

Design of a Co-simulation Environment for the Docking Maneuver of an Autonomous Underwater Vehicle using Radio Frequency and Acoustic Hybrid Communication

by
Oytun Erdemir

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

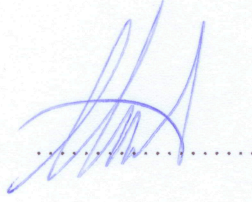
Graduate Program in Electronics Engineering
Sabanci University

2018

Design of a Co-simulation Environment for the Docking Maneuver of an Autonomous Underwater Vehicle using Radio Frequency and Acoustic Hybrid Communication

APPROVED BY:

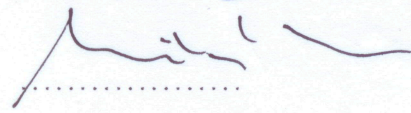
Assoc. Prof. Ahmet Onat
(Thesis Supervisor)



Prof. Özgür Gürbüz



Prof. Sandor Markon



DATE OF APPROVAL: 31.07.2018

© Oytun Erdemir 2018

All rights reserved.

to my beloved wife Begüm...

ACKNOWLEDGEMENTS

First and foremost, I would like to express my gratitude to my supervisor Dr. Ahmet Onat for his constant guidance, knowledge and support.

I am thankful to my thesis jury members, Dr. Özgür Gürbüz and Dr. Sandor Markon for accepting to be part of thesis jury and their valuable feedback.

I would like to thank my family for their constant support and encouragements. I would like to thank my wife Begüm for her constant love and support during my studies and thesis. I will forever be grateful for that.

I thank my fellow project mates and colleagues Saeed Nourizadeh Azar and Gökalp Çetin, for their support in this thesis. Also I thank my friends, Alper Güner, Didem Koçhan, Emrehan Durmaz, Abdurrahman Burak, Başak Tavşanoğlu for the fun environment they have created during my years in Sabancı University.

This survey has been done as part of the work that is being undertaken for the SWARMs (Smart and Networking Underwater Robots in Cooperation Meshes) research project (ECSEL project number: 662107).

ABSTRACT

Design of a Co-simulation Environment for the Docking Maneuver of an Autonomous Underwater Vehicle using Radio Frequency and Acoustic Hybrid Communication

OYTUN ERDEMİR

M.Sc. Thesis, July, 2018

Supervisor: Assoc. Prof. Dr. Ahmet Onat

In today's world, more research is needed on both underwater communication networks and autonomous underwater vehicle control. The main reason for this is, most of the modern technologies lose its function, partially or completely, due to negative conditions in the underwater environment. In this thesis, the starting point is to create a networked control system (NCS) using acoustic and Radio Frequency (RF) hybrid communication. Acoustic communication can be used even in long ranges, up to kilometers, but it offers low data rate and high propagation delays. On the other hand, RF communication provides high data rates and low delays, however, due to high attenuation, it can only be used in a 10-meter range in the modern technology. The proposed hybrid communication system uses RF communication in 10-meter range and acoustic communication in outside of that range. A co-simulation environment with Gazebo, a realistic physics simulator of the vehicle dynamics, and TrueTime, a realistic simulator of the real-time computer and physical characteristics and protocols of the communication channel, has been created for this purpose. However, since the selected simulators both have dynamic time step solvers, the time difference between simulation times

causes excessive time skew which can lead to instability of control loops of the NCS. Thus, a time synchronization is applied between the two simulation environments. The main contributions of this thesis are creating the co-simulation environment, solving the time synchronization problem and characterizing the synchronization delays and lowering the delays to ensure it will not affect the simulation realism.

ÖZET

Bir Otonom Sualtı Aracının Radyo Frekans ve Akustik Karma Haberleşme Kullanarak Kenetlenme Manevrası Uygulaması için Eşzamanlı Benzetim Ortamı Tasarlanması

OYTUN ERDEMİR

Yüksek Lisans Tezi, Temmuz, 2018

Danışman: Doç. Dr. Ahmet Onat

Günümüzde hem haberleşme hem de kontrol alanlarında sualtı robotlarıyla ilgili araştırmalar yapılmasına önemli ölçüde gerek duyulmaktadır. Bu durumun başlıca nedeni ise sualtındaki olumsuz koşullar nedeniyle halihazırdaki birçok teknolojinin işlevlerini tam veya kısmi olarak kaybetmesidir. Bu tez, bir karma sualtı haberleşme sisteminin bir kontrol uygulamasında kullanılarak bir ağ bağlantılı kontrol sistemi oluşturmasıyla ortaya çıkmıştır. Akustik haberleşme, yüksek mesafelerde, kilometreler mertebesinde, kullanılabilmesine rağmen, düşük veri hızına ve yüksek yayılma gecikmesine sahiptir. Diğer yandan, radyo frekans (RF) haberleşme, yüksek veri hızına ve düşük gecikmelere sahip olsa da suyun özellikleri sebebiyle yüksek sönmülmeye maruz kalmakta ve 10 metre menziline aşamamaktadır. Önerilen karma haberleşme sistemi, 10 metre menzilde RF haberleşme, bu menzilin dışında ise akustik haberleşme kullanılarak tasarlanmıştır. Bu amaçla, araç dinamiklerinin gerçekçi fizik benzetimini yapan Gazebo ile gerçek zamanlı bilgisayar ve haberleşme kanalının fiziksel karakteristiklerinin ve protokollerinin benzetimini yapan TrueTime ile bir eşzamanlı benzetim sistemi oluşturulmuştur. Ancak, iki benzetim aracı da dinamik zaman adımlarına sahip oldukları için, ar-

alarında zaman sapması oluşarak, ağ bağlantılı kontrol sisteminin kontrol çevrimlerinde kararsızlığa yol açabilmektedir. Bu sebeple, iki benzetim aracı arasında bir zaman senkronizasyonu yöntemi uygulanmıştır. Bu tezin ana katkıları, eşzamanlı benzetim sistemini oluşturmak, zaman senkronizasyonu problemine çözüm bularak, bu çözüm sonucunda ortaya çıkacak zaman farklarının karakterize edilerek, sistemin gerçekçiliğini etkilememesini sağlayacak seviyeye indirmektir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	v
ABSTRACT	vi
ÖZET	viii
LIST OF FIGURES	xiii
LIST OF TABLES	xiv
LIST OF SYMBOLS	xv
LIST OF ACRONYMS/ABBREVIATIONS	xvi
1. INTRODUCTION	1
2. BACKGROUND	3
2.1. Co-Simulation Approaches	3
2.2. Models of the Communication Channels for Networked Control Systems	4
2.2.1. Acoustic Communication	4
2.2.1.1. Acoustic Path Loss Model	4
2.2.1.2. Acoustic MAC Protocols	6
2.2.2. RF Communication	6
2.2.2.1. RF Path Loss Model	6
2.2.3. RF MAC Protocols	7
2.3. Realistic Simulation of Underwater Networked Control Systems	7
2.3.1. Gazebo	8
2.3.2. Robot Operating System	9
2.3.3. Unmanned Underwater Vehicle Simulator	11
2.3.3.1. Thruster Manager	11
2.3.3.2. Controllers	11
2.3.3.3. Vehicle Models	12
2.3.3.4. World Models	12
2.3.4. TrueTime	12
2.3.4.1. Network Block	12
2.3.4.2. Kernel Block	13

3. PROBLEM DEFINITION	15
3.1. Hybrid Communication	15
3.2. Docking Maneuver	15
3.3. Co-Simulation	16
4. CO-SIMULATION ENVIRONMENT	17
4.1. Time Synchronization	19
4.1.1. Proposed Method for Synchronization	20
4.1.2. Modifications on Real-Time Pacer Block	21
4.2. Control Method	22
4.2.1. PID parameter selection	23
4.2.2. Waypoint Management	26
4.2.3. Heading Reference Calculation	28
4.3. Networked Control Architecture	28
4.4. Communication Protocols	30
4.4.1. Medium Access Scheme for Acoustic Mode	30
4.4.1.1. Time-Division Multiple Access	30
4.4.1.2. Frame Time for Acoustic Mode	31
4.4.2. Medium Access Scheme for RF Mode	32
4.4.3. Packet Structure	33
4.4.4. Adaptive Power Control	34
5. EXPERIMENTS AND RESULTS	35
5.1. EXPERIMENTAL DESIGN	35
5.1.1. AUV Thruster Controller	35
5.1.2. Tasks Performed within the AUV Real-Time Computer	38
5.1.3. Tasks Performed within the Docking Station Real-Time Computer	40
5.1.4. Application of the Calculated Force to the AUV	41
5.1.5. TrueTime Model for Experiments	43
5.1.6. Implementation of TDMA	44
5.2. Simulation Results	44
5.2.1. Control Delay Characterization	44
5.2.2. Experiment Results	47

5.2.3. Approach to the Docking Station in Still Water	49
5.2.4. Docking Maneuvre with Water Currents	51
6. CONCLUSION	54
REFERENCES	55
APPENDIX A: MODIFIED REAL-TIME PACER CODE	59

LIST OF FIGURES

2.1	Gazebo Simulation Screen	8
2.2	TrueTime Network Block Settings Screen.	13
4.1	Block diagram overview and the detail of the Simulink model for the proposed system implementation	17
4.2	Real-Time Pacer Library Blocks.	21
4.3	Closed Loop of the Control System.	25
4.4	Waypoint Management.	27
4.5	Block diagram of AUV and docking station navigation systems. The acoustic and RF communication links are shown in red and blue respectively.	29
4.6	TDMA in both downstream and upstream periods	31
4.7	Packet structure from the docking station (downstream) packets .	34
4.8	Packet structure for AUV messages	34
5.1	Root locus analysis of the simplified linear model.	37
5.2	Step responses of Model versus Simulation.	38
5.3	World Frame and Body Frame.	42
5.4	TrueTime model used in the Experiments.	43
5.5	Time difference between Gazebo and Simulink simulation clocks. .	45
5.6	A representative screen-shot of the simulation environment in Gazebo.	48
5.7	AUV time to dock for acoustic only and hybrid.	50
5.8	Motive power for acoustic only and proposed hybrid.	51
5.9	Time to dock w.r.t. different current velocities.	52
5.10	Cumulative error w.r.t different current velocities.	53

LIST OF TABLES

5.1	Standard Deviation with respect to Real-Time Update Rate and Maximum Step Size	46
5.2	Simulation Parameters.	48
5.3	Communication Channel Parameters.	49

LIST OF SYMBOLS

r	Distance
k	Spreading Factor
f	Frequency
R	Rayleigh Fading Random Variable
P_c	Circuit power
t_{ds}	Docking Station Packet time
K_p	Proportional gain
K_i	Integral gain
K_d	Derivative gain
T_p	Derivative time
T_s	Sampling Period
α	Absorption Coefficient
ω	Frequency in rad/s
μ	Permeability of space
ϵ	Permittivity of water
σ	Conductivity of water

LIST OF ACRONYMS/ABBREVIATIONS

AUV	Autonomous Underwater Vehicle
NCS	Networked Control System
RF	Radio Frequency
ROS	Robot Operating System
SIL	Software-In-the-Loop
UUVSim	Unmanned Underwater Vehicle Simulator
DoF	Degree of Freedom
API	Application Programming Interface
MAC	Medium Access Protocol
SNR	Signal to Noise Ratio
TDMA	Time Division Multiple Access
CSMA	Carrier-Sense Multiple Access
CSMA/CD	Carrier-Sense Multiple Access with Collision Detection
FDMA	Frequency Division Multiple Access
USBL	Ultra Base Short Line
PID	Proportional Integral Derivative

1. INTRODUCTION

Leveraging the possibilities of swarms of underwater robots has attracted significant attention since it helps to avoid sending divers to hazardous underwater missions. Underwater applications range from instrument monitoring and climate recording to control and maintenance. To improve the efficiency of underwater missions, different types of Autonomous Underwater Vehicles (AUV) need to collaborate and communicate. Hence, there is a growing demand for high-speed communication between AUVs and also with base stations. However, data transmission in the harsh underwater environment poses major challenges, such as a limited bandwidth, long propagation delay and the unreliability of the environment. Under these conditions, establishing a reliable high rate data link is a critical task [1] [2].

The acoustic waves used in underwater acoustic communication can travel long distances for relaying information. However, due to high propagation delays and low data rates of the acoustic signal, it is not suitable for some applications such as remote guidance [3] [4]. Leveraging Radio Frequency (RF) communication provides high data rate and drastically reduced propagation delays, but, is constrained by short range resulting from high attenuation due to high conductivity and permittivity of water.

We consider the remote guidance task of an AUV which must land on a docking station anchored to the sea floor. Most AUVs do not have a position measurement device whereas the docking station has the capability to measure the position of the remote AUV, and then transmit the information to the AUV. This task can be regarded as a Networked Control System (NCS), where the position is measured by the docking station, relayed to the AUV over the communication network, which then makes use of the received position information for its guidance. The sequence is repeated at a constant rate, which becomes the sampling time of the discrete time NCS. Considering the data rate and propagation delay limitations of acoustic communication which may push the NCS to instability, and the short-range constraint of RF communication, we propose a novel underwater NCS with hybrid acoustic and RF communication modes

for implementing a docking maneuver application for AUVs. The system uses the acoustic mode for long range, which is greater than a threshold distance, and shifts to RF mode for shorter range. In the acoustic mode with low data rate and high latency at long range, low sampling rate and low control gains which are sufficient only for rough navigation must be used to avoid instability. In the RF mode, which offers high data rates, high sampling frequency and high control gains can be used instead, for precise maneuvering of AUVs at close proximity of the docking station. Since the system contains multiple vehicles, it is also required to implement different communication methods. However, the communication method is the subject of other ongoing work in parallel.

In this thesis, we will concentrate on a co-simulation platform designed for integrating the Unmanned Underwater Vehicle Simulator (UUVSim) [5] and a MATLAB Simulink based simulator named TrueTime [6]. UUVSim provides a realistic dynamic simulation of the underwater environment, AUV motions and hydrodynamic forces. TrueTime models and simulates hybrid real-time systems, communication networks and continuous plant dynamics. Especially, it models the real-time computers of the docking station and the AUVs in the Software-In-the-Loop (SIL) level, and communication links in the data transport level.

To the best of our knowledge, this is the first series of work which presents a real-time simulation of an underwater networked control application, considering both the full dynamics of the system and hybrid acoustic and RF communication for controlling AUVs in the networked control framework.

2. BACKGROUND

In this chapter, existing methods that are used for using two simulation environments together will be given followed by, underwater acoustic and radio frequency channel models, Medium Access Control (MAC) protocols for underwater communication channels and the simulation tools used in the thesis.

2.1. Co-Simulation Approaches

Co-Simulation is becoming more popular nowadays, as many different simulation environments are being developed which specialize in different aspects of a problem. To create a complex simulator that models a complex system, especially in interdisciplinary areas, it is often required to combine different simulators. The co-simulation environments are capable of modeling more complex systems in a desired level of detail.

There are several aspects that should be examined when a co-simulation environment is being constructed. In this thesis, the main simulators that will be combined in a co-simulation are MATLAB TrueTime and Gazebo. The main problem is the time synchronization between these two environments. Time synchronization is especially important in an NCS since excessive time skew can lead to instability of control loops which span different simulators. Extensive identifications and analyses have been conducted in [7]. These analyses have been made for discrete event based co-simulations, continuous time based co-simulations and hybrid co-simulations. In this thesis, the continuous time based co-simulation have been conducted.

There are examples of co-simulation environments that have been used in the past [8], which creates and examines co-simulation tools, for example integration of Modelica and ns-2. One of the examples of TrueTime being used in a co-simulation is also given in [9]. However, none of the works extensively handle the time synchronization problem, apart from several mentions of the time synchronization in [9]. Since the used tools include “ns-2” simulator in the co-simulation environment in the previous researches, it

has tools that enable the time synchronization. Since NS-2 is a batch based simulator, it is simpler to tackle time skews. However in this thesis, none of the described methods are applicable, since a closed loop control system needs to be implemented with one part residing in one simulator environment, and the other part residing on the other.

2.2. Models of the Communication Channels for Networked Control Systems

In this section, physical models such as path loss characteristics of the acoustic and RF communication channels will be provided, followed by suitable MAC protocols.

2.2.1. Acoustic Communication

In the underwater communication, the effect of path loss is relatively higher than the air environment. In the long range, the highest bit rate becomes less than 50kb/s, where a 20dB Signal to Noise Ratio (SNR) is observed in the modern modem characteristics. There are several reasons for the path loss to be higher in underwater, such as water characteristics (saltiness, density, temperature), terrain properties and objects in the water. The characteristics that has been used in this thesis for acoustic channel data rate and range, was based on the one of the modern modem. The modem that has been chosen was Evologics S2C R 48/78 Underwater Acoustic Modem [10].

2.2.1.1. Acoustic Path Loss Model. Path loss is the weakening of the signal while it travels from the transmitter to receiver. There are two elements in underwater acoustic channel, which are absorption loss and spreading loss [11]. Spreading loss happens because of the area that the acoustic signal covers at it spreads in the medium. Spreading loss is found by:

$$PL_{sp}(r) = k10\log(r) \tag{2.1}$$

where r is the distance in meters and k is the spreading factor. The spreading factor k depends on the geometric pattern of spreading of the acoustic waves. If the spreading is in spherical form spreading to every direction, then spreading factor becomes $k = 2$, and if it is bounded into a cylindrical shape, the spreading factor becomes $k = 1$. The spreading factor has been chosen as $k = 1.5$ in this design, since the system is neither bounded nor totally unbounded [12].

Absorption loss is caused by the friction and the ionic relaxation as the acoustic wave spreads in the the water body. Absorption loss is then given by:

$$PL_{ab}(r, f) = 10\log(\alpha(f))r \quad (2.2)$$

where r is the distance in kilometres and α is the absorption coefficient which depends on the specific body of water considered.

The total path loss is given as sum of spreading and absorption losses as [11]:

$$PL(r, f) = PL_{sp}(r) + PL_{ab}(r, f) \quad (2.3)$$

Then a Rayleigh fading model is applied on received power. Rayleigh fading model is generally used to describe the attenuation in the received power of a signal due to the scattering characteristics of the transmission channel. In most cases it can be described by a zero mean process with a phase evenly distributed between 0 and 2π , such as (2.5):

$$P_r(r) = \frac{2r}{\Omega} e^{-\frac{r^2}{\Omega}} \quad (2.4)$$

$$r \Rightarrow 0, \Omega = E(R^2). \quad (2.5)$$

where R is the random variable. Then the path loss is subtracted from the the signal

strength to find the received signal strength, such as (2.6).

$$P_r(dB) = P_t(dB) + PL \quad (2.6)$$

where $P_r(dB)$ is the received power in dB and $P_t(dB)$ is the transmitted power in dB .

2.2.1.2. Acoustic MAC Protocols. There are several MAC protocols that have been used extensively and effective comparisons have been made for underwater acoustics communication. An extensive research and comparison of underwater acoustic MAC protocols have been performed before [13]. The MAC protocols are divided into two categories, as contention-free and contention-based protocols. Since the purpose of this thesis is to create and analyze the performance of the co-simulation environment, it is sufficient to use one of the most widely used and relatively easy to implement MAC protocol in this design. Thus, Time Division Multiple Access (TDMA) protocol has been chosen among different protocols [14].

2.2.2. RF Communication

In RF communication, the path loss is significantly higher than the acoustic channel. It compensates greatly for this deficiency by its low propagation delay and high bandwidth. In freshwater, the data rate at 10 MHz frequency is more than 3 Mbit/s [15]. The RF modem that has been chosen was WFS seatooth® S300 [16] underwater RF modem.

2.2.2.1. RF Path Loss Model. RF communication path loss is mostly based on frequency, and the water's conductivity, permittivity and permeability. Since the system mostly operates on relatively high depths, the water-air boundary loss is neglected. The path loss is given in [17] as:

$$PL = RE(j\omega\sqrt{\mu\epsilon - j\frac{\sigma\mu}{\omega}}) \frac{20}{\ln(\alpha(f))} r \quad (2.7)$$

where RE denotes real part, r is distance, ω is the frequency in rad/s, μ is permeability of space, ϵ is permittivity of water, and σ is the conductivity of water [17]. After the path loss is subtracted from the transmitter power, Rayleigh fading is applied in a similar way to (2.5), using an exponential distribution, to get the final value of received power.

2.2.3. RF MAC Protocols

The RF MAC Protocols are in the early stage of development and definitive research results are not available. An investigation of TDMA for underwater RF communication protocol has been given in [18], and a survey of underwater RF protocols have been realized in [19]. The most appropriate and easy to use method since it is available in the TrueTime Network Block, and suitable for this design is Carrier-Sense Multiple Access (CSMA) protocol [20]. The selected MAC protocol was Carrier-Sense Multiple Access with Collision Detection (CSMA/CD) in this implementation.

2.3. Realistic Simulation of Underwater Networked Control Systems

There are some underwater simulation tools such as Aqua-Sim [21], which is based on NS-2 for simulating the underwater acoustic networks. However, they do not support the continuous time dynamics of the plant and disturbances since NS-2, the popular tool for analyzing communication channels, is batch processing based and therefore, does not support modelling of closed loop systems that we need to simulate. Hence to have a more realistic simulation we used Gazebo [22] which is a well-established simulator that makes it possible to quickly test algorithms, design robots and perform regression testing using realistic scenarios. Furthermore, we used Unmanned Underwater Vehicle Simulator (UUVSim) [5], which is an underwater robotics simulator running on Gazebo. Using UUV-Sim we can simulate multiple underwater robots with realistic underwater hydrostatic and hydrodynamic effects, thrusters, sensors, and external disturbances. In contrast to existing solutions, UUVSim reuses and extends the robotics simulation platform to underwater environments.

2.3.1. Gazebo

Gazebo is a simulation environment for simulating realistic physics and physical dynamics [22]. It is being developed by Open Source Robotics Foundation (OSRF) and is an open-source project. Gazebo is a versatile tool, and it has plugin support which enables significant modifiability. There is a vast community that develops variety of tools to enhance the Gazebo for different purposes. Moreover, Gazebo has an internal integration with Robot Operating System (ROS). Gazebo has been chosen for this work because of the mentioned reasons.

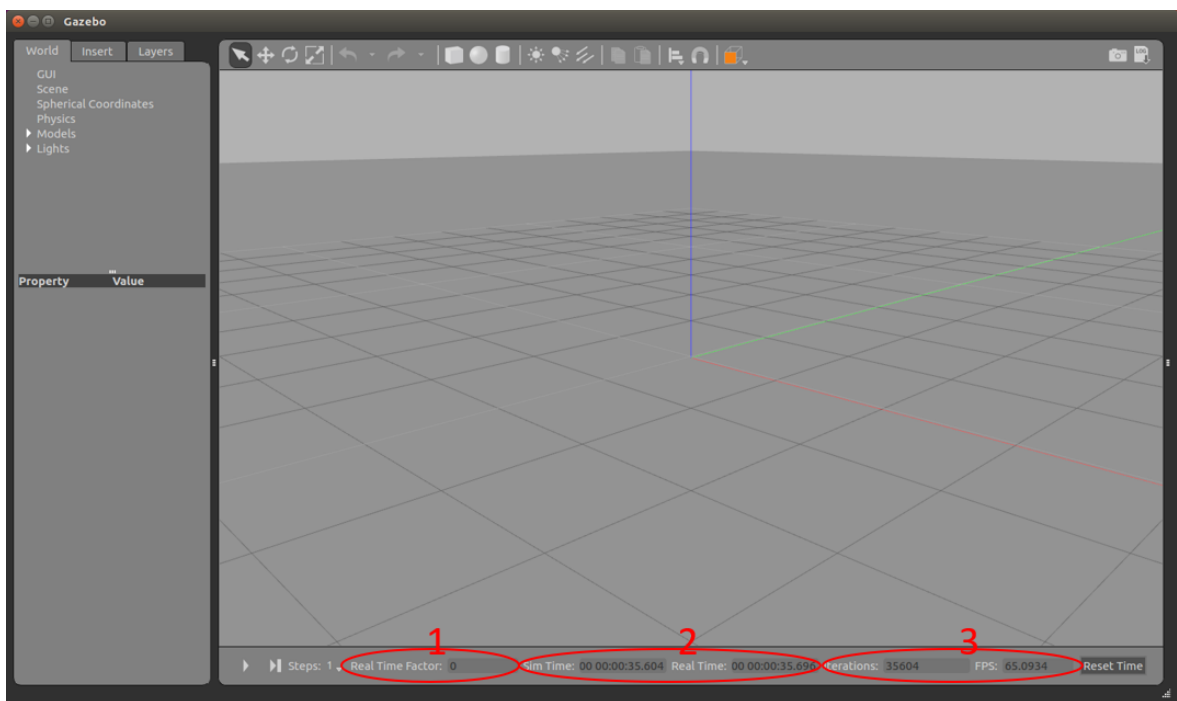


Figure 2.1: Gazebo Simulation Screen

In Gazebo tool, there are different variables that affect the simulation properties, such as the physics calculations per second, simulation time/real time ratio, etc. In Figure 2.1, the screen of Gazebo simulation interface example has been given. Some of the simulation properties can be observed and also accessed from the interface. Real Time Factor, which has been depicted as 1 in Fig. 2.1, is the ratio of Simulation Time and the Real Time. It displays the amount of simulation time that would be elapsed in one second of Real Time. The Simulation Time and Real Time, which has been

shown as 2 in Fig. 2.1, are the elapsed times for simulation time and the wall-clock time since the beginning of the simulation, respectively. Iterations and FPS, which has been numerated as 3 in Fig. 2.1, show the number of physics calculation iterations made since the beginning of the simulation, and the number of current frames per second the graphical interface provides.

In Gazebo, a world can be created by two different methods. The first approach is, opening an empty world in the Gazebo graphical user interface, and populating it manually with models through the interface. After the world has been populated, the world can be saved and re-opened through a command prompt. Saving the world with the graphical interface creates a file, which can be edited to alter some variables to desired values. The second method is to create a world description file. In the description file, all the models and world variables can be added or altered. With the description file, a launch file also is required which starts the Gazebo simulation with the given world description file. The first method is the simpler and more user-friendly; and the second method is the more reliable but time consuming method to create a world model among the two methods.

2.3.2. Robot Operating System

ROS is an operating system that has been developed as a robotics middleware [23]. It is an operating system that handles the implemented tasks concurrently. Since it supports concurrent task running, multiple vehicles can be operated using ROS. ROS has a low-level messaging system that provides a communication between the tasks inside a vehicle. Since it can simulate computer systems and handle the tasks concurrently, it is also a suitable tool for implementing real-time systems. In this work, ROS has been used for simulating and implementing the AUV dynamics side of the simulation.

ROS systems are built and compiled as a package. The packages have unique names and consist of tasks that are called nodes, unique message and service definitions. Every different node has a unique name and defined inside the ROS package before

they are compiled. The nodes are handled independently, and they run inside ROS concurrently. The messages and services are the corner stones of ROS's messaging system. The messages are created as topics inside the ROS system, topic is the unique name given to a message type when it is initiated. Any node in the system can subscribe to or publish to a topic, thus making it an anonymous system. On the other hand, the services are the two-way communication method in the ROS messaging system. A node can announce that it can handle a type of service, and it creates a callback function for that service. When an outside node calls the service, it can pass several variables to the callback function, then when the callback function concludes, it can return a value to the caller node. Thus, making the service method a two-way system.

The messaging system of ROS ensures the nodes can share information with each other, however physical communication networks and protocols are not natively implemented in ROS. Therefore, an additional simulation environment to have the most realistic combined network and messaging simulation is required. Also, even though ROS can simulate the real-time computer system part, TrueTime was used in this thesis, and reasons for that will be more apparent in the proceeding chapters.

Also, ROS is the binding layer that enable the integration of the two simulation environments that is used in this work, namely Gazebo and TrueTime, which will be explained next. Gazebo already has a structural integration with ROS. This provides the user to access Gazebo internal structures through ROS nodes. Gazebo and ROS also has a feature that enables them to synchronize to the same simulation time. Because of this feature, if the ROS can be synchronized with TrueTime, then the complete co-simulation can perform the simulation in a synchronized manner.

ROS also supports a graphical interface called "rviz". The rviz interface visualizes the messages and sensor data in the ROS system. It is mostly used together with Gazebo. Since Gazebo cannot visualize the sensor data rviz enhances the Gazebo's simulation realism. Gazebo, by providing extra data, such as position data and environment models, improves the performance of rviz. An example of Gazebo and rviz working together can be given as: a camera on the vehicle is visualized through rviz,

and Gazebo provides the position data, rviz can render the objects in Gazebo which then appear in the vehicle's camera.

2.3.3. Unmanned Underwater Vehicle Simulator

Underwater Unmanned Vehicle Simulator (UUVSim) [5], is a custom package that has been specially developed for simulating the underwater environment and vehicles. It has different plugins and models to simulate the environment. The most related plugins that has been utilized in this thesis are given in the below subsections.

2.3.3.1. Thruster Manager. UUVSim has a built-in global thruster manager for the vehicles to use. To integrate the thruster manager for a vehicle, a Thruster Allocation Matrix (TAM) should be provided.

$$\tau_C = (f_x, f_y, f_z, \tau_r, \tau_p, \tau_y)$$

This vector represents the output of the controller. f_i is the force values and the τ_i represents the torque values in the vehicle's body frame in Euler angles. The thruster manager then translates the output of the controller to the output thruster forces. It also manages the saturation of the forces, by taking the maximum thruster forces into account.

2.3.3.2. Controllers. UUVSim has internal controllers for the models. There is a controller module with a superclass for which can be adjusted to control any vehicle. One can create its own algorithm code and by tying the controller module to the algorithm code via superclass, and by determining the forces and torques, the vehicles can be controlled. The controller superclass also uses the thruster manager. RexROV, the vehicle model which has been used in this thesis, has an already developed controller, however since we require the communication channel packets, the vehicle cannot be controlled via this controller.

2.3.3.3. Vehicle Models. The model that has been used in this thesis is the RexROV model inside UUVSim. The models inside the UUVSim are constructed both visually and physically. Physical parameters are also available inside the model files which can be used to determine the underwater dynamics of the vehicle.

2.3.3.4. World Models. The world models inside the UUVSim are based on an empty underwater world. The empty underwater world only contains a water body with 100 meters depth, a flat terrain and a light source. There are different world models that has been populated with different types of terrains and objects such as windmills. In this thesis the empty underwater world model has been chosen, since the implementation does not require object avoidance or other mechanisms related to objects.

2.3.4. TrueTime

TrueTime is a simulation environment designed for real-time computer systems and the communication network dynamics [6] as a toolbox of MATLAB Simulink. TrueTime is a pre-built system, however the source codes are also available, should anyone wishes to modify and use them. There are mainly two blocks that has been utilized in this thesis as detailed below. There are also different blocks which will not be explained. Those blocks are send, receive, battery, wireless network and ultrasound network blocks.

2.3.4.1. Network Block. Network block is the block that is responsible for handling the communication network dynamics. Network block supports several built-in MAC protocols. An example of the settings and variables that can be set in Network block is given in Figure 2.2. This is a relative example, because when the network type changes, some related variables are added or removed from the block. Also, network block keeps track of a schedule which maps all of the nodes that are currently connected to the network block. The schedule shows when a node is transmitting, waiting to transmit or idle.

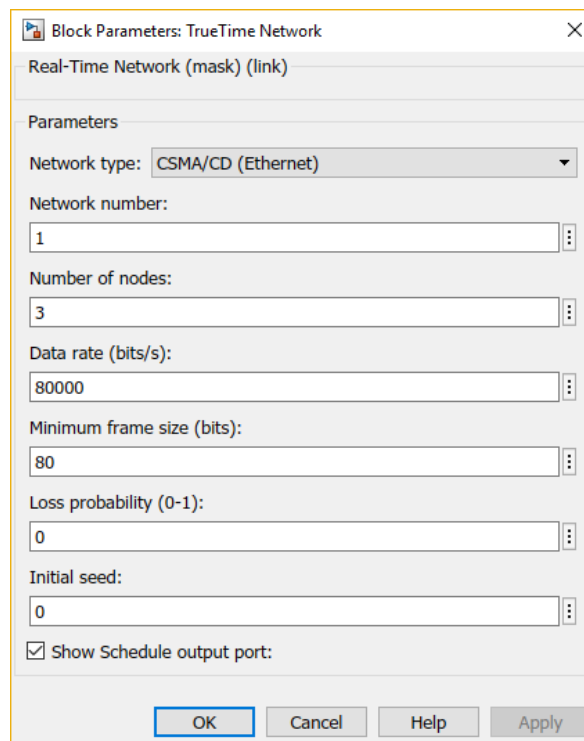


Figure 2.2: TrueTime Network Block Settings Screen.

In Figure 2.2, network number is the number assigned for this specific network node. Number of nodes is the number of nodes that are connected to this network. Data rate is self explanatory. Minimum frame time sets the minimum amount of transmission that will happen in one transmission instance. Loss probability, is the chance of a packet to be lost, and initial seed is the seed for the random number generator of the loss probability.

2.3.4.2. Kernel Block. Kernel block is the block that is responsible for the real-time computer system simulation in TrueTime. The kernel blocks consist of different tasks. Two types of tasks can be created inside a kernel block. The first type is asynchronous, which is not called periodically, however it is tied to an event, such as message arrival. Asynchronous tasks are The second type is synchronous tasks. Synchronous tasks are called periodically and can be assigned a start time and a deadline.

Kernel block also has support for three scheduling algorithm. The designer can

select one of the scheduling algorithms based on the requirements of the design. The supported scheduling algorithms are: fixed-priority scheduling, deadline-monotonic scheduling and earliest-deadline-first scheduling. In this thesis, fixed-priority scheduling was used.

3. PROBLEM DEFINITION

In this thesis, we will implement a realistic simulation of the docking maneuver using the proposed hybrid communication networked control method with realistic physics simulator for the vehicle dynamics as well as realistic simulation of the real-time computer in the firmware source code level and physical characteristics and protocols of the communication channel in the transport level.

3.1. Hybrid Communication

The starting point of this work was to implement a hybrid network controller scheme for underwater vehicles to increase the underwater control performance. The hybrid communication consists of an RF network and an acoustics network. As explained in 2.2.1, acoustics communication has a low bandwidth and high propagation delay thus provides a low performance for an NCS. However, it has significantly longer range with respect to RF communication. Since RF communication has a low range, it cannot be used as the sole communication method. By combining the high bandwidth and high-frequency properties of RF communication, we intend to increase the performance of the control. This idea has been examined previously in [24] [25] [26].

The underwater docking maneuver system consists a number of vehicles instead of only one vehicle. Since all of the vehicles would be sending and receiving communication packets through only one network, Medium Access Control (MAC) protocols will also have effect on the performance of the docking maneuver. Most appropriate MAC protocol should be chosen to have the best performance.

3.2. Docking Maneuver

A docking maneuver implementation has been chosen as the main experiment in this work. The docking maneuver is being performed by the AUV to a docking station. The vehicle cannot determine its position underwater by its own. The typical

technology that is currently used underwater to find the position of a vehicle is the Ultra Base Short Line (USBL). Since the USBL module is quite heavy and occupies a rather large space, it is not feasible to attach it to all vehicles. The solution for this is to have a docking station that hosts the USBL module and determines the position of any vehicle that is underwater and in the vicinity. By using the communication network, the docking station periodically sends the position data of the vehicle to the related vehicle. Then, using this position information, the vehicle applies the control and performs its task. Since the input information that is used in vehicle guidance and control is sent through the network, this scenario is an example of a Networked Control System (NCS).

3.3. Co-Simulation

The goal is to have a simulation environment that is as realistic as possible. To achieve this, we have selected one of the most realistic underwater physics simulators, namely Gazebo's UUVSim package. UUVSim can simulate the underwater dynamics such as buoyancy, currents, etc. However, to create a realistic NCS simulation, a communication network, protocol simulation and real-time computer system simulations are also required. Gazebo and ROS do not support a realistic network simulation; thus, we have selected MATLAB Simulink's TrueTime to simulate the communication network and protocol.

Implementing the system with two different simulation environments made the system a co-simulation. We had to ensure the information exchange between the two simulations. For this purpose, the MATLAB's integration library for ROS has been utilized. By enabling the information exchange between ROS and MATLAB, the connection with Gazebo and MATLAB has also been accomplished. However, having two different dynamic simulation environments has brought forth the synchronization problem, and solving the synchronization problem is one of the main contributions of this thesis.

4. CO-SIMULATION ENVIRONMENT

One of our main contributions in this thesis is achieving the co-simulation of the aforementioned simulation environments. Proposed configuration is depicted in Figure 4.1, with detail on how the Simulink part is implemented. Our objective is to integrate the different components of the co-simulation environments in a way that we can control the AUV accurately for accomplishing the tasks. However, there are timing difficulties for applying co-simulation between TrueTime and Gazebo. The main challenge comes from the dynamic time steps used in the solvers of the tools, which cause simulation times of the two simulators to advance in different time increments. This causes the simulation times to diverge from each other unless precautions are taken, which may result in the co-simulation to fail and yield unreliable results.

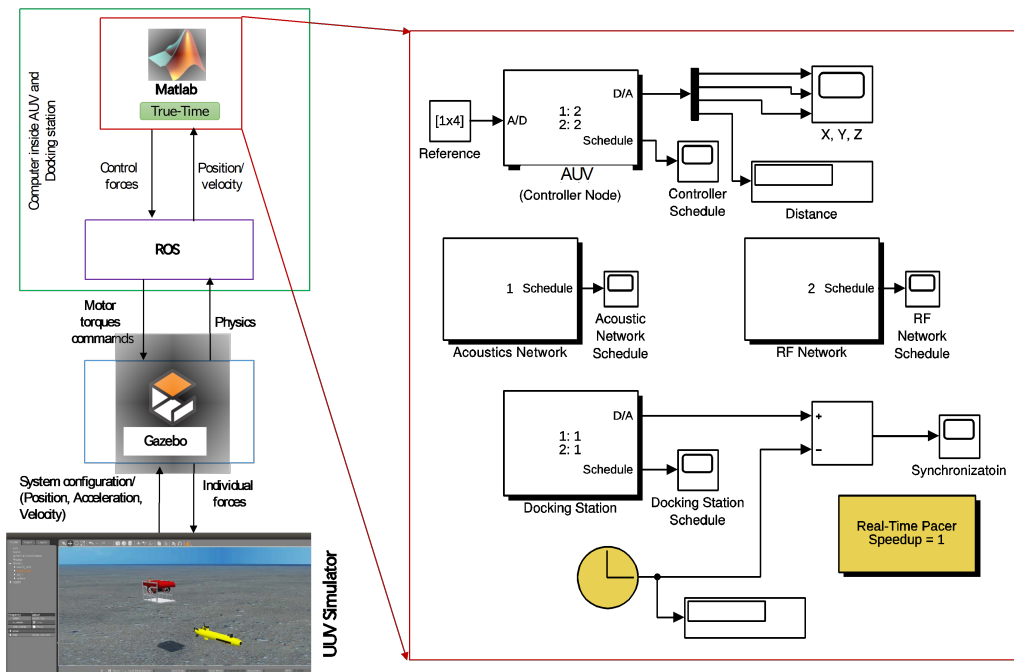


Figure 4.1: Block diagram overview and the detail of the Simulink model for the proposed system implementation

To provide consistency of the variables across the co-simulation system, messaging features of ROS are used. For this purpose, Gazebo and ROS are initialized together,

and ROS uses the clock of Gazebo for adjusting its timer. Gazebo is able to receive and send information through ROS, which makes ROS the messaging channel between Gazebo and TrueTime. We have implemented a position controller in Matlab to control the vehicle as well as a communication network simulator which take into consideration the physical conditions of the environment such as the instantaneous distance between nodes to calculate transmission power loss. As soon as the simulation is started the controller and communication networks are initiated as well.

In Figure 4.1, the real-time computer, network and the synchronization nodes can be seen in the TrueTime model in the inset. The nodes labeled as Acoustics Network and RF Network are the network modules that provide the communication network simulation between the docking station and the vehicles. Also, the real-time computers of docking station and AUVs are labeled as “AUV (Controller Node)” and “Docking Station”. These nodes are the kernels of AUV and Docking Station, respectively. The kernels operate the tasks, such as thruster controllers, sending and receiving communication packets, etc. The yellow block which is labeled as “Real-Time Pacer” in the model, is the method we use to synchronize the clocks of the two simulators, which will be explained further in upcoming sections in detail.

The docking station is responsible for determining the position of the AUV via using the position sensor and transmitting position data to the AUVs. The position data is taken from Gazebo’s ROS messages and sent through the communication network simulation infrastructure provided by TrueTime, to the AUVs. When the co-simulation environment is started, we observe that the computer in docking station first reads the vehicle position from Gazebo and then transmits vehicle position through the network node in TrueTime. By receiving the position information, AUV knows its location and can use it for controlling its thrusters. The thruster powers are calculated by the real-time embedded system simulator nodes labeled as “AUV (Controller Node)” in TrueTime, and transferred to Gazebo through ROS messages to be applied to the physics side of the co-simulation. This cycle iterates until AUV finishes its docking maneuver and lands on the docking station. By creating such a design, we have achieved a simulation for a networked control scenario, implementing

realistic underwater dynamics, communication networks and real-time computers.

4.1. Time Synchronization

Both Gazebo and MATLAB Simulink are dynamic simulation environments, which means that, they have differing time steps at each iteration. This causes them to run at different paces. Because of this reason, when they are in a co-simulation, there will be time skew between them which will cause problems in terms of causality and delays in control loops. The latter is a significant problem since a networked control system is implemented with the plant physics on the Gazebo side and communication network, controller algorithm and real-time computer implemented on the MATLAB side. To solve the time skew problem, the simulation environments must be synchronized, and the synchronization must be statistically characterized to show that its effect on the overall results can be negligible.

The simplest approach is to synchronize both simulation environments with wall-clock time [27]. This way, both environments would be synchronized with a common element. Even though both simulation environments can be synchronized with wall-clock time, using them have different uncertainties. Gazebo can be synchronized to wall-clock time by setting the Real Time Factor value to one. However, Gazebo does not guarantee that it will have a constant Real Time Factor. If the computational load becomes heavy, the system slows down the simulation to deal with the computations, thus creating a variance in the simulation time and wall-clock time. On the other hand, TrueTime can also be synchronized to the wall-clock time, however the method to do so also has variance between the simulation time and wall-clock time in the order of 10 to 30 milliseconds. When both methods are used together, the synchronization variance gets enhanced and creates further inconsistencies. Therefore, it was decided to synchronize one of the environments to the other.

For synchronizing one environment to another, the Real-Time Pacer Block of MATLAB Simulink has been chosen as the main synchronization method. Modifications have been made to the Pacer Block to change its function.

4.1.1. Proposed Method for Synchronization

Simulink has a built-in pacer block, which is called “Real-Time Pacer Block” [28]. The function of the pacer block is to synchronize the Simulink simulation time to wall-clock time. The block utilizes the Simulink’s “pause” function to pace the simulation. Based on the way it operates, the pacer block can only slow down the simulation, i.e. if the simulation runs slower than the target time, then the pacer block will have not function correctly. However, Simulink often runs faster than the target time depending on the complexity of simulated model. This situation makes Real-Time Pacer block to be useful in most cases.

The pacer block is a MATLAB S-function, which is a system function that is a description of a Simulink block written in MATLAB, C, C++, or Fortran. In this case, the pacer block was written in MATLAB language. The update section of the pacer block code is only called at the major time steps, which happens at every change of outputs in the rest of the current design. There are two types of steps in the Simulink simulation environment named as minor and major time steps. Minor step happens whenever a block’s lower level internal outputs change values. This does not affect the output values of the actual block at the current step. Major step happens when a block’s output value changes, which means one can see the change in the current design level. This influences our approach while using this block.

The library which can be seen in Figure 4.2, shows us the Real-Time Pacer Library. The library contains the Real-Time Pacer block (left) and Elapsed Real-Time (right) blocks. The Elapsed Real-Time block outputs the elapsed real time, which can be understood from the name. However, since we are only interested in the simulation times of the two simulation environments and not the real time, this block has not been used in this project.

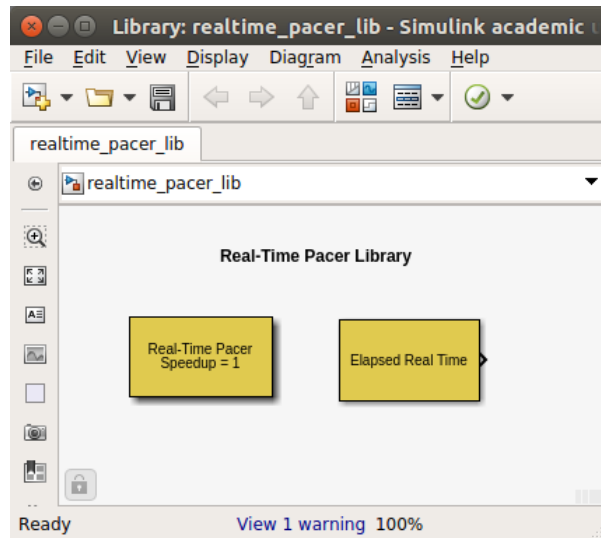


Figure 4.2: Real-Time Pacer Library Blocks.

4.1.2. Modifications on Real-Time Pacer Block

The pacer block has been implemented with some modifications in this design. The changes to the pacer block shifted its function from synchronizing Simulink with target time to synchronizing Simulink with the simulation time of Gazebo. At each synchronization period, the block is called and compares the difference between the Gazebo simulation time and the Simulink simulation time. It then proceeds to pause the Simulink environment for an amount of time which is equal to the difference between the simulation times.

The Real-Time Pacer Block contains an S-function and we have access to the internal code of the block. The code has been modified such that, instead of checking the difference between the Simulink simulation time and the real time as originally designed, it synchronizes to the `"/gazebo/clock"` topic which provides the simulation time of the Gazebo environment. The change has been made at the update section of the block. Instead of using the time difference between `"tic"` and `"toc"` operations, the `"/gazebo/clock"` message update difference is used to call the pause function of MATLAB Simulink. The modified code is provided in Appendix A.

4.2. Control Method

The AUVs are modelled in UUVSim and have full degree of freedom (DoF). Even though there are several options for selecting the AUV thruster controller, since there is no need for an advanced controller to analyze the delay characteristics, the proportional-integral-derivative (PID) controller has been chosen. The controller is implemented as separate PID controllers for each orthogonal axis that steers the AUV while damping its response. A discrete time approximation which is implemented in the simulations is shown in equations 4.1, 4.2, 4.3 and 4.4 [29].

$$P(kT_s) = K_p(r(kT_s) - y(kT_s)) \quad (4.1)$$

$$D(kT_s) = K_d(y(kT_s) - y((k-1)T_s))/T_s \quad (4.2)$$

$$I(kT_s) = K_i(T_s \frac{(y(kT_s) + y((k-1)T_s))}{2}) \quad (4.3)$$

The control signal is then given as:

$$u(kT_s) = P(kT_s) + I(kT_s) + D(kT_s) \quad (4.4)$$

The gains (K_p, K_i, K_d) and sampling period T_s of the digital controller are set according to the communication link used. The control signal for each axis comes from either of the two controllers; one for the acoustic link and other for RF link. At a long distance of more than the threshold distance, the acoustic link is used to send location and due to slow propagation speed and low data rate, the controller gains must be set small to avoid instability due to delay. When the distance is less than the threshold distance, high data rate and low propagation delay of RF communication link can be

used and controller gains are set to higher values. The controllers get the distance information via the communication links, sent by the docking station. In this thesis, threshold range which we consider is 10 meters [30].

4.2.1. PID parameter selection

The PID controller parameters must be calculated based on the linearized characteristics of the overall control system. If the linearized control system response is similar to the response obtained by the Gazebo simulation, we can calculate the PID controller parameters using an established method such as root locus. Else, we need to do hand tuning. The model of the control system in one degree of freedom, can be seen in Figure 4.3. Besides the plant and the controller, the communication delay is shown in the feedback loop since the measurement of the position takes time to transmit to the AUV.

The model's closed loop transfer function is in the form of:

$$G(s) = C(s) P(s) e^{-sT_d} \quad (4.5)$$

Where $C(s)$ and $P(s)$ are the transfer functions of the controller and plant, respectively, and the last term represents the communication time delay.

The plant is the vehicle's dynamic model, and we have access to its coefficients from the Gazebo simulation source files. It is in the form of $P(s) = Z(s)/F_z(s)$ (for the z axis in this example) and to derive this, $F_z(s)$ must be obtained from (for the example of Z dimension):

$$f_z(t) = a\ddot{z} + b\dot{z} \quad (4.6)$$

However, f_z is in time domain, and it has to be transformed to frequency domain. To

achieve this, the Laplace transform is applied.

$$\mathcal{L}(f_z(t)) = \mathcal{L}(a\ddot{z} + b\dot{z}) \quad (4.7)$$

Which is equal to:

$$F_z(s) = as^2Z(s) + bsZ(s) \quad (4.8)$$

Thus the plant equation becomes:

$$\frac{Z(s)}{F_z(s)} = \frac{1}{as^2 + bs} \quad (4.9)$$

After deriving the plant equation, the next equation is the PID controller. Since the root locus, which we will see further in this section, can only be used for one variable, the controller's I and D variables should be tied to P. The relation becomes $K_p = n_d K_d$ and $K_p = n_i K_i$, respectively, where n_d and n_i are the relation coefficients.

The controller equation becomes:

$$K_p \frac{n_d s^2 + s + n_i}{s} \quad (4.10)$$

Lastly, the delay must be characterized. The Padé approximation is used for this purpose. The natural response should be found and based on the delay relation to the natural response of the plant, the order of Padé approximation will be decided. In section 5.1.1, it has been found that natural response of the plant is relatively close to the delay, so we apply the first order Padé approximation, which is given by [31]:

$$e^{-T_d s} = \frac{1 - T_d s/2}{1 + T_d s/2} \quad (4.11)$$

The method that is used to find the appropriate controller parameters is the root

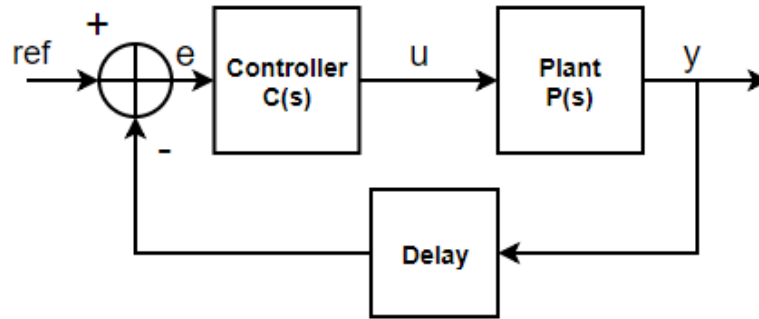


Figure 4.3: Closed Loop of the Control System.

locus method. The root locus method, places the poles and the zeros in a graphical manner. It is a method to derive closed loop analysis from the open loop transfer function. The poles and the zeros are derived from the mathematical model of the open loop system. The factors of the numerators of the open loop system are the zeros, and the factors of the denominators are the poles. The final open loop transfer function is in the form of from the equations 4.5, 4.9, 4.10 and 4.11:

$$G(s) = \frac{(k_d s^2 + s + k_i)(1 - T_d s/2)}{s(as^2 + bs)(1 + T_d s/2)} \quad (4.12)$$

Then, the zeros and the poles can be obtained from the UUVSim sources and the actually measured time delay in the simulation etc, and inserted in eq. 4.12 so that the root locus can be applied. Root locus analysis can be performed using numerical tools provided by MATLAB, which has libraries for deriving the root locus from the mathematical model. In this thesis, MATLAB's tools has been used for root locus analysis.

MATLAB's "rlocus" function creates a figure that depicts all the zeros and poles, and their change with respect to the controller gain. It is an efficient visualizer for root locus analysis and is used to locate the appropriate controller gains. Also MATLAB's "rltool", can find the n_d and n_i values which is used in equation 4.10 when the equations 4.11 and 4.9 are provided. Lastly, the "step" function creates the system's step response

with a given value of K_p which has been used to compare the modeled system response with the simulation response in section 5.1.1.

4.2.2. Waypoint Management

A networked control method is implemented in the docking maneuver. The acoustic communication links with relatively large time delays, requires the AUV thruster controller to have low control gains to maintain the vehicle's stability. The non-linearity is caused by the limited available thrust of the motors. The errors can be decreased by maintaining the error signal small, through providing interim waypoints for the navigation generated on-the-fly. The waypoints lay on the approach trajectory between the vehicle and the docking station at predetermined intervals. They are then used as the reference point for the controller.

As mentioned previously, the location of the docking station is sent to the vehicle within each position update message from the docking station, as the reference position. However, giving a far away reference position can cause the controller to be less stable by increasing the position error, considering the output limitation of the thrusters. Especially in relatively long distances (of more than 50 meters), the position error becomes drastically large, which causes the controller to be unstable, in some cases, even with low control gains. To solve this problem, a waypoint management method has been applied.

The main idea of the applied waypoint management method is to create a new waypoint at each position update message. The waypoint is determined by creating a vector from the vehicle's center point to the reference position, and selecting the waypoint to lie on this vector, at 2m distance from the vehicle center point. This creates waypoints which are 2 meters away from the vehicle's center, laying on the direction of the reference point, as shown in Figure 4.4. The calculation has been made in x, y, z dimensions and only the x component is shown in eq. 4.13:

$$x_{wp} = x_v + (2 * (x_{ref} - x_v)) / r \quad (4.13)$$

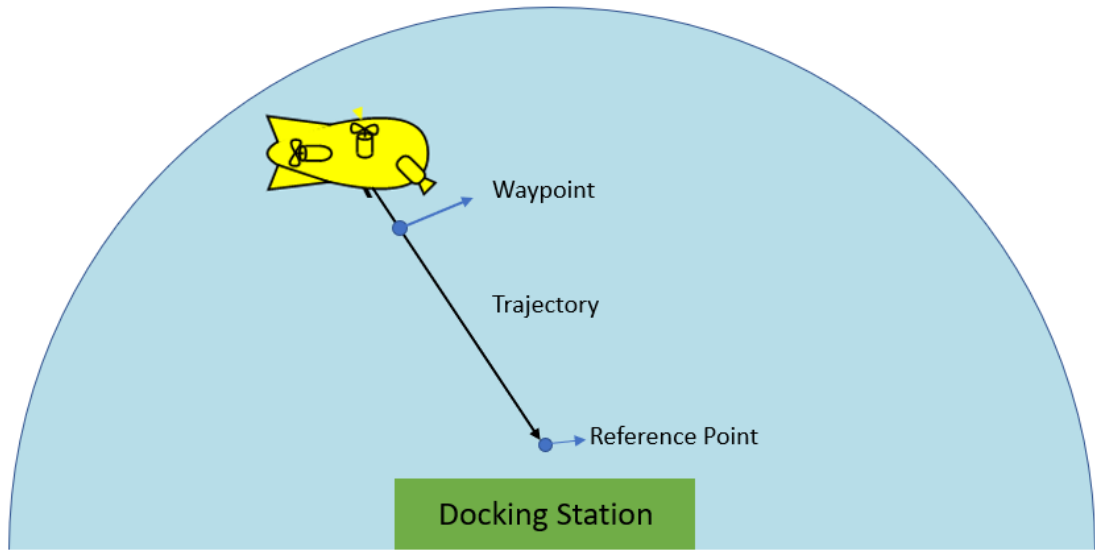


Figure 4.4: Waypoint Management.

Where x_{wp} is the x coordinate of the waypoint, x_v is the x coordinate of the vehicle, x_{ref} is the x coordinate of the reference and r is the distance between reference point and the vehicle position. The Euclidean distance is used in the calculations.

Before applying this method, the vehicle was unstable at relatively long distances (more than 50 meters). It could be observed that, the vehicle was shaking due to high controller gains because of the relatively large position errors. The waypoint management method limited the errors to not exceed a boundary value, which enabled the AUV thruster controller to perform in a consistent manner, both in short and long distances. After applying the waypoint management, it could be seen that the vehicle was stable, and the shaking motion was ceased, at relatively long distances. The waypoint management is not applied in RF mode, since the RF mode only works on short range (less than 10 meters), the waypoint management is not required for stability.

4.2.3. Heading Reference Calculation

The heading reference calculation is required to make the motion of the vehicle more natural and avoid skidding. Since the vehicle movement will be from its current position to the reference point, it is a good practice for the vehicle to head towards the reference direction. To accomplish this, the heading reference is calculated as:

$$\psi_{ref} = \arctan\left(\frac{y_{ref} - y}{x_{ref} - x}\right) \quad (4.14)$$

Where ψ_{ref} , y_{ref} and x_{ref} are the reference values for heading, y and x axes, respectively. Also y and x represent the current position of the vehicle in terms of y and x axes, in that order.

4.3. Networked Control Architecture

In this section, how the physical components of the problem map into the different simulators will be shown, as well as the description of the components of the networked control system. An accurate depiction of the NCS that has been built in this thesis can be seen in Figure 4.5. It also shows which parts of the system are simulated in which simulation environment in the co-simulation.

The AUV vehicle consists of the physical component (vehicle body, thrusters etc., subject to hydrodynamic forces) and the electronics; the real-time computer, the communication network interface system (acoustic and RF transmitters and receivers) and the position control algorithm (which is handled by the real-time computer). It is shown in purple in Fig. 4.5, labeled as “AUV”. The docking station side on the other hand, consists of the real-time computer, the position sensor and the communication network interface system. It is shown in green in Fig. 4.5, labeled “Docking Station”.

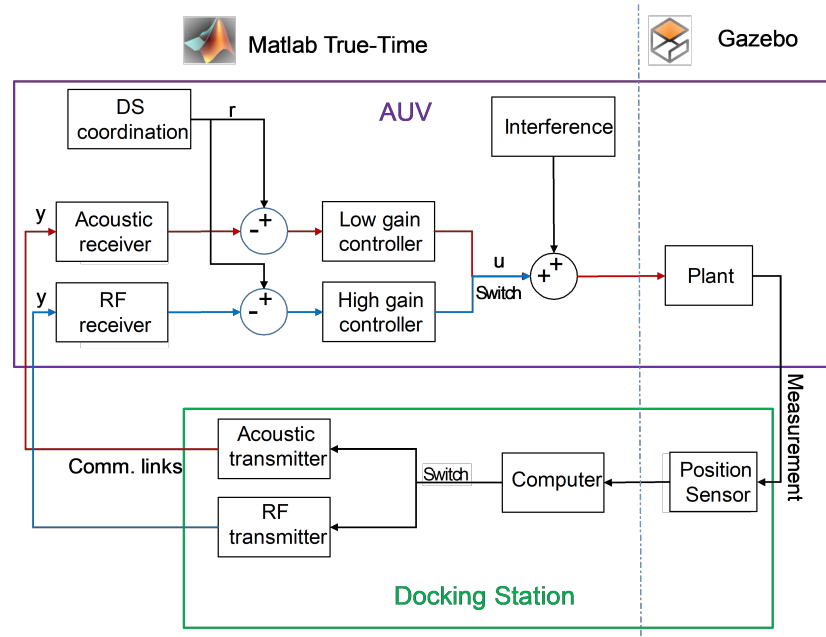


Figure 4.5: Block diagram of AUV and docking station navigation systems. The acoustic and RF communication links are shown in red and blue respectively.

Different parts of the two entities are implemented in different simulation environments depending on their specialization. The Physical component of the AUV and the distance measurement part of the docking station are implemented in Gazebo, whereas the remaining components are implemented in MATLAB TrueTime.

The position measurement of the AUVs in the vicinity are done by the docking station. Then the docking station computer decides whether to send the measurement data by acoustic transmitter or RF transmitter. Based on the decision, the AUV's acoustic or RF receiver, receive the transmitted packet. If the packet has been received by the acoustic receiver, the controller uses the low gains to control the vehicle, and if the packet has been received by the RF receiver the controller uses the high gains to apply the control. Since the position data is measured by the docking station, sent to the vehicle and received by the vehicle via the communication channel, the system is defined as a networked control system (NCS).

4.4. Communication Protocols

It is necessary to arbitrate the data transmissions by the docking station and the AUVs'. Several protocols are possible. As an example, we have selected Time Division Multiple Access (TDMA) protocol in acoustic communication and Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol in RF communication. In the first part we will explain the communication protocol which we used for data transmission. After that we will describe different packet types which we designed for the data and control transmission in the system.

4.4.1. Medium Access Scheme for Acoustic Mode

For the acoustic mode transmission, the frame time is divided into two parts, where the first period is for the broadcast downstream channel from the docking station to the AUVs, and the second period is for the upstream channel shared by the AUVs, for which we propose to implement TDMA protocol, described next.

For the control system to implement docking maneuver successfully, the docking station needs to send the location information to all AUVs reliably, hence the collision-free TDMA scheme has been chosen for the downstream acoustic channel. In this period, the docking station broadcasts AUVs' locations to all the AUVs sequentially.

There is no need to include propagation time in slot time as packets are sent one after the other from same source eliminating any chance of collision. The AUVs communicate with the docking station over the upstream channel, only when they have to send a "power control message" as explained in section 4.4.4. The AUVs transmit these messages in the AUV portion of the frame as can be seen in Figure 4.6.

4.4.1.1. Time-Division Multiple Access. In this scheme, the acoustic upstream channel, i.e., AUV period is allocated separately to each AUV via TDMA, so they can send packets to the docking station in a collision free fashion, as shown in Figure 4.6. The

time slot length is selected to include the propagation delay at the beginning of each slot to avoid collisions. Total slot time is equal to the sum of propagation delay and packet transmission time. Here the number of slots $N = V_{MAX}$.

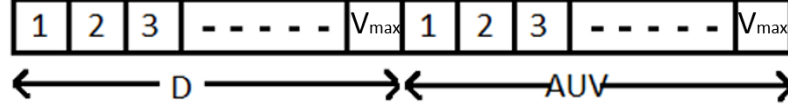


Figure 4.6: TDMA in both downstream and upstream periods

4.4.1.2. Frame Time for Acoustic Mode. The acoustic communication takes a long time due to the acoustic propagation delays. Since the docking maneuver is implemented as a networked control system, the communication delay must be considered as a source of phase delay and incorporated in the design of the control loop. In this section, the time duration of the communication frame where the position of several AUVs at the same time can be relayed, is calculated so that it can be incorporated in the control design as delay.

The frame times for AUVs and Docking Station have been selected to support a maximum of V_{MAX} vehicles. To calculate the docking station's frame time, we should consider its message length. The length of the messages of the docking station are M_{DS} bits long and the acoustic modem data rate is R_{AC} . The slot time T_{SL} , is then calculated as $T_{SL} = M_{DS}/R_{AC}$. The total docking station frame time T_{FDS} , is then the slot time multiplied by the maximum number of vehicles; $T_{FDS} = T_{SL}V_{MAX}$. In this case, we take: $V_{MAX} = 8$, $M_{DS} = 512bits$, $R_{AC} = 10kbps$ and therefore the total frame time for docking station becomes $T_{FDS} = 0.4096s$.

To calculate the slot times and total frame time for AUVs; T_{FAUV} , we again consider their message length. The message length of each vehicle is M_V bits. However, for the AUVs, we also should consider the propagation delay of the underwater acoustic wave which is calculated by dividing the average distance to the docking station x_{AVG} by the speed of underwater acoustic wave v_{AC} . The slot time for each vehicle is then

calculated as: $T_{SV} = M_V/R_{AC} + x_{AVG}/v_{AC}$. This must be multiplied by the number of maximum vehicles, V_{MAX} . In this case we take $M_V = 64bits$, $x_{AVG} = 50m$, $v_{AC} = 1500m/s$, the total frame time for the AUVs is then calculated as $T_{FAUV} = 0.3152s$.

We have calculated the total frame times of Docking Station and the AUVs, which are $T_{FDS} = 0.4096s$ and $T_{FAUV} = 0.3152s$, respectively. Then the total frame time of the MAC Protocol, and messaging period then, is the sum of the two; $T_F = T_{FDS} + T_{FAUV}$ which is $T_F = 0.7248s$. This value is appropriate, considering only the data rate of the acoustic communication. However, since this value also determines the sampling time of the control loop T_S , it is desirable that it is also a multiple of 0.02s; the orientation controller period, to ensure that there is no delay caused by the mismatch of the periods of the controller and the messages. Since the calculated value of T_F is already minimum value, we decided to round it up to the closest multiple of 0.02, which is 0.74s. Therefore in the simulations we take $T_F=0.74s$.

To be able to round up the frame time, the AUV slot times have been increased by adding a guard time, which can be considered as a safety margin. The time difference is divided by V_{MAX} and added to T_{SV} , so that now $T_{SV} = 0.0413s$. The total AUV frame time becomes $T_{FAUV} = 0.3304s$. Then the total frame time of the MAC protocol becomes $T_F = 0.74s$ as desired. In the simulations, the described timing was implemented on the TrueTime Toolbox using custom code written as part of the AUV control software.

4.4.2. Medium Access Scheme for RF Mode

Since the RF channel has high data rate and small propagation delay, we have chosen to use the CSMA/CD type access for both downstream and upstream channels in RF mode. In CSMA/CD, the node with packet to send first senses the medium and sends only if it is free. If the network is busy by another node, the node with a packet to send waits for a random amount of time and tries again. For downstream channel, the docking station starts sending its messages through the communication channel via CSMA/CD periodically, every 0.04 seconds. For the AUV transmissions,

a small number of AUVs share the upstream channel via CSMA/CD. Since only the AUV nodes that are within RF range of the docking station operate in RF mode (hence at the late part of the docking maneuver), the number of contending AUVs is typically only one. Furthermore, since the data rate is high, packet transmission times are small. Due to both facts, the probability of collisions is low, which makes CSMA/CD efficient during RF mode.

4.4.3. Packet Structure

The packets carry the control information between the multiple AUVs in the vicinity and the docking station, for implementing the docking maneuver by the underwater control system. The structure of the broadcast packet, for all protocols, sent by the docking station is shown in Figure 4.7. The packet length is 512 bits, including the following fields:

- Receiver ID: The messages originating from the docking station are broadcast containing the ID number of the intended recipient. The AUV with the matching ID uses the contents for position control. Note that, all AUVs can hear the packets intended for other AUVs, and hence they are aware of the location of each other. This way, AUVs can avoid physically colliding with other AUVs. This field is 16 bits long.
- Position (x, y, z): The current position of the AUV which is measured by the docking station's USBL. The field is 112 bits long.
- Reference Position (x, y, z): The position of the docking station. This field is 112 bits long.
- Time stamp: The time which is recorded before transmitting the message. This field is 64 bits long.

The position field contains the location information for the AUVs, obtained by the underwater positioning system. This information is used by the AUVs to calculate the control signal for its thrusters. Figure 4.8 shows the structure of the AUV packet, sent from the AUVs to the docking station. This packet is 64 bits long, involving the

Receiver ID	Position (x, y, z)	Reference Position (x, y, z)	Time stamp
--------------------	---------------------------	-------------------------------------	-------------------

Figure 4.7: Packet structure from the docking station (downstream) packets

node ID and data fields. The message carries the “power control” information, which contains the transmission power level in the data field. AUVs recognize packets from other AUVs based on packet size and discard them, as they only transmit and receive packets from the docking station. Note that, the packet structures are the same in

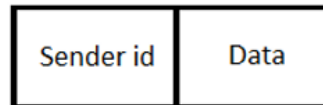


Figure 4.8: Packet structure for AUV messages

both RF and acoustic modes, but packet transmission durations are different because of different data rates supported by RF and acoustic modes.

4.4.4. Adaptive Power Control

The same transmission power control mechanism is employed in both acoustic and RF communication modes. The docking station sends the first packet with a predefined initial transmission power. The signal at the receiver is attenuated due to path loss and fading in the channel. The receiving node calculates the received power and compares it to a predefined threshold level. If the received power level is below the threshold level, it does not reach the AUV. If the AUV does not receive a message by next sampling time, it calculates a new transmission power level to compensate for the path loss and sends this power level to the docking station. The docking station, upon receiving the power control message, increases its transmission power so that reliable packet reception is guaranteed.

5. EXPERIMENTS AND RESULTS

5.1. EXPERIMENTAL DESIGN

The proposed method has been implemented in both UUVSim and TrueTime environments using the appropriate tools provided by each. In TrueTime, a Simulink model of the real-time computer, communication network and control method was created. The firmware for the real-time computer was written, and the controller parameters were adjusted for discrete time operation with the appropriate sampling time. The connection with UUVSim was prepared using the ROS API of MATLAB to draw the simulation variables into the TrueTime model. In the UUVSim side, the simulation time parameters were adjusted and a world model as described before was created containing the AUV and docking station.

5.1.1. AUV Thruster Controller

As explained in Section 4.2, the chosen method for AUV thruster control is the PID controller. Also the parameter selection method is explained in the same chapter. To create the transfer function of the AUV, three different equations should be identified, these are the controller, the plant and the delay functions.

To create the plant equation, model data should be extracted from Gazebo, since the dynamics of the vehicle is handled in Gazebo simulator. The vehicle model that is used in this thesis is RexROV, and the characterization variables are taken from [32]. The added mass matrix is given as:

$$\begin{bmatrix} 779.79 & -6.8773 & -103.32 & 8.5426 & -165.54 & -7.8033 \\ -6.8773 & 1222 & 51.29 & 409.44 & -5.8488 & 62.726 \\ -103.32 & 51.29 & 3659.9 & 6.1112 & -386.42 & 10.774 \\ 8.5426 & 409.44 & 6.1112 & 534.9 & -10.027 & 21.019 \\ -165.54 & -5.8488 & -386.42 & -10.027 & 842.69 & -1.1162 \\ -7.8033 & 62.726 & 10.775 & 21.019 & -1.1162 & 224.32 \end{bmatrix}$$

These values represent the cross coupling in x , y , z axes and, roll(ϕ), pitch(θ), yaw(ψ), respectively. Since the off diagonal elements in this matrix are comparatively small and PID control has difficulty in incorporating such cross coupling effects, it is desired to have a simpler model, the system is simplified by omitting them, which means that cross coupling is neglected.

Also, the linear and quadratic damping vectors are also available in [32], which are used to create the plant model. Since we are creating a linear model, only the linear damping vector has been taken into account. The linear damping vector is given as [74.82, 69.48, 728.4, 268.8, 309.77, 105].

For our first model, the Z dimension has been chosen. The plant variables from eq. 4.9 are a and b. In the case of Z dimension, a is 3659.9 and b is 728.4.

The next equation that has to be created is the delay equation. Our delay is 0.74 for the acoustic control, and 0.04 for the RF control. For analyzing this, the plant's time constant must be observed and compared to the delay of the controller. The time constant can be found from the characteristic equation of the plant; in this case as $\tau_z = 728.4/3659.9 \approx 0.2s$. The comparison of 0.74s, which is the highest expected communication delay, with the natural response time gives us that, the first order Padé approximation can be used, which we found in eq. 4.11. We set $T_d = 0.74s$ in the equation and find our delay function.

After obtaining the overall control transfer function, we can find the controller characteristics. In this thesis, the MATLAB's "rltool" and "rlocus" functions have

been used to obtain PID controller equation as mentioned in Section 4.2. Appropriate PID gains for damping factor $\zeta = 0.7$ and time constant $t_r = 1s$ were selected (The pole at $s = -0.02$ was attempted to be cancelled by a matching zero). The resulting values for $K_p = 995$, $n_d = 2.65$ (the ratio between K_p and K_d), and $n_i = 0.0094$ (the ratio between K_p and K_i), respectively, as in (4.10). With this controller, “rlocus” function provides Figure 5.1.

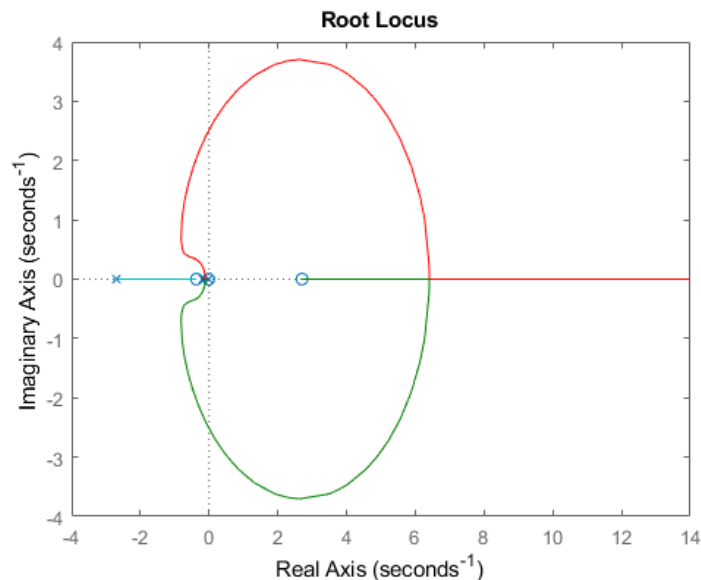


Figure 5.1: Root locus analysis of the simplified linear model.

The next step is to compare the step responses of the model and the simulation environment with the same controller. The model’s step response can be acquired by using the “step” function of MATLAB. Also, the discretized version of the same controller is applied to the co-simulation environment, and 1 meter reference response is observed and plotted. The comparison of the two plots can be seen in Fig. 5.2.

It can be observed that the simplified and linearized model does not mimic the simulation environment. Thus, it can be concluded that this model cannot be used for calculating the gains of the controller for this experiment. The reason for this can be explained by the nonlinearities in the physics simulator, Gazebo. The effect of the correlations in the added mass matrix and quadratic elements in the simulation are

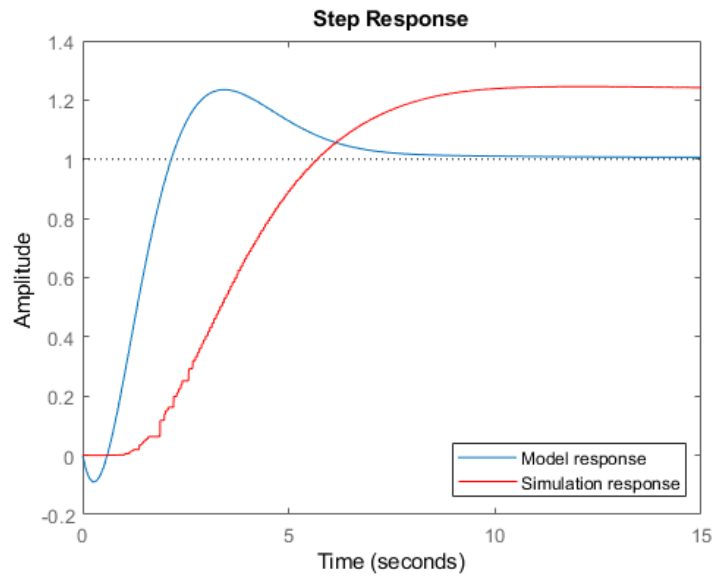


Figure 5.2: Step responses of Model versus Simulation.

also causing the nonlinearities.

The proposed solution is to tune the PID controller gains by trial and error. By observing the motion response of the vehicle while changing the PID coefficients, acceptable controllers for all axes were determined. The values that have been chosen with the experimental method are given in Section 5.2.

5.1.2. Tasks Performed within the AUV Real-Time Computer

In acoustic mode, to increase the efficiency of the orientation control of the AUV, an orientation controller has been developed with a lower period. With a separate orientation controller that operates faster than the acoustic thruster controller, the AUV becomes more stable in terms of orientation. It is safe to assume that the orientation controller can perform faster because the orientation information can be measured inside the AUV using inertial sensors. The thruster controller for the acoustic mode has a period of 0.74 seconds based on the message arrival period to the AUV from the docking station. The orientation controller has been implemented with a period of 0.02

seconds.

The tasks implemented within the real-time computer module of the AUV will be described next. True-Time provides a real-time kernel, which simulates actual code written in the 'C' language. The execution time for each code segment is specified by the designer. Each vehicle has a kernel that facilitates 3 tasks. These tasks handle the communication and the AUV thruster control. The tasks are named as the acoustic task, orientation task and RF task.

The acoustic task is an asynchronous task, which means it is called as a response to an event. The event that calls the acoustic task in this implementation is the arrival of a message packet through the simulated acoustic communication channel. When such a message is received by the acoustic task, it first checks if the message has been sent to the task's corresponding node. If the message is not for this node, then it terminates. And if the message is for this node, it checks the transmission power and decides whether to accept the message or not, based on the acoustic path loss model that has been explained in Section 2.2.1. If the message is accepted, the task updates the vehicle's internal position variables and calculates the AUV thruster forces with the designed PID controller with coefficients selected for acoustic mode as explained in this section. In this task, the thruster forces are not applied to the vehicle directly, instead, the forces are passed to the orientation controller to be used in the thruster manager message as stated above. If the message is not accepted, a new transmission power is calculated and sent as a message to the docking station, to be used by the docking station as the next message's transmission power based on the acoustic path loss model.

The RF task is also an asynchronous task, performing a similar function. The event that calls this task is the message arrival through the simulated RF communication channel to the vehicle from the docking station. This task first checks the transmission power of the arrived message and decides whether to accept the message or not based on RF path loss model that has been explained in Section 2.2.2. If the message is accepted, then both the position and orientation control forces and torques

are calculated by the PID controller with coefficients for the RF mode. If the arrived communication packet is not accepted, the RF task recalculates a new transmission power based on the RF path loss model and sends this information to the docking station.

Finally, the orientation task is a synchronous task with a fixed period of 0.02 seconds. Orientation task is responsible for calculating the control torques for orientation and applying the force and torque vector to the vehicle by sending an input message to the thruster manager of the physics simulator at the Gazebo side. The force values come to this task from the acoustic task, and this task only functions if the vehicle is in acoustic mode.

5.1.3. Tasks Performed within the Docking Station Real-Time Computer

The docking station's real-time computer contains two different tasks. As have been mentioned in 4.3, the real-time computer takes the measurement messages from Gazebo, then decides whether to send this information via RF network or acoustic network to the vehicle, based on the distance of the target vehicle. If the packet is decided to be sent by the acoustic network, the acoustic transmission task is called and if the packet is decided to be sent by RF network, the RF transmission task is called.

The acoustic transmission task works periodically. It first handles all the incoming packets from the vehicles and adjust the transmission power of their next packet that will be transmitted based on the information taken with the incoming packet. After adjusting the transmission powers, all vehicle position data is read from the Gazebo messages. The waypoint management is also applied here, since the docking station also sends the reference positions to the vehicles, the waypoint is created and sent from this node. From the position and reference point data, the transmission packet is then prepared and sent to the vehicles in an order. There is no priority among the vehicles, the packets are sent in an orderly manner. The transmission times are also handled in this node, each frame time is fixed and the docking station waits for the beginning of the frame time to send the next message.

The RF transmission task also works periodically. It first handles the incoming packets from the vehicles in the RF region and applies the power management scheme as explained in section 4.4.4. It then adjusts the power levels of next transmission, creates the packet by getting the position data and the reference position from Gazebo. The waypoint management is not applied in RF mode. After the packet is constructed, it is sent through the RF network.

5.1.4. Application of the Calculated Force to the AUV

To apply a force to the vehicle from MATLAB, there are two available methods. One of them is to use the “applyforce” function of MATLAB’s ROS API. This function applies the forces and torques to any object in the Gazebo world, for the given duration of time. The second method is using the thruster managers in UUVSim that have been developed for the vehicles, as explained in section 2.3.3. In this thesis, the chosen method for applying the force is using the thruster manager. There are two reasons for this decision. The first reason is, “applyforce” function takes relatively long time to complete, which slows down TrueTime simulator and limits the maximum call rate at 25Hz. The second reason is, “applyforce” function is an imaginary force with no physical counterpart, whereas using the thruster manager is a more realistic method, since the vehicle navigates using the thrusters in the real world, and each vehicle will have a thruster configuration implemented for it in UUVSim.

In UUVSim, the thruster manager applies the forces and torques in the body frame. Therefore, the controller forces should be calculated in the body frame. However, Gazebo provides the position data of an AUV in world frame. An example of the body frame and world frame can be seen in Figure 5.3. The black lines represent the world frame X, Y and Z dimensions, where the red lines represent the body frame X, Y and Z dimensions. Thus, a transformation must be made from world frame to the body frame.

The orientation data are given in body frame of the vehicle, however, the format of orientation data is quaternion, to be able to calculate the PID output torques we

must convert the quaternion values to Euler format. The method that is used in this thesis to convert a quaternion to Euler is the MATLAB’s “quat2eul” function. The function’s default settings change quaternion dimensions ordered as WZYX into Euler dimensions ordered as XYZ. We then calculate the orientation torques by applying the PID controller to the XYZ Euler variables.

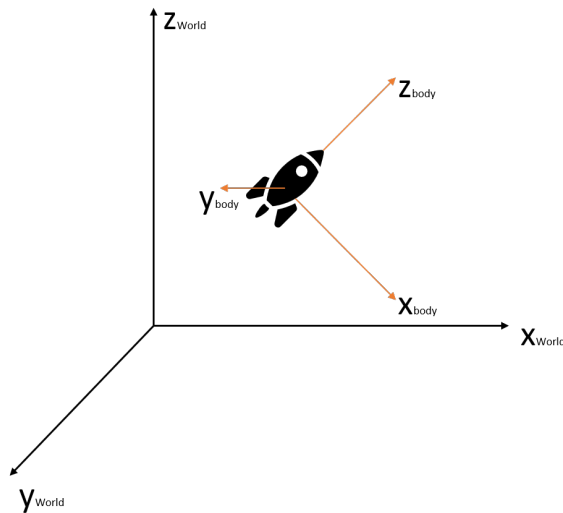


Figure 5.3: World Frame and Body Frame.

The UUVSim thruster manager as mentioned in Section 2.3.4, takes a ROS message as input at fixed intervals. However, the thruster manager takes only the newest input into consideration. As the AUV have two different controllers in acoustic mode, namely orientation controller, and position controller, both controllers would be operating at the same time. Then, while applying the forces and torques, they would overwrite each other’s messages. To solve this, only the orientation controller, which has the shorter period, has been selected to apply the forces and torques. The position controller in the acoustic mode solely updates the forces that the orientation controller sends periodically. The same case also happens in the RF mode, however since the RF control has sufficiently low period, there is no requirement of additional orientation controller in RF mode.

5.1.5. TrueTime Model for Experiments

In this section, the TrueTime, model which the experiments have been performed on, will be explained. All the results given in the section 5.2 have been taken by using this system.

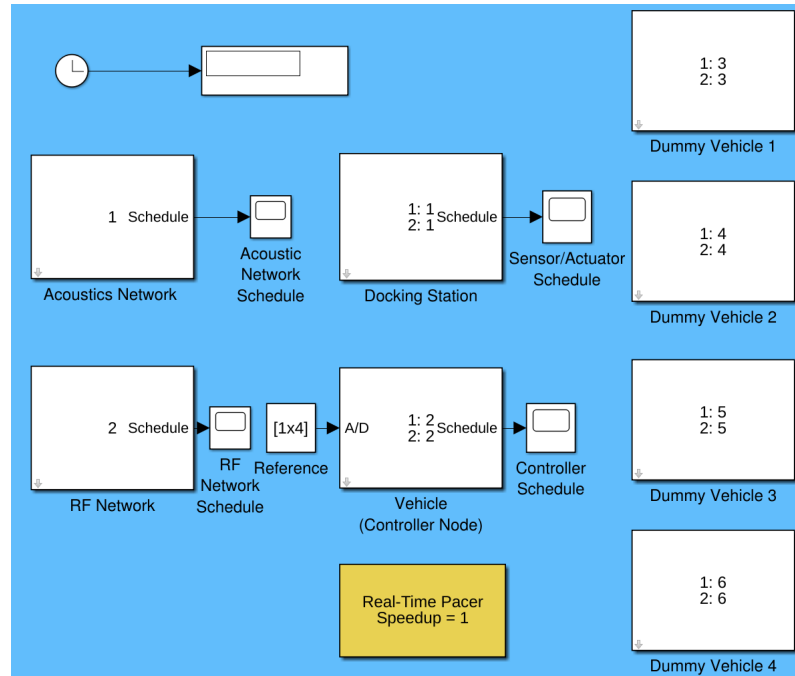


Figure 5.4: TrueTime model used in the Experiments.

In Figure 5.4, it can be seen that there are additional 4 dummy vehicle nodes inside the system. Dummy vehicle nodes, in this system, only work in acoustic mode, since there can be only one vehicle that can perform the docking maneuver at a time, and the other vehicles would not enter the RF range while a docking maneuver is happening. The dummy nodes are only used to create a traffic to exercise the MAC protocol. The dummy nodes, keep their fixed position, and create an acoustic communication load in the network. Since the packet accepting is based on the path loss model and the path loss contains a random distribution from the Rayleigh fading, as has been explained in section 2.2.1, the nodes will send a packet through the acoustic network based on the random distribution. The nodes labeled as “Docking Station” and “Vehicle (Controller Node)” work as explained previously in this section. The

node labeled as “Real-Time Pacer” is the modified version that has been created in this thesis, and works as explained in section 4.1.1.

5.1.6. Implementation of TDMA

There are two available ways to implement the TDMA protocol for acoustic mode. The first method is to use the TrueTime Network block with TDMA option. In this case, we would indeed have a TDMA protocol, however, since the TrueTime Network block does not take the propagation delay into consideration, the packet transfer times does not match the designed values. This mismatch causes our control to be unpredictable, since the period of our controller would be different for different total frame time (as explained in section 4.4) Thus, the proposed method is to create a custom TDMA protocol by using the TrueTime Network block’s Frequency Division Multiple Access (FDMA) option, and handling the packet transmission timing in the real-time computer nodes of both AUVs and Docking Station by extra coding.

For RF mode, the CSMA protocol has been implemented by using the TrueTime Network block’s CSMA/CD option, since in RF mode there is no significant propagation delay to take into consideration. This situation makes the TrueTime Network block calculations accurate.

5.2. Simulation Results

5.2.1. Control Delay Characterization

Since it is not possible to obtain zero time skew between the two simulation environments, the difference must be characterized to see its effects on the simulation, e.g. control system, and thus validate the results. To measure the time difference between two simulation environments, the Gazebo “/clock” message is read and stored in a Gazebo clock variable, and the TrueTime current simulation time is subtracted from it. The resulting value is the time difference between the times of the two simulation environments. This value is then statistically analyzed to see the variance. Ideally it

should be constant with no variance.

Figure 5.5 shows an example of the time difference between the two simulation environments in one second. The horizontal axis is the time (the graph time span is 1 simulation-second) and the vertical line is the time difference between Gazebo simulation time and Simulink simulation time. The mean value of the graph is not important in the simulations, and only the variance has an effect.

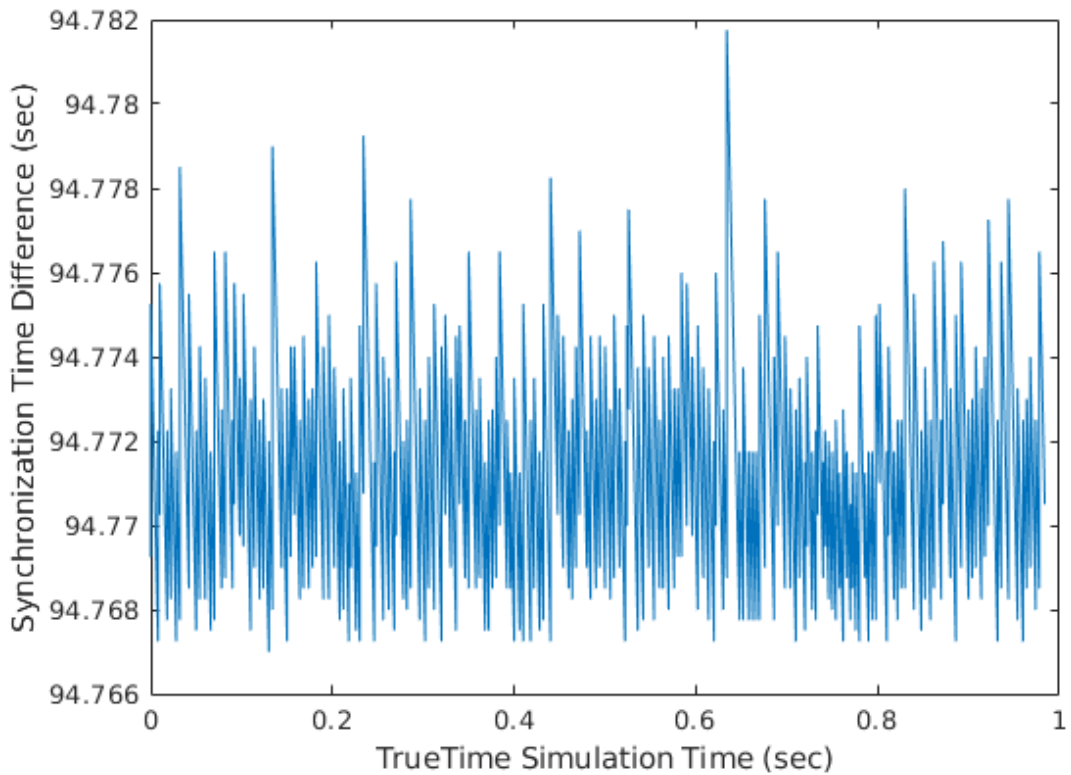


Figure 5.5: Time difference between Gazebo and Simulink simulation clocks.

To get the characterization results of synchronization from the Simulink, we have sampled the simulation time skews over 50000 sample points and calculated the standard deviation. The target value for twice the standard deviation was chosen as 10% of the control system sampling period, or the standard deviation should be less than 5% of the sampling period. This value is expected to have minimal effect on the controller behavior. Since the controller runs at 25Hz at RF, the targeted synchronization standard deviation must be less than 0.002 seconds.

Table 5.1: Standard Deviation with respect to Real-Time Update Rate and Maximum Step Size

	Real time update rate [Hz]	Max step size [s]	Actual Real Time Factor	Synchronization Standard Deviation [s]
1	2000	0.00050	0.61	0.0026
2	2500	0.00040	0.49	0.0023
3	4000	0.00025	0.31	0.0013
4	5000	0.00020	0.24	0.0013
5	8000	0.00013	0.15	0.0010

The performance of the computer that this test was carried out will also affect these values, since the time differences arise from the multicore processors, and Gazebo’s Real Time Factor which has been explained in section 2.3.1 depends on the CPU that the simulation environments run on. The specifications of the computer are presented next.

- Processor: Intel Core i7-8700K CPU @ 3.7GHz x 12
- Graphics: GeForceGTX 1060 3GB/PCIe/SSE2
- Memory: 16 Gb
- Operating System: Ubuntu 16.04.4 LTS 64-bit

The results can be seen in Table 5.1. By increasing the real time update rate while maintaining the calculated real time factor, the actual real time factor decreases which causes the Gazebo simulation to slow down. However, our synchronization standard deviation also decreases which means the synchronization performance is more stable. By adjusting the real time update rate and maximum step size variables in Gazebo, we have achieved a more consistent synchronization performance.

The Gazebo “/clock” messages are sent at the same rate as Real time update rate, and the synchronization function in the Simulink is at 500 function calls per second due to its period limit. The synchronization function calls per second must be

a multiple of Gazebo “/clock” message rate, since if there happens to be a difference, the synchronization would have a delay by the amount of a “/clock” message period. This is the reason why the Real Time update rates are chosen as a multiple of 500. Also, the synchronization function rate cannot be increased, because the function becomes inoperative if the rate increases.

Finally, the values used for the relevant variables are: Real time update rate is 4000 Hz and maximum step size is 0.00025 seconds. The actual real time factor becomes 0.31 for these values, which brings the standard deviation to 0.0013s and twice the standard deviation to 0.0026s which is not expected to significantly change the control system characteristics.

5.2.2. Experiment Results

In the simulations, freshwater parameters are used for the communication channel. For acoustic modem, the parameters of EvoLogics S2C R48/78 [10], and for RF modem, WFS seatooth® S300 [16] underwater RF modem were emulated. For motive energy calculations, parameters of NeuMotors 1925-3Y [33] were assumed.

The docking station is positioned at $[0,0,-99.825]$. The starting position of the AUV for the tests is $[-20,20,-75]$. There are a total of 5 vehicles in the simulation, 4 of them are the dummy vehicles and the fifth is the AUV that performs the docking maneuver. The dummy vehicles are started at 50 meter away from the docking station and they keep their position.

The other simulation parameters used are represented in Table 5.2. In order to simulate the physical underwater channels, the path loss models given in Section 2.2.1 and Section 2.2.2 for acoustic and RF links have been implemented. Table 5.3 shows the parameters that has been used for calculating the path loss characteristics. After path loss, Rayleigh fading is applied to obtain the instantaneous received power level.

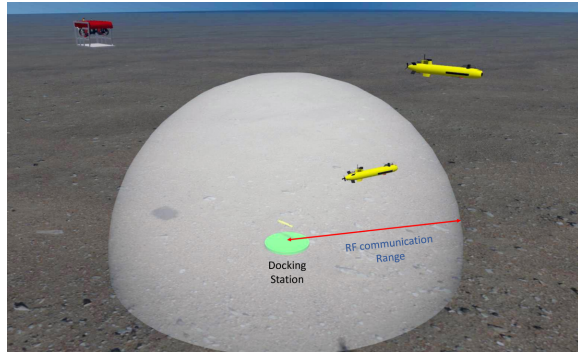


Figure 5.6: A representative screen-shot of the simulation environment in Gazebo.

Table 5.2: Simulation Parameters.

Parameters	Acoustic	RF
Frequency	100 KHz	10 MHz
Data rate	10 Kbps	3 Mbps
Frame time	0.74 s	0.04 s
Signal power limit	4.5 W	3 W
Circuit power P_c	1.1 W	4.5 W
DS Packet time t_{ds}	$3.84 * 10^{-2}$ s	$0.28 * 10^{-4}$ s
Sampling period T_s	0.74 s	0.04 s
Proportional gain $K_p[x, y, z]$	[75, 75, 225]	[155,155,455]
Integral gain $K_i[x, y, z]$	[10, 10, 10]	[195,195,195]
Derivative gain $K_d[x, y, z]$	[65,65,55]	[270,270,55]
Derivative time T_p	0.74	0.04

Table 5.3: Communication Channel Parameters.

Parameters	Acoustic
Acoustic Threshold	0.0019 W
Spreading Factor k	1.5
Max. No. of AUVs	8
RF Threshold	0.002 W
Permittivity ϵ	$80 * (8.854 * 10^{-12})$
Permeability μ	$4 * \pi * 10^{-7}$ s
Conductivity σ	0.01 S/m

5.2.3. Approach to the Docking Station in Still Water

The docking maneuver of an AUV to the base station is depicted in Figure 5.6. The goal is to reach the docking station quickly with no overshoot and small steady state error despite physics of the water. The final docking must be completed using other means such as mechanical guides. The distance of the AUV to the docking station is shown in Figure 5.7; a typical docking maneuver. The results for both conventional acoustic and proposed hybrid methods are shown. Up to about 95 seconds into the simulation both perform the same since they are both working only on acoustic communication links. From about 95 seconds onwards, RF communication is established in the proposed hybrid method, with higher gains providing a faster approach to the docking station. It can be observed that the docking is accomplished faster. It can be concluded that using RF links in close range give faster and steadier control. Time to dock, which denotes the time to reach 2% of the target distance, is reached at the end of each line, since when docking is complete the simulation stops. It demonstrates that the AUV using the proposed hybrid acoustic and RF networked control method can reach the docking station after 110 seconds which is shorter than acoustic-only method which takes around 138s; a 20% improvement.

In Figure 5.8, the motive power of thrusters with respect to time is shown. The motive power is calculated by multiplying the motor thrust with the velocity of AUV.

The thrust is calculated by multiplying the control signal by a constant, which is derived from the power of the thrusters and is given as $F_m = u/1.17$, where u is the control signal which is divided by a typical constant relating the thrust force to the control signal. Then the motive power is found by $P_m = F_m v$ where v is the velocity of AUV. This gives the instantaneous power. The motive energy can also be calculated by integrating motive power over time.

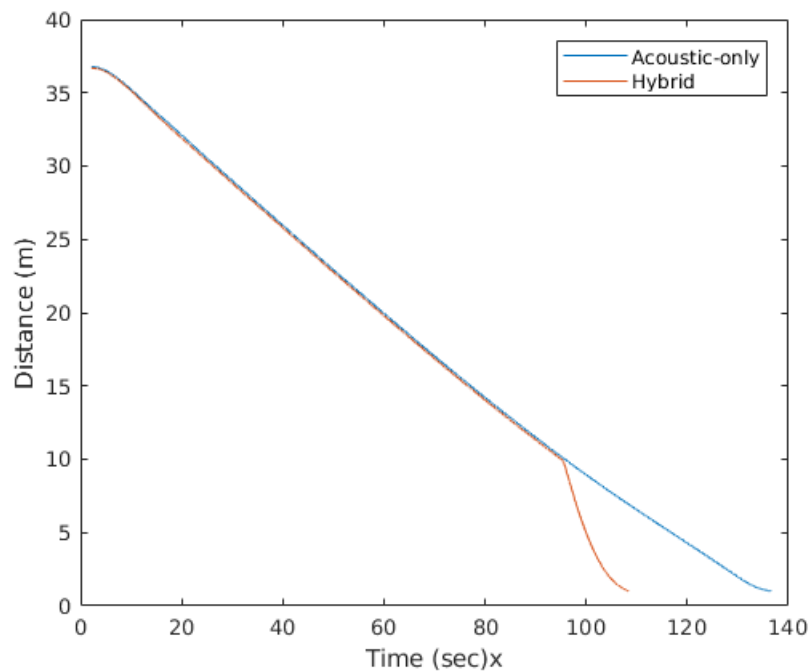


Figure 5.7: AUV time to dock for acoustic only and hybrid.

It can be seen by the low control effort in Figure 5.8 that initially the low gain controller is used during acoustic communication region. When AUV reaches the threshold distance from the docking station however, the proposed controller switches to the high gain controller which is getting position feedback via the RF communication link. The high gain controller affords faster response and higher control signals, but at the expense of higher motive power, the AUV approaches the docking station quickly and more precisely.

5.2.4. Docking Maneuvre with Water Currents

On our simulation platform, we have performed experiments to verify the accuracy of docking maneuver of the proposed hybrid networked control system in comparison to the acoustic-only system under the more realistic condition of water currents. From the results shown below, we infer that the proposed hybrid system outperforms the acoustic-only at the expense of higher motive energy. At the beginning, for both controllers, the motive power has almost the same values until the vehicle is in RF range. When the vehicle enters the RF range, motive power rises due to high forces because of the high control gains.

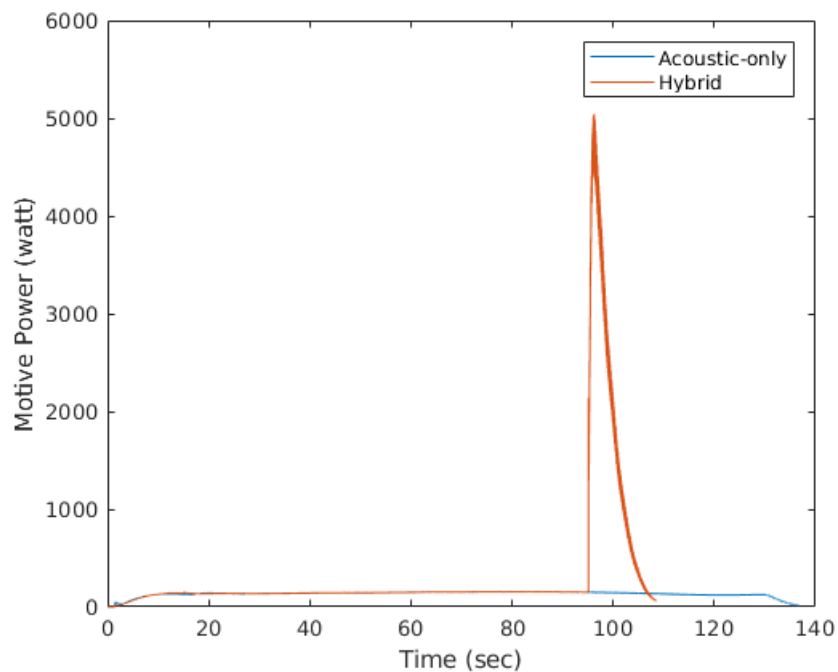


Figure 5.8: Motive power for acoustic only and proposed hybrid.

Figure 5.9 shows the performance with respect to different water current velocities. The current applied to the vehicle in these experiments were all head currents. The reason for selecting the head current is to consider the worst-case conditions. The results for the Figure 5.9 and Figure 5.10 were obtained by getting the average of 10 experiment results each.

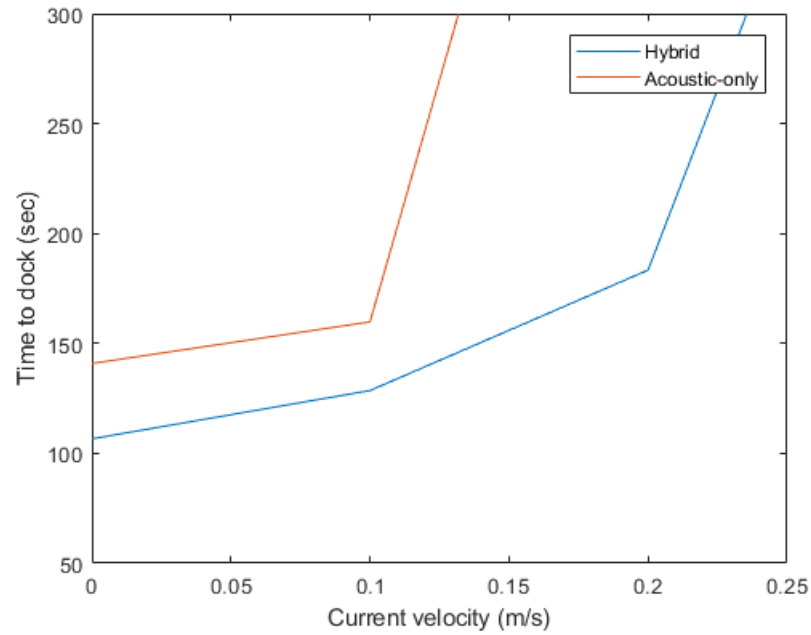


Figure 5.9: Time to dock w.r.t. different current velocities.

It can be seen in Fig.5.9 that the time to dock is consistently shorter for the proposed method under different water current strengths. Also, beyond a water current velocity exceeding 0.2m/s, the acoustic only method cannot accomplish the docking maneuver within an acceptable time period. Another measure of the performance of the maneuver is the time integral of the position error. It can be seen in Fig.5.10 that although the two methods have similar performances, at higher water current velocities, the proposed method performs better.

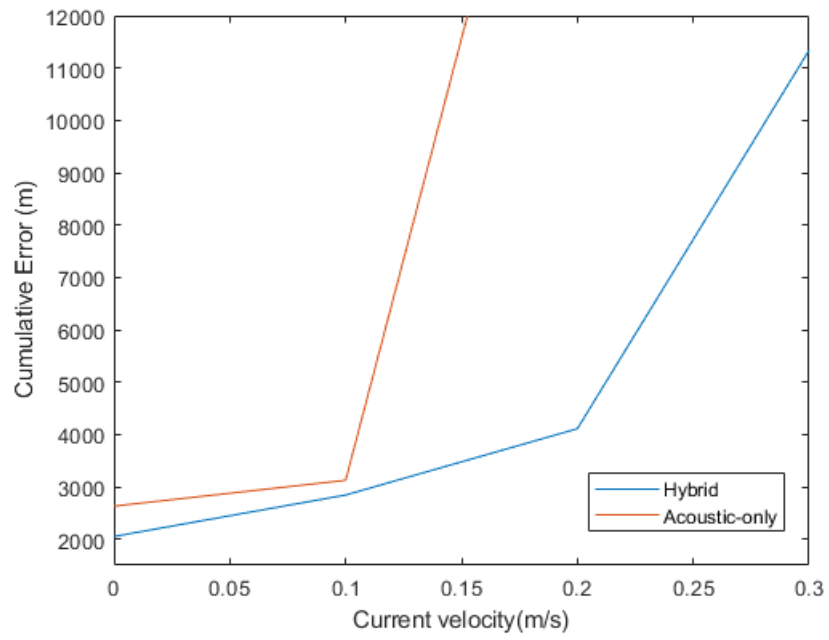


Figure 5.10: Cumulative error w.r.t different current velocities.

6. CONCLUSION

In this work, we have presented an integrated underwater co-simulation environment of an acoustic and RF hybrid networked control method [24] in a real-life scenario. This co-simulation environment containing AUVs with online communication networks and real-time computers has allowed us to study the effects of physics of hydrodynamics on AUV movement in the realistic underwater simulation environment. Using two simulation environments is beneficial for the precision of the results. However, variable time step solvers create time skew between the simulation environment which must be characterized and corrected. In this thesis we have proposed a method for time synchronization.

We have demonstrated results in the co-simulation environment and showed that the proposed networked control based on acoustic and RF hybrid communication is advantageous compared to the traditional acoustic-only method for the control of underwater AUVs. We have observed that under calm water conditions, the hybrid system takes about 20% less time to dock. Furthermore, the performance of the proposed hybrid method under disturbances in the form of water currents with different strengths was studied in the co-simulation environment. Adding the water currents to the co-simulation demonstrates that the proposed method performs better in completing the docking maneuver within a short time, whereas the conventional method fails to dock even at moderate current velocities.

We are currently working on the implementation of slotted ALOHA and waiting room MAC protocols to enrich and improve the performance of the proposed method.

REFERENCES

1. Rodríguez-Molina, J., B. Martínez, S. Bilbao and T. Martín-Wanton, “Maritime Data Transfer Protocol (MDTP): A Proposal for a Data Transmission Protocol in Resource-Constrained Underwater Environments Involving Cyber-Physical Systems”, *Sensors*, Vol. 17, No. 6, p. 1330, 2017.
2. Li, N., J.-F. Martínez, J. M. Meneses Chaus and M. Eckert, “A Survey on Underwater Acoustic Sensor Network Routing Protocols”, *Sensors*, Vol. 16, No. 3, 2016.
3. Climent, S., A. Sanchez, J. V. Capella, N. Meratnia and J. J. Serrano, “Underwater Acoustic Wireless Sensor Networks: Advances and Future Trends in Physical, MAC and Routing Layers”, *Sensors*, Vol. 14, No. 1, pp. 795–833, 2014, <http://www.mdpi.com/1424-8220/14/1/795>.
4. Chitre, M., S. Shahabudeen and M. Stojanovic, “Underwater acoustic communications and networking: Recent advances and future challenges”, *Marine technology society journal*, Vol. 42, No. 1, pp. 103–116, 2008.
5. Manhães, M. M. M., S. A. Scherer, M. Voss, L. R. Douat and T. Rauschenbach, “UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation”, *OCEANS 2016 MTS/IEEE Monterey*, IEEE, sep 2016, <https://doi.org/10.1109/Oceans.2016.7761080>.
6. Cervin, A., D. Henriksson, B. Lincoln, J. Eker and K.-E. Årzén, “How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime”, *Control Systems, IEEE*, Vol. 23, pp. 16 – 30, 07 2003.
7. Gomes, C., C. Thule, D. Broman, P. G. Larsen and H. Vangheluwe, “Co-simulation: State of the art”, *CoRR*, Vol. abs/1702.00686, 2017, <http://arxiv.org/abs/1702.00686>.

8. Al-Hammouri, A. T., M. S. Branicky and V. Liberatore, “Co-simulation Tools for Networked Control Systems”, M. Egerstedt and B. Mishra (Editors), *Hybrid Systems: Computation and Control*, pp. 16–29, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
9. Heimlich, O., R. Sailer and L. Budzisz, “NMLab: A Co-simulation Framework for Matlab and NS-2”, *2010 Second International Conference on Advances in System Simulation*, pp. 152–157, Aug 2010.
10. “S2CR 48/78 Product Information”, http://media.wix.com/ugd/417e9a_732e6502a96b43bb96257f0ed6692be2.pdf, accessed: 2018-07-20.
11. Burrowes, G. and J. Y. Khan, “Short-range underwater acoustic communication networks”, *Autonomous Underwater Vehicles*, InTech, 2011.
12. Urick, R. J., *Principles of underwater sound for engineers*, Tata McGraw-Hill Education, 1967.
13. Chen, K., M. Ma, E. Cheng, F. Yuan and W. Su, “A Survey on MAC Protocols for Underwater Wireless Sensor Networks”, *IEEE Communications Surveys Tutorials*, Vol. 16, No. 3, pp. 1433–1447, Third 2014.
14. Sozer, E. M., M. Stojanovic and J. G. Proakis, “Underwater acoustic networks”, *IEEE Journal of Oceanic Engineering*, Vol. 25, No. 1, pp. 72–83, Jan 2000.
15. Che, X., I. Wells, G. Dickers, P. Kear and X. Gong, “Re-evaluation of RF electromagnetic communication in underwater sensor networks”, *IEEE Communications Magazine*, Vol. 48, No. 12, pp. 143–151, 2010.
16. “Wfs seetooth s300”, <http://www.wfs-tech.com/>, accessed: 2018-07-20.
17. Hattab, G., M. El-Tarhuni, M. Al-Ali, T. Joudeh and N. Qaddoumi, “An underwater wireless sensor network with realistic radio frequency path loss model”,

- International Journal of Distributed Sensor Networks*, Vol. 9, No. 3, p. 508708, 2013.
18. Che, X., I. Wells, G. Dickers and P. Kear, “TDMA frame design for a prototype underwater RF communication network”, *Elsevier Journal of Ad Hoc Networks*, Vol. 10, No. 3, pp. 317–327, Jul 2011.
 19. Yunus, F., S. H. S. Ariffin and Y. Zahedi, “A Survey of Existing Medium Access Control (MAC) for Underwater Wireless Sensor Network (UWSN)”, *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, pp. 544–549, May 2010.
 20. Li, J., M. Toulgoat, M. Déziel, F. R. Yu and S. Perras, “Propagation Modeling and MAC-layer Performance in EM-based Underwater Sensor Networks”, *Proceedings of the Fourth ACM International Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, DIVANet '14, pp. 111–117, ACM, New York, NY, USA, 2014, <http://doi.acm.org/10.1145/2656346.2656359>.
 21. Xie, P., Z. Zhou, Z. Peng, H. Yan, T. Hu, J. Cui, Z. Shi, Y. Fei and S. Zhou, “Aqua-Sim: An NS-2 based simulator for underwater sensor networks”, *OCEANS 2009*, pp. 1–7, Oct 2009.
 22. Koenig, N. and A. Howard, “Design and use paradigms for Gazebo, an open-source multi-robot simulator”, *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3, pp. 2149–2154 vol.3, Sept 2004.
 23. Quigley, M., K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, “ROS: an open-source Robot Operating System”, *ICRA Workshop on Open Source Software*, 2009.
 24. Soomro, M., S. N. Azar, O. Gurbuz and A. Onat, “Work-in-Progress: Networked Control of Autonomous Underwater Vehicles with Acoustic and Radio Frequency

- Hybrid Communication”, *2017 IEEE Real-Time Systems Symposium (RTSS)*, pp. 366–368, Dec 2017.
25. Soomro, M., *Communication and Control of Autonomous Underwater Vehicles using Radio Frequency-Acoustic Hybrid MAC Schemes*, Master’s Thesis, Sabanci University, Turkey, 2017.
 26. Azar, S. N., O. Erdemir, G. Cetin, O. Gurbuz and A. Onat, “Co-Simulation of Networked Control System of Autonomous Underwater Vehicles Using Hybrid Communication”, *Proc. EMRA 2018*, 2018.
 27. Miettinen, T., *Synchronized Cooperative Simulation: OPC UA Based Approach; Synkronoitu yhteissimulointi: OPC UA -pohjainen ratkaisu*, G2 pro gradu, diplomityö, Aalto University, 2012, <http://urn.fi/URN:NBN:fi:aalto-201209213138>.
 28. “Real-Time Pacer for Simulink”, <https://www.mathworks.com/matlabcentral/fileexchange/29107-real-time-pacer-for-simulink>, accessed: 2018-07-20.
 29. Åström, K. J. and B. Wittenmark, *Computer-controlled systems: theory and design*, Courier Corporation, 2013.
 30. Lloret, J., S. Sendra, M. Ardid and J. J. Rodrigues, “Underwater wireless sensor communications in the 2.4 GHz ISM frequency band”, *Sensors*, Vol. 12, No. 4, pp. 4237–4264, 2012.
 31. Ozbay, H., *Introduction to feedback control theory*, CRC Press, 1999.
 32. Berg, V., *Development and Commissioning of a DP system for ROV SF 30k*, Master’s Thesis, Institutt for marin teknikk, 2012.
 33. “NeuMotors 1925-3Y”, <http://neumotors.com/1900-series-electric-motor-neumotors/>, accessed: 2018-07-20.

APPENDIX A: MODIFIED REAL-TIME PACER CODE

```

function msfun_realtime_pacer(block)
% Help for Writing Level -2 M - File S - Functions :
% web ([ docroot '/' toolbox / simulink / sfg / f7 -67622. html '])
% http :// www . mathworks . com / access / helpdesk / help /
% toolbox / simulink / sfg / f7 -67622. html
% Copyright 2009, The MathWorks , Inc .
global gazebo_clock %% THIS IS THE GAZEBO CLOCK MESSAGE
% instance variables
mySimTimePerRealTime = 1;
myRealTimeBaseline = 0;
mySimulationTimeBaseline = 0;
myResetBaseline = true;
myTotalBurnedTime = 0;
myNumUpdates = 0;
setup(block);
%% -----
function setup(block)
% Register the number of ports .
block.NumInputPorts = 0;
block.NumOutputPorts = 0;
% Set up the states
block.NumContStates = 0;
block.NumDworks = 0;
% Register the parameters .
block.NumDialogPrms = 1; % scale factor
block.DialogPrmsTunable = {'Nontunable'};
% Block is fixed in minor time step , i . e . ,
% it is only executed on major
% time steps . With a fixed - step solver , the
% block runs at the fastest
% discrete rate .
block.SampleTimes = [0 1];
block.SetAccelRunOnTLC(true); % run block in
% interpreted mode even w / Acceleration
% methods called during update diagram / compilation
.

```

```

        block.RegBlockMethod('CheckParameters', @CheckPrms);
        % methods called at run - time
        block.RegBlockMethod('Start', @Start);
        block.RegBlockMethod('Update', @Update);
        block.RegBlockMethod('SimStatusChange', @SimStatusChange);
        block.RegBlockMethod('Terminate', @Terminate);
    end
%%
function CheckPrms(block)
    try
        validateattributes(block.DialogPrm(1).Data, {'double'},{'
            real', 'scalar', '>', 0});
    catch %# ok < CTCH >
        throw(MSLException(block.BlockHandle, ...
            'Simulink:Parameters:BlkParamUndefined', ...
            'Enter a number greater than 0'));
    end
end
%%
function Start(block)
    mySimTimePerRealTime = block.DialogPrm(1).Data;
    myTotalBurnedTime = 0;
    myNumUpdates = 0;
    myResetBaseline = true;
    if strcmp(pause('query'),'off')
        fprintf('%s: Enabling MATLAB PAUSE command\n', getfullname
            (block.BlockHandle));
        pause('on');
    end
end
%%
function Update(block)
    if myResetBaseline
        myRealTimeBaseline = gazebo_clock;
        mySimulationTimeBaseline = block.CurrentTime;
        myResetBaseline = false;
    else

```

```

        if isinf(mySimTimePerRealTime)
            return;
        end
        elapsedRealTime = gazebo_clock - myRealTimeBaseline;
        differenceInSeconds = ((block.CurrentTime -
            mySimulationTimeBaseline) / mySimTimePerRealTime) -
            elapsedRealTime;
        if differenceInSeconds >= 0
            pause(differenceInSeconds);
            myTotalBurnedTime = myTotalBurnedTime +
                differenceInSeconds;
            myNumUpdates = myNumUpdates + 1;
        end
    end
end
end
%%
function SimStatusChange(block, status)
    if status == 0,
        % simulation paused
        fprintf('%s: Pausing real time execution of the model (
            simulation time = %g sec)\n', ...
            getfullname(block.BlockHandle), block.CurrentTime);
    elseif status == 1
        % Simulation resumed
        fprintf('%s: Continuing real time execution of the model\n
            ', ...
            getfullname(block.BlockHandle));
        myResetBaseline = true;
    end
end
end
%%
function Terminate(block)
    if myNumUpdates > 0
        fprintf('%s: Average idle real time per major time step =
            %g sec\n', ...
            getfullname(block.BlockHandle), myTotalBurnedTime /
            myNumUpdates);
    end
end

```

end
end