Bio-inspired motion control of rigidly connected systems with application to underwater robotics

Adrià Mompó Alepuz (s181366)

Master of Science in Electrical Engineering

2020

DTU Electrical Engineering
Department of Electrical Engineering

Bio-inspired motion control of rigidly connected systems with application to underwater robotics

Report written by:
Adrià Mompó Alepuz (s181366)

Advisor(s):
Roberto Galeazzi, Associate Professor at the Electrical Engineering Department of DTU
Silvia Tolu, Assistant Professor at the Electrical Engineering Department of DTU

DTU Electrical Engineering
Technical University of Denmark
2800 Kgs. Lyngby
Denmark

–elektro@elektro.dtu.dk–

Project period: 2. January- 7. August

ECTS: 35

Education: M.Sc.Eng.

Field: Electrical Engineering

Class: Public

Edition: 1. edition

Remarks: This report is submitted as partial fulfillment of the requirements for graduation in the above education at the Technical University of Denmark.

Abstract

New challenges in inspection, maintenance and repair tasks in offshore installations have given rise to an increasing demand of small autonomous and cooperative underwater vehicles that provide the required versatility to face a wide array of complex missions. An envisioned solution is to use multiple modular vehicles that can attach to each other rigidly into formations to share resources and extend their carrying or tool usage capabilities. These vehicle formations may be required to execute complex maneuvers with high accuracy to interact with the underwater installations in a precise manner.

These demanding motion capabilities can be framed as a trajectory tracking problem, which requires knowledge of the inverse dynamics of the vehicles in order to anticipate the forces to be executed by their motors to set the vehicles into a desired kinematic state. Since the dynamics model of the vehicle formations is not known, the proposed solution considers a biologically-inspired learning controller that can fit a relevant inverse dynamics model in a short period of time. This controller is based on the ideas presented in [30], and it is shown that it outperforms a classical PI controller with high gains in a set of designed test scenarios, both in simulation and with a real underwater vehicle.

For the simulation, the model of a single vehicle and the model of two vehicles based on [23] were implemented to rapidly iterate in the design process and tuning of the control algorithm, as well as to provide part of the validation of the controller capabilities. For the real vehicle validation, the control algorithm was implemented in ROS for both a single vehicle and two vehicles, although only the single-vehicle controller could be tested due to limitations out of the scope of this project.

Preface

This project corresponds to the Master of Science thesis developed in the Electrical Engineering department at the Technical University of Denmark (DTU) in order to fulfill the academical requirements to complete the MSc degree in Electrical Engineering with specialization in Automation and Robot Technology.

Acknowledgements

I would like to thank both of my supervisors Roberto Galeazzi and Silvia Tolu for providing me with the opportunity of working on this project and for guiding me throughout it. Their advice and feedback have been essential for me to learn beyond what I had imagined and for fulfilling the thesis requirements.

I also would like to thank Peter for his help and technical support provided during the ROS implementation of the controllers in the underwater vehicles.

Last but not least, to my family for supporting me morally and financially and to my home-country and international friends for making the overall experience more enjoyable and refreshing my mind from time to time.

Contents

List of Figures

List of Tables

Symbols

$\{b\}$	Single vehicle or vehicle formation reference frame
$\{n\}$	Inertial reference frame
$\{i\}$	Vehicle reference frame when it is in a formation
$\{j\}$	Thruster reference frame
$\mathcal{I}$	Set of vehicles in a formation
$\mathcal{J}$	Set of thrusters in a vehicle
N_i	Number of vehicles in a formation
N_j	Number of thrusters in a vehicle
x	Axis of inertial frame $\{n\}$, displacement of vehicle along such axis
y	Axis of inertial frame $\{n\}$, displacement of vehicle along such axis
z	Axis of inertial frame $\{n\}$, displacement of vehicle along such axis
ϕ	Rotation of the vehicle about the of x-axis of the inertial frame
θ	Rotation of the vehicle about the of y-axis of the inertial frame
ψ	Rotation of the vehicle about the of z-axis of the inertial frame
u	Velocity of the vehicle in the surge motion direction
v	Velocity of the vehicle in the sway motion direction
w	Velocity of the vehicle in the heave motion direction
p	Angular velocity of the vehicle in the roll motion direction
q	Angular velocity of the vehicle in the pitch motion direction
r	Angular velocity of the vehicle in the yaw motion direction
X	Force applied to the vehicle in the surge motion direction
Y	Force applied to the vehicle in the sway motion direction
Z	Force applied to the vehicle in the heave motion direction
K	Moment applied to the vehicle in the roll motion direction
M	Moment applied to the vehicle in the pitch motion direction
N	Moment applied to the vehicle in the yaw motion direction
$\boldsymbol{\eta}$	Vector of vehicle linear and angular displacements
$\boldsymbol{p}$	Vector of vehicle linear displacements
$\boldsymbol{\Theta}$	Vector of vehicle angular rotations
$\boldsymbol{\nu}$	Vector of vehicle linear and angular velocities
$\boldsymbol{v}$	Vector of vehicle linear velocities
$\boldsymbol{\omega}$	Vector of vehicle angular velocities

$\boldsymbol{R}_n^b(\boldsymbol{\Theta})$	Rotation Matrix of linear velocities from inertial reference frame to vehicle body reference frame
$\boldsymbol{J}_n^b(\boldsymbol{\Theta})$	Jacobian Matrix of transforming inertial angular velocities to vehicle angular velocities
$\boldsymbol{T}_n^b(\boldsymbol{\Theta})$	Transformation Matrix mapping from inertial velocities to vehicle velocities
$\boldsymbol{\tau}$	Vector of forces and moments applied to the vehicle
$\boldsymbol{\tau}_i$	Vector of forces and moments applied to a vehicle i in a formation
$\boldsymbol{\tau}_{\mathrm{F}}$	Vector of forces applied to the vehicle
$\boldsymbol{\tau}_{\mathrm{M}}$	Vector of moments applied to the vehicle
$\boldsymbol{M}$	Matrix of inertia of the system
$\boldsymbol{D}(\boldsymbol{\nu})$	Matrix of hydrodynamic damping terms
$\boldsymbol{C}(\boldsymbol{\nu})$	Matrix of Coriolis and centripetal terms
$\boldsymbol{g}(\boldsymbol{\eta})$	Vector of gravity and buoyancy terms
$\boldsymbol{M}_{\mathrm{RB}}$	Matrix of rigid-body mass and and inertia terms
$\boldsymbol{M}_{\mathrm{A}}$	Matrix of hydrodynamic added masses and and inertia terms
$\boldsymbol{\Lambda}$	Matrix of moment of inertia terms
$\boldsymbol{D}_{\mathrm{l}}$	Matrix linear hydrodynamic drag coefficients
$\boldsymbol{D}_{\mathrm{n}}(\boldsymbol{\nu})$	Matrix quadratic hydrodynamic drag terms
$\boldsymbol{\rho}_j^b$	Vector of thrust of a thruster j in the vehicle
$\boldsymbol{t}_j^b$	Unit Vector in the direction of thrust of a thruster j in the vehicle
$\boldsymbol{x}$	State Vector of the vehicle system
$\boldsymbol{x}_a$	Augmented state Vector of the vehicle system
$\boldsymbol{x}_d$	Desired state Vector of the vehicle system
$\hat{\boldsymbol{x}}$	Estimated state Vector of the vehicle system
$\hat{\boldsymbol{x}}_i$	Estimated state Vector of a vehicle i in a formation
$\boldsymbol{u}$	Total control action Vector to the vehicle system
$\boldsymbol{u}_a$	Augmented control action Vector
$\boldsymbol{u}_{\mathrm{FF}}$	Feedforward control action Vector
$\boldsymbol{u}_{\mathrm{FB}}$	Feedback control action Vector
$\boldsymbol{u}_{\mathrm{LWPR}}$	LWPR control action Vector
$\boldsymbol{u}_{\mathrm{CM}}$	Cerebellar module control action Vector
$\boldsymbol{u}_{\mathrm{CM}}^*$	Optimal cerebellar module control action Vector
$\boldsymbol{y}$	Output Vector of the vehicle system
$\boldsymbol{r}$	Reference Vector of the vehicle system
$\boldsymbol{e}$	Tracking error Vector of the vehicle system
$\boldsymbol{A}(\boldsymbol{x})$	Non-linear state dynamical Matrix
$\boldsymbol{A}_{\mathrm{l}}$	Linear component of the $\boldsymbol{A}(\boldsymbol{x})$ Matrix
$\boldsymbol{A}_{\mathrm{n}}(\boldsymbol{x})$	Non-linear component of the $\boldsymbol{A}(\boldsymbol{x})$ Matrix
$\boldsymbol{A}_a(\boldsymbol{x}_a)$	Augmented state dynamical Matrix
$\boldsymbol{A}_K$	Closed-loop state dynamical Matrix
$\boldsymbol{B}$	Input Matrix to the system
$\boldsymbol{B}_a$	Augmented input Matrix to the system

$\boldsymbol{C}$	Output Matrix of the system
$\boldsymbol{K}$	Matrix with proportional feedback control gains
$\boldsymbol{N}$	Input Matrix for the reference vector $\boldsymbol{r}$
$\boldsymbol{\Phi}$	Feeforward Matrix representing the Unit Learning Machine
$\boldsymbol{w}_{\mathrm{CM}}$	Cerebellar module weights Vector
$\boldsymbol{w}_{\mathrm{RF}}$	LWPR receptive fields weights Vector
$\boldsymbol{e}_{\mathrm{CM}}$	Cerebellar output error Vector
β	Cerebellar module learning rate
λ_f	LWPR update parameter
I_c	LWPR output confidence interval
γ_{CM}	Cerebellar module discrete-time update parameter
γ_{LWPR}	LWPR module discrete-time update parameter
γ'_{CM}	Cerebellar module continuous-time update parameter
γ'_{LWPR}	LWPR module continuous-time update parameter
$\boldsymbol{d}_e$	Tracking error disturbance Vector
$\boldsymbol{A}_e$	Tracking error state dynamical Matrix
$\boldsymbol{H}$	Thruster configuration Matrix for whole vehicle system
$\boldsymbol{H}_i$	Thruster configuration Matrix for vehicle i in the formation
$\boldsymbol{\rho}$	Thrust Vector containing the thrust of all thrusters in the vehicle system
$\boldsymbol{\rho}_i$	Thrust Vector containing the thrust of all thrusters in a vehicle i in the formation
$\rho_{i,j}$	Thrust of a single thruster j in vehicle i
$p_{i,j}$	Magnitude of the vector $\boldsymbol{p}_j^b$ for a thruster j in a given vehicle i
$\alpha_{i,j}$	Angle from the x-axis of the formation reference frame to the x-axis of the thruster j in vehicle i
$\beta_{i,j}$	Angle from the $\boldsymbol{p}_j^b$ vector to the to the x-axis of the thruster j in vehicle i
$\boldsymbol{p}_i^b$	Vector from the center of a vehicle reference frame $\{i\}$ to the center of the formation reference frame $\{b\}$
$\hat{\boldsymbol{v}}_b^b$	Estimated linear velocity Vector of the vehicle formation
$\hat{\boldsymbol{v}}_i^b$	Estimated linear velocity of a vehicle i in the formation expressed in the formation reference frame $\{b\}$
$\hat{\boldsymbol{\omega}}_b^b$	Estimated angular velocity Vector of the vehicle formation
$\hat{\boldsymbol{\omega}}_i^b$	Estimated angular velocity of a vehicle i in the formation expressed in the formation reference frame $\{b\}$

CHAPTER 1

Introduction

In recent years, new challenges have originated in the offshore operations sector, in which the the number of underwater facilities installed is growing and moving towards more remote and hostile environments. Due to the adverse conditions in the ocean, these undersea structures often require intervention in the form of inspection, maintenance and repair (IMR) tasks. These tasks are partially carried out by remotely operated underwater vehicles (ROV) of heavy configuration —*work-class* ROV—, which carry a variety of tools to perform the required work. However, many of the structures contain confined spaces which cannot be accessed by these heavy ROVs.

An envisioned solution consists on using multiple small-sized autonomous underwater vehicles (AUVs) with modular capabilities, which can cooperate and rigidly assemble into formations. Within these formations, the AUVs can distribute the workload and carry different instruments or objects with variable sizes, thus replicating the functionalities of the work-class ROVs while providing more flexibility in terms of configurability and access to tighter spaces. Additionally, these formations provide a means of fault tolerance by sharing motor and sensor resources.

The versatility that these modular AUVs provide comes with a control design cost. Essentially, since the physical structure of the formation can change in shape (vehicles arrangement) and size (number of vehicles), a vast number of control scenarios arises, each one being characterized by different system dynamics and control requirements.

In an IMR task, a vehicle may be required to precisely execute a specific maneuver or follow a path to access a specific location. In this project, it is assumed that the motion control problem to be addressed is trajectory tracking. This consists on making a set of kinematic variables of the system (usually position, velocity or acceleration) follow a reference path which is parameterized in time. For any arbitrary trajectory to be followed accurately, a dynamical model of the system needs to be known, such that the controller can provide the forces and moments required to set the system in a specific kinematic state, according to its physical properties such as mass and drag.

With this, the difficulty shows in the modeling of the system since the dynamics for each formation are different and need to be identified as accurately as possible to provide performant trajectory tracking. This extensive modeling becomes highly impractical if done with traditional system identification methods, which usually require high amounts of empirical data and a specific set of experimental setups. Some analytical approaches have been considered [23] which attempt to obtain a model of the whole system by as-

suming that the model of an individual vehicle is known and by mathematically defining the set of rigid connections among the different vehicles as motion constraints. The main disadvantage of this approach is that any mismatch between the individual model and the real vehicle will be present in all the defined structure, and that when attaching several vehicles together, new unmodeled dynamics arise due to the interaction between the fluid and the new shape.

Because of the notable impracticality and drawbacks of the aforementioned methods, a different direction in solving the control problem is to focus instead on designing self-adjusting control architectures that, given a physical reconfiguration of the formation, can gradually change their behaviour (dictated by a set of modifiable parameters) to keep performing as desired based on the data gathered in the interaction between the vehicles and the environment. The most common and traditionally used set of methods that falls into these specifications is the adaptive control paradigm [7]. Adaptive control assumes that a partially known (or at least a parameterized) plant model is available, and the main goal is to adjust the model parameters during the system operation so that it converges towards the real plant dynamics while at the same time optimizes a control objective. The main limitation of adaptive control is that, despite the flexibility provided in its parameter adaptation capabilities, the structure of these parameters is fixed and therefore it only defines a fixed set of models (linear models, non-linear with quadratic terms, etc). This implies that the model to be represented should be determined in advance, thus not allowing an arbitrary non-linear model to be captured.

The alternative set of methods that overcomes the adaptive control limitations (this is, allows for an arbitrary non-linear model representation) is that of learning controllers [7]. These may, however, require more data to be gathered before obtaining a reliable model. The biologically-inspired control methods can also be found inside this category, which are characterized for applying neuroscience principles into the control problem, either in structure, function or both.

In the following sections, the main work in the field of learning controllers applied to motion control in robotics will be reviewed, and special focus will be made on those applied to underwater autonomous vehicles.

1.1 Model Dynamics Learning

Generally, learning controllers in the scope of motion control problems aim to represent the mapping between the desired and/or measured kinematic states and the control action to be performed to achieve such states according to its physical properties properties (mass and drag). This corresponds to learning the inverse dynamics model of the plant. The opposite is also common, this is, learning the mapping between a control action and the successive states induced by it, which corresponds to the forward dynamics model of the plant. A combination of both is possible too.

An approach to learn the model of a plant is to use popular machine learning tech-

niques such as Artificial Neural Networks (ANN) and derived methods. This has been successfully applied in [31]. However, despite the ability of ANNs to represent any non-linear dynamics function, they usually require rich and vast amounts of data covering all the kinematics and control action space or, at least, the space that the robot will encounter during a typical mission. Hence, this data should be gathered in an exploration stage prior to any real mission of the vehicle, which is equivalent to performing a system identification stage with other traditional methods. If learning is done in an on-line fashion (i.e., during a mission), traditional neural networks may suffer from overfitting and catastrophic forgetting, since the network's parameters may reach saturation before exploring all the state and action space. A direction in solving this issue has been addressed in [33], which combines a statistical techique (LWPR, presented in next paragraph) with ANNs to gradually build a baseline model which does not suffer form the aforementioned ANN problems, and which is used by the ANN to learn the complete model of the plant.

Other statistical methods have been developed and employed to learn the dynamics of a system in an on-line fashion, thus overcoming the main limitations of ANNs. Among these, special emphasis should be made on Locally Weighted Projected Regression (LWPR) [32], Gaussian Process Regression (GPR) and derived methods, and Support Vector Regression (SVR). The work by J. Sun de la Cruz et al. [11][9] shows that LWPR and Sparse Online Gaussian Processes (SOGP). [10] can effectively be used learn an inverse dynamics model of a robotic arm manipulator. They also show that prior model knowledge, even if incomplete, can be introduced into these techniques to boost the performance and training speed [8].

Biologically inspired methods may combine some of the above mentioned statistical and machine learning techniques, while at the same time introducing neuroscience principles which aim to replicate some of the structures, functionalities or computations observed in the motor areas of the brain. These are assumed to be a reference in terms of control design since the brain successfully solves many complex control tasks. In many cases, bio-inspired methods successfully inherit and mimic such principles to a certain extent, giving the overall control algorithm an advantage in terms of data efficiency and control robustness. S. Tolu et al. [30] modelled a control structure with parallelisms to the cerebellum structure, which combines a LWPR module for long-term model learning and another module for short-term adaptation, while a feedback controller is used in parallel to ensure stability and assist with learning. This was applied successfully to a 3-degrees-of-freedom robot arm. Further work has been done by extending the learning module [5] and by combining it with spiking neural networks [24]. Another bio-inspired approach worth mentioning is the Cerebellar Model Articulation Controller (CMAC) [2], a perceptron-like algorithm that mimics the local learning properties of the cerebellum, which has been widely applied to many robotics tasks.

1.1.1 Dynamics Learning in Autonomous Underwater Vehicles

In the field of AUVs, and in particular for motion control and trajectory tracking, most of the techniques presented in the previous section have been successfully applied for individual vehicles. However, to the best of the author's knowledge, no learning methods have been applied to trajectory tracking in multiple-modular-AUV formations.

In the domain of ANNs, neural network-based controllers have been successfully applied in simulation in [36], outperforming lead compensators, pole cancelation techniques and adaptive sliding controllers in terms of error-tracking. In [12] robust trajectory tracking was achieved in simulation by decomposing the problem in system identification with a neural network and control by combining a feedback and an ANN-based learning controller. The work in [37] showed in simulation non-linear and coupling terms cancellation with a neural network, predictive control for accounting for input delay, and stability analysis of the proposed system. [31] presents a classification of other past implementations of ANN-based learning controllers for underwater robots. Additionally, reinforcement learning has also proven successful in trajectory tracking for AUVs [34].

In [14], a LWPR module was used to improve state estimation in AUV navigation.

Regarding the work on biologically-inspired methods, [21] combined a CMAC with a feedback controller in a simulated underwater vehicle, obtaining good performance compared to just using a feedback controller. [6] also used a CMAC network for motion control in sumulated AUVs.

1.2 Project Direction

Given the properties of the different techniques presented in previous work and the requirements of this project, the direction followed here can be established. Specifically, it is desired that the motion controller in the vehicles can learn a relevant model in few iterations and without presenting overfitting to a specific trajectory. This is critical in real-life IMR scenarios in which the vehicle formation may change during the operation to adapt to different needs, in which case the time to adapt and obtain a new model should be minimal. These requirements are incompatible with conventional ANN or RL methods which usually require an extensive period for collecting data.

Because of this, the proposed approach in this project is to design a learning controller based on cerebellar learning theory. The method will be described in detail in the following chapters, and it is inspired by the work presented in [30]. The interesting properties that it presents are on-line learning (can update the model during operation), fast learning (requires little data relative to other methods), and local learning (the model is built gradually around localized regions of the its input space, thus avoiding the overfitting problem that a traditional ANN would present if a trajectory only explored part of the space that can be represented by the input). The main drawback of the

adopted method is that the solution may be suboptimal with respect to other techinques such as RL, since the the control problem is not stated as an optimization problem.

With this, several research questions can be formulated, which will be addressed during the development of the project.

- *Can a biologically-inspired controller in the scope of AUVs learn a relevant dynamics model such that it outperforms other traditional techinques in terms of trajectory tracking?*
- *Can this controller learn a relevant model in a short enough period of time such that it does not hinder the operation of the vehicles in an IMR task?*
- *Can the learning controller be designed to be robust against external disturbances and changes in the dynamical model?*
- *Can the learning controller provide some level of fault tolerance?*

1.3 Objectives

The objective of this project is to design a biologically-inspired motion controller and implement it in the scope of modular underwater autonomous vehicles, such that it assists in the trajectory tracking of a multiple-vehicle rigid formation by learning its dynamical properties.

The project can be further divided into several sub-objectives:

- Review the state of the art in learning controllers for AUVs and choose the direction that this project will follow
- Define the control problem to be solved and the specific requirements that the control algorithm should meet
- Design a learning controller that meets the specified requirements
- Implement the controller in simulation to validate its properties and iteratively improve its design based on the results
- Design an experimental campaign to fully test and compare the controller both in simulation and in the real vehicle and facilities
- Implement the control algorithm in the Robot Operating System (ROS) framework to deploy and test it in the real vehicles

This project was affected by the safety measures adopted after the COVID-19 outbreak in spring 2020, which implied that the university campus and therefore the testing

facilities remained closed during most of the period that had been planned to carry out the experiments. This resulted in limited experimental validation activities and collected data.

1.4 Contributions

This project has made use of several already available resources that were not developed by the author. In this section a distinction will be made between these and the parts that were in fact developed for this project.

All hardware used was manufactured or acquired by the REMORA laboratory at DTU. The only exception is the personal computer from the author. Regarding the software, MATLAB and several of its libraries (for vehicle guidance, navigation and control utilities) were acquired before the start of the project. The MATLAB and C++ base implementations of the LWPR algorithm were downloaded from open source websites. The main design concept for the learning controller used was originally developed by [30]. Finally, for the ROS implementation, the libraries for communicating with the motors, joystick and motion capture system were part open-source and part previously developed by other students and staff related to the REMORA laboratory.

The contributions from the author include:

- The adaptation and implementation of the learning controller into the specific needs and software of this project,
- The dynamical analysis of the learning controller to assess its convergence and stability properties,
- The development of several additional guidance, navigation and control functions for MATLAB and C++ (ROS),
- The extension of the LWPR library (both MATLAB and C++) to meet the designed learning controller specifications
- The design of the experiments to be carried out both in simulation and in the real facilities

1.5 Thesis outline

In the next chapters, the following topics and respective contents will be presented:

Theoretical Background: In this chapter, a conceptual description of the core principles and fundamentals that lead to the adopted solution is provided. These are the the Cerebellar Learning Theory and the LWPR algorithm principles. At the end

of the chapter, these are tied together to describe the adopted solution at a conceptual level.

Mathematical Modeling: This chapter includes the mathematical description of the physical system and how it is formulated into the control problem. The main mathematical principles and the algorithmic outline of the adopted solution is also provided here, relating it to the presented control problem.

Materials and Methods: In this chapter the main software and hardware components are presented, as well as some details on how the proposed solution has been implemented.

Results: This chapter starts with the description of the designed experiments and follows with the results both from simulation and the real vehicle. A short discussion and analysis of the results is also provided for each subsection.

Discussion: In this chapter, the results are discussed at a more global level, relating them to the research questions and objectives presented in previous chapters.

Conclusion and Future Work: A general overview of the project highlighting its achievements, limitations and possible future improvements.

CHAPTER 2
Theoretical Background

This chapter presents the main theoretical concepts required to derive and understand the adopted control solution, which is based on the cerebellar learning theory and the LWPR algorithm.

2.1 Cerebellar Learning Theory

Biologically-inspired control methods rely on the known principles on which the brain processes the sensory and motor states from different regions of the brain and body, and produces the suitable motor commands for the muscles. There are several areas in the brain that are involved in such motor processes, and their overall structure follows a hierarchy that reflects different identifiable motor functions.

The main role in motor control in the brain is taken by the motor cortex, found in the cerebral cortex (as shown in red in Figure 2.1), and its functions include planning and executing voluntary movements. Additionally, there are several other parts in the brain which regulate other aspects of motion, and of special interest here is the cerebellum. This part integrates the sensorimotor state information of the organism with the objective of improving the accuracy and automating motor behaviors commanded by brain regions belonging to higher levels in the control hierarchy, such as the motor cortex.

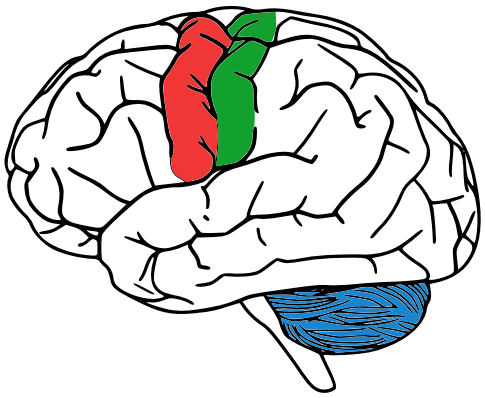

Figure 2.1: Brain with the areas involved in motion highlited. Red: motor cortex; green: sensory cortex; blue: cerebellum

The cerebellum has been suggested to be responsible for learning physical models of the world, representing both forward and inverse dynamics of different parts of the body and their interactions with objects and the environment Although the exact mechanisms by which these models are represented inside the cerebellum are not yet fully understood, a remarkable property that it presents and which may contribute to the wide range of models and functions that in can learn is its highly regular structure composed by repeated microcircuits. Each of these microcircuits have been regarded as a "neuronal machine" [13] which presents the characteristics of an adaptive filter [16]. The adaptive filter model of the cerebellar microcircuits builds upon the ideas and features identified by Marr and Albus [3][22], and is based on three key principles: analysis and decomposition of an input signal into its basic features; synthesis of these features by performing a weighted sum to produce an output; and adaptation of the weights according to a teaching signal. Additionally, these three principles have been associated with some specific regions and cells of the cerebellar microcircuit, providing thus empirical evidence for their existence and functionality.

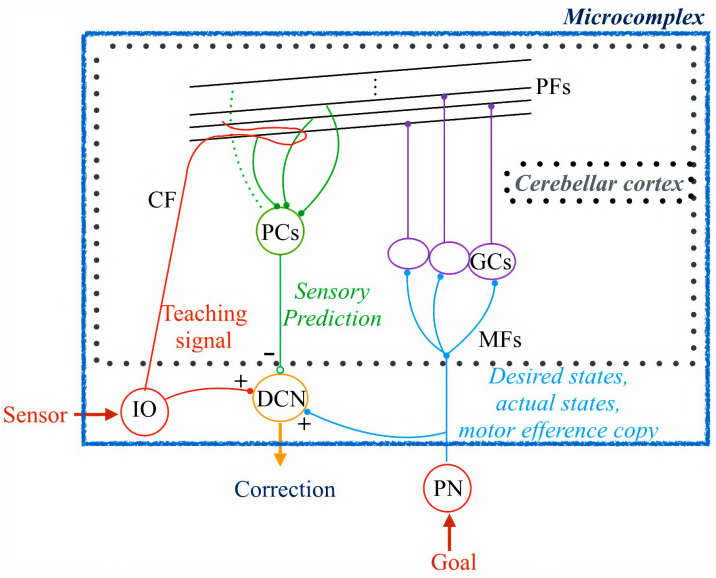

Figure 2.2: Cerebellar microcircuit. Adapted from [29]

As a brief overview, the structural anatomy of the cerebellar microcircuit can be described as follows. The input signal is carried into the circuit by the mossy fibers (MFs), which originate in the precerebellar nuclei (PN) found in the brain stem and convey sensorimotor information encoding desired and actual states, and motor efferent copies (copies of the motor output). These input signals are distributed among the granule cells (GCs), which in essence "encode" the signals into an expanded representation and transmit them through the parallel fibers (PFs). These PFs, in turn, extend the encoded signals over a portion of the cerebellar cortex, which facilitate their access to the Purkinje cells (PCs) they pass through. The PCs then integrate the information carried by tens of thousands of PFs, producing the main component of the microcircuit

output. This output is then sent to the deep cerebellar nuclei (DCN), which also takes information from other cells and produces motor corrections. Finally, the climbing fibers (CFs) coming from the inferior olive (IO) provide error signals related to movement and constitute the teaching signal that modifies synapses such as those formed by PFs-PCs, thus materializing the cerebellar learning mechanism.

It is important to note that the description of the cerebellar microcircuit provided above depicts only a simplified version of the real anatomy, since there are in fact additional cells involved in the process which play other roles and which have been omitted here, such as interneurons for memory consolidation [35].

As it can be observed, this structure reflects the three functions of the adaptive filter previously presented, and it has led to several computational models that exhibit the core properties of this circuit. The solution adopted in this project for the learning controller is based on one of these models, and will be presented in the last section of this chapter.

2.2 Locally Weighted Projected Regression

The Locally Weighted Projected Regression is an algorithm that combines several statistical techniques with the aim of approximating non-linear functions whose data may be sampled gradually. It has been successfully applied to a wide range of robotics and control tasks, both for learning forward and inverse dynamics models.

A short mathematical description of the elements that play a major role in the final adopted solution is provided here, describing only qualitatively the rest. The algorithm assumes that data is sampled from a non-liner function y which is dependent on the variable vector $\boldsymbol{x}$ and has been added zero-mean Gaussian noise ϵ.

$$y = f(\boldsymbol{x}) + \epsilon \tag{2.1}$$

The idea behind LWPR is to identify a set of linear models that locally approximate the non-linear function in a sub-region of the input space covered by $\boldsymbol{x}$. This is, for any input $\boldsymbol{x}$ located close enough to the center $\boldsymbol{x_k}$ of the k^{th} local linear model, the function can be approximated as

$$y_k = \boldsymbol{\beta_k}^\top \boldsymbol{x} + \epsilon \tag{2.2}$$

where $\boldsymbol{\beta}_k$ is the vector of parameters that defines the hyperplane corresponding to y_k. Now, formally stated, the locality of this linear model or, in other words, the degree to which an input $\boldsymbol{x}$ belongs to this local model, is determined by the expression

$$w_k = \exp\left(-\frac{1}{2}\left(\boldsymbol{x} - \boldsymbol{x_k}\right)^\top \boldsymbol{D_k}\left(\boldsymbol{x} - \boldsymbol{x_k}\right)\right) \tag{2.3}$$

where w_k is the weight that determines the degree of $\boldsymbol{x}$ belonging to the k^{th} local model, which is characterized by its center $\boldsymbol{x_k}$ and the matrix $\boldsymbol{D_k}$ which defines the shape and size of the region covered by this model. Such region is also known as receptive field (RF). The weights w_k will also be regarded as RFs activations p_k in following sections. Finally, the output $\hat{y}$ of the algorithm is obtained as

$$\hat{y}(\boldsymbol{x}) = \frac{\sum_{k=1}^{K} w_k y_k}{\sum_{k=1}^{K} w_k} \tag{2.4}$$

which corresponds to a weighted sum over all K local linear models' outputs y_k with their corresponding weights w_k. In essence, an input sample $\boldsymbol{x}$ activates all K receptive fields to a certain degree w_k and yields an output y_k for their respective linear models, which are then weighted in a way that those linear models that are closer to the input and therefore have a larger w_k will contribute the most to the final output $\hat{y}$.

The adaptive capabilities of the algorithm are achieved by updating the linear regression parameters $\boldsymbol{\beta}_k$ and the RF parameters $\boldsymbol{x_k}$ and $\boldsymbol{D_k}$ as more data gets collected. Additionally, the number of RFs are also updated dynamically according to two tunable parameters that determine when a new RF should be created and when it should be removed. Finally, a key principle of LWPR is that it assumes that for each RF, the distribution of the data for the corresponding local linear model can be represented in a lower dimensional space than the original one presented by $\boldsymbol{x}$. Thus, it attempts to reduce the dimensionality of $\boldsymbol{x}$ via Partial Least Square regression, essentially projecting the data into a lower dimensional space. This usually holds true for many robotic applications in which the many degrees of freedom and sensor variables may present several correlations, allowing for the compression of the information carried and saving in computational effort when processing the linear models. For the mathematical details of these parameter updates and dimesion reduction see the original paper [32].

In the case of wanting to represent a multiple-output function, the algorithm creates multiple single-output LWPR models that have the same input $\boldsymbol{x}$ but different target functions y.

2.3 Learning Controller, Unit Learning Machine

The learning controller (LC) used in this project can be classified as biologically-inspired, since, on the one hand it follows the three main principles of the cerebellar microcircuit adaptive filter model, namely, analysis, synthesis and adaptation. On the other hand, it replicates part of the microcircuit structure, finding analogies between the controller modules and the cerebellar cortex cells and their functions.

The controller is based on the ideas presented in [30], which aimed to design an

algorithmic analog of a simplified cerebellar circuit, based on the properties of the LWPR algorithm and other machine learning and control techniques which provide biologically-plausible components to the control structure.

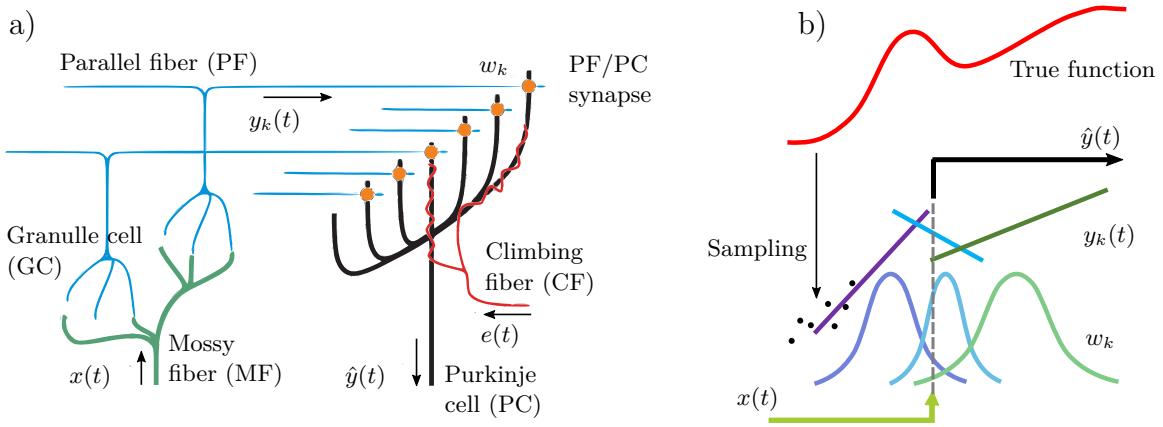

Figure 2.3: Qualitative representation comparing a cerebellar microcircuit (left, adapted from [26]) and the LWPR algorithm (right)

In control theory terms, the LC is composed by a feedback part and a feedforward part. The feedback part guarantees stability and provides a teaching signal to the learning components of the controller, as will be explained in the following paragraphs. Despite the fact that this part could be implemented with any form of feedback controller, a simple proportional velocity controller is chosen here since it satisfies the requirements for the learning algorithm in [30] and keeps the overall structure simpler and easier to tune.

The feedforward part corresponds to the adaptive component of the controller which learns the inverse dynamics of the plant. The input to this block is usually composed by a subset of the desired kinematic state (position, velocity and acceleration) and the current state, which from a biological perspective represents the previously described input signals to the cerebellar microcircuit carried by the MFs. These inputs are fed to the LWPR algorithm, which creates RFs with their respective weights (also referred to as activations) that spatially encode the signal analogously to the encoding done by the GCs in the cerebellum. The granular cell layer has been hypothesized to produce an optimal set of basis functions that enable accurate and fast learning [25]. This is replicated by the LWPR in the optimization of the size and location of its RFs. The output of the LWPR is obtained by performing a weighted sum over all linear local models, which corresponds biologically to the PF-PC synapses and the PC output that integrates the many PFs activations into a single signal.

The LWPR learns the inverse model at a rate that is not sufficient for fast adaptation against disturbances. To compensate this, the feedforward block also includes an additional component named cerebellar module (CM) which enables short term adaptation. This way, the LWPR builds a model that captures the long-term plant dynamics while the CM allows for quicker adjustments needed after unexpected changes in the model,

disturbances and other uncertainties. In terms of structure, the CM relies on the RFs created by the LWPR, and assigns them individual weights whose update rule is based on gradient-descent, as detailed in the following paragraphs and the next chapter. The output of the CM is computed by performing the inner product between the CM weights and the RFs activations. The cerebellar analogy here, similarly to the LWPR output, comes from the fact that the strength of the PFs-PC synapses can be interpreted by weights (the CM weights here) and the signals carried by the PFs can be regarded as the spatial encoding provided by the LWPR RFs activations.

The two elements composing the feedforward block (LWPR and CM) are known as the unit learning machine (ULM). The final output of the ULM is the sum of its components, namely the LWPR and the CM outputs, which represents the total feedforward action to be performed by the inverse dynamics model.

Finally, the learning mechanism is enabled by establishing the teaching signals for each module. These signals come from the motor (control action) signals in the system, just as in the cerebellum the teaching signals from the CFs carry motion-related errors. On the one hand, the teaching signal in the LWPR corresponds to the total control action sent to the plant (this is, the sum of the ULM output and feedback controller output) which is the output of the target function to be learnt by the algorithm. On the other hand, the CM updates its weights by performing gradient descent assuming that the error in its function output is provided by the feedback control action, which constitutes the teaching signal. The details of these learning mechanisms will be shown in Section 3.2.3 of the next chapter.

CHAPTER 3
Mathematical Modeling

In this chapter the control problem will be formulated making use of mathematical notation, starting from a physical description of the system to be controlled showing the relevant equations of motion, followed by a reformulation of these equations into the control problem, and concluding with the integration of the chosen learning controller.

3.1 Equations of Motion

This section describes the equations of motion of an actuated rigid body in an underwater medium. This may refer either to a single AUV or to multiple rigidly connected AUVs. The mathematical notation used here follows that presented both in [27] and [15]. The vehicle motion nomenclature corresponds to the *Society of Naval Architects and Marine Engineers* (SNAME), as shown in Table 3.1.

A series of simplifying assumptions will be introduced to reduce the modeling efforts while keeping the necessary detail for solving the control problem.

Assumption 3.1. The AUVs move in the horizontal plane at a constant depth. This reduces the degrees of freedom (DOF) of the vehicles form six to three: surge, sway and yaw.

The Assumption 3.1 will be addressed experimentally by providing active roll and pitch stabilization, as well as depth holding. The equations and variables presented in the next sub-sections will be formulated in a 3-dimensional space with all six DOF to provide a full understanding of the system, and afterwards Assumption 3.1 will be applied to focus only on the relevant DOF.

Motion description	Name	Forces and moments	Velocities	Positions
Translation along the x-axis	Surge	X	u	x
Translation along the y-axis	Sway	Y	v	y
Translation along the z-axis	Heave	Z	w	z
Rotation about the x-axis	Roll	K	p	ϕ
Rotation about the y-axis	Pitch	M	q	θ
Rotation about the z-axis	Yaw	N	r	ψ

Table 3.1: SNAME notation used.

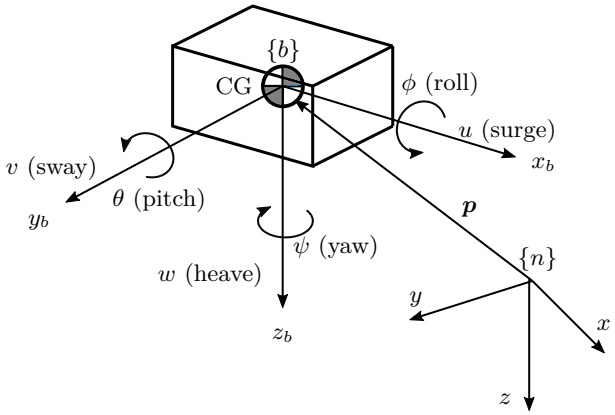

Figure 3.1: Main kinematic variables of the vehicle in a 3-dimensional environment

3.1.1 Kinematics

In order to describe the motion of the AUV in a 3-dimensional environment, two reference frames are used. One is fixed in the world, considered inertial and denoted by $\{n\}$, while the other is fixed to the vehicle and moves together with it, and is referred to as $\{b\}$. The pose of the vehicle in the world $\boldsymbol{\eta} \in \mathbb{R}^6$ is defined by a position vector $\boldsymbol{p} \in \mathbb{R}^3$ and a rotation vector $\boldsymbol{\Theta} \in \mathbb{R}^3$.

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{p}^\top & \boldsymbol{\Theta}^\top \end{bmatrix}^\top = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^\top \tag{3.1}$$

The velocity of the system $\boldsymbol{\nu} \in \mathbb{R}^{\mathbf{6}}$ is also determined by a linear component $\boldsymbol{v} \in \mathbb{R}^3$ and an angular component $\boldsymbol{\omega} \in \mathbb{R}^3$.

$$\boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{v}^\top & \boldsymbol{\omega}^\top \end{bmatrix}^\top = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^\top \tag{3.2}$$

The relationship between the body linear velocity and the time derivative of the position in the world can be determined by the mapping

$$\boldsymbol{v} = \boldsymbol{R}_n^b(\boldsymbol{\Theta})\, \dot{\boldsymbol{p}} \tag{3.3}$$

where $\boldsymbol{R}_n^b(\boldsymbol{\Theta})$ is the rotation matrix from the world frame to the body frame, and it is parameterized by the current rotation $\boldsymbol{\Theta}$

$$\boldsymbol{R}_n^b(\boldsymbol{\Theta}) = \begin{bmatrix} c_\psi c_\theta & s_\psi c_\theta & -s_\theta \\ -s_\psi c_\phi + c_\psi s_\theta s_\phi & c_\psi c_\phi + s_\psi s_\theta s_\phi & s_\phi c_\theta \\ s_\psi s_\phi + c_\psi s_\theta c_\phi & -c_\psi s_\phi + s_\psi s_\theta c_\phi & c_\phi c_\theta \end{bmatrix} \tag{3.4}$$

In the previous rotation matrix and in the following equations, c_α refers to $\cos(\alpha)$ and s_α to $\sin(\alpha)$.

The transformation from the world angular position's time-derivative to the body angular velocity is given by

$$\boldsymbol{\omega} = \boldsymbol{J}_n^b(\boldsymbol{\Theta})\, \dot{\boldsymbol{\Theta}} \tag{3.5}$$

with the Jacobian matrix $\boldsymbol{J}_n^b(\boldsymbol{\Theta})$ as a function of $\boldsymbol{\Theta}$

$$\boldsymbol{J}_n^b(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \tag{3.6}$$

Both the linear velocity and the angular velocity transformations can be expressed together as

$$\boldsymbol{\nu} = \boldsymbol{T}_n^b(\boldsymbol{\Theta})\dot{\boldsymbol{\eta}} = \begin{bmatrix} \boldsymbol{R}_n^b & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{J}_n^b(\boldsymbol{\Theta}) \end{bmatrix} \dot{\boldsymbol{\eta}} \tag{3.7}$$

3.1.2 Rigid Body Dynamics

The sum of forces and torques applied to the AUV gives rise to the vector of resultant forces $\boldsymbol{\tau}_\mathrm{F} \in \mathbb{R}^3$ and the vector of resultant moment $\boldsymbol{\tau}_\mathrm{M} \in \mathbb{R}^3$

$$\boldsymbol{\tau} = \begin{bmatrix} \boldsymbol{\tau}_\mathrm{F}{}^\top & \boldsymbol{\tau}_\mathrm{M}{}^\top \end{bmatrix}^\top = \begin{bmatrix} X & Y & Z & K & M & N \end{bmatrix}^\top \tag{3.8}$$

With this, the dynamics model of a rigid-body in a 3-dimensional aquatic environment can be described as follows

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} \tag{3.9}$$

where $\boldsymbol{M}$ is the matrix of inertia which contains both the rigid-body inertia and the hydrodynamic added mass, $\boldsymbol{D}(\boldsymbol{\nu})$ is the hydrodynamic damping matrix, the $\boldsymbol{C}(\boldsymbol{\nu})$ matrix accounts for the Coriolis and centripetal terms, and $\boldsymbol{g}(\boldsymbol{\eta})$ is the vector accounting both for the gravity and the buoyancy terms.

At this point, additional simplifying assumptions are introduced. Generally, the matrices $\boldsymbol{D}(\boldsymbol{\nu})$ and $\boldsymbol{C}(\boldsymbol{\nu})$ are dense, which implies that the motion in different degrees of freedom (DOF) will be coupled. This, in principle, does not pose a problem for a learning controller that aims to learn an approximation of the full inverse dynamics, including coupled motions. Indeed, the learning controller accounts for this, as mentioned at the end of Section 3.2.2. However, in order to make the analysis and the controller design simpler and for constituting the guidance system —as will be shown in Section 3.3.1—, Assumption 3.2 is applied. Assumptions 3.3 and 3.4 are also presented next.

Assumption 3.2. The body coordinate frame axes, the principal axes and the axes of symmetry of the vehicle are all aligned. This implies that the off-diagonal terms of $\boldsymbol{M}$ and $\boldsymbol{D}(\boldsymbol{\nu})$ are zero and that all degrees of freedom are uncoupled in motion.

This Assumption 3.2 is valid for vehicles that have three planes of symmetry and move at low velocities.

Assumption 3.3. The gravity and buoyancy force vectors have the same magnitude and the buoyancy vector acts on the center of gravity (CG) of the body. This implies that the vehicle is neutrally buoyant, that is, both the force and the moment components in $\boldsymbol{g}(\boldsymbol{\eta})$ are zero since on the one hand the gravity and buoyancy vectors cancel each other out in magnitude and on the other hand no moment is generated since the vectors act on the same point.

Assumption 3.4. The vehicle rotates at low speeds, thus making the Coriolis and centripetal effects negligible and the matrix $\boldsymbol{C}(\boldsymbol{\nu})$ is set to be zero.

The Assumption 3.3 will be ensured experimentally by adjusting the ballasts and flotation devices on the AUVs as well as by providing active depth holding. With Assumptions 3.3 and 3.4 in place, the system is simplified as

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3.10}$$

where $\boldsymbol{M}$ and $\boldsymbol{D}(\boldsymbol{\nu})$ are diagonal matrices according to Assumption 3.2. The matrix $\boldsymbol{M}$, as mentioned before, is composed by the rigid-body mass and the hydrodynamic added mass. This can be expressed as the sum of two matrices, $\boldsymbol{M}_{\mathrm{RB}}$ and $\boldsymbol{M}_{\mathrm{A}}$ respectively.

$$\boldsymbol{M} = \boldsymbol{M}_{\mathrm{RB}} + \boldsymbol{M}_{\mathrm{A}} \tag{3.11}$$

The $\boldsymbol{M}_{\mathrm{RB}}$ matrix is composed by the vehicle mass m and the moment of inertia terms

$$\boldsymbol{M}_{\mathrm{RB}} = \begin{bmatrix} m\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{\Lambda} \end{bmatrix} \tag{3.12}$$

where $\boldsymbol{I}_{3\times3}$ is the 3×3 identity matrix, and $\boldsymbol{\Lambda}$ is the moment of inertia tensor which is diagonal in this case due to assumption 3.2

$$\boldsymbol{\Lambda} = \mathrm{diag}\left(I_{xx},\ I_{yy},\ I_{zz}\right) \tag{3.13}$$

where I_{xx}, I_{yy}, and I_{zz} are the moments of inertia for the x, y and z axes of the vehicle respectively. The matrix of $\boldsymbol{M}_{\mathrm{A}}$ is defined as

$$\boldsymbol{M}_{\mathrm{A}} = \mathrm{diag}\left(-X_{\dot{u}},\ -Y_{\dot{v}},\ -Z_{\dot{w}},\ -K_{\dot{p}},\ -M_{\dot{q}},\ -N_{\dot{r}}\right) \tag{3.14}$$

where each term of the diagonal is the partial derivative of the force applied in a certain DOF direction with respect to the acceleration experienced in that direction. For example, for the surge direction this is

$$X_{\dot{u}} = \frac{\partial X}{\partial \dot{u}} \tag{3.15}$$

The matrix $\boldsymbol{D}(\boldsymbol{\nu})$ is also composed by the sum of two diagonal matrices

$$\boldsymbol{D}(\boldsymbol{\nu}) = \boldsymbol{D}_{\mathrm{l}} + \boldsymbol{D}_{\mathrm{n}}(\boldsymbol{\nu}) \tag{3.16}$$

one accounting for the linear drag coefficients

$$\boldsymbol{D}_{\mathrm{l}} = \mathrm{diag}\left(X_u,\ Y_v,\ Z_w,\ K_p,\ M_q,\ N_r\right) \tag{3.17}$$

and another for the quadratic drag terms

$$\boldsymbol{D}_{\mathrm{n}}(\boldsymbol{\nu}) = \mathrm{diag}\left(X_{u|u|}|u|,\ Y_{v|v|}|v|,\ Z_{w|w|}|w|,\ K_{p|p|}|p|,\ M_{q|q|}|q|,\ N_{r|r|}|r|\right) \tag{3.18}$$

where the terms X_u and $X_{u|u|}$ for surge (and similarly for all the other DOF) are found with the partial derivatives analogously to Equation 3.15, but taking the derivative with respect to u and $u|u|$ respectively.

From here on, applying Assumption 3.1, Equation 3.10 will refer to the system with the 3-DOF motion in surge, sway and yaw. This is, only the first two entries and the last entry in the diagonal matrices and in the vectors will be considered.

Thrusters

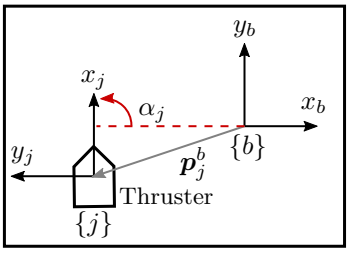

Figure 3.2: Generic 2-dimensional representation of the vehicle frame with a thruster

The single-vehicle motion is controlled by a set of thrusters distributed across its body, in a way that they provide full 6-DOF control, this is, the system is *just-actuated*. However, considering Assumption 3.1, only those thrusters involved in the 3-DOF motion control at a constant-depth plane will be considered here. Each thruster is assigned a reference frame $\{j\}$ that is rotated and/or displaced a certain magnitude with respect to the vehicle reference frame $\{b\}$, and the x-axis of the thruster frame is aligned with the thrust direction. The z-axis of both $\{j\}$ and $\{b\}$ are aligned. This way, the thrust vector $\boldsymbol{\rho}_j^b$ of a motor j expressed in $\{b\}$ is

$$\boldsymbol{\rho}_j^b = \boldsymbol{t}_j^b \rho_j = \begin{bmatrix} \cos(\alpha_j) \\ \sin(\alpha_j) \\ 0 \end{bmatrix} \rho_j \tag{3.19}$$

where $\boldsymbol{t}_j^b$ represents x-axis unit vector of $\{j\}$ expressed in $\{b\}$, α_j is the angle from $\{b\}$-x-axis to $\{j\}$-x-axis and ρ_j is the scalar value of the thrust. These thrust vectors act at point displaced from the center of the vehicle frame

$$\boldsymbol{p}_j^b = \begin{bmatrix} x_j \\ y_j \\ 0 \end{bmatrix} \tag{3.20}$$

where x_j and y_j are the displacements in surge and sway respectively of the j motor with respect to the vehicle frame center. With all this, the total thrust applied to the vehicle body is obtained as

$$\boldsymbol{\tau} = \sum_{j \in \mathcal{J}} \boldsymbol{h}\left(\boldsymbol{\rho}_j^b, \boldsymbol{p}_j^b\right) = \sum_{j \in \mathcal{J}} \begin{bmatrix} \boldsymbol{\rho}_j^b \\ \boldsymbol{p}_j^b \times \boldsymbol{\rho}_j^b \end{bmatrix} \tag{3.21}$$

where $\mathcal{J}$ represents the set of thrusters attached to the vehicle.

3.1.3 Multiple vehicles

For multiple vehicles, the following assumption is made:

Assumption 3.5. The links attaching together the vehicles that compose the multiple-vehicle system are considered rigid, and the vehicles themselves remain behaving as rigid bodies. This implies that the overall structure is treated dynamically as a single rigid body.

With Assumption 3.5 in place, all the kinematic and dynamic equations presented in the previous sections remain valid. Some parameters will of course change to adapt to the new physical properties of the system and to properly meet the previous assumptions. These changes are:

- The coordinate frame $\{b\}$ is now placed at the center of gravity of the whole structure, which coincides with the symmetry center and all axis are aligned, according to Assumption 3.2.
- The individual vehicle's reference frames are now designated as $\{i\}$ which are located at the center of gravity of each corresponding vehicle.
- The physical parameters accounting for the inertia and the drag of the system are now different, which implies that the coefficients of the matrices $\boldsymbol{M}$ and $\boldsymbol{D}(\boldsymbol{\nu})$ have changed.

Additionally, the system is now *overactuated*, since there exist redundancies in the control action space generated by the thrusters to achieve any particular motion. With these clarifications, from here on the terms vehicle, vehicle system and vehicles will refer to the same set of systems obeying the equations of motion presented in the previous sections, whether it be a single vehicle or a formation of several rigidly-connected vehicles.

3.2 Control Problem

As mentioned in previous chapters, the control problem to be addressed is trajectory tracking applied to the vehicle system. This corresponds to controlling a set kinematic

variables which describe the motion of the system CG, or equivalently the center body frame center $\{b\}$. These variables are usually either the position (if a path in space has to be followed over time) or velocity (if a certain velocity profile needs to be achieved) in any degree of freedom. Based on the definition presented in [1], it can be formally stated as:

Definition 3.1. *Trajectory tracking.* Let $y_d(t) : [0, \infty) \to \mathbb{R}$ be a trajectory reference to be followed by the kinematic variable $y(t)$ of a dynamical system. $y_d(t)$ is smoothly defined in time and its time-derivative is bounded. The controller of this system can be designed to keep all signals bounded while achieving an arbitrarily small tracking error $\|y_d(t) - y(t)\|$.

In this project the chosen variable to be controlled is the velocity and the solution will be designed for it. As implied in Assumption 3.2, the DOF of the system are not coupled in motion, which means that the trajectory tracking problem can be addressed for each DOF indepentently.

3.2.1 State-Space Formulation

Based on the dynamics equations developed in Section 3.1.2 of this chapter and the trajectory tracking problem just defined, a state-space formulation of the system can be developed. The vector of variables $\boldsymbol{x}$ that, together with its time derivative $\dot{\boldsymbol{x}}$, is sufficient to describe the dynamical state of the system, corresponds the velocity vector $\boldsymbol{\nu}$ of the vehicle

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{\nu} \\ \dot{\boldsymbol{x}} &= \dot{\boldsymbol{\nu}} \end{aligned} \tag{3.22}$$

This is deduced from Equation 3.10, which can be rewritten as

$$\begin{aligned} \dot{\boldsymbol{\nu}} &= \boldsymbol{M}^{-1} \left(-\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{\tau} \right) \\ &= -\boldsymbol{M}^{-1}\boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{M}^{-1}\boldsymbol{\tau} \end{aligned} \tag{3.23}$$

which represents the set of differential equations given by the vector $\boldsymbol{\nu}$ and its time-derivative $\dot{\boldsymbol{\nu}}$. Applying Equation 3.22, which defines the state vector of the system as $\boldsymbol{x}$, and letting the vector of applied forces and moments $\boldsymbol{\tau}$ be the control action vector $\boldsymbol{u}$, the state-space system representation becomes

$$\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{3.24}$$

where $\boldsymbol{A}(\boldsymbol{x}) = -\boldsymbol{M}^{-1}\boldsymbol{D}(\boldsymbol{\nu})$ and $\boldsymbol{B} = \boldsymbol{M}^{-1}$. Since the matrix $\boldsymbol{A}(\boldsymbol{x})$ depends on the state vector $\boldsymbol{x}$, this corresponds to a non-linear system. According to Equation 3.16, this matrix can be decomposed into a linear part $\boldsymbol{A}_{\mathrm{l}}$ and a non-linear (quadratic) time-variant part $\boldsymbol{A}_{\mathrm{n}}(\boldsymbol{x})$ as follows

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x}) &= \boldsymbol{A}_{\mathrm{l}} + \boldsymbol{A}_{\mathrm{n}}(\boldsymbol{x}) \\ &= -\boldsymbol{M}^{-1}\boldsymbol{D}_{\mathrm{l}} - \boldsymbol{M}^{-1}\boldsymbol{D}_{\mathrm{n}}(\boldsymbol{\nu}) \end{aligned} \tag{3.25}$$

For the implementation of the classical part of the controller (feedback gains and the alternative PI controller) it will be assumed that the system is Linear Time Invariant (LTI), by linearizing the system in Equation 3.24 either in an operation point $\boldsymbol{u} = \boldsymbol{u}_{\text{op}}$ and $\boldsymbol{x} = \boldsymbol{x}_{\text{op}}$ or at the zero-input equilibrium state, this is, when $\boldsymbol{u} = 0$ and $\boldsymbol{x} = 0$. On the other hand, for the design of the overall proposed control structure —including the learning module—, the system will be treated as being composed by the linear and non-linear parts presented in Equation 3.25, and the linear part $\boldsymbol{A}_{\text{l}}$ will be used to find the feedback controller gains. This is, it will be assumed that the linearization has been performed at the zero-input equilibrium state.

The vehicles are equipped with an inertial measurement unit (IMU) that provides linear accelerations, angular rates and heading of the vehicle. Additionally, a motion capture system provides the surge and yaw positions, as well as the yaw heading angle. Therefore, the output of the system —this is, the variables that can be read with the available sensors and that are of interest here— corresponds to the angular rates $\boldsymbol{\omega}$ and the linear and angular positions in $\boldsymbol{\eta}$. The velocities in $\boldsymbol{\omega}$ belong to the state vector, so a direct linear transformation can be establish between both. However, the positions $\boldsymbol{\eta}$ are obtained from an integration in time of the velocities that constitute the state vector $\boldsymbol{x}$. In order to achieve a direct linear transformation for the positions as well, the state vector should be augmented such that it includes both the velocities $\boldsymbol{\nu}$ and positions $\boldsymbol{\eta}$. This dynamical system would be defined according to the following state-pace formulation

$$\underbrace{\begin{bmatrix} \dot{\boldsymbol{\nu}} \\ \dot{\boldsymbol{\eta}} \end{bmatrix}}_{\dot{\boldsymbol{x}}_a} = \underbrace{\begin{bmatrix} \boldsymbol{A}(\boldsymbol{\nu}) & \boldsymbol{0}_{3\times3} \\ \boldsymbol{T}_b^n(\boldsymbol{\Theta}) & \boldsymbol{0}_{3\times3} \end{bmatrix}}_{\boldsymbol{A}_a(\boldsymbol{x}_a)} \underbrace{\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\eta} \end{bmatrix}}_{\boldsymbol{x}_a} + \underbrace{\begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{0}_{3\times3} \end{bmatrix}}_{\boldsymbol{B}_a} \underbrace{\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{0}_{3\times1} \end{bmatrix}}_{\boldsymbol{u}_a} \tag{3.26}$$

where $\dot{\boldsymbol{x}}_a$, $\boldsymbol{A}_a(\boldsymbol{x}_a)$, $\boldsymbol{x}_a$, $\boldsymbol{B}_a$ and $\boldsymbol{u}_a$ are the augmented versions of the state vector time-derivative, the state dynamical matrix, the state vector, the input matrix and the control action vector respectively. With this definition, the output $\boldsymbol{y}$ is found as

$$\underbrace{\left[\begin{array}{c} \psi \\ \hdashline \boldsymbol{\eta} \end{array}\right]_{4\times1}}_{\boldsymbol{y}} = \underbrace{\left[\begin{array}{cccc:ccc} 0 & 0 & 1 & & \boldsymbol{0}_{1\times3} & \\ \hdashline & & & & & \\ & \boldsymbol{0}_{3\times3} & & & \boldsymbol{I}_{3\times3} & \end{array}\right]_{4\times6}}_{\boldsymbol{C}} \underbrace{\left[\begin{array}{c} \boldsymbol{\nu} \\ \hdashline \boldsymbol{\eta} \end{array}\right]_{6\times1}}_{\boldsymbol{x}_a} \tag{3.27}$$

The drawback of using state-space system from Equation 3.26 is that the augmented state dynamical matrix $\boldsymbol{A}_a(\boldsymbol{x}_a)$ is now singular, which hinders the design of the controller following common techniques. For this reason, the system in Equation 3.24 will be used for the controller design, while the system from Equations 3.26 and 3.27 will be used in simulation for forward passes in order to obtain the output of the system. A kinematic Kalman filter (KKF), as will be shown later, will be used to obtain the estimate of the state $\boldsymbol{x}$ from the measured outputs. The block diagram of the proposed system in open loop is shown in Figure 3.3.

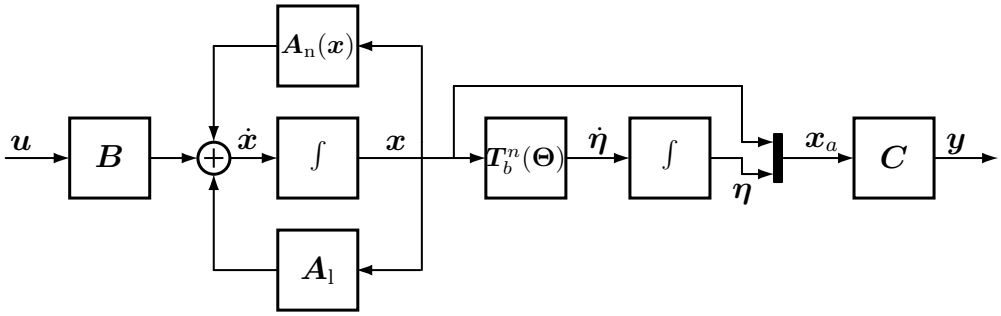

Figure 3.3: Open-loop block-diagram system representation

3.2.2 Controller Design

As mentioned in Section 2.3 from the previous chapter, the designed controller is composed of a feedback and a feedforward part. This section presents how both parts are related and how each one contributes to the total control action. An analysis of the tracking error will be presented which will serve as a guideline for the specific design and tuning of the learning part of the controller.

Feedback Controller

The feedback part aims to control the linearized system given by $\boldsymbol{A}_{\mathrm{l}}$ and is established as the combination of a full-state feedback controller with a reference-feedforward block. This is a commonly used classical control structure (see Figure 3.4) which allows for tuning of the state dynamics via the full-state feedback gain matrix $\boldsymbol{K}$, while providing a way to influence the output via the feedforward block $\boldsymbol{N}$, which gain is usually set to be the inverse of the closed-loop gain in steady state so that the reference $\boldsymbol{r}$ has the same magnitude as the output $\boldsymbol{y}$.

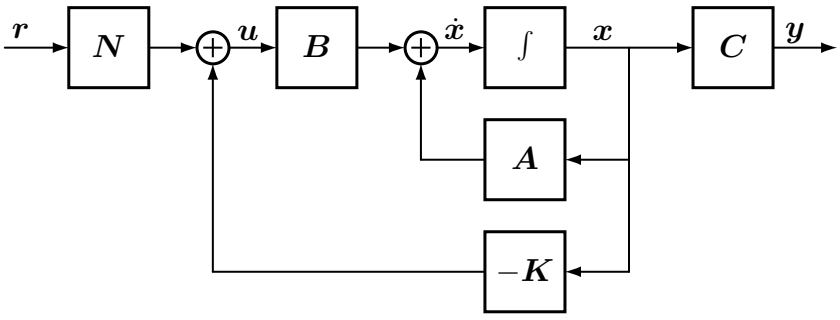

Figure 3.4: Typical full-state feedback ($\boldsymbol{K}$) and reference-feedforward ($\boldsymbol{N}$) control structure for a LTI system (here the matrix $\boldsymbol{A}$ is linear)

However, in this project, the set of variables that are to be controlled are the velocities, which constitute the state itself $\boldsymbol{x}$, and specifically, these velocities need to follow a reference $\boldsymbol{r}$ and the error between them $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{x}$ needs to be minimized. For this

reason, the matrix $\boldsymbol{N}$ is chosen to be the same as $\boldsymbol{K}$ but with opposite sign in the block diagram, since this is equivalent to act on the error with a proportional gain

$$\begin{aligned} \boldsymbol{u}_{\mathrm{FB}} &= \boldsymbol{K}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x} \\ &= \boldsymbol{K}(\boldsymbol{r} - \boldsymbol{x}) \\ &= \boldsymbol{K}\boldsymbol{e} \end{aligned} \tag{3.28}$$

thus constituting a classical proportional (P) controller. Despite this approach not being the common one used to tune the gains of a P controller, it has been employed here for two main reasons. On the one hand, it allows to modify the dynamics of the closed loop via the full-state feedback pole-placement technique, which is an intuitive way to formulate the desired behavior of the system. On the other hand, it facilitates the matrix analysis of the closed-loop state-space system given the inherent matrix formulation of the gain $\boldsymbol{K}$, as will be shown next.

There are two conditions that must be met in order to implement the full-state feedback control

- First, the state $\boldsymbol{x}$ must be known or estimated. Since no direct measurements of the linear velocities are available, the full state is not known a priori. However, given the position measurements in time, these velocities can be estimated with an observer such a KKF.
- Second, for guaranteeing arbitrary pole (or eigenvalue) placement, the system must be controllable. This is satisfied since the matrix $\boldsymbol{A}$ is diagonal and the input matrix $\boldsymbol{B}$ does not have any zero rows, implying that each state evolves independently of the rest and can be directly affected by one of the inputs.

With the control law from Equation 3.28, the closed-loop system expression now becomes, for the LTI system given by $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\left(\boldsymbol{K}\boldsymbol{r} - \boldsymbol{K}\boldsymbol{x}\right) \\ &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K})\boldsymbol{x} + \boldsymbol{B}\boldsymbol{K}\boldsymbol{r} \\ &= \boldsymbol{A}_K\boldsymbol{x} + \boldsymbol{B}\boldsymbol{K}\boldsymbol{r} \end{aligned} \tag{3.29}$$

where $\boldsymbol{A}_K$ is the closed-loop state transition matrix.

Feedforward Learning Controller

The feedforward part of the controller corresponds to the learning module, the ULM, presented in the previous chapter. Formulated in a general way, its inputs are the desired state $\boldsymbol{x}_d$, the desired state time-derivative $\dot{\boldsymbol{x}}_d$ —which in this case are equivalent to the references $\boldsymbol{r}$ and $\dot{\boldsymbol{r}}$, respectively— and current state estimate $\hat{\boldsymbol{x}}$, and its output is the feedforward control action $\boldsymbol{u}_{\mathrm{FF}}$. This module will be referred to as $\boldsymbol{\Phi}$ in the following

diagrams and equations. Since the ULM is composed by two sub-modules, the LWPR and the CM, the output of $\boldsymbol{\Phi}$ is the sum of both of them

$$\boldsymbol{u}_{\mathrm{FF}} = \boldsymbol{u}_{\mathrm{LWPR}} + \boldsymbol{u}_{\mathrm{CM}} \tag{3.30}$$

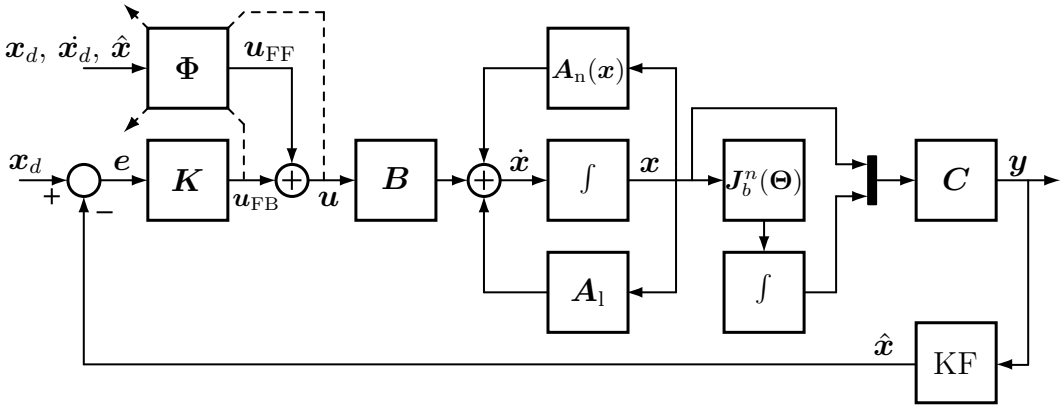

Figure 3.5: Proposed control structure including the feedback module and the feedforward learning module

The non-linear part of the system given by the product $\boldsymbol{A}_{\mathrm{n}}(\boldsymbol{x})\boldsymbol{x}$ is introduced again in order to provide a complete analysis, and for simplicity, it will be referred to as $\boldsymbol{n}(\boldsymbol{x})$ while the linear part will be represented by a matrix named $\boldsymbol{A}$ as opposed to $\boldsymbol{A}_{\mathrm{l}}$

The total control action $\boldsymbol{u}$ provided by the control system depicted in Figure 3.5 is

$$\begin{aligned} \boldsymbol{u} &= \boldsymbol{u}_{\mathrm{FB}} + \boldsymbol{u}_{\mathrm{FF}} \\ &= \boldsymbol{K}\boldsymbol{x}_d - \boldsymbol{K}\boldsymbol{x} + \boldsymbol{u}_{\mathrm{FF}} \end{aligned} \tag{3.31}$$

Inserting this control law into the non-linear state-space formulation of the system, the following expression is found

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{n}(\boldsymbol{x}) + \boldsymbol{B}\left(\boldsymbol{K}\boldsymbol{x}_d - \boldsymbol{K}\boldsymbol{x} + \boldsymbol{u}_{\mathrm{FF}}\right) \\ &= \boldsymbol{A}_K\boldsymbol{x} + \boldsymbol{n}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{K}\boldsymbol{x}_d + \boldsymbol{B}\boldsymbol{u}_{\mathrm{FF}} \end{aligned} \tag{3.32}$$

which provides a full mathematical description of the closed-loop system with the feedback and the feedforward control action as well as the non-linear components. Now, by considering the error definition and its time derivative

$$\begin{aligned} \boldsymbol{e} &= \boldsymbol{x}_d - \boldsymbol{x} \\ \dot{\boldsymbol{e}} &= \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}} \end{aligned} \tag{3.33}$$

Equation 3.32 can be rewritten into the error-dynamics form

$$\dot{\boldsymbol{e}} = \boldsymbol{A}_K\boldsymbol{e} + \dot{\boldsymbol{x}}_d - \boldsymbol{A}\boldsymbol{x}_d - \boldsymbol{n}(\boldsymbol{x}_d - \boldsymbol{e}) - \boldsymbol{B}\boldsymbol{u}_{\mathrm{FF}} \tag{3.34}$$

which contains an asymptotically stable part $\boldsymbol{A}_K\boldsymbol{e}$ that makes the error $\boldsymbol{e}$ converge to zero (since the matrix $\boldsymbol{A}_K$ is designed to be asymptotically stable) only if the rest of the components of the equation (highlighted in red) cancel each other out —this is, their sum is equal to zero—. In order to achieve this, the feedforward action $\boldsymbol{u}_{\text{FF}}$ provided by the ULM should approximate the following function

$$\boldsymbol{u}_{\text{FF}} = \boldsymbol{B}^{-1}\left(\dot{\boldsymbol{x}}_d - \boldsymbol{A}\boldsymbol{x}_d - \boldsymbol{n}(\boldsymbol{x}_d - \boldsymbol{e})\right) \tag{3.35}$$

Now, if the terms in Equation 3.35 are substituted by their original definitions based on the physical system and with $\boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{e}$, the following expression is found

$$\begin{aligned}\boldsymbol{u}_{\text{FF}} &= \boldsymbol{M}\left(\dot{\boldsymbol{x}}_d + \boldsymbol{M}^{-1}\boldsymbol{D}_{\text{l}}\boldsymbol{x}_d + \boldsymbol{M}^{-1}\boldsymbol{D}_{\text{n}}(\boldsymbol{x})\boldsymbol{x}\right)\\ &= \boldsymbol{M}\dot{\boldsymbol{x}}_d + \boldsymbol{D}_{\text{l}}\boldsymbol{x}_d + \boldsymbol{D}_{\text{n}}(\boldsymbol{x})\boldsymbol{x}\\ &= \boldsymbol{M}\dot{\boldsymbol{\nu}}_d + \boldsymbol{D}_{\text{l}}\boldsymbol{\nu}_d + \boldsymbol{D}_{\text{n}}(\boldsymbol{\nu})\boldsymbol{\nu}\end{aligned} \tag{3.36}$$

which corresponds to the inverse dynamics model of the vehicle evaluating the inertia and the linear drag terms at the desired accelerations and velocities, and the non-linear drag term at the current velocity. This shows that the the function to be learned by the ULM depends on these variables, which should be the inputs to the $\boldsymbol{\Phi}$ block. In the case of small errors or fast convergence, the current velocities can be approximated as the desired velocities ($\boldsymbol{x} \approx \boldsymbol{x}_d$), which may allow for using only the desired accelerations and velocities as inputs to $\boldsymbol{\Phi}$ to learn a sufficiently valid inverse dynamics function —in practice it was found that including the estimated state $\hat{\boldsymbol{x}}$ did not provide any significant improvement.

It is important to note at this point that, despite Assumption 3.2 being made, the learning controller does not require the matrices $\boldsymbol{M}$ and $\boldsymbol{D}$ to be diagonal since, as a general function approximator, the ULM can represent a function of the form shown in Equation 3.36 in which the matrices $\boldsymbol{M}$ and $\boldsymbol{D}$ are dense. This implies that, even if Assumption 3.2 is not fully met in practice, the ULM will still be able to represent a model with coupled motion between DOF.

3.2.3 Learning Mechanism, Stability and Convergence

In Section 2.2 of the previous chapter, the principles of the ULM were presented, providing a general understanding of the capabilities of this control method. Here, a more detailed mathematical description of its elements is shown, including the computation of the output and the update rule of the function. Additionally, although a full mathematical proof for its stability and convergence has not been developed, a brief, principled analysis is presented, which may also serve as a guideline for tuning of the algorithm.

The ULM is composed by the LWPR and the CM (cerebellar module). Regarding the LWPR, the main mathematical principles of its output computation and function update were already described in Section 2.2. After the analysis of the feedforward learning controller provided in the previous section, it can be now specified that the

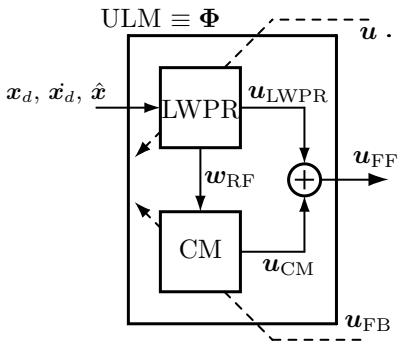

Figure 3.6: Detail of a ULM unit ($\boldsymbol{\Phi}$ in the block diagrams)

output of the LWPR is a function of the desired velocities and accelerations of the vehicle system and optionally the current state estimate

$$\boldsymbol{u}_{\mathrm{LWPR}} = \mathrm{LWPR}\left(\boldsymbol{x}_d,\ \dot{\boldsymbol{x}}_d,\ \hat{\boldsymbol{x}}\right) \tag{3.37}$$

The update of the LWPR is realized by setting the target function to be learned as the mapping from the current LWPR inputs to the the total control action at the current time step, this is

$$\mathrm{LWPR}_{\mathrm{update}}(t) := \left(\boldsymbol{x}_d(t),\ \dot{\boldsymbol{x}}_d(t),\ \hat{\boldsymbol{x}}(t)\right) \rightarrow \boldsymbol{u}(t) \tag{3.38}$$

where $\boldsymbol{u}(t) = \boldsymbol{u}_{\mathrm{FF}}(t) + \boldsymbol{u}_{\mathrm{FB}}(t)$. From a computational point of view, a data point is added to the algorithm, which is used to update the local linear regression models.

Regarding the CM, for a given controlled degree of freedom n —in the next equations, for simplicity, the subscript n will be dropped for simplicity, and the scalar signals $u_{\mathrm{CM},n}$, $e_{\mathrm{CM},n}$, $u^*_{\mathrm{CM},n}$ and $u_{\mathrm{FB},n}$ will be referred to as u_{CM}, e_{CM}, u^*_{CM} and u_{FB}, respectively—, the output u_{CM} is obtained from the inner product between the vector of the CM weights $\boldsymbol{w}_{\mathrm{CM}}$ and the vector of LWPR RFs activations $\boldsymbol{w}_{\mathrm{RF}}$ for such DOF controlled

$$u_{\mathrm{CM}} = \langle \boldsymbol{w}_{\mathrm{CM}}, \boldsymbol{w}_{\mathrm{RF}} \rangle = \boldsymbol{w}_{\mathrm{CM}}^{\top} \boldsymbol{w}_{\mathrm{RF}} \tag{3.39}$$

The update rule is found by applying gradient descent. An error function over the CM weights is defined

$$E\left(\boldsymbol{w}_{\mathrm{CM}}\right) = \frac{1}{2} e_{\mathrm{CM}}^2 \tag{3.40}$$

where e_{CM} is the error between the CM output u_{CM} and the target function u^*_{CM} to be learned by the CM

$$e_{\mathrm{CM}} = u^*_{\mathrm{CM}} - u_{\mathrm{CM}} \tag{3.41}$$

Since the target function is not known, it can be approximated to be

$$u^*_{\mathrm{CM}} \approx u_{\mathrm{CM}} + u_{\mathrm{FB}} \tag{3.42}$$

The reason behind this approximation is that the control action u_{FB} provided by the feedback controller reflects the current tracking error $\boldsymbol{e}$ in the system —for the specific DOF—, defined in Equation 3.33. If tracking was perfect, the error $\boldsymbol{e}$ would be zero, as well as the action $\boldsymbol{u}_{\mathrm{FB}}$, which would imply that, in Equation 3.42, the currently provided control action u_{CM} would already match the target. In the general case of imperfect tracking, the action u_{FB} can be seen as the deviation between the target and the current action output, thus playing the role of an "action error", this is

$$e_{\mathrm{CM}} \approx u_{\mathrm{FB}} \tag{3.43}$$

With this, according to gradient descent, the change of a CM weight i is determined by

$$\Delta w_{\mathrm{CM},i} = -\beta \frac{\partial E}{\partial w_{\mathrm{CM},i}} \tag{3.44}$$

where the partial derivative is found as

$$\frac{\partial E}{\partial w_{\mathrm{CM},i}} = \frac{1}{2}\frac{\partial e_{\mathrm{CM}}^2}{\partial w_{\mathrm{CM},i}} = e_{\mathrm{CM}}\frac{\partial e_{\mathrm{CM}}}{\partial w_{\mathrm{CM},i}} \tag{3.45}$$

and finally

$$\frac{\partial e_{\mathrm{CM}}}{\partial w_{\mathrm{CM},i}} = \frac{\partial}{\partial w_{\mathrm{CM},i}}\left(u_{\mathrm{CM}}^* - \sum w_{\mathrm{CM},i} w_{\mathrm{RF},i}\right) = -w_{\mathrm{RF},i} \tag{3.46}$$

Therefore, the gradient descent update rule is

$$\Delta w_{\mathrm{CM},i} = -\beta\, e_{\mathrm{CM}}(-w_{\mathrm{RF},i}) \approx \beta\, u_{\mathrm{FB}}\, w_{\mathrm{RF},i} \tag{3.47}$$

which is the approximation that will be used to change the weights of the CM.

Now, two additional parameters in the LWPR algorithm are presented which are relevant for the convergence analysis. The first one is the forgetting factor $\lambda_f \in [0, 1]$, which determines how much the current model is maintained against new data points. This is, for $\lambda_f \approx 1$, the model is virtually not updated with any new data and for $\lambda_f \approx 0$, the model represents only the last data point. The second parameter is the output confidence interval $I_c \in (0, +\infty)$, which accounts for the uncertainty in the model derived from the data scarcity and disparity (i.e., how much the gathered data "disagrees" with the function to be represented) and roughly represents the range in which the "true" function output may be located. For large, congruent amounts of data around a point in the input space, I_c will tend to zero, whereas it will tend to infinity for no data or for data representing a random function, since there is no knowledge or consensus on what the function should represent.

With this, the qualitative convergence and stability analysis of the learning algorithm is presented now. Focusing again on Equation 3.34, the error-dynamics equation can be seen as a dynamical system with state variable $\boldsymbol{e}$, state-transition matrix $\boldsymbol{A}_K$, control action $\boldsymbol{u}_{\mathrm{FF}}$ and disturbance $\boldsymbol{d}_e$ equal to the rest of the terms

$$\boldsymbol{d}_e = \dot{\boldsymbol{x}}_d - \boldsymbol{A}\boldsymbol{x}_d - \boldsymbol{n}(\boldsymbol{x}_d - \boldsymbol{e}) \tag{3.48}$$

Since $\boldsymbol{u}_{\text{FF}}$ is a signal that changes in time due to the update rules of the LWPR and the CM, an approximated dynamical formulation can be established. It is assumed here that the system is operating in a small enough region $\mathcal{O}$ around the ULM input variables $\dot{\boldsymbol{x}}_d^o$, $\boldsymbol{x}_d^o$, $\boldsymbol{x}^o$, in order to allow the ULM to collect data and learn a function for that input region. With this, the dynamical equations of the signals that compose $\boldsymbol{u}_{\text{FF}}$, this is, $\boldsymbol{u}_{\text{LWPR}}$ and $\boldsymbol{u}_{\text{CM}}$, are presented next.

For the CM —and again, for a given DOF n, despite the subscript not being included in the following equations for simplicity—, the variation in time can be found by first expressing Equation 3.39 in variation form

$$\Delta u_{\text{CM}} \approx \Delta \boldsymbol{w}_{\text{CM}}^{\top} \boldsymbol{w}_{\text{RF}} \tag{3.49}$$

where the approximation is due to the previous assumption, such that the variation in the input space is low (in the region $\mathcal{O}$) and therefore the RF activations $\boldsymbol{w}_{\text{RF}}$ between two time-steps remain approximately the same. Now, taking Equation 3.47 expressed in vector form and substituting in the previous equation, the following expression is found

$$\begin{aligned} \Delta u_{\text{CM}} &= \beta\, u_{\text{FB}}\, \boldsymbol{w}_{\text{RF}}^{\top} \boldsymbol{w}_{\text{RF}} \\ &= \gamma_{\text{CM}}\, u_{\text{FB}} \end{aligned} \tag{3.50}$$

which gives the update rule in discrete time of the CM as a function of u_{FB} with the parameter $\gamma_{\text{CM}} = \beta\, \boldsymbol{w}_{\text{RF}}^{\top} \boldsymbol{w}_{\text{RF}}$.

For the case of the LWPR, the rate of change in the model is initially large but after a few data points have been collected, the the update rule in the region $\mathcal{O}$ at the iteration k is approximated by λ_f as follows (derived from the original LWPR paper [32])

$$u_{\text{LWPR},k+1} = \lambda_f\, u_{\text{LWPR},k} + (1 - \lambda_f) u_k \tag{3.51}$$

where u_k is the total output for a given DOF n, this is

$$u_k = u_{\text{FF},k} + u_{\text{CM},k} = u_{\text{LWPR},k} + u_{\text{CM},k} + u_{\text{FB},k} \tag{3.52}$$

which substituted back in the previous equation, it becomes

$$u_{\text{LWPR,k+1}} = \lambda_f\, u_{\text{LWPR,k}} + (1 - \lambda_f)(u_{\text{LWPR},k} + u_{\text{CM},k} + u_{\text{FB},k}) \tag{3.53}$$

and in variation form

$$\begin{aligned} \Delta u_{\text{LWPR}} &= (1 - \lambda_f)(u_{\text{CM}} + u_{\text{FB}}) \\ &= \gamma_{\text{LWPR}}(u_{\text{CM}} + u_{\text{FB}}) \end{aligned} \tag{3.54}$$

which shows that the update of the LWPR depends both on the feedback and CM terms, and the update rate for the LWPR is given by $\gamma_{\text{LWPR}} = (1 - \lambda_f)$.

With both update rules defined in discrete time and their corresponding rates given by γ_{CM} and γ_{LWPR}, the next step is to formulate these equations in continuous time. For this, the following time-derivative approximation will be used

$$\dot{\boldsymbol{f}} \approx \frac{\Delta \boldsymbol{f}}{\Delta t} \tag{3.55}$$

where $\boldsymbol{f}$ is a generic function and Δt is the time interval between two consecutive time-steps in the discrete controller. This approximation will be more accurate as $\Delta t \to 0$. Applying this to the LWPR update rule and expressing it in vector form

$$\begin{aligned} \dot{\boldsymbol{u}}_{\text{LWPR}} &\approx \frac{\Delta \boldsymbol{u}_{\text{LWPR}}}{\Delta t} \\ &= \frac{\gamma_{\text{LWPR}}}{\Delta t}(\boldsymbol{u}_{\text{CM}} + \boldsymbol{u}_{\text{FB}}) \\ &= \gamma'_{\text{LWPR}}(\boldsymbol{u}_{\text{CM}} + \boldsymbol{u}_{\text{FB}}) \end{aligned} \tag{3.56}$$

where γ'_{LWPR} is the continuous-time update parameter for the LWPR. Now, for the CM update rule in vector form it becomes

$$\begin{aligned} \dot{\boldsymbol{u}}_{\text{CM}} &\approx \frac{\Delta \boldsymbol{u}_{\text{CM}}}{\Delta t} \\ &= \frac{\gamma_{\text{CM}}}{\Delta t}\boldsymbol{u}_{\text{FB}} \\ &= \gamma'_{\text{CM}}\,\boldsymbol{u}_{\text{FB}} \end{aligned} \tag{3.57}$$

where γ'_{LWPR} is the CM continuous-time update parameter.

As a last step, the feedback control action $\boldsymbol{u}_{\text{FB}}$ depends on the tracking error $\boldsymbol{e}$ such that $\boldsymbol{u}_{\text{FB}} = \boldsymbol{K}\boldsymbol{e}$. This defines the continuous-time update rule for the LWPR as

$$\dot{\boldsymbol{u}}_{\text{LWPR}} \approx \gamma'_{\text{LWPR}}(\boldsymbol{u}_{\text{CM}} + \boldsymbol{K}\boldsymbol{e}) \tag{3.58}$$

and for the CM

$$\dot{\boldsymbol{u}}_{\text{CM}} \approx \gamma'_{\text{CM}}\,\boldsymbol{K}\boldsymbol{e} \tag{3.59}$$

Finally, putting all main results together, the state-space system approximation for the error $\boldsymbol{e}$ and the feedforward control ation components $\boldsymbol{u}_{\text{LWPR}}$ and $\boldsymbol{u}_{\text{CM}}$ around an input region $\mathcal{O}$ can be defined as

$$\begin{bmatrix} \dot{\boldsymbol{e}} \\ \dot{\boldsymbol{u}}_{\text{CM}} \\ \dot{\boldsymbol{u}}_{\text{LWPR}} \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{A}_K & -\boldsymbol{B} & -\boldsymbol{B} \\ \gamma'_{\text{CM}}\boldsymbol{K} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\ \gamma'_{\text{LWPR}}\boldsymbol{K} & \gamma'_{\text{LWPR}} & \boldsymbol{0}_{3\times3} \end{bmatrix}}_{\boldsymbol{A}_e} \begin{bmatrix} \boldsymbol{e} \\ \boldsymbol{u}_{\text{CM}} \\ \boldsymbol{u}_{\text{LWPR}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{I}_{3\times3} \\ \boldsymbol{0}_{3\times3} \\ \boldsymbol{0}_{3\times3} \end{bmatrix} \boldsymbol{d}_e \tag{3.60}$$

This system shares similarities with a traditional control structure that includes integral action. Here, the integral signal would correspond to the sum of the feedforward signals $\boldsymbol{u}_{\text{CM}}$ and $\boldsymbol{u}_{\text{LWPR}}$, which builds up on the error $\boldsymbol{e}$ and provides the control action in a negative feedback way through $\boldsymbol{B}$, thus acting against the error. The signal $\boldsymbol{d}_e$ corresponds to a disturbance that is injected into the state to be stabilized, in this case the error $\boldsymbol{e}$. This way, the matrix $\boldsymbol{A}_e$ can be adjusted through the tunable parameters $\boldsymbol{K}$, γ'_{CM} and γ'_{LWPR} such that the system is made stable (the trajectory tracking error is bounded and converges to zero) and capable of rejecting disturbances (the terms in

$\boldsymbol{d}_e$ are compensated by $\boldsymbol{u}_{\mathrm{FF}}$, thus effectively learning the inverse dynamics model from Equation 3.36).

A remarkable observation that can be made from the formulated error system is that, the speed of convergence of the feedforward action $\boldsymbol{u}_{\mathrm{FF}}$ towards the function that compensates $\boldsymbol{d}_e$, is determined by $\boldsymbol{K}$ in both of its components $\boldsymbol{u}_{\mathrm{CM}}$ and $\boldsymbol{u}_{\mathrm{LWPR}}$. For larger values of this parameter, the convergence will be faster, meaning that the inverse model will update at a higher rate. This may be desirable in the initial stage of learning or upon a significant change in the physical model induced by sources such as the failure of some motor or an permanently added external force (a strong recurrent ocean current, an undesirable object attached to the vehicles, etc.). However, for small, randomly-fluctuating disturbances, having a higher gain $\boldsymbol{K}$ will amplify the noise carried by these external signals. In order to tackle this, it is proposed here to use the confidence interval I_c to distinguish between this double nature of external disturbances and adjust the update rate of the feedforward action accordingly.

Specifically, recalling the properties of I_c, the confidence interval will grow (representing more uncertainty) if little data has been gathered around an input region (initial phase of learning) or if a permanent change in the model has altered the function represented by $\boldsymbol{d}_e$ and over a given period of time this new data has been gathered by the LWPR thus increasing the disparity in the model. In these situations, the rate of change in $\boldsymbol{u}_{\mathrm{FF}}$ should increase, and this has been realized in this project by adjusting the gain of the feedback controller $\boldsymbol{K}$ as a function of I_c according to the following equation

$$\boldsymbol{K}(I_c) = \boldsymbol{K}_{\max} - (\boldsymbol{K}_{\max} - \boldsymbol{K}_{\min}) \exp\left(\frac{-I_c^2}{\sigma_c^2}\right) \tag{3.61}$$

which represents a smooth function varying from $\boldsymbol{K}_{\min}$ at high certainty ($I_c \approx 0$) to $\boldsymbol{K}_{\max}$ at lower certainty ($I_c \approx 3\sigma_c$). The parameter $3\sigma_c$ is be chosen to be lower than the maximum possible value represented by the inverse dynamics function or the maximum demandable control action (limited by the thrusters capacity). This is, for such levels represented by I_c, the uncertainty in the $\boldsymbol{u}_{\mathrm{FF}}$ output is "as high as it can get", and therefore the function should converge faster to the currently represented dynamics model, thus increasing the convergence rate by providing $\boldsymbol{K}_{\max}$. The effectivity of this modulated gain will be tested in the system.

3.2.4 Control Allocation and Distributed Control Architecture

So far the controller presented addresses the motion control of each DOF independently. However, given the fact that several thrusters may interact in the motion of a given DOF, the problem of how to properly assign the thrust to each motor needs to be solved. Additionally, the analysis of this problem will also open the possibility of another control architecture, as will be shown next.

The contribution of each thruster in the vehicle system to the force and moments applied in each DOF is determined by the thruster configuration matrix (TCM), designated here as $\boldsymbol{H}$. For a system composed of N_i vehicles, with N_j thrusters per vehicle, and N_x degrees of freedom to be controlled, the matrix $\boldsymbol{H}$ has dimensions $N_x \times N_i \cdot N_j$. For the specific case of the vehicle used in this project (BlueROV-R1) operating at a constant-depth plane (only three thrusters used, $N_j = 3$), the TCM of the i^{th} vehicle, with $i = 1, 2, \ldots, N_i$, is

$$\boldsymbol{H}_i = \begin{bmatrix} \cos(\alpha_{i,1}) & \cos(\alpha_{i,2}) & \cos(\alpha_{i,3}) \\ \sin(\alpha_{i,1}) & \sin(\alpha_{i,2}) & \sin(\alpha_{i,3}) \\ p_{i,1}\sin(\beta_{i,1}) & p_{i,2}\sin(\beta_{i,2}) & p_{i,3}\sin(\beta_{i,3}) \end{bmatrix} \tag{3.62}$$

where $\alpha_{i,j}$, with $j = 1, 2, 3$, is the angle described from the system $\{b\}$-x-axis to the thruster $\{j\}$-x-axis belonging to vehicle i; $\beta_{i,j}$ is the angle form the vector $\boldsymbol{p}_j^b$ —described in Equation 3.20— to the thruster $\{j\}$-x-axis belonging to vehicle i, and $p_{i,j}$ is the magnitude of the vector $\boldsymbol{p}_j^b$ for the vehicle i

For the specific case of a single BlueROV vehicle, $\{b\}$ and $\{i\}$ and, as shown in Figure 3.7, this matrix becomes

$$\boldsymbol{H}_i = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ -p_{i,1} & p_{i,2} & p_{i,3} \end{bmatrix} \tag{3.63}$$

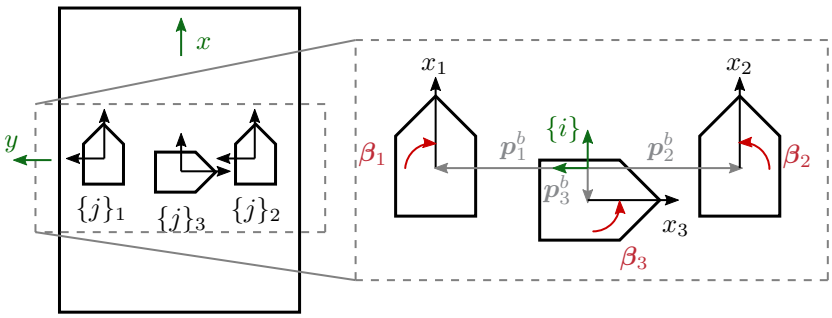

Figure 3.7: BlueROV-R1 thruster distribution. Only those thrusters involved in the motion control of the vehicle at a constant-depth plane are shown here, In this case $\{b\}$ and $\{i\}$ coincide since only one vehicle is represented

The contribution to the total torque and moment vector $\boldsymbol{\tau}_i$ from vehicle i is found with the multiplication

$$\boldsymbol{\tau}_i = \boldsymbol{H}_i\boldsymbol{\rho}_i = \boldsymbol{H}_i \begin{bmatrix} \rho_{i,1} \\ \rho_{i,2} \\ \rho_{i,3} \end{bmatrix} \tag{3.64}$$

where $\boldsymbol{\rho}_i$ is the motors' thrust vector of the vehicle i and $\rho_{i,j}$, with $j = 1, 2, 3$, is the thrust provided by the j^{th} thruster in vehicle i.

For a system composed of N_i vehicles, the total force and moment vector $\boldsymbol{\tau}$ applied is found as

$$\boldsymbol{\tau} = \boldsymbol{H}\boldsymbol{\rho} = \begin{bmatrix} \boldsymbol{H}_1 & \boldsymbol{H}_2 & \cdots & \boldsymbol{H}_I \end{bmatrix} \begin{bmatrix} \boldsymbol{\rho}_1 \\ \boldsymbol{\rho}_2 \\ \vdots \\ \boldsymbol{\rho}_I \end{bmatrix} \tag{3.65}$$

Now, the element provided by the controller is $\boldsymbol{\tau}$, whereas the actuators of the system provide $\boldsymbol{\rho}$. If the individual thrusts needed to be assigned to each motor —in some cases this is not required since the vehicle automatically makes this assignment—, these could be found by

$$\boldsymbol{\rho} = \boldsymbol{H}^{+}\boldsymbol{\tau} \tag{3.66}$$

where $\boldsymbol{H}^{+}$ denotes the Moore-Penrose inverse of $\boldsymbol{H}$, since it is not a square matrix for the case of multiple vehicles.

Distributed Control Architecture

At this point, after establishing the method to convert from total forces and torque to individual motor thrust commands, a distributed control architecture that incorporates such method is presented.

The inspiration behind this architecture is that, from a biological perspective, different muscles in the body are controlled by different regions in the motor cortex and the cerebellum, allowing for fine-tuning of the specific motion requirements of each body part. Here, the question of whether the same principle can be applied to a mobile system with several actuators arises. To address this question, the proposed architecture assigns an individual ULM to each thruster, while the feedback control is still performed at the level of the whole rigid-body system (an independent feedback controller for each DOF). Although this is not a truly distributed architecture, since the information is still processed at the body-level and then allocated to the motor-level where the ULMs use such information to update their functions, it presents a first step towards a biologically-plausible control structure.

The details of this control architecture are presented in Figure 3.8 and a step-by-step description of the internal operations is shown next.

1. The sensor measurements $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ from each vehicle are processed by their respective Kalman filters KF_1 and KF_2. This produces the state estimates $\hat{\boldsymbol{x}}_2$, $\hat{\boldsymbol{x}}_2$, which correspond to the velocities of each vehicle's frame of reference $\{i\}$.

2. Considering the geometry of the formation, the geometric transformation of the estimated velocities $\hat{\boldsymbol{x}}_2$, $\hat{\boldsymbol{x}}_2$ from the vehicles reference frames $\{i\}$ to the formation frame $\{b\}$ is performed, and the average is calculated, thus obtaining $\hat{\boldsymbol{x}}$.

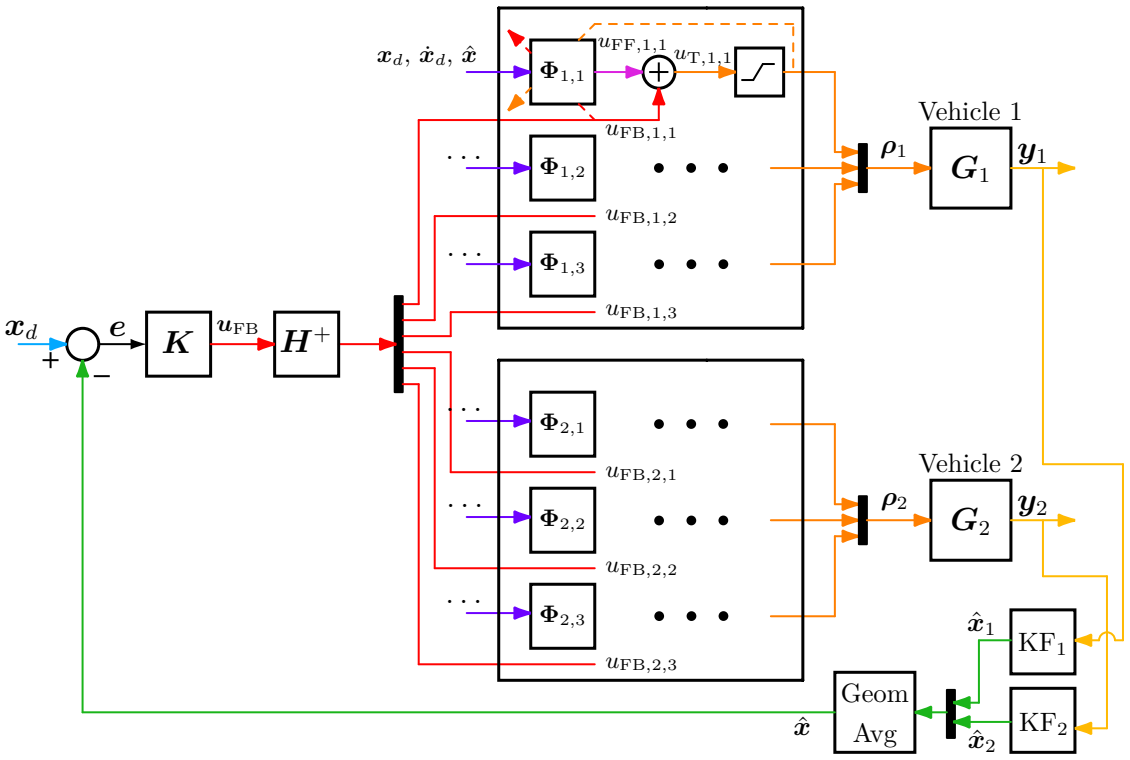

Figure 3.8: Distributed control architecture for the case of two vehicles

3. The estimated state $\hat{\boldsymbol{x}}$ of the formation is subtracted to the reference $\boldsymbol{x}_d$ to obtain the error $\boldsymbol{e}$. This is fed to the proportional controller $\boldsymbol{K}$ which yields the control action vector $\boldsymbol{u}_{\mathrm{FB}}$ containing the force or moment to be applied to each DOF.

4. The transformation of the control action signal $\boldsymbol{u}_{\mathrm{FB}}$ from DOF to individual motors is calculated via Equation 3.66, that provides the allocated feedback control actions $u_{\mathrm{FB},i,j}$ for motor j of vehicle i.

5. The feedforward control action $u_{\mathrm{FF,i,j}}$ for motor j in vehicle i is also obtained at this point from the corresponding ULM, $\boldsymbol{\Phi}_{1,1}$, and both the feedforward and feedback control actions are added together to produce $u_{\mathrm{T,i,j}}$. Note that the input for all ULMs is the same and corresponds to the whole formation' reference velocities $\boldsymbol{x}_d$ and accelerations $\dot{\boldsymbol{x}}_d$, and optionally the estimated velocities $\hat{\boldsymbol{x}}$.

6. The total control action signal $u_{\mathrm{T,i,j}}$ is limited by a saturation block which considers the maximum thrust obtainable at each thruster, and the saturated signals are merged into the vector of motor thrusts $\boldsymbol{\rho}_i$ for vehicle i.

7. Finally, the saturated total control action is fed to the LWPR as the objective function to be learned, and the feedback control action is used to update the weights of the CM (cerebellar module). Note that, although it has not been represented in the diagram, the weights of the CM are only updated if the total control action has

not reached saturation, in order to avoid error accumulation and excesive growinf od the weights (similar to an anti-windup mechanism in a PID controller).

3.3 Guidance, Navigation and Control

The previous sections presented the motion equations of the vehicle and how the proposed controller is designed to track a velocity trajectory. However, the full solution to the trajectory tracking problem requires other components that provide and process information needed by the controller to become truly functional. This section presents these components and how all of them are integrated in the full control solution.

Trajectory tracking can be encompassed in the Guidance, Navigation and Control (GNC) framework. GNC divides the motion control architecture of autonomous mobile systems into several connected modules (guidance, navigation and control) that play complementary roles. The **guidance** module accounts for the decisions to be made during the operation of the vehicle and the interaction with the environment. It follows a plan and, considering the state of the environment (such as the presence of other objects), it provides the right setpoints to the controller in order to execute the desired path. The **navigation** module addresses the state estimation of the vehicle by using the available sensors and applying observer and filtering techniques such as the Kalman filter. The **control** module deals with providing the right control actions to the actuators according to the references provided by the guidance system and the estimated state from the navigation system.

The control module corresponds to the controller architecture already presented in previous sections. The guidance and navigation modules are presented next.

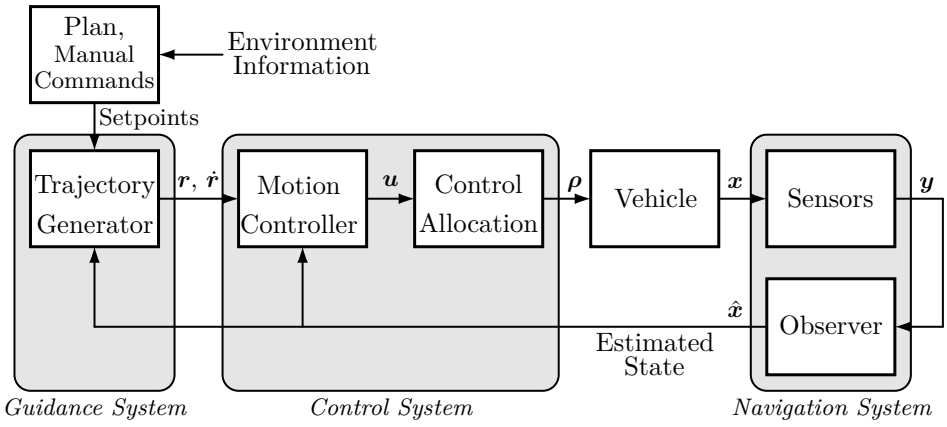

Figure 3.9: GNC signal flow diagram. Recreated from [15]

3.3.1 Guidance System

For trajectory tracking, the guidance system (GS) provides the trajectory to be followed in terms of the controlled variable and its time-derivative. As previously mentioned, in this case this corresponds to the desired velocity and the acceleration, which are sufficient to describe the dynamics of the vehicle and therefore can be used by the feedforward block in the control module to provide an inverse dynamics control action.

The method to generate the trajectory can vary and the main requisite is that the path generated in time is smooth and its time-derivative is bounded, as established in 3.1. In this project, it is assumed that a series of setpoints in velocities have to be reached at different arbitrary times, and the transition from the current velocity to the next setpoint should be described by a smooth curve in time.

These velocity setpoints can be established from several sources, whether it is from a planner module that considers the mission needs, a manual operator, or even a higher-level position controller whose control action directly affects these setpoints. For the purpose of testing the algorithm, as it is the case of this project, a set of setpoints describing a simple periodic path will be used.

In order to achieve smooth transitions between the setpoints, a simulated dynamical model of the closed-loop system is employed. This allows for generating a trajectory that is compatible with physics and that presents desirable properties in terms of convergence speed that can be tuned via the closed-loop controller. In fact, this *guidance controller* can be chosen arbitrarily to achieve the desired behavior of the system. There are, however, several considerations to account for in the implementation of the simulated model and the guidance controller.

1. The closed-loop bandwidth of the simulated model must be lower than the real system closed-loop bandwidth. This is, the rate of change of the signals generated in the GS should not exceed that achievable in the real system.

This condition can be achieved in two ways. On the one hand, the bandwidth of the open loop itself can be affected by the right choice of the inertia $\boldsymbol{M}$ and drag $\boldsymbol{D}$ matrices. Specifically, by increasing the magnitude of $\boldsymbol{M}$ and/or decreasing that of $\boldsymbol{D}$, the bandwidth is reduced since the poles of the system given by the matrix $\boldsymbol{A} = -\boldsymbol{M}^{-1}\boldsymbol{D}$ are moved closer to the origin (they become smaller in magnitude and therefore the time constant of the system increases). Additionally, increasing $\boldsymbol{M}$ also affects the scaling of the input to the system, given by the matrix $\boldsymbol{B} = \boldsymbol{M}^{-1}$. On the other hand, the the closed-loop bandwidth is directly influenced by the gains of the guidance controller, such that, for a higher proportional gain, the bandwidth is increased. The guidance controller chosen in this project is a PI velocity controller, which will be tuned manually.

An important remark at this point is that, as reflected on the objectives and scope of this project, it is assumed that the model of the plant is not fully known, specially in the case of multiple vehicles. For this reason, the model used by the GS consists of the

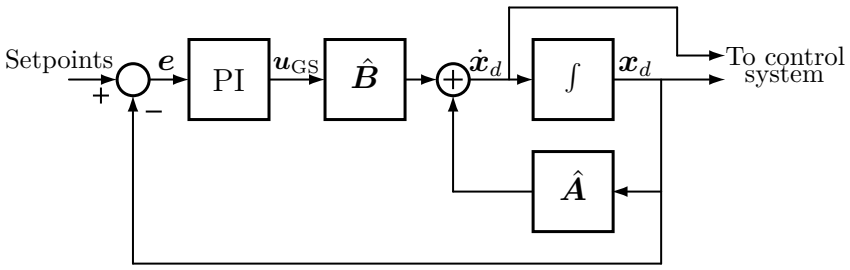

Figure 3.10: Guidance system block diagram

linear approximated matrices $\hat{\boldsymbol{M}}$, $\hat{\boldsymbol{D}}$, which constitute the linear state space matrices $\hat{\boldsymbol{A}}$, $\hat{\boldsymbol{B}}$. Furthermore, even though an estimate of the magnitude of $\boldsymbol{M}$ and $\boldsymbol{D}$ for a single vehicle may be available, it cannot be directly scaled to multiple vehicles since additional unaccounted fluid dynamics effects may appear. This can be addressed by establishing conservatively high values for the inertia matrix $\hat{\boldsymbol{M}}$ while keeping the gains of the guidance controller low enough such that it is ensured that the bandwidth of the simulated closed-loop system in the GS remains low.

2. The simulated model should account for any limitation of the vehicles such as the motor saturation levels or the maximum speed achievable.

This consideration is crucial in order to keep the signals bounded within the limits of the actual system, and can be easily achieved by implementing these saturation elements into the GS vehicle model.

3. The simulated model should include any information available regarding the distribution of the vehicles and their thrusters when they are attached to each other in a formation.

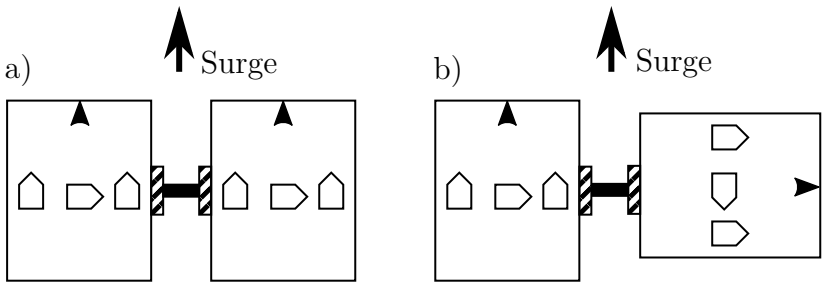

Figure 3.11: Two possible vehicle formations with the same number of vehicles but different configuration

The importance of this last consideration resides in the fact that, for a given set of vehicles, the spatial arrangement (position and orientation) of each vehicle within the formation will determine the availability of thrusters for each DOF, which in turn

affects the total control action (saturation level) that can be assigned to each DOF. This is illustrated in Figure 3.11, which shows two cases with the same number of vehicles but different configuration and therefore available thrust in each DOF.

3.3.2 Navigation System

The navigation module in the multiple-vehicle system is composed by the set of observers that provide the state estimates for each vehicle and a block accounting for the combination of all these filter outputs into a single state estimate corresponding to the frame $\{b\}$ of the whole system.

The filters used by the motion capture system in the experimental facilities is a kinematic Kalman filter (KKF) which provides linear velocity estimates from the position measurements. The angular velocities are provided by the inertial measurement units (IMUs) in the vehicles. For the simulation, a plane-restricted motion version of the KKF has been implemented, following the theory presented in [18].

The block in charge of providing the final estimate of the $\{b\}$ state, already represented in Figure 3.8, obtains the angular velocity as the average of all angular velocities

$$\hat{\boldsymbol{\omega}} = \hat{\boldsymbol{\omega}}_b^b = \frac{1}{N_i} \sum_i^{N_i} \hat{\boldsymbol{\omega}}_i^b \tag{3.67}$$

where $\hat{\boldsymbol{\omega}}_b^b$ is the estimated angular velocity of the vehicle formation, $\hat{\boldsymbol{\omega}}_b^i$ is the estimated angular velocity of a vehicle i in the formation expressed in the formation reference frame $\{b\}$ and N_i is the total number of vehicles. The linear velocity is found with the expression

$$\hat{\boldsymbol{v}} = \hat{\boldsymbol{v}}_b^b = \frac{1}{N_i} \sum_i^{N_i} \left(\hat{\boldsymbol{v}}_i^b + \boldsymbol{p}_i^b \times \hat{\boldsymbol{v}}_i^b \right) \tag{3.68}$$

where $\boldsymbol{p}_i^b$ is the vector from the center of a vehicle frame $\{i\}$ to the center of the formation frame $\{b\}$, $\hat{\boldsymbol{v}}_b^b$ is the estimated linear velocity of the vehicle formation, $\hat{\boldsymbol{v}}_b^i$ is the estimated linear velocity of a vehicle i in the formation expressed in the formation reference frame $\{b\}$.

CHAPTER 4
Materials and Methods

This chapter presents the hardware and software elements used in the development and execution of this project, as well as it provides a further description of some implementation details of the proposed solution.

4.1 Hardware and Facilities

The main hardware components are the underwater vehicles, the motion capture system and the remote operation elements. Some of these components are proprietary hardware that had been acquired by the REMORA laboratory at DTU while others had been deveolped in the laboratory.

Pool

The experimental campaign was held in the pool located inside the Autonomous Systems Test Arena (ASTA) belonging to the Automation and Control department at DTU, Denmark. It is 6 meters long, 3.5 m wide and 3 m deep.

BlueROV

The set of underwater vehicles used corresponds to several units of the *BlueROV* vehicle, from Blue Robotics® [4], which is a remotely operated underwater vehicle (ROV) that offers full-DOF motion control capabilities and it is highly configurable. The specific version used is the BluerROV-R1, which was the first commercially available version launched by Blue Robotics®.

The vehicle has six thrusters, which offer maneuverability in all six DOF although, as mentioned in previous chapters, only those involved in the motion at a constant-depth plane are used here. These are highlighted in Figure 4.1.

The sensors available in the vehicle include an inertial measurement unit (IMU) which provides the angular velocities and linear accelerations (although the latter is not used in this project), and a depth sensor which is used to hold the ROV at the constant-depth plane at which it will navigate.

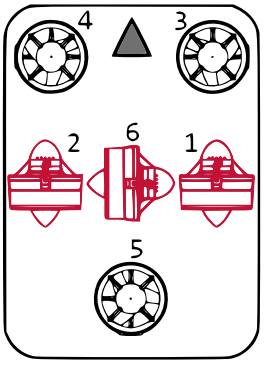

Figure 4.1: BlueROV frame diagram with the used thrusters highlighted in red

The ROVs are equipped with several additional electronic components that are involved in its operation, the most relevant ones being the on-board computer and the autopilot computer. The first hosts the operating system of the ROV, runs the programs for control and other functionalities and communicates with the autopilot computer as well as any other connected remote computer via an Ethernet cable. The second acts as a sensor-fusion unit and it manages the signals to be sent to the motors.

In order to connect several ROVs, there are custom-made docking modules available —developed by the REMORA laboratory— that can be fixed to either of the vehicle sides. A pair of these modules attach to each other magnetically ensuring a rigid connection between the ROVs carrying them.

Motion Capture System

The motion capture system (MCS) provides the position and rotation measurements of a vehicle that has been equipped accordingly to be detected by the system. Specifically, the MCS relies on the optical detection of a light-emmitting diode (LED) pattern that is carried by the vehicle. There main components of the MCS are:

- A monocular camera. It is equipped with an infrared filter and provides the captured images to the personal computer.
- LED superstructure. It consists of a carbon fiber and 3D printed plastic structure specifically designed to carry a LED pattern above the vehicle, as shown in Figure 4.2.
- Truss structure. It is an aluminium structure that is placed besides the pool and extends above it holding the camera, which is facing down to the area where the ROVs operate.

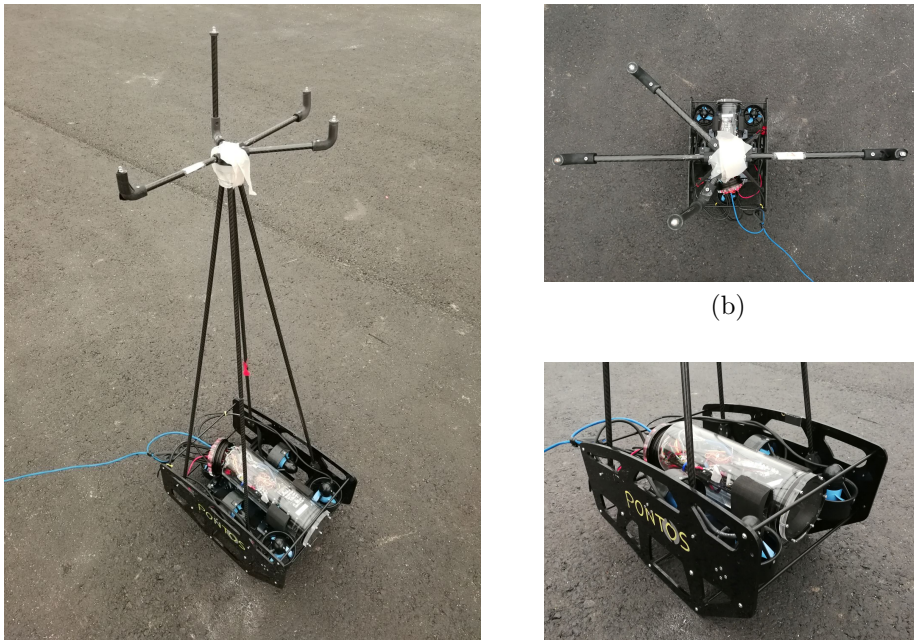

Figure 4.2: BlueROV with LED superstructure. (a) General overview (b) LED superstructure close-up (c) ROV close-up

Considering the field of view of the camera, the truss was mounted to have an altitude that would provide an area for tracking the LED superstructure of dimensions at least 5 meters long and 2.5 meters wide. This is illustrated in Figure 4.3

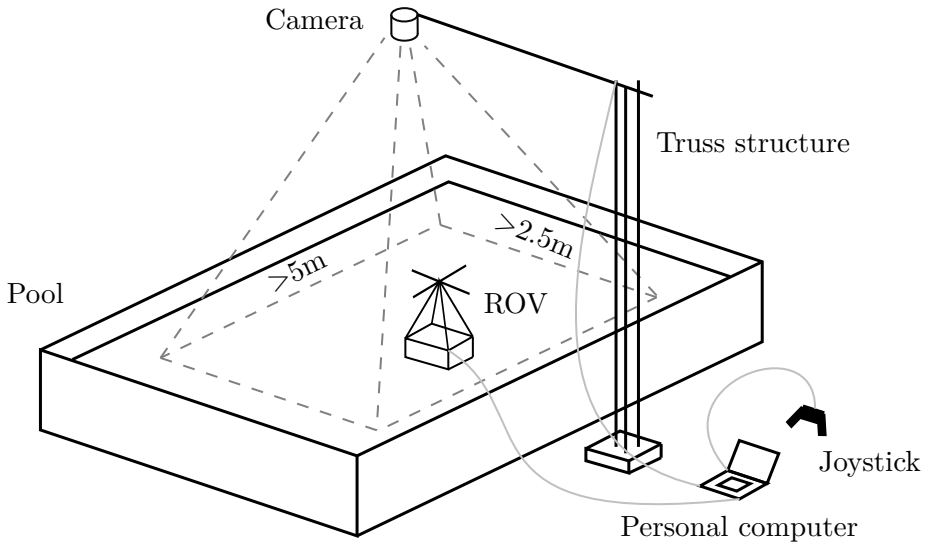

Figure 4.3: Representation of all the hardware components required in a typical experiment in the pool

Remote Operation

In order for a manual operator to interact with all the elements in the system, two components are used: a personal computer and a joystick. The personal computer is connected through Ethernet cable to the ROVs and through USB cable to the joystick and the camera of the MCS. The joystick is used to place the vehicles in the starting positions for the experiments while the personal computer is used as an intermediate node between the MCS, the joystick and the ROVs, and may assist in some of the computations needed for the control program.

4.2 Software

The software used can be mainly divided into two groups, one used for the simulation of the system and another for the final implementation and deployment in the vehicles.

Simulation Software

The proprietary software MATLAB® has been used for the simulation. It provides a working environment as well as a programming language which are aimed to address a wide range of numerical computing problems. It also offers several toolboxes —which are packages that add functionalities to the base software— and other extensions to address specific scientific and engineering domain needs.

In this project, the *Control System Toolbox* and the *Statistics and Machine Learning Toolbox* have been used. Part of the simulation has also been implemented with the Simulink® extension, which provides a block-diagram graphical environment to model and simulate dynamical systems.

Additionally, other MATLAB modules that have been developed by other entities have also been employed. These include a model of a single BlueROV vehicle whose physical parameters had been identified for the relevant degrees of freedom, including the non-linear (quadratic) drag terms; a set of functions for GNC kinematic transformations, and the LWPR library for MATLAB.

Vehicle Operation Software

The framework used for the implementation of the motion control system and for the communication with all the elements presented in the previous hardware section is *Robot Operating System* (ROS).

ROS acts as a middleware in a network of connected nodes and provides extensive functionalities for the communication between them. The nodes are constituted by individual computation units that provide functionalities or relevant information to the network. The code for these units is usually implemented in ROS “packages”, which

support several programming languages —in this project, C++ has been used—. Apart from the control system package that has been developed for this project and which includes a the controller nodes and a setpoint generator node, the rest of the packages had already been developed by other sources and have only suffered minor modifications to adapt some of their functionalities to the needs of this project. These include

- Motion capture package: provides the image processing functions and implements the kinematic Kalman filter node to obtain the linear velocities estimates.
- Joystick package: provides the node to communicate with the joystick.
- BlueROV package: provides several functions and services to communicate with the ROV and access its motors and sensors.

Additionally, the C++ library for the LWPR algorithm has also been used and adapted to this project, integrating it in the controller nodes.

4.3 Implementation

This section presents some implementation considerations for both the simulated system and the final vehicle control system.

The LWPR code was extended both cases to account for the cerebellar module capabilities presented in the previous chapters.

Simulation

The ROV models were simulated using the empirically identified physical parameters in [17] which include both linear and quadratic drag terms.

In order to account for the discrepancies between the real ROV dynamics and the model used by a controller when it is operating in the real vehicle, the simulation in this project also included differences between the model of the plant and the model used by the controller. Specifically, the dynamical model of the plant used the best available information, —this is, the full model with the non-linear quadratic drag terms—, whereas the models used by the control algorithm (in obtaining the full-state feedback controller gains $\boldsymbol{K}$ and in the guidance system to generate the trajectories), only considered up to the linear drag terms, and all the parameters were slightly tweaked with respect to the identified ones.

The noise in the sensors was implemented with random zero-mean Gaussian noise, and the standard deviation was chosen to match that of the real sensors (the IMU sensors and the motion capture system).

For simulating two rigidly connected ROVs, the methodology presented in [23] was implemented. However, an alternative, simpler method was also developed to obtain faster simulation speeds, and was used in most of the project. It consisted on simulating just one rigid body with altered mass and drag to match the properties and behavior of the system obtained with the aforementioned, complete method. The motors and sensor sources were also duplicated and positioned in the respective places.

For obtaining the gains of the full-state feedback controller, the *place* MATLAB command was used, which implements the pole placement method in [19]. Although these were later tuned manually as well, the initial method served as a way to find the right order of magnitude.

Real Vehicle System

The depth-hold functionality of the ROVs was achieved by using the services provided by the BlueROV ROS package that communicates with the ROV firmware to establish what is called "flight modes". These modes aim to assist several control scenarios needs, and the depth-hold mode is included among them. This mode stabilizes the ROV for roll and pitch, and controls the depth to keep it at a constant value with the help of the on-board depth sensor.

For the implementation of the control architecture for two vehicles, the design considered a ROS node to be executed by each vehicle, which processed their respective sensor data and sent the motor commands processed by a third node. This third node acted as a central unit that merged the filtered sensor data from the two ROVs and provided an allocated control action to each vehicle, according to Figure 3.8 in the previous chapter. Due to lack of time from the limitations menrioned in the first chapter, this dual-ROV control system ROS implementation could not be tested.

CHAPTER 5

Results

This chapter presents the main results obtained in a series of simulations and real experiments. In order to test the proposed control algorithm, several control scenarios are designed, with varying levels of complexity and requirements. The algorithm was first implemented in a single-vehicle simulation in order iterate and tune it faster, and was then adapted to the dual-vehicle simulation. In the real experiments, it was only possible to test it in one ROV, due to the limitations mentioned in Section 1.3 in the first chapter.

5.1 Control Scenarios

To assess the controller performance, a subset of the simulation experiments is performed with only one controlled degree of freedom. Focusing on one DOF allows to obtain one-dimensional metrics which can be easily interpreted and provide a good understanding on how the control algorithm behaves in the system. Apart from the main learning controller (LC), which includes the feedback term and the ULM, other controllers are also considered to contrast the results. These include a LC without the cerebellar module, a LC with the feedback gain $\boldsymbol{K}$ modulated by the confidence interval I_c and a classical PI controller which is tuned to have the same proportional gain as the LC and an integral gain that provides fast convergence of the tracked variable without introducing overshooting. The specific test cases considered for one DOF are:

- Step response. A step response analysis to evaluate the convergence characteristics of the tracked velocity is performed on the LC and the PI controller.
- Disturbance rejection. The ability of the proposed controller to adapt to disturbances in the plant is tested, and compared against the LC with no CM and the PI controller.

Additionally, three main metrics are measured to provide a numerical analysis of the performance.

- Standard deviation of the tracking error, $\sigma(\boldsymbol{e}_{\text{traj}})$, for a whole trajectory after the LC has completed an initial phase of learning. The vector $\boldsymbol{e}_{\text{traj}}$ is composed of the tracking errors for all time-steps in the trajectory, for the particular DOF tested. This provides a magnitude reference for the overall tracking error in the trajectory.

- L-infinity norm on the tracking error, $\|\boldsymbol{e}_{\text{traj}}\|_{\infty}$, for the same situation as the L2 norm. This shows the largest tracking error in the trajectory.
- Standard deviation $\sigma(\boldsymbol{u}_{\text{SS}})$ of the control action for a given period of time in steady state, $\boldsymbol{u}_{\text{SS}}$. This indicates how much noise is being amplified by the controller and a reference for its energy usage.

After this, the controlled degrees of freedom are increased to evaluate the ability of the ULM to learn a more complex function. Specifically, two cases are of special interest:

- In the case that Assumption 3.2 holds true —the motion of each DOF is uncoupled— check whether the LWPR output for each DOF only depends on the input kinematic variables for that specific DOF —the function learned should be insensitive to the rest of the input variables—. For example, for the LWPR controlling the surge motion, its output should only vary when the surge reference velocities and accelerations change, and not when the yaw or sway reference variables change. This can be tested in simulation by establishing diagonal inertia $\boldsymbol{M}$ and drag $\boldsymbol{D}$ matrices to ensure that Assumption 3.2 is met.
- In the case that Assumption 3.2 is not met —which may be the case in the real vehicles—, confirm that a change in the input vector of a single DOF actually affects several LWPR outputs —those corresponding to the DOF that are coupled—. For example, if the yaw motion is coupled with the surge motion, a change in the desired velocities and accelerations in surge should activate the LWPR models in both surge and yaw. The coupling can be ensured in simulation by adding off-diagonal terms to the matrices $\boldsymbol{M}$ and $\boldsymbol{D}$ to break Assumption 3.2.

The trajectories used for testing are classified into those that only involve one DOF and those that involve two. For a single controlled DOF, the trajectory is composed of step changes in velocity which increase in magnitude over time, in order to provide the ULM with a gradual approach in learning the dynamics function. This trajectory is generated by providing the velocity setpoints as a reference to the *guidance controller*, which outputs a velocity signal with smooth transitions between these setpoints.

For two DOF (specifically surge and yaw), a zigzag trajectory is generated. The guidance controller for surge is fed with velocity steps that increase in magnitude over time, but always remain positive (to ensure a forward zigzag motion). The yaw guidance controller receives a reference step signal which also increases in magnitude over time but alternates the sign every time the vehicle reaches a certain turned angle in the corresponding direction. This specific trajectory is common in marine vehicles to test the maneuvering capabilities, and it is also known as *Kempf's zigzag maneuver* [20].

All the previous tests are performed with the centralized control architecture which assigns a LC to the motion control of each DOF. Afterwards, an analysis of the distributed control structure presented in Section 3.2.4 is also included.

5.2 Simulation Results

5.2.1 1-DOF experiments

Controller Performance

The previously defined metrics are applied to three versions of the LC and two versions of a PI controller. The LC controller is tested with its base design, the design including the modulation of the feedback gain matrix $\boldsymbol{K}$ (a scalar K in the case of one controlled DOF) as a function of the confidence interval I_c of the LWPR output, and the base LC without the CM. The classical PI is evaluated with a version that has the same proportional gains as the base LC feedback module and another with higher proportional gains, with their respective integral gains adjusted to maximize tracking performance. The results of these tests are found in Table 5.1, which shows the statistical values of mean $\hat{X}$ and standard deviation σ_x for five executed simulations for each controller.

	$\sigma(\boldsymbol{e}_{\text{traj}})$ $[\text{m s}^{-1}]$		$\|\boldsymbol{e}_{\text{traj}}\|_\infty$ $[\text{m s}^{-1}]$		$\sigma(\boldsymbol{u}_{\text{SS}})$ [N]	
	$\hat{X}$ $\times 10^{-2}$	σ_x	$\hat{X}$	σ_x	$\hat{X}$ $\times 10^{-2}$	σ_x
LC base	3.46	1.41×10^{-2}	0.097	9.96×10^{-4}	6.49	5.96×10^{-3}
LC (no CM)	3.44	1.92×10^{-2}	0.091	3.19×10^{-3}	5.65	2.68×10^{-3}
LC $K(I_c)$	3.29	6.28×10^{-3}	0.108	1.01×10^{-3}	6.23	1.01×10^{-2}
PI equal gain	11.3	8.25×10^{-3}	0.336	2.18×10^{-3}	8.19	1.47×10^{-2}
PI high gain	8.58	1.04×10^{-2}	0.280	3.42×10^{-3}	9.02	5.38×10^{-3}

Table 5.1: Average and standard deviation values for the three metrics applied to the each controller

It can be observed that the LC in all its configurations provides better performance than the classical PI controller both in terms of trajectory tracking error $\boldsymbol{e}_{\text{traj}}$ for the whole trajectory and steady-state noise amplification from σ_u. Among the different configurations of the LC, both the base LC and the LC with no CM have similar tracking performance, but the noise amplification is larger in the base LC since the CM dynamics introduce additional frequency components in the control action. According to these results it might seem that the LC with no CM is preferable. However, these results were obtained from a simulation scenario with no disturbances. As it will be shown later, the CM is be crucial for disturbance rejection. Finally, the LC with K gain modulation provides a similar performance, with lower general tracking error but slightly higher maximum error given by the L-infinity norm. The value σ_u remains lower than the base LC. This has been achieved by setting a K_{max} slightly higher than the base K and a K_{min} slightly lower, which provides faster convergence for areas of the trajectory space which may have not been fully explored, while keeping a softer control action in steady state where the dynamics have been learned by the LWPR and the I_c is lower, thus diminishing σ_u.

Disturbance Rejection

Next, the capacity of the controller to reject disturbances in the plant is tested. The disturbance has been modeled as an ocean current that is applied to the vehicle following a step signal. The magnitude of the step is equal to the present vehicle velocity, 0.5 m/s, acting in the direction of motion. It is applied in steady state, after the LC has finished a previous phase of learning.

The controllers tested here are the base LC, the LC with modulated K, the LC with no CM and the PI with equal proportional gains as the LC.

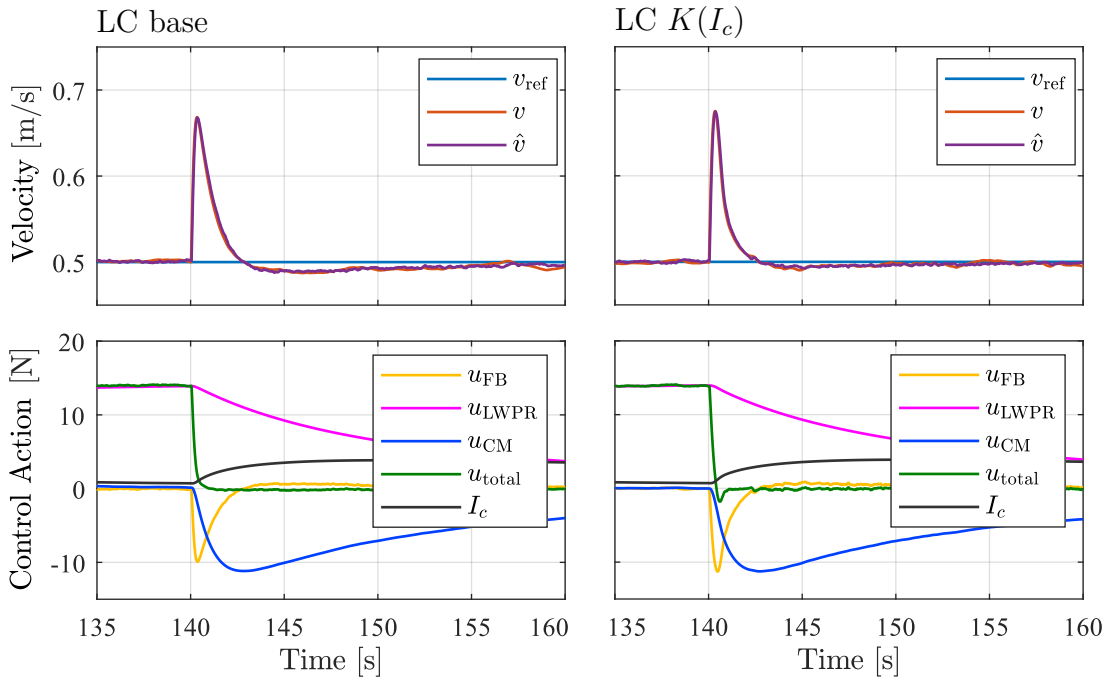

Figure 5.1: Disturbance response for the base LC (left) and the LC with modulated K (right)

As it can be appreciated in Figure 5.1, the CM provides the required control action (u_{CM} in blue) to adapt rapidly to the disturbance and bring the tracked variable back to the level specified by reference. This signal provided by the cerebellar module is then slowly incorporated into the LWPR module, which in the long term learns the new function to be represented. In the case of removing the CM (Figure 5.2 left), the LC fails to adapt in an acceptable amount of time to the disturbance. These figures clearly show the nature of both the LWPR and the CM, and the importance of the later to provide short-term adaptation.

In Figure 5.1 it is observed that the K gain modulation endows the LC with faster adaptation against disturbances, with the settling time in the base LC being over 2.6 seconds —with longer time to remove offset in velocity observed after the initial peak— and in the LC $K(I_c)$ under 2.4 seconds —with faster offset removal. The confidence interval I_c of the LWPR output is shown in black and, as it grows, it increases the K

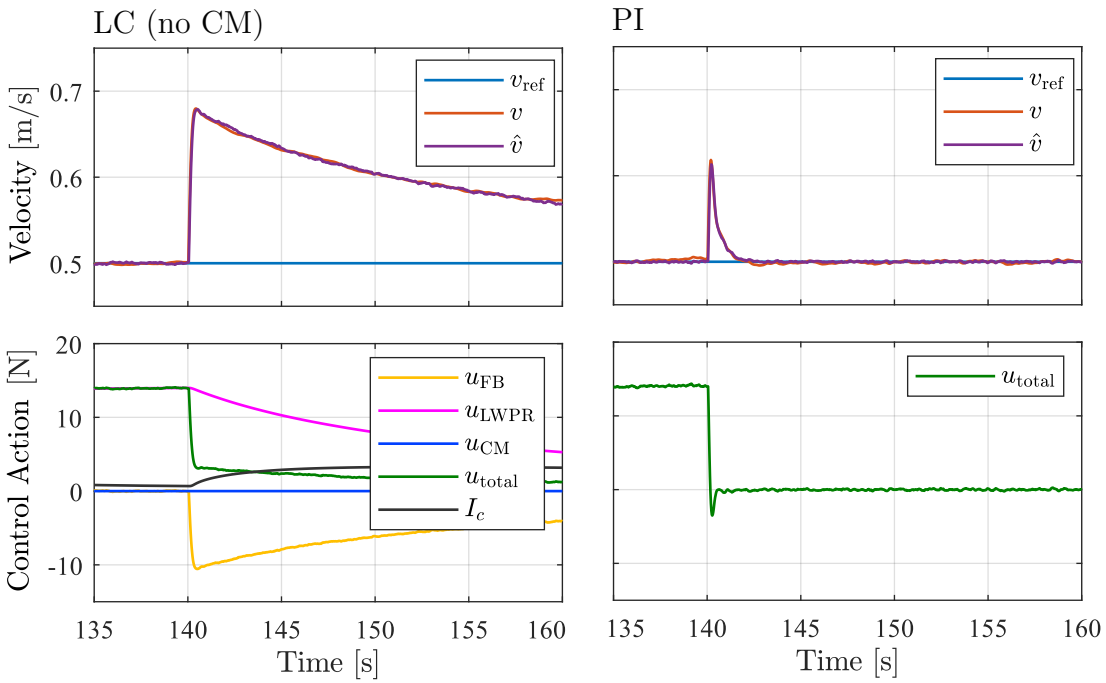

Figure 5.2: Disturbance response for the LC with no CM (left) and the PI controller (right)

gain which in term accelerates the convergence of the total feedforward action to the value required to cancel the disturbance. This confirms the established statements in the last part of Section 3.2.3 of Chapter 3 about the role played by the K gain in the convergence of u_{FF} and proves that the I_c signal can be used to alter the update rules of the LWPR and the CM.

Finally, the PI controller shows an overall faster convergence and lower amplitude of the velocity variation induced by the disturbance. This is due to the fact that the PI has been tuned to provide a fast response to changes in velocity, to attempt to match the performance of the LC in normal trajectory tracking conditions. Nevertheless, LC could also be tuned to reduce the convergence time in front of disturbances by increasing the update rate β of the cerebellum or the magnitude of feedback gain matrix K, but this would increase as well the noise amplification measured by σ_u. A compromise could be achieved according to the specific needs of the particular mission of the vehicles.

Step Response

Now, the response of the system after a change in the velocity reference is analyzed. Since the step is applied in the guidance system as a velocity setpoint, the reference signal provided to the plant is a smooth function, thus not corresponding to a pure step. This analysis does not consider a pure step since the LC has not been designed to follow abrupt changes in the tracked variable. Rather, it expects a smooth function in the

tracked velocity, and the corresponding time-derivative function. These are provided by the GS and their specific profile can be shaped with the choice of the GS parameters. However, an analysis of the LC performance without the time-derivative as an input to the feedforward module will be provided, to evaluate the its influence in the LC behavior.

The controllers tested in this section are only the base LC and the PI with equal proportional gain. This is because the behavior of the LC under normal operation conditions (trajectory tracking with no disturbances) does not vary significantly between the versions presented before.

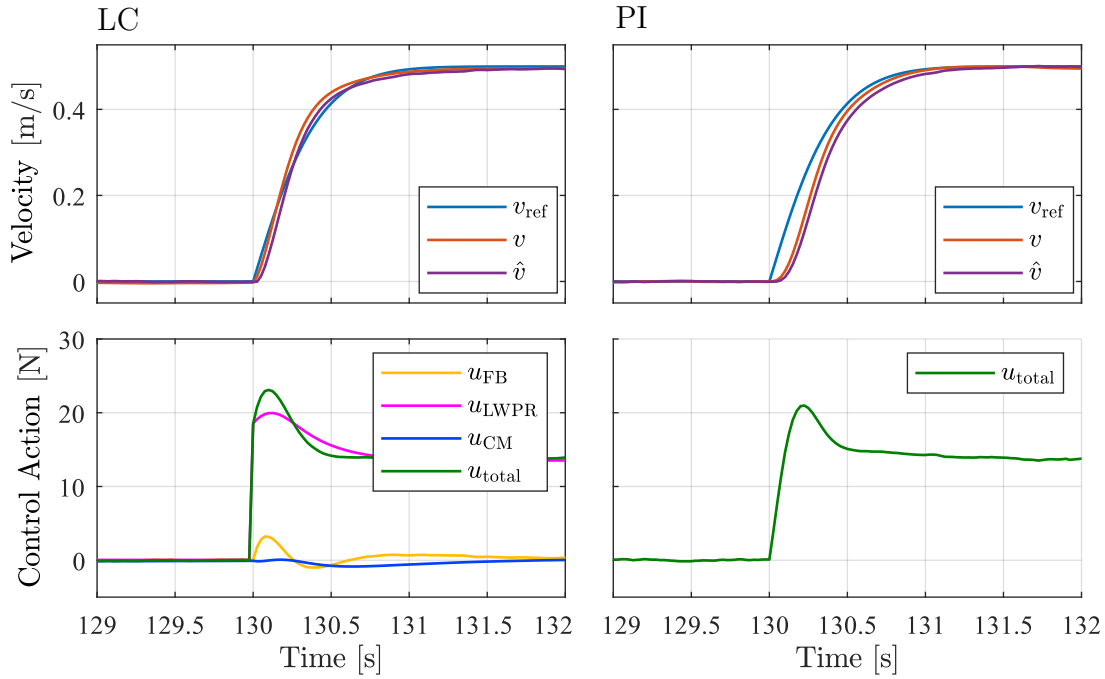

Figure 5.3: Step response for the base LC (left) and PI (right)

Figure 5.3 shows the smooth-step response of both the LC and the PI controllers. It is clearly seen that the LC provides a significant control action immediately after the reference velocity starts to rise. This corresponds to the feedforward control action, provided mostly by the LWPR, which matches the required force to set the vehicle to the desired velocity according to the inverse dynamics model. Indeed, this control signal makes the vehicle velocity follow the reference closely, as can be observed. On the other hand, the PI response lags behind the reference signal, since there exists an inherent delay in a feedback-based controller to settle after a reference change.

The LWPR is able to provide the instant feedforward control action due to the instant change in its intput corresponding to the desired acceleration of the vehicle (time-derivative of the tracked variable). In fact, the profile of this signal, together with the velocity, is shown in Figure 5.4 (right). If the acceleration is removed as an input to the LWPR for that particular step change, the plant response takes the form shown in Figure 5.4 (left).

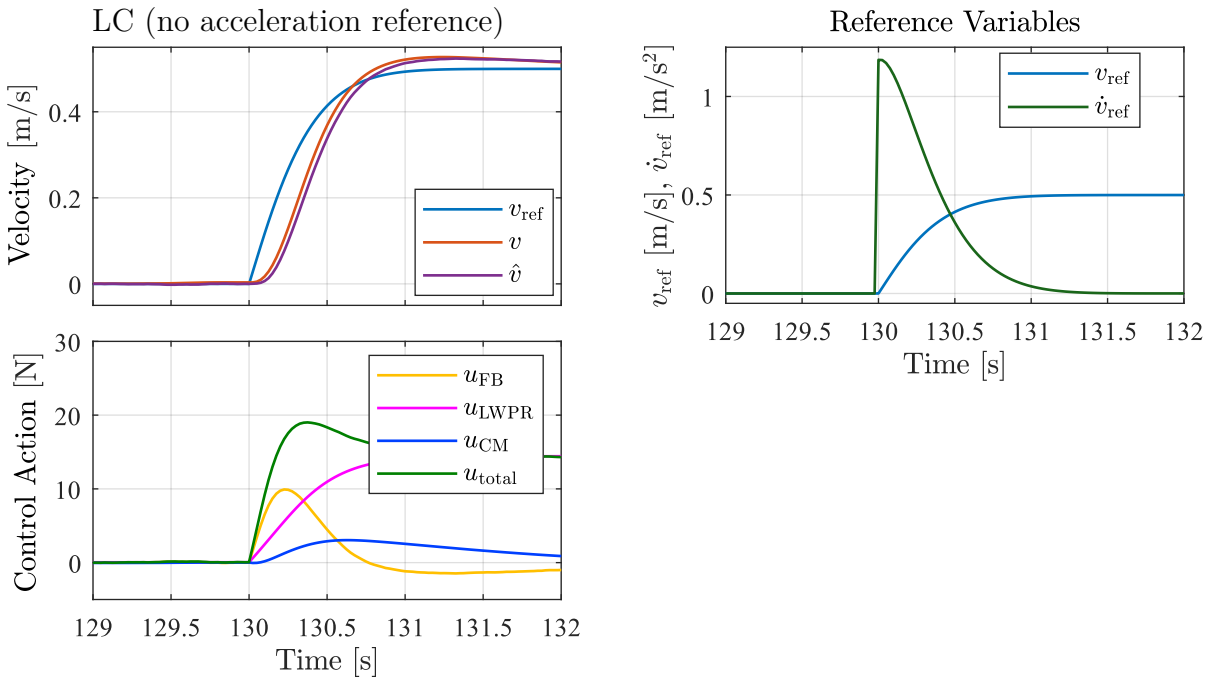

Figure 5.4: Step response for the LC with no acceleration reference (left) and reference variables (right)

This shows that the time-derivative of the reference variable —$\dot{v}_{\text{ref}}$ in Figure 5.4 (right)— is crucial for the LWPR to provide the feedforward control action, since this action represents the inverse dynamics function of the plant, which is dependent on the acceleration.

5.2.2 2-DOF experiments

Uncoupled Motion and LWPR input-output Independence

The zigzag trajectory was used to test the question of whether, for a vehicle with uncoupled motion in all DOF, the LWPR output for a particular DOF is independent of the inputs for the rest of DOF. As stated in previous chapters, the LWPR inputs are the reference velocities and accelerations for all controlled DOF, since it is desirable that the function learned can capture the interdependecies in motion between all DOF (the coupling effects). However, if only one DOF input is excited, only the LWPR output assigned to that DOF should be excited.

In this zigzag trajectory, the velocity reference for the surge motion was kept at a constant value for the entire path, whereas the yaw velocity reference was gradually increased in magnitude and was periodically switched in sign to produce the zigzag motion.

The results are shown in Figure 5.5. The first 30 to 40 seconds represent the initial

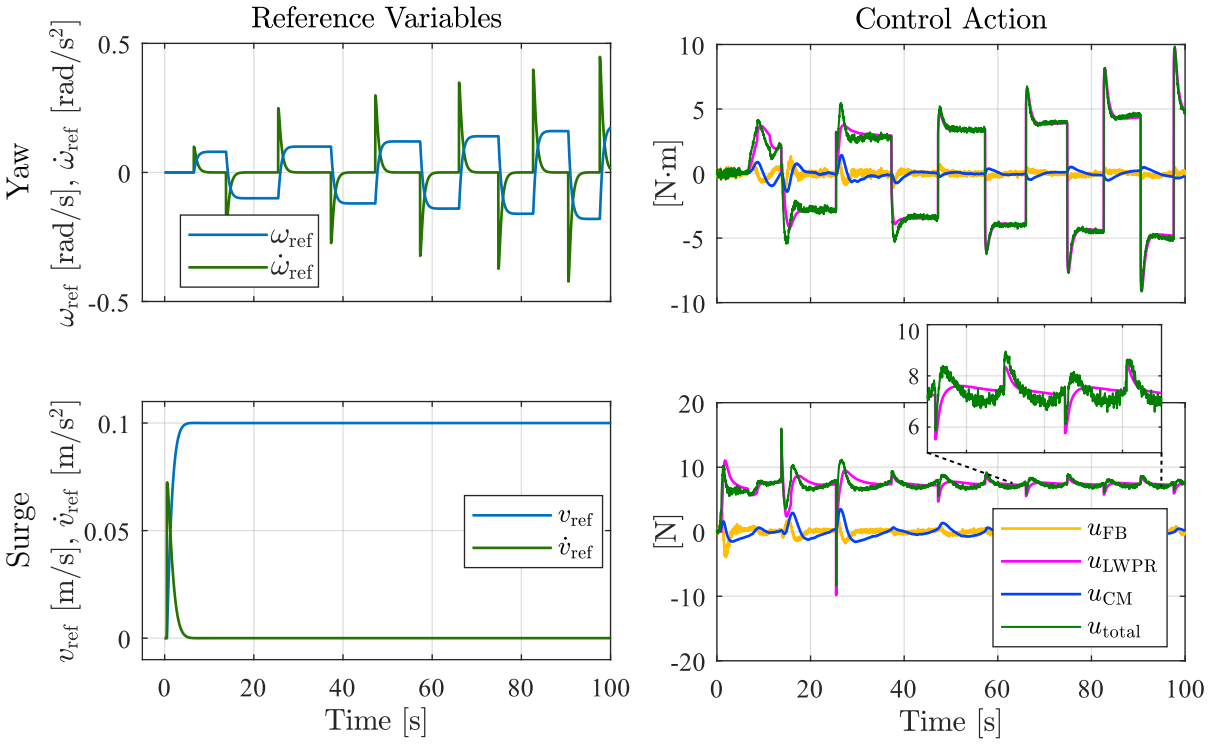

Figure 5.5: Uncoupled system analysis. The top plots correspond to the yaw motion and the bottom plots to the surge. The left plots show the reference variables for each these DOF and the right plots the control action. The bottom-right plot has a floating window that shows a detail of the LWPR output and total control action from time 65s to 95s

learning phase, in which the LWPR output still does not approximate correctly the inverse dynamics model and presents some random fluctuations. As it can be observed, despite the reference for the surge being set to a constant velocity, the output of the LWPR controlling this DOF reacts to the reference changes in the yaw motion. The tendency of this LWPR output is to converge towards a stable value that makes the surge velocity be steady, as it can be observed from the diminishing magnitude of the peaks over time in the u_{LWPR} signal at the bottom-right plot. However, this is not a desirable behavior in a real control scenario.

It is hypothesized that this outcome is produced by the fact that, when the first change in the yaw reference variables is produced, the teaching signal for the surge LWPR may not correspond to the stable value that represents the true inverse dynamics function —either because the LWPR has not learned this function yet or because any random component has been added to the target function (total control action), via noise amplification from the feedback controller or the CM. Because of this, and considering that the LWPR algorithm assigns more credit to the first data points gathered in a new explored region of the input space, any bias introduced in this initial data will be

captured by the LWPR for the long term for this input space region. Even though this issue has not been addressed in this project, two possible approaches to solve it are proposed.

First, in order to ensure that the learned LWPR function is stable for a particular DOF when a change in the reference of another DOF is introduced, the algorithm should be given more time to consolidate the function for the first DOF. This would correspond in this case to delaying the yaw reference changes further in time.

Second, since even in the case that the LWPR function has been properly learned any random noise component could be added to the reference function, the commonly used technique in machine learning of batch training could be applied here. Indeed, even though it has not been explored in this project, the LWPR offers the possibility of fitting its function via batches of data. This would average out any random component in the target function and would avoid the problem of assigning too much credit to the first set of data points in a new input space region.

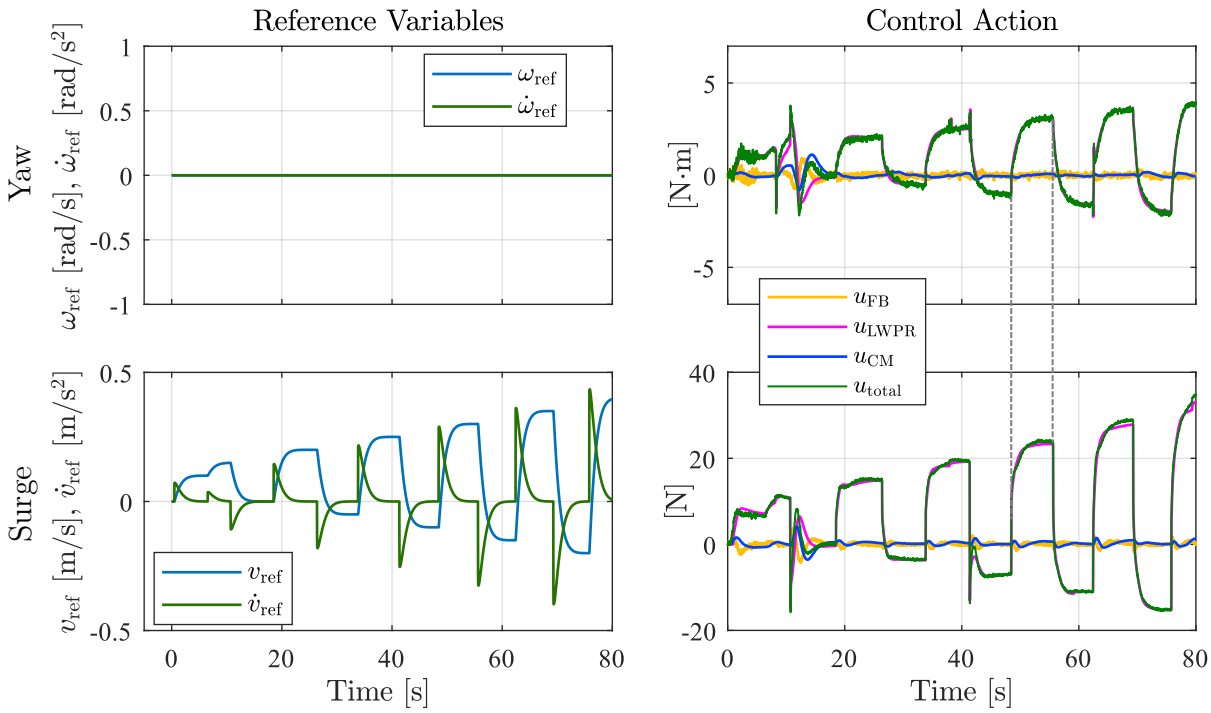

Figure 5.6: Coupled system analysis. The top plots correspond to the yaw motion and the bottom plots to the surge. The left plots show the reference variables for each of these DOF and the right plots the control action.

Coupled Motion and LWPR input-output Dependence

To test the ability of the LWPR to learn the relationship between different coupled DOF, it is proposed to use a straight trajectory with only changes in the surge reference. The

coupling is set from the surge motion to the yaw motion through the drag matrix $\boldsymbol{D}$, such that a forward motion induces an anti-clockwise (negative in the z axis) rotation. Therefore, for a positive reference velocity in surge, the output of the LWPR for yaw should also be positive to counter the coupling effect. The results are shown in Figure 5.6.

These results prove that the LWPR algorithm can learn a representation of the dynamics model in which coupled motion exists. It is observed that for a positive reference velocity (and control action) in surge, the control action in yaw is also positive, thus confirming the statement from the previous paragraph.

Distributed Control Architecture

The distributed control architecture presented in Section 3.2.4 shows a similar behavior in motion as the previous results. It has been applied to the case of two vehicles connected on their sides, similarly to the diagram shown in Figure 3.11 (a). An observation made in the simulations is that since multiple ULMs are involved in the motion control of a given DOF, the combination of all their outputs provides the adequate control action fur such DOF but their particular outputs do not correspond to the optimally allocated control action for their corresponding motors.

This is, given a total control action for each motor (which is the sum of the ULM output and the proportional feedback control action allocated for that motor, as shown in Figure 3.8) represented by the vector $\boldsymbol{\rho}$, the corresponding force or moment applied to each DOF is determined by Equation 3.65, restated here $\boldsymbol{\tau} = \boldsymbol{H}\boldsymbol{\rho}$.

This total control action $\boldsymbol{\tau}$ is provided correctly by the distributed control system to achieve the desired trajectory tracking performance. Since the system is overactuated, infinite solutions for $\boldsymbol{\rho}$ exist which provide the same control action $\boldsymbol{\tau}$, the optimal one being that determined by $\boldsymbol{\rho} = \boldsymbol{H}^{+}\boldsymbol{\tau}$. However, the distributed structure usually provides a set of solutions for $\boldsymbol{\rho}$ which do not correspond to the optimal one, since each ULM learns an independent function that evolves on its own, and the teaching signal ultimately comes form the tracking error $\boldsymbol{e}$ (as shown in Equation 3.60, which can be minimized for any $\boldsymbol{\rho}$ that provides the right $\boldsymbol{\tau}$.

For this reason, even though the distributed control architecture may present a more biologically plausible solution to the motion control problem, in the current formulation there is not any considerable advantage of using it over the centralized architecture. However, it provides a first step towards a more complete and comprehensive solution which may include additional mechanisms to address the optimization of $\boldsymbol{\rho}$, and which may constitute a more biologically faithful architecture.

5.3 Experimental Results

As mentioned before, the experimental validation of the LC was performed with one vehicle and one controlled degree of freedom. A pattern of alternating velocity references in surge with varying magnitude was used. Several experiments were performed to tune the controller parameters and the results shown here represent an example of the final version achieved. Additionally, a PI controller was also tuned to provide a similar tracking performance as the LC, which required to increase the proportional gain to four times as much as the feedback gain of the LC, and the integral gain was adjusted to converge fast with no overshooting.

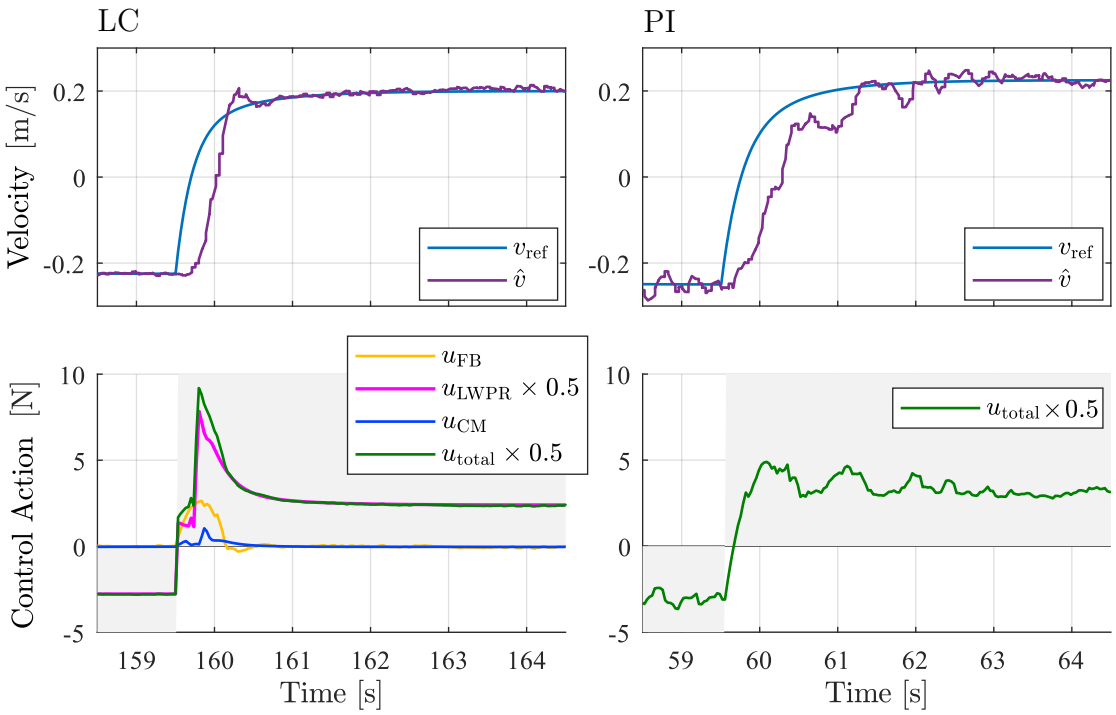

Figure 5.7: Comparison of the learning (LC) and the PI controllers upon a change in the velocity reference, tested in the real vehicle

In Figure 5.7 the response of the LC after a previous training period is shown on the left, and it is compared against the PI on the right. The magnitudes of both the LWPR control action u_{LWPR} and the total control action u_{total} are represented as 50% of their original values to bring them to a similar scale as u_{CM} and u_{FB} in the plot. The gray areas represent the time period and control action (force) sign required to move the vehicle in the direction specified by the reference velocity.

As it can be observed, the LC offers a faster convergence than the PI while requiring lower feedback gains. This difference in the gains can also be appreciated in the profile of the total control action, which is significantly smoother in the LC than in the PI after the initial transient period, meaning that less noise is amplified in the feedback loop.

Another characteristic observed in the LC is that it induces a slight overshooting in the vehicle velocity respect to the desired values. The reason for this is that, as will be shown next, there exists a delay between the control commands and the vehicle motion response, which affects how the LC learns its feedforward function. Specifically, the delay translates into a higher tracking error during the initial transient of a change in the velocity reference. This error is amplified by the proportional feedback controller, which adds to the total control action that constitutes the teaching signal of the LWPR. Thus, during the delay phase, the total control action increases in every iteration, since the LWPR keeps incorporating into its model this additional control signal —this can be observed in Figures 5.7 and 5.8, where the total control action is larger than the LWPR one during the transient. Over a period of training time, this shows as a feedforward control action that exceeds that required by the plant during the transient in the velocity change, producing the observed overshooting

The delay is shown in detail in Figure 5.8. Here, it is observed that after the first control action in the surge direction (positive force in surge) is commanded by the controller, the vehicle takes about 0.19 seconds to start moving in such direction. A method to deal with delays in the plant will be proposed in the future work section.

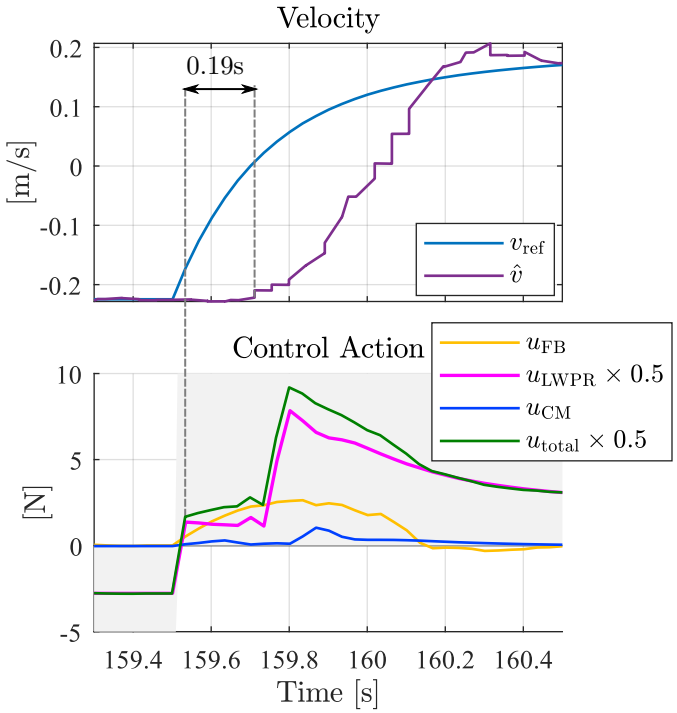

Figure 5.8: Detail of the LC controller upon a change in the reference signal showing the delay in the plant response

Finally, the tracking performance of both the LC and the PI is also analyzed numerically by applying both the σ_e and $\|\boldsymbol{e}_{\text{traj}}\|_\infty$ metrics to a trajectory executed over 30 seconds. Figure 5.9 shows the tracking error for both controllers over such period of time, and Table 5.2 shows the metrics. It is observed that the delay affects similarly

both the LC and the PI controller, with slightly higher deviations in the LC (as per the $\|\boldsymbol{e}_{\text{traj}}\|_\infty$ metric). However, the LC presents a better performance overall according to σ_e, and it is again observed in Figure 5.9 that the LC converges faster and provides a more steady tracking of the velocity. The higher value of $\|\boldsymbol{e}_{\text{traj}}\|_\infty$ in the LC might be caused by an incomplete learning of the feedforward action by the LWPR. An earlier, higher control action from the LWPR, specially in the 159.6s to 159.8s time period in Figure 5.8, would provide an higher velocity response in the vehicle during the first instants of the transient.

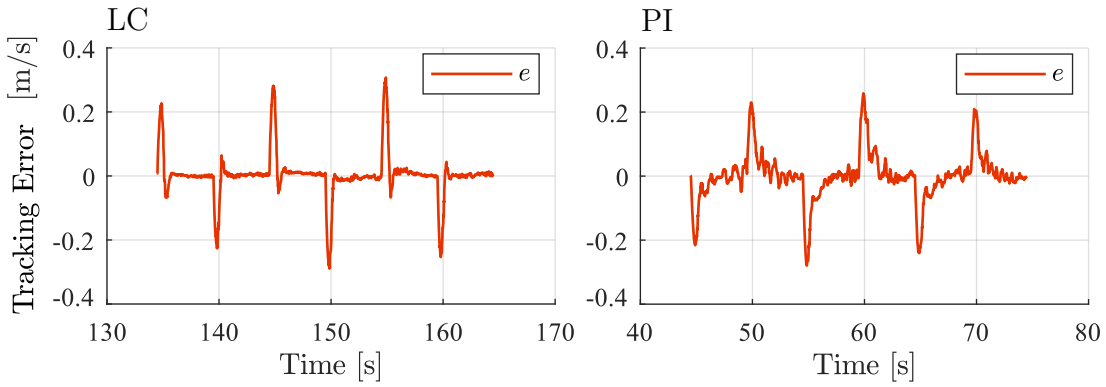

Figure 5.9: Tracking error for the LC and the PI controllers for a period of 30 seconds

	$\sigma(\boldsymbol{e}_{\text{traj}})$ $[\text{m s}^{-1}]$	$\|\boldsymbol{e}_{\text{traj}}\|_\infty [\text{m s}^{-1}]$
LC	7.55×10^{-2}	0.2796
PI	6.92×10^{-2}	0.3063

Table 5.2: Tracking metrics for a trajectory of 30 seconds executed in the real vehicle, for the LC and the PI controllers

CHAPTER 6

Discussion

The main results and considerations from the previous chapter are highlighted here.

The proposed learning controller proves effective to learn an inverse dynamics model of the plant, represented by the ULM. This model is represented as a feedforward module in the control structure, which allows the LC to provide better trajectory tracking accuracy than a PI controller that has higher feedback gains, thus reducing as well the noise amplification in the feedback loop.

The cerebellar module shows essential to deploy the LC in a real scenario such as the ocean, in which disturbances in the form of currents are constantly affecting the vehicle and thus fast adaptation is required.

The LC version incorporating the $\boldsymbol{K}$ gain matrix modulation as a function of I_c has demonstrated that it can be advantageous in some situations, since the $\boldsymbol{K}(I_c)$ function shown in Equation 3.61 can provide lower gains when the learned model has low uncertainty —thus reducing the noise amplification in the feedback loop— and higher gains when a permanent disturbance or change in the model appears, which allows the ULM to learn this model faster, as shown in Equation 3.60. In general, a proper choice of the $\boldsymbol{K}$ matrix, even if it remains constant, is crucial to ensure a balance between fast feedforward model convergence and low noise amplification in the feedback loop.

Finally, in the case of controlling several degrees of freedom, some undesired interactions appeared between motion directions that were not coupled, since the LWPR algorithm did not capture the function correctly in the initial stages of learning. Solutions were proposed involving mainly delayed training between different motion directions and batch training.

Next, the research questions formulated in the Introduction chapter are revisited and analyzed according to the presented results.

Can a biologically-inspired controller in the scope of AUVs learn a relevant dynamics model such that it outperforms other traditional techinques in terms of trajectory tracking? It has been proven both in simulation and in real experiments that the proposed LC successfully learns and takes advantage of the inverse dynamics model of the plant, which allows it to outperform traditional feedback control approaches such as a PI controller.

Can this controller learn a relevant model in a short enough period of time such that it does not hinder the operation of the vehicles in an IMR task? The LWPR algorithm

—the part of the ULM that consitutes the long them model learning— presents fast converge properties with respect to other machine learning methods, especially during the initial training period in a region of its input space. This makes it suitable to obtain a relevant model of the plant in a short amount of time, allowing the vehicles to switch earlier to their assigned tasks or even learn part of the model while performing them. The specific convergence rate can be adjusted according to 3.60 to meet the required needs.

Can the learning controller be designed to be robust against external disturbances and changes in the dynamical model? As shown in the results, the cerebellar module handles the short-term tracking errors induced by sudden disturbances, while the LWPR assimilates the new model to be represented in the long term in the case of persistent disturbances, which may correspond to changes in the dynamical model of the vehicles. Again, the convergence speed of both the CM and the LWPR can be modulated as desired according to Equation 3.60.

Can the learning controller provide some level of fault tolerance? Although this question was not fully explored, it constituted part of the motivation to develop the distributed control architecture. An enhanced and truly distributed control architecture that individually optimized the function of each thruster's ULM according to a global control objective (such as the mimimization of the trajectory tracking error and the total control action), would be able to adapt each thruster's function in the event of a motor failure or capacity loss. The new functions learned by the ULMs could be analyzed to find the source of the failure.

CHAPTER 7

Conclusions and Future Work

The novelty of the problem addressed by this project endows it with a research nature, since no prior solutions had been proposed to address the motion control of multiple rigidly-connected vehicles in a real-life working environment. The poject started with a review of the literature on dynamics-model learning techniques, which found previous publications on learning controllers for single underwater vehicles that were generally tested in more simplified and less demanding scenarios. Other techniques were also found for different robotics applications, and of special interest were the biologically-inspired ones, since they provided the desired properties of fast learning and no overfitting.

With this, the proposed learning controller was designed, which was constituted by a feedback module and a feedforward learning module represented by a Unit Learning Machine [30]. An analysis was made of its dynamical properties, which allowed to acquire insights into its convergence capabilities and provided a way to alter them by adjusting several parameters such as the feedback gain matrix $\boldsymbol{K}$, the cerebellar module learning rate β and the LWPR update parameter λ_f. A distributed control architecture was also proposed, which mimicked more faithfully the human brain motor structure.

In order to test the effectiveness of the proposed learning controller, several control scenarios were designed, both in simulation and real experimental facilities. In simulation, a model of a single vehicle and a model of two connected vehicles were implemented in MATLAB, which allowed for extensive tuning and design improvement based on iterative testing. The results showed that the proposed solution considerably outperformed traditional approaches such as a PI controller with high gains in terms of trajectory tracking. However, an undesired behavior in the LWPR algorithm was observed when multiple controlled degrees of freedom interacted in the vehicle motion. Two solutions were proposed, which will be revisited in the next future work section.

For the experimental validation, a full implementation of the learning controller was developed in ROS, both for a single controlled vehicle and for multiple connected vehicles. However, due to the stated limitations, only the single vehicle algorithm could be tested. The results in this case were also promising, outperforming as well a high gain PI controller in the trajectory tracking problem. A delay in the system was observed, which induced an undesired behavior in the learning controller, although it did not hinder its ability to perform the control task. The proposed LC was not designed to

deal with delays in the controlled plant, and therefore an extension is proposed in the future work section.

7.1 Future Work

This section addresses the aspects of the project that could not be fully completed or that were identified during the solution implementation as possible components to be improved in the future.

The distributed control architecture, as previously stated, does not provide a truly distributed approach since the information used by the learning modules is by default established by a centralized source, namely the global feedback controller. The simulation results obtained with this structure did not provide any advantage either with respect to the centralized control structure, and showed a sub-optimal solution to the functions represented by the learning modules. Therefore, it is proposed that, to fully harness the properties of a distributed control architecture, a more comprehensive design is provided. In this new approach, each learning module function would be optimized to minimize the trajectory tracking error of the vehicle formation and the total control action used. This could also provide a means for fault tolerance capabilities, as discussed in the previous chapter.

The LWPR in a multiple DOF control setting, as shown in the results, presented an undesired behavior by which the output of a LWPR module assigned to control a given DOF, reacted to the reference commands of other DOF that were not coupled in motion. In order to tackle this, the possibility of using batch training for learning the LWPR functions arised. This is a direction worth exploring since it is an already implemented functionality in the LWPR algorithm, and could provide better learning and convergence capabilities for a general setting, and specifically for the case of multiple controlled DOF.

The time delay from a control command to the vehicle motion response found in the experimental tests impaired to a certain extent some of the learning controller capabilities, since its design did not account for such delays. In order to address this issue, the Smith predictor [28] control structure is suggested, which incorporates a model of the plant delay in the control loop.

The ROS implementation of the control system for two connected vehicles should be tested to validate the effectiveness of the proposed controller to learn the inverse dynamics function in a more complex setting.

Finally, two other ways to expand the project are also suggested. On the one hand, the LWPR algorithm allows to save the learned model in a format that can be ported to other devices that also make use of the LWPR libraries. It would be of interest to test the viability of learning a LWPR model in simulation which would then be deployed to the vehicles which, depending on the accuracy of the simulated vehicle models, would

allow to skip or reduce the initial learning phase of the vehicles in a real working scenario. On the other hand, the opposite would also show very convenient. That is, learning the models in the real vehicles and then translating then into a simulated environment to perform other model validation activities or tasks planning in preparation for a new mission. The inconvenient here is that the LWPR model represents an inverse dynamics model of the plant, whereas for replicating the behavior of the vehicles in simulation, a forward model would be needed.

Appendices

APPENDIX A

Supplementary Material

Further resources regarding code, videos or other data generated in this project may be published at `https://sites.google.com/view/adria-msc-thesis` or at `https://github.com/neutrinum`.

Bibliography

[1] Aguiar, A Pedro and Hespanha, Joao P. "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty". In: *IEEE transactions on automatic control* 52.8 (2007), pages 1362–1379.

[2] Albus, J. S. "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)". en. In: *Journal of Dynamic Systems, Measurement, and Control* 97.3 (September 1975), pages 220–227.

[3] Albus, James S. "A theory of cerebellar function". en. In: *Mathematical Biosciences* 10.1-2 (February 1971), pages 25–61.

[4] *Blue Robotics - Underwater ROVs, Thrusters, Sonars, and Cameras*. en-US. URL: `https://bluerobotics.com/` (visited on August 6, 2020).

[5] Capolei, Marie Claire et al. "A Biomimetic Control Method Increases the Adaptability of a Humanoid Robot Acting in a Dynamic Environment". In: *Frontiers in Neurorobotics* 13 (August 2019), page 70.

[6] Comoglio, Rick F. and Pandya, Abhijit S. "CMAC neural network architecture for control of an autonomous undersea vehicle". In: edited by Rogers, Steven K. Orlando, FL, September 1992, page 517.

[7] *Control system | technology*. en. URL: `https://www.britannica.com/technology/control-system` (visited on August 6, 2020).

[8] Cruz, Joseph Sun de la, Kulić, Dana, and Owen, William. "Online Incremental Learning of Inverse Dynamics Incorporating Prior Knowledge". In: *Autonomous and Intelligent Systems*. Edited by Kamel, Mohamed et al. Volume 6752. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pages 167–176.

[9] Cruz, Joseph Sun de la, Owen, William, and Kulic, Dana. "Online learning of inverse dynamics via gaussian process regression". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2012), pages 3583–3590.

[10] Csató, Lehel and Opper, Manfred. "Sparse On-Line Gaussian Processes". en. In: *Neural Computation* 14.3 (March 2002), pages 641–668.

[11] De La Cruz, Joseph, Kulic, Dana, and Owen, William. "Learning inverse dynamics for redundant manipulator control". In: *2010 International Conference on Autonomous and Intelligent Systems, AIS 2010}* (2010), pages 1–6.

[12] Eski, İkbal and Yildirim, SCahin. “Design of Neural Network Control System for Controlling Trajectory of Autonomous Underwater Vehicles”. en. In: *International Journal of Advanced Robotic Systems* 11.1 (January 2014), page 7.

[13] Evarts, E. V. “The Cerebellum as a Neuronal Machine. John C. Eccles, Masao Ito, and Janos Szentagothai. Springer-Verlag, New York, 1967. 343 pp.” en. In: *Science* 158.3807 (December 1967), pages 1439–1440.

[14] Fagogenis, Georgios, Flynn, David, and Lane, David M. “Improving Underwater Vehicle navigation state estimation using Locally Weighted Projection Regression”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, May 2014, pages 6549–6554.

[15] Fossen, Thor I. *Handbook of Marine Craft Hydrodynamics and Motion Control: Fossen/Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: John Wiley & Sons, Ltd, April 2011.

[16] Fujita, M. “Adaptive filter model of the cerebellum”. en. In: *Biological Cybernetics* 45.3 (October 1982), pages 195–206.

[17] Hansen, Andreas Baldur Nørregård. “System Identification and Dynamic Positioning of Autonomous Underwater Vehicle”. Master’s thesis. Denmark: Danmarks Tekniske Universitet, 2016.

[18] Jeon, Soo, Tomizuka, Masayoshi, and Katou, Tetsuaki. “Kinematic Kalman filter (KKF) for robot end-effector sensing”. In: *Journal of dynamic systems, measurement, and control* 131.2 (2009).

[19] Kautsky, J., Nichols, N. K., and Van Dooren, P. “Robust pole assignment in linear state feedback”. en. In: *International Journal of Control* 41.5 (May 1985), pages 1129–1155.

[20] Kempf, G. “Measurements of the propulsive and structural characteristics of ships”. In: *Transactions of SNAME* 40 (1932), pages 42–57.

[21] Liu, Furong and Chen, Hui. “Motion control of intelligent underwater robot based on CMAC-PID”. In: (2008), pages 1308–1311.

[22] Marr, David. “A theory of cerebellar cortex”. en. In: *The Journal of Physiology* 202.2 (June 1969), pages 437–470.

[23] Nielsen, Mikkel Cornelius et al. “Constrained multi-body dynamics for modular underwater robots — Theory and experiments”. en. In: *Ocean Engineering* 149 (February 2018), pages 358–372.

[24] Ojeda, Ismael Baira, Tolu, Silvia, and Lund, Henrik H. “A scalable neuro-inspired robot controller integrating a machine learning algorithm and a spiking cerebellar-like network”. In: *Springer. Conference on Biomimetic and Biohybrid Systems* (2017), pages 375–386.

[25] Olivier, J-MD Coenen et al. “Parallel fiber coding in the cerebellum for life-long learning”. In: *Autonomous robots* 11.3 (2001), pages 291–297.

[26] Porrill, John, Dean, Paul, and Stone, James V. "Recurrent cerebellar architecture solves the motor-error problem". In: *Proceedings of the Royal Society of London. Series B: Biological Sciences* 271.1541 (2004), pages 789–796.

[27] Siciliano, Bruno and Khatib, Oussama. *Springer handbook of robotics*. Springer, 2016.

[28] Smith, Otto J. M. "Closer Control of Loops with Dead Time". In: *Chemistry Engineering Progress* 53.5 (1957), pages 217–219.

[29] Tolu, Silvia et al. "A Cerebellum-Inspired Learning Approach for Adaptive and Anticipatory Control". en. In: *International Journal of Neural Systems* 30.01 (January 2020), page 1950028.

[30] Tolu, Silvia et al. "Bio-inspired adaptive feedback error learning architecture for motor control". en. In: *Biological Cybernetics* 106.8-9 (October 2012), pages 507–522.

[31] Ven, Pepijn W.J. van de, Flanagan, Colin, and Toal, Daniel. "Neural network control of underwater vehicles". en. In: *Engineering Applications of Artificial Intelligence* 18.5 (August 2005), pages 533–547.

[32] Vijayakumar, Sethu and Schaal, Stefan. "Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space". In: *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*. Volume 1. 2000, pages 288–293.

[33] Williams, Grady et al. "Locally Weighted Regression Pseudo-Rehearsal for Online Learning of Vehicle Dynamics". In: *arXiv preprint arXiv:1905.05162* (2019).

[34] Wu, Hui et al. "End-to-end sensorimotor control problems of AUVs with deep reinforcement learning". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pages 5869–5874.

[35] Wulff, Peer et al. "Synaptic inhibition of Purkinje cells mediates consolidation of vestibulo-cerebellar motor learning". en. In: *Nature Neuroscience* 12.8 (August 2009), pages 1042–1049.

[36] Yuh, J. and Gonugunta, K.V. "Learning control of underwater robotic vehicles". In: *[1993] Proceedings IEEE International Conference on Robotics and Automation*. Atlanta, GA, USA: IEEE Comput. Soc. Press, 1993, pages 106–111.

[37] Zhao, Jiemei. "Neural Network Predictive Control for Autonomous Underwater Vehicle with Input Delay". en. In: *Journal of Control Science and Engineering* 2018 (May 2018), pages 1–8.

DTU Electrical Engineering
Department of Electrical Engineering
Technical University of Denmark

Ørsteds Plads
Building 348
DK-2800 Kgs. Lyngby
Denmark
Tel: (+45) 45 25 38 00

www.elektro.dtu.dk