MDPI

Article

Simu2VITA: A general purpose underwater vehicle simulator

Pedro Daniel de Cerqueira Gava *, Cairo Lúcio Nascimento Júnior , Juan R. B. F. Silva and Geraldo José Adabo

Division of Electronic Engineering, Instituto Tecnológico de Aeronáutica, São José dos Campos, SP, Brazil; pdcg@ita.br, cairo@ita.br, juan@ita.br, adabo@ita.br

* Correspondence: pdcg@ita.br

Abstract: This article presents an Unmanned Underwater Vehicle simulator named Simu2VITA which was designed to be rapid to setup, easy to use, and simple to modify the vehicle's parameters. Simulation of the vehicle dynamics is divided into three main Modules: the Actuator Module, the Allocation Module and the Dynamics Model. The Actuator Module is responsible for the simulation of actuators such as propellers and fins, the Allocation Module translates the action of the actuators into forces and torques acting on the vehicle and the Dynamics Module implements the dynamics equations of the vehicle. Simu2VITA implements the dynamics of the actuators and of the rigid body of the vehicle using the MATLAB/Simulink® framework. To show the usefulness of the Simu2VITA simulator, simulation results are presented for an unmanned underwater vehicle navigating inside a fully flooded tunnel and then compared with sensor data collected when the real vehicle performed the same mission using the controllers designed employing the simulator.

Keywords: Underwater Unmanned Vehicle, Simulation, Mobile Vehicle Dynamics

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Sensors* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Working with mobile vehicles often proves to be time consuming and, adding to the natural complexity of the matter, typically there is also the additional burden of using complicated simulators. Simulators are a necessity when dealing with mobile vehicles since they allow the design team to increase its knowledge about the vehicle's behaviour and to test different scenarios. Quality of simulation is a requisite that rapidly grows in importance as the cost of equipment increases and the environment gets more hazardous to operate.

Our research project aims to design an Underwater Unmanned Vehicle (UUV) to be used for inspection of adduction tunnels in hydroelectric power plants. Initially a search was done for possible simulators for this scenario that would satisfy the following requisites:

- overall design simple and easy to understand,
- easy description and modification of the vehicle physical parameters, its actuators and its sensors,
- rapid testing of the different types of speed and position controllers, and
- simple to add features on top of it such as vehicle autonomous behaviours.

Nowadays popular consolidated robotics simulators like Gazebo [1] offers great physics accuracy in simulation and in customization but its learning curve is steep. The same happens with rich-feature simulators like Webots [2] . The setup of these simulators was considered too complicated by our team since they require complex file-based descriptions of the vehicle and other elements.

In this article we show a simple, yet complete, UUV simulator which was built on top of the MATLAB/Simulink® software framework[1] given its popularity among engineers and for being the academia and industry standard for simulation of mechanical

[1] MATLAB® and Simulink® are registered trademarks of The MathWorks, Inc.

and electrical systems. To use this simulator one needs to define explicitly only the vehicle parameters. Modelling the vehicle dynamics and its actuators is simple and accurate.

In this paper we introduce our simulator named **Simu2VITA**, owing to it was was developed using MATLAB/Simulink® to simulate underwater vehicles designed by our team at ITA (Instituto Tecnológico de Aeronáutica, Brazil). The Simu2VITA software can be found at this repository[2], which includes an example of a simulation session and an animation produced by it.

The remaining sections of this article are organized as follows:

- Section 2 discuss other popular simulators and their main characteristics.
- Section 3 presents our simulator Simu2VITA, considerations taken in implementation and some possible extensions. Besides the presentation of the internal design functioning of Simu2VITA, this section also provides an overview on the modeling of a rigid-body vehicle and its actuators.
- Section 4 presents the simulation results for an UUV navigating inside a fully flooded tunnel and a comparison of these results with sensor data collected when the real vehicle performed the same mission, showing that Simu2VITA can be used for fast concept validation.
- Section 5 highlights the main points of the article and presents some possible improvements for this work.

2. Background

There are well established vehicle simulators already in use such as Gazebo [1] and Webots [2]. Gazebo is a general-purpose 3D simulator that can handle multiple robots and has an extensive library of ready-to-use vehicle models. Gazebo was originally built to satisfy the need for a high-fidelity vehicle simulator in outdoor environments. Being in development since early 2000's, the simulator now includes many features like over the network and cloud simulation. A simulated scenario configuration in Gazebo is done using SDF (Simulation Description Format) files [3], a markup language which was derived from URDF (Unified Robotic Description Format) [4] (SDF and URDF are XML formats). SDF allows the description of the vehicle (in terms of its joints) and its environment.

However, Gazebo was not designed to handle simulations including vehicles with rigid bodies moving through a dense fluid such as water. Taking advantage of the Gazebo plugin architecture, an extension to add fluid simulation named *Fluids* [5] was created. However, its own web page states that this plugin is experimental and outdated. There are also the Buoyancy Plugin[6] and the Lift-Drag Plugin[7] which make possible the creation of simplified underwater vehicle simulations but have complex parameters configurations such as defining the slope of the lift curve.

A possible alternative to add hydrodynamics and hydrostatics to Gazebo with lower complexity is to use the UUV Simulator[3] [8] which uses the modular design of Gazebo to enable simulation of multiple underwater vehicles. However, UUV Simulator does not implement fluid simualtion, instead it implements the extra forces caused by the presence of the fluid. Both Simu2VITA and the UUV Simulator use the same equations to simulate an underwater vehicle. Our simulator uses an similar approach building the simulation block on top of a more complete framework, in our case Simulink®. The difference is that our simulator does not require neither the edition of URDF files nor SDF files to describe the vehicle. Therefore we argue that it is easier to input the vehicle description in our simulator.

The Webots Open Source Robot Simulator [2] is a solution in many ways more suited for underwater simulation than Gazebo and UUV Simulator since it includes

---

[2] https://gitlab.com/aqualab/simu2vita
[3] https://uuvsimulator.github.io/

fluid simulation by design. It shares many similarities with Gazebo like multiple robot simulation, collision detection between bodies, headless simulation over network (when the visualization is not required or shown in a different machine and only the background computation of the simulation is performed in the simulator host machine) and ready-to-use models of sensors and robots. Webots also allows the addition of external forces to be added to the physics engine to create, for instance, a constant wind force affecting the vehicle. External communication with the simulator is possible using different approaches such as through a generic TCP/IP socket or using an API (Application Programming Interface) to an external application such as a program written in C/C++, java, python or MATLAB®.

In comparison to Webots, Simu2VITA can also accept inputs from outside the MATLAB/Simulink® framework using functionalities from MATLAB toolboxes such as the Instrument Control Toolbox or the Robotics System Toolbox. For someone used to using MATLAB/Simulink®, the learning curve to acquire the external signal input is very small. Simu2VITA lacks the visual aspect and the detailed physics descriptions of Webot but its simplicity to achieve good quality rigid body simulation its inherited communication functionalities from the MATLAB/Simulink® framework justify it as a good choice for an UUV simulator and its use for rapid controller design and testing.

3. The Simu2VITA Simulator

The Simu2VITA simulator implements the mathematical structure describing the laws of motion of an underwater vehicle. Such structure is composed of the actuator module, allocation module and the dynamics module of the vehicle. Compared with other solutions, the simulator Simu2VITA has the advantage of inheriting tool knowledge from the MATLAB/Simulink® framework, where one would only need to understand the concepts regarding the dynamics of the underwater vehicle.

It is worthy noting that our solution can be easily adapted to simulate other types of vehicles (e.g., ground and aerial vehicles) by changing the values of the dynamic model which is described ahead. This possibility will not be explored in this article. However, it will be explained in this article how to adapt Simu2VITA to simulate different types of underwater vehicles. The software usage can be found in Appendix A.

3.1. Simulator Description

Simu2VITA has three main modules describing different components of the vehicle. This modules are briefly described below and more details are given in the subsections ahead.

- The **Actuator Module** contains the actuator dynamics modeled each with a input signal saturation followed by a simple first order system. Inputs are handled by this module.
- The **Allocation Module** describes how the forces generated by the vehicle actuators are mapped into forces and torques acting on the body of the vehicle.
- The **Dynamics Module** has two main software components: the **kinematics component** that treats only geometrical aspects of the vehicle motion, and the **kinetics component** which deals with the effect of forces and torques applied to the body of the vehicle.

On Simu2VITA modeling is restricted to mechanical forces and torques acting in the vehicle and generated by its actuators. Therefore, an eventual electronic activation system of an actuator would have to be attached externally to the simulation block as shown in Figure 1. A typical case is the translation of a PWM input signal to the expected thrust input signal of a propeller.

At this point it is necessary to define the notation regarding vectors, matrices and linear transformations used herein. A vector $\mathbf{v}$ that is from some frame $\{U\}$ is shortly written as $\mathbf{v}_U$ or ${}^{U}\mathbf{v}_U$, and if the same vector has to be transformed yet to another frame

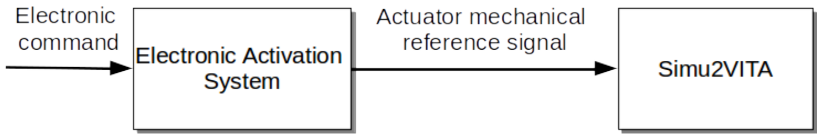

Figure 1. A simple schematic showing the logic to add some electronic activation dynamics of the actuators when using Simu2VITA.

$\{W\}$ then is denoted ${}^{W}\mathbf{v}_U$. A matrix P that represents a linear transformation from frame $\{U\}$ into frame $\{W\}$ is written as ${}^{W}P_U$. Therefore the expression linking $\mathbf{v}_U$ and ${}^{W}\mathbf{v}_U$ is

$$ {}^{W}\mathbf{v}_U = {}^{W}P_U \mathbf{v}_U \,, \tag{1} $$

and the transformation in opposite direction is given by:

$$ {}^{U}P_W = {}^{W}P_U^{-1} \,, \tag{2} $$

$$ \mathbf{v}_U = {}^{U}\mathbf{v}_U = {}^{U}P_W \, {}^{W}\mathbf{v}_U \,. \tag{3} $$

Figure 2 shows a three-dimensional frame attached to the body of some vehicle and the components regarding each axis of the body frame $\{b\}$. Observe the frame $\{b\}$ is defined according to the North-East-Down convention and centered at a chosen point $\mathbf{O}_b$ in the body called the body frame origin. The independent vectors forming frame $\{b\}$ are denominated:

- **n** for the forward pointing axis in red,
- **e** for the axis normal to the sagital plane of the vehicle in blue,
- and **d** for the axis pointing down in green.

Each axis of the body frame $\{b\}$ is named according to the nomenclature defined by the Society of Naval Architects and Marine Engineers - SNAME [9]. The vector **n** is named Surge Axis, **e** is the Sway Axis and **d** is the Heave Axis. The vector $\boldsymbol{v}_b = [u \; v \; w]^T$ represents the linear velocity of the vehicle written in respect to its own body frame $\{b\}$ and the components in each axis following the *Surge*, *Sway*, *Heave* order. The angular velocity is $\boldsymbol{\omega}_b = [p \; q \; r]^T$, with each component being the gyros around each axis. Both vectors can be put together in vector $\mathbf{v}_b = [\boldsymbol{v}_b^T \; \boldsymbol{\omega}_b^T]^T$. The forces and torques working on the vehicle body are all put in one single vector $\boldsymbol{\tau}_b = [X \; Y \; Z \; K \; M \; N]^T$, with X, Y and Z being the force components, and K, M and N the torque components.

Next the implementation of the three modules forming Simu2VITA are presented from a rear-to-front perspective. The Dynamics Module is presented in subsections 3.1.1 and 3.1.2. Subsection 3.1.3 explains how the Allocation Module transforms the forces generated by the actuators to forces and torques acting on the vehicle. Finally we show how an actuator is modeled and how the forces they generate are obtained in subsection 3.1.4 – the Actuator Module.

3.1.1. The Dynamics Module - Kinematics Component

Defining the global reference frame adopted by the simulator as the NED (North-East-Down) frame convention and calling it $\{w\}$, the simulated vehicle state is described as follows:

1. The pose of the vehicle written with respect to (w.r.t.) the $\{w\}$ frame,

$$ {}^{w}\boldsymbol{\eta}_b = [{}^{w}\mathbf{p}_b \; {}^{w}\mathbf{q}_b]^T \,, \tag{4} $$

where ${}^{w}\mathbf{p}_b$ is the position and ${}^{w}\mathbf{q}_b$ is a unit quaternion [10] describing the orientation of the vehicle with respect to $\{w\}$. Also ${}^{w}\mathbf{p}_b = [n \; e \; d]^T$, where n, e and d are the three euclidean components in the $\{w\}$ frame. The quaternion ${}^{w}\mathbf{q}_b = [q_0 \; \boldsymbol{\epsilon}]^T$ has its real part as its first component and the imaginary part encapsulated in $\boldsymbol{\epsilon}$.

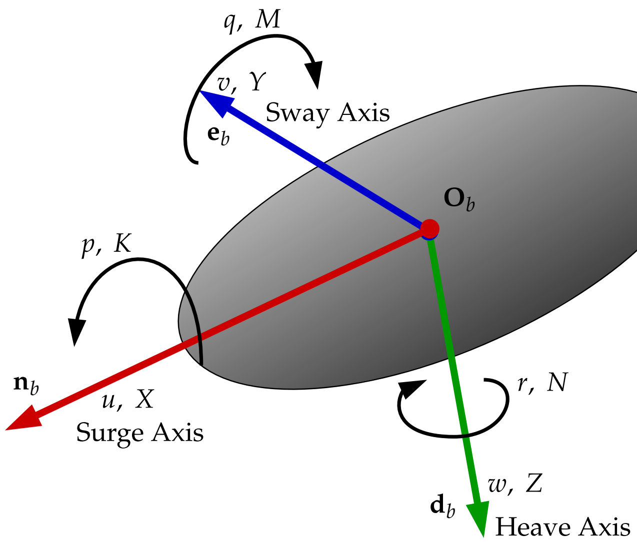

Figure 2. Definition of the body frame b of the vehicle. Note the components of $\mathbf{v}_b$ and $\boldsymbol{\tau}_b$ in each corresponding axis.

Notice that quaternion vector ${}^w\mathbf{q}_b$ can be interpreted as "orientation of frame $\{b\}$ in respect to frame $\{w\}$".

2. The linear and angular velocities w.r.t. the vehicle's own body frame

$$\mathbf{v}_b = [\boldsymbol{v}_b^T \; \boldsymbol{\omega}_b^T]^T \; . \tag{5}$$

The displacement of the vehicle w.r.t. $\{w\}$ is calculated using ${}^w\dot{\boldsymbol{\eta}}_b$ obtained from [11]

$${}^w\dot{\boldsymbol{\eta}}_b(\mathbf{v}_b, {}^w\boldsymbol{\eta}_b) = \left[{}^w\dot{\mathbf{p}}_b \;\; {}^w\dot{\mathbf{q}}_b\right]^T = \left[{}^w\mathbf{q}_b\boldsymbol{v}_b\,{}^w\mathbf{q}_b^* \;\; T_q({}^w\mathbf{q_b})\boldsymbol{\omega}_b\right]^T, \tag{6}$$

with ${}^w\mathbf{q}_b^*$ being the inverse of ${}^w\mathbf{q}_b$ [10] and $T_q(\mathbf{q})$ being a matrix with the form [11]

$$T_q(\mathbf{q}) = \frac{1}{2}\begin{bmatrix} -\boldsymbol{\epsilon}^T \\ q_0 I_{3\times 3} + S(\boldsymbol{\epsilon}) \end{bmatrix}, \tag{7}$$

where $S(\cdot)$ is the skew-symmetric matrix operator.

3.1.2. The Dynamics Module - Kinetics Component

The differential equation describing the behavior of the vehicle [11] is

$$M\dot{\mathbf{v}}_b + M_a\,{}^b\mathbf{v}_r + (C({}^b\mathbf{v}_r) + C_a({}^b\mathbf{v}_r))\,{}^b\mathbf{v}_r + D({}^b\mathbf{v}_r)\,{}^b\mathbf{v}_r + g({}^w\boldsymbol{\eta}_b) = \boldsymbol{\tau}_b \; , \tag{8}$$

already accounting for hydrodynamics and hydrostatic components, where

- $\dot{\mathbf{v}}_b$ is the acceleration vector of the vehicle.
- ${}^b\mathbf{v}_r$ is the relative velocity of the vehicle when accounting for constant water currents ${}^b\mathbf{v}_c$,

$${}^b\mathbf{v}_r = \mathbf{v}_b - {}^b\mathbf{v}_c \; , \tag{9}$$

with

$$ {}^{b}\mathbf{v}_c = \left[u_c \; v_c \; w_c \; 0 \; 0 \; 0\right]^T , \tag{10} $$

where u_c, v_c, w_c are respectively the components of the water current velocity in Surge, Sway and Heave.

- Matrix M is the rigid body Inertial Matrix and can be derived using Newton-Euler equations of motion. Here, M is defined using an arbitrary point $\mathbf{O_b}$ in the body of the vehicle as origin for frame $\{b\}$ and has the structure

$$ M = \begin{bmatrix} mI_{3\times 3} & -mS(\mathbf{r}_b) \\ mS(\mathbf{r}_b) & I_b \end{bmatrix} . \tag{11} $$

Vector $\mathbf{r}_b$ describes the displacement of the center of gravity of the vehicle w.r.t. $\{b\}$, and shall be informed when using the simulator. The scalar m is the mass of the vehicle. Matrix $I_b \in \mathbb{R}^{3\times 3}$ is the Inertia Matrix defined around the origin of $\{b\}$. One possibility to obtain the value of I_b is to first obtain the Inertia Matrix I_g around $\mathbf{r}_b$ and perform

$$ I_b = I_g - mS^2(\mathbf{r}_b) \; . \tag{12} $$

- C is the Coriolis–Centripetal Matrix, and the form used here can be found using Newton-Euler method,

$$ C = \begin{bmatrix} mS(\boldsymbol{\omega}_b) & -mS(\boldsymbol{\omega}_b)S(\mathbf{r}_b) \\ mS(\boldsymbol{\omega}_b)S(\mathbf{r}_b) & -S(I_b\boldsymbol{\omega}_b) \end{bmatrix} . \tag{13} $$

- M_a is the Added-Mass Matrix, that accounts for the extra inertia added to the system because of the water volume the accelerating vehicle must displace in order to move through it. This matrix is normally computed using an auxiliary numeric modeling software [12].
- C_a is the Hydrodynamic Coriolis–Centripetal Matrix and have the following form

$$ C_a = \begin{bmatrix} 0 & S(M_{a,11}\boldsymbol{v}_b + M_{a,12}\boldsymbol{\omega}_b) \\ S(M_{a,11}\boldsymbol{v}_b + M_{a,12}\boldsymbol{\omega}_b) & S(M_{a,21}\boldsymbol{v}_b + M_{a,22}\boldsymbol{\omega}_b) \end{bmatrix} . \tag{14} $$

- D is the Hydrodynamic Damping Matrix, which is simplified in our model. Here we assume the vehicle to perform relatively decoupled movements in each direction resulting in diagonal matrices for the linear and non-linear diagonal dumping.
- Vector $g({}^{w}\boldsymbol{\eta}_b)$ account for the static and hydrostatic forces acting on fully submerged vehicles, meaning gravitational force ${}^{w}\mathbf{f}_W = [0 \; 0 \; W]^T$ and buoyancy force ${}^{w}\mathbf{f}_B = -[0 \; 0 \; B]^T$, with $W = mg$ and $B = \rho g \nabla$. Scalar g is gravity acceleration, ρ is the water density and ∇ the volume displaced by the vehicle. Finally

$$ {}^{b}\mathbf{f}_W = {}^{w}\mathbf{q}_b^{-1} \, {}^{w}\mathbf{f}_W ({}^{w}\mathbf{q}_b^{-1})^* , \tag{15} $$

$$ {}^{b}\mathbf{f}_B = {}^{w}\mathbf{q}_b^{-1} \, {}^{w}\mathbf{f}_B ({}^{w}\mathbf{q}_b^{-1})^* , \tag{16} $$

$$ g({}^{w}\boldsymbol{\eta}_b) = -\begin{bmatrix} {}^{b}\mathbf{f}_W + {}^{b}\mathbf{f}_B \\ \mathbf{r}_b \times {}^{b}\mathbf{f}_W + \mathbf{b}_b \times {}^{b}\mathbf{f}_B \end{bmatrix} , \tag{17} $$

and observe that $\mathbf{b}_b$ is the center of buoyancy in the body of the vehicle.

- $\boldsymbol{\tau}_b$ is the vector of disturbing forces and torques applied to the vehicle in each axis of the body frame, including those generated by the actuators. We divide this vector into two main components as described in eq. (18)

$$ \boldsymbol{\tau}_b = \left[X \; Y \; Z \; K \; M \; N\right]^T = {}^{b}\boldsymbol{\tau}_a + {}^{b}\boldsymbol{\tau}_e , \tag{18} $$

where X, Y and Z are forces applied into the Surge, Sway and Heave Axis respectively. Torques are K, M and N following roll, pitch and yaw movements respectively. See Figure 2. With ${}^{b}\boldsymbol{\tau}_e$ encapsulating any external forces and torques from any source and ${}^{b}\boldsymbol{\tau}_a$ coming from the actuators.

Internally, we compute the acceleration of the vehicle by simply isolating $\dot{\mathbf{v}}$ in eq. (8) transforming it into

$$\dot{\mathbf{v}}_b = (M + M_a)^{-1}(\boldsymbol{\tau}_b + M_a\,{}^{b}\dot{\mathbf{v}}_c - C(\mathbf{v}_b)\mathbf{v}_b - C_a({}^{b}\mathbf{v}_b)\,{}^{b}\mathbf{v}_r - D({}^{b}\mathbf{v}_r)\,{}^{b}\mathbf{v}_r - g({}^{w}\boldsymbol{\eta}_b))\,, \quad (19)$$

considering ${}^{b}\dot{\mathbf{v}}_c$ to be

$${}^{b}\dot{\mathbf{v}}_c = \begin{bmatrix} S(\boldsymbol{\omega}_b) & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} \end{bmatrix} {}^{b}\mathbf{v}_c\,, \quad (20)$$

implicitly assuming the water current to be constant and irrotational [11]. Figure 3 shows the internal flow of information, input and output o this module. Observe that here we also present the initial state vectors ${}^{w}\boldsymbol{\eta}_{b,0}$ and ${}^{b}\mathbf{v}_{b,0}$ as inputs to the Kinematics part.

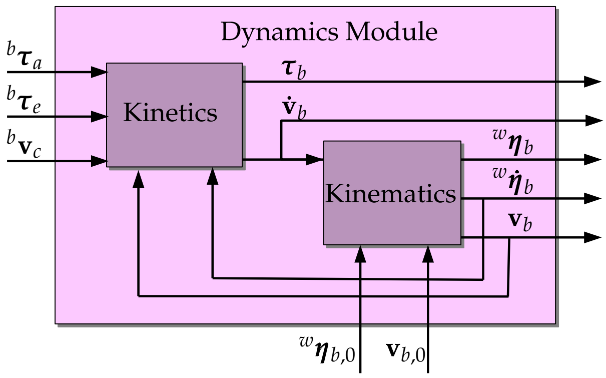

Figure 3. Logic representation of both Kinetics and Kinematics inside the Dynamics Model. Inputs and outputs are also represented.

3.1.3. The Allocation Module

The Allocation Module task is to transform the output of the modeled actuators $\mathbf{y}$ into forces and torques inputs of the vehicle described by ${}^{b}\boldsymbol{\tau}_a$, i.e., a function $\boldsymbol{f} : \mathbb{R}^n \rightarrow \mathbb{R}^6$ with $n \geq 0$ being the number of actuators contributing to the generation of forces and torques in all six degrees of freedom. Commonly this transformation is linear, and so it is in our design. This linear transformation is firstly considered static, and later a time-variant possible solution is shown. Eq. (21) show the static case transformation,

$${}^{b}\boldsymbol{\tau}_a = [{}^{b}X_a\ {}^{b}Y_a\ {}^{b}Z_a\ {}^{b}K_a\ {}^{b}M_a\ {}^{b}N_a]^T = H\mathbf{y}_a\,, \quad (21)$$

with $\mathbf{y}_a$ being the vector containing the output of the actuators written in respect to these and matrix H is the allocation matrix, accounting for the contribution of each actuator in forces and torques acting in each axis of the vehicle. The computation of this matrix can be made pragmatically for the case where the actuators are propellers attached to the body of the vehicle. First we consider the position of this actuators w.r.t $\{b\}$ frame and their orientations using Euler angles. We denote the position of the k-th propeller in this case as ${}^{b}\mathbf{p}_{a,k} = [{}^{b}n_{a,k}\ {}^{b}e_{a,k}\ {}^{b}d_{a,k}]^T$ and its orientation as ${}^{b}\boldsymbol{\alpha}_{a,k} = [{}^{b}\phi_{a,k}\ {}^{b}\theta_{a,k}\ {}^{b}\psi_{a,k}]^T$ representing roll, pitch and yaw components. Now assuming the propeller pushes the vehicle only in its $\mathbf{n}_{a,k}$ axis direction as in Figure 4, we compute ${}^{b}\mathbf{n}_{a,k}$ as the resultant first column vector from the rotation matrix ${}^{b}R_{a,k}$ describing the misalignment of the actuator frame with respect to the body frame of the vehicle

$$ {}^{b}R_{a,k} \;=\; [{}^{b}\mathbf{n}_{a,k}\;{}^{b}\mathbf{e}_{a,k}\;{}^{b}\mathbf{d}_{a,k}] \;=\; R({}^{b}\phi_{a,k})R({}^{b}\theta_{a,k})R({}^{b}\psi_{a,k})\,. \tag{22} $$

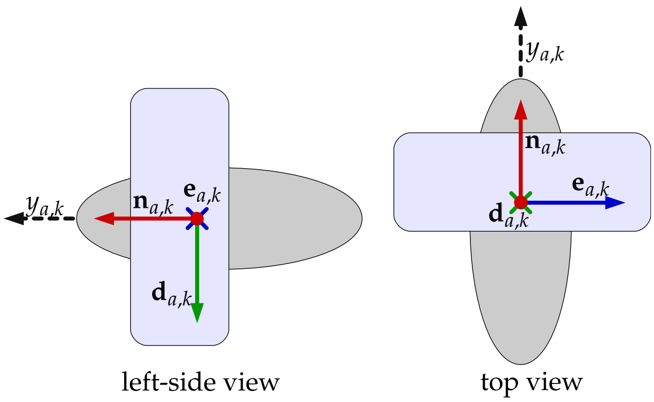

Figure 4. This image shows the direction of the force generated by a propeller. The left view shows a side way view from the left of the propeller, the right view gives the top view. Observe the output force vector $y_{a,k}$ is always aligned with the $\mathbf{n}_{a,k}$ axis.

We then change the name of vector ${}^{b}\mathbf{n}_{a,k}$ to express the distribution of the force $y_{a,k}$ generated by the k-th propeller in each axis of $\{b\}$.

$$ {}^{b}\mathbf{f}_{a,k} \;=\; [{}^{b}f_{X,k}\;{}^{b}f_{Y,k}\;{}^{b}f_{Z,k}]^{T} \;=\; {}^{b}\mathbf{n}_{a,k}^{T}\,. \tag{23} $$

This way, the resultant force of the k-th propeller in each axis is given by

$$ {}^{b}y_{a,k} = \begin{bmatrix} {}^{b}X_{a,k} \\ {}^{b}Y_{a,k} \\ {}^{b}Z_{a,k} \end{bmatrix} = {}^{b}\mathbf{f}_{a,k}y_{a,k}\,. \tag{24} $$

The torque generated by the k-th propeller in the body is calculated using the cross product of ${}^{b}\mathbf{p}_{a,k}$ by $y_{a,k}$ resulting in

$$ {}^{b}\mathbf{M}_{a,k} = \begin{bmatrix} {}^{b}K_{a,k} \\ {}^{b}M_{a,k} \\ {}^{b}N_{a,k} \end{bmatrix} = \underbrace{[{}^{b}m_{K,k}\;{}^{b}m_{M,k}\;{}^{b}m_{N,k}]^{T}}_{{}^{b}\mathbf{m}_{a,k}} y_{a,k} = {}^{b}\mathbf{p}_{a,k} \times {}^{b}\mathbf{f}_{a,k}y_{a,k}\,. \tag{25} $$

Figure 5 shows the geometric relation of ${}^{b}\mathbf{p}_{a,k}$ and ${}^{b}\mathbf{n}_{a,k}$. It is now clear that the full allocation vector is ${}^{b}\boldsymbol{h}_{a,k} = [{}^{b}\mathbf{f}_{a,k}^{T}\;{}^{b}\mathbf{m}_{a,k}^{T}]^{T}$, and we can align all allocation vectors in the matrix

$$ H_{6\times n} \;=\; [{}^{b}\boldsymbol{h}_{a,1}\;{}^{b}\boldsymbol{h}_{a,2}\;{}^{b}\boldsymbol{h}_{a,3}\;...\;{}^{b}\boldsymbol{h}_{a,n}]\,, \tag{26} $$

with n being the total number of propellers, we obtain the allocation matrix. Now multiplying H by a column vector $\mathbf{y}$ containing the forces coming from the propellers, the resultant forces and torques vector ${}^{b}\boldsymbol{\tau}_{a}$ is generated and shown in eq. (21). Figure 6 shows a block diagram of this transformation.

Observe H can be time-dependent if the vehicle has movable actuators, for instance, a rotating propeller or even a fin for roll and pitch maneuvers. These rotating and movable actuators can be also modeled in the actuator module as will be show in Subsection 3.1.4, but H will need to be calculated outside the simulator and this output fed back into Simu2VITA. For the simple case of a rotating propeller the procedure we presented is the basis, with just the constant changing orientation needing to be tracked,

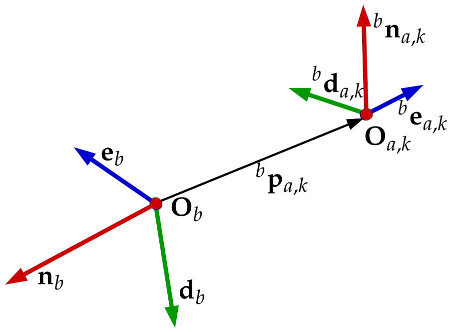

Figure 5. The representation of the frame of any k-th actuator w.r.t. the body frame $\{b\}$ of the vehicle.

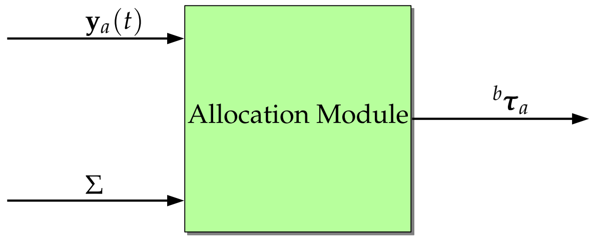

Figure 6. A graphic representation of the operation performed by the Allocation Module, with $\mathbf{y}_a$ and H as inputs and $^b\boldsymbol{\tau}_a$ as output.

see Figure 7. For fins, perhaps a non-linear approach is needed and the final $^b\boldsymbol{\tau}_a$ must be fed back directly using the $^b\boldsymbol{\tau}_e$ input of the Dynamics Module as the Allocation Module internal machinery expects a matrix to perform a linear transformation, in this case $H = 0$, i.e., the Allocation Module is bypassed. A future refining is to turn needless this bypass for the non-linear case of force allocation.

3.1.4. The Actuator Module

The input of Simu2VITA represents the reference signal the actuators of the vehicle should follow. For instance if the actuator is a propeller, the input reference signal should be the desired force to be generated by the actuator. In the case of a fin, the reference signal should would be the desired fin angle. These input signals are handled by the Actuator Module. Each actuator is modeled as a saturation function followed by a first order linear system with a user-defined time constant T (transfer function $G(s) = 1/(Ts+1)$). Therefore each actuator output $y_a(t)$ can be computed in time in closed form like

$$y_a(t) = \exp\left[-\frac{(t-t_0)}{T}\right] y_a(t_0) + \frac{1}{T}\int_{t_0}^{t} \exp\left[-\frac{(t-\tau)}{T}\right] \bar{u}_a(\tau) d\tau\,, \tag{27}$$

where t_0 is the initial simulation time, $y_a(t)$ is the output at time t, $y_a(t_0)$ is the initial state and $\bar{u}_a(t)$ is the limited input signal received by the actuator. This $\bar{u}_a(t)$ is defined as

$$\bar{u}_a(t) = sat(u_a(t), u_{min}, u_{max}) = \begin{cases} u_{min} & \text{if } u_a(t) < u_{min} \\ u_a(t) & \text{if } u_{min} \le u_a(t) \le u_{max} \\ u_{max} & \text{if } u_{max} < u_a(t) \end{cases}, \tag{28}$$

with u_{min} and u_{max} being respectively the lower and upper limit values for the actuator input signal $u_a(t)$. Note that the actuator output $y_a(t)$ is also bounded by u_{min} and u_{max}.

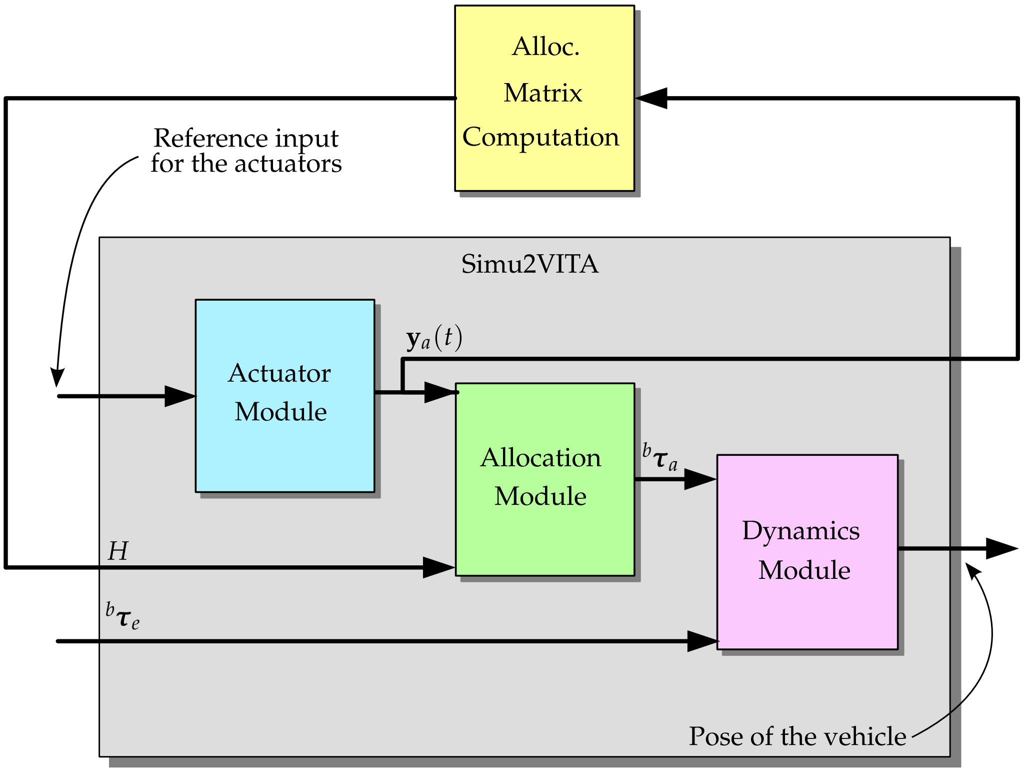

Figure 7. The logic representation of an external calculator for the allocation matrix in case of a moving propeller.

Since a vehicle usually has multiple actuators, we need to define some useful vectors to express the whole system in a compact form

$$\begin{aligned}
\mathbf{y}_a(t_0) &= [y_{a,1}(t_0) \dots y_{a,n}(t_0)]^T, && (29)\\
\mathbf{T} &= [T_1 \dots T_n]^T, && (30)\\
\mathbf{u_a}(t) &= [u_{a,1}(t) \dots u_{a,n}(t)]^T, && (31)\\
\mathbf{u_{min}} &= [u_{min,1} \dots u_{min,n}]^T, && (32)\\
\mathbf{u_{max}} &= [u_{max,1} \dots u_{max,n}]^T, && (33)\\
\mathbf{\bar{u}_a(t)} &= sat(\mathbf{u_a}(t), \mathbf{u_{min}}, \mathbf{u_{max}}), && (34)
\end{aligned}$$

where for all actuators $\mathbf{y}_a(t_0)$ is the initial output vector, $\mathbf{T}$ gathers the time constants, $\mathbf{u}_a(t)$ contains the input signals, $\mathbf{u_{min}}$ and $\mathbf{u_{max}}$ contains the input lower and upper limits respectively.

The actuator output vector $\mathbf{y}_a(t)$ is then computed using:

$$\begin{aligned}
\mathbf{y}_a(t) &= \begin{bmatrix} y_{a,1}(t) \\ \vdots \\ y_{a,n}(t) \end{bmatrix} \\
&= \begin{bmatrix} \exp^{[-T_1^{-1}(t-t_0)]} y_{a,1}(t_0) + T_1^{-1} \int_{t_0}^{t} \exp^{[-T_1^{-1}(t-\tau)]} \bar{u}_{a,1}(\tau) d\tau \\ \vdots \\ \exp^{[-T_n^{-1}(t-t_0)]} y_{a,n}(t_0) + T_n^{-1} \int_{t_0}^{t} \exp^{[-T_n^{-1}(t-\tau)]} \bar{u}_{a,n}(\tau) d\tau \end{bmatrix}.
\end{aligned} \quad (35)$$

Figure 8 represents graphically the Actuator Module as a block. Figure 9 shows the connection of all modules as a whole greater block, Simu2VITA. This can serve as initial point to visualize possible ways to adapt it to other types of marine-crafts other than underwater vehicles.

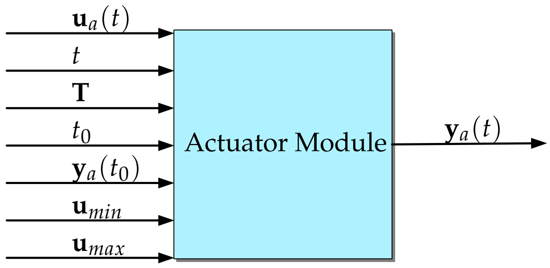

Figure 8. The Actuator Module as a block. Observe this Module also outputs the state of the actuator before it passes through the saturation.

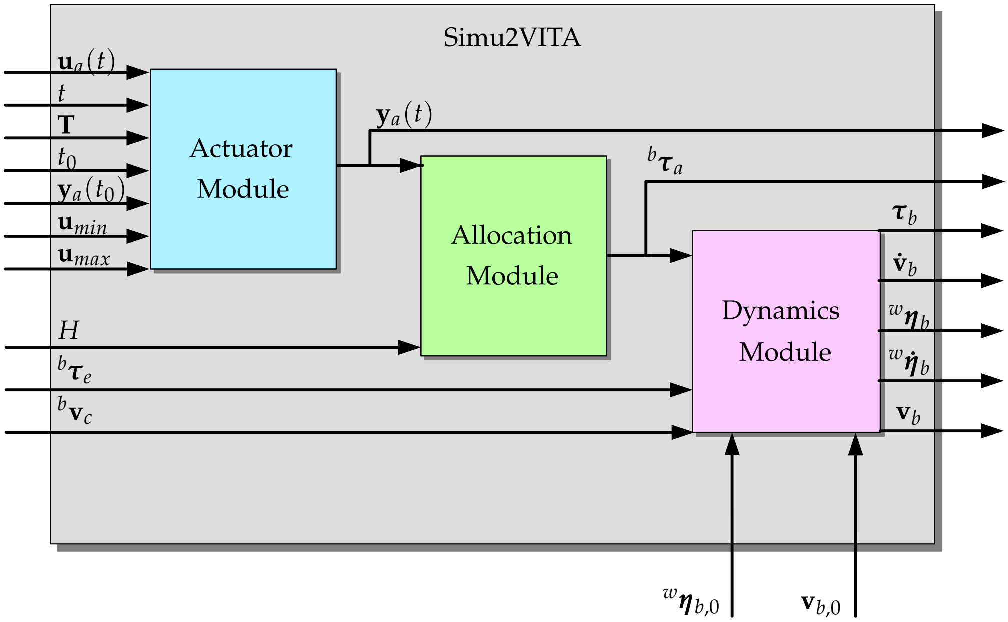

Figure 9. Logic connection of all three modules and their input and output signals.

4. Experiments

In this section an experiment is presented to demonstrate the flexibility of usage of Simu2VITA. The simulated results are then compared with telemetry data captured when a real UUV was deployed *in loco*. We simulate the UUV named VITA1[13], shown in Figure 10, which is a modified version of the BlueROV2 sold by Blue Robotics [14]. VITA1 has eight fixed propellers acting as actuators and the following sensors:

- a set of four echosounders from Bluerobotics pointing outwards the vehicle[15],
- an imaging sonar, model Tritech Gemini 720im[16],
- a profiling sonar, model Imagenex 881L[17], and
- a high definition (1080p, 30fps) wide-angle low-light camera[18] equipped with four small lights.

We use the Simulink 3D Animation toolbox for visualization of the dynamics of the vehicle. This visualization shows the vehicle pose over time as a 3D animation, see Figure 11. The echosounders are simulated as lines going out from them. The distance between an echosounder and an object is obtained when its line intersects the object. This intersection detection is made automatically by the Simulink 3D Animation toolbox.

4.1. Simulated Experiment

In the simulated experiment presented in this section the vehicle navigates inside a fully flooded underwater straight tunnel. The vehicle should move with a constant desired forward speed, in the center of the cross section of the tunnel and oriented as the tunnel main axis. The tunnel itself is oriented in the same direction as the **n** of the

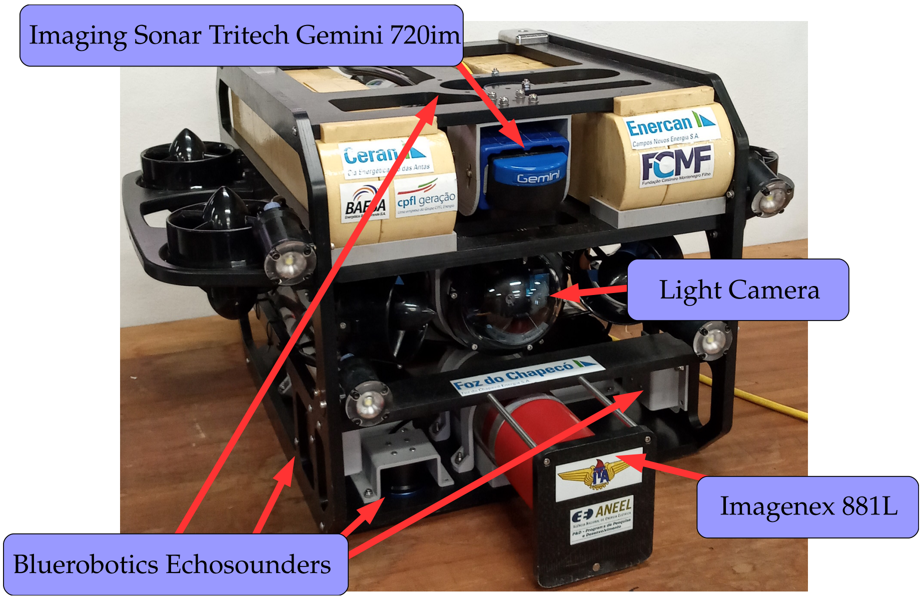

Figure 10. The vehicle VITA1 and its sensors.

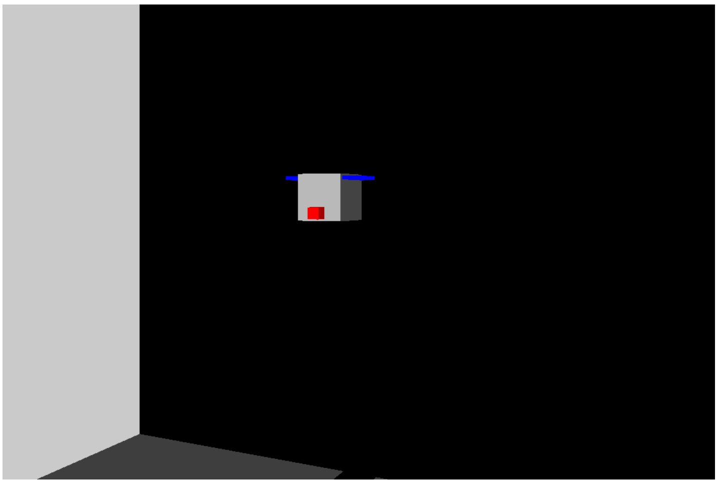

Figure 11. Visualization of the 3D model of the vehicle and the scenario. The red dot in front of the simulated vehicle is an allusion to the red of the Imagenex Profiling Sonar 881L. The blue lines on both sides are the representation of the "wings" carrying the propellers on VITA1. The side wall and floor of the simulated tunnel can be seen in gray and dark gray, respectively, on the left.

frame $\{w\}$. To achieve these goals, two additional systems were attached to Simu2VITA: a Guidance System and a Control System. The Guidance System continuously updates the desired path the vehicle should follow. The Control System generates the command signals for the vehicle actuators such that it follows the desired path generated by the Guidance System as close as possible. The general picture of the problem can be seen in Figure 12, with the four echosounders readings (d_1 to d_4) shown as blue and green arrows and the red arrow point forward indicating the direction of the desired forward speed.

The Guidance System receives the desired values for the vehicle forward speed u_d, the desired vehicle orientation ${}^{w}\mathbf{q}_{b,d}$, the desired offsets between lateral echosounder readings ${}^{b}\mathbf{e}_{sw,d}$ and vertical echosounder readings ${}^{b}\mathbf{e}_{he,d}$. The lateral and vertical distances of the vehicle to the center of the tunnel cross section are computed using ${}^{b}\mathbf{e}_{sw} = d_2 - d_1$ and ${}^{b}\mathbf{e}_{he} = d_3 - d_4$. These desired values are then smoothly interpolated with the current state of the vehicle and sensor readings generating a smooth path to be followed. The signal outputs of the Guidance System are used as reference

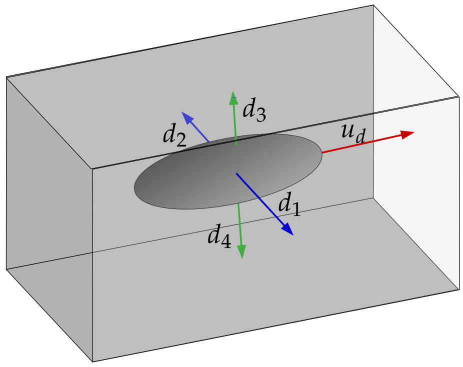

Figure 12. Distances measured by the VITA1 echosounders.

values when entering the Control System. Note that these references are smooth paths meaning that for the case of speed there is an acceleration reference too, and for the case of offsets and orientation that are constraints on speed and acceleration.

The Control System is responsible for generating command signals to the vehicle actuators to move the vehicle. The Control System is composed of four distinct controllers: a forward speed controller, a centralization controller, an orientation controller and a stabilizer controller.

To reach the reference velocity u_{ref} coming from the Guidance System, the forward speed is implemented as a PI controller with a feedforward reference acceleration term $\dot{u}_{ref}$ is used. The idea is that once the error between the measured forward speed of the vehicle and the reference speed approach zero only the reference acceleration input remains. For a constant desired forward speed the final reference acceleration value will be zero.

The centralization controller is responsible for positioning the vehicle in center of the tunnel cross-section. It is implemented using two separated PID controllers for both lateral and vertical position correction.

The orientation controller is a nonlinear controller that uses quaternion directly based on the work of Fresk and Nikolakopoulos [19].

Finally the stabilizer controller compensates the nonlinear parts of the model using a state feedback linearization approach. More details about the derivation and implementation of the Guidance and Control Systems are given by de Cerqueira Gava *et al.* [20].

The forward speed and the centralization controllers generate force commands. The orientation controller generate torque commands. To transform forces and torques into actuator inputs (propellers in this case), the simplest form were used. From eq. (21) we use the pseudo-inverse of H to obtain the actuators input

$$\mathbf{u}_a = \underbrace{H^T(HH^T)^{-1}}_{H^\dagger}\, {}^b\boldsymbol{\tau}_c \tag{36}$$

with ${}^b\boldsymbol{\tau}_c$ being the output of the Control System of forces and torques. Figure 13 shows how the Guidance and Control Systems are connected to Sim2VITA and their respective input and output signals.

The simulation was performed using the variable step size ODE solver ode45 with step size of 0.001 s, in MATLAB R2019a. The Guidance System runs at 20 Hz as well as the Control System controllers but the stabilizer controller running at 400 Hz. We opted to put this high control rate to resemble the hardware we have in the real vehicle, a PixHawk micro-controller board [21] running the ArduSub software [22]. The PixHawk runs its internal stabilizer controller at 400 Hz.

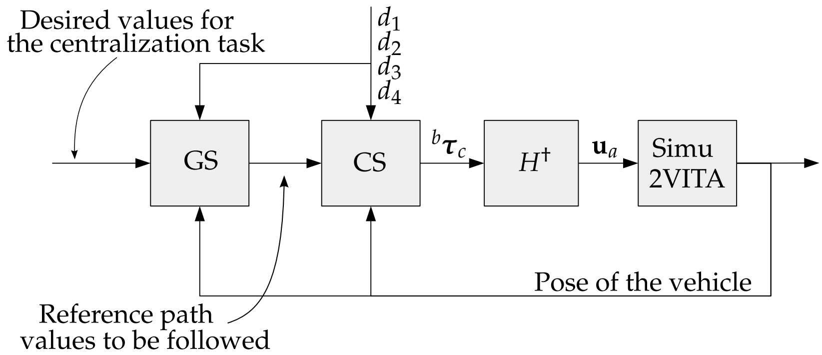

Figure 13. Diagram of the connection of the Guidance System (GS), Control System (CS) and Simu2VITA.

Setting the tunnel to begin at the origin of the inertial system $\{w\}$ alongside the direction of $\mathbf{n}$, the simulated tunnel has a square profile with each side measuring 8 meters. The vehicle initial state, as explained in Subsection 3.1.2, is

$$ {}^{w}\boldsymbol{\eta}_{b,0} = \begin{bmatrix} 3 \\ 1 \\ -1 \\ 0.9764 \\ -0.0199 \\ 0.1776 \\ 0.1209 \end{bmatrix}, \tag{37} $$

$$ {}^{w}\mathbf{v}_{b,0} = \mathbf{0}_{6\times 1}, \tag{38} $$

with the quaternion part being equivalent to an orientation of $-5°$ in roll, $20°$ in pitch and $15°$ in yaw. The desired final surge velocity u_d is 0.2 m/s. Desired ${}^{b}\mathbf{e}_{sw}$ and ${}^{b}\mathbf{e}_{he}$ are zero. The centralization task may be seen from the signals of the simulated echosounders in Figure 14. Observe the lateral and vertical echosounders readings converging to the same value (3.70 m), leading to errors ${}^{b}\mathbf{e}_{sw}$ and ${}^{b}\mathbf{e}_{he}$ to zero. The lateral and vertical echosounders readings converge to 3.70 m since the simulated tunnel has a square cross-section with 8 m side length the vehicle shape is a cube with 0.6 m side length and the echosounders are assumed to placed at the vehicle surfaces, not at its center.

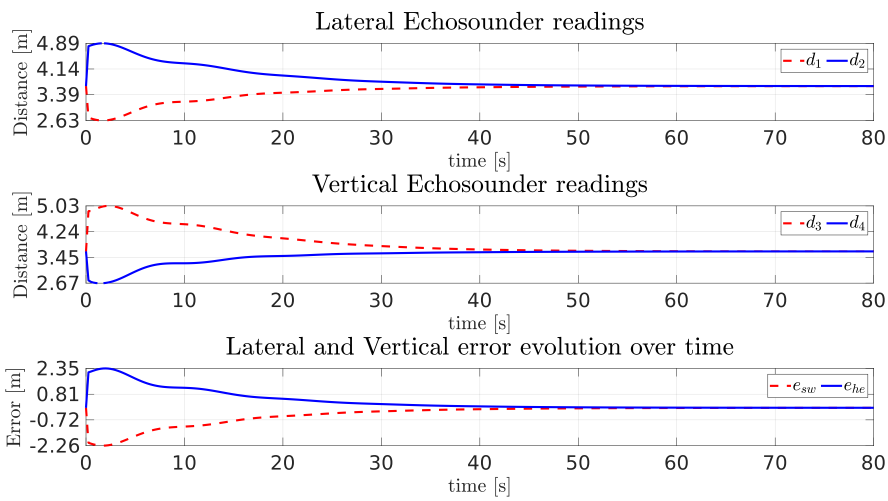

Figure 14. Readings of echosounders of the simulated vehicle and the vertical and horizontal errors over time.

Considering regulation of vehicle orientation, the dynamics of vehicle are stable for the roll and pitch axes, so these angles naturally converge to zero. However, the yaw angle must be actively controlled in order to follow the referencing signal. In this case, as the tunnel sagittal plane is oriented orthogonal to the coronal plane of the world frame $\{w\}$ (**ed**-plane) and the vehicle must cruise the tunnel with ${}^{w}\mathbf{n}_b$ parallel to the walls, the desired final yaw value should be zero. Figures 15 and 16 show the evolution of the angular velocities in gyros and orientation angles, respectively, smoothly converging to zero.

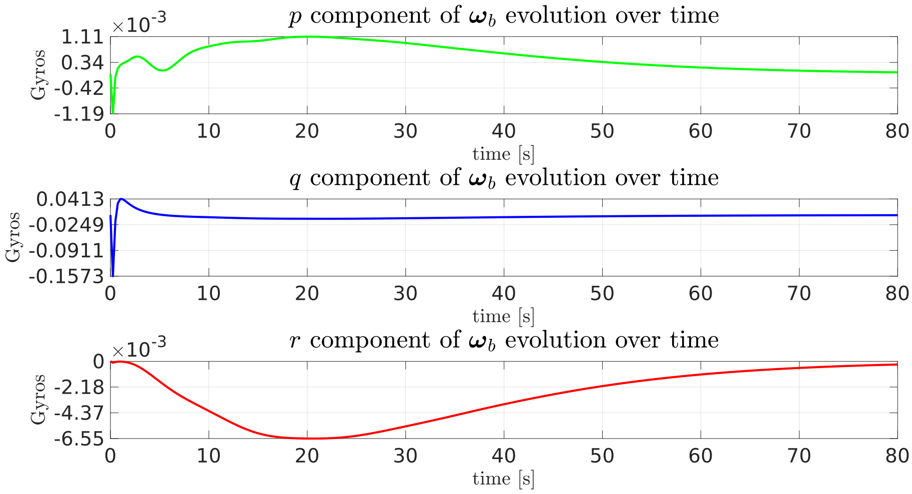

Figure 15. The evolution of angular speed components of the simulated vehicle over time.

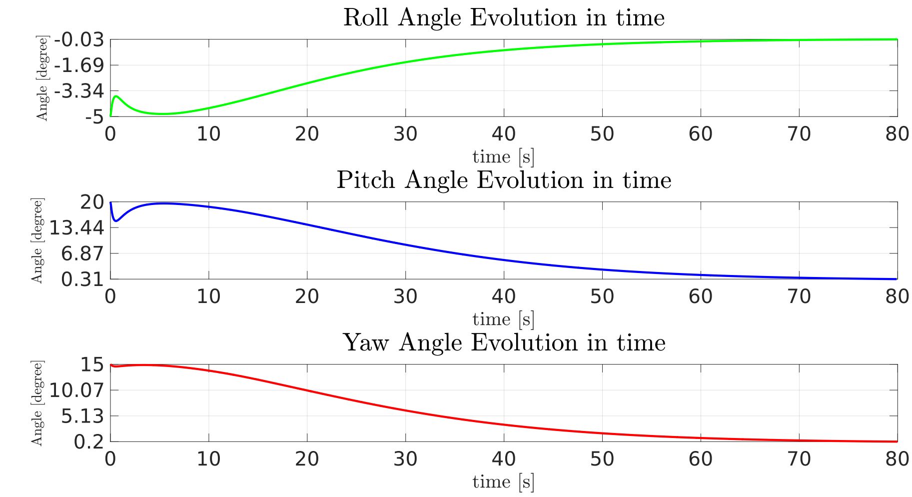

Figure 16. The evolution of orientation components of the simulated vehicle over time.

As the vehicle started the simulation displaced by a meter up and to the right, and rotated, is expected to exist considerable horizontal and vertical velocities. Figure 17 shows the simulated vehicle velocity vetor evolution in the 3 axis as depicted in Figure 2. Observe how the desired forward velocity $u_d = 0.2$ m/s is achieved, while the vehicle centralizes itself. In this case v and w velocities evolution present similar profile.

The evolution of components of the position ${}^{w}\mathbf{p}_b$ of the vehicle can be seen in Figure 18. As expected the n component has grown as the time passed, and both e and d components converged to zero as was previously show in Figure 14. This happens because the center of the tunnel profile occurs at the origin of the coronal plane.

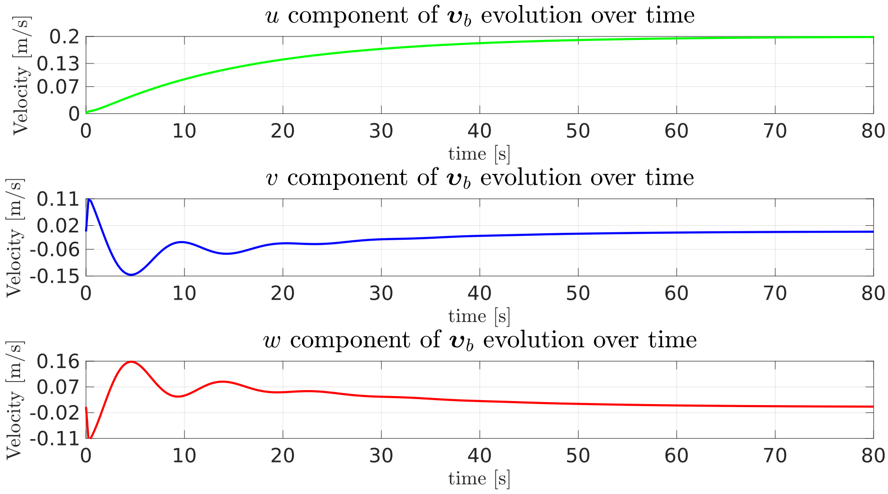

Figure 17. The evolution of the components of the simulated vehicle linear velocity over time.

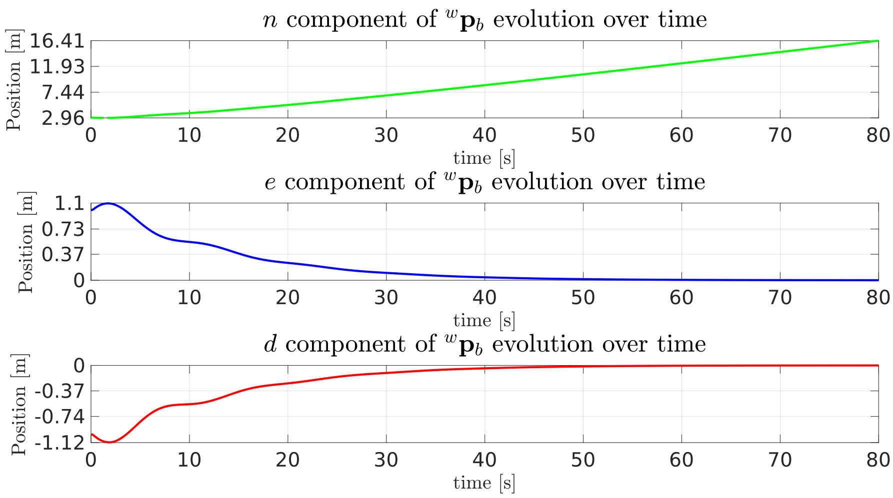

Figure 18. The evolution of the position components of the simulated vehicle over time.

The simulated propellers were eight, all having the same lower and superior limits 39.91 N and 51.48 N respectively. These values are informed by the manufacturer of the real propeller [23] used in the real vehicle for the specified tension of 16 V. For the time constant we have used 0.1754 s, a value also used by Manhães *et al.*[8]. Figure 19 shows the evolution of a propeller over time, with the lower graph depicting the transitory response for a series of changing values of input.

4.2. Real Experiment

For the real experiment the VITA1 vehicle was placed inside a hydro-power plant adduction tunnel which is 100 m long and 3.80 m wide. The vehicle was attached to a topside station through a tether cable, with the Guidance and Control Systems executing at the station. The only controller executing embedded of the vehicle was the stabilizer controller running in the PixHawk board. The control rate of the systems previous mentioned are the same as those in the simulated experiment. For a detailed explanation of the functioning of VITA1, please refer to the work of Jorge *et al.*[24].

In this experiment, the main differences in relation to the simulated experiment are:

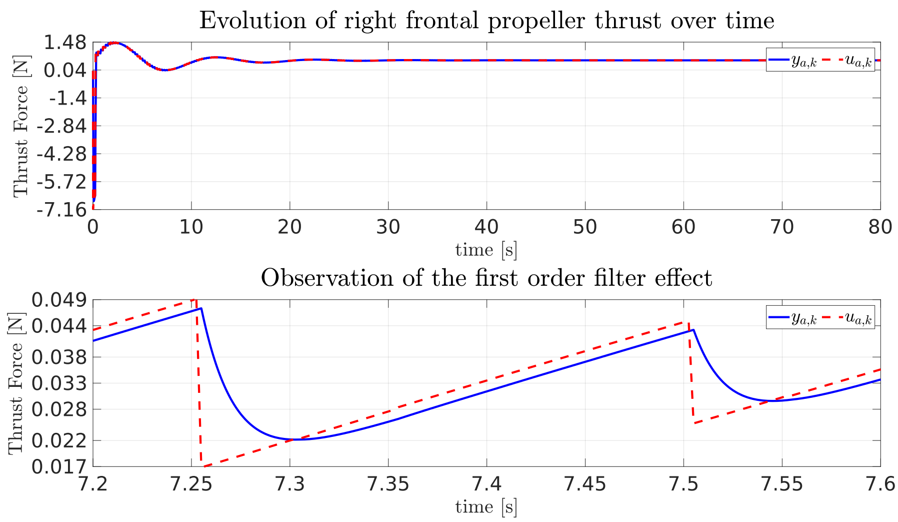

Figure 19. The evolution of the position components of the simulated vehicle over time.

- instead of going straight across the tunnel, the vehicle vertical desired path ${}^{b}\mathbf{e}_{he}$ is sinusoidal,
- the controller compensating nonlinear terms is a cascade PID running at 400 Hz on the micro-controller PixHawk [21] using readings from its own internal accelerometers and gyrometers.
- The orientation controller operates separately in each orientation degree of freedom using also cascade PID inside PixHawk while the simulated vehicle used a composed orientation controller in quaternion form.

The forward speed and the centralization controller remains with the same structure, but now they generate input for the internal controllers of the PixHawk. The state observation algorithm used it the one presented by Pittelkau[25] and embedded in the PixHawk. The vehicle departs from the entrance of the tunnel almost pointing to the desired yaw orientation of $-137°$ and almost centralized.

Figure 20 exhibits the evolution of the orientation overtime, where roll and pitch remain in a well bounded box around zero, also the yaw track the desired yaw angle and remains around it.

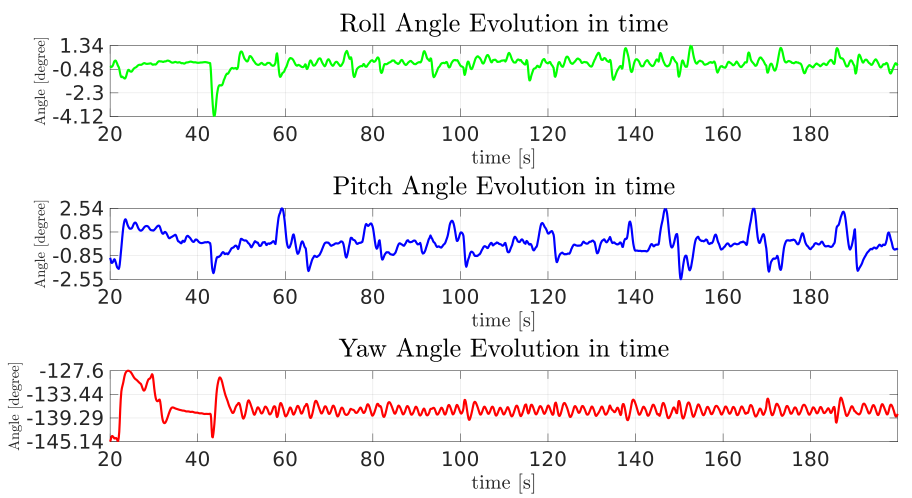

Figure 20. The evolution of orientation components of VITA1 over time.

The echosounders signals in Figure 21 show that in the horizontal movement the bouncing converges to a oscillatory pattern near zero but around 0.25 meters, while the sinusoidal pattern is quite evident.

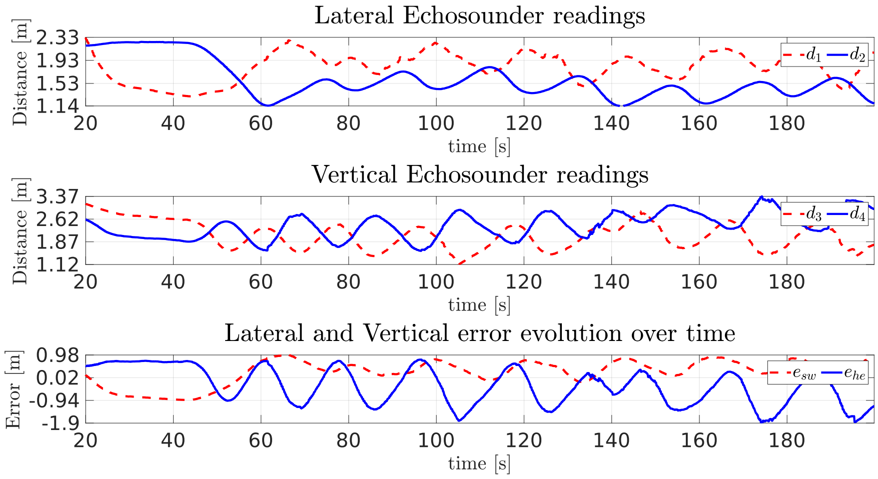

Figure 21. Readings of echosounders of VITA1 and the vertical and horizontal errors over time.

Last, observer the constant surge velocity around the desired surge velocity at 2.5 m/s in Figure 22. Also, as expected from Figure 21 there were expressive velocities in both horizontal and vertical directions of the vehicle. The velocities were measured using the DVL sensor A50 from Waterlinked [26] attached to the bottom of VITA1 pointing downwards.

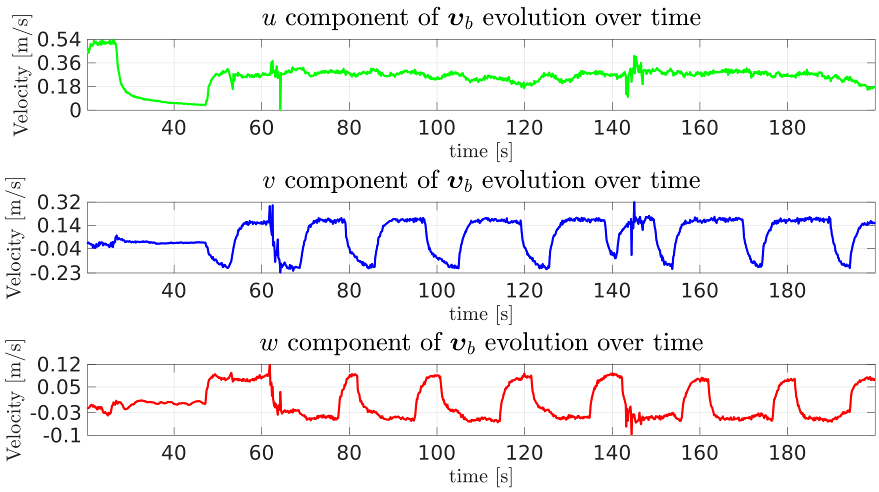

Figure 22. The evolution of linear velocities components of VITA1 over time.

From the previous experiments we have shown that is possible to use Simu2VITA to model and test controllers and behaviors for some desired vehicle, even for the case the simulated experimented used perfect sensing while the real case used an Extended Kalman Filter to estimate its states. Also, the assumption of slow decoupled movements held, as a high-rate PID was able to "linearize" the dynamics of the vehicle.

5. Conclusion

This article reports the development and some use cases of Simu2VITA, a simulator designed for UUV simulation. Simu2VITA is easy to setup and facilitates rapid prototyping and validation of concepts. Our simulator has been demonstrated to be a simple and resourceful tool when designing controllers and autonomous behaviors for an UUV. The simplicity of Simu2VITA and its easiness of use allowed our research project team to become familiar with the behavior of the real underwater vehicle before any *in loco* experiment.

We plan to implement simulated versions of some types of sonar sensors. There are already some models for a complex sensor like the sonar as the one proposed by Mai *et al.*[27]. Another possible future work is to provide an animation model for the 3D animation system of Simulink®. Previous prepared controllers and behavior algorithms are in the sight of this research too, to enable rapid prototyping of autonomous underwater vehicles. In comparison to other simulators, Simu2VITA still lacks collision detection but this can be implemented outside of the simulator and is a possible future improvement to be made. Another future addition to Simu2VITA is the possibility to choose more complex dynamic models for the actuators.

Funding: The work reported in this paper was performed as part of an interdisciplinary research and development project undertaken by Instituto Tecnológico de Aeronáutica (ITA). The authors acknowledge the financial funding and support of the following companies: CERAN, ENERCAN and FOZ DO CHAPECÓ, under supervision of ANEEL - The Brazilian Regulatory Agency of Electricity. Project number PD 02476-2502/2017.

Acknowledgments: The authors wish to thank Waldir Vieira, Thaís Machado Mancilha and Luiz Eugênio Santos Araújo Filho for their support when retrieving some parameters of the underwater vehicle VITA1.

Appendix A Simu2VITA block on SIMULINK

Simu2VITA is a piece of software built on top of Matlab and Simulink machinery. It is a self-contained block. Figure A1 shows the block as it is on Simulink with its inputs and outputs. Almost all inputs and outputs in Figure A1 have a correspondence to some previously mentioned variable in Section 3, except inputs

- *init_actuator_time*, and
- *simulation_time*,

and outputs

- *vehicle_resultant_forces*.

Starting with *init_actuator_time*, it receives a column vector $n \times 1$ containing the time an actuator will start receiving inputs, the k-th element references the k-th actuator time to start. The *simulation_time* input enables external clock to be used, for example if one would like to control the vehicle in Simu2VITA using ROS[28] network and its clock, in this case the simulator makes the first value received as the base time and the simulation internal time is in reference to that base time. The output *vehicle_resultant_forces* is equivalent to the right hand-side of eq. (19) multiplied on left by $(M + M_a)$. The other inputs are

- *u_input* that is equivalent to $\mathbf{u}_a$ in eq. (31),
- *Alloc_matrix* is equivalent to *Sigma* matrix in Subsection 3.1.3,
- *external_forces* is ${}^b\boldsymbol{\tau}_e$ in eq. (18), and
- *current_velocity* is ${}^b\mathbf{v}_c$ in eq. (9).

Outputs are

- *actuator_output* being $\mathbf{y}_a$ as in eq. (35),
- *tau_input* being ${}^b\boldsymbol{\tau}_a$ as in eq. (18),
- *nu_dot* being $\dot{\mathbf{v}}_b$ as in eq. (19),
- *nu* being $\mathbf{v}_b$ as in eq. (5),
- *eta_dot* being ${}^w\dot{\boldsymbol{\eta}}_b$ as in eq. (6),
- *eta* being ${}^w\boldsymbol{\eta}_b$ as in eq. (4),

- *tau* being $\boldsymbol{\tau}_b$ in eq. (19),
- *current_velocity* is ${}^b\mathbf{v}_c$ in eq. (9).

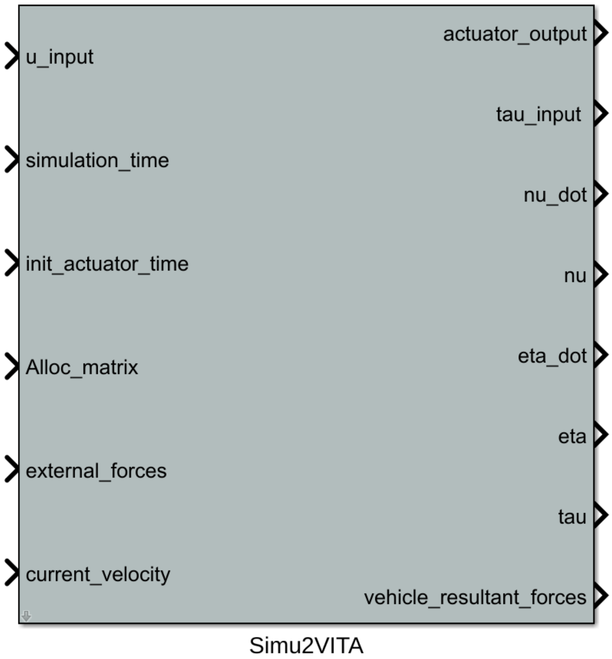

Figure A1. This is how Simu2VITA block is presented on Simulink.

Each one of the three modules of Simu2VITA has a tab dedicated to entering information. The tab for the Actuator Module needs the total number of actuators to be simulated, their initial state, the initial time they start to receive input and if Simu2VITA is going to use some external clock base. Next it presents the saturation options, the lower and upper limits and one can also disable the saturation. Last the time constant for each actuator is informed. See Figure A2.

The tab for the Allocation Module contains only the field for entering with a static allocation matrix, but an option to use an external source is also available. The internal machinery of Simu2VITA will correctly pick the chosen matrix based on the option of use an external allocation matrix. See Figure A3.

Information regarding the Dynamics Module involves scalars, matrices and vectors presented on subsections 3.1.1 and 3.1.2. From the vehicle, are required its mass, volume, center of gravity and its inertia matrix. For the hydrodynamics parameters, the added mass, linear and non-linear damping factors, and the water current velocity. Observe that the damping factors are considered diagonal matrices thus the input is a column vector for both fields. The hydrostatic parameters are the gravity constant, the water constant and center of bouyancy of the vehicle. The last two parameters are the initial pose and the initial velocity of the vehicle. See Figure A4.

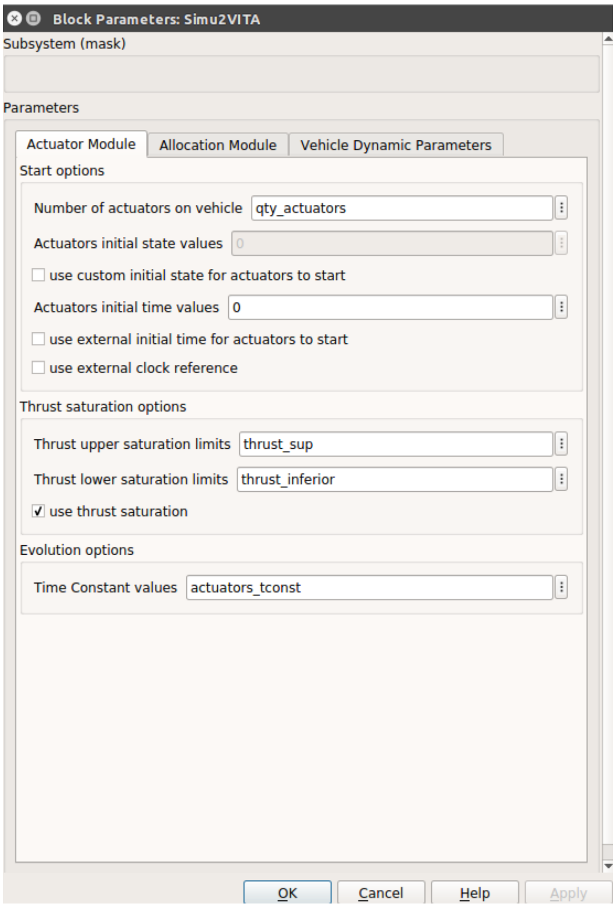

Figure A2. Simu2VITA interface for entering with simulation parameters for the Actuator Module.

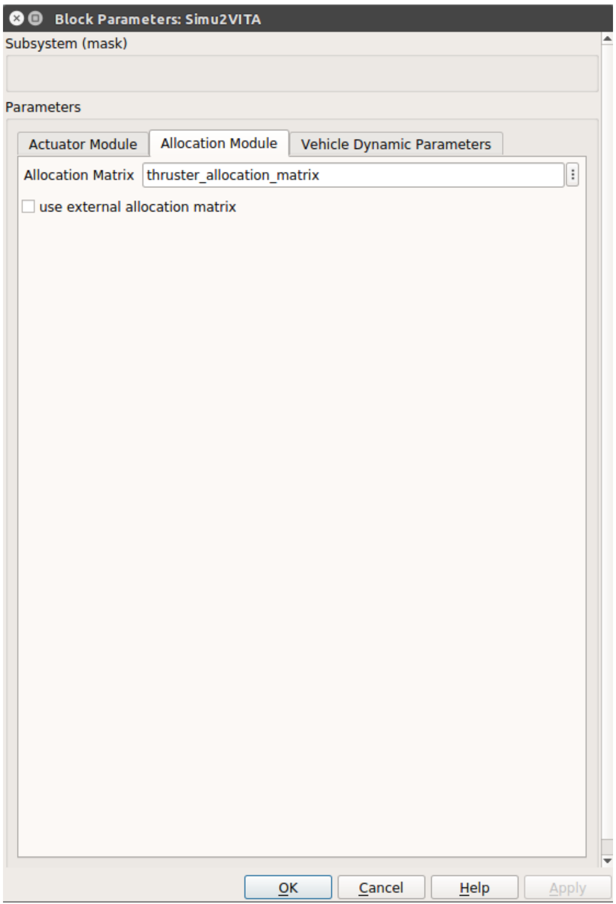

Figure A3. Simu2VITA interface for entering with simulation parameters for the Allocation Module.

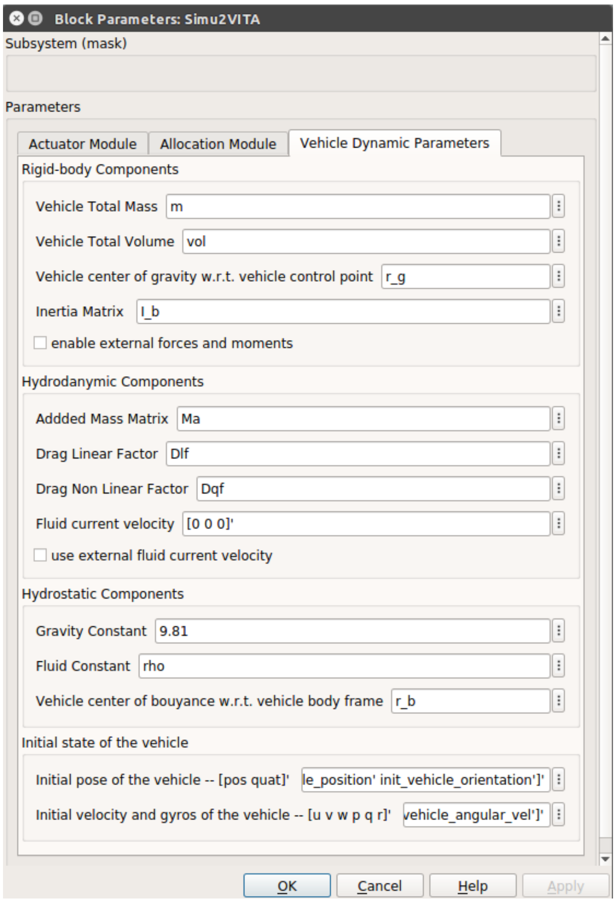

Figure A4. Simu2VITA interface for entering with simulation parameters for the Dynamics Module.

References

1. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, Vol. 3, pp. 2149–2154.
2. Michel, O. Webots: Professional Mobile Robot Simulation. *Journal of Advanced Robotics Systems* **2004**, *1*, 39–42.
3. Gazebo, an Open Source Robotics Foundation simulator. Simulation Description Format (SDF). URL: http://sdformat.org/, January, 2022.
4. Robot Operating System – ROS, an Open Source Robotics Foundation software development kit. Unified Robot Description Format (URDF). URL: https://wiki.ros.org/urdf, January, 2022.
5. Haidu, A.; Hsu, J. Fluids. URL: https://gazebosim.org/tutorials?tut=fluids&cat=physics, 2014. Accessed March 9, 2022.
6. Haidu, A.; Hsu, J. Bouyancy. URL: https://gazebosim.org/tutorials?tut=fluids&cat=physics, 2014. Accessed March 9, 2022.
7. Foundation, O.S.R. Aerodynamics. URL: http://gazebosim.org/tutorials?tut=aerodynamics&cat=physics, 2014. Accessed March 9, 2022.
8. Manhães, M.M.M.; Scherer, S.A.; Voss, M.; Douat, L.R.; Rauschenbach, T. UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation. OCEANS 2016 MTS/IEEE Monterey, 2016, pp. 1–8. doi:10.1109/OCEANS.2016.7761080.
9. The Society of Naval Architecture and Marine Engineers. Nomenclature for treating the motion of a submerged body through a fluid. *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin No.* **1950**, pp. 1–5.
10. Hart, J.C.; Francis, G.K.; Kauffman, L.H. Visualizing Quaternion Rotation. *ACM Trans. Graph.* **1994**, *13*, 256–276. doi:10.1145/195784.197480.
11. Fossen, T.I. *Handbook of marine craft hydrodynamics and motion control*; John Wiley & Sons, 2011.
12. Dukan, F. ROV motion control systems. PhD thesis, 2014.
13. Jorge, V.A.M.; Gava, P.D.d.C.; Silva, J.R.B.F.; Mancilha, T.M.; Vieira, W.; Adabo, G.J.; Nascimento Jr., C.L. VITA1: An Unmanned Underwater Vehicle Prototype for Operation in Underwater Tunnels. 2021 IEEE International Systems Conference (SysCon); IEEE: Vancouver, BC, Canada, 2021; pp. 1–8. doi:10.1109/SysCon48628.2021.9447108.
14. Blue Robotics Inc. . BlueROV2 Heavy Configuration Retrofit Kit. URL: https://bluerobotics.com/store/rov/bluerov2-upgrade-kits/brov2-heavy-retrofit-r1-rp/, January, 2022. SKU: BROV2-HEAVY-RETROFIT-R2-RP.
15. Blue Robotics Inc. . Ping Sonar Altimeter and Echosounder. URL: https://bluerobotics.com/store/sensors-sonars-cameras/sonar/ping-sonar-r2-rp/, January, 2022. SKU: PING-SONAR-R3-RP.
16. Tritech International Limited, a Moog Inc. Company. Gemini 720im Multibeam Sonar. URL: https://www.tritech.co.uk/product/gemini-720im, January, 2022.
17. Imagenex Technology Corp. . 881L Profiling – Digital Multi-Frequency Profiling Sonar. URL: https://imagenex.com/products/881l-profiling, January, 2022.
18. Blue Robotics Inc. . Low-Light HD USB Camera. URL: https://bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/, January, 2022. SKU: CAM-USB-WIDE-R1-RP.
19. Fresk, E.; Nikolakopoulos, G. Full quaternion based attitude control for a quadrotor. 2013 European Control Conference (ECC). IEEE, 2013, pp. 3864–3869.
20. de Cerqueira Gava, P.D.; Jorge, V.A.M.; Nascimento Jr., C.L.; Adabo, G.J. AUV Cruising Auto Pilot for a Long Straight Confined Underwater Tunnel. 2020 IEEE International Systems Conference (SysCon), 2020, pp. 1–8. doi:10.1109/SysCon47679.2020.9275846.
21. Meier, L.; Tanskanen, P.; Fraundorfer, F.; Pollefeys, M., PIXHAWK: A system for autonomous flight using onboard computer vision. In *2011 IEEE International Conference on Robotics and Automation*; 2011; pp. 2992–2997. doi:10.1109/ICRA.2011.5980229.
22. ArduPilot Project . ArduSub. URL: https://www.ardusub.com/, January, 2022.
23. Blue Robotics Inc. . T200 Thruster. URL: https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/, January, 2022. SKU: T200-THRUSTER-R2-RP.
24. Jorge, V.A.M.; de Cerqueira Gava, P.D.; de França Silva, J.R.B.; Mancilha, T.M.; Vieira, W.; Adabo, G.J.; Nascimento Jr., C.L. Analytical Approach to Sampling Estimation of Underwater Tunnels Using Mechanical Profiling Sonars. *Sensors* **2021**, *21*. doi:10.3390/s21051900.
25. Pittelkau, M.E. Rotation Vector in Attitude Estimation. *Journal of Guidance, Control, and Dynamics* **2003**, *26*, 855–860, [https://doi.org/10.2514/2.6929]. doi:10.2514/2.6929.
26. Water Linked . DVL A50. URL: https://store.waterlinked.com/product/dvl-a50/, January, 2022.
27. Mai, N.; Ji, Y.; Woo, H.; Tamura, Y.; Yamashita, A.; Hajime, A. Acoustic Image Simulator Based on Active Sonar Model in Underwater Environment. 15th International Conference on Ubiquitous Robots (UR), 2018, pp. 775–780. doi:978-1-5386-6334-9/18.
28. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. ROS: an open-source Robot Operating System. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) Workshop on Open Source Robotics; , 2009.