, the argument that the oscillations are likely due to numerical integration is plausible. However, it is necessary to perform further analyses or sensitivity studies to confirm the accuracy of the results and assess the impact of numerical errors on the conclusions drawn.

In conclusion, the simulated heave motion of Blucy perfectly describes the behavior of the object that can actually be observed in a real-world scenario. Also, the birth of small surge and sway velocities is in accordance with the mathematics described in Chapter 2 and with the approximations introduced by the CFD simulations; nevertheless, it might be difficult to accurately measure a drift in the x and y directions of less than one meter during an ascent of 40 meters in seawater (see Figure 5.1).

However, Blucy oscillations in roll, pitch, and yaw, although small, do not match reality and most likely represent a persistent numerical error. The next test explores a possible explanation for why these oscillations do not decay.



(a) $\boldsymbol{\nu}_1$ $[m/s]$



(b) $\boldsymbol{\nu}_2$ $[deg/s]$

Figure 5.2: Results of $\boldsymbol{\nu}$ from Test I

Test II

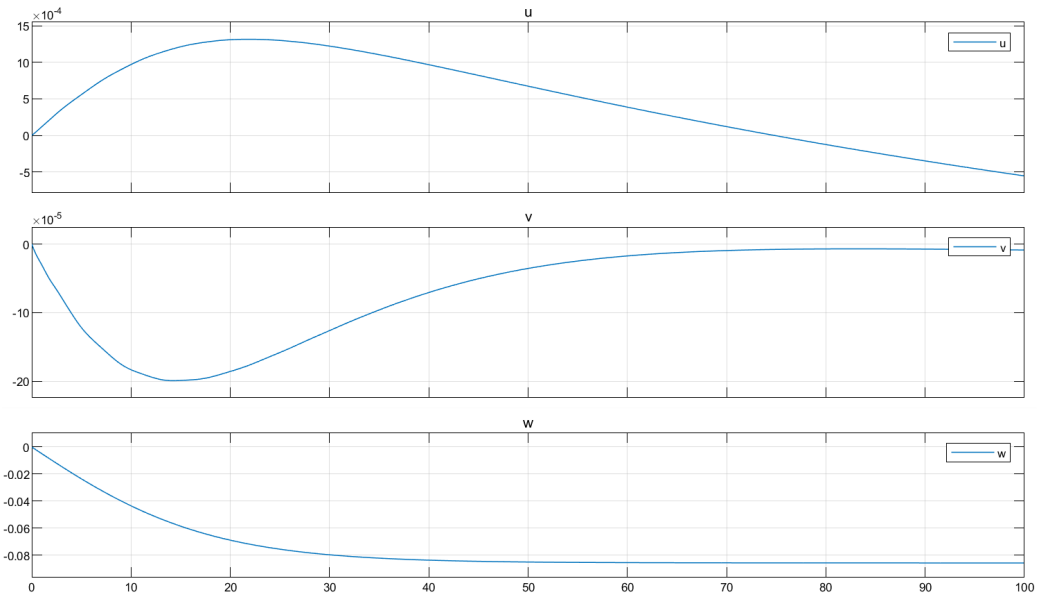
In this test, a different approach was applied, consisting of selecting the "ode1b" solver, which is the backward Euler method with a fixed step size. All the other parameters were kept constant to compare the new results with those of Test I.

Parameter	**Setting**
r_b	$(\,0, 0, z_b\,)$
$B - W$	> 0
$\boldsymbol{D}_l$	complete
$\boldsymbol{M}_A$	AMCOMP
$\boldsymbol{\tau}_{\text{th}}$	step input along x_b
Solver	ode1be
Step size	fixed
t	$100\,s$

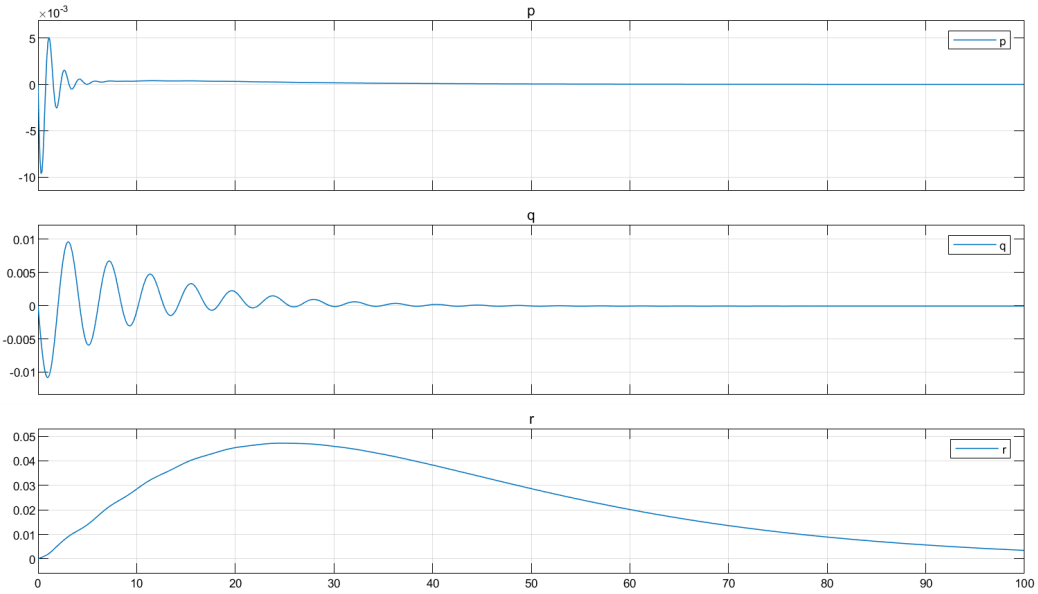
Table 5.2: Test II settings

In Figure 5.3 on the next page, the results of the current simulation are shown. It is clear that the linear velocity curves have the same general behavior as those in the previous case. However, focusing on v, a smoother trend can be seen that suggests the vanishing of the numerical error in place of a more realistic attitude.

Moreover, looking at the angular rates, it can be concluded that their behavior is now more reasonable: the oscillations are still small in amplitude, but they decay over time. The roll and pitch rates go back to zero as w approaches its constant value, whereas the yaw rate remains different from zero and reaches a negative constant value after $300\,s$. This last observation is in accordance with the CFD simulations.



(a) $\boldsymbol{\nu}_1$ $[m/s]$



(b) $\boldsymbol{\nu}_2$ $[deg/s]$

Figure 5.3: Results of $\boldsymbol{\nu}$ from Test II

Test III

In this test, two main changes in the settings were introduced: first, an external force in the surge direction was applied for a finite interval of time. Second, the buoyancy force acting on Blucy was adjusted to be equal to its weight. These changes were made to investigate the transition of the system from a moving state to a still state: since no additional energy is entering the system after the input, unless numerical integration errors are present, the small angular oscillations are expected to vanish, as well as any other movement of the body.

The shape of the input is presented in Figure 5.4 on the following page: the step represents an external force X, measured in newtons, that acts on the system for $100\,s$.

Parameter	**Setting**
r_b	$(\,0, 0, z_b\,)$
$B - W$	$= 0$
$\boldsymbol{D}_l$	complete
$\boldsymbol{M}_A$	AMCOMP
$\boldsymbol{\tau}_{\text{th}}$	step input along x_b
Solver	ode1be
Step size	fixed
t	$600\,s$

Table 5.3: Test III settings

Looking at Figure 5.5a on page 89, some considerations can now be done regarding the outcome of the simulation: concerning the linear velocity results, Their behavior is reasonable since there is a forward velocity that starts with the input and reaches a constant speed in accordance with the CFD simulations. Subsequently, the velocity decays. Moreover, the motion coupling also triggers small velocities in sway and heave that decay as well.

Focusing on the roll, pitch and yaw rates, it is clear that their trend can actually describe a physical phenomenon: once Blucy has returned to its state

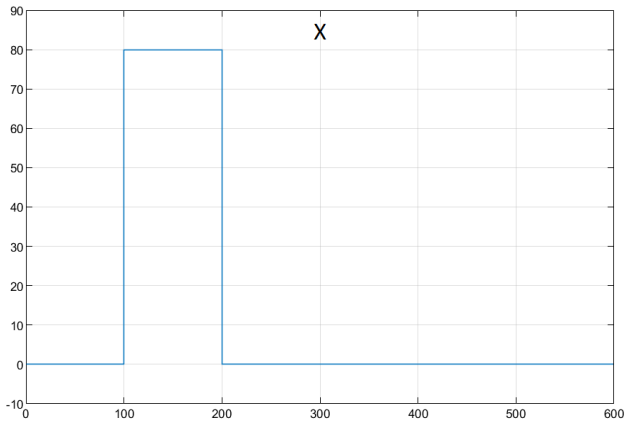


Figure 5.4: Step input

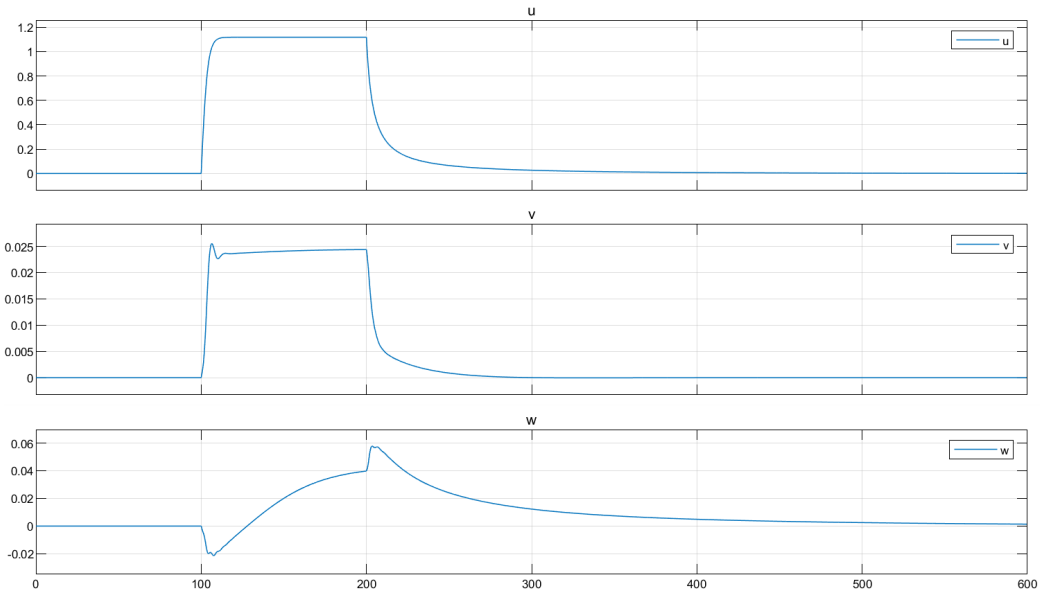
of rest, also the angular dynamics ceases its course after the initial trigger due to the coupling of the system.

Overall, these results seem to effectively capture the dynamics of the system. However, further investigation is needed to better understand the observed phenomena and their implications for the modeling of complex systems.

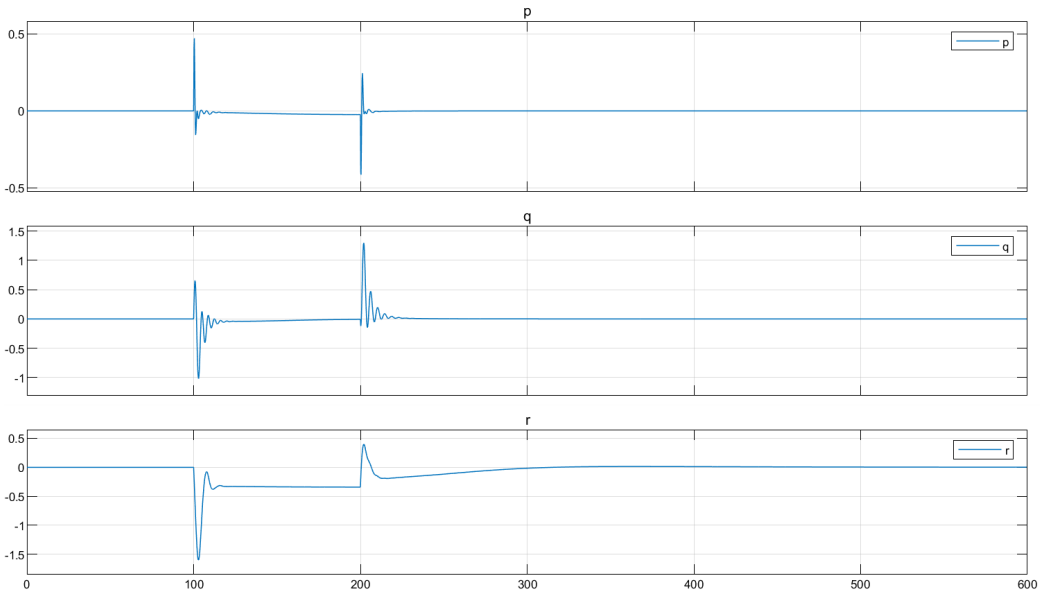
Test IV

For this test, an essentially different approach has been used: in fact, the results presented in the current sections are obtained through fractional order modeling of the system.

Fractional order modeling refers to a mathematical modeling approach where differential equations or transfer functions with fractional orders are used to describe the behavior of physical systems. In traditional modeling, the orders of the derivatives used to describe the system are integers, while in fractional order modeling, they can be any real number, typically between 0 and 1. This allows for a more flexible and accurate representation of complex systems with non-integer order dynamics, such as those found in biology, engineering, and finance. Fractional order modeling has gained increasing attention in recent years due to its potential for better understanding and



(a) $\boldsymbol{\nu}_1$ $[m/s]$



(b) $\boldsymbol{\nu}_2$ $[deg/s]$

Figure 5.5: Results of $\boldsymbol{\nu}$ from Test III

controlling complex systems.

Fractional order models are suitable for describing the behavior of real systems for several reasons:

- memory effect: many physical systems exhibit a memory effect, meaning that their current state depends on their previous states. This memory effect can be modeled using fractional derivatives, which allow for the consideration of non-instantaneous interactions in a system;
- irregular behavior: some physical systems exhibit irregular or non-smooth behavior, such as slow convergence, power-law decay, or long-tailed distributions. Fractional order models can better capture these irregular behaviors compared to integer order models;
- complexity: many real-world systems are complex and exhibit multiple time scales. Fractional order models can capture the multiple time scales and their interactions, which is difficult to achieve using integer order models;
- flexibility: fractional order modeling provides additional degrees of freedom, making it suitable to capture unmodeled dynamics, parameters uncertainties and unknown disturbances.

As already said, fractional calculus is a generalization of integration and differentiation to non-integer order fundamental operator ${}_aD_t^q$, where a and t are the bounds of the operation and $q \in \mathbb{R}$. The continuous integro-differential operator is defined as

$${}_aD_t^q = \begin{cases} \frac{d^q}{dt^q} & \text{for } q > 0 \\ 1 & \text{for } q = 0 \\ \int_a^t (d\tau)^q & \text{for } q < 0 \end{cases}$$

Among the most frequently used definitions for the general fractional operator, there is Caputo's definition of the fractional differential operator of order q:

$${}_aD_t^q f(t) = \frac{1}{\Gamma(n-q)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{q-n+1}}\, d\tau \qquad \text{for } (n-1 < q < n)$$

where

$$\Gamma(\alpha) = \int_0^\infty x^{\alpha-1} e^{-x}\, dx \qquad \text{for } \alpha > 0$$

is the gamma function.

For a deeper understanding of the mathematical theory behind these definitions, the reader may refer to [7, 24].

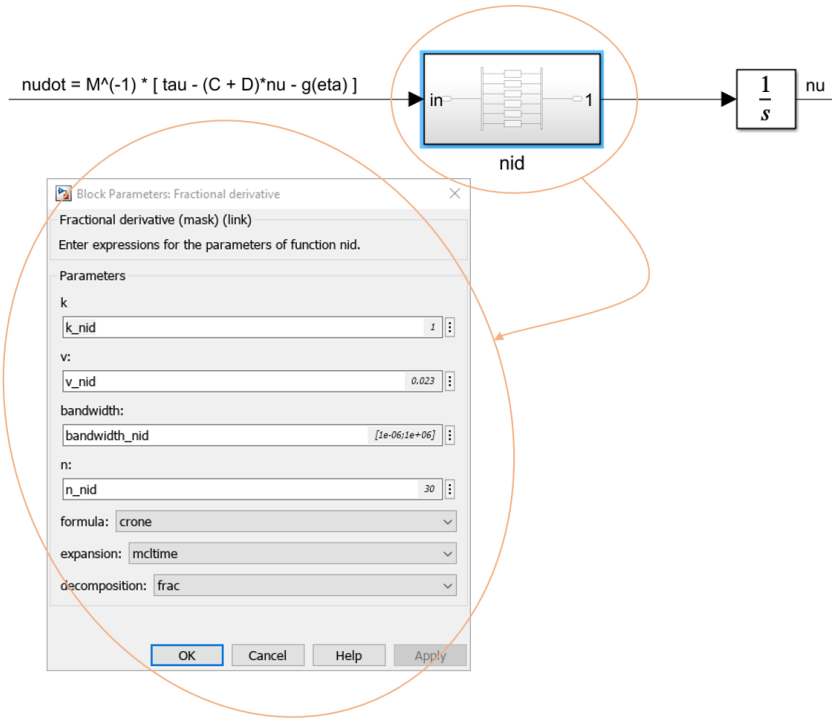


Figure 5.6: Usage of the nid block

Concerning the Simulink model, the implementation of fractional modeling consists in employing a fractional derivative before the integrator. To this aim, the Ninteger toolbox for Matlab [23] was used: in particular, the block designed to perform the fractional derivative is the nid block, where "nid" stands for non-integer derivative. The block is described by the following frequency domain transfer function:

$$C(s) = ks^{\nu}, \qquad \nu = 1 - \alpha \in \mathbb{R}$$

The dialogue window of the block is shown in Figure 5.6.

Parameter	**Setting**
r_b	$(\,0, 0, z_b\,)$
$B - W$	> 0
$\boldsymbol{D}_l$	complete
$\boldsymbol{M}_A$	AMCOMP
$\boldsymbol{\tau}_{\text{th}}$	step input along x_b
Solver	ode1be
Step size	fixed
t	$600\,s$

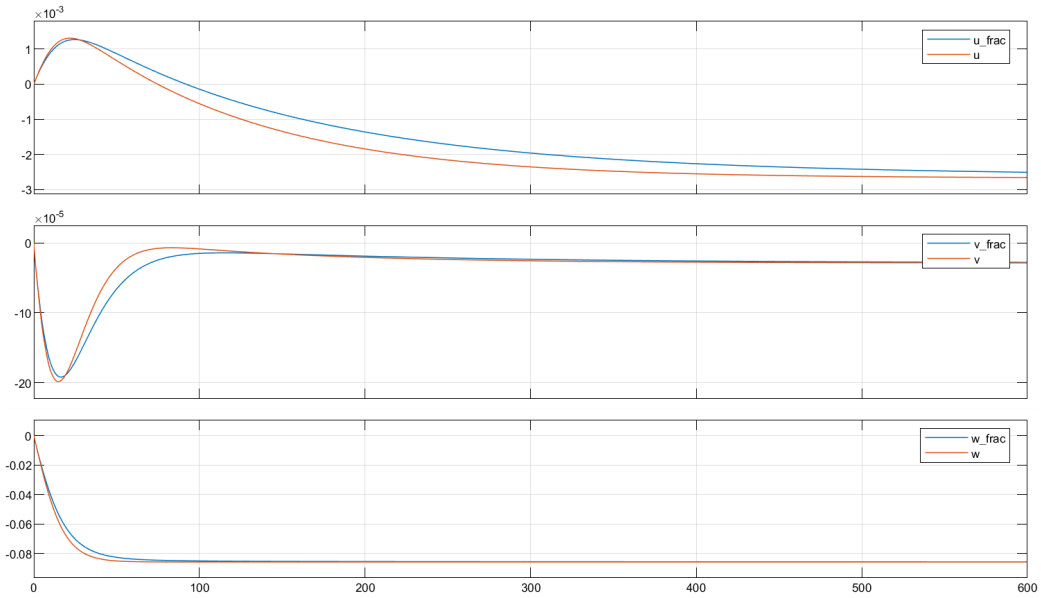
Table 5.4: Test IV settings

Table 5.4 displays the Simulink configuration used in this test, which is similar to the configurations used in Test I and Test II, but with a longer simulation time of $600\,s$. It is worth noting that the only parameter used to tune the fractional model was "v_nid", which corresponds to the exponent "v" previously introduced.

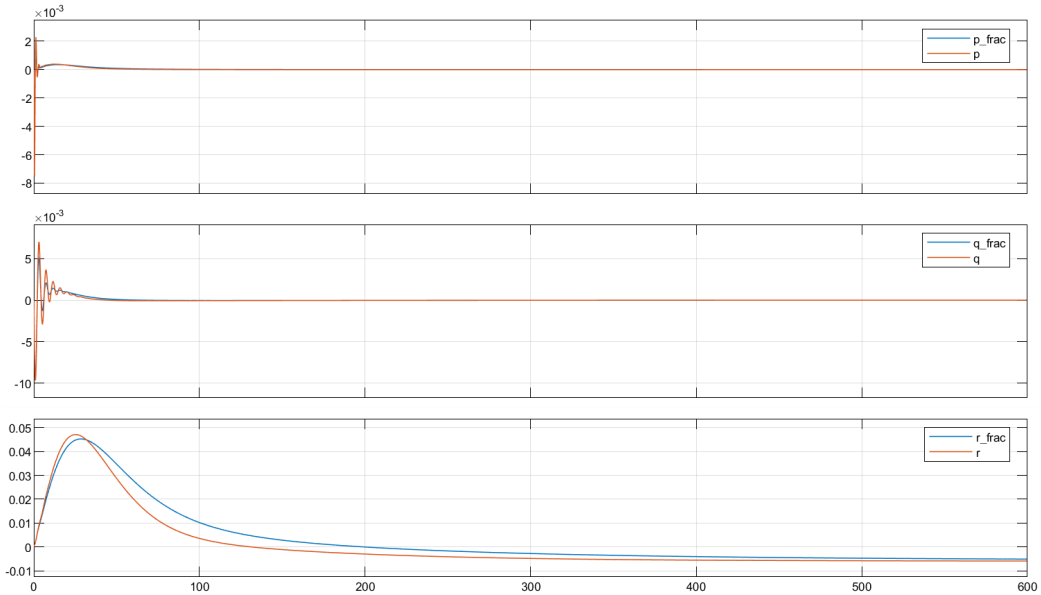
In Figure 5.7 on the next page, the results for $\alpha = 0.95$ are plotted together with the results for $\alpha = 1$ (which is the integer model case). In the figure, the fractional model curves are those in blue denoted by "_frac" in the legend.

The plotted velocities indicate that the fractional results exhibit the same tendency as the integer results, which suggests that the fractional system still accurately describes the system dynamics. Moreover, the initial oscillations of the roll and pitch rates are slightly damped away in the fractional model. This indicates that the fractional model has a smoothing effect on the system response, which could be advantageous in certain applications where noisy or erratic behavior needs to be minimized. Overall, the results might represent a first step in demonstrating the effectiveness of the fractional calculus approach in modeling the dynamics of complex systems. Of course, further investigation and validation on more complex systems will be necessary to fully assess its potential.

Furthermore, it may be interesting for this discussion to show one more result varying the parameter α. In Figure 5.8 on page 94, we can see the



(a) $\boldsymbol{\nu}_1$ $[m/s]$



(b) $\boldsymbol{\nu}_2$ $[deg/s]$

Figure 5.7: Results of $\boldsymbol{\nu}$ from Test IV, $\alpha = 0.95$

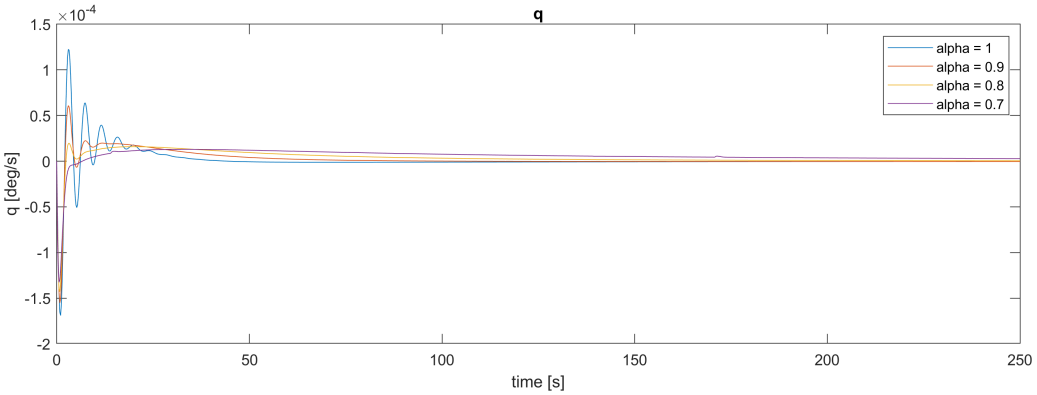


Figure 5.8: Results of q at different α

results for different values of α, ranging from $\alpha = 0.9$ to $\alpha = 0.7$. It can be observed that the initial oscillation of the pitch rate is influenced by the fractional exponent, as it decreases along with decreasing α. This again suggests that the fractional order model has a smoothing effect on the system response, which is more pronounced for lower values of α.

It is worth noting that the asymptotic behavior remains the same for all values of α. Nevertheless, further analyses on the steady state value should be made to assess whether or not a certain value of α can be deemed to be optimal for this particular system.

Chapter 6

Conclusions

The objective of this thesis was to identify the Fossen mathematical model [9] for the unmanned underwater drone Blucy developed within the Interreg Italy–Croatia SUSHI DROP Project. To do so, different tools and methods have been used throughout the thesis.

The first step consisted in studying the mathematical model (i.e. the Fossen model) for underwater vehicles. The assessment of the kinematics and kinetics parts and the understanding of the different hydrodynamic and hydrostatic forces acting on an underwater vehicle allowed for an aware selection of the parameters necessary for the model to work. Moreover, the mathematical model was preliminary translated into blocks and functions in the Matlab and Simulink environment.

To identify the parameters, starting from the real prototype and from the 2D drawings of its main components, the detailed 3D CAD model of Blucy was generated by means of SolidWorks. It was important to take into account the features of the real object, such as the accurate dimensions, densities and geometries of its parts. The 3D CAD model provided the values for the main mass, geometric and inertial parameters needed in the kinematic equations of the Fossen model.

Subsequently, the hydrodynamic parameters were identified by simplifying the complex 3D CAD model using SolidWorks. The simplified model was then pre-processed through SolidWorks and ANSA. Therefore, the meshing

of the body was performed. After the meshing, the CFD simulations were performed by means of OpenFOAM, for the linear motion simulations, and SolidWorks, for the simulations of angular motion. From the CFD analyses, all the forces and moments acting on Blucy in the different scenarios were obtained as functions of the simulated velocities (the simulated velocities have been selected based on real data from Blucy in field operations). These functions were analyzed through Matlab in order to derive the hydrodynamic parameters that characterize the dynamic part of the Fossen model. These parameters comprise the linear and nonlinear damping, which were accounted for by the aforementioned CFD simulations, as well as the hydrodynamic added mass coefficients.

Concerning the added mass coefficients, they were determined through two different methodologies: using the AMCOMP environment [6], specifically designed to estimate the added masses of complex bodies by employing the potential theory, and using Lamb's theory, which provides the exact solution for the added mass computation of an ellipsoid. The so obtained results were then compared.

Successively, the identified parameters were incorporated into Simulink to simulate Blucy dynamics. This served as a test bench for both the mathematical model and the parameters obtained during the CAD and CFD analyses. In fact, these tests were meant to investigate whether or not the identified coefficients well predicted Blucy behavior.

Referring to the results reported in Chapter 5, they were obtained by performing simulations of different maneuvers. From the outcomes, it was observed that the velocity curves exhibited a reasonable behavior consistent with what can be observed during a real mission. This suggests that the model and estimated parameters can already offer a valid description of Blucy.

The objective of applying fractional order modeling to the Simulink model of Blucy was to explore the potential benefits of this novel approach in capturing the complex dynamics of the underwater drone. By utilizing the Ninteger toolbox of Matlab to implement the fractional integrator, the aim was to investigate whether this approach could improve the accuracy and efficiency of the mathematical model.

The results obtained from the simulations showed that the fractional order model exhibited a stable and realistic trend with similar overall behavior to the integer model. These findings suggest that fractional order modeling could be a promising approach for accurately modeling the dynamics of complex systems such as underwater drones. Further exploration of this approach could lead to significant advancements in the design and development of such systems.

In conclusion, this work effectively identified and validated for the first time the Fossen mathematical model for the unmanned underwater drone Blucy. This achievement sets the fundamental basis for the development of novel navigation, guidance and control algorithms that can be employed to make Blucy autonomous. Furthermore, fractional modeling shows potential for enhancing control strategies by increasing their precision and adaptability.

Appendix A

Matlab Code

A.1 Initialization Script

```
%% BLUCY %%
% Initial conditions (for integrators)
x0     = 0;
y0     = 0;
z0     = 0;
phi0   = 0;
theta0 = 0;
psi0   = 0;

u0 = 0;
v0 = 0;
w0 = 0;
p0 = 0;
q0 = 0;
r0 = 0;

% Environment
g   = 9.81;  % [m/s^2]
rho = 1025;  % [kg/m^3] density of salt water

m                 = 216.4;  % [kg] Blucy mass
m_displaced_water = 216.6;  % [kg]
W = m_blucy*g;              % [N]  Blucy weight force
B = m_displaced_water*g;    % [N]  buoyancy force

% Inertia matrix [kg*m^2]
Ixx = 11.318;
```

```
Ixy = 0.015;
Ixz = 2.033;
Iyx = 0.015;
Iyy = 50.169;
Iyz = 0.017;
Izx = 2.033;
Izy = 0.017;
Izz = 42.682;

I_g = [Ixx  -Ixy  -Ixz;
      -Iyx   Iyy  -Iyz;
      -Izx  -Izy   Izz];

% Centre of gravity and centre of buoyancy position [m]
xg = 0;
yg = 0;
zg = 0;
r_g = [xg yg zg].';

xb =  0;
yb =  0;
zb = -0.0968;
r_b = [xb yb zb].';

% Added mass coefficients
Xudot = -56.41;
Xvdot =  5.8372;
Xwdot = -7.0623;
Xpdot =  2.2791;
Xqdot =  7.8604;
Xrdot = -6.5026;

Yudot =  5.8372;
Yvdot = -327.41;
Ywdot =  4.1352;
Ypdot =  4.7939;
Yqdot =  1.5077;
Yrdot =  62.28;

Zudot = -7.0623;
Zvdot =  4.1352;
Zwdot = -185.61;
Zpdot =  2.2847;
Zqdot =  6.2788;
Zrdot = -1.6736;

Kudot =  2.2791;
Kvdot =  4.7939;
Kwdot =  2.2847;
Kpdot =  0.1137;
```

```
Kqdot = -0.4063;
Krdot = -3.9641;

Mudot =  7.8604;
Mvdot =  1.5077;
Mwdot =  6.2788;
Mpdot = -0.4063;
Mqdot = -39.14;
Mrdot =  7.3145;

Nudot = -6.5026;
Nvdot =  62.28;
Nwdot = -1.6736;
Npdot = -3.9641;
Nqdot =  7.3145;
Nrdot = -82.40;

% Blucy useful matrices
M_RB = M_RB_fun(m_blucy,r_g,I);

M_A = -[Xudot Xvdot Xwdot Xpdot Xqdot Xrdot;  % from AMCOMP
        Yudot Yvdot Ywdot Ypdot Yqdot Yrdot;
        Zudot Zvdot Zwdot Zpdot Zqdot Zrdot;
        Kudot Kvdot Kwdot Kpdot Kqdot Krdot;
        Mudot Mvdot Mwdot Mpdot Mqdot Mrdot;
        Nudot Nvdot Nwdot Npdot Nqdot Nrdot];

M_A = diag([34 756 177 9 30 158]);            % from Lamb theory

M = M_RB + M_A;
Minv = inv(M);   % inverse of M

% Quadratic damping coefficients
Xuu = -61.96;
Yvv = -595.2;
Zww = -243;
Kpp = -23.67;
Mqq = -95.52;
Nrr = -240.4;

% Linear damping coefficients
Xu = -2.368;
Xv =  2.13;
Xw =  0.08366;
Xp = -0.09607;
Xq = -0.01479;
Xr = -0.7742;

Yu = -0.008119;
Yv = -27.58;
```

```
Yw =  0;
Yp = -1.436;
Yq =  0.004849;
Yr =  5.068;

Zu =  0.09057;
Zv =  18.02;
Zw = -2.06;
Zp = -0.1694;
Zq = -0.07099;
Zr = -0.3614;

Ku =  0.1185;
Kv =  5.269;
Kw = -0.006648;
Kp =  0.3026;
Kq =  0;
Kr =  0.7409;

Mu =  0.04597;
Mv =  0;
Mw = -0.1555;
Mp =  0.01799;
Mq = -0.06258;
Mr = -0.2689;

Nu =  0.1342;
Nv = -21.73;
Nw =  0;
Np =  0.4122;
Nq =  0;
Nr = -7.005;

% Linear damping matrix
Dl = -[Xu Xv Xw Xp Xq Xr;
       Yu Yv Yw Yp Yq Yr;
       Zu Zv Zw Zp Zq Zr;
       Ku Kv Kw Kp Kq Kr;
       Mu Mv Mw Mp Mq Mr;
       Nu Nv Nw Np Nq Nr];
```

A.2 M_RB_fun and skew_fun

```
% Computes the rigid-body inertia matrix
function M_RB = M_RB_fun(m,r_g,I_g)
```

```
skew_rg = skew_fun(r_g);

M_RB = [m*eye(3)    -m*skew_rg;
        m*skew_rg    I_g      ];
end



% Computes the skew symmetric matrix of a 3x3 vector
function skew_symmetric = skew_fun(lambda)
skew_symmetric = [    0      -lambda(3)  lambda(2);
                  lambda(3)     0      -lambda(1);
                 -lambda(2)  lambda(1)     0    ];
end
```

A.3 Ellipsoid Added Mass

```
% Ellispoid added mass from Lamb theory
rho = 1025;  % [kg/m^3] density of salt water

% Semiaxes [m]
a_blucy = 1;         % x-semiaxis
b_blucy = 0.350/2;   % y-semiaxis
c_blucy = 0.735/2;   % z-semiaxis
V_blucy = 0.42249;   % [m^3] Blucy volume from CAD

% Ellipsoid parameters: its semiaxes must have same ratio of Blucy
% dimensions and its volume must be equal to Blucy volume
b = (3/4*V_blucy*b_blucy^2/(pi*a_blucy*c_blucy))^(1/3);
c = c_blucy/b_blucy*b;
a = a_blucy/b_blucy*b;

% Make a visual comparison btw Blucy and Ellipsoid
figure()
ellipsoid(1.05,-0.5,0.375,a,b,c,50)  % plot ellipsoid
axis('equal'); hold on
fid = stlread("blucy_noTh.STL");  % read STL file
stl = trimesh(fid);               % plot STL
stl.FaceColor = '#0079c1';
stl.EdgeColor = '#ddbf5f';

m_w = 4/3*pi*rho*a*b*c;  % mass of displaced water

Ixx = 4/15*pi*rho*a*b*c*(b^2+c^2);  % inertia of displaced water
Iyy = 4/15*pi*rho*a*b*c*(a^2+c^2);
```

```
Izz = 4/15*pi*rho*a*b*c*(a^2+b^2);

% Lamb's formulas
funA0 = @(u) 1./((a.^2+u).*sqrt((a.^2+u).*(b.^2+u).*(c.^2+u)));
funB0 = @(u) 1./((b.^2+u).*sqrt((a.^2+u).*(b.^2+u).*(c.^2+u)));
funC0 = @(u) 1./((c.^2+u).*sqrt((a.^2+u).*(b.^2+u).*(c.^2+u)));

A0 = a*b*c*integral(funA0,0,inf);
B0 = a*b*c*integral(funB0,0,inf);
C0 = a*b*c*integral(funC0,0,inf);

% Dimensional added mass coefficients
lambda11 = A0/(2-A0)*m_w;
lambda22 = B0/(2-B0)*m_w;
lambda33 = C0/(2-C0)*m_w;
lambda44 = 1/5*(b^2-c^2)^2*(C0-B0)/(2*(b^2-c^2)+(b^2+c^2)*(B0-C0))*m_w;
lambda55 = 1/5*(c^2-a^2)^2*(A0-C0)/(2*(c^2-a^2)+(c^2+a^2)*(C0-A0))*m_w;
lambda66 = 1/5*(a^2-b^2)^2*(B0-A0)/(2*(a^2-b^2)+(a^2+b^2)*(A0-B0))*m_w;

% Nondimensional added mass coefficients
k11 = lambda11/m_w;
k22 = lambda22/m_w;
k33 = lambda33/m_w;
k44 = lambda44/Ixx;
k55 = lambda55/Iyy;
k66 = lambda66/Izz;

% Dimensional added mass matrix
dim_AM_mat = diag([lambda11 lambda22 lambda33 lambda44 lambda55 lambda66]);

% Nondimensional added mass matrix
nondim_AM_mat = diag([k11 k22 k33 k44 k55 k66]);
```

Appendix B

OpenFOAM Code

B.1 "0" Directory

```
/*--------------------------------------------------------*/
/*--------------------------U---------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    arch        "LSB;label=32;scalar=64";
    class       volVectorField;
    location    "0";
    object      U;
}
/*--------------------------------------------------------*/
dimensions      [0 1 -1 0 0 0 0];
internalField   uniform (-1 0 0);
boundaryField
{
    Drone
    {
        type            noSlip;
    }
    Inlet
    {
        type            fixedValue;
        value           uniform (-1 0 0);
    }
    Outlet
    {
```

```
        type            zeroGradient;
    }
    Symmetry
    {
        type            symmetry;
    }
}
/*---------------------------------------------------------*/
/*--------------------------p--------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      p;
}
/*---------------------------------------------------------*/
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    Drone
    {
        type            zeroGradient;
    }
    Inlet
    {
        type            zeroGradient;
    }
    Outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
    Symmetry
    {
        type            symmetry;
    }
}
/*---------------------------------------------------------*/
/*--------------------------k--------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
```

```
    object      k;
}
/*---------------------------------------------------------*/
dimensions      [0 2 -2 0 0 0 0];
internalField   uniform 0.003;
boundaryField
{
    Drone
    {
        type            kqRWallFunction;
        value           uniform 0;
    }
    Inlet
    {
        type            turbulentIntensityKineticEnergyInlet;
        intensity       0.01;
        value           uniform 0.003;
    }
    Outlet
    {
        type            inletOutlet;
        inletValue      uniform 0.003;
        value           uniform 0.003;
    }
    Symmetry
    {
        type            symmetry;
    }
}
/*---------------------------------------------------------*/
/*-----------------------epsilon-------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      epsilon;
}
/*---------------------------------------------------------*/
dimensions      [0 2 -3 0 0 0 0];
internalField   uniform 9e-05;
boundaryField
{
    Drone
    {
        type            epsilonWallFunction;
        value           uniform 9e-05;
    }
```

```
    Inlet
    {
        type            turbulentMixingLengthDissipationRateInlet;
        mixingLength    2;
        value           uniform 9e-05;
    }
    Outlet
    {
        type            inletOutlet;
        inletValue      uniform 9e-05;
        value           uniform 9e-05;
    }
    Symmetry
    {
        type            symmetry;
    }
}
/*--------------------------------------------------------*/
/*--------------------------nut---------------------------*/
FoamFile
{
    version     2.0;
    format      binary;
    arch        "LSB;label=32;scalar=64";
    class       volScalarField;
    location    "0";
    object      nut;
}
/*--------------------------------------------------------*/
dimensions      [0 2 -1 0 0 0 0];
internalField   uniform 0;
boundaryField
{
    Drone
    {
        type            nutUSpaldingWallFunction;
        value           uniform 0;
    }
    Inlet
    {
        type            calculated;
        value           uniform 0;
    }
    Outlet
    {
        type            calculated;
        value           uniform 0;
    }
    Symmetry
    {
```

```
        type            symmetry;
    }
}
/*---------------------------------------------------------*/
```

B.2 fvSchemes

```
FoamFile
{
    version 2.0;
    format binary;
    class dictionary;
    location "system";
    object fvSchemes;
}
/*---------------------------------------------------------*/
ddtSchemes
{
    default         CrankNicolson 0.7;
}

gradSchemes
{
    default         cellMDLimited Gauss linear 0;
    grad(U)         cellMDLimited Gauss linear 0.333;
    grad(CDO)       Gauss cubic;
}

divSchemes
{
    default         none;
    div(phi,U)      Gauss linearUpwind grad(U);
    div(phi,k)      Gauss linearUpwind default;
    div(phi,omega)  Gauss linearUpwind default;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
    div(phi,R)      Gauss upwind;
    div(R)          Gauss linear;
    div(phid,p)     Gauss limitedLinear 1;
    div(phi,K)      Gauss linearUpwind default;
    div(phi,e)      Gauss limitedLinear 1;
    div(((rho*nuEff)*dev2(T(grad(U))))) Gauss linear;
    div(phi,epsilon) bounded Gauss linearUpwind grad(epsilon);
    div(phi,CDO)     Gauss limitedLinear 1;
    div((nu*dev2(T(grad(U))))) Gauss linear;
    div(phi,nuTilda) bounded Gauss linearUpwind grad(nuTilda);
```

```
}

laplacianSchemes
{
    default         Gauss linear limited 1.0;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    default         limited 1.0;
}

oversetInterpolationSuppressed
{
    grad(p);
    surfaceIntegrate(phiHbyA);
}

fluxRequired
{
    default         no;
    pcorr           ;
    p               ;
}

oversetInterpolation
{
    method          cellVolumeWeight;
}

wallDist
{
method meshWave;
}
```

B.3 fvsolution

```
FoamFile
{
    version 2.0;
```

```
    format binary;
    class dictionary;
    location "system";
    object fvSolution;
}
/*---------------------------------------------------------*/
solvers
{
"pcorr.*"
    {
        solver           GAMG;
        tolerance        0.01;
        relTol           0;
        smoother         GaussSeidel;
        cacheAgglomeration on;
        agglomerator faceAreaPair;
        nCellsInCoarsestlevel 1000;
        mergelevels 1;
   }
    p
    {
        $pcorr;
        tolerance        1e-7;
        relTol           0.0;
    }
    pFinal
    {
        $p;
        tolerance        1e-7;
        relTol           0;
    }
    "(k|omega)"
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-07;
        relTol          0;
    }
    "(k|omega|epsilon|R|nuTilda)Final"
    {
        $U;
        tolerance       1e-7;
        relTol          0;
    }
    U
    {
       solver           PBiCGStab;
       preconditioner   DILU;
       tolerance        1e-8;
       relTol           0;
```

```
}
UFinal
{
  $U;
  tolerance         1e-9;
}
cellDisplacement
{
    solver          GAMG;
    tolerance       1e-8;
    relTol          0;
    smoother        DIC;
}
Phi
{
    solver          GAMG;
    smoother        DIC;

    tolerance       1e-06;
    relTol          0.01;
}
cellMotionUz
{
    solver          PCG;
    preconditioner  DIC;
    tolerance       1e-08;
    relTol          0;
}
 residualControl
{
    "(p)"
    {
        tolerance 1e-3;
        relTol 0;
    }

    "(U)"
    {
        tolerance 1e-3;
        relTol 0;
    }
}
 CDO
{
solver          smoothSolver;
// preconditioner  DILU;
smoother         GaussSeidel;
tolerance       1e-08;
relTol          0;
minIter      1;
```

```
    }
}
SIMPLE
{
    nNonOrthogonalCorrectors 10;
    tolerance 1.0e-14;
    relTol 5e-3;
    consistent  true;
    residualControl
    {
        "(p|U)" 1e-3;

    }
}
potentialFlow
{
    nNonOrthogonalCorrectors 3;
}
relaxationFactors
{
    fields
    {
        p               0.7;
    }
    equations
    {
        "(U|k|omega)"   0.3;
        "(U|k|omega)Final" 1.0;
    }
}
```

B.4 controlDict

```
FoamFile
{
    version 2.0;
    format binary;
    class dictionary;
    location "system";
    object controlDict;
}
/*---------------------------------------------------------*/
application simpleFoam;
startFrom startTime;
startTime 0;
```

```
stopAt endTime;
endTime 1000;
ΔT  1;
writeControl timeStep;
writeInterval   10;
purgeWrite  0;
writeFormat binary;
writePrecision  6;
writeCompression    uncompressed;
timeFormat  general;
timePrecision   6;
graphFormat raw;
runTimeModifiable   yes;
adjustTimeStep  off;
maxCo   0.5;
maxAlphaCo  0.5;
maxDeltaT   0.01;

functions {
    forces1
{
    // Mandatory entries
    type            forces;
    libs            ("libforces.so");
    patches         (Drone);
    // Optional entries
    // Field names
    p               p;
    U               U;
    rho             rhoInf;
    rhoInf          1025;
    // Reference pressure [Pa]
    pRef            0;
    // Include porosity effects?
    porosity        no;
    // Store and write volume field representations of forces and moments
    writeFields     yes;
    writeInterval   10;
    // Centre of rotation for moment calculations
    CofR            (-0.86 0 -0.325);
}
    yPlus
{
        // Mandatory entries (unmodifiable)
        type            yPlus;
        libs            (fieldFunctionObjects);
        writeInterval   10;
}
}
```

Appendix C

CFD Results

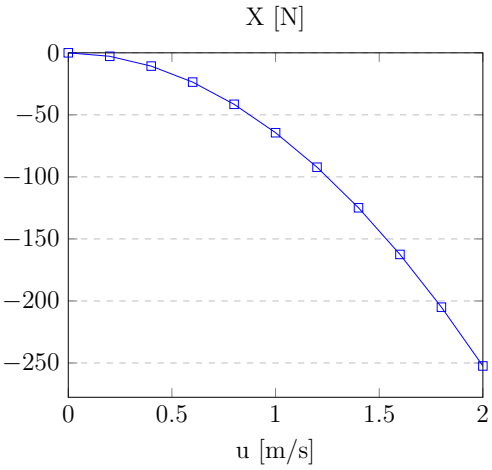
This appendix presents the results of the CFD simulations carried out on the AUV Blucy submersible through OpenFOAM (used to simulate the linear motion) and SolidWorks Flow Simulation (used to simulate the angular motion). The results are presented in the form of graphs showing the forces and moments acting on the AUV under various motion conditions, including different linear velocities of surge, sway, and heave, as well as different angular velocities of roll, pitch, and yaw. These results are among the most important findings of this study, as they provided key insights into the hydrodynamic performance of the AUV and enabled the identification of its main parameters.

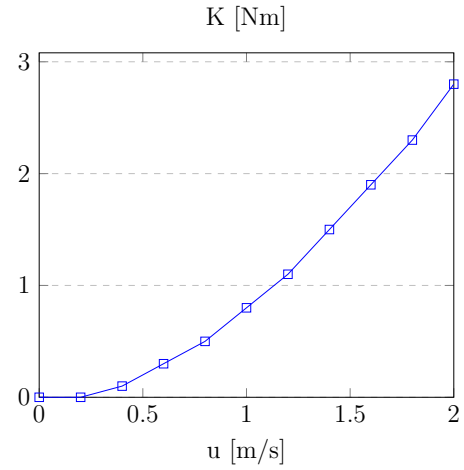
The comprehensive analysis of these results allowed for the identification of the linear and nonlinear damping terms, which are essential for building an accurate underwater drone model.

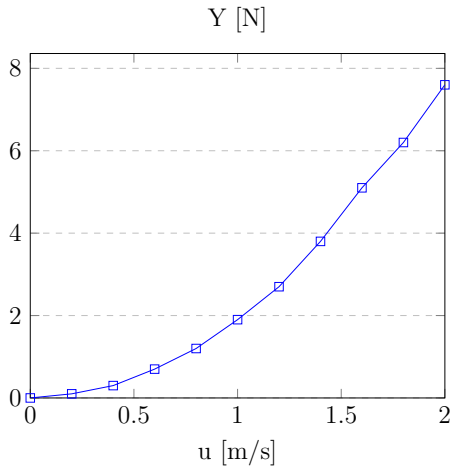
It is important to note that some of the graphs in this appendix contain scattered data that could not be accurately fit by a polynomial function of the second degree. This is likely due to the fact that the forces and moments acting on Blucy in these cases were relatively small, causing the simulation error to be larger than the physical quantities being analyzed. These curves resulted in the respective hydrodynamic coefficients being set equal to zero.

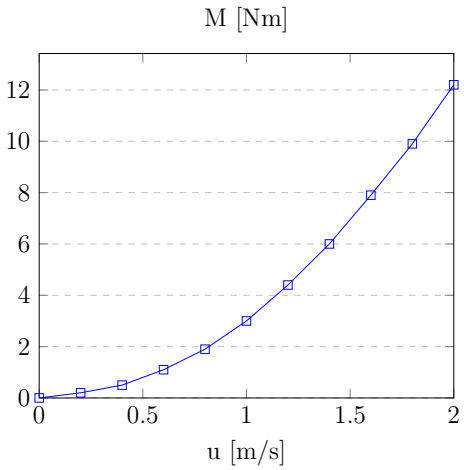
The results are presented divided into different sections, with each section providing the curves for a different kind of motion.

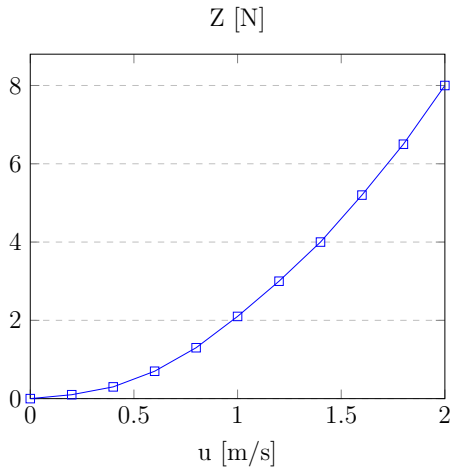
C.1 Surge

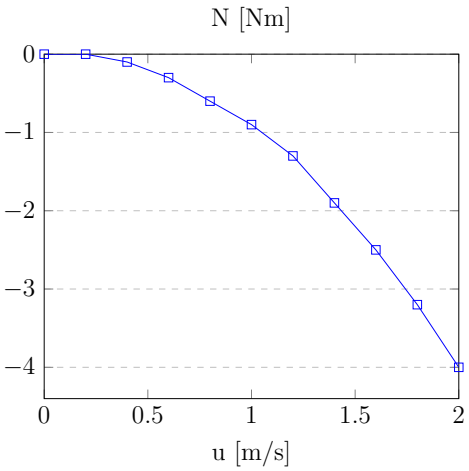

X [N]
0
−50
−100
−150
−200
−250
0
0.5
1
1.5
2
u [m/s]


K [Nm]
3
2
1
0
0
0.5
1
1.5
2
u [m/s]

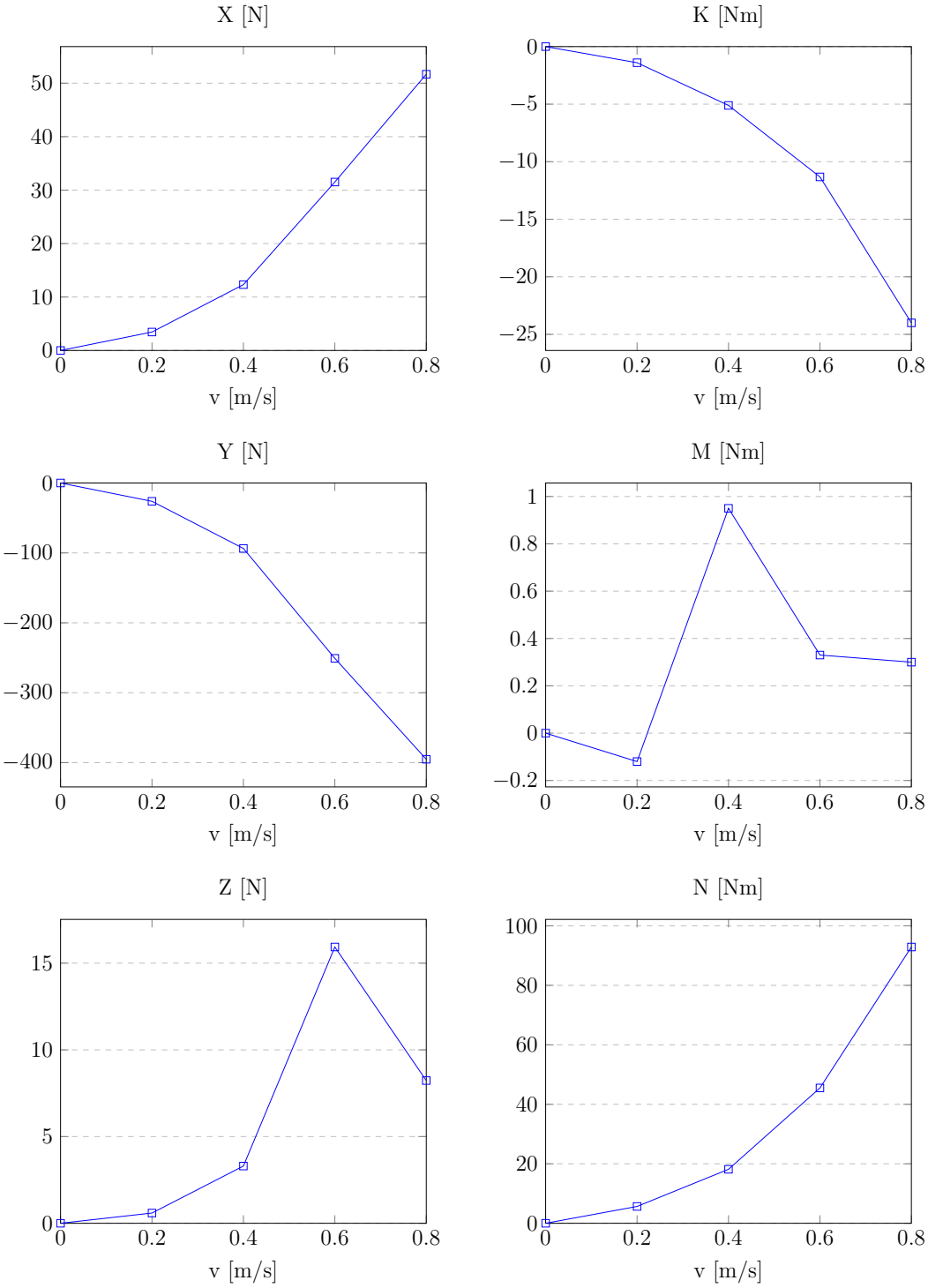

Y [N]
8
6
4
2
0
0
0.5
1
1.5
2
u [m/s]


M [Nm]
12
10
8
6
4
2
0
0
0.5
1
1.5
2
u [m/s]

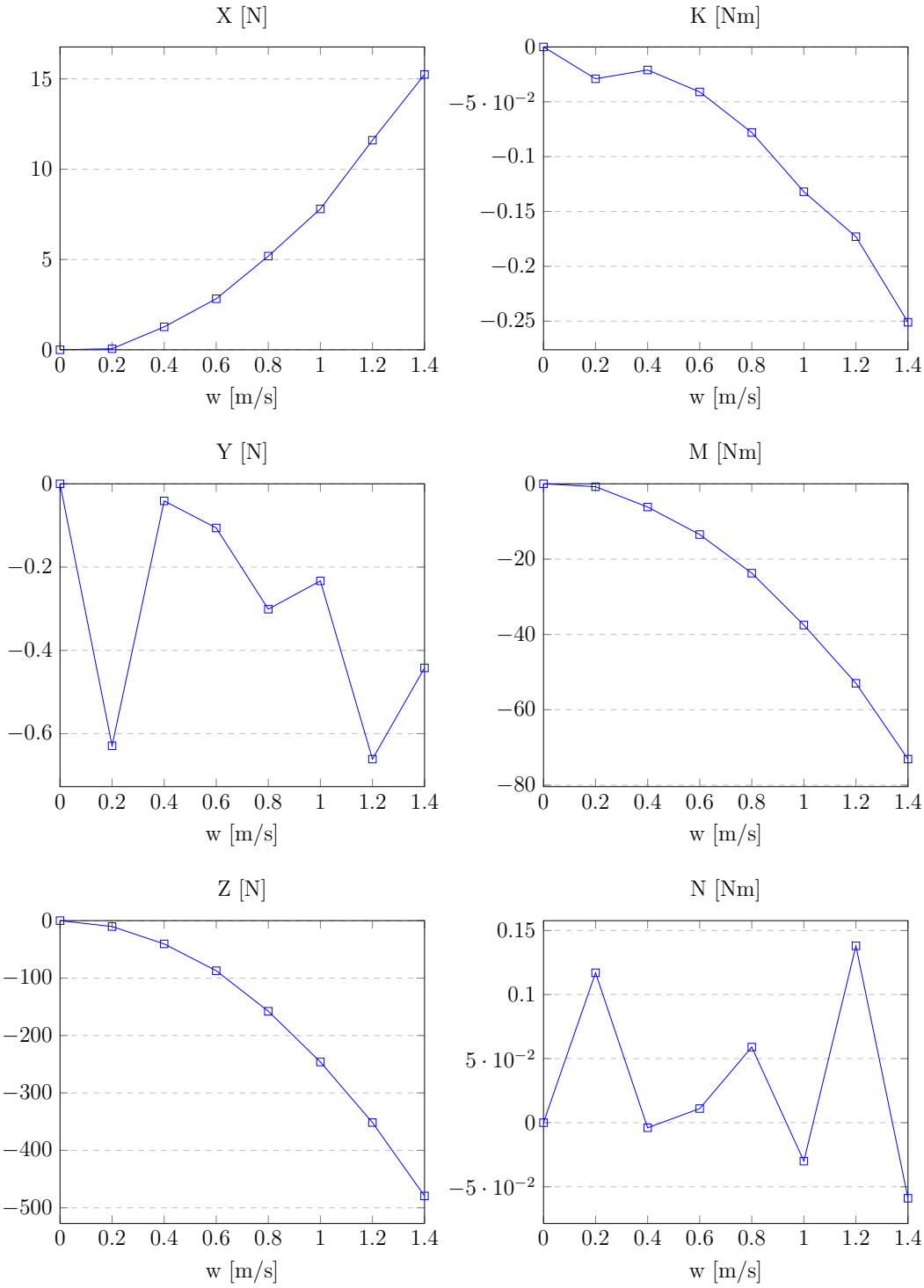

Z [N]
8
6
4
2
0
0
0.5
1
1.5
2
u [m/s]


N [Nm]
0
−1
−2
−3
−4
0
0.5
1
1.5
2
u [m/s]

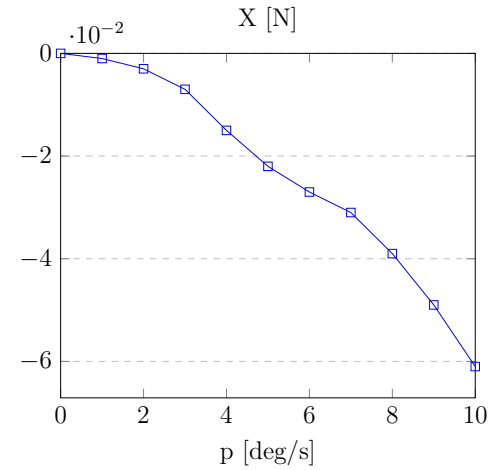
C.2 Sway

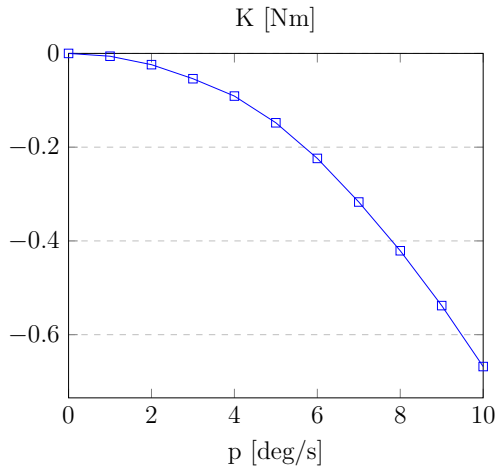

X [N]
50
40
30
20
10
0
0
0.2
0.4
0.6
0.8
v [m/s]
K [Nm]
0
−5
−10
−15
−20
−25
0
0.2
0.4
0.6
0.8
v [m/s]
Y [N]
0
−100
−200
−300
−400
0
0.2
0.4
0.6
0.8
v [m/s]
M [Nm]
1
0.8
0.6
0.4
0.2
0
−0.2
0
0.2
0.4
0.6
0.8
v [m/s]
Z [N]
15
10
5
0
0
0.2
0.4
0.6
0.8
v [m/s]
N [Nm]
100
80
60
40
20
0
0
0.2
0.4
0.6
0.8
v [m/s]

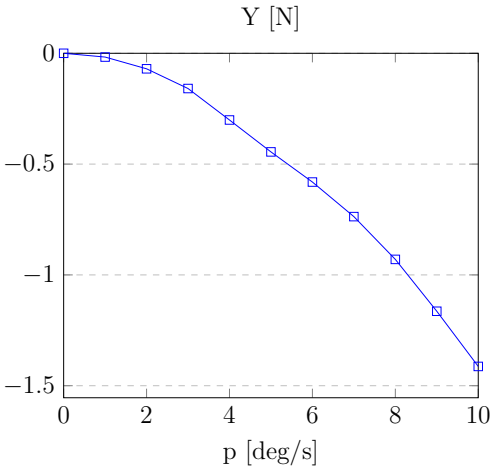
C.3 Heave

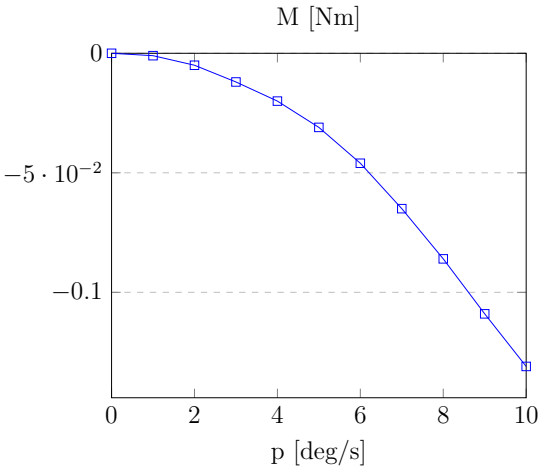

X [N]
15
10
5
0
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]
K [Nm]
0
$-5 \cdot 10^{-2}$
−0.1
−0.15
−0.2
−0.25
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]
Y [N]
0
−0.2
−0.4
−0.6
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]
M [Nm]
0
−20
−40
−60
−80
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]
Z [N]
0
−100
−200
−300
−400
−500
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]
N [Nm]
0.15
0.1
$5 \cdot 10^{-2}$
0
$-5 \cdot 10^{-2}$
0 0.2 0.4 0.6 0.8 1 1.2 1.4
w [m/s]

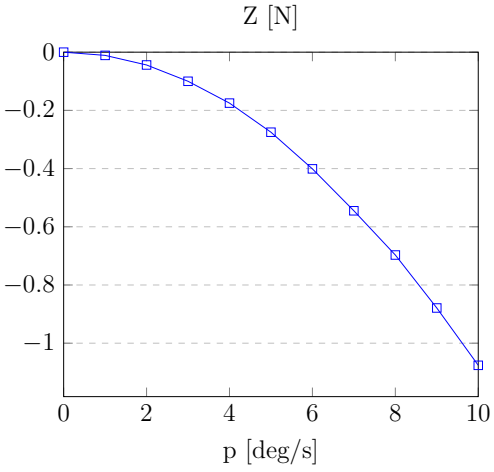
C.4 Roll

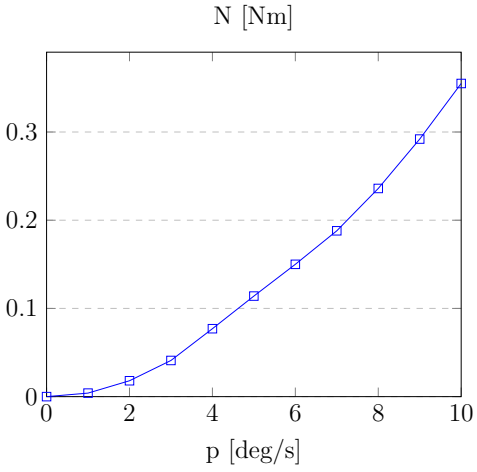

X [N]
$\cdot 10^{-2}$
0
−2
−4
−6
0
2
4
6
8
10
p [deg/s]


K [Nm]
0
−0.2
−0.4
−0.6
0
2
4
6
8
10
p [deg/s]

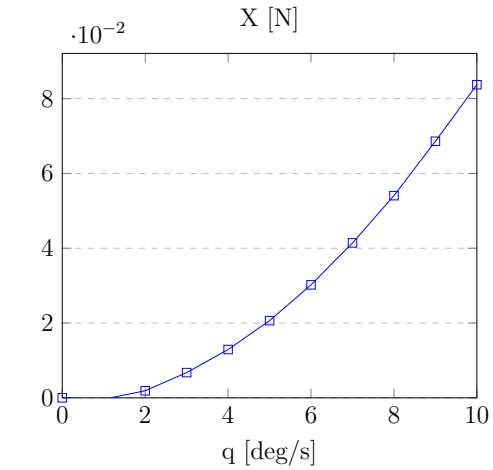

Y [N]
0
−0.5
−1
−1.5
0
2
4
6
8
10
p [deg/s]

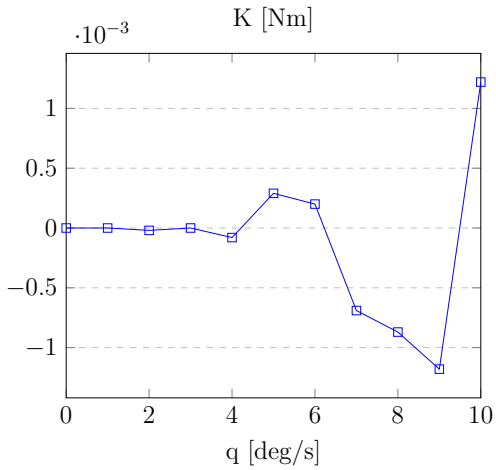

M [Nm]
0
$-5 \cdot 10^{-2}$
−0.1
0
2
4
6
8
10
p [deg/s]

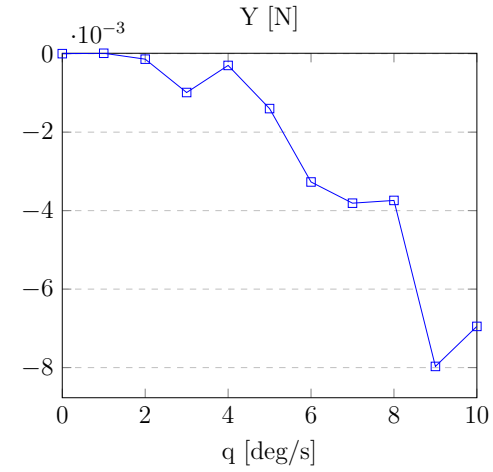

Z [N]
0
−0.2
−0.4
−0.6
−0.8
−1
0
2
4
6
8
10
p [deg/s]

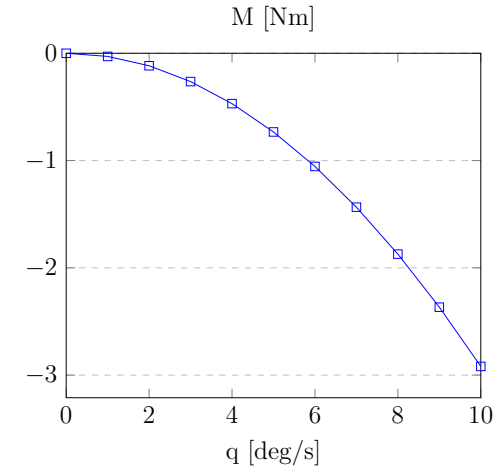

N [Nm]
0.3
0.2
0.1
0
0
2
4
6
8
10
p [deg/s]

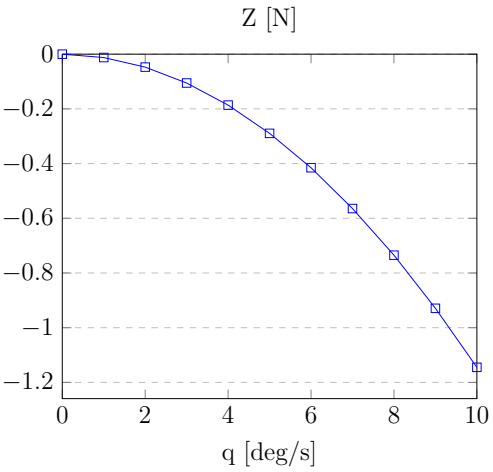
C.5 Pitch

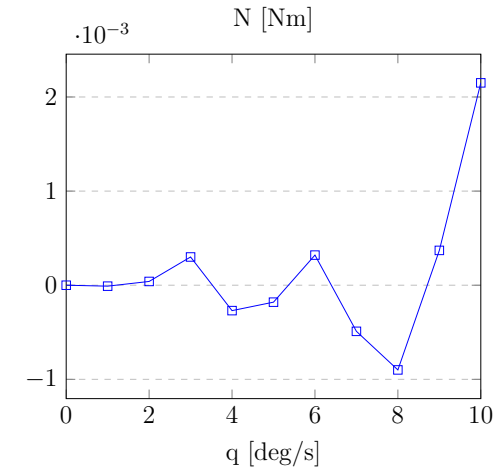

X [N]
$\cdot 10^{-2}$
8
6
4
2
0
0
2
4
6
8
10
q [deg/s]


K [Nm]
$\cdot 10^{-3}$
1
0.5
0
−0.5
−1
0
2
4
6
8
10
q [deg/s]

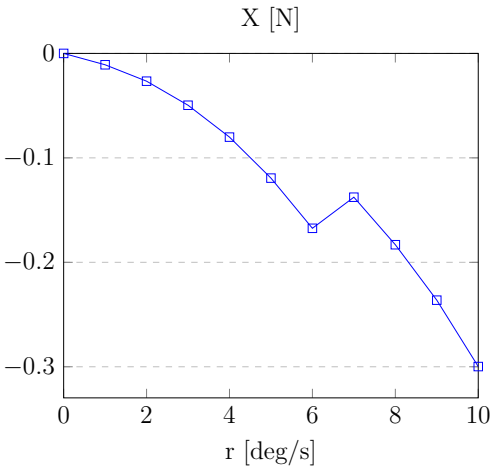

Y [N]
$\cdot 10^{-3}$
0
−2
−4
−6
−8
0
2
4
6
8
10
q [deg/s]

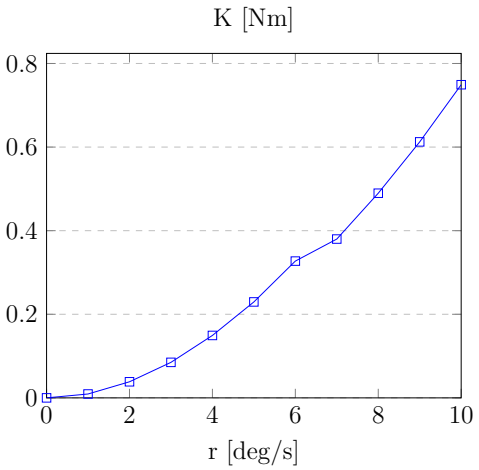

M [Nm]
0
−1
−2
−3
0
2
4
6
8
10
q [deg/s]

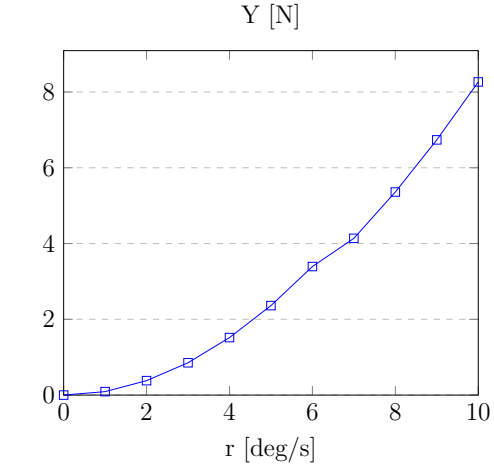

Z [N]
0
−0.2
−0.4
−0.6
−0.8
−1
−1.2
0
2
4
6
8
10
q [deg/s]

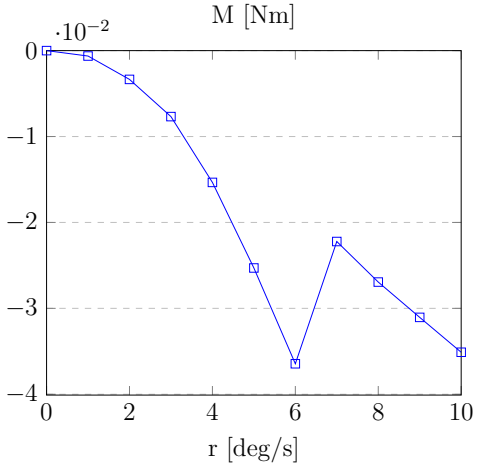

N [Nm]
$\cdot 10^{-3}$
2
1
0
−1
0
2
4
6
8
10
q [deg/s]

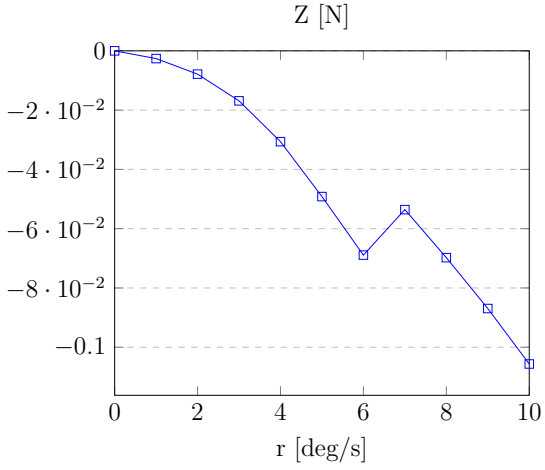
C.6 Yaw

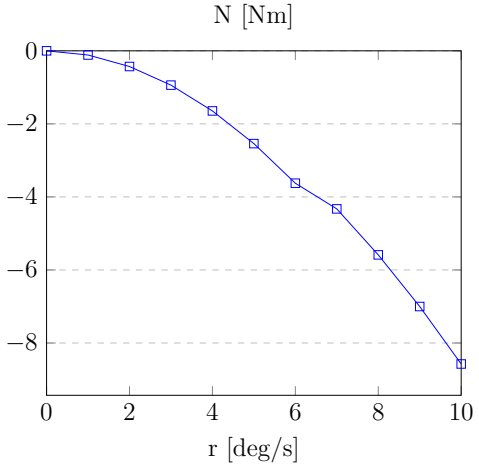

X [N]
0
−0.1
−0.2
−0.3
0
2
4
6
8
10
r [deg/s]


K [Nm]
0.8
0.6
0.4
0.2
0
0
2
4
6
8
10
r [deg/s]


Y [N]
8
6
4
2
0
0
2
4
6
8
10
r [deg/s]


M [Nm]
·10−2
0
−1
−2
−3
−4
0
2
4
6
8
10
r [deg/s]


Z [N]
0
−2 · 10−2
−4 · 10−2
−6 · 10−2
−8 · 10−2
−0.1
0
2
4
6
8
10
r [deg/s]


N [Nm]
0
−2
−4
−6
−8
0
2
4
6
8
10
r [deg/s]

Bibliography

[1] MATLAB. Version 9.13.0 (R2022b). Natick, Massachusetts: The MathWorks Inc., 2022. URL: https://www.mathworks.com.

[2] Yannick Allard and Elisa Shahbazian. *Unmanned underwater vehicle (UUV) information study*. Tech. rep. OODA Technologies Inc Montreal, Quebec Canada, 2014.

[3] Gianluca Antonelli et al. *Underwater robots: motion and force control of vehicle-manipulator systems*. Vol. 2. Springer, 2006.

[4] Raffaele Borgognoni. *Mathematical model identification of the AUV-ROV Blucy from the SUSHI DROP Project*. Unpublished. Internship report submitted to the University of Bologna. 2021.

[5] P. Castaldi et al. "Autonomous Underwater Vehicle Actuators Health Monitoring for Smart Harbour Application". In: *2020 5th International Conference on Smart and Sustainable Technologies (SpliTech)*. 2020, pp. 1–6. DOI: 10.23919/SpliTech49282.2020.9243818.

[6] Alessandro Ceruti, T Bombardi, and Piergiovanni Marzocca. "A CAD environment for the fast computation of Added Masses". In: *Ocean Engineering* 142 (2017), pp. 329–337.

[7] Shantanu Das. *Functional fractional calculus*. Vol. 1. Springer, 2011.

[8] Thor I. Fossen. "Guidance and Control of Ocean Vehicles. John Willey & Sons". In: *Inc., New York* (1994).

[9] Thor I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[10] Thor I. Fossen. *Nonlinear modelling and control of underwater vehicles*. Universitetet i Trondheim (Norway), 1991.

[11] Thor I. Fossen and Ola-Erik Fjellstad. "Nonlinear modelling of marine vehicles in 6 degrees of freedom". In: *Mathematical Modelling of Systems* 1.1 (1995), pp. 17–27.

[12] Salimzhan A. Gafurov and Evgeniy V. Klochkov. "Autonomous unmanned underwater vehicles development tendencies". In: *Procedia Engineering* 106 (2015), pp. 141–148.

[13] Julián González Agudelo. "Contribution to the model and navigation control of an autonomous underwater vehicle". In: (2015).

[14] Ali Hammoud et al. "Design and dynamic modeling of ROVs: estimating the damping and added mass parameters". In: *Ocean Engineering* 239 (2021), p. 109818.

[15] Frederick H. Imlay. *The complete expressions for added mass of a rigid body moving in an ideal fluid*. Tech. rep. DAVID TAYLOR MODEL BASIN WASHINGTON DC, 1961.

[16] ITALY-CROATIA.EU. *SUSHI DROP SUstainable fiSHeries wIth DRONes data Processing*. 2019. URL: `https://www.italy-croatia.eu/web/sushidrop`.

[17] Inc. Kitware. *ParaView*. 2021. URL: `https://www.paraview.org`.

[18] Horace Lamb. *Hydrodynamics, 6th edition*. Dover Publications, Inc., 1945.

[19] A Lambertini et al. "Monitoring and Surveying from an Underwater Vehicle in SUSHI DROP Project". In: *2021 International Workshop on Metrology for the Sea; Learning to Measure Sea Health Parameters (MetroSea)*. IEEE. 2021, pp. 189–193.

[20] Alessandro Lambertini et al. "Underwater Drone Architecture for Marine Digital Twin: Lessons Learned from SUSHI DROP Project". In: *Sensors* 22.3 (2022), p. 744.

[21] OpenCFD Ltd. *v2212*. 2022. URL: `https://www.openfoam.com`.

[22] U.S. Navy. *The Navy Unmanned Undersea Vehicle (UUV) Master Plan*. CreateSpace Independent Publishing Platform, 2014.

[23] D Pedro Mata de Oliveira Valério. "Ninteger v. 2.3 Fractional control toolbox for MATLAB". In: *Lisboa, Universidade Technical* (2005).

[24] Ivo Petráš. "Modeling and numerical analysis of fractional-order Bloch equations". In: *Computers & Mathematics with Applications* 61.2 (2011), pp. 341–356.

[25] Consiglio Nazionale delle Ricerche. *SushiDrop AUV/ROV, Manuale di uso e manutenzione*. Oct. 2018.

[26] SOLIDWORKS. *Version 2021*. Dassault Systèmes, 2021. URL: `https://www.solidworks.com`.

[27] BETA CAE Systems. *Version 2021*. 2021. URL: `https://www.beta-cae.com`.

[28] Marco Tuveri, Alessandro Ceruti, and Pier Marzocca. "Added masses computation for unconventional airships and aerostats through geometric shape evaluation and meshing". In: *International Journal of Aeronautical and Space Sciences* 15.3 (2014), pp. 241–257.

[29] Christopher Von Alt. "Autonomous underwater vehicles". In: *Autonomous Underwater Lagrangian Platforms and Sensors Workshop*. Vol. 3. 2003, p. 2.