Actuator faults in autonomous underwater vehicles: simulation and computer-aided control synthesis

by

Davide Grande

A thesis submitted for the degree

of

Doctor of Philosophy

Department of Mechanical Engineering

University College London

September 26, 2024

Principal Supervisor:	Professor Giles Thomas
Subsidiary Supervisor:	Dr Yuanchang Liu
Industrial Supervisors:	Dr Enrico Anderlini
	Dr Georgios Salavasidis
	Dr Alexander B. Phillips
	Dr Catherine A. Harris

I, Davide Grande, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

I, Davide Grande, declare that Artificial Intelligence technologies, including the ones based on Large Language Models, were not employed at any stage of the writing process of the presented research. Artificial Intelligence-assisted technologies were, at times, consulted to assist in the research of keywords or resources. An Artificial Intelligence-based tool was used to generate icons, as acknowledged in the relevant section in Chapter 4.

Abstract

Autonomous Underwater Vehicles (AUVs) are routinely deployed worldwide to record real-time in-situ data, at times taking part in months-long missions. Collision or entanglement with drifting debris or interference by marine creatures can cause damage to an AUV's structure and exposed actuators, with the possibility of compromising the vehicle's operations. To mitigate potentially catastrophic effects of such hazards, fault-tolerant control (FTC) systems can be employed.

Conventional FTC systems rely on monitoring the health status of the actuators using sensors or detection algorithms, which in themselves require energy resources and dedicated design efforts. Alternatively, passive FTC (pFTC) systems employing fixed gain controllers can be devised, eliminating the need for confirmatory data as to the occurrence of faults. Currently, pFTCs are designed through linearisation of the system dynamics. Recent advancements in computer-aided control methods comprising the use of Artificial Neural Networks (ANNs) hold significant potential to design control laws without relying on linear approximations, but lack formal closed-loop stability guarantees.

To address the limitations of the available FTCs, three open-source software tools were developed as part of this research. The first software tool labelled OpenMAUVe is a modular simulator for innovative AUV concepts. The second software tool named ANLC facilitates the automatic synthesis of ANN-based nonlinear control laws with formal proof of stability. The third software tool titled pFT-ANLC devises pFTC laws for systems subject to faults at actuators. Finally, the control laws generated with the pFT-ANLC were verified using the OpenMAUVe simulator, thus demonstrating the capability of the control laws to cope with faults at multiple thrusters on AUVs,

without requiring the faults having to be detected.

Impact Statement

The outcomes of this work are aimed at benefiting academia, industry and policymakers in the Autonomous Underwater Vehicle (AUV) technology field. The research and results of this work will advance the understanding of the effect of faults during AUV operations. Moreover, this thesis illustrates strategies to mitigate potentially catastrophic outcomes of faults, in turn reducing the risk of vehicle loss, or, worse, of erratic behaviours leading to damage to property, harm to marine life in the form of marine pollution or harm to humans. The outreach potential of the outputs of this work extends beyond the classical AUV control field, but can impact the oceanographic exploration of planetary bodies beyond Earth.

Impact on academia

- A first contribution of this research is the development of OpenMAUVe[1], an open-source software tool to simulate complex AUVs composed of several rigid-bodies constituted by the hull and internal moving masses. Such software in turn streamlines the simulation of the vehicles' dynamics, supporting further research in the field of guidance, control and navigation systems of AUVs.

- A second key outcome is an open-source software tool to automatically synthesise control laws with certified stability guarantees[2]. Such software can aid students and researchers in the field of control systems to assist with the design of stabilising control laws for generic nonlinear dynamics. Owing to the intrinsic nonspecific nature of the dynamical systems control field, the proposed

[1]Link to software: https://github.com/grande-dev/OpenMAUVe.
[2]Link to software: https://github.com/grande-dev/Augmented-Neural-Lyapunov-Control.

method (and software tool) can be employed in its present iteration to virtually any dynamical system (subject to the limitations that will be illustrated), not being specifically designed for use in AUV applications only.

- A final key outcome entails a new method to design nonlinear passive fault-tolerant control laws for systems affected by actuator faults. Associated with the method devised, a corresponding software tool was released open-source[3]. This control method sheds new light on a research direction that is often dismissed due to, among other reasons, a lack of control frameworks to address the problem.

Impact on industry

- Owing to the underlying modular nature of the AUV simulator, the software tool is particularly suited to the testing, verification and performance assessment of the manoeuvring capability of innovative AUV concepts. The tool facilitates the exploration of unconventional designs and of their consequent ability to complete the intended missions, both when operating in nominal modes and when subjected to faults.

- This study provides industry with an "as is" software framework to design fault-tolerant-capable control laws. This is currently the only software available open-source to design formally-correct nonlinear control laws to stabilise faulty dynamical systems.

- The fault-tolerant method proposed in this work holds a significant financial potential to reduce the cost of designing AUVs. The method can drive a paradigm shift in safety-critical applications. The current industrial practice in fault-tolerant control systems entails the reliance on the presence of monitoring sensors or algorithms to detect the occurrence of faults, followed by a tailored control system that reacts by adapting to the new fault modes. The method proposed in this work eliminates the need for detection sensors and algorithms. As a con-

[3]Link to software: https://github.com/grande-dev/pFT-ANLC.

sequence, the mechanical, electronic and control design processes can be simplified, reducing the overall system costs and enhancing the resilience of automatic systems to faults.

Impact on policymakers

In the last two decades, policymakers in the United Kingdom and internationally started to direct interest towards the regulation of autonomous maritime vehicles. As of summer 2024, there is no specific international regulation covering the design, certification and operation of Unmanned Maritime Systems (UMSs)[4] [2].

Specific focus has recently been on surface vessels. With the rapidly paced development of Maritime Autonomous Surface Ship (MASS) systems[5], in 2021, with the MSC.1/Circ.1638 the International Maritime Organisation (IMO) recognised the need to provide standards and best practices to designers, builders, owners and operators of MASSs [3]. The first requirements for Remotely Operated Vehicles (ROVs) were set in 2024 by the Maritime & Coastguard Agency (MCA), defining standards for safety and cyber security measures [4], with the regulation coming in place in 2027. Plan on adopting a mandatory operational code for MASSs is set to come into place in 2032, with an updated regulatory code being currently drafted by IMO.

On the contrary, no extensive guidelines or defined standards exist with regards to the deployment and operation of AUVs. Regulation has the potential to be ever more convoluted for certain classes of vehicles, such as the Underwater Gliders (UGs) (which will be discussed in this thesis), that might be classified as ships, submarines or scientific research equipment [5]. Nonetheless, embedding fault-tolerant capabilities onboard AUVs appears to be on the agenda of regulatory bodies, such as highlighted in a British Parliamentary committee [6]. Similarly, the European Defence Agency (EDA) in 2022 remarked that UMSs need to be able to handle abnormal situations such as faults and critical system failures [2].

[4]Within North Atlantic Treaty Organization (NATO) terminology, a UMS is defined as a system operating in the maritime environment (subsurface, surface, air) whose primary component is at least one unmanned vehicle [1]

[5]A MASS is defined as a ship which, to a varying degree, can operate independent of human interaction [3]. The degree refers to the level of autonomy onboard the ship, ranging from the assistance in supervisory tasks, to fully autonomous ships without onboard crew.

To conclude, within this thesis, diverse technical solutions to embed fault-tolerant capabilities onboard AUVs were surveyed, offering an introductory perspective to technical committees on the matter. Additionally, the proposed solution devised in this work might represent a starting point to discuss whether current design paradigms are suited to safety-critical systems such as UMSs.

Impact on human space exploration

In the most recent decades, space exploration efforts have been directed towards the planetary bodies confirmed to (or speculated to) possess frozen or liquid water, owing to the interest in search of alien life forms and habitability of other planets. A first case is represented by the possibility of water in liquid form being hosted beneath the Martian ice caps [7]. Another remarkable study regards the icy moons of Jupiter, specifically Europa, which is surmised to harbour liquid water oceans under the outer icy surface layer, following exploration by the National Aeronautics and Space Administration (NASA) Galileo spacecraft in 1996-1998 [8].

Two follow up missions were envisioned. European Space Agency (ESA) Jupiter Icy Moons Explorer (JUICE), currently in its deep-space voyage towards Jupiter and set to reach the icy moons Europa, Ganymede and Callisto in 2032 [9], and NASA Europa Clipper, set to reach Europa in 2031 (provided a successful launch within the scheduled launch window in February 2025). Should these missions confirm the presence of liquid water, the natural next stage will entail the deployment of probes to penetrate the ice shelf and to collect in-situ data. To this end, early concepts (2001) such as cryobots were envisioned with the idea of penetrating the Martian ice caps or Europa's ice layer [10]. Similar ice-melting submarine concepts were developed more recently [11], ARTEMIS [12] and TRIPLE [13]. In the design of the vehicle guidance and control system, specific focus was dedicated to fault detection and recovery, highlighting the growing attention on the matter.

The output of this thesis is well suited to these concept missions, where no human-in-the-loop possibility exists and where the detrimental effects of faults on the mission success should be kept under control with every known technical solution.

Acknowledgements

This section was redacted to anonymise names and personal identifiers.

To Professor Giles Thomas, for the unique chance to undertake this experience, for being the rational Pole Star, and for the care taken in nurturing me.

To my family and extended family, for the support and the patience despite my absence during these years.

To my grandparents, for their efforts through time.

To all my friends, for the fantastic experiences despite the time *given with the dropper*.

To all my present and past friends and colleagues of the 5th floor, for creating the environment that made this time unforgettable.

To the friends and colleagues who provided specific support over the years. To F, for the support with robust control concepts and for the excellent reading recommendations. To A, for the help with calculating obscure hydrodynamic coefficients. To M, for sharing the journey together.

To Y, for finding this PhD posting, for lighting up the research fire and for supporting me on countless occasions.

To Dr Andrea Grech La Rosa, for being a reason to start this experience, for always sparing time to help and for the teaching about all-things-tanks.

To T, A, S, F, J, C and A, for being my first family in London.

To D, G, S and E, for sharing this experience even when at great distance, and for always being present when needed.

To D, for the long days in the office and beyond, for the discussions about fundamental concepts, which had thus far remained a mystery.

To Dr Catherine A. Harris, Dr Yuanchang Liu, Prof Elias B. Kosmatopoulos, Dr Rachel Pawling, Prof Helge Wurdemann, Dr Francesca Boem, Dr Sicun Gao, Dr Soonho Kong, Dr Riccardo M.G. Ferrari, Prof Steven Bishop, Dr Davide Fenucci and Dr Daniele Masti, for the time and advice, which helped shaping the work that followed.

To Prof Alfredo Martins and Prof Luca Bascetta, for the early support, and for transmitting the passion for control systems and underwater robotics.

To my Professors during my years at Polimi and UCL, especially Prof Riccardo Scattolini, Prof Paolo Bolzern, Prof Nicola Schiavoni, Dr Fabio Dercole and Prof Matteo Corno, for their precious and passionate teachings.

To R and G, for having trusted me as a growing child, bearing long-lasting impact.

To the staff at UCL, especially Anthony, Peter, Ray, Adam, Luna, Angelo, Steve, Mykal, Matthew, Rachel, Marvin and Kevin, for their support on every occasion.

To Anthony Grout, Prof Paul Hellier and Dr Llwellyn Morse, for the commitment to their roles and for the assistance during these years as a student at UCL.

To Dr Ben Hanson, Prof Ryo Torii, Dr Ali Abolfathi, Dr Ema Muk-Pavic, Prof Giles Thomas and Dr Tom Smith, for trusting me to help with the teaching activities at UCL.

To the University College London and National Oceanography Centre, especially Dr Alexander B. Phillips, Dr Enrico Anderlini and Prof Giles Thomas, for funding this research and for gifting me with great research freedom for these many years.

To Dr Adam Wojcik, Prof Ian Eames, Dr Sara Abad Guaman, Prof Nader Saffari, Prof Andrea Ducci and Prof Manish K. Tiwari, for the precious time gifted and for the valuable discussions about the engineering discipline, teaching and knowledge sharing.

To Konrad Yearwood, for the painstaking review of this thesis (not *manuscript*), and for the help in understanding my own work to another level.

To Dr Georgios Salavasidis, for the invaluable advice and for the rational perspectives over research topics, and more.

To Dr Enrico Anderlini, for the forever-lasting help, for the research direction, for the help with navigating through this experience and for supporting this project until the very end.

To Dr Andrea Peruffo, for the research advice, for the technical support, for the patience and for representing a great role model.

To L, for the tireless support in taking care of me. Without you, I would have not arrived to write these lines.

Contents

List of Figures

List of Tables

Acronyms

2D two-dimensional. 53

3D three-dimensional. 53, 72, 75, 78–80

AFA Actuator Fault Accommodation. 19–21, 31, 32

AFDI Actuator Fault Detection, Identification. 20

AFDIA Actuator Fault Detection, Identification, and Accommodation. 19, 20

aFTC active fault-tolerant control. 21–25, 27, 31–33

AI Artificial Intelligence. 11–13, 230

ANLC Augmented Neural Lyapunov Control. 111, 115, 119–122, 129, 131–133, 140, 177, 199–201

ANN Artificial Neural Network. 30, 35, 36, 92–94, 99, 100, 104, 105, 107, 108, 113–115, 121, 122, 125, 137, 142, 149, 150, 161, 183, 193

AUV Autonomous Underwater Vehicle. vi–ix, 9, 10, 12–18, 20, 31–43, 45, 47–49, 52–55, 57–59, 61–63, 67–75, 81–85, 130–132, 138, 144–146, 148–150, 156, 167, 169, 171, 178–181, 183, 185, 187, 189–191, 194–196, 198, 199, 201

BMI Bilinear Matrix Inequalities. 28

CA Control Allocation. 24, 25, 27, 32

CE counterexample. 94, 97, 101, 103, 110–115, 119, 120, 127, 128, 133, 135, 137, 138, 176

CEGIS Counterexample-Guided Inductive Synthesis. 94, 95, 98, 112, 113, 115

CEM Control Effectiveness Matrix. 24

CFD Computational Fluid Dynamics. 158

CLF Control Lyapunov Function. 29, 89, 92–94, 96–98, 100, 101, 103–108, 110, 112, 114, 115, 119–122, 126, 128, 132–135, 137, 150, 160, 167, 175, 176, 185, 193, 199, 232

COB Centre of buoyancy. 55, 64, 67, 82

COG Centre of gravity. 64, 65, 67, 70, 71, 82, 145, 183

COP Centre of pressure. 60, 68

CPU Central Processing Unit. 119, 149

DAE Differential Algebraic Equation. 39, 51, 54, 85, 197

DC Direct Current. 70

DF Discrete Falsifier. 112

DOF Degree Of Freedom. 24, 32, 38, 41, 48, 49, 54, 71, 146

DRL Deep Reinforcement Learning. 12, 30

DVL Doppler Velocity Logger. 37, 157

ECEF Earth-centred Earth-fixed. 55

EDA European Defence Agency. viii

ESA European Space Agency. ix

FDI Fault Detection and Isolation. 20, 24, 25, 27, 31–34, 130

FT-BIC fault-tolerant bio-inspired controller. 30, 31

FTC fault-tolerant control. 9–13, 18–21, 25, 26, 28, 34, 41, 67, 74, 131, 132, 138, 169, 197

GD Gradient Descent. 92

GNC Guidance, Navigation and Control. 201, 202

GPU Graphics Processing Unit. 119, 149

IDE Integrated Development Environment. 50, 52

IMM Interacting Multiple Models. 22

IMO International Maritime Organisation. viii

JUICE Jupiter Icy Moons Explorer. ix

LF Lyapunov Function. 87, 89, 92, 93, 98

LNN Lyapunov Neural Network. 92, 93, 98

LQ Linear-Quadratic. 161

LQR Linear Quadratic Regulator. 100, 121, 122, 140, 141, 143

LSTM Long Short-Term Memory. 30, 31, 35

MASS Maritime Autonomous Surface Ship. viii

MCA Maritime & Coastguard Agency. viii

MILP Mixed Integer Linear Programs. 98

ML machine learning. 29–31, 35, 42–44, 86, 99, 129, 196

MM Multiple Model. 21, 22, 27

MMST Multiple Model Switching and Tuning. 22

MPC Model Predictive Control. 23, 24, 27, 33

MRAC Model Reference Adaptive Control. 22

MSL Modelica Standard Library. 52, 73, 180

NASA National Aeronautics and Space Administration. ix

NATO North Atlantic Treaty Organization. viii

NED North-East-Down. 55, 154

NLC Neural Lyapunov Control. xxv, 36, 43–45, 86, 94, 98–104, 106, 111, 113, 119, 121, 122, 127, 128

NMPC Nonlinear Model Predictive Control. 24, 26, 27, 33

NOC National Oceanography Centre. 17, 183

NP nondeterministic polynomial time. 111, 112, 137

ODE Ordinary Differential Equation. 49, 51, 54, 87, 88

pFT-ANLC passive Fault-Tolerant-Augmented Neural Lyapunov Control. 135, 138, 140–144, 161, 169, 171, 173–175, 178, 179, 183, 191, 194, 195, 199–201

pFTC passive fault-tolerant control. 27–31, 33, 34, 41, 42, 44, 130, 132, 140, 177, 178, 196, 199, 201

PID proportional integral derivative. 30, 170

PWL piecewise linear systems. 98

RAM Random Access Memory. 119, 149

REA Reconfigurable Eigenstructure Assignement. 25, 27

RL Reinforcement Learning. 35

RMSE Root Mean Square Error. 163

RNN Recurrent Neural Network. 35

ROA Region of Attraction. 87, 89, 100, 106, 107, 124, 125, 129, 134, 175, 176, 199

ROS Robot Operating System. 51

ROV Remotely Operated Vehicle. viii, 16, 20, 34, 37, 38, 49, 180, 196

RPM Repetition Per Minute. 67

SFA Sensor Fault Accommodation. 20

SFDI Sensor Fault Detection, Identification. 19

SFDIA Sensor Fault Detection, Identification, and Accommodation. 19

SGD Stochastic Gradient Descent. 104, 114, 116, 136, 137

SMC Sliding Mode Control. 26, 27, 33

SMT Satisfiability Modulo Theories. 94–98, 110–113, 115, 120, 130, 137, 175, 200, 201

SNAME Society of Naval Architects and Marine Engineers. 56, 61

SSW selective sliding window. 113, 120

TAM Thruster Allocation Matrix. 24, 25, 32

TCM Thruster Configuration Matrix. 24

UG Underwater Glider. viii, 15–17, 20, 34–38, 42, 47, 49, 52, 53, 55, 68, 70, 71, 75, 76, 78–81, 84, 138, 153, 154, 156, 196, 201

UMS Unmanned Maritime System. viii, ix

vANLC vanilla Augmented Neural Lyapunov Control. 140–144

VBD Variable Buoyancy Device. 37, 38, 43, 52, 53, 64, 65, 69, 72, 79, 82, 156, 157

WOCE World Ocean Circulation Experiment. 7

wrt with respect to. 147, 184

Nomenclature

Vectors, Matrices and Sets

Bold symbols are used exclusively to denote vectors, matrices, and sets.

Greek symbols

α	Angle of attack
α_j	Tuning parameters of the ANLC/pFT-ANLC loss functions
β	Sideslip angle
β_j	Orientation of the j-th thruster with respect to the y_b-axis
β_s	Constant incremental step
β_ρ	Value of the Lyapunov Function level set
β_{max}	Maximum value of the Lyapunov Function level set
$\boldsymbol{\dot{\omega}_0}$	Angular acceleration of a frame $\{O\}$
$\boldsymbol{\mu}$	Set of the actuator efficiency values
$\boldsymbol{\omega_0}$	Angular velocity of a frame $\{O\}$
$\boldsymbol{\Phi}$	Set of possible system faults
$\boldsymbol{\Theta}$	Set of the Euler angles
δ	dReal verification precision

δ_{cs} Deflection angle of a control surfaces

ε ε-stability bound

η ANN hyperparameters

κ Control-branch training selector

λ Learning rate of the Lyapunov ANN

λ_c Learning rate of the control ANN

μ_j Efficiency of the j-th actuator

Ω_β Inner approximation of the ROA

$\overline{\gamma}$ Upper boundary of the verification domain

$\overline{\sigma}_j$ Saturation limits of the j-th actuator

$\overline{\tau}$ Optional prescribed precision of approximation of the CLF

ϕ Roll angle

ϕ_j Fault mode j-th

ψ Yaw angle

ρ Fluid density

σ_j Activation function of the j-th ANN layer

σ_L Scalar parameter of the controlled Lorentz system

θ Pitch angle

ξ Glide path angle

ζ_{DF} Maximum cardinality of CE_{DF}

ζ_{SMT} Maximum cardinality of CE_{SMT}

Other Symbols

$\boldsymbol{\nabla(V)_j}$ Gradient of a function V, calculated with respect to the independent vector $\boldsymbol{j}$

$\|\boldsymbol{a}\|_2$ 2-norm of a vector $\boldsymbol{a}$

$\mathbb{Q}^+$ Set of the positive rational numbers

$\mathbb{R}$ Set of the Reals

$\mathbb{R}^+$ Set of the positive Reals

$\mathscr{C}^1$ A function with continuous first derivative with respect to its argument

$\mathcal{D}$ Domain over the Reals

$\mathcal{H}_2$ $\mathcal{H}_2$-norm of system $\mathcal{H}$

$\mathcal{H}_\infty$ $\mathcal{H}_\infty$-norm of system $\mathcal{H}$

$\mathcal{R}(\cdot)$ Rectified Linear Unit function

$\Rightarrow$ implies

© Copyright symbol

Roman Symbols

$\boldsymbol{\dot{v}_0}$ Linear acceleration expressed in frame $\{O\}$

$\boldsymbol{\dot{x}}$ Time-derivative of the space vector

$\boldsymbol{\hat{k}_i}$ Versor of the i-th axis of the inertial frame

$\boldsymbol{B_h}$ Buoyancy force vector due to vehicle hull volume

$\boldsymbol{B_j}$ Bias of the j-th ANN layer

$\boldsymbol{B}_V$ Buoyancy force vector due to the VBD volume

$\boldsymbol{f}^j$ Force vector expressed in the j-th frame

$\boldsymbol{f_0}$ External forces applied to a frame $\{O\}$

$\boldsymbol{F}_{hd}^{f}$ Hydrodynamic force vector in the flow frame

$\boldsymbol{I_0}$ Inertia tensor calculated with respect to a frame $\{O\}$

$\boldsymbol{I}$ Identity matrix

$\boldsymbol{K_{\omega^1}}$ Vector of the linear damping coefficients of the viscous moment

$\boldsymbol{K_{\omega^2}}$ Vector of the quadratic damping coefficients of the viscous moment

$\boldsymbol{m_0}$ Moment of the external forces and external torques with respect to a frame $\{O\}$

$\boldsymbol{M_A}$ Added inertia matrix

$\boldsymbol{M}_{hd}^{f}$ Hydrodynamic moment vector with respect to the flow frame

$\boldsymbol{r_b}$ Distance of the variable ballast mass from $\{O_b\}$

$\boldsymbol{r_g}$ Distance of the centre of gravity from the body-fixed frame $\{O_b\}$

$\boldsymbol{r_p}(t)$ Distance of the movable mass from $\{O_b\}$

$\boldsymbol{r_w}$ Distance of the non-homogeneous from $\{O_b\}$

$\boldsymbol{r_w}$ Distance of the non-homogeneous mass from $\{O_b\}$

$\boldsymbol{R}_{i}^{j}$ Rotation matrix from the j-th frame to the i-th frame

$\boldsymbol{s_j}$ Sample j within the set S

$\boldsymbol{u}$ Control input vector

$\boldsymbol{v_0}$ Linear velocity vector expressed in frame $\{O\}$

$\boldsymbol{v_c}$ Linear velocity vector of the ocean currents expressed in the body-fixed frame

$\boldsymbol{W_j}$ Weight of the j-th ANN layer

$\boldsymbol{W}$ Gravity force vector

$\boldsymbol{x}^{\star}$	State-space vector equilibrium
$\boldsymbol{x}$	State-space vector
$\boldsymbol{x}_0$	State-space vector at time t=0
$\boldsymbol{z}_{j-1}$	Input to the j-th ANN layer
$\boldsymbol{z}_j$	Output of the j-th ANN layer
$\bar{\boldsymbol{e}}$	RMSE of the reference tracking error
$\boldsymbol{e}$	Reference tracking error
$\boldsymbol{f_n}$	Nominal system dynamics
$\boldsymbol{Q}$	State weighting matrix
$\boldsymbol{R}$	Control input weighting matrix
$\boldsymbol{r}$	Reference tracking value
$\boldsymbol{u_f}$	Control input embedding the faults
$\dot{m}_b$	Rate of change of the variable ballast mass
$\dot{V}$	Lie derivative
$\dot{V}_{\eta}^{S}$	Candidate translated Lie derivative
$\dot{V}_n$	Lie derivative of the nominal system mode
$\dot{V}_{\phi_j}$	Lie derivative of the j-th fault mode
∇_h	Volume of the vehicle hull
∇_V	Volume of the VBD
$\{O_b\}$	Origin of the body-fixed frame
$\{O_f\}$	Origin of the flow frame

$\{O_i\}$ Origin of the inertial frame

A Vehicle cross sectional area

a Minor-semi axis of an ellipsoidal body along the y_b-body axis

b Minor-semi axis of an ellipsoidal body along the z_b-body axis

B_h Modulus of the buoyancy force vector due to vehicle hull volume

b_L Scalar parameter of the controlled Lorentz system

b_p Drag coefficient of the pendulum

B_V Modulus of the buoyancy force vector due to the VBD volume

c Major-semi axis of an ellipsoidal body along the x_b-body axis

C_D Drag coefficient

C_L Lift coefficient

C_{DLj} Drag coefficient of the viscous moment around the j-axis of the flow frame

C_{SF} Side-force drag coefficient

CE_{DF} CE returned by the DF Falsifier

CE_{SMT} CE returned by the SMT Falsifier

D Drag force

d Number of actuators that can be affected by faults

D_p Propeller diameter

F_j Force generated by the j-th thruster

g Earth's gravity constant

g_j Number of neurons within the j-th ANN layer

h_j Health status of the j-th actuator

I_k Added mass coefficient along the I-th body axis due to an acceleration along the k-th axis

J Performance index

J_e Performance index associated to the tracking error

J_p Moment of inertia of the pendulum

J_u Performance index associated to the control input

J_z Moment of inertia of the AUV around the vertical axis (including the added inertia)

k Total number of ANN layers

K_j j-th quasi-steady state drag coefficient

K_T Propeller constant coefficient

L Lift force

l_j Distance of j-th actuator from the COG

l_p Length of the pendulum arm

L_{cs} Lift force generated by a control surface

L_{ELR} Empirical Lyapunov Risk Loss function

$l_{j,x}$ Distance of j-th actuator from the COG, projected along the x_b-axis

$l_{j,y}$ Distance of j-th actuator from the COG, projected along the y_b-axis

L_{SLR} Strict Lyapunov Risk Loss function

m Mass

m_0 Net buoyancy

m_b Variable ballast mass

m_h Mass of the hull

m_p Movable mass

m_s Stationary mass

m_v Overall vehicle mass

m_w Non-homogeneous mass distribution

M_{cs} Moment of the lift force generated by a control surface

n_p Propeller rotation rate

P Power consumption

p Angular velocity around the x-axis of the body-fixed frame

P_g Generalised plant

p_w Water pressure

q Angular velocity around the y-axis of the body-fixed frame

q^I Optional initial control gain

r Angular velocity around the z-axis of the body-fixed frame

r_L Scalar parameter of the controlled Lorentz system

r_{cs_x} Control surface axial distance from $\{O_b\}$

Re Reynolds number

S ANLC/pFT-ANLC training set

S_I ANLC/pFT-ANLC initial dataset

S_{CE} ANLC/pFT-ANLC counterexample dataset

S_{cs} Control surface planform area

SF Side-force (or crossflow drag)

t Time

T_p Thrust generated by a propeller

T_w Water temperature

T_{DLj} Viscous moment around the j-th axis of the flow frame

u Surge velocity, or linear velocity along the x-axis of the body-fixed frame

u_r Linear velocity of the ocean currents along the x-axis of the body-fixed frame

V (Control) Lyapunov Function

v Sway velocity, or linear velocity along the y-axis of the body-fixed frame

V_c Candidate Control Lyapunov Function

V_r Magnitude of the flow speed vector

v_r Linear velocity of the ocean currents along the y-axis of the body-fixed frame

V_η^S Candidate translated Control Lyapunov Function

$v_{p,in}$ Propeller flow inlet velocity

$v_{p,out}$ Propeller flow outlet velocity

W Modulus of the gravity force vector

w Heave velocity, or linear velocity along the y-axis of the body-fixed frame

w_r Linear velocity of the ocean currents along the z-axis of the body-fixed frame

$x_j^\star$ Equilibrium value of the j-th state-space variable

x_j x-axis of a frame of origin $\{O_j\}$

y_j y-axis of a frame of origin $\{O_j\}$

z_j z-axis of a frame of origin $\{O_j\}$

Relevant publications

The core ideas illustrated in this thesis were published in an original version in the following publications, and were later extended in the development of this thesis:

1. **Modelica simulator**: "Open-source Simulation of Underwater Gliders", D. Grande, L. Huang, C. Harris, P. Wu, G. Thomas, E. Anderlini, IEEE Oceans Conference Record, 2021 [19].

2. **Computed-aided control synthesis**: "Augmented Neural Lyapunov Control", D. Grande, A. Peruffo, E. Anderlini, G. Salavasidis, IEEE Access, 2023 [22].

3. **Fault-tolerant computed-aided control synthesis**: "Systematic Synthesis of Passive Fault-Tolerant Augmented Neural Lyapunov Control Laws for Nonlinear Systems", D. Grande, D. Fenucci, A. Peruffo, E. Anderlini, A. B. Phillips, G. Salavasidis, G.Thomas, IEEE Conference on Decision and Control, 2023 [24].

4. **Fault-tolerant computed-aided control synthesis**: "Passive Fault-Tolerant Augmented Neural Lyapunov Control: A method to synthesise control functions for marine vehicles affected by actuators faults", D. Grande, A. Peruffo, G. Salavasidis, E. Anderlini, D. Fenucci, A. B. Phillips, E. B. Kosmatopoulos, G. Thomas, Control Engineering Practice, 2024 [23].

Correlated research was carried out with lessons learnt embedded within this thesis:

1. **Fault detection**: "Smart anomaly detection for Slocum underwater gliders with a variational autoencoder with long short-term memory networks", Z. Bedja-Johnson, P. Wu, D. Grande, E. Anderlini, Applied Ocean Research 120, 2021 [26].

2. **Modelica simulation**: "Vector Field-based Guidance Development for Launch Vehicle Re-entry via Actuated Parafoil", S. Farì, D. Grande, International Astronautical Congress, IAC, 2021 [27].

Research Paper Declaration Form

1. **Relevant publication:**

   (a) **Title:** Open-source Simulation of Underwater Gliders

   (b) **DOI:** https://doi.org/10.23919/OCEANS44145.2021.9705852

   (c) **Publication venue:** IEEE OCEANS 2021

   (d) **Publisher:** IEEE

   (e) **Publication date:** 15-02-2022

   (f) **Manuscript authors:** D. Grande, L. Huang, C. Harris, P. Wu, G. Thomas, E. Anderlini

   (g) **Peer-review:** Extended Abstracted peer reviewed

   (h) **Copyright status:** Held by publisher

   (i) **Earlier forms of the manuscript:** None

   (j) **Statement of contribution:** The author of this thesis is the main contributor of the research publication as titled at point (a).

   (k) **Relevant chapters:** Chapter 2, Chapter 4.

   ☑ *I acknowledge permission of IEEE to include in this thesis portions of the publication. Material from this publication is included in this thesis with the associated label "© 2022 IEEE [19]".*

2. **Relevant publication:**

   (a) **Title:** Augmented Neural Lyapunov Control

   (b) **DOI:** https://doi.org/10.1109/ACCESS.2023.3291349

(c) **Publication venue:** IEEE Access, Volume 11, Pages 67979 - 67986

(d) **Publisher:** IEEE

(e) **Publication date:** 03-07-2023

(f) **Manuscript authors:** D. Grande, A. Peruffo, E. Anderlini, G. Salavasidis

(g) **Peer-review:** Regular peer-review

(h) **Copyright status:** Held by publisher

(i) **Earlier forms of the manuscript:** None

(j) **Statement of contribution:** The author of this thesis is the main contributor of the research publication as titled at point (a).

(k) **Relevant chapters:** Chapter 2, Chapter 5.

☑ *I acknowledge permission of IEEE to include in this thesis portions of the publication. Material from this publication is included in this thesis with the associated label "© 2023 IEEE [22]".*

3. **Relevant publication:**

(a) **Title:** Systematic Synthesis of Passive Fault-Tolerant Augmented Neural Lyapunov Control Laws for Nonlinear Systems

(b) **DOI:** https://doi.org/10.1109/CDC49753.2023.10383378

(c) **Publication venue:** 2023 62nd IEEE Conference on Decision and Control (CDC)

(d) **Publisher:** IEEE

(e) **Publication date:** 19-01-2024

(f) **Manuscript authors:** D. Grande, D. Fenucci, A. Peruffo, E. Anderlini, A. B. Phillips, G. Salavasidis, G.Thomas

(g) **Peer-review:** Regular peer-review

(h) **Copyright status:** Held by publisher

(i) **Earlier forms of the manuscript:** None

(j) **Statement of contribution:** The author of this thesis is the main contributor of the research publication as titled at point (a).

(k) **Relevant chapters:** Chapter 2, Chapter 6.

☑ *I acknowledge permission of IEEE to include in this thesis portions of the publication. Material from this publication is included in this thesis with the associated label "© 2024 IEEE [24]".*

4. **Relevant publication:**

(a) **Title:** Passive Fault-Tolerant Augmented Neural Lyapunov Control: A method to synthesise control functions for marine vehicles affected by actuators faults

(b) **DOI:** https://doi.org/10.1016/j.conengprac.2024.105935

(c) **Publication venue:** Control Engineering Practice

(d) **Publisher:** Elsevier

(e) **Publication date:** 07-2024

(f) **Manuscript authors:** , D. Grande, A. Peruffo, G. Salavasidis, E. Anderlini, D. Fenucci, A. B. Phillips, E. B. Kosmatopoulos, G. Thomas

(g) **Peer-review:** Regular peer-review

(h) **Copyright status:** Held by publisher

(i) **Earlier forms of the manuscript:** None

(j) **Statement of contribution:** The author of this thesis is the main contributor of the research publication as titled at point (a).

(k) **Relevant chapters:** Chapter 2, Chapter 6.

☑ *I acknowledge permission of Elsevier to include in this thesis portions of the publication. Material from this publication is included in this thesis with the associated label "© 2024 Elsevier [23]".*

e-Signatures confirming that the information above is accurate: digitally signed

Candidate: Davide Grande

Date: September 16, 2024

Supervisor: Giles Thomas

Date: September 16, 2024

Chapter 1

Introduction and motivation

1.1 Problem definition

Deepening our knowledge of the oceans is vital to understand the links to the Earth's climate and to better protect its natural resources [28]. Studying the oceans additionally assists with forecasting sea bed phenomena such as earthquakes and tsunamis [29]. Moreover, ocean research supports long-term weather forecasts linked to food production and water supply, in turn helping to investigate, forecast and mitigate the impact of undesirable climate anomalies and extreme weather events [30].

In the 1980s and 1990s, concerns started to grow regarding the monitoring of the ocean environment, its temperature, the concentration of pollutants and how the ocean currents affect the spreading and dilution of human-made byproducts. Early attempts at understanding the oceans' circulatory systems were carried out by World Ocean Circulation Experiment (WOCE), involving research vessels as a means of data collection [31]. It was soon realised that methods relying on extensive human involvement, such as oceanographic ships, were not an economically viable option to collect sufficient data to construct and validate theoretical oceanographic models.

The need to envision new solutions to provide subsea data at a higher scale and frequency, matching the remote sensing acquisition capability of satellites for surface data, became the focus of a new ocean engineering research stream. In the two decades that have followed, ocean observing technologies such as tide gauges, surface drifters and profiling floats were developed, systems today routinely deployed worldwide in

their thousands to record real-time in-situ data [32].

Tide gauges are docked instruments measuring trends in mean sea level, wind speed and direction, air and water temperature, and are employed to support harbouring operations, navigation and to detect extreme conditions such as storm surges and tsunamis [33]. In contrast, surface drifters and profiling floats are free floating ocean instruments. The drifters are transported by the ocean currents and assist validating satellite sea surface temperature measurements which are used to improve weather models and to verify hurricane intensity forecasts [34], while profiling floats are robotic platforms that move in a vertical motion by adjusting their buoyancy while mapping parameters of the water profiles. The extent of the deployment of the Argo drifters is reported as an example in Fig. 1.1.

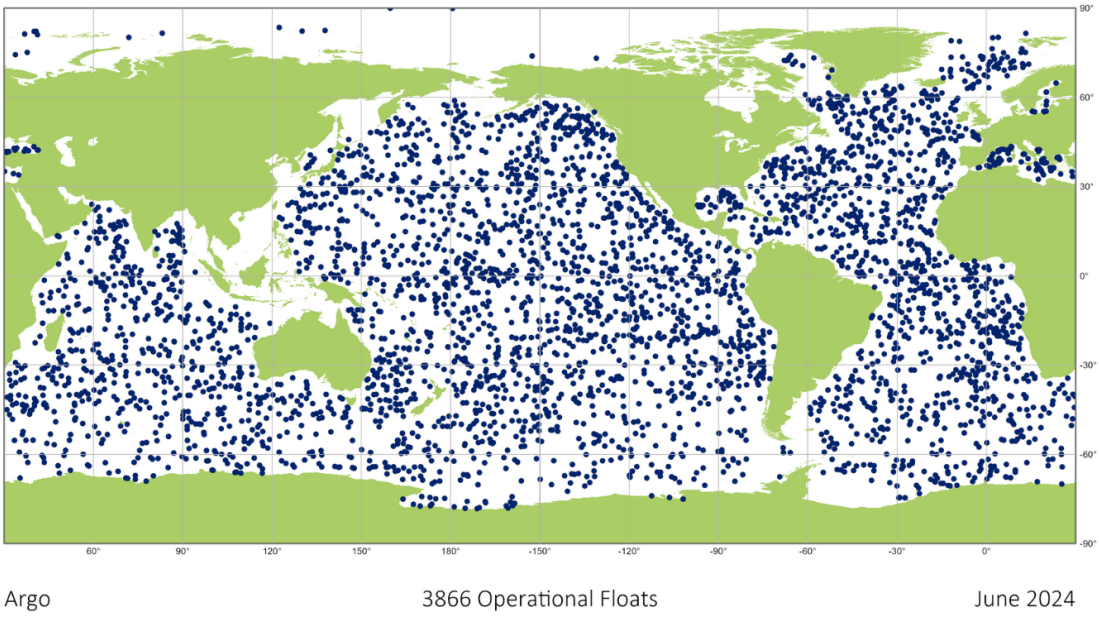


Figure 1.1: Argo floats deployment map as of June 2024 [14].

In short, satellites offer a large spatial coverage but are constrained to surface observations, profiling floats and drifters are suited to collect and categorise subsurface low-scale data, while the use of research ships is constrained by the prohibitive costs associated with their operations [35]. These limitations highlighted the need to develop new instruments to advance data collection on a large scale within accessible costs, while not being constrained to surface or purely vertical data.

In parallel, in the second half of the 20th century, design efforts focused on the

design of AUVs, inheriting lessons learnt from earlier studies derived from the use of torpedoes for military operations [36]. AUVs were developed for different purposes ranging from ocean monitoring for scientific purposes, to inspection of underwater installations and to military applications such as underwater mine detection and removal [37]. Significant achievements were accomplished in the past seventy years of AUVs design, with some of these platforms today being capable of months-long deployments and with operational ranges of up to thousands of kilometres [38].

Despite the advantages, AUVs suffer from a range of potential issues during the deployment missions. Given the unstructured nature of the underwater environment, a variety of unforeseen and unplanned for events can occur that may have the potential to jeopardise the mission's success. In the case of AUVs, collisions with debris, or with other vessels during the communication phase at surface, or attacks by large ocean predators and growth of marine life on the vehicle hull can all lead to malfunctions. These elements encompass damages to the vehicle hull or to the exposed components, such as the actuators.

Faults at actuators can manifest as thruster obstruction, thruster flooding and rotor failure [39]. Sea borne solid objects such as ice, seaweed or other marine detrita entering between the blades, the propeller and the enclosure, can result in damage to the propeller blades and impact the torque required to rotate the propulsion motor, while water ingress inside sealed electronic compartments can lead to shorts or current dispersion. At other times, despite the vastness of the oceans, instantaneous faults at control surfaces can appear due to sudden collisions, most likely with other vessels [40]. During long-lasting deployments in challenging environments, such as when conducting missions under the ice caps [41], AUVs have been lost, and often the causes have remained unknown [42].

As with space systems or aircraft, AUVs need appropriate recovery strategies to cope with unforeseen and unplanned for events. Since there is no "panic button" to press, such as turning the power off or engaging an emergency brake [43], tailored control logic needs to be embedded within the vehicle prior to deployment. The branch of control system that tackles such a need is referred to as fault-tolerant control (FTC).

FTC involves the design of an automatic control system based on one of two underlying principles: either the sources of faults are detected and counteracting measures are taken accordingly, or the system is designed to be intrinsically robust to faults. The first type of control system relies on sensors or algorithms that constantly monitor the working status of the actuators and can react appropriately when anomalies are detected. These strategies, which currently represent the industrial solution most commonly applied, introduce additional issues in the context of AUV operations. First, the presence of additional sensors render the overall vehicle design more complex, and hence more prone to additional faults, i.e. mitigating strategies need to be put in place to manage those cases when the monitoring sensors fail. Next, these control strategies assume fast and reliable identification of the faults, with delays in identification potentially leading to instabilities in the closed-loop dynamics. Next, these strategies require dedicated energy to power the sensors, with energy consumption being already at a premium and currently representing the main limitation to extending deployment time.

Alternatively, robust FTC systems can be developed, designed to be resilient to loss of efficiency of actuators, to jamming of control surfaces, or to full loss of individual thrusters. This type of control system does not require monitoring sensors or algorithms, thereby delivering overall improved reliability and energy consumption. On the downside, to achieve robustness, robust FTC systems compromise performance, for instance involving the relaxing of target control objectives such as the asymptotic stability of a prescribed equilibrium. Currently, robust FTC systems can either be devised by relying on linearisation of dynamical models, or, when accounting for the non-linearised dynamics, can guarantee closed-loop stability only under specific cases of loss of efficiency of the actuators [44].

To increase the AUV operational capabilities, namely to enhance deployment times while maximising vehicle resilience to unforeseen external factors and while minimising the need for human intervention, robust FTC represent an attractive solution. Nonetheless, a lack of a unified and generic control solution to increase reliability of AUVs when subjected to actuator faults provides the motivation for further research

efforts in robust FTC solutions.

In recent years, a surge in Artificial Intelligence (AI)-based applications has steered research focus in academia and industry. Applications ranging from image and speech detection and synthesis, to driving assistance technologies and tools supporting the motion-picture industry, semi-automatic coding and generative text are quickly finding their way at an incredible pace. As recently as 2016, it was speculated that a super intelligent AI could complete the writing of a PhD thesis in one afternoon [45]. Within less than ten years of that forecast, in the summer of 2024, such a feat is already possible, although the quality of the thesis would still be debatable, luckily for today's PhD candidates.

AI-based technologies have been gaining interest within the control system community, owing to their potential they hold to solve complex problems, such as designing nonlinear control functions for high-dimensional dynamical systems, a notably complicated endeavour. Academic research is increasingly being focused on AI-based technologies, as illustrated in Fig. 1.2, reporting the number of works published in the interval 2000-2023, containing the keywords *control systems* and *machine-learning*[1].

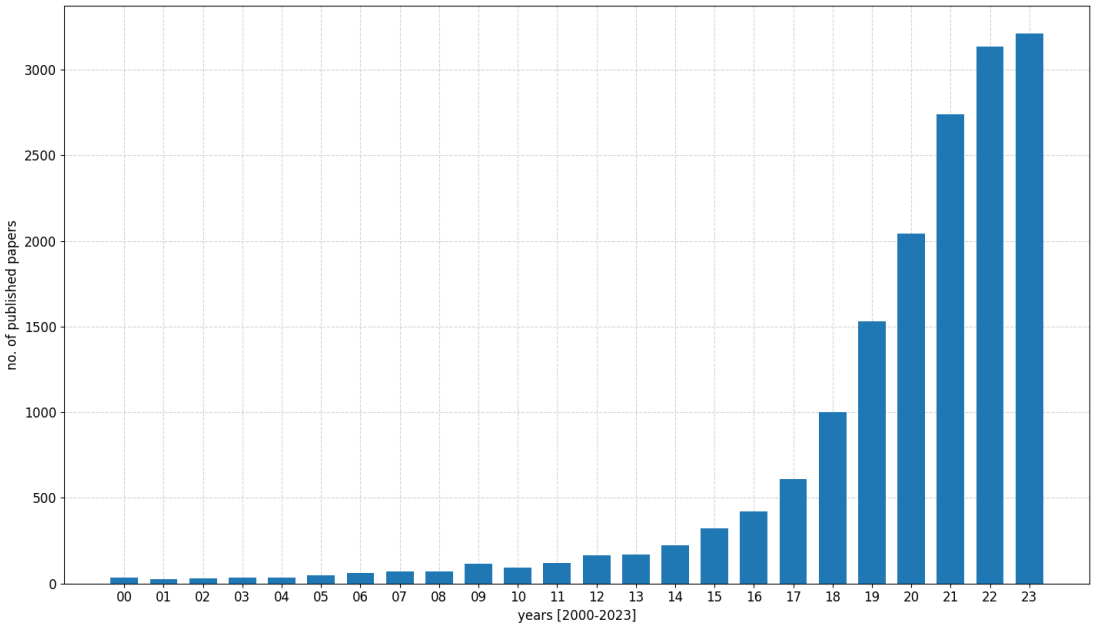


Figure 1.2: Publications containing the keywords *control system* and *machine-learning* [15].

[1]Comparable results can be obtained by combining similar keywords, such as *artificial intelligence*, *machine-learning*, *control systems* and *automatic control*.

Nonetheless, issues persist as the formality of such solutions lying within black-box approach and often based on paradigms such as the Deep Reinforcement Learning (DRL), is not guaranteed. The conundrum is rendered more complicated. On the one hand AI-based solutions are often empirically found to work in practice, but do not offer any formal proof of correctness and are, most rightfully, often viewed with scepticism. On the other hand, classical control techniques offer a solid theoretical base, but either require approximations such as linearisation, or do not always offer closed-form solutions, often becoming a prohibitive endeavour even to graduate scholars in the field.

The question is then as follows: in control system engineering, a field characterised by sound theories and precise mathematical formulations, is there a place for AI-based solutions?

When AI-based models deliver false or misleading information, it is said that the model is undergoing a *hallucination* [46]. In other terms, the result of the model is incorrect, and, despite the fact that it works in the overall majority of instances, it is at times failing. How can we then trust a model that is producing excellent results in the overall majority of cases but, at random times, hallucinates? One example of a hallucination is as follows [47]:

> ***Question*** *A man and a goat are on one side of the river. They have a boat. How can they go across?*
>
> ***Answer*** *The man takes the goat across the river first, leaving the boat on the original side. Then, the man returns alone with the boat. He leaves the goat on the other side and takes the boat back to the original side. Finally, he takes the cabbage across the river.*

Clearly, in safety critical applications, such as the driving an autonomous car, or the control of an AUV navigating under the ice caps, hallucinations are not tolerable.

In this thesis, an attempt is made to exploit novel AI-based technologies, without compromising on the classical formal theories and the associated mathematical formulations, eliminating any possibility of an occurrence of a hallucination. Specifically, AI-based technologies will be employed to assist the tackling of non-trivial control problems, such as computing robust FTC solutions for AUVs, without requiring the

use of sensors to confirm the presence of faults and without compromising on dynamical model linearisation.

To find a solution to the problem, the first section of the work was dedicated to researching FTC methods, as well as the possible causes of faults on AUVs and their impacts on the vehicle components. The following segment was dedicated to devising a simulation framework allowing the investigation of dynamical models of different AUVs. Next, an AI-based control method with formal proof of stability was designed and verified. Finally, the method that was developed was extended to account for the presence of faults at actuators, and showcased in diverse AUV case studies.

1.2 Thesis outline

The remainder of this thesis is organised as follows.

- Chapter 2 covers previous research in the field of FTC methods, their application to the case of AUVs affected by actuator faults and modelling and simulation of AUVs.
- Chapter 3 illustrates the research gaps prompting the research questions, and the research approach selected to tackle the research questions that have been posed.
- In Chapter 4, dynamic modelling of AUVs is introduced and a framework to simulate different vehicles is presented.
- Chapter 5 outlines the proposed method to synthesise control functions with AI-based techniques, without compromising on formal stability guarantees.
- Chapter 6 extends the method to account for different types of faults at actuators.
- A conclusive case study illustrating how the methods and tools devised can be integrated in an organic study is presented in Chapter 7.
- Conclusions are presented in Chapter 8, alongside discussing future research directions.

Chapter 2

Literature Review

2.1 Introduction

This chapter outlines the current state-of-the-art in the field of faults at actuators in underwater vehicles. First, the causes and effects of actuator faults will be investigated. Next, an overview of the fault-tolerant control field is presented, followed by a taxonomy of the methods available against actuator faults. The prevailing status of actuator fault accommodation in AUVs will then be detailed, with emphasis on the recent trend of machine learning-based methods. To conclude, current available methods that can be employed to model and simulate complex AUV concepts are discussed.

2.2 Preliminary concepts

In this thesis, the term actuator is used to denote a device that responds to an input signal with an action into the physical world [48]. Such action can be a linear displacement, an expansion or contraction, a change of electrical signals, amongst others. Conversely, a sensor is a device that responds to a physical input with a recordable, functionally related output, usually being in the form of a calibrated electrical, mechanical, pneumatic or optical response [49].

Next, a clarification of the use of words faults and failure as they will be employed in this thesis is provided, as diverse definitions of such words have been provided over the years. In this thesis, a fault is defined as an undesired abrupt change in the dynamics of a signal of a sensor or an actuator [50]. More specifically, a fault was formally defined as a change in the characteristics or in the performances of a component that does

not compromise the entire functionality of the parent system [51]. In other words, a fault is an event causing the change of the characteristics of a component from its nominal behaviour to act in an undesired manner. A fault can be worked around so that the overall system can still accomplish its original task, even with a level of performance degradation [52].

Faults can be classified based on temporal behaviour as *permanent* (the fault persists without ever disappearing after its first occurrence), as *transient* (the fault appears only "una tantum" for a finite time), as *intermittent* (the fault occurs repetitively, for finite time period each time) or as *steadily increasing* (due to progressive performance degradation – at times referred to as *incipient*) [53, 17]. Permanent faults are usually the result of hardware damage. Transient faults are at times linked to temporary actuator freezing, clogging or jamming. Intermittent faults can be due to odd electrical contacts. Steadily increasing faults are often associated with wear and ageing [52].

In contrast to component faults, failures denote the inability of a system to accomplish its function [54]. Once a failure is recorded, the overall system is irrecoverable and it needs to be shut down and external intervention is required.

2.3 Causes of faults in AUVs

During an underwater vehicle deployment, a noticeably significant variety of unplanned for conditions can occur and jeopardise the mission success. In a survey of 58 UGs operating over a four year period, faults were reported as being due to mechanical, logistical and environmental causes [55]. The issues linked to mechanical failures comprise incorrect design, leaks and malfunctioning components, while the logistic-driven faults are typically linked to human error, examples being incorrect ballasting and trimming set up. Mechanical faults in AUVs can be associated with any of up to nine subsystems: power, leak detection, diving, environmental detection, collision avoidance, computer hardware, propulsion, communication and navigation [56].

Further research surveyed faults in two REMUS-100 AUVs [57]. Out of the 186 missions recorded, 37 unique faults were detected and 28 missions were aborted due to failures and not by pilot decision. The majority of faults were not hardware or

software-related, but rather driven by external environmental factors, e.g. collisions with debris or other vessels. Another recent survey supports these results, reporting that 84% of the research on faults in AUVs entails actuator malfunctions, owing to the persistent exposure of the actuators to the surrounding environment as compared to the sensors which are normally protected by being housed within the vehicle's hull [58].

While most of the mechanical faults can be mitigated with effective planning and preliminary mission tests, environmental failures can be more severe and difficult to forecast due to their abrupt and chaotic nature. The frequency and the severity of faults are likely to increase as AUVs are deployed in more challenging environments, such as when conducting missions under ice caps [41], or near hydrothermal vents (TUNA-SAND AUV) [59], or other otherwise inaccessible areas such as flooded mine pits (EVA ROV/AUV) [60], (UX-1 AUV) [61].

As an example of failures that occur while operating in extreme environments, a modified version of the *Autosub* AUV designed to penetrate into cavities beneath the Fimbul Ice Shelf, Antarctica [42] was lost in 2005 due to unknown causes. In 2010, the ABE AUV disappeared on its journey to the surface during the ascent phase, and the final cause of the loss was never confirmed [62]. During another Antarctic mission, the *Romeo* ROV experienced a block of ice getting trapped within the propeller blades, requiring remote human intervention to disable the affected thruster [39]. In 2018, during one of the Antarctic deployments, the *Autosub Long Range* AUV encountered an issue while penetrating 20 km under the ice shelf, when the rudder got stuck at full travel for 3.5 hours due to the combination of almost freezing temperature and water within the actuator voids [63].

Collisions with other vessels were reported in AUVs during the communication phase of the missions, when the vehicles are floating at the sea surface. When the vehicles surface, particularly in congested areas, pilots deliberately try to minimise the communication time in an attempt to reduce the risks of accidents, but, at times, collisions with passing vessels have occurred. For instance, a Seaglider UG was reported losing communication and could not be subsequently recovered [35]. It was assumed that there was most likely a collision with a ship navigating in the area. It is surmised

that the antenna, one of the most fragile and critical vehicle components, was compromised during the collision and prevented further retrieval operations. At least four UGs were confirmed as having collided with other vessels in just four years [55]. In further reports, two UGs each lost a wing, theorised to be the result of collisions with other vessels [40].

Obstacles such as drifting fishing debris pose an additional threat to AUVs. Research efforts are ongoing to equip AUVs with algorithms for the early detection of fishing nets using sonars [64]. Several UGs got tangled in deployed fishing nets and were subsequently taken onboard fishing vessels [65, 66, 67]. Two further UGs got trapped in deployed nets and were consequently recovered by the associated fishing boats during the period between 2008 and 2012.

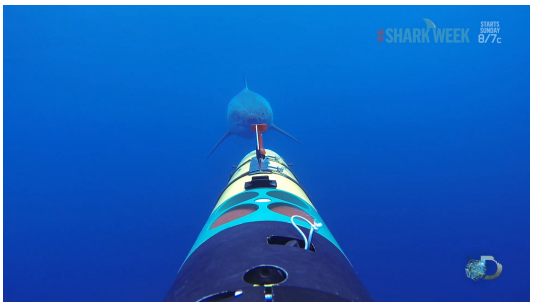
Another set of disruptive events encompass encounters with marine creatures. Shark attacks were identified after recovering the vehicles as testified by bite marks left on the hull, or were detected during the mission time due to damaged wings or external sensors [68], or due to very erratic movements detected in the depth profile. For instance, a white shark attacked a MBARI LRAUV [69], leaving bite marks and a set of teeth in the pressure hull. The attack resulted in the AUV being almost flipped over, with the roll angle abruptly spiking to almost 180 [deg] in just a few seconds. An example of the bite marks left on the hull of Seaglider UG operated by the National Oceanography Centre (NOC) (Southampthon, UK) is reported in Fig. 2.1.

Additional shark attacks were reported in the seas off Australia and Central America, where sizable sharks severely damaged the hulls of an UG and an AUV, as illustrated in Fig. 2.2a and Fig. 2.2b, respectively.

Additionally, biofouling effects encompassing the growth of barnacles and mussels can become critical for vehicles deployed for weeks or months, as illustrated in Fig. 2.3. Marine growth can reduce the efficiency and the range of motion of the control surfaces or reduce the thrust generated by a propeller. Long duration UGs missions deployed in shallow tropical waters near the equator are particularly prone to biofouling [72].

Figure 2.1: Shark bite marks on Seaglider operating in UK waters - courtesy of NOC.

(a) Shark bite marks on an UG [70].



(b) Shark chasing a REMUS AUV before striking [71].

Figure 2.2: Shark attacks recorded on autonomous vehicles.

With the increased interest towards deploying AUVs in ever more challenging environments and on long-lasting missions, a significant variety of unpredictable environmentally-driven faults can occur. The effects of faults, especially at the actuators, need to be mitigated to prevent a component fault evolving into a system failure.

2.4 The fault-tolerant control field: an overview

In accordance with the definitions provided in Section 2.2, FTC refers to the set of hardware and software systems responsible for preventing a component fault from

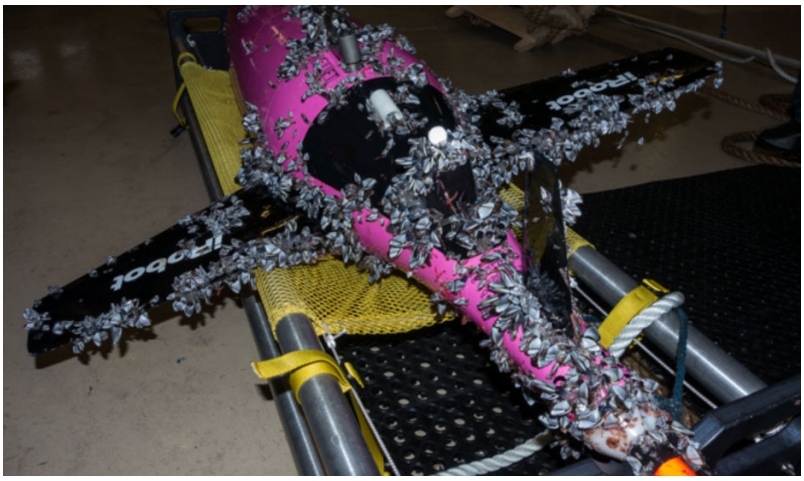


Figure 2.3: Seaglider affected by barnacles' growth [16].

evolving into a failure of the overarching system [54]. Naturally, FTC constitutes one of the building blocks of a feedback loop [73]. The FTC block adds robustness and operating margin to the overall feedback loop, which is conventionally designed assuming that the system is operating around target nominal conditions. FTC architectures aim at keeping the system operational, guaranteeing system stability even under degraded performance conditions (stability-related concepts are formally introduced in Section 5.2) [74].

Confusion with terminology often arises as the term FTC is at times used to broadly denote a specific branch of control theory, and, at other times, FTC is used to denote the methods employed to overcome the faults at sensors and actuators. In this thesis, the term *FTC field* will be used to denote the overall field of research, while FTC will be used as a synonym of the term Actuator Fault Accommodation (AFA), as further detailed in the present section.

The FTC field dates back to the 1970s originating from within the aerospace industry. A significant number of surveys on the FTC field were presented over time, remarkably in 1976 [50], 1988 [75], 1997 [76], 2015 [77] and 2020 [78].

An FTC architecture needs to manage two tasks: Sensor Fault Detection, Identification, and Accommodation (SFDIA) and Actuator Fault Detection, Identification, and Accommodation (AFDIA) [79]. The Sensor Fault Detection, Identification (SFDI) involves the monitoring of the degree of deterioration of the accuracy in the measure-

ments, this conventionally being achieved by inspecting threshold violations or rapid variation in spectral property [52]. The Sensor Fault Accommodation (SFA) performs the task of replacing the corrupted measurements with appropriate estimations.

More specifically, Fault Detection refers to deciding whether a fault has occurred in the first place, while Fault Isolation entails determining which component is being affected, and Fault Estimation defines the estimation of the fault severity [52]. Fault Detection and Isolation (FDI) is traditionally carried out by *consistency-based* techniques, which are function of residuals between a nominal signal and its measured (or estimated) current value [52].

In a similar fashion, the AFDIA includes the Actuator Fault Detection, Identification (AFDI) and the AFA. The AFDI manages the anomalies and the possible causes of actuators malfunctions, while AFA is tasked with applying suitable control actions to the system in keeping with to the most up to date information supplied by the AFDI.

A taxonomy of FTC research field is provided in Fig. 2.4.

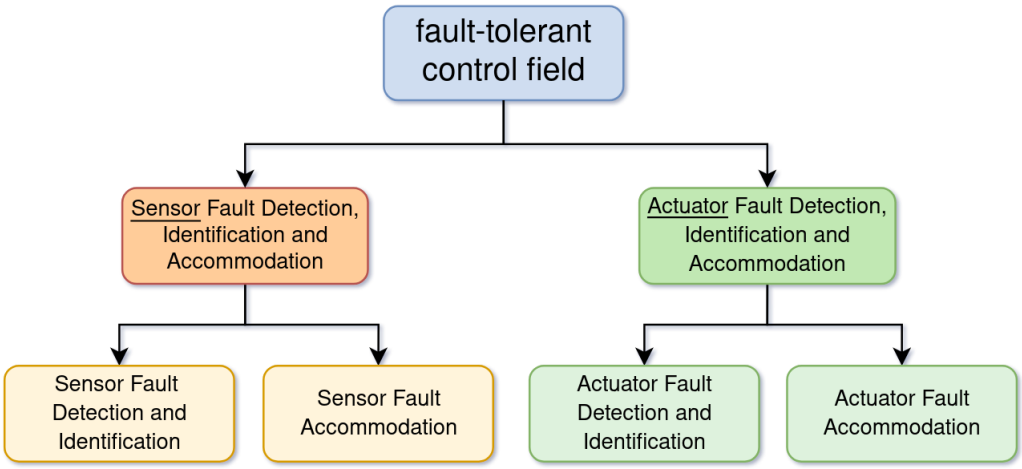


Figure 2.4: The fault-tolerant control field: a taxonomy.

Given the causes of faults surveyed in AUVs proffered at Section 2.3, it is apparent that actuators represent a key point of weakness in the operation of these vehicles. Owing to the hazardous nature of the ocean environment, faults at actuators have been recorded as the cause of disruption of a range of missions, involving AUVs, ROVs and UGs. On top of identifying the faults and their severity, it is therefore relevant to

delve deeper into the AFA strategies proposed thus far to tackle the case of an actuator malfunction.

2.5 Actuator Fault Accommodation: a taxonomy

Given the specific focus of this research into the effects of actuator faults, in accordance with the definitions provided in Section 2.4, the term FTC will be used henceforth as synonym of AFA. This choice of terminology is consistent with that used in the majority of the resources surveyed in the following section.

FTC methods are conventionally divided into either *Active* or *Passive* techniques. The active fault-tolerant control (aFTC) methods exploit Fault Detection and Isolation diagnosis systems [80] and cover a range of diverse control architectures. aFTC architectures comprise a range of strategies, such as switching between a set of pre-computed control laws, online re-tuning of control gain, or even redesigning of the controller structure, among other approaches. In other terms, aFTC refers to all the methods that entail the updating of control gain following the occurrence of a fault [52].

In the following subsections, a review of the key concepts and methods within the field of aFTC is provided.

2.5.1 Multiple-Model aFTC

The Multiple Model (MM) control is a class of methods based on the definition of a finite set of linear models capturing the dynamics of the system at different operating points, for instance associated with a set of (actuator) faults [17]. MM aFTC is classified as a projection-based method, which relies on the idea of dynamically selecting control laws based on a set of pre-computed gains [81].

MM aFTC consists of defining a parametric dynamical model, in approximating the nonlinear dynamics with a corresponding linear model and in associating to each linearised model an ad hoc control law which has been pre-designed off-line [17]. The applied control action is based on a probabilistic weighted combination of possible control actions, where probabilities are assigned based on the likelihood that each linear model has to represent the current dynamics.

In short, the overarching concept relies on four stages: a) anticipate possible faults; b) develop a (linearised) dynamical model for each fault; c) design a control law for each fault; d) design a gain switching logic.

Within the MM aFTC, two methods stand out: Multiple Model Switching and Tuning (MMST) and Interacting Multiple Models (IMM). When a fault is detected, if the MMST is employed the problem is formulated as "which dynamical model best resembles the current operating dynamics" and hence which model-control pair should be applied. If the IMM is employed instead, when a fault is detected, the applied control law is based on a convex combination of all the pre-computed fault models.

Both MMST and IMM suffer from a range of issues. First, a closed-loop system controlled using the MMST method is stable if the set of models is sufficiently dense and if the sampling rate of the dynamics is fast enough [82, 17]. Additionally, a time delay is required to prevent fast switching of the control law which could lead to closed-loop dynamic instability. With the IMM method, the assumption that every fault mode can be described as a convex combination of other models has not been proved, and no clear approach to define the model set has yet been established [17].

Moreover, MM methods cannot account for uncertainties in the dynamics and cannot cope with actuator saturation. Finally, both methods rely on linearising the dynamics, leading to the standard local-only arguments associated to the linearised control theory.

2.5.2 Adaptive aFTC

The adaptive aFTC method attempts to reconfigure the control law online, without necessarily detecting and isolating the fault. The adaptive aFTC problem assumes to deal with a time-varying dynamical system. Time-varying dynamics might be associated with environmental disturbances, to unmodelled dynamics or to system bifurcating, e.g. due to the occurrence of a fault. To cope with dynamic variations, the feedback control gain is continuously updated in response to the current operating conditions of the dynamical system [83].

A popular control method within this class is the Model Reference Adaptive Control (MRAC). MRAC relies on designing a target dynamical model representing an

ideal behaviour, and setting the closed-loop to track such target, independent of the current state of the dynamics. This scheme relies on two key elements: an adjustment mechanism, which is in charge to evaluate the residuals, i.e. the difference between the ideal system output and the measured (or estimated) output, and the control law, that is updated to mimic the desired dynamics [84].

Several control schemes capable of handling different combinations of disturbances, by modelling uncertainties and actuator faults were proposed [81, 84, 85]. In general terms, the key advantage of the adaptive aFTC is its capability to react to unmodelled dynamics. In other words, there is no need to specify an apriori set of possible dynamics associated with the faults.

On the downside, adaptive aFTC copes better when parameter variations are slow, with such method possibly failing to deliver satisfactory performance under abrupt or severe fault conditions [83]. Upon the occurrence of a permanent fault, typically associated with a hardware fault, adaptive aFTC can suffer from closed-loop instability unless the gain is rescheduled in a timely manner. In general, adaptive aFTC does not formally offer such a guarantee on timely updating of the control gain, and thereby leading to the system possibly entering instability regions (i.e. a failure mode).

2.5.3 Model Predictive Control aFTC

Model Predictive Control (MPC) is established as a widely employed control method in fault-free control scenarios. MPC has its roots within optimal control theory applied to multivariable nonlinear systems with input and state constraints [86]. As computing an analytical solution for the optimal control problem for nonlinear systems with constraints is computationally demanding, MPC attempts to achieve results within a finite time span, usually relying on linearised system dynamics [87].

Due to the nature of the MPC formulation, such a control method is particularly suited to systems with faults, as the limits of the actuators (or model dynamics) can be promptly re-defined following the identification of a malfunction [17]. Owing to the same features, the MPC aFTC method can account for an actuator that is jammed at a fixed position, by setting a control variable to a constant value with its associated rate of change being set equal to zero.

However, the performance of the MPC method is tightly bound to the accuracy of the underlying dynamic model [17, 86]. Additionally, MPC-derived control laws can only guarantee local closed-loop stability. During the development of this thesis, a first attempt was carried out to extend the conventional (linear) MPC aFTC method as a nonlinear method [88]. Such extension entails employing Nonlinear Model Predictive Control (NMPC) to compensate for the faults on motors, and was demonstrated in a real-time control application. On the downside, this method assumes, at this stage, exact knowledge of the system faults through an idealised FDI block.

2.5.4 (Embedded) Control allocation aFTC

The Control Allocation (CA) aFTC, at times designated as Embedded CA, represents an instance of a cascade control system, where several loops are nested one inside another, each outputting a value that is fed as the reference to the next inner loop. In rigid-body control applications, conventionally, an outer loop generates a reference vector of desired virtual control inputs (i.e. forces and torques) to the innermost loop, tasked with generating commands to the actuators to match the required force and torque outputs [89].

The innermost loop, referred to as the CA loop, allocates the control effort required from the available actuators using a matrix denoted as Thruster Allocation Matrix (TAM), or Control Effectiveness Matrix (CEM), or Thruster Configuration Matrix (TCM). The TAM is a matrix collecting the geometric properties of the actuators, namely position and orientation with respect to a designated origin, mapping how the force of an actuator affects each Degree Of Freedom (DOF) of the body it is attached to. Within the CA aFTC, the TAM is updated in real-time, allocating the desired virtual inputs to the available actuators, excluding the faulty ones.

The solution of the CA problem depends on the number of available actuators. In the most general formulation, the TAM is not a square matrix, leading to an ill-posed problem, i.e. the solution is not unique. The solution is usually computed employing a pseudo-inverse (or generalised inverse) function. Under the special condition regarding the TAM being full-rank, the solution can be trivially computed by means of the Moore-Penrose pseudoinverse, otherwise more advanced techniques employing regu-

larisation terms are required. When fault information is available, the inverse of the TAM can be computed by including a diagonal weighting matrix, set to an identity in the nominal (faultless) case, and where entries set to zero denote a fault at the associated thruster. Solving the weighted Moore-Penrose pseudo-inverse problem produces a feasible thruster allocation solution, excluding the faulty thrusters.

In line with the previous aFTC methods, CA relies on precise information regarding the status of the actuators being provided by an external FDI block. Moreover, closed-form solutions only exist when the actuator saturation is neglected. When actuator saturation is to be included, more advanced solution techniques are required, among which there are the redistributed pseudo-inverse, the daisy chaining, the direct allocation and numerical solutions based on linear, quadratic, nonlinear and mixed-integer programming [90].

2.5.5 Reconfigurable Eigenstructure Assignement aFTC

The idea behind the Reconfigurable Eigenstructure Assignement (REA) method is to place the eigenvalues of a linear (or linearised) system in desired locations (within the Gauss plane) using a state-feedback control law, and using any remaining design freedom to attempt aligning the eigenvectors as desired [91]. As the eigenspace determines the shape of the dynamic response, the REA attempts to match the eigenspace of a faulty system to that of a nominal dynamics [17].

A control law leading to the eigenspace that exactly matches a target solution only exists when redundant actuators are available [52]. When the dynamical system does not possess redundant actuators, approximate solutions minimising the difference between the target eigenvalues and the real eigenvalues can be designed.

The REA aFTC is a particularly attractive method owing to the availability of closed-form solutions, which in turn results in the possibility to compute and implement control laws with limited computational burden. In the context of FTC, the control gain can be re-calculated online based on the current available actuators and their associated Jacobian control matrix.

On the downside, the ability to place the eigenspace is limited by conditions related to the control and output Jacobians remaining full rank following any one fault.

Additionally, actuator saturation cannot be accounted for using this method. Finally, the effect of the eigenvectors of the faulty system not fully matching those of the nominal system is yet to be extensively studied and understood [17].

2.5.6 Sliding Mode Controller aFTC

The Sliding Mode Control (SMC) method tackles the design of control laws for nonlinear dynamical systems that are affected by model and parametric uncertainties [92]. SMC is characterised by a series of feedback control laws, and a decision rule (at times termed a switching function) [93]. The decision rule is tasked with applying the most suitable control law based on the input measurements.

The SMC method is suited for use in FTC applications owing to the method's ability to cope with model uncertainties, which faults are are a particular case of. On the positive side, SMC can cope with actuator saturation and with actuator rate limits, at times employing terms that dynamically adapt to fault modes [94], setting this method apart from those illustrated thus far.

A first disadvantage of the method is the necessity to implement a discontinuous control signal which, in theoretical terms, must switch at infinite frequency to provide total rejection of uncertainty [93]. The discontinuous nature of the control signal generates high-frequency control commands, conventionally referred to as chattering, lowering control accuracy and increasing wear in mechanical systems [95] and heat in electrical power circuits [96]. Actuator chattering can be mitigated by smoothing the control functions, at the cost of sacrificing the full ability to reject disturbances [93]. Further limitations of the method are linked to the robust nature of the SMC, that requires the knowledge of (or the assumption of) an upper bound to the uncertainty [94]. Additionally, in the general formulation, total loss of actuator efficiency can not be accommodated [17]. Extended SMC formulations were proposed to deal with full loss of efficiency scenarios, but might result in overly excessive conservative controllers.

2.5.7 Conclusions on aFTC methods

Active methods can generally ensure satisfactory control performance following a fault, but are at times computationally expensive (i.e. NMPC) and rely on precise fault

information provided by an FDI block (i.e. MM, NMPC, CA, REA). When closed-form solutions exist, thus avoiding the need to solve optimisation problems online, either the methods rely on linearisation assumptions (i.e. MM, MPC, REA), or are prone to limitations regarding the parameter evolution being slow (i.e. Adaptive aFTC) or assumptions on redundancy need to hold (i.e. REA, SMC), or the ability to cope with actuator saturation in an explicit form is lost (i.e. MM, CA, REA). Further limitations associated with each method were described in the dedicated sections.

A range of additional disadvantages characterises this class of methods. aFTCs might suffer from dynamical instabilities during the transient period (from when the faults occur to when they are detected and isolated), as, during this time interval, the faulty system is still controlled by the nominal controller [53]. More specifically, the term reaction time is used to define the sum of the time the FDI requires to identify the fault, the time to reschedule the control law, and the time to apply the corrective control actions [97]. If the reaction time is excessive (dependent on the specific application and fault), the system might enter an unrecoverable state, i.e. a failure mode. Similarly, if the FDI system is not conveying precise and timely information, the dynamics might violate state and input constraints [54]. Moreover, the control reconfiguration mechanism needs to be sufficiently fast, and must select the correct control law to prevent triggering instabilities [48].

A further speculative issue, i.e. not supported by current theories, is that a FDI system designed to detect a fault with an minimum of delay (in an attempt to minimise possible instabilities associated with the reconfiguration time) might in turn lead to further closed-loop instabilities. In actuality, a fast acting FDI algorithm might increase the detection of false positives, triggering the reconfiguration of a control law designed for a different operating mode.

In opposition to aFTC methods, passive fault-tolerant control (pFTC) methods entail designing a unique set of static gains that guarantees stability in both nominal and fault modes [98]. In other terms, pFTC refers to all the methods used to devise control laws which remain static, independent of whether or not the fault is detected, isolated and evaluated [52].

2.5.8 $\mathcal{H}_\infty$ pFTC

In general terms, the pFTC can be formulated as an optimisation problem with the aim of minimising the worst-case setpoint tracking performance of a set of dynamics affected by faults [99]. Stabilising controllers for linear and linearised systems can be designed with the objective of minimising target conditions such as the $\mathcal{H}_\infty$-norms (or $\mathcal{H}_2$-norm) between the exogenous inputs and the desired performance [52].

The $\mathcal{H}_\infty$ control method is particularly suited for application to FTC problems. Given a set of prescribed faults, the $\mathcal{H}_\infty$ pFTC aims to design a controller for the worst-case scenario, usually coinciding with the fault at one actuator [100]. This method is also known as reliable control [99].

In its original form, the $\mathcal{H}_\infty$ control method is frequency-domain based. This formulation, when applied to multiple-input, multiple-output systems, renders the design phase to be more convoluted when compared to time-domain control specification descriptions. Additionally, when design constraints such as desired state limits are included in the formulation, the solution of an $\mathcal{H}_\infty$ problem becomes very challenging or even undecidable [101]. Moreover, such a method often results in high-order control laws, increasing the computational burden to synthesise a solution, especially for high dimensional systems [102].

Upgraded variants of the method were proposed, such as the structured $\mathcal{H}_\infty$, comprising a fixed structure of the controller with the possibility to add control requirements in the time-domain [101]. Nonetheless, the structured $\mathcal{H}_\infty$ problem results in a system of Bilinear Matrix Inequalitiess (BMIs) to be solved, which is conventionally non-convex. To conclude, further limitations of the $\mathcal{H}_\infty$ pFTC include the linearised nature of the system description, and the inability to strictly enforce state and input constraints.

2.5.9 Lyapunov-based pFTC

In an attempt to overcome the limitations of the linear pFTC methods, nonlinear control theory was employed to envision new nonlinear pFTC methods. One such example comprises the use of Lyapunov theory (which will be discussed in Section 5.2) to ensure stability of the closed-loop system [44]. This method is henceforth referred to

as Lyapunov-based pFTC.

The Lyapunov-based pFTC method involves the augmentation of a nominal control law designed for the nominal (fault-free) dynamics with an additional term tasked with addressing possible faults. In such a formulation, faults at actuators are described as additional or multiplicative factors. Starting from the nominal control law and assuming to have computed a Control Lyapunov Function (CLF) for the nominal closed-loop system (with the formal definition of CLF provided at Definition 4), this method proposes an extra control term to manage the fault modes. This extra control term, once added to the nominal control law, ensures robust stability in all operational modes.

The most attractive feature of this method lies in the simple (and elegant) closed-form solution. This method follows an intuitive design and tuning process, and results in a computationally inexpensive implementation. However, the method has three inherent disadvantages. First, it assumes the availability of a CLF, which is not trivial to compute for generic nonlinear systems. Second, formally the method can only cope with partial loss of efficiency, i.e. the control law cannot cope with the total loss of an actuator. Finally, the greater the loss of actuator efficiency that needs to be accommodated, the more conservative the resulting control law becomes. When practical fixes are attempted by including an almost complete loss of actuator efficiency (e.g. 99%), the control law often demands unreasonably high control effort.

2.5.10 Machine Learning-based Passive pFTC

The field of machine learning (ML) focuses on the design of computer programs that can automatically improve the ability to complete their tasks as their experience progresses [103]. Despite the success in other fields of computer science and engineering disciplines such as speech and vision synthesis and robotics, as of 2020, the start time of the present project, no ML-based had been applied to solving pFTC problems [104]. During the development of this thesis, two methods employing ML methods tailored to pFTC applications were proposed in parallel.

A first method was envisioned for application in cyber-attacks [48]. The underlying concept is built upon employing a reliable simulated version of the system to be controlled. Next, fault at actuators (as well as sensors) are injected in a set of sim-

ulation runs, while the system is controlled via a traditional control law. During the simulations, input and output dynamics are recorded. At the end of the data collection stage, a neural controller is trained to maximise the capacity to track the time-varying reference while being fed with the current target reference, the previous control input and the system output. Within this control framework, different neural architectures were proposed, encompassing Long Short-Term Memorys (LSTMs) and ensembles of LSTMs at times combined with proportional integral derivative (PID) controllers. Such method was trialled in a simulation environment showing the ability to reject faults at actuators. This method is denoted as LSTM-pFTC in the following sections.

A second method, named fault-tolerant bio-inspired controller (FT-BIC) [105], is designed to compute a control law in the form of an Artificial Neural Network (ANN) based on the DRL training paradigm. The control synthesis problem is formulated as the iterative solution of an optimising problem, aimed at minimising the state distance from a target equilibrium. During the training the faults are injected at the actuators, but never fed back to the control system, that remains oblivious on whether a fault has occurred.

The LSTM-pFTC and FT-BIC control methods represent a first step to the employing of an ML-based approaches to solve the pFTC problem, but report certain limitations. First, the synthesised control laws are empirically verified to show resilience to faults, but, at this stage, no theoretical guarantee as regards to closed-loop stability is proffered. In other terms, classical control performance criteria, such as robustness in the presence of uncertainties, phase or gain margins, linear or nonlinear stability properties, cannot be certified.

These methods offer innovative solutions that overcome the limits previously highlighted in the other pFTC methods, such as $\mathcal{H}_\infty$ pFTC and Lyapunov-based pFTC. Nonetheless, the lack of stability guarantees is a source of concerns in real-world safety critical applications, e.g. control of nuclear fusion reactors and power grid regulation [106]. Further effort is required to bridge the gap between the significant potential such methods hold and their widespread applicability to real-world research and industrial applications.

2.5.11 Conclusions on pFTC methods

pFTC methods can overcome different types of faults while preserving the system stability under different operating modes. However, these methods suffer from common limitations such as either relying on linearisation assumptions (i.e. $\mathcal{H}_\infty$ pFTC), or compromising on the loss of actuator efficiency that can be tolerated (i.e. Lyapunov-based pFTC). Other methods need an accurate simulator to generate the training dataset (i.e. LSTM-pFTC, FT-BIC). Additionally, none of the reviewed methods can explicitly ensure actuator saturation is not exceeded (i.e. $\mathcal{H}_\infty$ pFTC, Lyapunov-based pFTC, ML-based methods). Moreover, pFTC methods can only overcome the predefined set of faults they are designed to mitigate, and cannot formally cope with fault modes beyond their design scope [44]. Finally, pFTC methods comprise reduced operational performance, both in the nominal and fault modes, due to the intrinsic epistemic uncertainty under which these control laws operate.

Nonetheless, pFTC methods provide certain advantages when compared to aFTC methods. First, pFTC methods do not require the design of a FDI scheme, reducing both the overall cost and complexity of the control architecture design. In turn, the absence of a FDI scheme prevents a range of issues related to closed-loop instability that may be triggered either by delayed or inaccurate information flow. pFTC methods are also comparatively easier to implement and can be run without limitation on comparatively basic hardware, as the control laws are based on fixed gains as opposed to relying on the computation of solutions online, as required by the aFTC methods.

It is apparent that choosing whether to employ either an aFTC or a pFTC control scheme depends on the application of interest and on the acceptable trade off between stability requirements and control performance [17]. A taxonomy of these AFA methods is reported in Fig. 2.5.

2.6 Actuator Fault Accommodation in AUVs

Having now detailed a taxonomy covering the main aFTC and pFTC methods, this section focuses on a review of the current state of the art of AFA, specifically in AUV

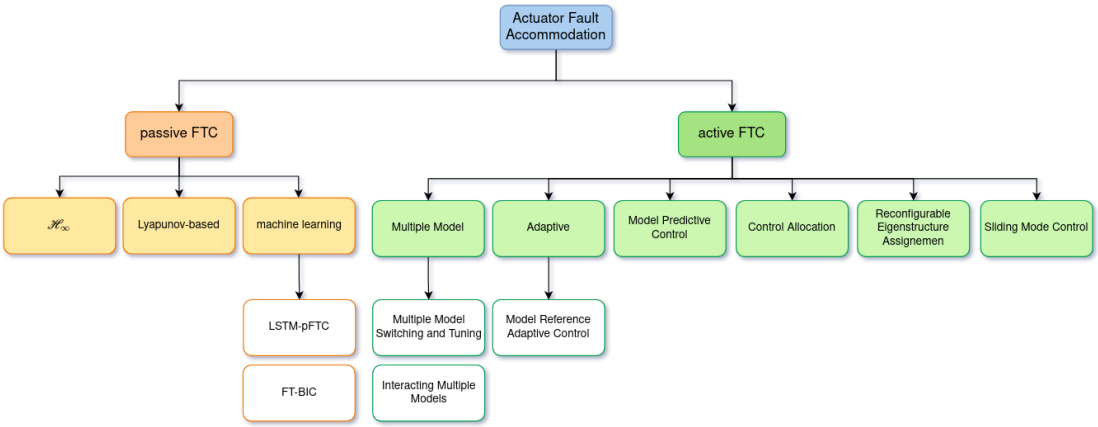


Figure 2.5: Actuators Fault Accommodation methods taxonomy (adapted from [17]).

applications.

Recent surveys reveal that a significant stream of research of AFA in AUV applications assumes the availability of both an FDI scheme and redundant thrusters [107, 58]. Under such assumptions, the CA aFTC method is usually employed, which entails the redistribution of the required control forces across the healthy thrusters, initially with actuator saturation limits assumed to be negligible [108, 89, 109]. When the thruster set is redundant, i.e. when the TAM is rectangular, it is possible to turn off the broken thruster and still retain control of the AUV over the 6 DOFs. More advanced CA formulations were devised to account for actuator saturation [110, 111], in turn reducing issues linked to limit cycling instabilities [112]. The common choice of employing a CA method follows on from the relative simplicity of embedding AFA capabilities in the traditional control schemes, which can be achieved by means of a closed-form solution without modifications to the classical cascade control scheme where actuator forces are generated independent from the outer loops.

Another research stream focused on developing adaptive aFTCs, based on the idea of considering the faults as part of the model uncertainty to be compensated for [113]. Such methods rely on defining two control laws: a fixed-gain controller designed for an ideal model without disturbance nor uncertainties, and a second adaptive control law to compensate for model uncertainty. In line with the general adaptive aFTC concepts reviewed thus far, this class of methods suffers from a delay from the fault estimation and its compensation (up to 35 [s] in this exploration). Additionally, this methods

assumes a certain onboard computation availability to complete the required adaptation online.

Next, the MPC aFTC method was used to control faulty AUVs. One such example involves designing an outer kinematic control loop with a standard MPC law combined with an inner robust loop to control the dynamics [114]. Once a fault is detected, the fault information is propagated in the form of an equivalent maximum desired force vector, which is in turn translated in to maximum desired velocities, which the MPC can account for during the solution of the optimisation problem. To overcome the linearised nature of the MPC aFTC method, a control system combining NMPC and SMC was proposed and validated in the context of AUVs [115]. Such a scheme inherits the capability to handle constraints inherent to the NMPC method, and embeds the capacity to accommodate uncertainties that results from the SMC formulation. Despite the significant advantages, this method suffers from a lack of formal theoretical stability.

SMC schemes alone, i.e. without being mixed with other methods (e.g. MPC), were also successfully designed and implemented onboard AUVs. Assuming that a reliable FDI system is available, SMC aFTC methods were developed and were shown to be able to promptly compensate thruster faults by rescheduling (i.e. adapting) the control gain, while additionally tackling conventional SMC limits such as chattering [116].

A different stream of research focused on pFTC methods, eliminating the need for FDI schemes. When compared to the aFTC sources surveyed thus far, a comparatively limited number of reports on pFTC methods applied to the control of AUVs is available. A robust $\mathcal{H}_\infty$ pFTC method was proposed to manage the guidance (in the form of waypoint following) and control (low-level control allocation) problems [117]. Such $\mathcal{H}_\infty$-based methods can additionally be employed to detect and isolate the faults, addressing what is defined as an integrated regulation and diagnosis problem, designed to switch to another controller should excessive change in the dynamics be detected [117]. In line with comparable research reports employing $\mathcal{H}_\infty$ methods in the AUV field [118, 119], issues that were highlighted included the difficulty encountered when synthesising the controller as the model uncertainty grows, the non-intuitive nature of

the weighting matrices, and the need to select a sensible model linearisation condition.

Finally, in line with the idea of extending the robust linear results to nonlinear control systems, pFTC methods were recently applied to submersible vehicles [120]. Due to the straighforward implementation of the Lyapunov-based pFTC method [44], further applications are expected to arise in the near future in the field. Nonetheless, the method, at this stage, cannot ensure stability for complete loss of actuator efficiency, i.e. can only tolerate partial loss of efficiency, requiring further theoretical advancement before being deployed onboard real vehicles.

After reviewing the information sources detailed in this section, it is apparent that underwater vehicles, such as AUVs, ROVs and UGs, represent a category of platforms operating in highly uncertain conditions, where reliability and safety are often placed at a higher priority than outright performance. In practical applications, such as in UG operations, where energy efficiency determines the ability to extend operational endurance so as to collect additional or more detailed data, employing dedicated sensors to continuously monitor the status of the actuators does not represent an attractive option. Onboard sensors used to estimate position, velocity and attitude are, at times, completely turned off and the entire control system module is, in such extreme circumstances, only switched on intermittently to perform periodic corrective actions [20].

In such operational scenarios, it becomes prohibitive to rely on a control system encompassing an FDI block and on a reconfiguration mechanism, or on prompt online gain adaptation. Moreover, even with assuming that such systems are designed and enough onboard energy is dedicated to the FTC scheme, all these subsystems need to work in harmony to ensure closed-loop stability, with any one failing determining potentially catastrophic outcomes [97].

More broadly, in any application where widespread sensor and algorithm use cannot be assumed due to there being financial cost, power or complexity constraints, pFTC represents the option with the most significant potential impact. Therefore, the pFTC class of controllers is chosen as the area with higher applicability potential and worthy being investigated.

2.7 Recent trends in machine learning-based control for AUVs

In the last two decades, ML-based control systems surged from within the control and computer science communities. Most of the proposed methods employ control laws (or policies) based on training ANNs through learning-based approaches [18]. Upon training completion, an ANN deployed in a control loop solves a sequential decision-making problem, namely the determination of what control input should be applied given the current state of the system [121].

Early works in the research and use of ANNs, e.g. [122] (1989), [123] (1991), and [124] (1993), increased the ANN capability to approximate complex nonlinear functions, eventually culminating in the universal approximation theorem [125] (1998). The universal approximation theorem states that any arbitrary nonlinear continuous function can be approximated with an ANN of sufficiently complex architecture. In actuality, ANNs architectures of increasing complexity were developed in the past decades, with notable results being Recurrent Neural Networks (RNNs) [126], multilayer feedforward networks [124], LSTM networks [127] and Transformers [128].

Architectures employing ANNs were specifically employed to tackle a diverse range of the AUV guidance and control problems. For instance, path planning algorithms were devised using the Reinforcement Learning (RL) learning paradigm to train ANN laws within the Q-learning framework [129], the Deep Deterministic Policy Gradient [130] and the Actor-Critic [131]. Further notable examples in the field encompass the docking of an AUV where the control laws were derived via RL [132], and the control of a hybrid UG dynamics via an ANN-inverse control [133].

These research reports are motivated by the need to devise control laws for complex nonlinear dynamical systems, i.e. AUVs and UGs, subject to dynamic parameters that are particularly costly to derive, and subject to model and environmental uncertainties and disturbances. Under such conditions, ANN-based control laws clearly stand out as an invaluable tool to derive both robust and adaptive laws. On the downside, a shared characteristic of the surveyed works is the lack of formal stability guarantees. In other terms, classical stability results, such as those based on classical control theory as

in Bode and Nyquist criteria, are not proffered. Conversely, ANN-based control laws are often verified with practical (i.e. simulated) tests or validated upon deployment on the real systems.

To cope with the lack of stability guarantees, a most recent control stream focuses on equipping ANN-based control with formal proof of stability. One such example is the NLC, a method to devise ANN-based control laws while guaranteeing formal stability guarantees based on Lyapunov's stability theory [134]. Such a method exploits a training loop generating at the same time a candidate control law and a stability certificate function. Upon minimisation of the target loss function, the correctness of the candidate control law and stability certificate pair is verified over a continuous domain, e.g. a subset of the Reals, either formally guaranteeing or disproving the stability of the obtained solution. Since the proposal of the original NLC method, a range of works followed, extending the nature of stability guarantees that can be provided [135, 136, 137], and focusing on discrete-time systems [138, 139].

The NLC method holds significant potential to exploit the advantages of the ANN-based control methods employed in AUV applications, without compromising the stability guarantees provided. Nonetheless, common limitations of the NLC-works surveyed persist. First, the derived control laws are linear [134, 137], while the nonlinear nature of the ANNs is only exploited in the stability certificate functions. Next, an initial value of the control law is always provided through classical state-feedback solutions, e.g. optimal control methods. Moreover, as the system dimension increases, training and verification times of up to several days were reported [138]. Given the recent nature of this research stream, a limited number of works are currently available, highlighting the possibility for further research to provide clarifications and solutions to the stated limitations.

2.8 Dynamic model and simulation

Different terms exist within the context of modelling and simulation (and control) of underwater vehicles. In this work, the term *traditional* AUV defines an AUV actuated by means of thrusters and control surfaces only. In contrast, UGs are the vehicles

employing a combination of Variable Buoyancy Device (VBD) and internal moving masses as means of propulsion. More recently, the term *hybrid* AUV was used to denote vehicles with mixed actuator sets, namely employing thrusters, VBDs, steering surfaces and internal moving masses [140]. As reported in Fig. 2.6, the term AUV is selected henceforth to denote generically any one of these three classes of vehicles, while the specific terms (i.e. traditional AUV, UG, hybrid AUV) will be employed to refer to the vehicles in individual contexts.

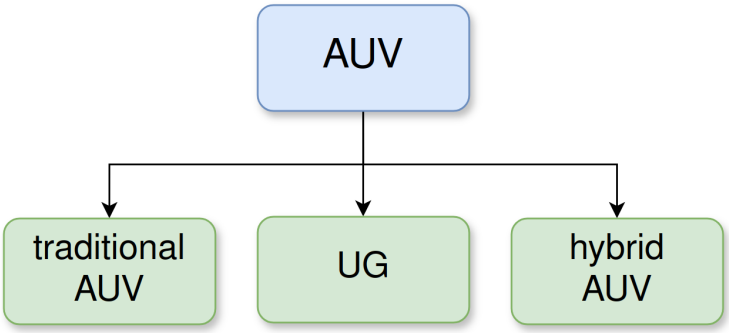


Figure 2.6: AUV terminology as employed in this thesis.

To design and validate control laws in a safe manner, AUVs can either be tested in dedicated tanks or in simulation environments. However, specific size criticalities persist in the cases of certain vehicles, e.g. UGs, the dynamics of which require dozens of meters to reach steady-state gliding conditions, making their tests in physical facilities a highly complex undertaking.

Several AUV simulators were developed for the purpose of testing new vehicle designs, to validate the functionality of sensors and to analyse the interaction with the marine environment [141, 142, 143, 144]. Recently, research focused on designing general underwater simulators, which include prompt adaption to different underwater platforms (e.g. ROVs, UGs, sonobuoys, etc.), while accounting for the integration of conventional underwater sensors (e.g. Doppler Velocity Logger (DVL)) and environmental effects [145]. As of summer 2024, development is still ongoing to simulate the dynamics of the more complex vehicles, i.e. vehicles with internal moving masses

such as the UGs[1].

Underwater vehicles of different types, including ROVs, traditional and hybrid AUVs and UGs, share key elements of fluid dynamic modelling owing to the fluid dynamic interaction of a hull with the surrounding environment. However, these vehicles can be subject to different dynamic modelling procedures given the diverse natures of their physical designs. AUVs and UGs differ in terms of their sets of available actuators: while AUVs mainly use thrusters as the main means of propulsion, UGs employ VBDs and internal moving masses to perform spatial manoeuvres. These different types of actuators lead to specific modifications in the development of the dynamical models.

Dynamical models of underwater vehicles derived following the Newton-Euler formulation were extensively researched in the previous decades [146, 147, 148]. These analytical formulations were focused on rigid-body vehicles with 6 DOFs, suited to describe conventional vehicles with a hull and a standard set of actuators (such as thrusters and control surfaces). Such formalism demonstrated to be satisfactorily suited to describe the dynamics of traditional AUVs.

With the introduction of the UGs, a new vehicle design paradigm started to appear, namely where traditional actuators were combined with internal moving masses, as a means to enhance control ability of the vehicle [149, 150]. An example of UG with a shifting mass is the Slocum (Webb Research Corporation) [151], while vehicles with a mass shifter allowing for both pitch and roll control are the Spray (Scripps Institution of Oceanography) [152], the Seaglider [153] and the Deepglider [154] (both from the University of Washington's Applied Research Laboratory). Advanced glider design concepts as in hybrid AUVs were also developed to further increase manoeuvrability and available speed range [155]. Hybrid AUV designs comprising a combination of control surfaces, jet pumps and VBD include the Sea-Whale 2000 [156], the Folaga [157] and the Liberdade class gliders (Stingray, XRay, ZRay) inspired by the Flying Wing design (2006-2011, US Navy Office of Naval Research) [149].

[1]Ongoing plan to extend the vehicle dynamics to account for moving masses (as of August 2024): https://github.com/Field-Robotics-Lab/dave/blob/master/urdf/robots/glider_description/urdf/glider-_hybrid_whoi.xacro.

Once the dynamic model of a vehicle comprising an arbitrarily complex set of actuators is available, the equations of motions are implemented in a simulator. AUV simulators are conventionally designed based on the causal paradigm. In a causal simulator, the existing connections between subsystem blocks represent the flow of analytical quantities from the output of a block to the input of the next block. In other words, the output of each block is a function of the block inputs in a causal fashion. This is the case for models built, for instance, in Simulink, where the designed scheme reflects the solver calculation procedure more than the physical structure of the simulated system [158].

The second possible design paradigm is defined as declarative or acausal, where each building block embeds a set of equations associated with a physical component (examples being the characteristic equation of a motor, the equations of motions of a rigid-body, etc.), and the connections among blocks serve as a proxy for the exchange of physical variables. As an example, in an acausal simulator, an interface between two blocks might be represented by a mechanical flange, denoting an exchange of force and torque.

In causal simulators, minor changes in the system equations can require substantial modifications of the simulation model. Conversely, the acausal modelling design grants modularity by simplifying the addition of sequential components without the need to assemble a new dedicated system of equations. However, acausal models might give rise to sets of Differential Algebraic Equation (DAE) due to algebraic constraints set between variables, linked to the connection of components one to another [159]. DAEs lead to higher order systems or algebraic loops, in turn requiring more sophisticated solution algorithms to perform symbolic manipulation of variables and reduce the order of the system [159].

Currently, there is a lack of simulators for AUV applications that were designed following the acausal approach. This is most likely due to the recent interest towards such an approach, when compared with the traditional causal approach. All the simulators surveyed thus far are designed with a causal approach, being more limited in the range of actuators that can be promptly embedded onboard the vehicles. In the

currently available simulators, the addition of a new shifting mass, for instance, would require the update of the equations of motion and, in turn, prompt a significant re-design of the simulator. The same task, in an acausal simulator, would more simply require the connection of a new rigid-body (the sliding mass) to the original vehicle body, via a mechanical flange (e.g. a prismatic joint).

Despite disadvantages at the solver level, the acausal approach appears to be better suited to simulate a wider range of underwater vehicles. Moreover, the enhanced modularity capability could be further exploited to support the design of new concept vehicles. For instance, an acausal simulator would facilitate exploring the addition of an actuator on an existing vehicle, in turn investigating the effect on the vehicle's manoeuvrability, or its resilience to faults.

2.9 Conclusions

In this chapter, the causes and effects of actuator faults in AUVs missions were first outlined. An overview of the fault-tolerant control field was presented, with specific focus on the methods available to mitigate the problem of faults at actuators. Finally, methods which can be used to model and simulate different AUV concepts were compared and discussed.

In the following chapter, a summary of the identified research gaps is presented, followed by the research vision defining this work. Research questions are formulated next, and an associated research approach is envisioned.

Chapter 3

Research Approach

3.1 Research gap analysis

Following the literature review presented in Chapter 2, clear research gaps were identified, which will be discussed and summarised in this chapter.

First, given the growing interest in underwater marine robotics applications, an ever more diverse range of autonomous vehicles is proliferating. Simulation of AUVs is rendered particularly complicated by the increasing employment of internal mechanical components to serve as actuators. Modern AUV concepts leverage internal shifting and rolling masses to enhance the ability to control the AUV's pitch and roll, in place of (or in combination with) traditional control surfaces. The presence of these internal actuators leads to the simulation of the vehicle dynamics becoming a significantly more convoluted process when compared to modelling traditional AUVs, resulting from the multibody dynamics nature of the vehicles. Currently available simulators implement standard 6 DOF rigid-body models, which lack the modularity to be readily extended to suit the multibody dynamics characteristics of this new generation of AUVs.

Second, developing FTC capability was shown to be of paramount importance for AUV missions, owing to the unstructured nature of the operating environment within which the vehicles operate. One of the limitations reported in relevant research is the difficulty encountered when devising a unique control law for multiple fault modes, with no clear method that stood out as offering a conventional solution to the problem. Relevant research in the field of FTC conveys the message that efforts to find pFTC

solutions can not be justified both due to a lack of available methods and due to the control solution possibly being over-conservative.

Recent control methods based on ML hold the potential to provide innovative perspectives to unresolved control problems, such as the design of pFTC laws. However, ML-based methods conventionally suffer from a lack of stability guarantees that make their application unsuited to safety-critical control applications, such as when controlling AUVs. Therefore, a new investigation into pFTC methods by applying recent results in ML without compromising on the classical formal stability theories shall be conducted. Should an effective pFTC method be devised, the potential to reduce the vehicle design complexity and cost by eliminating the need for detection algorithms and monitoring sensors would be substantial.

3.2 Research vision

Based on the identified research gaps, this work aims first to develop a unified framework to simulate diverse underwater vehicles, ranging from traditional AUVs to UGs and hybrid vehicles. Differing from the traditional causal simulation approach, this work intends to explore the acausal simulation paradigm, employing class-oriented modelling to enhance simulation design modularity along with the capability to simulate the dynamics of a wider range of more complex underwater vehicles.

Next, in line with encouraging recent works in the field of pFTC, this thesis challenges the common view that pFTC has limited applicability due to the difficulty encountered in the synthesis of the control law. Recent advancements in ML-based control methods, together with the increased availability of open-source software resources, have the potential to shed new light on the field of pFTC engineering.

3.3 Research questions

Three research questions were derived from the identified research gaps, these in turn representing the three focal points that would need to be addressed to advance the public knowledge in the field of simulation and control of a diverse range of autonomous underwater vehicles with ML-based methods.

RQ1 Are multibody object-oriented simulators suitable for simulating the dynamics

of autonomous underwater vehicles?

RQ2 Is the Neural Lyapunov Control method an effective approach to design stable control laws for nonlinear systems?

RQ3 How can the Neural Lyapunov Control method be extended to a passive fault-tolerant control approach to ensure closed-loop stability for platforms affected by actuator faults?

3.4 Research approach

To simulate the latest generation of AUV concepts based on a class-oriented modelling paradigm, a comprehensive library encompassing the dynamics of the hull and the internal moving components, together with hydrostatic and hydrodynamic effects will be developed. The full range of actuators conventionally mounted on AUVs, such as control surfaces, thrusters and VBDs, will be included for use in the library. The simulation of any AUV comprising the use of any combination of such actuators will be made possible by assembling the sub-components of which the vehicle is comprised.

Experimental evaluation of the simulator will follow in two steps of increasing complexity. First, a verification of the associated equations of motion will be carried out, namely a comparison of the simulated results against theoretical analytical results. Next, the validation of the simulation output will be carried out against deployment data, providing an overview with regard to the accuracy of the developed simulation tool.

In the second part of this research, focus will be shifted to the control methods. Specifically, the NLC method will be investigated to assess its capability to systematically design control laws in an automated manner. The NLC mixes sound theoretical results rooted in nonlinear control theory with the computational attractiveness of ML-based approach and stability guarantees certified through formal verification, as illustrated in Fig. 3.1.

As the NLC was initially designed to extend the closed-loop performance of a state-feedback control law, target interest will be directed to analyse the capability of the method to automatically design control laws. The NLC method will be challenged

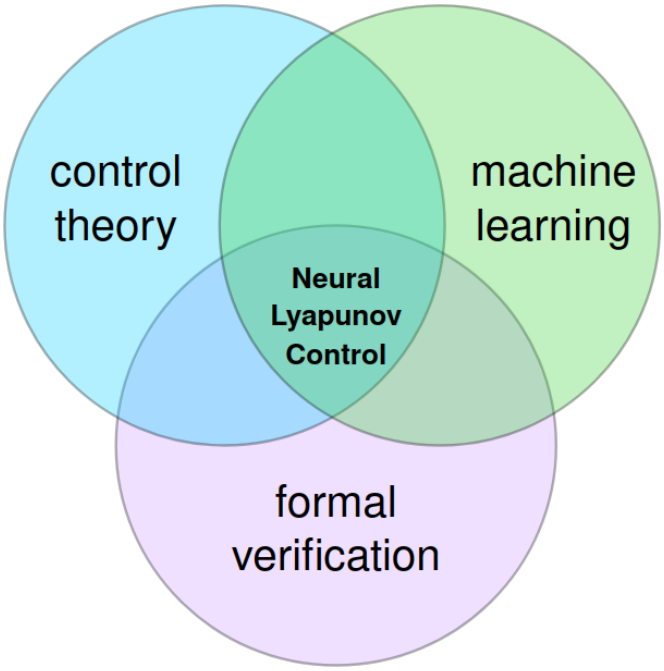


Figure 3.1: Venn diagram of the control method selected for the present research (adapted from [18]).

to synthesise stabilising control laws for a set of benchmark dynamics. Two outcomes will follow. If the NLC is found to already be suitable for the intended purpose in its current form, the research will proceed by extending the NLC as a pFTC method. Otherwise, tailored modifications will be trialled to render the NLC suitable to the automatic synthesis of control laws.

The third part of this research will focus on enhancing the NLC method with fault-tolerant capability. Once the NLC has the proven capability to synthesise control laws, the enhanced method will be challenged to simultaneously stabilise a series of system dynamics associated to the nominal and to the fault modes. The fault-tolerant problem is thus formulated as the feasibility of the design of a unique control law to stabilise a prescribed set of operational modes problem. Naturally, the fault-tolerant-capable NLC will inherit the same properties of the unmodified NLC, namely bringing together the fields of fault-tolerant control with ML and formal verification, as shown in Fig. 3.2.

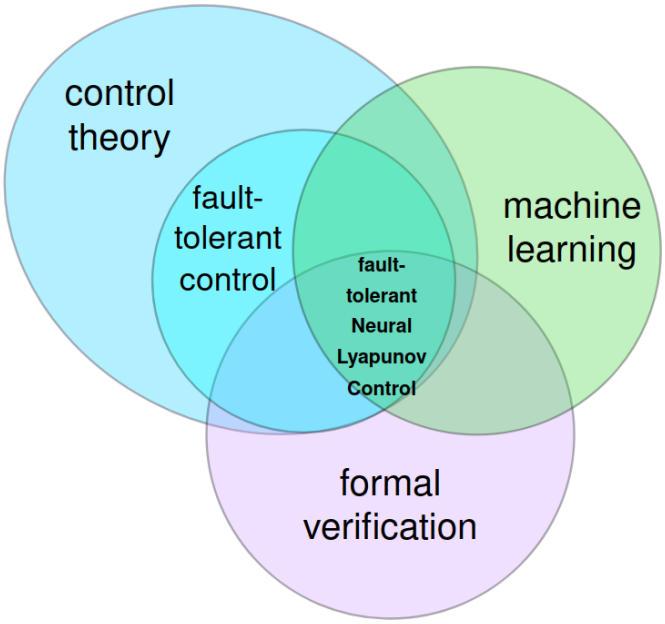


Figure 3.2: Venn diagram of fault-tolerant control methodology.

To conclude, the complete process entailing the design of a fault-tolerant-capable NLC law applied to a AUV operating under fault conditions and verified with the developed simulation tool will be designed and verified.

The overall research approach that will be followed in this thesis is illustrated in Fig. 3.3.

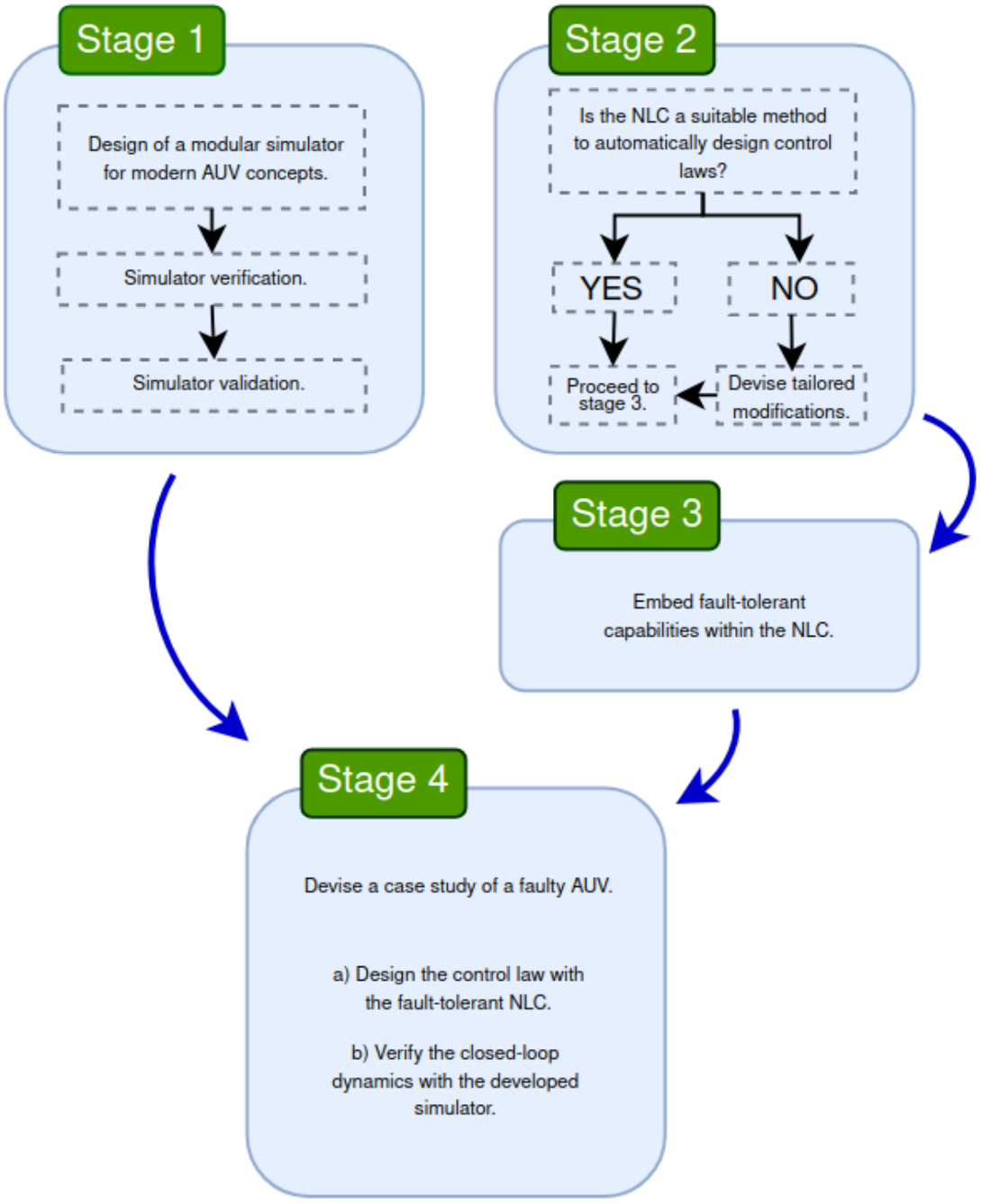


Figure 3.3: Summary of the research approach envisioned for this thesis.

Chapter 4

OpenMAUVe: an open-source Modelica simulator for Autonomous Underwater Vehicles and Gliders

4.1 Introduction

This chapter outlines the design and verification of a generic simulator for a wide range of AUVs, including traditional AUVs, UGs and hybrid AUVs. First, the modelling theory used to describe the fluid dynamics interaction of the vehicles with the surrounding environment is reported. Next, a modular Modelica-based simulator architecture is designed, accommodating the implementation of different vehicles. Then, the simulator is verified through comparison of the obtained results with an analytical dynamical model. Following, the simulator is (quasi-) validated through comparison with another AUV simulator. To conclude, instructions on how to employ the simulator are provided and implementation examples are showcased[1].

The outcome of this chapter provides answers to RQ1: **are multibody object-oriented simulators suitable for simulating the dynamics of autonomous underwater vehicles?**

[1]The complete software framework associated with the simulator described in this chapter is scheduled to be released in a dedicated publication, and will be released open-source at: https://github.com/grande-dev/OpenMAUVe.

4.2 Overall simulator design

Simulator design for AUVs is most often carried out by employing a MATLAB/Simulink framework [160, 161, 162, 163, 164]. This choice of design tools allows the designer to employ libraries inherited from other applications, such as those stemming from the aerospace community. One such example is the *6-DOF Motion Platform*[2], designed within the Simulink *Aerospace Blockset*.

The MATLAB/Simulink framework constitutes an instance of *causal* simulators: individual components or *blocks*, each with their own input and output signals are connected in series, where each block input is determined by the output of the previous block. In such a framework, the dynamic response is then simulated through time-based propagation of an initial input signal and/or conditions, in what is referred to as *signal flow mode* [165].

Employing a methodology that differs radically from the causal simulator described above, the *acausal* (or non-causal) paradigm involves designing component models and connecting them via ports through which data may be exchanged [165]. A component model encapsulates the set of describing equations with the interfaces representing relevant physical information (e.g. the model of a spring might be designed with an interface representing an external applied force, or a displacement) [166].

When modelling complex mechanical structures, acausal simulators offer a high degree of design flexibility when compared to the causal counterpart. An articulated mechanical structure is assembled within the simulator after defining the dynamic equations of the constituent sub-components, e.g. several 6 DOFs rigid-bodies can be connected by physical interfaces and re-arranged as needed. This modelling paradigm greatly simplifies the design process of multibody dynamics, which are error prone, less scalable, and need to be implemented in an ad hoc simulator. In the context of simulating a wide range of AUVs encompassing several combinations of actuators, opting for acausal simulators allows for a more straightforward re-design of the vehicle, for instance by adding or removing target components.

When simulating AUVs, the equations of motion encompassing both the rigid-

[2] https://uk.mathworks.com/help/simulink/slref/six-degrees-of-freedom-6-dof-motion-platform.html.

body dynamics and the external forces and moments, such as hydrostatics and hydrodynamics, are to be implemented. Different AUVs such as traditional AUVs and UGs share similarities in the modelling of external forces and moments owing to the common nature of the hydrodynamic interaction of the hull with the surrounding fluid. However, internal actuators involving the displacement of moving masses or determining a time-varying system mass, lead to differences in the equations of motion.

Hence different actuators lead to equations of motions that are described by systems of Ordinary Differential Equations (ODEs) of different *dimensions*.

Definition 1 (Dynamic model dimension [92]) *Given a dynamic model defined by a system of ODEs, the model dimension or, more simply, the dimension, defines the number of dependent variables.*

Traditional AUV (and ROV) are conventionally modelled by means of 6 DOFs rigid-body dynamics, resulting in 6-dimensional models [147]. In contrast, the modelling of UGs and hybrid AUVs involves additional DOFs associated to the internal moving components, resulting in higher dimensional models. Including these additional dimensional considerations can result in models of up to 9 dimensions and above [20].

In the context of employing acausal simulators in underwater vehicles applications, high-dimensional AUVs models can be simulated by assembling the constituent sub-components. Each physical component, such as the hull or a movable mass is described as an element with a mass and an inertia, defined by a set of equations of motion.

Once the individual sub-components are designed, connecting them by means of mechanical constraints results in a leaner and less complex design solution than implementing ad hoc the complete set of equations of motions, as required by the causal paradigm. This design choice renders it possible to simulate and test several vehicle architectures with different combinations of current available actuators or even including new one actuators with less design effort and complexity.

There are two main alternative languages and tools that currently represent the

current state-of-the-art for acausal modelling: Simscape and Modelica. Simscape[3] is an add-on *tool* to the MATLAB/Simulink environment, offering several libraries for different physical domains, such as SimElectronics, SimHydraulics and SimMechanics. Modelica[4], first developed in 1997 from the Modelica Association [167], with its initial theoretical foundation proposed in 1978 [168], is an open-source object-oriented modelling *language*. Similar to Simscape, it offers libraries that may be used to model mechanical, electrical, electronic, hydraulic, thermal, and control elements.

When compared to Simscape (owned by Mathworks), Modelica has open-source fully customisable packages. A growing pool of academic and industrial partners share new results at the annual or biennial *International Modelica Conference*[5].

Different Integrated Development Environments (IDEs) for Modelica are available including, among others, Dymola and OpenModelica. Dymola[6] (DYnamic MOdeling LAboratory), developed by Dassalut Systèmes, represents a widely used commercially available option with a user-friendly interface encompassing diagram and icon layers [169].

OpenModelica[7] is currently deemed the most complete open-source development environment [170]. Although not currently offering the same level of interface sophistication of Dymola, OpenModelica users are consistently developing additional capabilities, as shown in the recent advancements in the *2023 Modelica Association Project FMI* [171] and corresponding developed tools[8]. As an example, these tools allow the user to export the designed models via Functional Mock-up Unit, an extension based on the Functional Mock-up Interface, which is an open standard defining a structure and interface for exchanging dynamic simulation models using a combination of binary files, a *XML* description, and *C* code.

Given the recent advancements in the Modelica language and associated tools, it is possible to employ the Modelica language for simulating complex equations of motions and interfacing to other popular software designed for different tailored pur-

[3] https://uk.mathworks.com/products/simscape.html
[4] https://www.modelica.org/
[5] https://modelica.org/events/
[6] https://www.3ds.com/products-services/catia/products/dymola/
[7] https://www.openmodelica.org/
[8] https://github.com/modelica/Reference-FMUs

poses. For instance, it is possible to complete the dynamics design process in Dymola and then import the model in Simulink to exploit the control toolboxes designed by Mathworks [172]. Moreover, Modelica offers standard communication interfaces to interact with popular programming languages such as *Python* or with middleware software applications such as Robot Operating System (ROS). This makes it an optimal open-source choice for complex academic and industrial applications that can exploit the best functionalities of each software tool, such as designing the system dynamics in Modelica, the control laws in *Python* or *C++*, and the testing of closed-loop performances in ROS.

Due to its unique simulation architecture, Modelica requires tailored numerical methods to simulate the resulting dynamics. In actuality, connecting physical elements each with an associated dynamics, might give rise to systems of DAEs. On top of the dynamics of the individual objects, explicitly described during the component model definition and ususally described by ODEs, the physical constraints add algebraic non-differentiated equations. As an example, assuming there is a requirement to design a class denoted as *RigidBodyPair*, representing the dynamics of a two generic rigid-body joint together. Two single RigidBody objects can be instantiated, and connected to one another by means of a prismatic joint. The physical constraint imposed by the joint is represented at solver level as an algebraic equation that is automatically added to the set of ODEs associated with the rigid-body dynamics defined during the design of the RigidBodyPlane model. This initial step of model assembly through the creation of a system of DAEs is defined as *Flattening* [167]. Next, equation manipulation (such as automatic simplification or rearrangement) is performed to attempt solving subsystems of equations more efficiently in what is defined as *Causalisation*. Finally, the *Time integration* steps are performed to simulate the required dynamics.

Owing to the different physical domains that can be simulated, Modelica models can be composed such that both fast and slow dynamics need to be simulated, giving origin to *stiff* problems. Fast dynamics force the solvers to take unreasonably small steps, in turn leading to significantly degraded simulation performance when combined to slow dynamics. To this end, Modelica offers acausal resolution algorithms capable

of increasing the simulation execution rate of complex dynamical systems, becoming a key element when carrying out simulations replicating, for instance, long deployment missions, where each one can last up to several weeks or months in actuality.

Despite the noted benefits, OpenModelica does present some deficiencies. First, the graphical interface is not comparable with those provided by the commercially available packages, namely Dymola or Simscape. Second, the resolution of bugs or coding inefficiencies is left to the Modelica community, which often results in there being longer delay for such issues being resolved. Despite the deficiencies, OpenModelica is currently deemed to be the best option within the framework of simulating complex multibody models employing open-source tools.

Modelica features a range of free mechanical, electrical, magnetic, thermal, fluid and control systems models, collected within the Modelica Standard Library (MSL). A growing number of free and commercial libraries developed by the Modelica association are also accessible within the OpenModelica IDE. Several libraries were developed based on the basic components offered from within the MSL to model different vehicles, among which ground vehicles [173], reusable launch vehicles [172], rockets landing via a parachute [27] and rigid and flexible aircrafts [174]. Despite the availability of a wide range of free libraries, there is currently no library specifically dedicated to the modelling and simulation of underwater vehicles.

Several components constituting the AUVs can already be found from within the MSL. For instance, mechanical components available within the MSL can be used to instantiate the rigid-bodies, which constitute the building blocks for the hull and internal moving masses of the UGs. However, these components do not include key hydrodynamic factors such as dynamic change of mass or volume of the vehicle (driven by the VBD), as well as the ability to account for added mass effects. Additionally, forces specific to the underwater environment such as hydrodynamic effects and conditions such as buoyancy are not represented among the pre-defined blocks and need dedicated development. The lack of a unified framework tailored to underwater factors led to the necessity to develop a new library to facilitate ready prototyping and testing of control laws for AUVs and UGs. The output of this chapter will therefore be a new set of tools

to simulate a broad range of underwater vehicles encompassing effects and different actuators.

Following the library development, verification of the simulator will be carried out in two phases of increasing complexity. First, a comparison of the devised simulator results against theoretical results will be performed. This is to verify whether the vertical (two-dimensional (2D)) dynamics of an AUV simulated in OpenModelica replicates the theoretical (i.e. analytical) model to within an acceptable margin of accuracy. Second, data from a second simulation architecture implementing an AUV performing more advanced vertical and spiralling (three-dimensional (3D)) manoeuvres will be used for purposes of empirical comparison.

4.3 Autonomous Underwater Vehicles modelling

AUVs dynamic modelling has followed different approaches based on the type of onboard actuators. Modern AUVs leveraging internal shifting and rolling masses to enhance the ability to control the AUV's pitch and roll dynamics prompted to extend the traditional modelling approach characterised by the description of the vehicle as a single rigid-body.

From a mechanical perspective, a traditional AUV has conventionally been represented as a single rigid-body, owing to the vehicle being actuated by means of thrusters and steering surfaces [175]. The inertia of the thrusters (which at times might be commanded to rotate, as in the case of azimuth thrusters) and of the control surfaces is often considered negligible when compared to the inertia of the vehicle hull. This modelling choice allows to consider the vehicle as having a unique lumped mass and a constant inertia tensor, simplifying the design of relevant dynamic models. Conversely, UGs have been represented as multibody systems actuated with a combination of VBD and internal moving masses. This modelling choice derives from the moving masses generating significant dynamics on the hull, when set into motion.

According to the Newton-Euler formalism[9], dynamic models can be constructed

[9] A preliminary version of *Newtonian* formalism was first published by Isaac Newton in 1687 for point-particle systems and later extended to rigid-bodies by Leonhard Euler in 1736. In associated literature, this modelling approach is commonly referred to as *Newton-Euler* formalism [176].

based on the principle of conservation of linear and angular momentum. This approach was imported in 1991 into the marine industry in the form of the robot-like vectorial model description [177], and became ubiquitous in modelling, guidance and control applications. Selecting a vehicle of mass m and (constant) inertia tensor $\boldsymbol{I_0}$, and defining an arbitrary origin $\{O\}$ (typically attached to the vehicle body), the Newton-Euler formalism applied to AUVs yields the following equations of motion $\in \mathbb{R}^6$:

$$\begin{cases} m(\dot{\boldsymbol{v}}_\mathbf{0} + \boldsymbol{\omega}_\mathbf{0} \times \boldsymbol{v}_\mathbf{0} + \dot{\boldsymbol{\omega}}_\mathbf{0} \times \boldsymbol{r_g} + \boldsymbol{\omega}_\mathbf{0} \times (\boldsymbol{\omega}_\mathbf{0} \times \boldsymbol{r_g})) = \boldsymbol{f_0} & \text{(4.1a)} \\ m\boldsymbol{r_g} \times \dot{\boldsymbol{v}}_\mathbf{0} + m\boldsymbol{r_g} \times (\boldsymbol{\omega}_\mathbf{0} \times \boldsymbol{v}_\mathbf{0}) + \boldsymbol{I_0}\dot{\boldsymbol{\omega}}_\mathbf{0} + \boldsymbol{\omega}_\mathbf{0} \times (\boldsymbol{I_0}\boldsymbol{\omega}_\mathbf{0}) = \boldsymbol{m_0} & \text{(4.1b)} \end{cases}$$

where $\boldsymbol{v}_\mathbf{0} \in \mathbb{R}^3$ is the velocity of $\{O\}$, $\boldsymbol{\omega}_\mathbf{0} \in \mathbb{R}^3$ the angular velocity about $\{O\}$ and $\boldsymbol{r_g} \in \mathbb{R}^3$ the distance from $\{O\}$ to the centre of gravity. Additionally, $\boldsymbol{f_0} \in \mathbb{R}^3$ represents the external forces applied to $\{O\}$ and $\boldsymbol{m_0} \in \mathbb{R}^3$ the moment of the external forces and the external torques about $\{O\}$. Finally, $\dot{\boldsymbol{v}}_\mathbf{0} \in \mathbb{R}^3$ and $\dot{\boldsymbol{\omega}}_\mathbf{0} \in \mathbb{R}^3$ denote the time derivatives of $\boldsymbol{v}_\mathbf{0}$ and $\boldsymbol{\omega}_\mathbf{0}$, respectively.

Assuming that an AUV is considered composed of a number j of internal bodies, such as the hull and moving masses, one modelling approach involves using j 6 DOFs models in parallel (from Eq. (4.1)) linked by the exchange of the reaction forces and moments at the interface by means of $\boldsymbol{f_0}$ and $\boldsymbol{m_0}$. As is apparent, the derivation of such a mathematical model is highly convoluted. Second, this approach requires significant manipulation every time an additional moving mass is to be considered, as when a different mechanical design is developed.

To simplify the modelling task and the corresponding implementation in a simulator, computer-aided tools such as Modelica were developed. The Modelica compiler automatically generates a system of DAEs, composed of one derivation of Eq. (4.1) for each rigid-body (a set of ODEs), combined with algebraic equations associated to the exchange of momenta at the physical constraints (e.g. prismatic joints, rotational links etc.). Owing to the nature of the proposed approach, thorough comparison against theoretical results is rendered necessary. The focus of this section is the investigation of external forces and moments experienced by a vehicle's hull (namely $\boldsymbol{f_0}$ and $\boldsymbol{m_0}$), to later verify whether a Modelica-based approach is a viable simulation option for this

class of complex multibody vehicles.

Dynamic models of underwater vehicles are conventionally developed employing three reference frames: the inertial, the body and the flow frames [178, 20, 179]. This modelling method allows the designer to express hydrostatic and hydrodynamic forces (and moments) acting on the vehicle body in the most convenient reference and then subsequently rotated onto the body-fixed frame.

The *inertial frame* is defined as a non-accelerating frame, usually set to be Earth-fixed [175]. As a first approximation, a North-East-Down (NED) frame of origin $\{O_i\}$ can be selected as Earth-fixed and non-rotational, or *inertial*, such that Newton's laws of motion apply. This approximation is generally referred to as *flat Earth navigation*. Care must be taken when missions of a size comparable to the Earth's radius are being simulated, or when several latitudes and longitudes are spanned during the operations. In such cases, other frames that can account for the Earth's angular rate of rotation such, as the Earth-centred Earth-fixed (ECEF), need to be adopted.

Next, a *body-fixed frame* of origin $\{O_b\}$ that moves with the vehicle is located at the Centre of buoyancy (COB) of the hull. A standard procedure involves assuming the COB to be geometrically fixed at the centroid of the hull volume. The hull volume expands and contracts based on the operating depth, but the change in deformation is assumed to be uniform, i.e. the hull does not experience different deformation rates on different axes due to pressure. As the AUVs are typically designed with an ellipsoidal body, the COB is set to coincide with the centroid of the ellipsoid. For non-ideal ellipsoidal bodies, e.g. when the vehicle hull is composed of several sections with differing geometries, further mechanical design effort is required to preserve the vehicle shape under compression. An example of the need for this consideration is the Slocum UG, which encompasses five sections of elliptical shapes and truncated cones and where angles are chosen to provide comparable compressibility as the UG dives [180, 181]. For modelling convenience, the $\{O_b\}$ axes of the vehicle are usually selected to align to the principal axes of inertia of the vehicle. This means that there might be a translation from $\{O_b\}$ to the principal axes of inertia of the vehicle, but no change of relative orientation. Therefore, the x_b-axis of the body frame is aligned with the longitudinal

axis of the vehicle and points towards the nose end, the z_b-axis points downward and the y_b-axis completes the right-hand tern pointing starboard.

Since the hydrodynamic forces and moments are traditionally expressed as function of the relative velocity of the vehicle with respect to the surrounding fluid, a third frame defined as *flow frame*, is introduced. The origin of the flow frame is coincident with the origin of the body frame ($\{O_f\} \equiv \{O_b\}$), and its orientation is obtained from the body frame by performing two sequential rotations of the angles α and β (detailed later in this section). The flow frame is specifically convenient when modelling hydrodynamic effects, such as lift and viscous damping. The frame definitions and the selected convention of the axes are shown in Fig. 4.1.

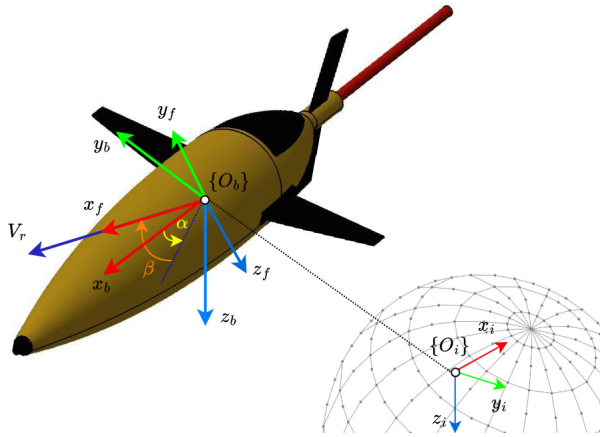


Figure 4.1: Selected reference frames in use for AUV modelling (Seaglider UG - ogive model), © 2022 IEEE [19].

In accordance with the Society of Naval Architects and Marine Engineers (SNAME) nomenclature (1950) [182], $\boldsymbol{v_0} = (u, v, w)$ denote the components of the body-fixed linear velocity with respect to $\{O_i\}$ expressed in $\{O_b\}$. When a marine vehicle is exposed to ocean currents of components $\boldsymbol{v_c} = (u_c, v_c, w_c)$ as measured with respect to $\{O_b\}$, the *relative velocity* vector along the three axes can be expressed in the body-fixed frame as [175]:

$$u_r = u - u_c \tag{4.2a}$$

$$v_r = v - v_c \tag{4.2b}$$

$$w_r = w - w_c. \tag{4.2c}$$

The magnitude of the flow speed vector, denoting the relative speed of the vehicle with respect to the surrounding fluid, is defined as [175]:

$$V_r = \sqrt{u_r^2 + v_r^2 + w_r^2}. \tag{4.3}$$

Additionally, the hydrodynamic angles α and β, defined as angle of attack and the sideslip angle, respectively, are computed as:

$$\alpha = \tan^{-1}\left(\frac{w_r}{u_r}\right) \tag{4.4a}$$

$$\beta = \sin^{-1}\left(\frac{v_r}{V_r}\right). \tag{4.4b}$$

Following the introduction of the reference frames and their associated convention, the modelling focuses on the definition of the elements composing the vehicle's dynamics. AUV dynamics encompass several masses associated with the internal elements. The overall vehicle *stationary mass* (m_s) is referred to as:

$$m_s = m_h + m_w + m_b \tag{4.5}$$

where m_h denotes the hull mass (uniformly distributed throughout the vehicle), m_w a fixed point mass that captures non-homogeneous mass distribution elements and m_b the variable ballast mass. Fig. 4.2 illustrates the definition of the masses and their relative locations with respect to the $\{O_b\}$. m_b is displaced from the $\{O_b\}$ by the distance $\boldsymbol{r_b}$ and m_w by the distance $\boldsymbol{r_w}$, with $(\boldsymbol{r_b}, \boldsymbol{r_w}) \in \mathbb{R}^3$. The mass m_w is used during the trimming of the vehicle to balance the pitching and rolling moments that results from uneven mass distribution, that may be due to, for example, internal components or manufacturing imperfections. The overall vehicle mass is defined as:

$$m_v = m_s + m_p \tag{4.6}$$

where m_p denotes a movable mass, the position of which, with respect to to the $\{O_b\}$,

is time-variant and denoted by $\boldsymbol{r_p}(t)$.

The mass of the fluid displaced by an AUV is typically referred to as m, and the *net buoyancy* as $m_0 = m_v - m$. According to this convention, the vehicle is negatively buoyant when m_0 is positive and positively buoyant when $m_0 < 0$.

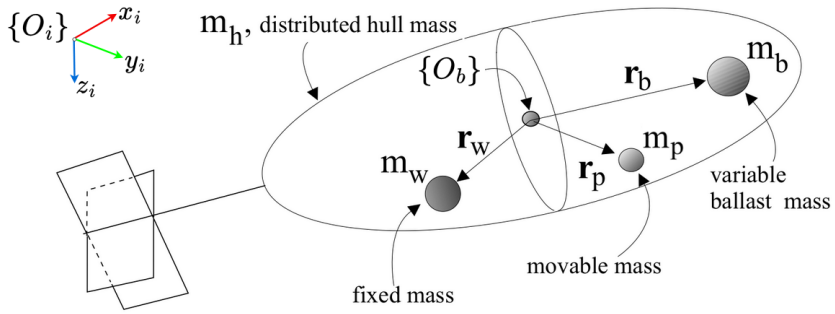


Figure 4.2: AUV masses definition (adapted from [20]).

4.3.1 Hydrodynamics

Forces and moments acting on the vehicle will now be discussed. Hydrodynamic forces and moments are highly nonlinear functions of the speed and attitude of a rigid object's interaction with a surrounding fluid [183]. While *drag* is the term employed in aerodynamic applications, *viscous damping* or *drag* are interchangeable notations in marine modelling and are used as synonyms in this thesis [147, 20]. Viscous damping encompasses several phenomena as *potential damping* (associated to the oscillation of a body subjected to wave excitation), *skin friction damping* (linked to shear forces from water flowing alongside the hull) and *form damping* (due to pressure differences between the bow and stern of the object) [184].

As it is generally difficult to isolate the effect of each component on the overall damping, viscous effects are usually modelled as the sum of a linear and a quadratic term [175]. The linear factor mostly captures potential damping and skin friction, while the quadratic term encapsulates the form damping and the effect of vortex shredding at sharp edges. Conversely, as an object moves in a fluid, the separation of the flow induces an unbalanced pressure drop that in turn gives rise to forces that act perpendicularly to the flow speed vector, denoted as lift. Lift forces are satisfactorily captured by quadratic terms with respect to the object velocity [148]. Viscous linear and nonlinear

terms are calculated with respect to the flow frame: this modelling is based on the size of the wetted area of the object (or its cross sectional area), the density of the fluid and the speed relative to the surrounding fluid. All these factors can be retrieved through analysis of the vehicle's geometry and through sensoring.

A typical modelling choice of the hydrodynamics of submerged AUVs, as reported in [20, 179], is represented by:

$$\boldsymbol{F}_{hd}^{f} = \begin{bmatrix} -D & SF & -L \end{bmatrix}^{T} \tag{4.7a}$$

$$\boldsymbol{M}_{hd}^{f} = \begin{bmatrix} T_{DL1} & T_{DL2} & T_{DL3} \end{bmatrix}^{T} \tag{4.7b}$$

with D denoting the drag, SF the side-force (sometimes referred to as crossflow drag), L the lift, and T_{DLi} the viscous moment around the i-th axis of the flow frame. Hydrodynamic effects are highly nonlinear functions of the vehicle's geometric properties, speed and relative motion with respect to the surrounding fluid [175]. Models encompassing the use of linear terms only are employed for slow moving vehicles, (i.e. less than 2 knots), while models that utilise quadratic terms are used for vehicles that move at higher speeds, where the quadratic terms dominate. There are occasions in maritime vehicles modelling when both linear and quadratic terms are employed in combination [185]. In the specific case of underwater vehicles where the hydrodynamic effects due to wave excitation can be neglected, models developed with quadratic terms only represent the de-facto modelling choice [184, 20, 148].

Hydrodynamic models as quadratic function of the flowspeed can be described, for the forces, as:

$$\begin{aligned} D &= -\frac{1}{2}\rho V_r^2 A C_D(\alpha, \beta, \delta_{cs}, Re) \\ SF &= \frac{1}{2}\rho V_r^2 A C_{SF}(\alpha, \beta, \delta_{cs}, Re) \\ L &= -\frac{1}{2}\rho V_r^2 A C_L(\alpha, \beta, \delta_{cs}, Re) \end{aligned} \tag{4.8}$$

and, for the moments, as:

$$\left.\begin{aligned} T_{DL1} &= \tfrac{1}{2}\rho V_r^2 A C_{DL1}(\alpha,\beta,\delta_{cs},Re) \\ T_{DL2} &= \tfrac{1}{2}\rho V_r^2 A C_{DL2}(\alpha,\beta,\delta_{cs},Re) \\ T_{DL3} &= \tfrac{1}{2}\rho V_r^2 A C_{DL3}(\alpha,\beta,\delta_{cs},Re) \end{aligned}\right\} + \boldsymbol{K}_{\boldsymbol{\omega}^1}\boldsymbol{\omega} + \boldsymbol{\omega}\boldsymbol{K}_{\boldsymbol{\omega}^2}\boldsymbol{\omega} \tag{4.9}$$

where δ_{cs} denotes the effect of the control surfaces (e.g. rudders and stern-planes), A the vehicle cross sectional area, Re the Reynolds number of the flow on the vehicle, and $C_D, C_{SF}, C_L, C_{DL1}, C_{DL2}, C_{DL3}$ the hydrodynamic coefficients, typically obtained through Computational Fluid Dynamics modelling, $\boldsymbol{K}_{\boldsymbol{\omega}^1}, \boldsymbol{K}_{\boldsymbol{\omega}^2}$ the linear and quadratic damping terms and $\boldsymbol{\omega}$ the vector of the vehicle angular velocities.

While Eq. (4.8) and Eq. (4.9) capture high degrees of nonlinear hydrodynamic effects, simplified models are often sufficient in dynamic modelling applications. One such simplification choice entails considering the *quasi-steady state* hydrodynamic form, as:

$$D \approx (K_{D0} + K_D \alpha^2) V_r^2 \tag{4.10a}$$

$$SF \approx K_\beta \beta V_r^2 \tag{4.10b}$$

$$L \approx (K_{L0} + K_L \alpha) V_r^2 \tag{4.10c}$$

$$T_{DL1} \approx (K_{MR}\beta + K_p p) V_r^2 \tag{4.10d}$$

$$T_{DL2} \approx (K_{M0} + K_M \alpha + K_q q) V_r^2 \tag{4.10e}$$

$$T_{DL3} \approx (K_{MY}\beta + K_r r) V_r^2 \tag{4.10f}$$

with the hydrodynamic coefficients K_i being exclusively dependent on the geometry of the vehicle, and, different from the coefficients C_i of Eq. (4.8) and Eq. (4.9), being constant.

Hydrodynamic effects can be modelled as lumped forces acting in the Centre of pressure (COP) of the vehicle. As the COP usually does not coincide with the centre of the body frame $\{O_b\}$, the hydrodynamic forces give rise to a moment around the origin of the body frame, captured by the T_{DL}-terms of Eq. (4.9) and Eq. (4.10). The

hydrodynamic forces can be defined as $\boldsymbol{F}_{hd}^{f}$ and considered acting on the centre of the flow frame, with the generated moments denoted as $\boldsymbol{M}_{hd}^{f}$ and where the superscript f explicitly denotes that the forces and moments are calculated with respect to the flow frame.

As the equations of motions are conventionally expressed with respect to the body-fixed frame, $\boldsymbol{F}_{hd}^{f}$ and $\boldsymbol{M}_{hd}^{f}$ can be rotated into the body-fixed frame by means of the following rotation matrix [20]:

$$\boldsymbol{R}_{f}^{b} = \begin{bmatrix} \cos\alpha\cos\beta & -\cos\alpha\sin\beta & -\sin\alpha \\ \sin\beta & \cos\beta & 0 \\ \sin\alpha\cos\beta & -\sin\alpha\sin\beta & \cos\alpha \end{bmatrix}. \tag{4.11}$$

Next, since marine applications involve the operation of vehicles that are fully immersed in a fluid of density comparable to that of the object, hydrodynamic added mass effects need to be considered and taken into account [20]. Added mass effects capture the exchange of inertia between an object in motion with the surrounding fluid and are dependent on the geometry of the vehicle. Hydrodynamic added mass can be treated as a virtual mass that is added to the object when accelerating in a fluid [175]. Further added mass effects are linked to the frequency of the surrounding fluid due to the currents and surface wave effects. For surface applications or for submerged vehicles operating at shallow depth, potential added mass effects need to be modelled. However, the de-facto standard model for underwater marine vehicles, initially proposed in the early 1990s [147], relies on frequency-independent added mass coefficients, assuming that the AUVs operate at sufficient depth such that wave excitation is negligible. The added mass coefficients can be collected in a hydrodynamic added inertia matrix, referred to as $\boldsymbol{M}_{\boldsymbol{A}} \in \mathbb{R}^{6\times6}$, that is positive definite and is defined according to the SNAME notation:

$$\boldsymbol{M_A} = -\begin{bmatrix} X_{\dot{u}} & X_{\dot{v}} & X_{\dot{w}} & X_{\dot{p}} & X_{\dot{q}} & X_{\dot{r}} \\ Y_{\dot{u}} & Y_{\dot{v}} & Y_{\dot{w}} & Y_{\dot{p}} & Y_{\dot{q}} & Y_{\dot{r}} \\ Z_{\dot{u}} & Z_{\dot{v}} & Z_{\dot{w}} & Z_{\dot{p}} & Z_{\dot{q}} & Z_{\dot{r}} \\ K_{\dot{u}} & K_{\dot{v}} & K_{\dot{w}} & K_{\dot{p}} & K_{\dot{q}} & K_{\dot{r}} \\ M_{\dot{u}} & M_{\dot{v}} & M_{\dot{w}} & M_{\dot{p}} & M_{\dot{q}} & M_{\dot{r}} \\ N_{\dot{u}} & N_{\dot{v}} & N_{\dot{w}} & N_{\dot{p}} & N_{\dot{q}} & N_{\dot{r}} \end{bmatrix} \tag{4.12}$$

where the hydrodynamic derivatives are employed, i.e. constant parameters. The hydrodynamic derivatives notation is such that, for example, the hydrodynamic added mass Z along the z_b-axis due to an acceleration $\dot{u}$ along the x_b-axis is defined as:

$$Z = -Z_{\dot{u}}\dot{u}. \tag{4.13}$$

For underwater vehicles moving at low speed, symmetries can be exploited to reduce the number of coefficients in $\boldsymbol{M_A}$. For a vehicle with three planes of symmetry, e.g. a spherical vehicle [150], all the off-diagonal terms are identically null, namely:

$$\boldsymbol{M_A} = \text{diag}\{X_{\dot{u}}, Y_{\dot{v}}, Z_{\dot{w}}, K_{\dot{p}}, M_{\dot{q}}, N_{\dot{r}}\}. \tag{4.14}$$

In the case of an AUV, the hull is usually designed with an ellipsoidal or ogival form. Approximating the ogival form to a prolate spheroid gives two planes of symmetry, namely port-starboard and top-bottom. Even after such an approximation, the off-diagonal elements are significantly smaller than the diagonal elements, and the diagonal approximation as shown in Eq. (4.14) is found to be sufficient for the majority of applications. For such a geometry, the added mass coefficients can be computed as

reported in [186]:

$$
\begin{aligned}
e^2 &= 1 - \frac{b}{a} \\
\alpha_0 &= \frac{2(1-e^2)}{e^2}\left(\frac{1}{2}\log\left(\frac{1+e}{1-e}\right) - e\right) \\
\beta_0 &= \frac{1}{e^2} - \frac{1-e^2}{2e^2}\log\left(\frac{1+e}{1-e}\right) \\
k_1 &= \frac{\alpha_0}{2-\alpha_0} \\
k_2 &= \frac{\beta_0}{2-\beta_0} \\
k' &= \frac{e^4(\beta_0 - \alpha_0)}{(2-e^2)[2e^2 - (2-e^2)(\beta_0 - \alpha_0)]} \\
X_{\dot{u}} &= -k_1 m \\
Y_{\dot{v}} &= Z_{\dot{w}} = -k_2 m \\
K_{\dot{p}} &= 0 \\
M_{\dot{q}} &= N_{\dot{r}} = -k' I_{yy}
\end{aligned}
\tag{4.15}
$$

where a and b represent the minor semi-axes, c the major semi-axis. In the case of AUVs, c is aligned with the x_b-axis, a with y_b-axis and b with z_b-axis, where the values of a and b in cylindrical shaped hulls. An open-source utility to compute the added mass coefficients of a prolate spheroid has been made available in the repository[10]. More advanced models that cater for additional terms to factor in the added mass and added inertia caused by the fins, are available [148]. For vehicles with sensors and actuators that protrude extensively the hull surface, as in the case of the UX-1 AUV, added mass coefficient estimation can be improved by considering the protrusions as fins above the vehicle centre line [150].

4.3.2 Hydrostatics

Marine vehicles are under the influence of *restoring* forces, given by the combined effect of gravity and buoyancy. Restoring forces are function of the vehicle mass, the volume of water displaced and the water density.

AUVs are typically designed with a choice of two different types of ballast systems. One option entails the use of a syringe-type tank that can pump water in to or

[10] https://github.com/dave-ai/Added-mass-prolate-spheroid

out of the vehicle enclosure, thereby affecting the overall net buoyancy by changing the vehicle mass. A second system exploits a balloon-type oil-filled bladder, usually located in the vehicle nose or aft section, that is inflated with oil pumped from an internal reservoir. The latter system affects the buoyancy of the vehicle by acting on the overall volume. When these ballast systems are situated in a location not at the COB, as is usually the case due to physical design constraints, an additional moment with respect to the Centre of gravity (COG) is generated. This moment can be exploited to affect the pitch dynamics, offering an additional control input. The actuator tasked with changed either the overall vehicle mass or volume is referred to as VBD.

Given the possible presence of a VBD, it is convenient to split the overall effect of the buoyancy force into two components, one that is a function of the volume of the vehicle hull (B_h), acting on the centroid of the hull volume, and the other that is generated by the VBD, acting on the centroid of the VBD (B_V). Overall, restoring forces can be computed as:

$$\boldsymbol{W} = m_v g \hat{\boldsymbol{k}}_{\boldsymbol{i}} \tag{4.16a}$$

$$\boldsymbol{B_h} = -\rho g \nabla_h \hat{\boldsymbol{k}}_{\boldsymbol{i}} \tag{4.16b}$$

$$\boldsymbol{B}_V = -\rho g \nabla_V \hat{\boldsymbol{k}}_{\boldsymbol{i}} \tag{4.16c}$$

where $\hat{\boldsymbol{k}}_{\boldsymbol{i}}$ represents the versor of the z_i-axis (inertial-fixed frame), g the Earth's gravity constant, ∇_h the volume of the hull and ∇_V the volume of the VBD.

For underwater vehicles applications, the origin of the body frame $\{O_b\}$ is conventionally set to coincide with the COB, in turn coincident with the hull volume centroid. With such modelling choice, the gravity force applies a moment with respect to the origin of the body frame [147]. Conversely, when $\{O_b\}$ is set to coincide with the COG, the buoyancy force generates a moment with respect to $\{O_b\}$. By denoting the modulus of the hydrostatic forces (namely neglecting the versor $\hat{\boldsymbol{k}}_{\boldsymbol{i}}$), and defining $W = m_v g$, $B_h = -\rho g \nabla_h$, and $B_V = -\rho g \nabla_V$, hydrostatic forces can be expressed (in the inertial-fixed frame) as:

$$\boldsymbol{f}_{\boldsymbol{g}}^{i} = [0 \quad 0 \quad W]^{T} \tag{4.17a}$$

$$\boldsymbol{f}_{\boldsymbol{h}}^{i} = [0 \quad 0 \quad B_h]^{T} \tag{4.17b}$$

$$\boldsymbol{f}_{\boldsymbol{V}}^{i} = [0 \quad 0 \quad B_V]^{T} \tag{4.17c}$$

It is worth noting that the hull volume of the vehicle is not constant as the compressibility of the hull in turn depends on the water temperature (T_w) and water pressure (p_w). More detailed volume formulations that consider $\nabla_h(T_w, p_w)$ are available [187], but will be left to future extensions of this preliminary simulator design and verification.

Next, to model the moment that the hydrostatic forces Eq. (4.17) exert with respect to the vehicle body-fixed frame, it is possible to introduce the vectors $\boldsymbol{r}_{\boldsymbol{g}}^{b} = [x_g^b, y_g^b, z_g^b]^T$, $\boldsymbol{r}_{\boldsymbol{h}}^{b} = [x_h^b, y_h^b, z_h^b]^T$ and $\boldsymbol{r}_{\boldsymbol{V}}^{b} = [x_V^b, y_V^b, z_V^b]^T$ denoting the distances from $\{O_b\}$ to the COG, to the centroid of the hull and to the centroid of the VBD, respectively. Overall, the contribution of the hydrostatic effects in the body-fixed frame results as:

$$\boldsymbol{g}^{b} = -\begin{bmatrix} \boldsymbol{f}_{\boldsymbol{g}}^{b} + \boldsymbol{f}_{\boldsymbol{h}}^{b} + \boldsymbol{f}_{\boldsymbol{V}}^{b} \\ \boldsymbol{r}_{\boldsymbol{g}}^{b} \times \boldsymbol{f}_{\boldsymbol{g}}^{b} + \boldsymbol{r}_{\boldsymbol{h}}^{b} \times \boldsymbol{f}_{\boldsymbol{h}}^{b} + \boldsymbol{r}_{\boldsymbol{V}}^{b} \times \boldsymbol{f}_{\boldsymbol{V}}^{b} \end{bmatrix} \tag{4.18}$$

To pass from Eq. (4.17) representing the hydrodynamic forces in the inertial-fixed frame, to their representation in the body-fixed frame as $\boldsymbol{f}_{\boldsymbol{g}}^{b}$, $\boldsymbol{f}_{\boldsymbol{h}}^{b}$ and $\boldsymbol{f}_{\boldsymbol{V}}^{b}$, a rotation needs to be performed. Generically, to define the orientation of the marine craft defined by $\{O_b\}$ with respect to the inertial reference frame centred in $\{O_i\}$, a set of three parameters denoted as *Euler angles* $\boldsymbol{\Theta} \in \mathbb{R}^3$ is often employed [147]. Euler angles can be employed as a means of rotating vectors (e.g. forces) from the inertial to the body-fixed frame and vice versa.

Euler angles are an instance of a *minimal representation*, i.e. a parametric representation with 3 independent factors that fully describe a frame orientation with respect to a given reference. Given an orthonormal group $SO(m)$, a number of $m(m-1)/2$ parameters are sufficient for a minimal representation [188]. Therefore, for a $SO(3)$

group, any representation entailing the use of 3 parameters is minimal. The Euler angles parameterisation entails three successive rotations around three chosen axes, provided that two successive rotations are not performed around the same axis. There are 12 possible combinations of angle sequences, denoted with three letters, e.g. *yzx* to indicate that the first rotation is performed around the original *y*-axis, to be followed by a rotation around the newly obtained *z*-axis to conclude with a rotation around the most recent *x*-axis. In aerospace and maritime applications, the most used combination is represented by the *zyx* sequence. When this rotation sequence is applied starting from a body-fixed reference frame, an orientation defined by three angles $\boldsymbol{\Theta} = [\phi, \theta, \psi]^T$, representing the *roll*, *pitch* and *yaw* angles respectively, is obtained. The final frame orientation, derived by composing the three rotations starting from an original body-fixed frame, is computed by performing pre-multiplication of the matrices of elementary rotation, being defined as:

$$\begin{aligned} \boldsymbol{R}_b^i(\boldsymbol{\Theta}) &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\phi) = \\ &\begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\phi s\theta s\psi + c\phi c\psi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}. \end{aligned} \tag{4.19}$$

where $s(\cdot) = sin(\cdot)$ and $c(\cdot) = cos(\cdot)$, and where:

$$\boldsymbol{R}_z(\psi) = \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \boldsymbol{R}_y(\theta) = \begin{bmatrix} c\theta & 0 & -s\theta \\ 0 & 1 & 0 \\ s\theta & 0 & c\theta \end{bmatrix}, \boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & s\phi \\ 0 & -s\phi & c\phi \end{bmatrix}$$

with $\boldsymbol{R}_i(j)$ is a rotation matrix describing a rotation angle j about the i-axis.[11]

With the *zyx* rotation sequence, non-minimal parameterisations are required when the vehicle can engage in manoeuvres including pitch angles of $\pm\pi/2$, as this results in a kinematic singularity. In such a scenario, experienced for instance by spherical vehicles designed to operate in a "nose down" configuration when diving into vertical mine

[11] Should it be necessary to perform the inverse rotation, namely to rotate a vector from the inertial-fixed frame to the body-fixed frame, it is worth recalling that the coordinate transformation $\boldsymbol{R}_b^i(\boldsymbol{\Theta})$ is orthogonal, i.e. $\boldsymbol{R}_b^i(\boldsymbol{\Theta})\boldsymbol{R}_b^i(\boldsymbol{\Theta})^T = \boldsymbol{I}$ or else $\boldsymbol{R}_b^i(\boldsymbol{\Theta}) = \boldsymbol{R}_i^b(\boldsymbol{\Theta})^T$.

shafts [61], other parameterisations, such as the quaternions, need to be employed. For both surface vehicles and standard AUVs, such a singularity does not constitute a matter of concern and Euler angles can typically be employed. The overall expression for the hydrostatic vector Eq. (4.18) rotated in the body-fixed frame can be reported as [175]:

$$\boldsymbol{g}^b = -\begin{bmatrix} (W - B_h - B_V)\sin\theta \\ -(W - B_h - B_V)\cos\theta\sin\phi \\ -(W - B_h - B_V)\cos\theta\cos\phi \\ -(y_g^b W - y_h^b B_h - y_V^b B_V)\cos\theta\cos\phi + (z_g^b W - z_h^b B_h - z_V^b B_V)\cos\theta\sin\phi \\ (x_g^b W - x_h^b B_h - x_V^b B_V)\cos\theta\cos\phi + (z_g^b W - z_h^b B_h - z_V^b B_V)\sin\theta \\ -(x_g^b W - x_h^b B_h - x_V^b B_V)\cos\theta\sin\phi - (y_g^b W - y_h^b B_h - y_V^b B_V)\sin\theta. \end{bmatrix} \tag{4.20}$$

It should be noted that for neutrally buoyant vehicles, i.e. when $W = B_h + B_V$, gravity and buoyancy forces cancel each other out, while the restoring moments of the forces due to there being any COG-COB separation remain.

4.3.3 Actuators

Actuators and their contributions to control forces and moments need to be modelled next. AUV control strategies typically make use of four classes of actuators. These classes encompass movable masses, changes in the vehicle ballast, thrusters and control surfaces.

In practical implementations, the control signals employed are the acceleration along one or two directions of the movable mass, the ballast rate, the thruster force (or the Repetition Per Minute (RPM)) and the deflection angles of the control surfaces [20].

As this work is focused on the investigation of FTC applications where redundancy plays a key role, it is necessary to develop a simulator model that can accommodate all types of available actuators. In the following subsections, the modelling of the four classes of actuators are reported.

4.3.3.1 Control surfaces

First, control surfaces are considered, namely stern-planes and rudders. AUVs utilise the actuation of stern-planes and rudders (when the moving speed is sufficiently high) in order to control the attitude of the vehicle with minimal energy expenditure.

The *Grace* UG, for instance, makes use of a whale fluke-type tail to provide control of the vehicle vertical dynamics [189] or the *Sea-Whale 2000* employs two coaxial rudders and two coaxial elevators for control of the horizontal and vertical dynamics, respectively [156].

From a modelling perspective, a control surface has an effect on the overall vehicle lift. When actuated, the control surface contributes to the vehicle lift with an additional term proportional to the deflection angle. Additionally, as the control surface is typically located far from the COP due to design constraints, the generated control force exerts an additional moment with respect to the COP, affecting the pitch or yaw dynamics [190].

Under the previously stated assumption that the lift force in the typical range of underwater vehicle applications can be accurately approximated by a quadratic term with respect to velocity, the lift force and moment contributions of a control surface can be modelled as:

$$L_{cs} = \frac{1}{2}\rho C_{L_{cs}}(\alpha) S_{cs} \delta_{cs} u^2 \tag{4.21a}$$

$$M_{cs} = r_{cs_x} L_{cs} \tag{4.21b}$$

with $C_{L_{cs}}(\alpha)$ the control surface lift coefficient, and S_{cs} the control surface planform area, respectively, r_{cs_x} the axial position of the control surface with respect to the vehicle centroid in terms of body-fixed coordinates and δ_{cs} is the control surface deflection angle.

Under the quasi-steady state approximation introduced in Eq. (4.10), the control surface lift force and moment contributions can be simplified as [191]:

$$L_{cs} \approx K_{F_\delta} V_r^2 K_{u_\delta} \delta_{cs} \tag{4.22a}$$

$$M_{cs} \approx -K_M K_{u_\delta} \delta_{cs} V_r^2 \tag{4.22b}$$

where, following the convention previously introduced, $K_{F_\delta}, K_{u_\delta}$ and K_M are fixed-values coefficients.

4.3.3.2 Variable ballast devices

The effect of the ballast adjusting system is now considered. Two different designs of ballast systems can be employed: a syringe-type mechanism that displaces water into the hull to increase the vehicle mass or an external oil-filled bladder to modify the vehicle volume. According to the two mechanism designs, two modelling methodologies are required to capture the effect of the VBD.

For the syringe-type of ballast adjustment system the methodology is based on the control of the rate of change of the ballast mass $\dot{m}_b$ [20, 192, 193, 178]. With such modelling methodology, a VBD is modelled as an additional control input, as:

$$u_b = \dot{m}_b. \tag{4.23}$$

For the ballast adjustment systems employing oil-based bladders, the control is achieved by means of a reference volume ∇_V that can be inflated as needed, whose effect is embedded in the hydrostatic vector $\boldsymbol{g}^b(B_V)$ reported in Eq. (4.20), namely:

$$B_V = \rho g u_{VBD} \tag{4.24}$$

where the control input u_{VBD} defines the target volume of the bladder as $u_{VBD} = \nabla_V$.

Due to the extensive use of the aforementioned ballast mechanisms onboard AUVs, both models need to be included in the simulator.

4.3.3.3 Thrusters

Next, thruster modelling is considered. Underwater thrusters are devices designed to overcome the resistance to motion of the vehicle [184]. Underwater thrusters are com-

monly composed of a screw propeller. For traditional AUVs and hybrid UGs, electrically powered thrusters employing brushless Direct Current (DC) motors in a duct with either air-sealed, oil-sealed, or fully flooded cases represent the standard design choice. The spinning blades of the propeller induce a different pressure upstream and downstream of the propeller that in turn generates an exchange of momentum between the fluid and the device, in turn setting into motion the vehicle.

To quantify the generated thrust (T_p), nonlinear models that factor in the propeller diameter (D_p), propeller inlet and outlet velocities ($v_{p,in}$ and $v_{p,out}$, respectively) and the propeller rotation rate (n_p) can be employed as:

$$T_p = \rho D_p n_p (v_{p,out} - v_{p,in}). \tag{4.25}$$

Due to the impracticality of measuring inlet and outlet fluid velocities, simplified models such as the following can be utilised instead [147]:

$$T_p = \rho D_p^4 K_T n_p^2 \tag{4.26}$$

with K_T denoting a non-dimensional thruster coefficients, characteristic of each motor design.

More advanced models catering for the motor dynamic transient are available [112]. Nonetheless, dynamics responses inherent to underwater applications are comparatively slow to those of underwater thrusters, being the first time constants in the order of several seconds, while in the order of [ms] the second ones [194]. Due to the difference in these time constants, thrusters are usually considered as comparatively instantaneous sources of applied forces, and their dynamics is conventionally neglected [148].

4.3.3.4 Movable masses

Movable masses can be employed to modify the attitude of an AUV, by displacing the overall vehicle COG from its original location. Movable masses are usually of two different types, sliding masses or rotating masses. Internal components comparatively heavy with the overall vehicles mass, such as the batteries, are conventionally

employed for this purpose.

Some AUVs, such as the Slocum UGs, can actuate the movable mass along the longitudinal axis of the vehicle, namely along the x_b-axis [151, 153]. With such an actuator design, the pitch dynamics of the vehicle can be modified. Other vehicles, such as the Spray and Seaglider UGs, can also rotate the mass around the x_b-axis, allowing the roll dynamics to be affected [152]. Also, specific AUVs, such as the spherical UX-1, use a device rotating the battery pack to affect the pitch dynamics [150].

Different to the other three class of actuators described so far, no explicit expression to embed the effects of movable masses is available. As moving an internal mass displaces the COG of the vehicle, from a modelling perspective, this class of actuators requires dedicated extension of the vehicle equations of motions. In more detail, while a traditional AUV moving by means of thrusters and control surfaces can be modelled as a single rigid-body[12], any moving mass transforms the vehicle mounting such actuator in a multibody system, where the hull represents one rigid-body and a moving mass another rigid-body connected to the hull with dedicated joints.

To conclude, moving masses represent an element of major complexity when modelling AUVs. To tackle this critical element, object-oriented simulators, such as the one proposed in this chapter, can be utilised effectively. Specifically, moving masses can be modelled as standalone rigid-bodies, connected to the hull of the vehicle through mechanical joints. In the case of a sliding mass, a prismatic joint is employed to allow one (controlled) DOF, represented by the linear displacement of the mass, while the rotations between the mass and the hull are physically constrained. In the case of a rolling mass, a revolute joint can be employed to constrain the mass displacement while allowing on DOF as rotation. Combinations of linear and rolling joints can be employed to define more complex actuators.

In Section 4.5.1, 4.5.2 and 4.5.3, further examples and details are provided to better illustrate the concept of movable masses used as actuators.

[12]This modelling formalism holds under the stated assumption that the hull deformation with depth can be neglected.

4.3.4 Modelling assumptions summarised

Before moving to the simulator architecture design, the five modelling assumptions discussed so far are hereby recapped:

1. the hull compresses uniformly as the operating depth increases and, hence, the centre of buoyancy of the vehicle remains fixed over time with respect to hull;
2. added mass terms can be calculated under the assumption that the hull can be approximated as a prolate spheroid;
3. the quasi-steady state state hydrodynamic formulation is sufficient to characterise AUVs dynamics;
4. the vehicle operates fully submerged, and at a depth such that potential damping and potential added mass effects can be neglected;
5. thrusters are instantaneous sources of forces.

4.4 Simulator architecture design

Based on the arguments proffered at Section 4.2, OpenModelica was the selected IDE to simulate the AUV dynamics and the proposed operational environment. The simulator design process follows the structure proposed in [195], i.e. the vehicle is designed as a multibody model by linking components from the *Mechanics.MultiBody* library where available, and custom designed where preconfigured components are not available.

The rigid-bodies representing the hull, movable masses, static offset mass, VBD and steering surfaces were joint together through the use of prismatic and 3D revolute joints. External forces that comprise the hydrostatics, the hydrodynamics, added mass effects and control inputs were applied and exchanged between the bodies as reaction forces throughout interfaces representing the mechanical flanges.

A preliminary simulator design is shown in Fig. 4.3. In this design, the simulated system comprises the rigid-bodies associated with the hull, the sliding, the rolling, the offset, the ballast and the VBD masses. Two bodies representing the rudder and stern-planes are also added for the purpose of enhanced modularity.

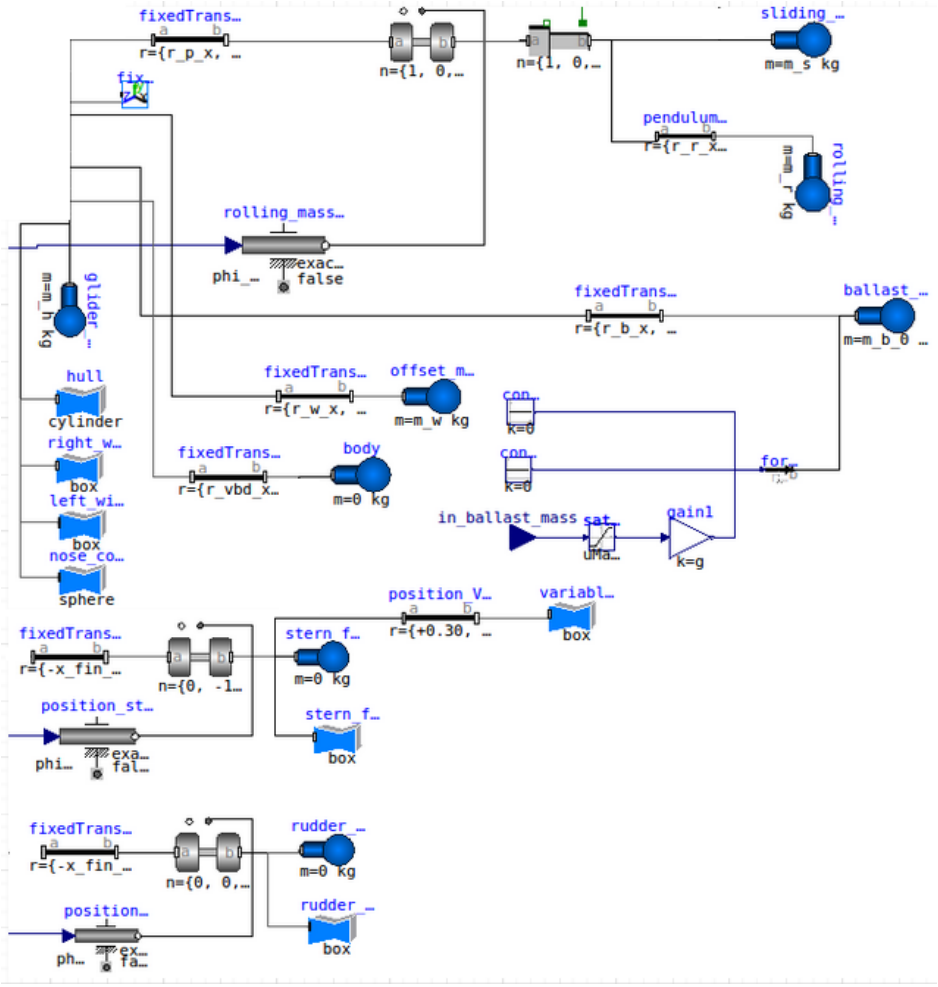


Figure 4.3: OpenModelica - preliminary simulator architecture for a multibody AUV, © 2022 IEEE [19].

Overall, the structure of the *OpenMAUVe* (Open Modelica AUV) library, encompassing the constituent elements of a generic AUV, is detailed in Fig. 4.4. The OpenMAUVe library relies on the MSL, or more specifically on the Modelica.Mechanical.Multibody for the definition of the *VehicleBody* and *Joints* classes, while it implies the design from scratch of the *ForcesMoments* and *Actuators* classes.

Three classes of external forces were then integrated into the simulator. Restoring forces, namely forces due to gravity and buoyancy, are expressed as global forces, meaning that they act on the entire simulation space as a uniformly distributed force field. Restoring forces are implemented as *ForcesMoments.Restoring* according to Eq. (4.20). Next, the hydrodynamic forces defined in Eq. (4.8), (4.9), (4.10) were implemented as *ForcesMoments.Hydrodynamics* and applied to the body representing

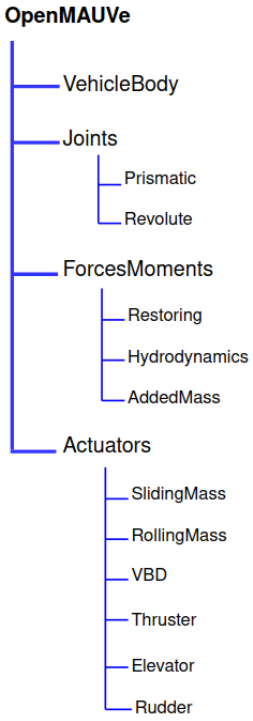


Figure 4.4: Proposed OpenMAUVe Modelica library, where *VehicleBody* and *Joints* are imported from the MSL, while *ForcesMoments* and *Actuators* are custom developed.

the hull of the vehicle. Similarly, added mass effects as defined in Eq. (4.12) were implemented as *ForcesMoments.AddedMass* and, once more, applied to the body representing the hull of the vehicle.

Hydrodynamic and added mass forces are dependent on the AUV velocity and acceleration, which are retrieved by means of *virtual sensors* measuring relative quantities between different reference frames. In this case, *Modelica.Mechanics.MultiBody.Sensors* were employed to measure the relative speeds and accelerations of the AUV's body-fixed frame with respect to the inertial fixed-frame.

Following the definition of the forces, the implementation of the elements linked to the actuators is discussed. As an example, the implementation of the elevator control surface class, defined within *OpenMAUVe.Actuators.Elevator*, is reported in Appendix A. The same design steps were followed for all the other components within the *ForcesMoments* and *Actuators* package.

As this study is ultimately focused on FTC applications, minor additions were

later needed to manage the occurrence of faults at actuators. Further details regarding the introduction of faults into the simulation are given in Chapter 7 with a dedicated case study.

4.5 Simulator verification and quasi-validation

In this section, the simulator verification and quasi-validation will be illustrated in steps of increasing complexity. The results generated by the OpenMAUVe were compared against simulated results of two different vehicles available in associated literature.

UGs were preferred over traditional AUVs for this comparison as they encompass the use of internal moving masses. As previously discussed, such moving masses significantly increase the complexity of the models to be simulated due to the resulting multibody dynamics involving the exchange of reaction forces and moments between the external hull and internal moving components.

The vehicles selected for the comparison were chosen based on both the public accessibility of geometric and hydrodynamic parameters and on the availability of results to allow for a quantitative comparison. First, a comparison of solely the vertical dynamics in the sagittal plane (namely the plane defined by the x_i-z_i axes) was carried out to perform an initial simulator verification. Next, complete dynamic motions in the 3D-space were performed and compared against the result output from a second UG simulation and related deployment data.

It is worth noting that there is a current lack of publicly available data to perform a definitive simulator validation against deployment data. Specifically, either deployment data are available but hydrodynamics coefficients or key geometric parameters (such as the location of the internal masses) are missing, or, vice versa, the coefficients are available but the deployment data is not public. At times, the control inputs to the actuators are not logged instead, making impossible to attempt reproducing the same manoeuvres.

This lack of results is believed to be due to the current practices in operating UGs. The vehicles are first trimmed and ballasted in dedicated tanks before shipping the vehicle to a mission location. Before deployment, due to small variations in the

expected water density and due to possible changes in the arrangement of actuators and sensors, the vehicles undergo a final ballasting procedure in loco. During the latter procedure, small weights are used to render the vehicle neutrally buoyant, and to trim the attitude to avoid undesired pitch and roll angles. These changes are usually logged by the deployment teams, but rarely released to the public as associated to the specific mission. Even if these changes are minor, in order to claim the complete validation of the OpenMAUVe simulator, further work is required to collect the required data and coefficients for a set of deployment missions and quantify the discrepancies of the simulation results with the recorded data.

Therefore, an initial verification, i.e. a comparison with the theoretical model is presented first in Section 4.5.1. Then, a quasi-validation, i.e. a comparison with another simulator that has been validated against deployment data, but for which deployment data is not public, is presented in Section 4.5.2.

4.5.1 ROGUE glider

The ROGUE UG is the first vehicle integrated and tested to verify the vertical dynamics [20]. The ROGUE is a laboratory-scale UG employing an internal ballast tank, designed with a small ellipsoidal body with axes of 0.20x0.31x0.15 [m], allowing for experiments of dynamics and control in small water tanks (less than 10 [m] in length) or Olympic sized pools.

According to the original study [196], the quasi-steady hydrodynamic force approximation can be employed for this study, obtained by restricting Eq. (4.7) to the vertical plane only as:

$$\boldsymbol{F}_{hd}^{f} = \begin{bmatrix} -D & 0 & -L \end{bmatrix}^{T} \tag{4.27a}$$

$$\boldsymbol{M}_{hd}^{f} = \begin{bmatrix} 0 & M_{DL} & 0 \end{bmatrix}^{T} \tag{4.27b}$$

where:

$$D \approx (K_{D_0} + K_D \alpha^2)(u_r^2 + w_r^2) \tag{4.28a}$$

$$L \approx (K_{L_0} + K_L \alpha)(u_r^2 + w_r^2) \tag{4.28b}$$

$$M_{DL} \approx (K_{M_0} + K_M \alpha)(u_r^2 + w_r^2). \tag{4.28c}$$

Hydrodynamic coefficients and geometric parameters of the vehicle selected for this simulation are reported in Table 4.1.

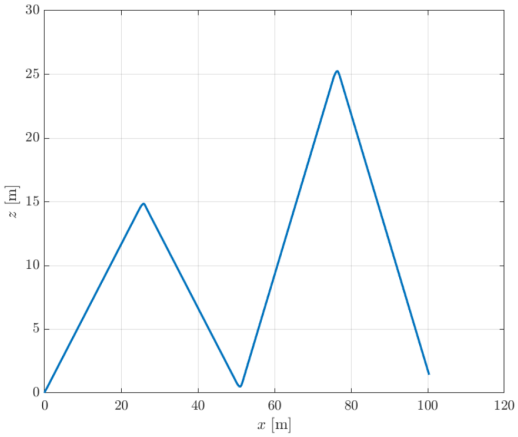
Table 4.1: ROGUE dynamical parameters.

Parameters	**Values**	**Units**	**Explanation**
m_h	8.22	kg	Vehicle hull mass
m_p	2.0	kg	Shifting mass
∇_h	11.22×10^{-3}	m^3	Vehicle volume
z_g^b	4×10^{-3}	m	COB-COG separation along z_b-axis
$X_{\dot{u}}$	2.0	kg	Added mass coeff. on x_b-axis generated by acceleration along x_b-axis
$Z_{\dot{w}}$	2.0	kg	Added mass coeff. on z_b-axis generated by acceleration along z_b-axis
I_{yy}	0.1	kg m^2	Moment of inertia around y_b-axis
K_{D0}	18.0	kg/m	Quasi-steady state drag force (offset) coefficient
K_D	109.0	kg/(m rad^2)	Quasi-steady state drag force coefficient
K_{L0}	0.0	kg/m	Quasi-steady state lift force (offset) coefficient
K_L	306.0	kg/(m rad)	Quasi-steady state lift force coefficient
K_{M_0}	0.0	kg	Quasi-steady state moment (offset) coefficient
K_M	-36.5	kg/rad	Quasi-steady state moment coefficient

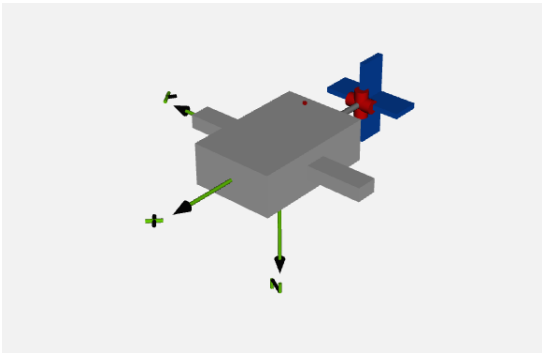
To verify the simulator, four different sets of command inputs were used to test the gliding dynamics of the vehicles, as selected from the relevant reference [20]. A

single path comprised of four segments encompassing two ascending and the two descending sections with different characteristics can be simulated. When operating an UG, a series of defined, steady straight-line paths are commanded, by setting a constant flowspeed and a *glide path angle* for each segment. The *glide path angle* ξ is defined as $\xi = \theta - \alpha$.

For this simulation, a flowspeed of $V_r = 0.30$ [m/s] was selected to perform two paths at $\xi = -30°$ and $+30°$. Next, a higher flowspeed of $V_r = 0.37$ [m/s] was set to perform two steeper paths at $\xi = -45°$ and $+45°$. Fig. 4.5a reports the UG path in the sagittal plane, as composed by the four defined segments. A 3D rendering of the vehicle during the first descending segment is shown in Fig. 4.5b.



(a) Path in the sagittal plane, © 2022 IEEE [19].



(b) OpenModelica 3D rendering, © 2022 IEEE [19].

Figure 4.5: ROGUE dynamics simulation.

Finally, the numerical results obtained from the OpenMAUVe simulator during the path were compared with the theoretical dynamic performance [20]. For each ascending-discending section, the theoretical quasi-steady model shows symmetrical behaviours, namely opposite values in terms of the flowspeed and angles ξ, θ, and α, as reported in Table 4.1 of reference [20]. Hence, the results of the first and fourth sections of the glide were compared, whilst the second and third were not reported as the segments are symmetrical. Table 4.2 reports the comparison of V_r, ξ, θ, and α in terms of absolute values and corresponding relative errors.

The results used for comparison ([20]) were obtained by solving the equations

Table 4.2: Theoretical vs. simulation results, © 2022 IEEE [19].

Quantity	ROGUE theoretical dynamics [20]	OpenMAUVe	Relative error
Down 30°			
ξ [°]	-30.0	-30.18	0.59%
θ [°]	-23.0	-23.97	4.05%
α [°]	6.3	6.21	1.43%
V_r [m/s]	0.30	0.30	0.0%
Up 45°			
ξ [°]	45.0	45.12	0.27%
θ [°]	41.5	41.7	0.01%
α [°]	-3.5	-3.42	0.02%
V_r [m/s]	0.37	0.37	0.0%

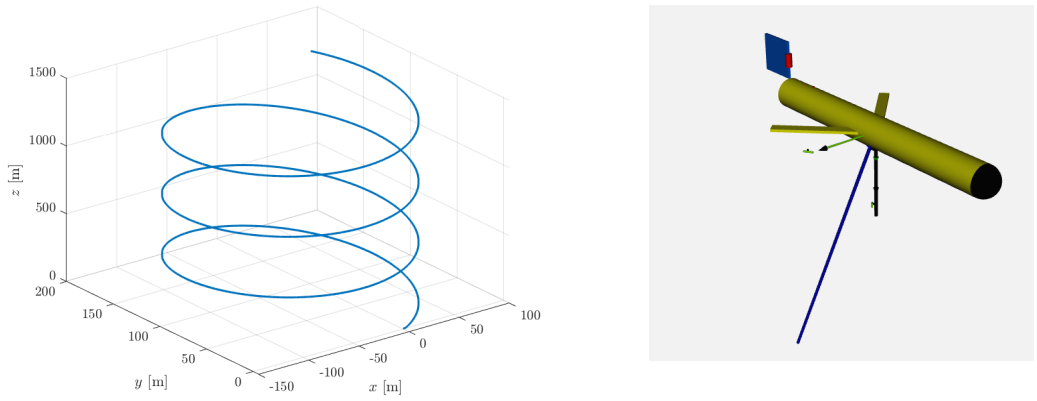
of motion via a numerical solver tasked with explicitly solving the theoretical model equation. This analysis highlights that the proposed simulator can produce results from dynamic simulation with a maximum relative error bounded below 4.05% when compared to the theoretical values. This first verification stage shows that, given a prescribed quasi-steady dynamic model of an UG in the sagittal plane, the simulator can qualitatively capture the dynamic effects as defined by closely matching the theoretical equations of motion. A more advanced study expanding these preliminary results from the sole sagittal plane to the whole 3D space is proposed next in Section 4.5.2.

4.5.2 Seawing glider

As a second step of quasi-validation of the OpenMAUVe simulator, a more complex dynamic manoeuvre encompassing changes of both linear and angular momenta was simulated. The Seawing UG was selected for this study as the associated literature was deemed the most complete source of publicly available data [179]. The vehicle dynamics are analysed while performing a single continuous descending spiralling dive, spanning over the whole 3D underwater space.

For this second test case, a broader range of actuators within the OpenMAUVe was employed. First, the VBD was commanded to increase the overall vehicle mass, second the sliding mass was moved towards the bow of the vehicle and, third, the

rolling mass was shifted to the port side so that the vehicle attains a constant roll angle. The resulting 3D path, obtained by simulating the UG dynamics for 5000 [s], is reported in Fig. 4.6a. A corresponding rendering of the Seawing UG is illustrated in Fig. 4.6b, where the blue arrow denotes the overall force exerted on the vehicle during the spiralling manoeuvre.



(a) Path in the 3D space, © 2022 IEEE [19].

(b) OpenModelica 3D rendering.

Figure 4.6: Seawing dynamics simulation.

The numerical results obtained from the OpenMAUVe simulator were then compared to those computed from a MATLAB-based simulator designed in [179], and reported in Table 4.3.

Table 4.3: Simulation results comparison, © 2022 IEEE [19].

Quantity	**Seawing reference [179]**	**OpenMAUVe**	**Relative error**
Radius [m]	100.83	93.88	6.89%
α [°]	1.267	1.258	0.71%
β [°]	-1.283	-1.396	8.81%
V [m/s]	0.490	0.491	0.2 %
r [rad/s]	0.0039	0.0033	15.39%
θ [°]	-13.703	-13.011	5.05%
ϕ [°]	-35.641	-35.643	0.01%

As already noted in the ROGUE case study, the OpenMAUVe simulator generated dynamic behaviours of the Seawing comparable to the ones obtained in associated literature [179]. Overall, the OpenMAUVe architecture generated a descending spiral

at the prescribed flowspeed of 0.49 [m/s] with a 0.2% error when compared to [179].

Numerical discrepancies between the proposed simulator and the MATLAB-based model (at reference [179]) exist. On one hand, a comparatively small steady-state roll angle relative error of 0.01% was obtained. On the other hand, the worst-case difference was recorded in the yaw angular rate r, with a relative error of 15.39% on r as expressed in [rad/s]. It should be noted that, even if this relative error appears to be comparatively large, it represents a difference of 0.03 [deg/s] (in other terms, the absolute error is comparatively low).

The two simulators under comparison employ different integrator methods, as DASSL is selected as the solver for the OpenMAUVe design, an option not available within MATLAB. Small differences in the absolute values within the dynamics are naturally linked to the selected simulation setup, specifically to the choice of the integration method and solver tolerance. The different setups considered for this study render impossible to provide a conclusive answer as simulations with the exact same choice of integration method and tolerance cannot be performed.

As it can be noticed in the results offered within the Seawing report [179], discrepancies between the MATLAB simulator and deployment data naturally exist. To resolve the conundrum about which two simulation methods (MATLAB from [179], or OpenMAUVe) most accurately reproduces UGs dynamics, a comparison with real-deployment data is necessary. This step requires precise information regarding both the hydrodynamic parameters of a vehicle, the ballasting and trimming configuration chosen for a specific mission, and the presence of a dataset containing the actuators commands and the corresponding estimate of velocities, accelerations and attitude. Such information is currently not available to the public domain, and this study is highlighted as future work.

4.5.3 Simulator utilisation example: hybrid AUV

Following the verification and quasi-validation of the OpenMAUVe simulator, an example of utilisation of the tool is illustrated in this section. After proposing the utilisation of the OpenMAUVe to simulate two UGs, in this section an hybrid AUV is showcased instead. Hybrid AUVs represent the most complex category of AUVs, as

they feature both internal moving masses, thrusters and control surfaces.

Assume that an underwater vehicle with a set of five actuators is to be simulated. The vehicle is comprised of a sliding mass, a rolling mass, as VBD and two elevators. The vehicle will be denoted as *HybridAUV1* (the lack of imagination derives from how OpenModelica automatically instantiate a new class name).

The vehicle is to be subjected to hydrostatic, hydrodynamic, added mass and control forces and moments. The hybrid AUV structure was built, starting from the hull, by importing a single rigid-body as *OpenMAUVe.VehicleBody*. Next, restoring forces were added, namely those forces due to gravity and the buoyancy *OpenMAUVe.ForcesMoments.Restoring*. Added mass and hydrodynamics effects were added following the same procedure, by importing the corresponding elements from *OpenMAUVe.ForcesMoments.Hydrodynamics* and *OpenMAUVe.ForcesMoments.AddedMass*, respectively. Restoring, Hydrodynamics and AddedMass were then connected to the reference point within the hull volume, a point that coincides with the COB of the hull, or its volume centroid. The hull's COG, where the gravity force is applied, was displaced from the point of application of the forces by means of a translation element, imported from the *Mechanics.MultiBody.Parts.FixedTranslation*.

Next, the dedicated set of actuators was imported. A rolling mass and and sliding mass were added, interfaced to the rigid hull by means of a revolute and a prismatic joint, respectively. Finally, the two elevators were included as defined from *OpenMAUVe.Actuators.Elevator*. After importing the elements as described, the newly defined *HybridAUV1* class resulted in the connection diagram as illustrated in Fig. 4.7.

The HybridAUV1 class can in turn be imported as a single element in a wider scope simulator, for instance to analyse its dynamics when a change to the VBD reference is commanded, or to test the performance of a control system. The instanced object, reported in Fig. 4.8, exhibits the five actuators signals as inputs. In this example, the set of three linear velocities expressed with respect to the body-fixed frame was selected as output variables. In place of the set of linear velocities, variables, such as position, attitude, or angular velocities can be selected, based on the desired

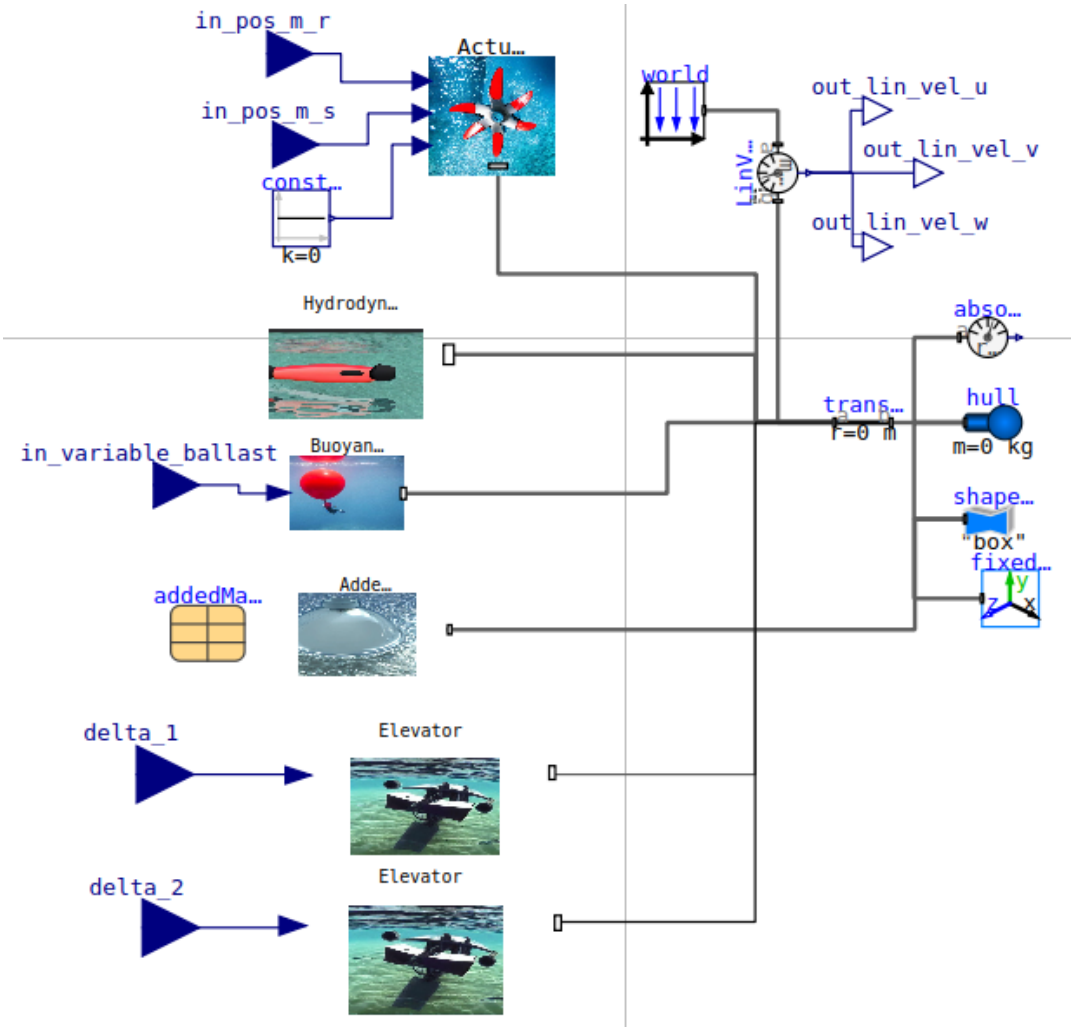


Figure 4.7: *HybridAUV1* class design.

application.

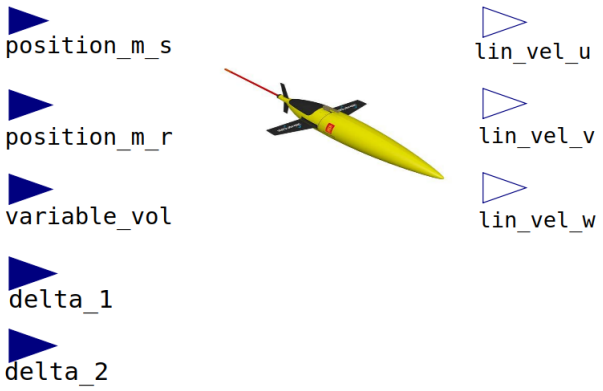


Figure 4.8: *HybridAUV1* icon, to be imported in a wider scope simulation.

A complete example of how an AUV can be designed within the OpenMAUVe

and used in a wider control-oriented simulator is presented and discussed in Chapter 7.

4.6 Conclusions

Simulators of underwater vehicles are of particular interest to investigate the complex dynamics involved in underwater vehicles modelling and their interaction with the highly nonlinear marine environment. The wide range of underwater vehicles stems into significantly different dynamic models that usually require ad hoc re-implementation. UGs and hybrid AUVs are specifically difficult to simulate given the multibody nature of their designs, comprising several internal moving masses that modify the momentum of the overall body.

In control applications, simplified control-oriented models are typically devised, for instance by decoupling the sagittal and horizontal plane. Nonetheless, to verify the correctness and performance of a synthesised control functions over the complete nonlinear dynamics, a simulator accounting for the entirety of the nonlinear effects is required. Given the large set of underwater vehicles and possible actuator configurations available, modularising the simulation design process can significantly simplify the testing and verification of control laws, eventually improving vehicle resilience in a challenging environment, such as the underwater domain.

In this chapter, an open-source simulator architecture for underwater vehicles was first designed, allowing the designer to compose dynamic models by assembling the vehicle through selecting elements that replicate its physical components. Next, the simulator was verified and quasi-validated by performing two test scenarios of increasing complexity and comparing the results obtained with those made available from other research.

Current simulation limitations encompass environmental disturbances, depth-related hull expansion phenomena and non-idealised behaviours linked to the presence of hull appendages. These elements were deemed not critical for this investigative study and will be addressed in future work. Finally, a complete validation of the simulator is required, which will be a comparison with real deployment data. The validation stage requires at the same time the availability of an open-access dataset, the specific

details regarding the vehicle trimming configuration and the logging of real-time actuator commands. Specific information regarding the position of ballast masses, the sensors mounted onboard and the trimming conditions are necessary to achieve a sufficient level of modelling accuracy to provide a credible level of validation, and are suggested as future extension to the work presented in this chapter.

Moreover, the OpenMAUVe simulator allows for efficient simulation of long-lasting missions, owing to the enhanced capabilities to solve systems of DAEs when compared to standard causal simulators. This can allow testing the effect of slow dynamical phenomena, such as the growth of biofouling agents on the hull. Finally, given the capability to assemble vehicles through combination of mechanical components, the simulator can be used as a fast prototyping tool for novel vehicle design, allowing to promptly investigate the vehicle dynamics when subjected to different control inputs.

Having now completed the design of the simulation architecture, focus is shifted onto the design of the control method and in the introduction of fault, which will be object of the following two chapters. To conclude, the OpenMAUVe simulator will be used in a complete scenario covering an AUV control application in Chapter 7.

Chapter 5

Augmented Neural Lyapunov Control: a method to automatically synthesise nonlinear control functions for nonlinear dynamical systems

5.1 Introduction

This chapter outlines the design of the ML-based control method for devising control laws for generic nonlinear systems. First, the Lyapunov theory is introduced, followed by an overview of methods aimed at the automatic synthesis of control laws. Next, the NLC method, identified as the favourite alternative to automatically synthesis control laws for nonlinear systems, is tested over a benchmark nonlinear system to evaluate the robustness to parameter initialisation. After the critical aspects of the methods are identified, an upgraded method is devised, addressing the NLC limitations through tailored modifications. An open-source software tool is then designed and illustrated. The newly defined method is challenged over two nonlinear systems and a comparison with the NLC is carried out, while performances are evaluated. Finally, conclusions are drawn and future research direction is presented.

As part of this research, a preliminary paper titled "Augmented Neural Lyapunov Control" was published [22]. The software framework associated with this

chapter is released open-source at: https://github.com/grande-dev/Augmented-Neural-Lyapunov-Control.

The outcomes described in this chapter provide answers to RQ2: **is the Neural Lyapunov Control method an effective approach to design stable control laws for nonlinear systems?**

5.2 Preliminaries

In this section, an overview of the Lyapunov's stability theory is provided first. Fundamental concepts and definitions such as a system's equilibrium, Region of Attraction (ROA) and an equilibrium's stability are introduced. Next, the different methods currently available to synthesise Lyapunov Functions (LFs) and utilisation of formal methods to ensure correctness of results are discussed.

5.2.1 Lyapunov's theory preliminaries

The aim of the control engineering discipline is the study of the dynamical behaviours of nonlinear systems and, consequently, how such behaviours can be modified to attain prescribed performance. The dynamical behaviour of a nonlinear system can be generically described as an n-dimensional ODE;

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u}) \tag{5.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the vector of the *states*, $\boldsymbol{u} \in \mathbb{R}^m$ is the vector of the *inputs*, $t \in \mathbb{R}^+$ denotes that the system is time-variant, the $\dot{\boldsymbol{x}} \in \mathbb{R}^n$ notation stands for the time-derivative of $\boldsymbol{x}$; and, $\boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u})$ denotes a continuous *vector field* defined over $\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, where a vector field defines a mapping from a domain $\mathcal{D}$ to an n-dimensional column vector.

A key element of the control engineering design process is the analysis of the *equilibria* of a dynamical system.

Definition 2 (Equilibrium [197]) *A point $\boldsymbol{x} = \boldsymbol{x}^\star$ within the state-space is said to be an equilibrium of Eq.* (5.1) *if it has the property that whenever the state of the system starts at $\boldsymbol{x}^\star$, the system state will remain at $\boldsymbol{x}^\star$ for all future time.*

In other words, a state of equilibrium of system (5.1) is found by solving $\dot{\boldsymbol{x}} = 0$, $\forall t \in \mathbb{R}^+$, which represents a constant solution of the ODE.

Upon identification of the equilibria of a system, the control engineering discipline focuses on analysing the nature of the nonlinear dynamical phenomena and on how said phenomena may be modified via tailored control laws. Different to linear systems where the (unique) equilibrium is either asymptotically stable, stable (in the sense of Lyapunov) or unstable (with these concepts later introduced in this chapter) [92], a range of different behaviours are to be expected in nonlinear systems. Equilibria can stem in foci, in limit cycles in second-order systems, and, among others, in toroidal attractors or chaotic (or strange) attractors in higher-order systems [198].

One of the most common techniques employed to analyse the stability of a dynamical system is Lyapunov's theory [92, 199]. In Alexandr Mikhailovich Lyapunov's PhD thesis (1892) [200], two methods are introduced to analyse the stability of nonlinear systems, conventionally referred to as *Indirect* and *Direct* Lyapunov methods in associated literature [92].

The *Indirect method* (or *linearisation method*) allows one to infer the local stability properties of an equilibrium by analysing the sign of the eigenvalues associated with the corresponding Jacobian matrix, obtained by means of linearisation around the equilibrium point. The Indirect method states that the stability properties of the nonlinear system in the neighbourhood of an equilibrium resemble the behaviour of its linearised approximation.

In contrast to the Indirect method, the more sophisticated *Direct method* does not require linear approximations for the assessment of the nature of an equilibrium, allowing conclusions to be drawn as regards the global behaviour of the nonlinear system. This latter method is currently considered the fundamental pillar of modern nonlinear control theory [92]. The Direct method can be employed to both assess the stability of an equilibrium, or to support the design of control laws that modify the nature of a system's equilibrium.

In this work, the focus lies on the Direct method due to the fundamental properties it encompasses. The study of closed-loop stability allows the control engineer to

draw specific conclusions about the global stability properties, providing key insight on the system trajectories over the full state-space domain. Additionally, the Lyapunov Direct method allows the ROA of an equilibrium to be estimated, so assessing the robustness of a closed-loop system. Note that the ROA is at times also defined as *region of asymptotic stability*, *domain of attraction*, or *basin* [197].

Definition 3 (Region of Attraction [197]) *Let $\boldsymbol{x}^\star = 0$ be an (asymptotically) stable equilibrium point for the nonlinear system;*

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}) \tag{5.2}$$

where $\boldsymbol{f}(t,\boldsymbol{x}) : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is defined over the domain $\mathcal{D}$ containing $\boldsymbol{x}^\star$. Let $\bar{x}(t,\boldsymbol{x})$ be the solution of (5.2) *that starts at initial state $\boldsymbol{x}_0$ (reads $\boldsymbol{x}$ at time t = 0). The ROA of the origin, denoted as R_A, is defined by;*

$$R_A = \{\boldsymbol{x} \in \mathcal{D} : \bar{x}(t,\boldsymbol{x}) \to \boldsymbol{x}^\star \text{ as } t \to \infty\}. \tag{5.3}$$

In other terms, the ROA of an equilibrium defines the set of initial values of the state-space that gives rise to trajectories that eventually converge to the equilibrium.

The Lyapunov Direct method concludes that the stability of a generic equilibrium $\boldsymbol{x}^\star$ can be inferred by means of a scalar pseudo-energy function that decreases as the state-space trajectories approach $\boldsymbol{x}^\star$. Such a function is referred to as a LF in nonlinear control literature. When a LF is associated with a dynamical system with exogenous inputs, such as those considered in this work, it is referred to as a CLF instead [201].

Definition 4 (Control Lyapunov Function [202]) *Given a domain $\mathcal{D}$ and a system model $\boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{u}) : \mathcal{D} \times \mathcal{U} \to \mathcal{D}$ with unique equilibrium $\boldsymbol{x}^\star \in \mathcal{D}$, i.e. $\boldsymbol{f}(t,\boldsymbol{x}^\star,\boldsymbol{u}) = 0$ with $\boldsymbol{x} \in \mathcal{D} \subset \mathbb{R}^n$ and $\boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m$, consider a function $V : \mathcal{D} \to \mathbb{R}$, $V \in \mathscr{C}^1$, then V is a Control Lyapunov Function if there exists $\boldsymbol{u}$ such that:*

$$V(\boldsymbol{x}^\star) = 0, \tag{5.4a}$$

$$V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}, \tag{5.4b}$$

$$\dot{V}(\boldsymbol{x},\boldsymbol{u}) = \langle \boldsymbol{\nabla}(\boldsymbol{V})_{\boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \rangle < 0, \quad \exists \boldsymbol{u}\ \forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}, \tag{5.4c}$$

where $\mathscr{C}^1$ denotes a function with continuous first derivative[1], and where $\langle \boldsymbol{a}, \boldsymbol{b} \rangle$ denotes the *inner product* of the terms $\boldsymbol{a}$ and $\boldsymbol{b}$, namely $\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i$ [197]. Additionally, $\langle \boldsymbol{\nabla}(\boldsymbol{V})_{\boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \rangle$ represents the Lie derivative of V with respect to $\boldsymbol{x}$ (or *along* $\boldsymbol{x}$), and defines the derivative of V along the trajectories of the system $\dot{\boldsymbol{x}} = \boldsymbol{f}(t,\boldsymbol{x},\boldsymbol{u})$.

The Lyapunov Direct method determines that, if such a function V that satisfies (5.4) exists, the point of equilibrium $\boldsymbol{x}^\star$ is *stable* (or *stable in the sense of Lyapunov*) [92]. As a consequence, the stability of $\boldsymbol{x}^\star$ implies that the system trajectories can be kept arbitrarily close to $\boldsymbol{x}^\star$ if the initial conditions of these trajectories are set sufficiently close to $\boldsymbol{x}^\star$.

Definition 5 (Stability of an equilibrium [92]) *An equilibrium $\boldsymbol{x}^\star = 0$ is said to be stable if, for any $R > 0$, there exists $r > 0$ such that if $\|\boldsymbol{x}_0\|_2 < r$, then $\|\boldsymbol{x}(t)\|_2 < R\ \forall t \in \mathbb{R}^+$,*

with $\|\boldsymbol{a}\|_2 = \sqrt{a_1^2 + ... + a_n^2}$ denoting the 2-norm of vector $\boldsymbol{a}$. In other terms, the stability definition implies that initialising the system within a domain B_r defined by the n-dimensional sphere of radius r (or hyper-sphere), no trajectory will ever leave a domain B_R defined by the n-dimensional sphere of radius R.

An example of a stable trajectory is illustrated in Fig. 5.1, highlighted by trajectory t_1: starting from a value $\boldsymbol{x}_0$ within B_r, the trajectories never leave the domain B_R (in this example $\boldsymbol{x}^\star$ is a stable node, or focus).

Additionally, if $\dot{V}(\boldsymbol{x},\boldsymbol{u}) < 0\ \forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}$ in (5.4) (it reads Lie derivative strictly less than 0), the equilibrium is defined as *asymptotically stable*. Asymptotic stability implies that $\boldsymbol{x}^\star$ is *stable* and that, additionally, $\boldsymbol{x} \to \boldsymbol{x}^\star$ as $t \to \infty$. One such example of

[1] In this case, V is $\mathscr{C}^1$ with respect to its argument, namely the first partial derivative with respect to $\boldsymbol{x}$ exists and it is continuous.

asymptotically stable $\boldsymbol{x}^\star$ is reported in Fig. 5.1 as trajectory t_2.

Conversely, if it is possible to find a radius r such that when initialising the system within B_r, at least one trajectory will no longer satisfy the conditions to remain within B_R, $\boldsymbol{x}^\star$ is defined as *unstable* (no matter how small r is). Instability of $\boldsymbol{x}^\star$ implies that either the trajectories *"blow up"*, or, in other terms, that the trajectories move further away from $\boldsymbol{x}^\star$ as time progresses (e.g. trajectory t_3 in Fig. 5.1), or that more complicated nonlinear phenomena are to be expected. For instance, in the presence of a limit cycle, as illustrated in Fig. 5.1 in pink solid line, it is possible to select a value of B_R such that for every $r > 0$, the trajectories initialised within B_r leave B_R, eventually converging to the limit cycle (e.g. trajectory t_4 in Fig. 5.1). Without loss of generality, $\boldsymbol{x}^\star$ is conventionally considered such that $\boldsymbol{x}^\star = 0$, as it is always possible to translate the equilibrium through application of an appropriate substitution of variables, as will be shown later in this work.

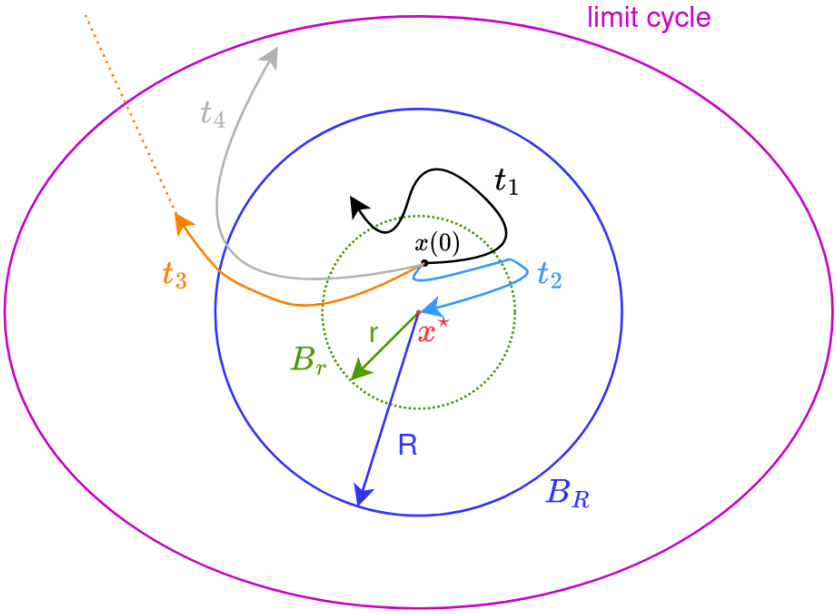


Figure 5.1: Lyapunov stability theory: t_1 defines a *(Lyapunov) stable* equilibrium, t_2 an *asymptotically stable* equilibrium, and (t_3, t_4) an *unstable equilibrium*, with t_4 covering the specific case of trajectories converging to a *limit cycle*.

5.2.2 Automatic synthesis of (Control) Lyapunov Functions

Despite the advantages of the Lyapunov Direct method, the composition (or *synthesis*) of (C)LFs is a highly complex undertaking [92]. In the 132 years that have elapsed

since the publication of Lyapunov's theory to the writing of this thesis, no general closed-form solution has yet been derived to synthesise CLFs for generic nonlinear systems. Simplifications are available in the case of linear systems or for nonlinear systems with specific characteristics, such as when $\dot{\boldsymbol{x}}$ is a polynomial vector field [199]. Control engineers conventionally attempt to synthesise quadratic CLFs, but, in general terms, CLFs can have arbitrarily complex shapes.

The matter is rendered more complicated as the inability to find a CLF does not imply the instability of the equilibrium [21]. In actuality, when no CLF is derived, one can only state that *despite the design efforts, no CLF has been obtained*, and not that the system is unstable.

Several numerical methods have thus far been proposed in the literature to devise and automate the synthesis of (C)LFs: notably, piecewise (linear or quadratic) and Sum-of-Squares LFs [203, 204], Genetic Algorithms [205, 206] and ANNs [207, 21, 199]. Within the context of ANNs, the concept of Lyapunov Neural Network (LNN) takes shape, representing the use of a feedforward ANN as a function approximator for LFs [21].

Owing to their universal approximation capabilities, ANNs are good candidates to be employed in the quest for deriving LFs. The original LNN concept deals with discrete-time systems, while this thesis focuses on continuous-time dynamics. Nonetheless, as the LNN introduces key ideas that will be employed later on in this work, it is reported here for completeness. In the case of the LNN, the input of the ANN is represented by the dynamics of the system ($\boldsymbol{x}_{k+1}$), while the output is represented by the associated scalar value of the LF ($V(\boldsymbol{x}_{k+1})$), as reported in Fig. 5.2.

To assess the stability of the system, the error between two consecutive time steps is computed, as:

$$e_L = V(\boldsymbol{x}_{k+1}) - V(\boldsymbol{x}_k). \tag{5.5}$$

Should the LNN approximate a LF, then $e_L \geq 0$ must hold, namely the function value decreases as the time progresses. Additionally, the e_L is used to update the weights of the LNN through a Gradient Descent (GD) training algorithm, such that the LNN incrementally increases the degree of approximation of a LF. If, during the training

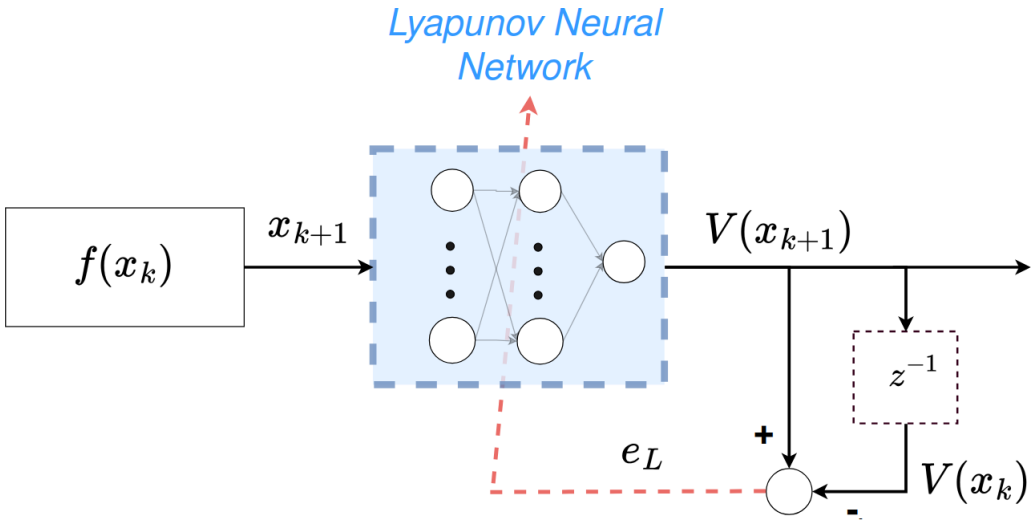


Figure 5.2: Using a feedforward ANN to represent a Lyapunov Function, with the red line denoting the loss function back propagation (adapted from [21]).

process, the LNN weights do not require update for a *sufficiently high* number of training iterations, the LNN is assumed to be a good approximation of the LF. On the other hand, if the LNN does not converge, no inferences can be drawn as regards the stability of the target equilibrium; it is possible that either the LNN architecture has been poorly designed, or that the training data has not been sensibly selected, or even that the system is indeed unstable, or a combination of the above.

Another early example of the use of ANNs to assess the stability of equilibria with ANN is the *Lyapunov Machine* [207]. A Lyapunov Machine is an ANN employed to test positive definite semi-trajectories and, in turn, to assess the global asymptotic stability of an equilibrium of a nonlinear system by proposing a candidate LF.

These works show a long-lasting effort to provide empirical methods to synthesise CLFs with modern tools, in an attempt to devise more efficient techniques when compared to the manual trial and error process. However, at the present stage, these methods suffer certain shortcomings, mainly the lack of a formal proof of stability.

The problem of the automatic synthesis of CLFs is further complicated. The candidate CLFs need to be proposed such that they obey the theoretical Lyapunov conditions while, in parallel, a formal procedure to establish their correctness needs to be put in place. In fact, even with assuming that a CLF can be obtained via an automated procedure, the use of numerical methods alone cannot guarantee the correctness of the

generated CLFs, for which further certification needs to be devised.

5.2.2.1 Lyapunov Function synthesis via Neural Lyapunov Control

Recent focus has been placed on providing a *formal certification* of the correctness of CLFs, resulting into the NLC method [134]. Several works explore different ANN architectures combined with formal verification tools to return correct-by-design solutions [135, 136]. This new stream of learning architectures exploits a loop between a *Learner* and a *Falsifier*. The Learner, using a finite set of samples, trains a neural network to represent a candidate CLF, resembling the training stages of the early methods previously introduced [207, 21]. In addition, the Falsifier module checks whether the candidate CLF satisfies the conditions in (5.4) in a domain $\mathcal{D}$ over the Reals.

Advancements in formal techniques allow the verification of candidate CLFs as *certificate functions* [208], by exploiting Satisfiability Modulo Theories (SMT) solvers, capable of evaluating the correctness of a symbolic expression over a continuous domain, e.g. a subset of the Reals, and of assessing whether a formula holds true with respect to a prescribed set of constraints [209]. While purely empirical (i.e. numerical) methods rely on a finite-sized dataset, and assess whether a candidate CLF is *sufficiently correct* via prescribed rules (e.g. by evaluating a sufficiently large number of points, or through interrupting the training after a pre-defined number of successful learning iterations), formal methods allow conclusions to be reached regarding the unconditional correctness of a CLF.

The SMT class of solvers guarantees the *soundness* of the solution: when a logic expression is evaluated as *True*, the result is deemed to be valid over the entire search domain [210]. SMTs guarantee that even if the training relies on a finite number of data points, the correctness of a CLF over a dense domain can be formally certified, producing results equivalent to those of analytical proofs [135].

Furthermore, when the SMT solvers verify that an expression does not hold true, a point violating the expression is returned. These points, referred to as counterexamples (CEs), are added to the training set to further assist the scouting of CLFs. Such an approach, based on a learning-verifying paradigm, with CEs iteratively added to the training dataset is known as Counterexample-Guided Inductive Synthesis (CEGIS)

[211]. The CEGIS architecture relies on the adversarial interaction of two modules, jointly supporting the design of a desired function from an initial hypothesis space, and has found success in several recent works [212, 134, 213, 214].

5.2.2.2 Satisfiability Modulo Theories

SMT solving involves the problem of deciding whether a mathematical formula is *satisfiable*. Given a mathematical formula with specified variables, the goal of the SMT solvers lies in searching for values of the variables that render the formula true, or, in other words, that makes it satisfiable. As an example, it is possible to ask via SMT if:

$$\exists x : (x \in \mathcal{D}) \Rightarrow x > 3.0 \tag{5.6}$$

that entails asking if "there exists an assignment of a value of $x \in \mathcal{D} \equiv \mathbb{R}$ such that $x > 3.0$ holds true". In this case the formula in (5.6) is *satisfiable*, for instance when $x = 4.1$. Conversely, a formula is *unsatisfiable* if there is no assignment of the variables that satisfies the formula.

Within the frame of SMT solving, *satisfiable* is to be distinguished from *valid*: a formula is *valid* only if it evaluates to true for every assignment of its variables (also called a tautology). The formula in (5.6) is not *valid* over the whole domain of the real numbers, as, for instance, $x = -1.5$ leads to the inequality not being satisfied. On the other hand, if the domain $\mathcal{D}$ is bounded, namely rather than considering $\mathcal{D} \equiv \mathbb{R}$, one considers $\mathcal{D} = [6.0, 8.5]$, formula (5.6) is *valid* over $\mathcal{D}$.

Formally, an SMT solver is an automated reasoning tool that verifies the satisfiability of a logical formula. The complication is that the decision problem for SMT formulas when applied to real numbers with generic nonlinear functions is undecidable[2] (e.g. when trigonometric functions are involved) [215]. In other words, it is demonstrated that there cannot be a procedure capable of always returning a correct answer within a finite number of iterations. Thus, relaxations were proposed to render the problem decidable, e.g. via the δ-weakening / δ-complete formulation [216, 217].

[2]Undecidable means that there is no sound and complete decision procedure to solve the decision problem. In turn, a procedure for a decision problem is sound if it returns valid when the input formula is valid. A procedure for a decision problem is instead complete if it returns valid when the input formula is valid, and if it always terminates [210].

Such a formulation is defined as follows:

Definition 6 (δ-complete [216]) *A decision procedure is δ-complete if for every SMT formula ϕ in a set S, with δ arbitrary such that $\delta \in \mathbb{Q}^+$, the procedure returns one of the following answers correctly:*
a) unsat *if ϕ is unsatisfiable;*
b) δ-sat *if ϕ^δ is satisfiable;*

where ϕ^δ denotes the δ-weakening of ϕ, that, in turn, encodes a numerical perturbation of ϕ: for instance, the δ-weakening of $x = 0$ (with $x \in \mathbb{R}$) is $|x| \leq \delta$ [215]. The effect of the δ-weakening relaxation is crucial as it transforms the SMT problem with generic nonlinear formulae from undecidable to a correctly solvable numerically-driven procedure, i.e. decidable. Additionally, if the formula ϕ is satisfiable, also ϕ^δ is satisfiable (but not vice versa). This leads to the following conclusion: when a formula is δ-sat, either it is in fact satisfiable, or its δ-weakening renders is satisfiable [215].

Several SMT solvers have been developed in recent times, with notable mentions being *Z3*, *CVC* and *dReal*. *Z3* (also known as the *Z3 Theorem Prover*), is an SMT solver developed by Microsoft (and released open source in 2015) [218]. *Cooperating Validity Checker* (CVC) [219] collects a family of tools (developed from 2002 onwards) used (in its latest versions) to proof defect-free software and correct functionalities of cloud services (e.g. Amazon Web Services).

dReal is an open-source SMT solver developed in 2013, specifically devised for SMT formulas over the real numbers encompassing nonlinear functions, proved to be δ-complete [216]. As this thesis specifically focuses on the problem of evaluating the satisfiability of generic nonlinear formulae (i.e. polynomial and transcendental) over the theory of real numbers, i.e. encompassing the verification of CLFs, dReal will be the selected tool further discussed hereby. When dReal returns *unsat*, dReal guarantees that the formula is unsatisfiable, namely dReal is a *sound* solver. On the other hand, when the answer is δ-*sat*, it returns a solution within a precision δ. In practical terms, a solution returned with δ-*sat* precision means that, rather than providing an exact value as the solution, a possible *box* is returned within which the solution resides.

Assume now that one wants to check whether formula (5.6) is valid over a domain

$\mathcal{D} = [2.0, 4.0]$, using dReal as SMT solver. Recalling that dReal is δ-complete, it is convenient to formulate the decision problem as the negation of formula (5.6): to confirm that (5.6) is valid, the negation must be unsatisfiable. In the specific example, the decision problem can be formulated as the negation of (5.6), namely:

$$\forall\, x : (x \in \mathcal{D}) \Rightarrow x \leq 3.0 \tag{5.7}$$

that entails asking if "for all the assignments of x in $\mathcal{D}$, x always satisfies $x \leq 3.0$". If dReal was to return unsat, no assignment of x such that $x \leq 3.0$ would exist, or, in other terms, $x > 3.0, \forall\, x \in [2.0, 4.0]$. Naturally, this is not the case, as the formula (5.6) is not valid and dReal would in actuality return an assignment of x that makes (5.7) satisfiable. By selecting an arbitrary value $\delta = 0.01$, dReal would terminate the decision problem with δ-sat, returning, for example $x = [2.39, 2.41]$ as an instance satisfying (5.7). This value of x is denoted as CE [217].

This uncertainty on the precision of the returned CE becomes of paramount importance in the vicinity of the origin, where dReal returns *spurious* CEs. A *spurious* CE denotes a false positive CE: even when an SMT query should return *unsat*, a fictitious CE is returned instead. Due to the numerical nature of the verification approach, the issue of the *spurious* CEs is inevitable.

This problem is clearly a limitation from a theoretical perspective, especially when verifying formulae where inequalities and strict inequalities bear a specific meaning. In the context of CLFs design, the equilibrium of a dynamical system is deemed stable if there exists a function that vanishes exactly at the origin with its Lie derivatives decreasing along the system trajectories. One could therefore claim that the employment of computer-aided methods introduces numerical errors that in turn render any verification effort to be unfeasible. Since any numerical error blurs the difference between strict and non-strict inequality, one can conclude that numerically-driven methods are not suitable for verifying these strict constraints [216]. However, even verifying that a system is stable within an arbitrarily small neighbourhood around the origin is of great importance in practical control applications [220].

To this aim, the original concept of stability in the sense of Lyapunov was ex-

tended and refined in later formulations. In practical terms, bounding a dynamical system to oscillate sufficiently near the target equilibrium is a desired outcome of a control system, tapping into the concept of *practical stability* introduced by LaSalle and Lefschetz [221]. A modification to the concept of practical stability is introduced in [220] as:

Definition 7 (ε-stability) *An equilibrium $\boldsymbol{x}^\star = 0$ is said to be ε-stable if, for any $R > \varepsilon$, there exists $r > 0$ such that if $\|\boldsymbol{x}_0\|_2 < r$, then $\|\boldsymbol{x}(t)\|_2 < R \; \forall t \in \mathbb{R}^+$,*

Proving the ε-stability of $\boldsymbol{x}^\star$ guarantees that, at steady-state, the state-space trajectories contract to $\|\boldsymbol{x}\|_2 \leq \varepsilon$ [220], with $\|\boldsymbol{a}\|_2 = \sqrt{a_1^2 + ... + a_n^2}$ denoting the 2-norm of vector $\boldsymbol{a}$. The difference with respect to the classical definition of stability in the sense of Lyapunov (Definition 5) lies in the fact that R is strictly bounded from below by a quantity ε, and no longer by 0. Further details regarding the interpretation of the ε-stability together with a numerical example are provided in Section 6.5.2.

It is worth reporting that, in opposition to the continuous time domain formulation of the NLC, as for the original case of LNN, recent interest focuses on the synthesis of CLF for discrete-time systems with formal certification of correctness. Associated literature aims at designing LFs for specific hybrid dynamical systems, i.e. piecewise linear systems (PWL) [138, 213]. Recent extensions focus on extending these works to generic discrete-time models with a Lipschitz continuous map [139]. This stream of research targeting discrete-time systems shares with the NLC the choice of the CEGIS architecture as the learning paradigm to synthesise CLFs. In place of the SMT solvers, these works use Mixed Integer Linear Programs (MILP)-based verifiers. As the majority of marine models and dynamics studies focus on continuous time models, this thesis hereby focuses solely on continuous-time methods.

5.2.3 Declaration of aim

Following recent developments in the NLC framework [134, 135, 137], the initial goal of the analyses proposed in this chapter is evaluating whether the NLC can systematically synthesise control laws for generic continuous-time nonlinear dynamical systems, while providing stability certificates via suitable CLFs. In the sections that fol-

low, the NLC method will be used to attempt stabilising complex unstable dynamics, constituting the foundation method for the subsequent extension to fault-tolerant control system design.

5.3 Neural Lyapunov Control method: initial assessment

In line with standard practice in control system design, new methods are conventionally verified using the inverted pendulum benchmark. The inverted pendulum dynamics can be written as follows:

$$\begin{cases} \dot{x}_1 = x_2 & \text{(5.8a)} \\ \dot{x}_2 = J_p^{-1}(mgl_p \sin x_1 - b_p x_2 + u) & \text{(5.8b)} \end{cases}$$

where x_1, x_2 represent the angular position and velocity of the swinging mass respectively, and u the control input, denoting a torque applied at the supporting joint. The scalar parameters b_p, l_p, m and J_p denote the drag coefficient, the length of the pendulum arm, the value of the lumped mass and the moment of inertia, respectively.

The inverted pendulum system is used as a first case study in the original NLC work [134] to demonstrate the convergence capabilities of the method. It is also employed in this work as a simple example to begin *unwrapping the black box*, i.e. to better understand how the method and the associated tool work *under the hood* and how the synthesis results are affected by changing the hyperparameters.

When devising ML-based methods, there is some difficulty in evaluating and comparing different techniques, as the choice of the hyperparameters can, and usually have, a significant effect on output performance. To this aim, the initial hyperparameters are setup as proposed in the code repository[3] associated with the original NLC work [134], and are then varied one at a time. Table 5.1 reports the hyperparameters initially proposed for the ANN architecture, illustrated in terms of layer size, presence of bias and selected activation function (σ) for the hidden layer and for the output layer.

[3] https://github.com/YaChienChang/Neural-Lyapunov-Control

Table 5.1: Inverted pendulum campaign – NLC architecture encompassing the input layer, one hidden layer and the output layer.

Parameter	Lyapunov ANN
Layer size	[2, 6, 1]
Bias	[Yes, Yes]
σ	[tanh, tanh]

The parameters characterising the dynamics are selected as proposed in the NLC work [134], namely as $m = 0.15$ [kg], $l_p = 0.5$ [m], $g = 9.81$ [m/s^2], $J_p = 0.0375$ [kg m^2] and $b_p = 0.1$ [(kg m^2)/s].

NLC makes use of a (linear) state-feedback control law, in this case initialised as: $u = -23.59x_1 - 5.31x_2$, obtained by solving a Linear Quadratic Regulator (LQR) optimisation problem. Additional key hyperparameters encompass:

1. the learning rate set as λ=0.01;
2. the initial dataset of cardinality $|S_I|$=500;
3. the dReal verification precision set as δ=0.01.

The general goal of the NLC method is to synthesise a control law that extends the ROA with respect to that obtained by means of a LQR solution. In this case scenario, the specific aim is to synthesise a control law to stabilise the pendulum in the upright position, together with a CLF to certify the stability of the equilibrium, while ensuring a ROA as large as possible. If such a solution is not obtained within 2000 learning iterations, the ANN weight is re-initialised and the learning is re-started a second time for up to another 2000 iterations. If a solution is still not obtained, the synthesis run is concluded and it is reported as unsuccessful.

A systematic investigation was carried out by varying the original NLC parameters, with the ranges of the training settings and of the hyperparameters selected in accordance with related literature [134, 135, 136, 222]. The ranges of explored values are reported in Table 5.2.

Additional key elements were investigated to further test the robustness of the NLC to changes in framework setup. These key elements were identified as the initial-

Table 5.2: NLC test campaign – varied parameters range.

Parameter	Min	Max
hidden layer size	2	250
learning rate	10^{-6}	10^{2}
$\lvert S_I \rvert$	50	2000
δ	10^{-4}	10^{-1}

isation of the control gain (supplied as LQR solution in the original formulation), and the number of learning iterations, as reported in Table 5.3.

Table 5.3: NLC test campaign – additional elements investigated.

Parameter	Option 1	Option 2
control gain initialised	Yes [134]	No
maximum learning iterations	2000 [134]	10^{5}

To understand the effects on the learning outcome, over 1800 tests were run by permuting different values of the parameters. The duration of the tests, ranging from a few seconds to a maximum of 10 hours, was as expected significantly affected by the overall number of learning iterations, initial dataset sample size and dReal precision. The key gained practical takeaways are summarised here:

1. if a test converges, it does so within a *relatively low number* of learning iterations. It follows that it is more beneficial to run several short tests rather than a single test of extended duration;

2. increasing the hidden layer size significantly speeds up the learning of the candidate CLFs, but prevents the verification procedure from converging. This finding is confirmed during later works [208], as 30 neurons was confirmed as the maximum practical value that was within the verification engine's processing capability;

3. setting a verification precision δ to *low values* (e.g. $\delta = 10^{-6}$) prevents the verification stage from converging; setting the δ value *too high* (e.g. $\delta = 10^{-1}$) renders the returned CEs uninformative as their associated uncertainty is overly excessive to be of practical use;

4. starting with a high $|S_I|$ value has a detrimental effect on the computational time of the learning stage and requires computational resources that are beyond the capability of a standard office laptop;

5. initialising the control gain with random values leads the learning framework to fail systematically.

This preliminary analysis returned practical insights relating to a sensible initialisation of the hyperparameters. More importantly, it highlighted the initialisation of the control gains as the key limiting factor of this framework.

It is worth noting that, at times, control functions were successfully synthesised. This was observed when the random control gains are *in the neighbourhood* of sensible state-feedback solutions. In the specific case that was analysed, an example of a favourable random initial solution is $u = -15.21x_1 - 8.72x_2$, which lead the framework to converge, while e.g. $u = -0.0023x_1 - 0.01x_2$ did not. This element is typically symptomatic of a low learning rate value selected. Naturally, one could try increasing the value of the learning rate. However, choosing learning rate values that are too high, lead the framework to fail to find a solution either.

The initialisation of the control gain becomes even more critical as a state-feedback control typically encompasses matrices $\boldsymbol{K}$ with high dimensions (in control of perfectly actuated single-bodied underwater vehicles, for instance $\boldsymbol{K} \in \mathbb{R}^{6\times 6}$, i.e. 36 scalar values [61]), more than just the two parameters of this preliminary case study. Moreover, if nonlinear control laws were to be devised instead, as per one of the stated aims of this thesis, finding sensible initialisation of the control gain would be even more complicated, if not too much of an endeavour to even consider the option for practical use. As the goal is to streamline the automatic synthesis of control laws, the sensitivity to the control gain initialisation was identified as a clear limiting factor that would need to be addressed.

The results derived from this preliminary analysis showed that despite the NLC method held significant promise when it comes to devising control laws while certifying the closed-loop stability, improvements are necessary to increase the applicability of the method. Specifically, as shown, the NLC method suffers when a favourable ini-

tialisation of the control gains is not provided, highlighting that further work is needed to improve the method such that control functions for a range of benchmark systems can be derived in a systematic manner.

Research question RQ2, initially formulated as:

Is the Neural Lyapunov Control method an effective approach to design stable control laws for nonlinear systems?

is thus updated as follows:

Can an upgraded version of the Neural Lyapunov Control method be devised to systematically design stable control laws for continuous-time nonlinear dynamical systems?

The following sections outline the tailored improvements that were devised to expand the learning ability of the framework, with a final comparison between the NLC and its upgraded version provided in Section 5.5.

5.4 Synthesis of Control Lyapunov Functions

The framework outlined in this section is logically built upon the original NLC method, with tailored modifications devised on all its constituent components. Minor modifications are reported in this section, while major elements are detailed in Section 5.5.

The proposed upgraded method employs sequential iterations of the Learner and the Falsifier in order to *i)* tune the control gains while computing a candidate CLF, and *ii)* formally verify the stability of the resulting closed-loop system. While the Learner trains the ANNs, the Falsifier oversees the formal verification of the candidate CLF and returns (if any) counterexamples to the Learner. The overall procedure is illustrated in Fig. 5.3, where η embeds the network hyperparameters, $V_C(x)$ and $u_C(x)$ represent the candidate CLF and the candidate control law, respectively, and S the training dataset, initially populated with points generated from a Uniform distribution from within a domain $\mathcal{D}$. Additionally, CE_{SMT} and CE_{DF} denote the CEs returned by the two verification modules that comprise the (Augmented) Falsifier, as it will be detailed in the following sections. The constituent elements reported in Fig. 5.3 are detailed in the following sub sections.

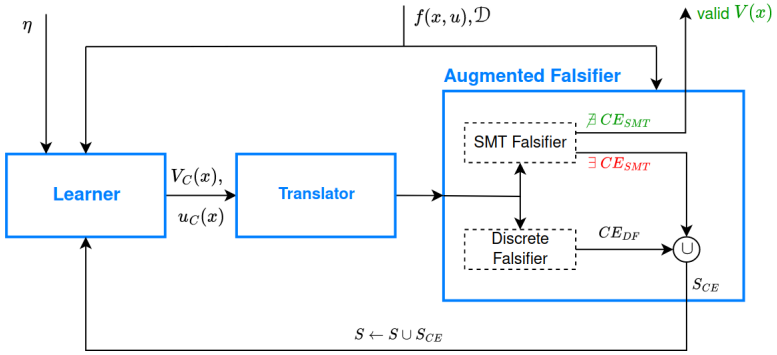


Figure 5.3: CEGIS learning method with Augmented Falsifier, © 2023 IEEE [22].

5.4.1 Learner

The Learner is the module tasked with training the ANNs to tune the control gains and to propose candidate CLFs.

Given a dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ (note that the dependency of $\dot{\boldsymbol{x}}$ on time is henceforth dropped to simplify the notation) and a target equilibrium $\boldsymbol{x}^\star$, the training procedure is initiated by using a (small) initial sample set $S = S_I$ composed of randomly selected states ($\boldsymbol{s_i}$) generated within a domain $\mathcal{D}$ (containing $\boldsymbol{x}^\star$). During each learning iteration, a cost function is evaluated and the ANN parameters (η), namely the weight and bias, are updated according to the Stochastic Gradient Descent (SGD) algorithm [223]. A detailed description of the cost functions is outlined in Section 5.4.1.2.

At the end of the training, this procedure returns a candidate control law and a candidate CLF that satisfy the Lyapunov conditions (5.4) over the *finite* sample set S, i.e. $V_c(\boldsymbol{s_i}) > 0$, $\dot{V}_c(\boldsymbol{s_i}, \boldsymbol{u_i}) < 0$, $\forall \boldsymbol{s_i} \in S$, where $\boldsymbol{u_i}$ represents the control law evaluated at sample $\boldsymbol{s_i}$.

Next, the selected ANN architecture, the loss function and the tuning of the learning parameters are discussed.

5.4.1.1 Control architecture

Following the seminal NLC method, several other works propose the synthesis of (C)LFs employing neural architectures, from shallow ANNs [135, 136] to deep networks [222, 137, 138]. Moreover, the selected activation functions vary significantly: e.g. ReLU, polynomial, tanh [134, 136], and softplus [222, 137]. Different to the NLC, the control architecture proposed in this work is composed of three networks embody-

ing the CLF, the linear and the *nonlinear control laws*, with the nonlinear control laws constituting the element of novelty. A user-defined Boolean parameter, denoted *control-branch training selector* (κ), is employed to train either one of the two control ANNs. The resulting architecture is illustrated in Fig. 5.4.

Naturally, a wider and deeper network ensures more flexibility and approximation power. However, a deeper network increases the computational training time and convolutes the symbolic expression of the CLF, which would typically present a challenge for the Falsifier. As a consequence of these conflicting requirements, the network architecture must be carefully selected, balancing flexibility against overly excessive expressions that cannot be verified. To this end, a network composed of two hidden layers was experimentally found to be a sensible trade off for benchmark dynamics of up to 12 dimensions, as it was shown capable of consistently delivering CLFs, in accordance with related findings [135, 137, 222, 138].

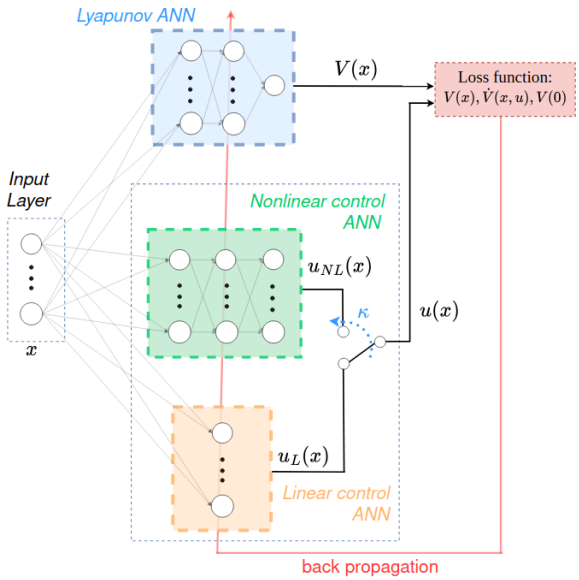


Figure 5.4: Augmented Neural Lyapunov Control architecture with Lyapunov ANN (blue box), nonlinear control ANN (green box) and linear control ANN (orange box). The red line represents the loss function back propagation, and κ the *control-branch training selector*.

Once the learning stage terminates, i.e. upon successful synthesis of a CLF, the control function is deployed in a classical closed-loop control scheme. An example of the closed-loop scheme is reported in Fig. 5.5.

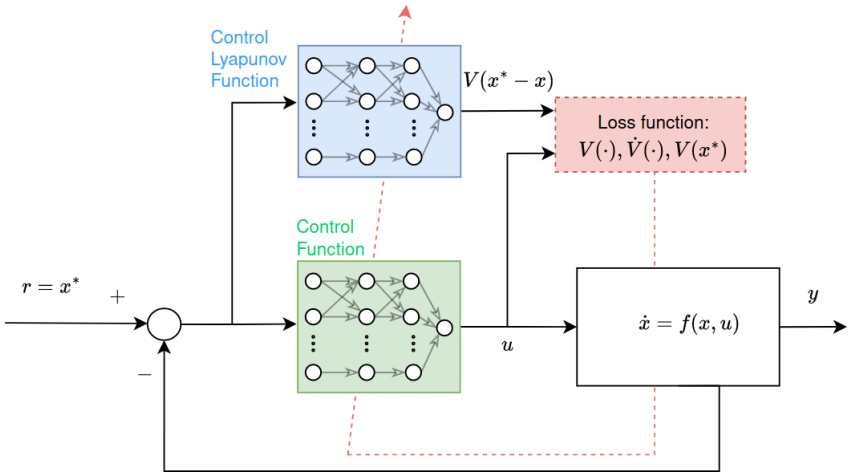


Figure 5.5: Control architecture upon deployment in closed-loop applications, showcased when a nonlinear control function is employed, © 2024 Elsevier [23].

5.4.1.2 Loss functions

The loss function has a twofold objective in that it is required to drive the candidate CLF to satisfy the Lyapunov constraints, while maximising the ROA. As such, a modification to the original NLC loss function was devised. The *Empirical* Lyapunov Risk Loss (L_{ELR}) is introduced as:

$$L_{ELR} = \alpha_1 \sum_{i=1}^{|S|} \mathcal{R}(-V(\boldsymbol{s_i})) + \alpha_2 \sum_{i=1}^{|S|} \mathcal{R}(\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i})) + \alpha_3 V(0)^2 + \alpha_4 \frac{1}{|S|} \sum_{i=1}^{|S|} (\|\boldsymbol{s_i}\|_2 - \alpha_{ROA}\, V(\boldsymbol{s_i}))^2 \quad (5.9)$$

where $\mathcal{R}(a) = ReLU(a) = \max(0, a)$ for a general input a, and where $\alpha_1, ..., \alpha_4, \alpha_{ROA}$ are tuning parameters and $|S|$ is the cardinality of the dataset S (containing samples $\boldsymbol{s_i}$), and where $\|\boldsymbol{s_i}\|_2$ denotes the 2-norm of the sample $\boldsymbol{s_i}$. The first three terms of Eq. (5.9) account for the theoretical Lyapunov conditions in Eq. (5.4), while the final term's function is maximising the size of the ROA [134].

By weighting the first three terms of the loss function higher than the fourth term, the ROA tuning term can be considered as an additional side feature. The fourth component induces a parabolic shape of V, with α_{ROA} regulating the function steepness. The effect of tuning α_{ROA} is shown in Fig. 5.6, with $\overline{\gamma}$ denoting the upper boundary of

the domain. Assuming that, ideally, at the end of the training:

$$\|\boldsymbol{s}\|_2 - \alpha_{ROA}\, V = 0 \tag{5.10}$$

holds true, the effect of different tuning values can be illustrated by selecting two numerical values:

(i) case $\alpha_{ROA} = \overline{\gamma}$: from Eq. (5.10) at the edge of the domain (i.e. $\|\boldsymbol{s}\|_2 = \overline{\gamma}$), it holds: $\overline{\gamma} - \overline{\gamma}\, V = 0 \rightarrow V = 1$.

(ii) case $\alpha_{ROA} = 2\overline{\gamma}$: at the edge of the domain it holds: $\overline{\gamma} - 2\overline{\gamma}\, V = 0 \rightarrow V = \frac{1}{2}$.

Overall, the lower the α_{ROA} value, the steeper the CLF. The presence of the forth term in Eq. (5.9) has the effect of shaping the CLF to approximate a parabola (or a paraboloid of revolution for n-dimensional systems), as it will be shown later by numerical analysis in Section 6.8.1.

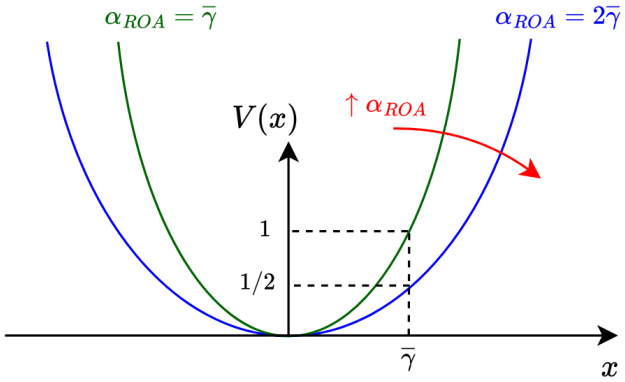


Figure 5.6: Effect of tuning α_{ROA} in the loss function (5.9), © 2024 Elsevier [23].

When the fourth term in not employed, i.e. when maximising the ROA is not a priority, the values of α_1, α_2 and α_3 can be set up to 1.0, and $\alpha_4 = 0.0$, without any loss of generality (as shown in the case studies of Section 5.6). Additionally, when $V(0) = 0$ holds true by construction, such as when the Lyapunov ANN has no bias and the activation function is zero in zero, α_3 can also be set to zero.

In parallel to Eq. (5.9), to monitor whether the Lyapunov constraints alone are

satisfied, the *Strict Lyapunov Risk Loss* is introduced (L_{SLR}):

$$L_{SLR} = \sum_{i=1}^{|S|} \mathcal{R}(-V(\boldsymbol{s_i})) + \sum_{i=1}^{|S|} \mathcal{R}(\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i})) + V(0)^2 \quad (5.11)$$

which returns a positive value whenever any point in the dataset S violates Eq. (5.4). With the logic further detailed in Section 5.5.1, this function is used to reduce the computational burden of the procedure by executing callbacks to the Falsifier only when the L_{SRL} is equal to zero.

In the following Section 5.4.2, details are provided as regards to how to evaluate the quantities $V(\boldsymbol{s_i})$ and $\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i})$ in (5.9)-(5.11).

5.4.2 Translator

Once a candidate pair (V_c, u_c) is obtained, a corresponding symbolic expression needs to be passed to the Falsifier. This step is carried out by a module typically referred to as the *Translator* [136], that is derived here in the most general formulation for a feedforward ANN encompassing bias. First the output of a feedforward ANN layer i is recalled as:

$$\boldsymbol{z_i} = \sigma_i(\boldsymbol{W}_i \, \boldsymbol{z}_{i-1} + \boldsymbol{B}_i), \; i = 1, \ldots, k \quad (5.12)$$

where $\boldsymbol{z}_{i-1}$ represents the input to the i-th layer, and $\boldsymbol{W_i}, \boldsymbol{B_i}, \sigma_i$ are the corresponding weight, bias and activation function, respectively, and k is the total number of ANN layers.

The symbolic expression of a CLF $V(\boldsymbol{x})$ can thus be obtained by a forward pass of the Lyapunov network as:

$$V(x) = \sigma_k(\boldsymbol{W}_k \, \boldsymbol{z}_{k-1} + \boldsymbol{B_k}). \quad (5.13)$$

Next, the symbolic expression of a Lie derivative needs to be derived. Formally, a Lie derivative is defined as:

$$\dot{V} = \langle \boldsymbol{\nabla(V)_x}, \boldsymbol{f(x,u)} \rangle \quad (5.14)$$

with:

$$\nabla(V)_{\boldsymbol{x}} \coloneqq \frac{\partial V}{\partial \boldsymbol{x}} = [\frac{\partial V}{\partial x_1} \cdots \frac{\partial V}{\partial x_n}]^T. \tag{5.15}$$

Next, the gradient $\frac{\partial V}{\partial \boldsymbol{x}}$ can be evaluated as the following chain rule:

$$\frac{\partial V}{\partial \boldsymbol{x}} = \frac{\partial z_i}{\partial \boldsymbol{z}_{i-1}} \frac{\partial \boldsymbol{z}_{i-1}}{\partial \boldsymbol{z}_{i-2}} \cdots \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} \tag{5.16}$$

with $\boldsymbol{z_0} = \boldsymbol{x}$ and $z_i = V$. The partial derivative of a generic layer i with respect to the previous layer can be evaluated by means of (5.12) as:

$$\frac{\partial z_i}{\partial \boldsymbol{z}_{i-1}} = \frac{\partial \sigma(\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})}{\partial \boldsymbol{z}_{i-1}} = \frac{\partial \sigma(\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})}{\partial (\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})} \frac{\partial (\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})}{\partial \boldsymbol{z}_{i-1}}. \tag{5.17}$$

It is thus possible to compute the first factor of Eq. (5.17) as:

$$\frac{\partial \sigma(\boldsymbol{W_i}\, \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial (\boldsymbol{W_i}\, \boldsymbol{z_{i-1}} + \boldsymbol{B_i})} = \text{diag}[\sigma'(\boldsymbol{W_i}\, \boldsymbol{z_{i-1}} + \boldsymbol{B_i})], \tag{5.18}$$

where diag$[\boldsymbol{a}]$ represents a diagonal matrix whose entries are the elements of vector $[\boldsymbol{a}]$ and with g_i denoting the number of neurons of the i-th layer, such that $\boldsymbol{W_i} \in \mathbb{R}^{g_i \times g_{i-1}}$, $\boldsymbol{B_i} \in \mathbb{R}^{g_i}$ and $\boldsymbol{z_i} \in \mathbb{R}^{g_i}$.

Next, the second factor of Eq. (5.17) can be calculated as:

$$\frac{\partial (\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})}{\partial \boldsymbol{z}_{i-1}} = \boldsymbol{W_i} \tag{5.19}$$

Overall, Eq. (5.17) can be expressed as:

$$\frac{\partial \sigma(\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})}{\partial \boldsymbol{z}_{i-1}} = \text{diag}[\sigma'(\boldsymbol{W_i}\, \boldsymbol{z}_{i-1} + \boldsymbol{B_i})]\, \boldsymbol{W_i} \tag{5.20}$$

To conclude, the Lie derivative of $V(x)$ is computed as:

$$\dot{V} = \left(\prod_{i=1}^{k} \text{diag}[\sigma'_{k-i+1}(\boldsymbol{W}_{k-i+1}\, \boldsymbol{z}_{k-i} + \boldsymbol{b}_{k-i+1})]\, \boldsymbol{W}_{k-i+1} \right) \boldsymbol{f(x,u)} \tag{5.21}$$

where $\boldsymbol{f(x,u)}$ embeds the symbolic expression of $\boldsymbol{u(x)}$, which is in turn obtained via

Eq. (5.12).

5.4.3 SMT Falsifier

The Falsifier module is designed to formally verify whether a candidate function $V_c(x)$ is in fact an unconditionally correct CLF. This statement corresponds to evaluating the expression:

$$\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{D}) \Rightarrow (V_c(\boldsymbol{x}^\star) = 0 \wedge V_c(\boldsymbol{x}) > 0 \wedge \dot{V}_c(\boldsymbol{x}, \boldsymbol{u}) < 0) \tag{5.22}$$

This expression can be simplified: recalling Eq. (5.13), it is possible to note that $V(\mathbf{0}) = 0$ holds true if $\boldsymbol{b_i} = 0$ and activation functions such that $\sigma_i(0) = 0$ for all i are chosen. The falsification step is thus formulated as the logical negation of Eq. (5.22), meaning that the SMT solver searches for a solution of:

$$\exists \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{D}) \Rightarrow (V_c(\boldsymbol{x}) \leq 0 \vee \dot{V}_c(\boldsymbol{x}, \boldsymbol{u}) \geq 0) \tag{5.23}$$

It should be noted that condition $V(\boldsymbol{x}^\star) = 0$ is omitted from the Falsifier constraints Eq. (5.22). This condition is verified separately since it simply represents a pointwise evaluation.

When solving Eq. (5.23), if there exists such a point $\boldsymbol{x}$, the candidate $V_c(\boldsymbol{x})$ does not satisfy the constraints Eq. (5.4); hence, V_c is discarded and $\boldsymbol{x}$ is returned to the Learner as a CE.

Additionally, it is recalled that dReal is a δ-complete solver. This ensures that whenever V_c is deemed valid, then V_c is a CLF. However, dReal may return spurious counterexamples within a δ-error. While this may generate an infinite number of loops between Learner and Falsifier, it does *not* impair the correctness of the proposed procedure.

The δ precision is problematic when checking the constraints (5.4) within a neighbourhood of the origin. The exclusion of a small neighbourhood of the origin from the verification step was proposed as mitigation [134] and it is adopted in this work, introducing a lower boundary on the domain of validity of the CLF. In the case of a spherical domain, the domain is defined as: $\varepsilon \leq \|\boldsymbol{x}\|_2 \leq \overline{\gamma}$, for given radii ε and $\overline{\gamma}$, with

$\varepsilon, \overline{\gamma}$ scalar design parameters. As such, this method guarantees the ε-stability of the underlying model [220], formally guaranteeing that the state remains bounded within ε upon convergence. In other words, at steady-state the state-space trajectories are guaranteed to contract to $\|\boldsymbol{x}(t)\|_2 \leq \varepsilon,\ t \to \infty$. Further details regarding the meaning of the ε-stability are provided in Section 6.5.2, together with numerical examples and illustrations.

In line with the NLC framework, in this work, *dReal* is selected as SMT solver to evaluate the SMT query (5.23) owing to its capability to handle polynomials, trigonometric and exponential expressions.

5.5 Augmented Neural Lyapunov Control: tailored improvements

In this section, the specific NLC limitations, identified in Section 5.3, are discussed in more detailed and mitigation strategies are proposed. Four main elements of improvements will be discussed: the generation of CEs, the selection of the learning rate, the composition and cardinality of the dataset and algorithmic logic refinements. The upgraded procedure devised is henceforth identified as *Augmented* NLC (Augmented Neural Lyapunov Control (ANLC)).

5.5.1 Augmented Falsifier

At each callback of the SMT Falsifier, one single CE box is identified and a point cloud within such box is returned. The SMT callback process is generally time-consuming, causing a bottleneck in the procedure.

It should be noted that the bounded (i.e. when considering a finite-spaced domain) SMT problem with exponential and trigonometric functions over the Reals is nondeterministic polynomial time (NP)-complete [215], while the problem, for generic nonlinear functions, is NP-hard instead [134].

A decision problem belongs to the complexity class NP if, given an input u, it is possible to verify that u is an instance satisfying the problem in polynomial time [224]. In other words, NP is the set of problems that can be efficiently verified [225]. Proofs

about whether efficient algorithms exist to verify NP-complete problems are currently the object of extensive research effort, but it is believed that an NP-complete problem might only have non-polynomial verification solutions, i.e. non-efficient solutions [224].

NP-complete problems are the more complex problems in NP, and NP-hard are at least as hard as the hardest problem in NP. These theoretical insights point towards an intrinsic limitation of the method linked to the computational burden required during the verification stage.

To overcome this computational bottleneck, a numerical unit, called Discrete Falsifier (DF), is introduced to attempt an initial generation of CEs before the SMT call. In more detail, the DF is designed to minimise the number of callbacks to the SMT Falsifier, while the SMT Falsifier is tasked with only providing a certificate of correctness at the end of the learning procedure, rather than being employed recursively to provide CEs.

With such architecture to generate CEs in place, the overall learning loop evolves as follows. Starting from an initial dataset S, the training attempts to minimise L_{ELR}, with the training ceasing when the condition $L_{SRL} = 0$ is attained. When this occurs, the candidate CLF has satisfied the constraints (5.4) over all samples. Next, the DF is called. The domain is discretised over a grid specified by user-defined precision, and the values of $V(\boldsymbol{s_i})$ and $\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i})$ are evaluated over the discrete points. If points violating Eq. (5.4) are identified, these points are stored in the set CE_{DF} and added to the dataset and the training resumes. If the cardinality of CE_{DF} exceeds a threshold ζ_{DF}, a randomly selected subset of cardinality ζ_{DF} is added to the dataset. Adding a limited number of points is more in line with the CEGIS paradigm, where the solution is recursively refined, as opposed to being excessively updated at each learning iteration. In those particular cases where CEs are not generated, the SMT is invoked to *formally* verify the correctness of the candidate. Finally, if the SMT locates a CE, a set containing ζ_{SMT} new points is added to the dataset (defined as S_{CE} in Fig. 5.3), otherwise $V_c(x)$ is formally verified to be a CLF.

5.5.2 Network-specific Learning Rate and Scheduler

In line with previous observations, the learning rate has a significant impact on the learning ability of ANNs [226, 227]. Low values of learning rate often lead to the loss function getting stuck in local minima, while high values typically result in unstable training [228].

In the original NLC work [134], both the Lyapunov and the control ANNs use a single learning rate value; following the preliminary investigation reported thus far, this design choice is believed to leading the training to at times becoming stalled. As a consequence, in this work, the Lyapunov and control learning rates of the Lyapunov and control ANNs are separated, and are referred to as λ and λ_c, respectively.

Additionally, to resemble the re-initialisation of the ANN weights carried out in the NLC algorithm, a cosine annealing scheduler is added [229]. This scheduler cyclically oscillates the learning rate between a minimum and a maximum value, while briefly undertaking both conservative (low values) and aggressive (high values) learning rate tuning. This element is introduced to avoid the requirement for hard reset of the ANN weights at pre-defined intervals, mitigating the risk of losing the learning progress in the midst of the training process.

5.5.3 Counterexample Selection

The CEGIS paradigm gives rise to a dynamic dataset that increases in size as successive CEs are returned by the Falsifier. As the CEGIS loops progress, the algorithm slows down, since the computational burden derived from the training of the ANNs is impacted by the size of the dataset, in addition to other parameters. Note that, experimentally, the SMT often returns similar CEs at successive iterations, i.e. a significant portion of the dataset becomes clustered in a small region of the state-space. This event is denoted *counterexample overfitting*, as illustrated in Fig. 5.7.

In this work, a device called selective sliding window (SSW) is introduced as a means of mitigation. The initial dataset S_I, which contains samples scattered over the entire domain, is held unchanged. A sliding window is applied to the counterexample dataset S_{CE}. When a prescribed maximum dataset size $|S| = S_{max}$ is attained, the least recent CEs are deleted in order to keep the dataset cardinality bounded, as illustrated in

Fig. 5.8. This element limits the risk of CE overfitting – already identified as a possible cause of reduced algorithm efficiency [138], while the fixed dataset dimension prevents the training performance from deteriorating over successive loops.

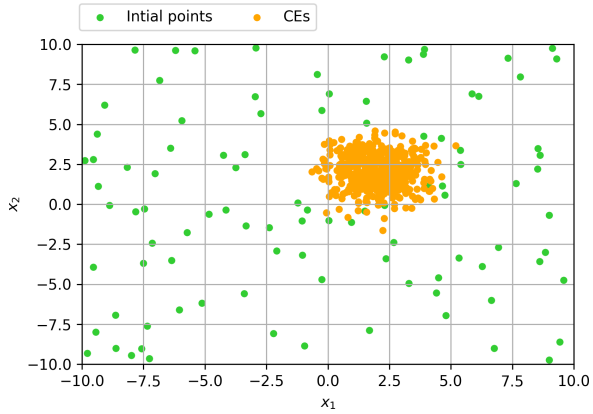


Figure 5.7: Dataset with *initial points* (S_I) and clustered counterexamples, © 2023 IEEE [22].

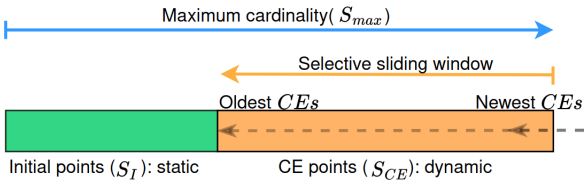


Figure 5.8: Dataset with *selective sliding window* logic, with $S = S_I \cup S_{CE}$, © 2023 IEEE [22].

5.5.4 Algorithm and software

The algorithm starts by generating a training dataset S composed of random samples $\boldsymbol{s_i}$ from a uniformly distributed hypercube of side length $\overline{\gamma}$. If the linear control branch has been selected, the linear control weight can be optionally set to an initial value (q^I). Next, at each training iteration the *Learner* gradually minimises the loss L_{ELR} by updating the ANN weights through a SGD step. In any training iteration, the current values of the ANNs weight and bias are collected in a vector η, defining V_η and u_η. Iterations of the learning steps continue until the Lyapunov conditions are verified $\forall \boldsymbol{s_i} \in S$, i.e. when $L_{SLR} = 0$. Once $L_{SLR} = 0$ holds true, the candidate CLF (V_η^S) and the Lie derivative ($\dot{V}_\eta^S$) are symbolically obtained by means of the *Translator*. The *Discrete Falsifier* is thus invoked to find CEs (CE_{DF}) over a prescribed discretisation of the domain $\mathcal{D}$. If CEs are found, these are added to S and the learning step is restarted. If no CEs are obtained, the *SMT Falsifier* is tasked with verifying the candidate CLF

(V_η^S) over the dense domain of the Reals. Next, if a CE is obtained via SMT-solving (CE_{SMT}), the CE_{SMT} is added to S and the learning step restarted. Otherwise, the candidate CLF is formally verified to be correct (the SMT callback terminates with `unsat`), and the algorithm returns the synthesised control law.

The schematic algorithm of the ANLC method is reported in Algorithm 1, where the `timeout` defines a time threshold within which the SMT Falsifier is required to compute the CEs. If such a timeout value is reached, the verification procedure is stopped and the training is flagged as unsuccessful.

The core modules of the library are structured as reported in Fig. 5.9. The *main* file retrieves the values of the hyperparameters and the description of the dynamical system from the *configuration* file. Next the callback to the CEGIS loop is performed and the training stage is set up. Within the CEGIS file, the ANNs are instantiated, and the training stage started. Next, the learning is iterated until a candidate CLF is obtained. The candidate CLF is then passed to the *Falsifier* to verify its correctness. Upon termination, the postprocessing module is called, tasked with producing the plots and saving the logs of the training run. Finally, optional closed-loop testing runs are performed to confirm successful synthesis of the control law.

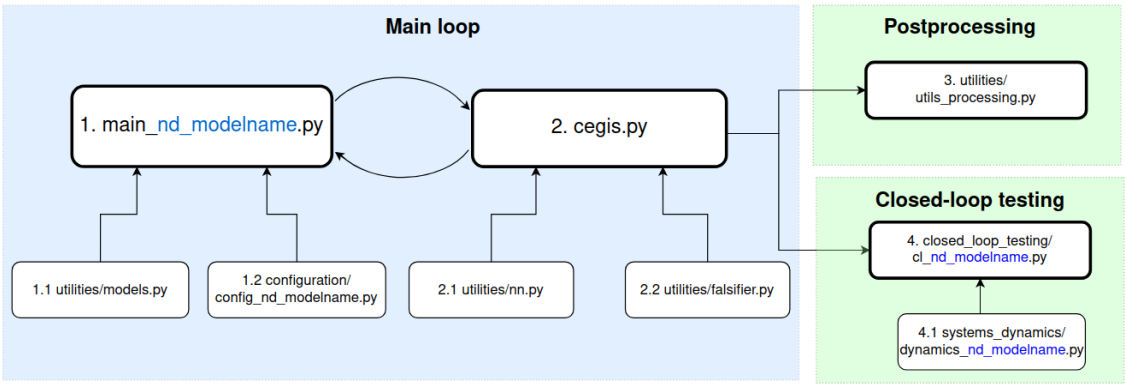


Figure 5.9: Code modules designed.

The *call graph* representing the flow of the subroutines involved in the run of a template 2-dimensional system is illustrated in more detail in Fig. 5.10[4]. The software tool employs either *Python 3.7*, *dReal 4.21* and *PyTorch 1.4*, or *Python 3.9*, *dReal 4.21*

[4]The call graph is generated with the *code2flow* library (https://github.com/scottrogowski/code2flow)

Algorithm 1 Augmented Neural Lyapunov Control

function LEARNER(S, f, ANN_η)
  repeat
    $V_\eta(x), u_\eta(x) \leftarrow ANN_\eta(x)$ ▷ ANN forward pass
    $\dot{V} \leftarrow$ Translator
    Compute loss L_{ELR}, L_{SLR}
    $\eta \leftarrow \eta - \nabla_\eta L_{ELR}$ ▷ Update weights via SGD
  until ($L_{SLR} > 0$)
  return $V_\eta(x), u_\eta(x)$

function DISCRETE FALSIFIER($V_\eta^S, \dot{V}_\eta^S, \mathcal{D}, \zeta_{DF}$)
  Discretise $\mathcal{D}$
  Numerically evaluate ($V_\eta^S \leq 0, \dot{V}_\eta^S \geq 0$)
  $CE_{DF} \leftarrow$ Violations points (max size ζ_{DF})
  return CE_{DF}

function SMT FALSIFIER($V_\eta^S, \dot{V}_\eta^S, \mathcal{D}$)
  Using SMT solver to verify conditions
  return `unsat` or CE_{SMT}

function MAIN()
  Input: f, S, ζ_{DF}, ζ_{SMT}, $\mathcal{D}$, ε, $\overline{\gamma}$, η, λ, λ_c, ϕ, $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_{ROA}$, optional control gains (q^I)
  Initialise ANN size (optional: initialise linear control gains with q^I)
  repeat
    if ($N \geq S_{max}$): Apply sliding window
    $V_\eta(x), u_\eta(x) \leftarrow$ LEARNER(S, f, ANN_η)
    Compute symbolic values $f_\eta^S, u_\eta^S, V_\eta^S, \dot{V}_\eta^S$
    if ($L_{SLR} == 0$) **then**
      $CE_{DF} \leftarrow$ DISCR. FALSIFIER($V_\eta^S, \dot{V}_\eta^S, \mathcal{D}, \zeta_{DF}$)
      if CE_{DF} is `None` **then**
        $CE_{SMT} \leftarrow$ SMT FALSIFIER($V_\eta^S, \dot{V}_\eta^S, \mathcal{D}$)
      $S_{CE} \leftarrow (CE_{DF} \cup CE_{SMT})$
      if (not `unsat`**)**: $S \leftarrow (S \cup S_{CE})$
  until (`unsat` **or** `timeout`)

and *PyTorch 1.7*. Installation instructions designed to enhance portability over different Operating Systems and a user-guide are provided within the associated repository page at https://github.com/grande-dev/Augmented-Neural-Lyapunov-Control, under the *Installation* section. Different approaches to install the software tool are made available, such as installing it at system level, or within an *Anaconda* environment, or within a

virtual environment.

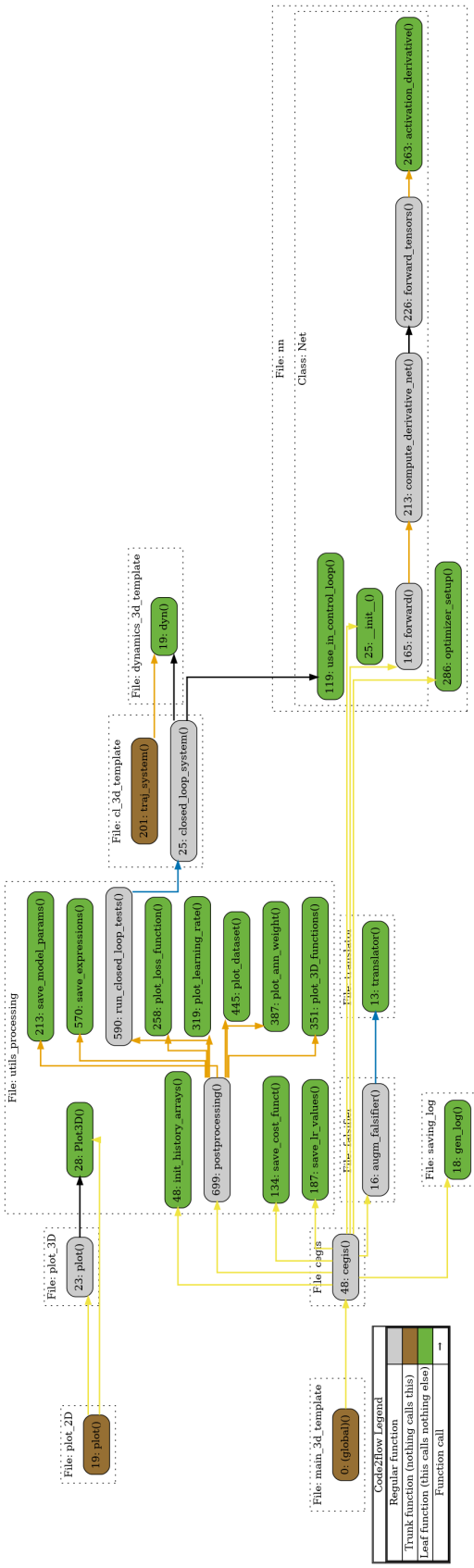


Figure 5.10: ANLC library call graph.

5.5.5 ANLC upgrades summary

Having now completed the investigation into the NLC algorithmic details, the different learning capabilities of the NLC and ANLC can be narrowed down to five factors. The improved elements are summarised in Table 5.4, which illustrates the aspects of the NLC that required further refinement, together with the corresponding selected mitigating measures.

In summary, the NLC requires improvements to be capable of generating control laws systematically and automatically without requiring ad hoc initialisation. The elements in need of improvements are highlighted as: inefficient CEs generation, CE overfitting, algorithmic inefficiencies, dataset size and sensitivity to learning rate. All these elements lead to the training process getting stuck in local minima or to learning instabilities, to a continuously growing dataset and to the possibility of infinite CE generation loops. The ANLC tackles the issues mentioned one by one, significantly expanding the learning capabilities such that both CLFs and the control functions can be consistently learnt altogether.

A quantitative comparison of the performance of the NLC and the ANLC methods is provided in the following section.

5.6 Numerical Evaluation

In this section, following the proposed methodological upgrades, the original NLC and the new ANLC methods are compared with respect to the initial benchmark of the inverted pendulum system. Next, the ANLC is challenged over a Lorenz attractor system, which has been chosen as a more advanced benchmark. The results reported in the following section highlight the improvements gained when compared to the original approach. All the training runs illustrated are executed on a standard low-specification office laptop computer with 8 Central Processing Units (CPUs) at 1.90GHz and 16 GB of Random Access Memory (RAM), no Graphics Processing Unit (GPU).

5.6.1 Control system without initialisation

The problem of the stabilisation of an inverted pendulum is once again selected as the comparison benchmark test, following the analysis of Section 5.3. In the original NLC

Table 5.4: Breakdown of the ANLC improvements.

Element	NLC	ANLC	ANLC improvement interpretation
Falsifier as CEs generator	The SMT Falsifier returns one CE at each callback.	The Augmented Falsifier returns several CEs at each callback.	The improvement is twofold: a) the CE overfitting issue is reduced, as the points added to the dataset are more likely to be spread over the full search domain; b) the algorithm learns much faster (as multiple CEs are added every time), reducing the risk to hit a *timeout*.
Algorithmic logic	The Falsifier is called at fixed intervals.	The Augmented Falsifier is called only when the loss function (L_{SLR}) is equal to zero.	The improvement is twofold: a) adding new CEs at pre-defined time often leads to two successive callbacks of the SMT Falsifier returning the same CE, as the violation identified can still be located in the same subspace of $\mathcal{D}$. This logic is inefficient, as the dataset grows in size by adding multiple instance of the same CE, progressively slowing down the learning and without bringing any added benefits; b) as the SMT Falsifier callback represents the computational bottleneck of the *Lyapunov Control* method, our approach invokes the CLF verification only when strictly necessary, further reducing the overall learning time and the risk to hit the *timeout* threshold.
Dataset composition	It can cluster around single CE.	The SSW limits clustering by preserving points spreaded out over $\mathcal{D}$.	The risk of overfitting clusters of CE is reduced.
Dataset size	It grows indefinitely.	The cardinality is capped at a specified value (S_{max}).	The ANLC algorithm does not slow down once $\lvert S\rvert > S_{max}$ as the SSW starts removing old CEs points. A large dataset can render this procedure unusable, especially when higher dimensional systems are considered, thus $\lvert S\rvert$ is kept bounded via the SSW.
Learning rate	Is fixed.	Is varied with a scheduler.	The scheduler allows to explore a wider range of learning rate values, preventing the loss function getting stuck in local minima (if the learning rate is too low), and reducing the risk of training instability (if the learning rate is too high).

Table 5.5: ANLC: inverted pendulum campaign parameters.

ε [deg]	$\overline{\gamma}$ [deg]	δ	$\lvert S_I \rvert$	λ	λ_c
5.73	343	10^{-6}	500	[0.0, 0.01]	[0.0, 1.0]

work, the control ANN is initialised with a pre-computed LQR law. When the control weight is not initialised, the NLC is found to not be able to systematically synthesise CLFs. In this section, the effect of not initialising the control ANN weight is quantified and synthesis statistics are reported.

Table 5.5 details the test parameters, chosen according to relevant literature [134, 137]. Note that as ANLC uses two distinct learning rates for the control and Lyapunov ANNs, the values of λ and λ_c are reported in terms of the minimum and maximum values of the cosine annealing scheduler. To allow for a fair comparison, in the ANLC tests, the loss function gains have intentionally not been fine tuned (i.e. $\alpha_{1,2,3} = 1$ and $\alpha_4 = 0$), while the values of the NLC loss function are selected as reported in the associated literature [134]. Each hidden layer is composed of 10 neurons in accordance with the considerations provided in Section 5.3. The selected activation functions are linear for the first hidden layer, quadratic for the second hidden layer and linear for the output layer.

To compare the performance of the NLC and the ANLC algorithms, a simulation campaign composed of six case scenarios was designed, with the results reported in Table 5.6. For each scenario, 50 tests were run, where each run was started from a different *seed*. A seed is an integer value that is used to initialise the values of free (random) parameters. Within the context of the ANLC, the choice of the seed determines the initial (random) values of the weights and bias on the ANNs, and the initial (random) sample composing the training set. By employing the same value of the seed over successive training runs, one can ensure reproducible results, as the ANNs are guaranteed to have the same initial values and the training sets comprise the same samples. The results are presented in terms of iterations required for the algorithms to converge and reported as per mean value (mean) and standard deviation (std.). A test was defined as having *converged* when a stabilising control law and a CLF were obtained within the prescribed threshold of the maximum number of learning

iterations.

An initial NLC comparison scenario was designed to verify the necessity to periodically re-initialise the ANNs weight, listed as Scenarios 1a and 1b in Table 5.5. Each test was run for up to a maximum of 2×1000 iterations, value selected in accordance with the original NLC work [134]. If the test had not converged to a solution within the first 1000 iterations, the weights of the ANNs were re-initialised and the test was run once more for up to 1000 more iterations. The results showed that when the control network was not initialised with a LQR solution, the convergence percentage of the algorithm drops from 78% (Scenario 1a) of successful syntheses to 0% (Scenario 1b). These two scenarios underlined that the weight re-initialisation (after 1000 learning iterations) is not a key element of the algorithm, while the result clearly highlights that the control initialisation is a fundamental component of the NLC method.

The next two scenarios, denoted as Scenarios 2 and 3 in Table 5.5, were designed to compare the performance of the ANLC algorithm against that of the NLC algorithm, with further focus on the control weight initialisation. Each test comprised a maximum of 1000 iterations to resemble Scenario 1 without weights re-initialisation. Once more, it was observed that the performance of the NLC algorithm reduced when the control network was not initialised, with convergence reducing from from 52% (Scenario 2a, with initialisation) to 0% (Scenario 3a, without initialisation). The ANLC method outperformed the NLC when the control was initialised returning an 84% convergence (Scenario 2b, with initialisation) vs. a 52% convergence (Scenario 3b, without initialisation). This result was further reinforced when comparing the results for Scenario 1a to those for Scenario 2b the convergence rate increased from 78% to 84% even though in Scenario 2b the weights were not reinitialised. It is worth noting that, when comparing Scenario 2b with Scenario Scenario 1a, ANLC was run for half of the iterations with respect to the NLC, while still delivering a higher successful synthesis rate.

The CLF synthesised in one of the ANLC tests, obtained by selecting $V = \boldsymbol{W}_3(\boldsymbol{W}_2(\boldsymbol{W}_1\boldsymbol{x}))^2$ as the architecture, is illustrated in Fig. 5.11. The associated CLF symbolic expression is reported in Appendix B. The corresponding Lie derivative function, is depicted in Fig. 5.12, with the symbolic expression reported in Appendix C.

Table 5.6: Sensitivity to control weights initialisation, © 2023 IEEE [22].

Scenario	Method	Control pre-initialised	Weights re-initialised	Iterations mean(std.)	Converged tests [%]
1a	NLC	Yes	Yes	1107(746)	78
1b	NLC	No	Yes	2000(0)	0
2a	NLC	Yes	No	698(318)	52
2b	ANLC	Yes	No	478(148)	84
3a	NLC	No	No	1000(0)	0
3b	ANLC	No	No	464(373)	60

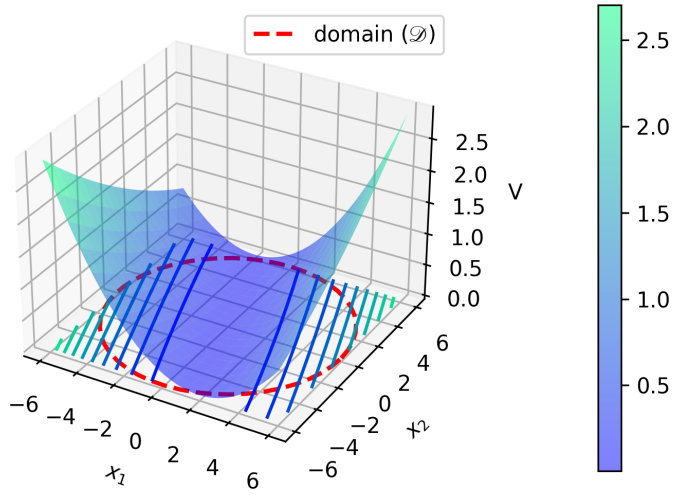


Figure 5.11: CLF inverted pendulum, © 2023 IEEE [22].

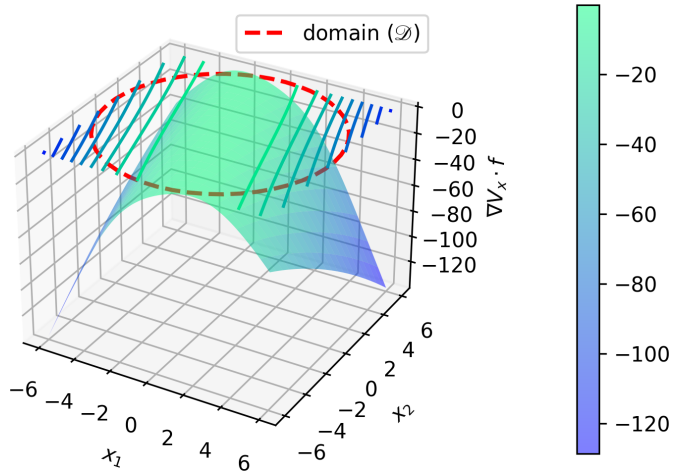
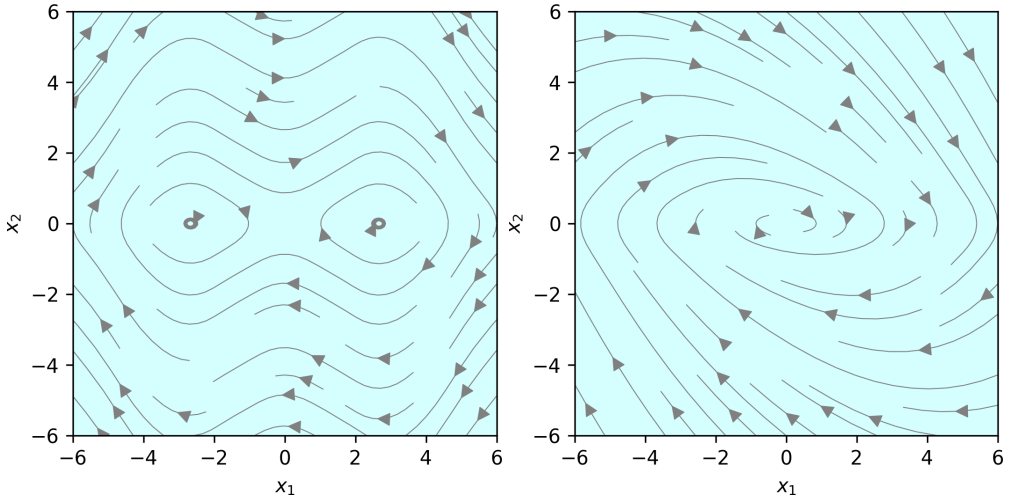


Figure 5.12: Lie derivative inverted pendulum, © 2023 IEEE [22].

Finally, the phase planes that are associated with the open-loop (no control law) and closed-loop inverted pendulum systems are compared. Fig. 5.13a reports the open-loop phase plane, with two visible stable foci, associated to the pendulum in the down-

ward position ($\boldsymbol{x}^{\star}_{1,2} = [\pm\pi, 0]$). Fig. 5.13b reports the closed-loop system following the synthesis of a control law $u = -4.65223x_1 - 4.19965x_2$, with a unique stable focus in $\boldsymbol{x}^{\star} = [0,0]$, namely when the pendulum is stable in the upright position. Further considerations regarding the ROA of the closed-loop systems will be discussed in Section 6.8.1.



(a) Open-loop system. **(b)** Closed-loop system.

Figure 5.13: Inverted pendulum phase plane.

5.6.2 Controlled Lorenz system

To investigate how the framework performs over more advanced benchmarks, the synthesis of *nonlinear control laws* was trialled for the 3-dimensional Lorenz system. The Lorenz system is a simplified model of highly nonlinear and coupled phenomena of atmospheric convection [230], identified as a system of interest owing to its complex and chaotic dynamics. For this analysis, the *controlled Lorenz system* [231, 232] is considered, i.e. the system described by:

$$\begin{cases} \dot{x}_1 = -\sigma_L(x_1 - x_2) + u_1 & \text{(5.24a)} \\ \dot{x}_2 = r_L x_1 - x_2 - x_1 x_3 + u_2 & \text{(5.24b)} \\ \dot{x}_3 = x_1 x_2 - b_L x_3 + u_3 & \text{(5.24c)} \end{cases}$$

where $\mathbf{x} = [x_1, x_2, x_3]^T$ represents the state-space vector, $\mathbf{p} = [\sigma_L, r_L, b_L]^T$ the scalar parameters and $\boldsymbol{u} = [u_1, u_2, u_3]^T$ the control input. By selecting $\sigma_L = 10.0$, $b_L = 8/3$ and

$r_L = 28.0$, as in [231, 232], and by setting the control input to zero (namely considering the open-loop case), the origin can be rendered as an unstable equilibrium and the characteristic butterfly-like *strange attractor* (or *Lorenz attractor*) is obtained, as reported in the open-loop trajectory of Fig. 5.14. For this study, a nonlinear control law was selected by employing *softplus* activation functions, and two hidden layers of 8 neurons each. The Lyapunov ANN is chosen as $V = \boldsymbol{W}_3(\boldsymbol{W}_2(\boldsymbol{W}_1\boldsymbol{x})^2)$, with the hidden layers composed of 10 neurons each.

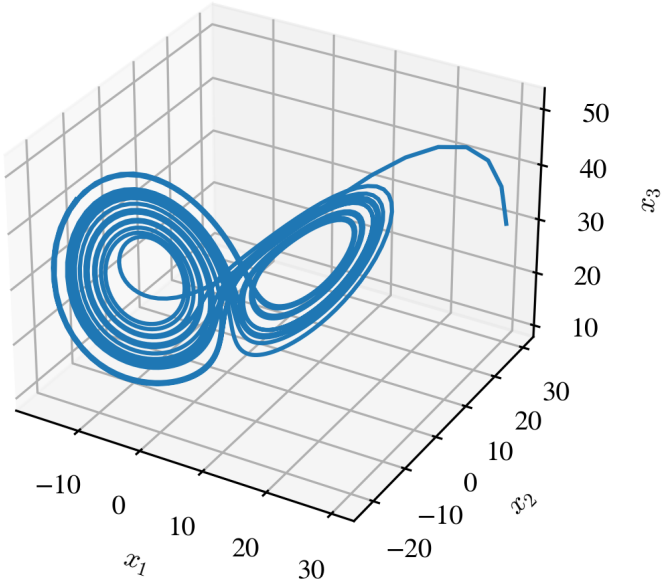
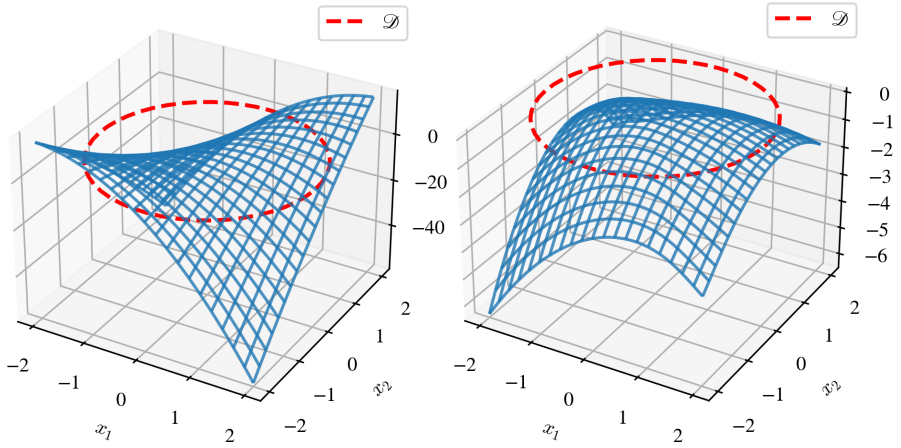


Figure 5.14: Controlled Lorenz system: open-loop trajectory (no control applied), © 2023 IEEE [22].

Out of ten tests, each one run with a different seed, seven converged within 1000 iterations (maximum computational time of 352 $[s]$). The evolution of $\dot{V}(\mathbf{x},\mathbf{u})$, plotted in the (x_1,x_2)-plane (with x_3 set to zero), is reported in Fig. 5.15a and in Fig. 5.15b, corresponding to the first and last training iterations respectively. Note that the Lie derivative is initially non-negative due to the random character of the ANNs initialisation, and evolves into a negative definite function upon convergence.

As a verification of the correctness of the closed-loop behaviour, Fig. 5.16 reports 30 closed-loop trajectories starting from points selected *sufficiently close* to the system origin (Section 6.8.1 discusses this matter in terms of the associated ROA in greater detail), while Fig. 5.17 highlights the corresponding fast decreasing Lyapunov values as the solutions approach the equilibrium. The control function values over time, cor-

responding to one converged controller, are reported in Fig. 5.18.



(a) First training iteration. **(b)** Last training iteration.

Figure 5.15: Controlled Lorenz system: Lie derivative in the (x_1, x_2)-plane over subsequent training iterations, © 2023 IEEE [22].

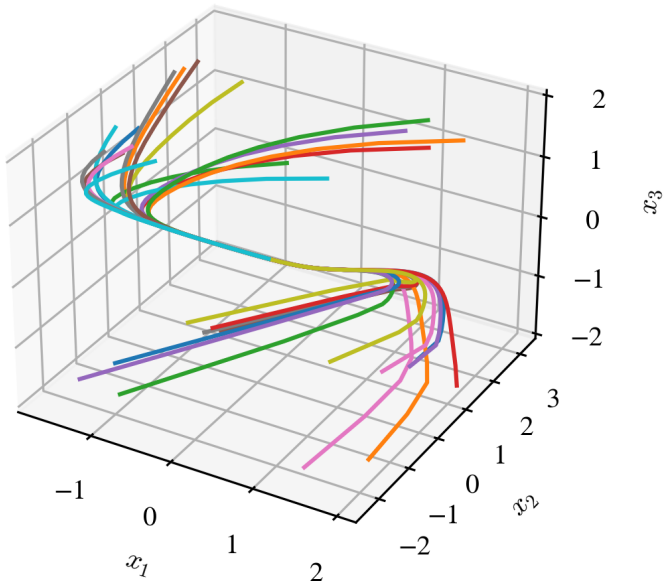


Figure 5.16: Controlled Lorenz system: closed-loop trajectories, © 2023 IEEE [22].

As expected, the synthesis of CLFs with nonlinear control laws is computationally more demanding and takes more time for the processes to complete, both on the Learner side, as more iterations are needed for the convergence of the training, and on the Falsifier side, as a more complex symbolic expression is being evaluated. Hence, a trade off must be made between the flexibility of the network and the computational time, in particular when higher dimensional systems are analysed. This would equate

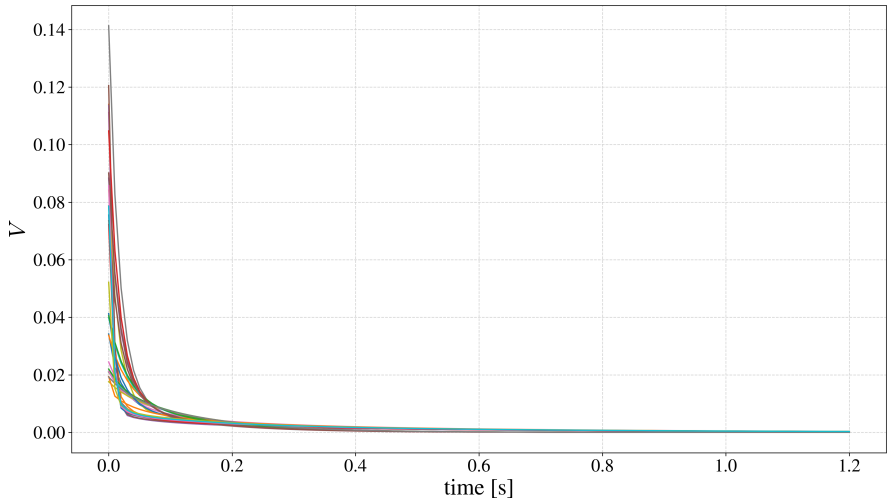


Figure 5.17: Lyapunov values along Lorenz system trajectories, © 2023 IEEE [22].

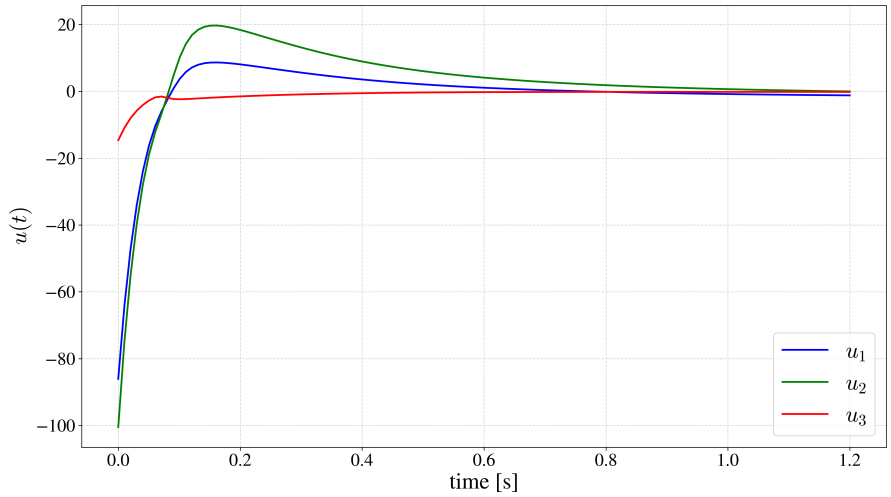


Figure 5.18: Control function values over time, © 2023 IEEE [22].

to the system having more than 10 dimensions, in line with previous findings [222].

5.7 Conclusions

In this chapter, an initial investigation of the NLC method was carried out, followed by the design of upgraded method tackling some of the NLC limitations, concluding with numerical simulations and evaluations.

The initial NLC investigation highlighted the need to upgrade the method to render it capable of synthesising control laws without the need to pre-initialise the control gain via other solutions (such as via optimal state-feedback control laws). Five elements that would benefit from improvements were highlighted and addressed: the CEs

generation logic, the dataset composition, the possible dataset overfitting, the algorithmic inefficiencies and the sensitivity to the learning rate.

As a consequence, an upgraded Falsifier module mitigating the issues linked to the overfitting of clustered CEs and rendering the CE generation more efficient was proposed. Next, a sliding window to selectively disregard previously targeted old dataset points was devised, reducing the required computational burden. Finally, algorithmic refinements were introduced.

The improvements to the learning scheme help to overcome critical limitations, such as the need to pre-initialise the control gains and the periodic re-start of the neural network training. The architecture offers the possibility to design nonlinear control laws, and in so doing presents another element of novelty. As a result, a method, which is robust to well-known issues, to inductively synthesise CLFs, by devising an upgraded Falsifier and careful selection of useful counterexamples, was presented.

The Augmented NLC method was shown to be capable of computing stabilising control laws without the need to pre-initialise the control gains, further increasing the generalisation power and overall applicability of this method. Both linear and nonlinear control laws were synthesised for unstable dynamics, showing the modularity of the proposed architecture, while limitations were highlighted.

Possible extensions should focus on scalability to both higher dimensional and uncertain dynamics, issues that will require dedicated assessment. Methods and tools introduced in this chapter serve as the base to compute fault-tolerant control laws, which is the focus of the following chapter.

Chapter 6

Passive Fault-Tolerant Augmented Neural Lyapunov Control

6.1 Introduction

This chapter outlines the design of the ML-based control method for devising control laws for nonlinear systems affected by actuator faults. First, an introductory overview to the problem is provided. Next, the underlying idea behind the extension of the ANLC method previously presented in Chapter 5 to account for actuator faults is illustrated. Then, modifications to both the Learner and Falsifer are introduced. According to the new fault-tolerant-capable method, an open-source software tool is designed and described. A numerical evaluation covering three case studies of increasing complexity is carried out next. Further refinements to the methods are then presented, such as the introduction of performance criteria to select which of the synthesised control laws best suits target control performance, and how to account for actuator saturation. A comparison with robust $\mathcal{H}_\infty$ control laws is presented next. To conclude, a procedure to use the proposed method to shape the ROA of a desired equilibrium is illustrated.

The software framework associated with this research is released open-source at: https://github.com/grande-dev/pFT-ANLC.

The outcome of this chapter provides answers to RQ3: **how can the Neural Lyapunov Control method be extended to a passive fault- tolerant control approach to ensure closed-loop stability for platforms affected by actuator faults?**

6.2 Preliminaries

During AUV deployments, a variety of unforeseen and unplanned for events can occur and may have the potential to jeopardise the mission success. Collision with drifting debris, growth of marine organisms, mechanical and electrical faults or attacks by ocean predators were identified as possible causes of the total or partial loss of efficiency of thrusters and of control plane jamming. To prevent an irrecoverable loss of the vehicle without relying on external information from a FDI system, pFTC methods can be devised.

Reliable control methods are often employed to design pFTCs, by minimising, for instance, the $\mathcal{H}_\infty$-norms between exogenous inputs and desired performances [52]. Moving forward from the linear methods, nonlinear control methods such as the Lyapunov-based pFTC can also be employed. The Lyapunov-based pFTC methods start from the design of a nominal control law that can cope with the faultless case, and encompass the addition of an extra term to compensate for the effect of a fault [44]. Such nonlinear methods can cope with partial loss of actuator efficiency, but not with the case of full loss of actuator efficiency. No nonlinear pFTC method is currently available to cope with the full range of possible faults, including partial loss of actuator efficiency, total loss of actuator efficiency, or a jammed actuator.

In the underwater environment, the application of linear pFTC methods, has been thoroughly investigated [117, 233]. The first limiting factor when employing linear pFTCs is represented by the necessity to define sensible operating points for linearisation [118]. This poses a twofold challenge: a) a sub-optimal linearisation choice naturally leads to degraded performance, an issue that is unavoidable when the system is highly nonlinear; b) the controller could turn out to be ineffective when the system operates far away from the linearisation point, especially under faulty conditions. These elements point to the necessity to advance the studies of nonlinear pFTCs such that they are better suited for use in the underwater environment.

A new trend focuses on designing controllers for nonlinear dynamical systems in an automatic fashion. These automatically synthesised controllers can be equipped with formal proof of stability, for instance by exploiting SMT-solving

[135, 136, 234, 235]. One such architecture relying on SMT-solving is the ANLC, which was introduced in Chapter 5.

In the reminder of this chapter, tailored modification to the ANLC learning-verification paradigm are devised, allowing the method to be extended to the case of nonlinear systems affected by actuator faults.

6.3 Design of a Passive Fault-Tolerant Control method based on the ANLC

The ANLC method illustrated in Chapter 5 can be extended to guarantee FTC properties by devising tailored modifications to both the *Learner* and the *Falsifier*.

Based on literature relevant to control and operation of AUVs, three underlying assumptions of the problem need to be stated first.

Assumption 1 *It is assumed and accepted that a maximum of only a single fault will exist at any period in time. A situation where multiple faults materialise at the same time is often symptomatic of a non-recoverable problem on the platform, and more drastic countermeasures (e.g. abort of the mission and recover the vehicle) are required.*

Assumption 2 *The dynamical system being considered is (locally) controllable in both the nominal and fault modes.*

Analysing the local controllability (or Kalman controllability) of a nonlinear system around a prescribed system state allows for a preliminary assessment regarding whether it is possible to control the nonlinear system in the neighbourhood of such state in each mode (faultless and fault mode) [236]. It is worth recalling that pathological cases exist: as with the case of stability analysis, a nonlinear system may be controllable, while a linearisation of such a system might not be [92]. To this regard, if the local controllability check is passed, namely if the linearisation of the nonlinear system around the desired state is controllable, it is worth proceeding with the attempt to synthesise the control law. If the local controllability of the system is not guaranteed, more comprehensive nonlinear controllability analyses need to be performed.

Assumption 3 *Faults of different nature are expected during AUV operations, namely partial loss of actuator efficiency, total loss of actuator efficiency and jammed actuators. It is assumed that once a fault occurs, it remains unresolved; in other words, faults are permanent events, and intermittent faults are not being considered.*

Focus of this study is the design of passive fault-tolerant control laws for a nonlinear dynamical system:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \phi_i), \tag{6.1}$$

where $\boldsymbol{x} \in \mathcal{D} \subseteq \mathbb{R}^n$ is the system's state, $\boldsymbol{u} \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input, and ϕ_i denotes possible system faults. It is assumed that each one of the d actuators can be affected by faults (collected in a set $\boldsymbol{\Phi}$). For brevity, the shorthand $\boldsymbol{f_n}(\boldsymbol{x}, \boldsymbol{u})$ is employed with reference to the nominal system, i.e. in the absence of faults, while $\boldsymbol{f}_{\phi_j}(\boldsymbol{x}, \boldsymbol{u})$ denotes the dynamics characterised by the fault of the j-th actuator (with $\phi_j \in \boldsymbol{\Phi}$).

The aim of this work is the design of a control law that drives the system to an equilibrium state $\boldsymbol{x}^\star \in \mathcal{D}$. Without loss of generality, it is assumed that the system exhibits an equilibrium $\boldsymbol{x}^\star = 0$. Details regarding how to extend the formulation when $\boldsymbol{x}^\star \neq 0$ are provided in Section 6.5.2. Along with the design of a control law, a certificate of the closed-loop stability via a CLF is provided.

The ANLC procedure can be extended to FTC by a) defining a set of dynamics capturing the nominal and faulty systems and b) rendering all the associated Lie derivatives as negative definite. Assuming to have d actuators that can be affected by faults, overall $(d+1)$ dynamics can be used to describe the nominal and faulty scenarios. To guarantee that $\boldsymbol{x}^\star$ is stable for all the $(d+1)$ dynamics, the corresponding $(d+1)$ Lie derivatives need to be rendered as negative definite; formally:

$$(\dot{V}_n(\boldsymbol{x}, \boldsymbol{u}) < 0) \wedge (\forall j \in \boldsymbol{\Phi} : \{\dot{V}_{\phi_j}(\boldsymbol{x}, \boldsymbol{u}) < 0\}). \tag{6.2}$$

where $\dot{V}_i(\boldsymbol{x}, \boldsymbol{u}) = \langle \boldsymbol{\nabla(V)_x}, \boldsymbol{f_i(x, u)} \rangle$, with $i = n$ for the nominal mode or $i \in \boldsymbol{\Phi}$ for the fault modes.

A key element used to extend the ANLC method to solve pFTC problems lies in the formal stability certificates that can be provided with dReal. It is worth recalling

that dReal is a *sound* solver, namely, when no CE is obtained, the CLF is formally valid over (a domain of) the real numbers. Nonetheless, dReal is δ-*complete*, so spurious CEs might be returned in the neighbourhood of the origin (within a precision δ). Therefore, a small neighbourhood that encompasses the origin is excluded from the SMT solver domain. This limits the stability certificate that can be provided with dReal to the ε-*stability* of $\boldsymbol{x}^\star$: namely, at steady-state the state-space trajectories contract to:

$$\|\boldsymbol{x}\|_2 \leq \varepsilon \tag{6.3}$$

with $\|\boldsymbol{a}\|_2 = \sqrt{a_1^2 + ... + a_n^2}$ denoting the 2-norm of vector $\boldsymbol{a}$ [220]. This feature, rather than being a limitation, represents a useful additional tuning parameter, especially within the fault-tolerant framework. In the presence of an actuator fault, the dynamical model is expected to deviate from the reference trajectory: the scope of fault-tolerant control includes the minimisation of this deviation. The ε-stability property proves that trajectories *never* exit a neighbourhood of the target setpoint, or, in other words, guarantees the forward invariance of the ε-bound. The validity domain is thus defined as:

$$\boldsymbol{x} \in \mathcal{D} : \varepsilon \leq \|\boldsymbol{x}\|_2 \leq \overline{\gamma} \tag{6.4}$$

where $\varepsilon, \overline{\gamma}$ are design parameters. This property fulfils one of the most significant requirements of FTC, i.e. guarantees a graceful performance degradation within a prescribed region that can be tuned during the control system design.

In the following section, modification to the learning-verification paradigm to synthesise a CLF abiding Eq. (6.2) is discussed.

6.3.1 Fault-tolerant Learner

As proposed for the ANLC method, two loss functions are employed in this work. The first, denoted as *Strict Lyapunov Risk Loss* ($\mathrm{L_{SLR}}$), is employed to detect when, over every sample ($\boldsymbol{s_i}$) within the dataset S, the theoretical Lyapunov conditions are verified, namely occurring when:

$$\forall \boldsymbol{s_i} \in S : \{V(\boldsymbol{s_i}) > 0, \dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i}) < 0\}. \tag{6.5}$$

The L_{SLR}, is therefore defined as:

$$L_{SLR} = \sum_{i=1}^{|S|} \mathcal{R}(-V(\boldsymbol{s_i})) + \sum_{i=1}^{|S|} \mathcal{R}(\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i})) + V(0)^2 \tag{6.6}$$

where $\mathcal{R}(a) = ReLU(a) = \max(0, a)$ for a generic input a and $|S|$ the cardinality of the sample set S.

When the L_{SLR} is equal to zero, all the points within S respect the theoretical Lyapunov conditions (5.4) as the Learner finds a candidate CLF, that can be translated and passed to the Falsifier for formal verification. Although a candidate CLF is obtained when $L_{SLR} = 0$, a user may be interested in encouraging a paraboloid shape to the CLF, in order to increase its ROA. To this end, an improved loss function is defined, referred to as *Empirical Lyapunov Risk Loss* (L_{ELR}), while L_{SLR} is employed as a logical condition to terminate the learning procedure.

The Empirical Risk Loss extends the definition of L_{SLR} by accounting for a fourth term regulating the size of the ROA through modification of the CLF shape, and for further properties of the Lie derivative as:

$$L_{ELR} = \alpha_1 \sum_{i=1}^{|S|} \mathcal{R}(-V(\boldsymbol{s_i})) + \alpha_2 L_{\dot{V}} + \alpha_3 V(0)^2 + \alpha_4 \frac{1}{|S|} \sum_{i=1}^{|S|} (\|\boldsymbol{s_i}\|_2 - \alpha_{ROA}\, V(\boldsymbol{s_i}))^2, \tag{6.7}$$

with $\alpha_1, \alpha_3, \alpha_4, \alpha_{ROA}$ tuning coefficients that can be selected as discussed in Chapter 5. The newly introduced loss term associated with the Lie derivative ($L_{\dot{V}}$), capturing both nominal and faulty dynamics is:

$$L_{\dot{V}} = \sum_{i=1}^{|S|} \mathcal{R}(\nabla \boldsymbol{V}(\boldsymbol{s_i}) \cdot \boldsymbol{f_n}(\boldsymbol{s_i}, \boldsymbol{u_i}) + \alpha_{\text{off}}) + \sum_{j=1}^{d} \sum_{i=1}^{|S|} \mathcal{R}(\nabla \boldsymbol{V}(\boldsymbol{s_i}) \cdot \boldsymbol{f}_{\phi_j}(\boldsymbol{s_i}, \boldsymbol{u_i}) + \alpha_{\text{off}}) \tag{6.8}$$

where α_{off} is an optional tuning term used to enforce more negative Lie derivatives [134].

This loss function at the same time imposes V being positive-definite, the Lie derivatives being $\dot{V}_i$ negative definite, the condition $V(0) = 0$, and it fosters circular level sets of the CLF. It is important to recall that the goal of the training is to achieve $L_{ELR} \approx 0$, namely it is required that the CLF *approximately* resembles a paraboloid of revolution, while verifying $L_{SLR} = 0$ exactly.

6.3.2 Fault-tolerant Falsifier

Once the training procedure reaches the end of its operations, the Learner passes a suitable candidate CLF, which is checked by the verification engine to formally certify that the Lyapunov conditions are satisfied over the whole continuous domain $\mathcal{D}$. If the CLF is found to not conform to the theoretical Lyapunov conditions, a CE is returned. This CE added to the training set S and the training is restarted. The verification check introduced in Chapter 5 as Eq. (5.23) can be modified to accommodate faulty dynamics as follows:

$$\exists \boldsymbol{x} : \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\} \implies \Big(V(\boldsymbol{x}) \leq 0 \vee \dot{V}_n(\boldsymbol{x}, \boldsymbol{u}) \geq 0 \vee \{\dot{V}_{\phi_j}(\boldsymbol{x}, \boldsymbol{u}) \geq 0\}_{j=1}^{d}\Big), \quad (6.9)$$

where the term $\{\dot{V}_{\phi_j}(\boldsymbol{x}, \boldsymbol{u}) \geq 0\}_{j=1}^{d}$ denotes the sequence of the d Lie derivatives associated to the fault modes. Similar to the nominal case, if the falsifier cannot find an instance that satisfies Eq. (6.9), the control Lyapunov function is certified to be valid and, consequently, the system stability is guaranteed even when an actuator within the set $\boldsymbol{\Phi}$ fails. The proposed method will henceforth be designated as passive Fault-Tolerant-Augmented Neural Lyapunov Control (pFT-ANLC).

6.4 Algorithm and software

Following the description of the proposed pFT-ANLC method, the pseudocode underlying the open-source software tool is introduced in Algorithm 2. The algorithm details the Learner, Discrete and SMT falsifiers, while illustrating the overall learning logic.

The algorithm starts by generating a training dataset S composed of random samples $\boldsymbol{s_i}$ from a uniformly distributed hypercube of side length $\overline{\gamma}$. If the linear control

Algorithm 2 Passive Fault-Tolerant-Augmented Neural Lyapunov Control

function LEARNER(S, f, ANN_η)
  repeat
    $V_\eta(s_i), u_\eta(s_i) \leftarrow \text{ANN}_\eta(s_i)$ ▷ ANN forward pass
    $\dot{V} \leftarrow$ Translator
    Compute loss L_{ELR}, L_{SLR}
    $\eta \leftarrow \eta - \nabla_\eta L_{ELR}$ ▷ Update weights via SGD
  until ($L_{SLR} > 0$)
  return $V_\eta(s_i), u_\eta(s_i)$

function DISCRETE FALSIFIER($V_c^S, \dot{V}_{n_c}^S, \dot{V}_{\phi_{i,c}}^S, \mathcal{D}, \zeta_{DF}$)
  Discretise $\mathcal{D}$ and numerically evaluate ($V_c^S \leq 0, \dot{V}_{n_c}^S \geq 0, \dot{V}_{\phi_{i,c}}^S \geq 0$)
  $CE_{DF} \leftarrow$ Violations points (max size ζ_{DF})
  return CE_{DF}

function SMT FALSIFIER($V_c^S, \dot{V}_{n_c}^S, \dot{V}_{\phi_{i,c}}^S, \mathcal{D}$)
  Using SMT solver to verify conditions
  return `unsat` or CE_{SMT}

function MAIN()
  Input: dynamics (f_n, f_{ϕ_i}), initial dataset (S), Falsifier domain $\mathcal{D}$ ($\varepsilon, \overline{\gamma}$), loss function ($\alpha_{(\cdot)}$, $\overline{\tau}$), learning rates, optional initial linear control gains (q^I)
  Initialise ANN size (optional: initialise linear control gains with q^I)
  repeat
    if ($N \geq S_{max}$): Apply sliding window
    $V_\eta(x), u_\eta(x) \leftarrow$ LEARNER(S, f, ANN_η)
    Compute symbolic values: $f_{n_c}^S, f_{\phi_{i,c}}^S, u_c^S, V_c^S, \dot{V}_{n_c}^S, \dot{V}_{\phi_{i,c}}^S$
    if ($L_{SLR} == 0$ and $L_{ELR} \leq \overline{\tau}$) **then**
      $CE_{DF} \leftarrow$ DISCR. FALSIFIER($V_c^S, \dot{V}_{n_c}^S, \dot{V}_{\phi_{i,c}}^S, \mathcal{D}, \zeta_{DF}$)
      if CE_{DF} is `None` **then**
        $CE_{SMT} \leftarrow$ SMT FALSIFIER($V_c^S, \dot{V}_{n_c}^S, \dot{V}_{\phi_{i,c}}^S, \mathcal{D}$)
      $S_{CE} \leftarrow (CE_{DF} \cup CE_{SMT})$
      if (not `unsat`**)**: $S \leftarrow (S \cup S_{CE})$
  until (`unsat` **or** `timeout`)

branch has been selected, the linear control weight can be optionally set to an initial value (q^I). Next, at each training iteration the *Learner* gradually minimises the loss L_{ELR} by updating the ANN weights through a SGD step. At each learning iteration, the current values of the ANNs weight and bias are collected in a vector η, defining V_η and u_η. Iterations of the learning steps continue until the Lyapunov conditions are verified $\forall \boldsymbol{s_i} \in S$, i.e. when $L_{SLR} = 0$. Once $L_{SLR} = 0$ holds true, the candidate CLF (V_c^S), the nominal Lie derivative ($\dot{V}_{n_c}^S$) and the set of Lie derivatives associated to the faulty systems ($\dot{V}_{\phi_{i,c}}^S$) are symbolically obtained by means of the *Translator*. The *Discrete Falsifier* is thus invoked to find CEs (CE_{DF}) over a prescribed discretisation of the domain $\mathcal{D}$. If CEs are found, these are added to S and the learning step is restarted. If no CEs are obtained, the *SMT Falsifier* is tasked with verifying the candidate CLF (V_c^S) over the dense domain of the Reals. Next, if a CE is obtained via SMT-solving (CE_{SMT}), the CE_{SMT} is added to S and the learning step restarted. Otherwise, the candidate CLF is formally verified to be correct (the SMT terminates with `unsat`), and the algorithm returns the synthesised control law.

As the aim of the tool is to synthesise CLFs that minimise the loss function L_{ELR}, a parameter $\overline{\tau}$ is introduced, setting a prescribed precision of approximation of the CLF, i.e. the smaller the magnitude of $\overline{\tau}$ the closer the CLF will resemble the paraboloid of revolution with the desired characteristics. Note that this is an additional feature that can be turned off when not required.

It is worth recalling that the decision problem of first-order logical formulae with generic nonlinearities over the theory of nonlinear arithmetic is an undecidable problem. The SMT solver chosen for this study, i.e. dReal, solves a δ-complete falsification constraint resulting in an NP-hard task [134]. Such an SMT choice guarantees that the problem can be formally solved provided there is the exclusion of the ε-bound in the neighbourhood of the origin. As the verification step represents the computational bottleneck of the procedure, a `timeout` value is introduced, defining an allocated maximum time threshold for the SMT Falsifier to compute the CEs. If the time threshold is exceeded, the training run is halted, flagged as not successful and a new run initialised with a different seed is initiated.

Finally, the use of a *selective sliding window* is recalled from Chapter 5, ensuring the maximum dataset cardinality remains bounded as the training proceeds and new CEs are generated.

The software tool aims at synthesising a *unique* set of gains that guarantees closed-loop stability in both the nominal and the faulty-case scenarios. The control function, which has been trained offline, is then deployed online in closed-loop applications without requiring further adjustments or real-time tuning. The proposed software tool employs *Python 3.9*, *dReal 4.21* and *PyTorch 1.7*. Installation instructions designed to enhance portability over different Operating Systems and a user-guide are provided within the associated repository page at https://github.com/grande-dev/pFT-ANLC, under the *installation* section. Different approaches to install the software tool are made available, such as installing it at system level, or within an *Anaconda* environment, or within a *virtual* environment.

6.5 Numerical Evaluation

In this section a series of tests are performed to verify the efficacy of the proposed method. First, the method is tested using an inverted pendulum system adapted to FTC applications. Next, the pFT-ANLC is tested against more complex benchmarks encompassing several AUVs, such as a traditional AUV affected by thruster malfunctions and an UG suffering the jamming of one stern-plane.

6.5.1 Case study 1: control of an inverted pendulum with actuator redundancy

In this section, a preliminary study showcasing the pFT-ANLC method applied to a simple dynamical system is illustrated. Once again, as per standard practice in control engineering applications, the method is first tasked with stabilising a pendulum in its upright position.

Naturally, the classical pendulum model presenting only one actuator is not suited for FTC studies, as the controllability of the system is lost upon injection of a complete fault, namely Assumption 2 is not satisfied. As a result, a modification to the inverted pendulum system is provided by introducing a redundant actuator set, namely two

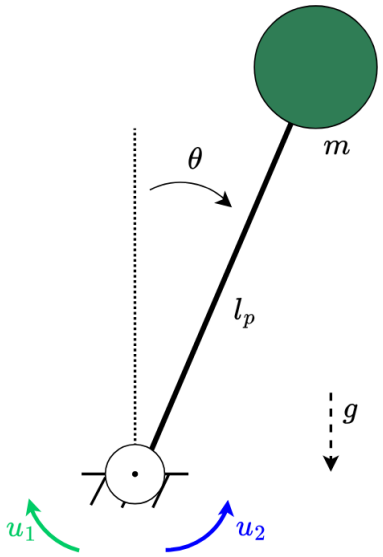


Figure 6.1: Inverted pendulum with redundant actuator set, © 2024 IEEE [24].

motors at the pendulum joint are available, representing two distinct control inputs.

The nominal dynamics $\boldsymbol{f_n}$ of an inverted pendulum with a redundant actuator set can be written as:

$$\begin{cases} \dot{x}_1 = x_2 & \text{(6.10a)} \\ \dot{x}_2 = \dfrac{mgl_p \sin x_1 - b_p x_2 + h_1 u_1 - h_2 u_2}{J_p} & \text{(6.10b)} \end{cases}$$

where x_1 and x_2 represent the pendulum angular position and velocity respectively, u_1 and u_2 indicate the torques generated by two distinct motors, b_p, l_p, m and J_p denote the drag coefficient, the length of the pendulum arm, the value of the lumped mass and the moment of inertia, respectively. Also, h_i denotes the *health status* of the i-th actuator, namely $h_i = 1$ when the actuator is functioning nominally, and $h_i = 0$ when the fault occurs. It follows that in the nominal scenario $h_1 = h_2 = 1$ holds true. A schematic depiction of the pendulum system with redundant actuators is shown in Fig. 6.1, while the numerical values of the parameters selected for this study are reported in Table 6.1.

Initially, the local controllability of the nominal and faulty models around $\boldsymbol{x}^\star$ was checked. First, the model (6.10) was linearised, resulting in the state and control Jacobians (**A** and **B**, respectively) being:

Table 6.1: Inverted pendulum with redundancy dynamical parameters.

Parameters	Values	Units	Explanation
m	0.15	kg	Pendulum mass
l_p	0.5	m	Length of the pendulum arm
b_p	0.1	kg m^2 s^{-1}	Drag coefficient
J_p	0.0375	kg m^2	Moment of inertia
g	9.81	m s^{-2}	Earth's gravity constant

$$\mathbf{A}|_{\mathbf{x}=\mathbf{x}^\star} = \begin{bmatrix} 0 & 1 \\ \frac{mgl_p \cos x_1}{J_p} & \frac{-b_p}{J_p} \end{bmatrix} \tag{6.11}$$

$$\mathbf{B}(h_1,h_2) = \begin{bmatrix} 0 & 0 \\ \frac{h_1}{J_p} & \frac{-h_2}{J_p} \end{bmatrix}. \tag{6.12}$$

Then, the state Jacobian $\mathbf{A}$ was evaluated at the target unstable equilibrium (namely $\boldsymbol{x}^\star = [0,0]^T$). Next, three values of the control Jacobian $\mathbf{B}$ were obtained by selecting $\mathbf{B}_\mathbf{n}(h_1 = 1, h_2 = 1)$, $\mathbf{B}_{\phi_1}(h_1 = 0, h_2 = 1)$ and $\mathbf{B}_{\phi_2}(h_1 = 1, h_2 = 0)$, corresponding to the nominal case scenario, and to the two modes characterised by there being one actuator fault at a time, respectively. Recalling that for the system under consideration $\boldsymbol{x} \in \mathbb{R}^{n=2}$ and $\boldsymbol{u} \in \mathbb{R}^{m=2}$, the controllability matrices were obtained as:

$$\boldsymbol{C_o} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} \end{bmatrix} \in \mathbb{R}^{n \times mn} \tag{6.13}$$

where $\mathbf{B} = \{\mathbf{B}_\mathbf{n}, \mathbf{B}_{\phi_1}, \mathbf{B}_{\phi_2}\}$. The local controllability could hence be assessed by checking the rank of the controllability matrices, resulting in full rank in both faultless and faulty operational mode.

In the remainder of this chapter, the notation vanilla Augmented Neural Lyapunov Control (vANLC) is introduced to denote the ANLC method to indicate the non-fault-tolerant version of the methods (presented in Chapter 5), versus the newly devised fault-tolerant-capable version, namely pFT-ANLC. The pFT-ANLC is hereby compared using the redundant inverted pendulum benchmark against two LQR laws and against one vANLC, in order to demonstrate the preliminary capabilities of the pFTC

method. In order to provide comparable results, in this section only linear control laws are synthesised via vANLC and pFT-ANLC, while more advanced tests encompassing the design of nonlinear control laws will follow in the subsequent Section 6.5.2 and Section 6.5.3.

6.5.1.1 LQR control

The LQR method was chosen as a comparative approach owing to its intrinsic robustness properties, namely of $[\frac{1}{2}, +\infty]$ of gain margin and of $[-60°, +60°]$ of phase margin [237, 238].

Two LQR laws were tuned based on different weights assigned to the actuators (denoted LQR_1 and LQR_2). In the first case, u_1 and u_2 were considered as equally important; in the second case, one motor was considered as the preferred control source, i.e. different control weights are imposed within the matrix $\boldsymbol{R}$. The gains were tuned using the legacy Bryson and Ho's rule [239]. For the state weighting matrix (common to both the LQR laws) a maximum desired error of $q_{1,max} = 10$ [deg] (equivalent to 0.175 [rad]) and 10 [deg/s] (equivalent to 0.175 [rad/s]) was imposed, resulting in $\boldsymbol{Q} = \text{diag}(\frac{1}{0.1754^2}, \frac{1}{0.1754^2}) = \text{diag}(32.83, 32.83)$.

Next, the first control tuning was obtained by imposing a conservative choice of the control effort. A maximum desired target torque was selected as $u_{1,max} = u_{2,max} = 0.1$ [Nm], resulting in the matrix $\boldsymbol{R_1} = \text{diag}(\frac{1}{0.1^2}, \frac{1}{0.1^2}) = \text{diag}(100.0, 100.0)$.

The second LQR law was instead designed by assuming a limited use of the first actuator as $u_{1,max} = 0.1$ [Nm], while prioritising the second actuator by setting a higher torque threshold as $u_{2,max} = 10.0$ [Nm]. The control input matrix associated to this choice of actuator priority resulted in $\boldsymbol{R_2} = \text{diag}(100.0, 0.1)$.

6.5.1.2 Augmented Neural Lyapunov Control

Next, a linear control law was synthesised employing the vANLC method. To replicate the case of actuators with different weights covered by LQR introduced above, a modification to the loss function described in Eq. (6.7) was introduced as:

$$L_{vANLC} = L_{ELR} + \alpha_{u_1} \sum_{i=1}^{|S|} u_{1,i}^2 + \alpha_{u_2} \sum_{i=1}^{|S|} u_{2,i}^2, \tag{6.14}$$

with $\alpha_{u_1} \in \mathbb{R}$, $\alpha_{u_2} \in \mathbb{R}$ being scalar parameters. These last two terms present in Eq. (6.14) add, on top of the stability requirements, the minimisation of the energy spent by the actuators as a further optimisation objective. By tuning the parameters α_{u_1} and α_{u_2} the effort of the two actuators can be penalised differently. For this training scenario, the parameters were chosen as $\alpha_{u_1} = 0.7$, $\alpha_{u_2} = 0.0$ to demonstrate qualitatively different behaviours of the two actuators.

6.5.1.3 Passive Fault-Tolerant ANLC

Finally, a pFT-ANLC law was devised. The pFT-ANLC encompasses, besides the nominal model, two faulty systems associated to the fault at u_1 and u_2, namely derived from Eq. (6.10) as $f_{\phi_1}(h_1 = 0, h_2 = 1)$ and $f_{\phi_2}(h_1 = 1, h_2 = 0)$.

For this scenario, 10 training runs were set up with the same hyperparameters (with the exception of the seed, cycled from 1 through 10), where the same learning parameters as the vANLC training were selected (architecture of the ANNs, learning rates, initial dataset size and loss function coefficients tuning). Note that each test results with a controller with slightly different performance, but are all certified to stabilise the closed-loop system. Given target performance criteria, such as optimum energy consumption or minimum convergence time, the best performing controller could be selected. The definition of such an indicator is provided in Section 6.6, but, for this preliminary study, a random law is selected from within those that converged.

Control laws comparison The following analysis includes the nominal case along with two faulty scenarios, for each of the four control schemes, resulting in 12 different case studies. The selected ANN architecture shared by the vANLC and pFT-ANLC is reported in Table 6.2, where the input, hidden, and output layers' sizes are outlined. The resulting closed-loop system eigenvalues are reported in Table 6.3.

Table 6.2: Redundant inverted pendulum: ANN architecture of the vANLC and pFT-ANLC.

Parameter	**Lyapunov ANN**	**Control ANN**
Layer size	[2, 10, 10, 1]	[2, 3]
Bias	[No, No, No]	No
σ	[x^2, linear, linear]	linear

As expected, every control scheme guarantees closed-loop stability under the

Table 6.3: Redundant inverted pendulum: closed-loop poles location © 2024 IEEE [24].

Scenario	Controller	Dynamics		
		Nominal (f_n)	**Fault 1** (f_{ϕ_1})	**Fault 2** (f_{ϕ_2})
1	LQR_1	-1.29, -22.06	-0.37, -12.92	-0.37, -12.92
2	LQR_2	-1.01, -160.39	-0.99, -145.98	0.18, -17.28
3	vANLC	-1.91, -128.18	-1.02, -132.29	$0.28 \pm 9.50j$
4	pFT-ANLC	-1.68, -208.76	-1.59, -101.87	-1.58, -108.08

nominal scenario, namely when the system is faultless. Additionally, every control law is able to stabilise the pendulum when a fault is introduced on the first actuator, even those methods that are not fault-tolerant. This behaviour is expected as each non fault-tolerant control law is either designed with both actuators having the same weight, i.e. for the LQR_1, or with the second actuator prioritised, i.e. for the LQR_2 and vANLC. Prioritising the second actuator means that the first actuator performs an accessory role and, in presence of a fault at u_1, closed-loop behaviour is affected but not up to the point of reaching instability. It should be noted that the poles reported in the first three scenarios of Table 6.3 undertake a shift towards the imaginary axis in the *Fault 1* mode when compared to the *Nominal* mode, hinting towards a reduced stability margin.

Finally, when the fault is injected on the second motor, two of the closed-loop systems become unstable, namely the LQR_2 and vANLC. On the other hand, both the LQR_1 and pFT-ANLC methods retain stability.

The closed-loop dynamics of the LQR_2, vANLC, and pFT-ANLC are illustrated in Fig. 6.2 (LQR_1 is not shown for readability, but the dynamics qualitatively resemble those of the pFT-ANLC). The corresponding graphical animations of the pendulum motion with the three control laws are provided within the repository page: https://github.com/grande-dev/pFT-ANLC-preview. All three control laws illustrated in Fig. 6.2 can be seen stabilising the pendulum in the nominal case (solid lines) and first faulty model (square markers). When a fault at the second motor occurs, only the pFT-ANLC can stabilise the system; both the green and orange lines with round markers do not converge to the desired equilibrium point.

This preliminary analysis results in two conclusions. First, the analysis trivially

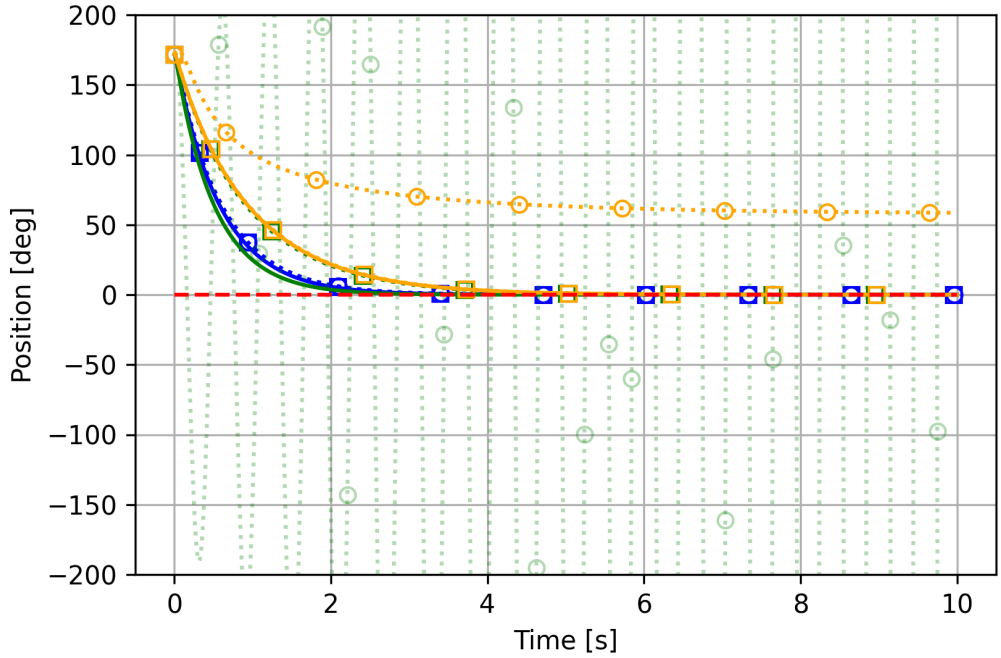


Figure 6.2: Inverted pendulum closed-loop tests. Color code: blue lines (pFT-ANLC), orange lines (LRQ_2), green lines (vANLC). Line style: solid (nominal dynamics), dashed with square markers (fault 1), dashed with round markers (fault 2), © 2024 IEEE [24].
The systems dynamics can be visualised at: https://github.com/grande-dev/pFT-ANLC-preview.

remarks how state-feedback control laws are not always able to guarantee stability when the underlying system deviates from the nominal dynamics, such as the loss of one actuator. Additionally, the analysis confirms how the modification to the vANLC leading to the pFT-ANLC method enables the synthesis of control laws capable of guaranteeing stability when subjected to faulty conditions.

6.5.2 Case study 2: Control of an Autonomous Underwater Vehicle

In this section, the proposed method is applied to a case study involving the control of an AUV. The AUV actuator configuration is inspired by the hover-capable AUV developed at the National Oceanography Centre[1] and shown in Fig. 6.3. The schematic representation of the actuator configuration of the vehicle is illustrated in Fig. 6.4. Each thruster is capable of generating force along the thruster axis in both the positive and

[1] https://noc.ac.uk/technology/technology-development/marine-autonomous-robotic-systems

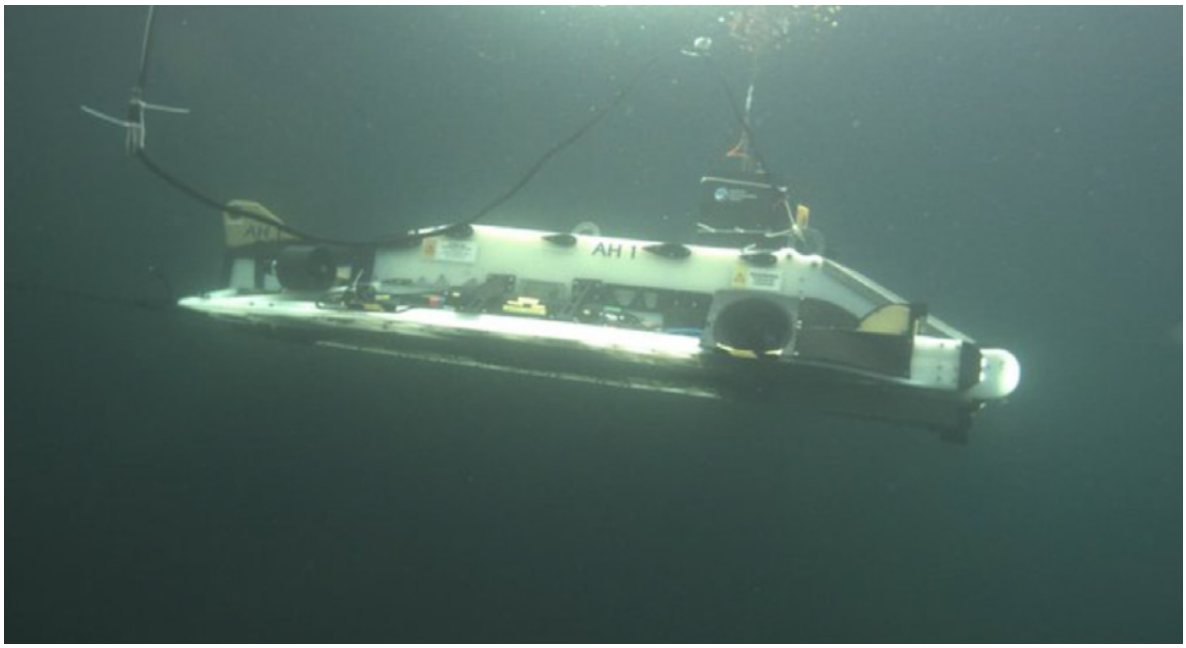


Figure 6.3: Hover-capable AUV developed at the National Oceanography Centre, illustrated from [25].

negative directions.

For this study, a two dimensional dynamical model accounting for surge speed (u) and angular velocity around the vertical axis (r), is used to describe the planar motion of the AUV, while the sway speed (v) is neglected. Each of the thrusters F_i is oriented at an angle β_i with respect to the y-axis of the body frame (y_B) and is located at distance l_i from the COG, with $l_{i,x}$ and $l_{i,y}$ indicating the projections along the x_b and y_b-axis, respectively. By denoting m as the mass of the vehicle and with J_z the vehicle's moment of inertia around the vertical axis, the AUV dynamics in a horizontal plane, characterised by $\boldsymbol{x} = [u, r]^T$ and $\boldsymbol{u} = [F_1, F_2, F_3]^T$, are described as:

$$\begin{cases} \dot{x}_1 = \dfrac{-X_u x_1 - X_{uu} x_1^2 + h_1 F_{1,x} + h_2 F_{2,x} + h_3 F_{3,x}}{m} & \text{(6.15a)} \\ \dot{x}_2 = \dfrac{-N_r x_2 - N_{rr} x_2^2 + (-F_{1,x} l_{1,y} + F_{1,y} l_{1,x}) h_1}{J_z} + \\ \qquad \dfrac{(-F_{2,x} l_{2,y} + F_{2,y} l_{2,x}) h_2 + (-F_{3,x} l_{3,y} + F_{3,y} l_{3,x}) h_3}{J_z} & \text{(6.15b)} \end{cases}$$

where $F_{i,x} = F_i \sin(\beta_i)$ and $F_{i,y} = F_i \cos(\beta_i)$ represent the projection of F_i along the x_B and y_B-axes, respectively; X_u, X_{uu} denote the linear and quadratic surge drag coeffi-

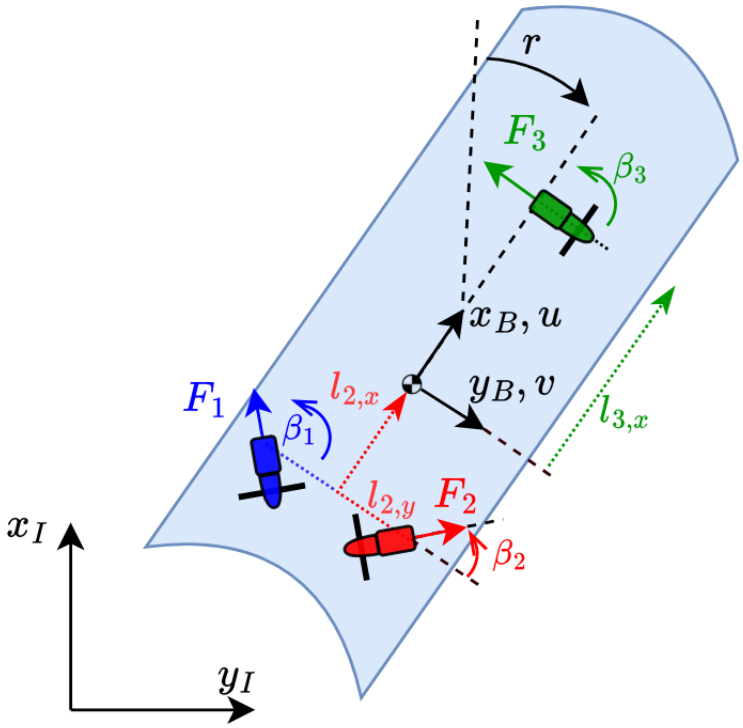


Figure 6.4: AUV with three (fixed) thrusters moving over the horizontal plane, characterised by surge speed (u), sway speed (v) and angular rate (r).

cients, while N_r, N_{rr} the linear and quadratic yaw drag coefficients. Note that, with respect to the dynamical models introduced in Chapter 4, the Coriolis effects, inducing cross coupling between the DOFs, do not explicitly appear in Eq. (6.15), due to the sway dynamics being neglected. The numerical values of the parameters selected for this study are reported in Table 6.4.

Different to the neural-Lyapunov analyses discussed in relevant literature [134] and in the previous sections, the goal of this study is to synthesise a control law to stabilise the system around a non-zero equilibrium. In this specific case, a non-zero equilibrium coincides with maintaining the AUV at a desired target speed.

6.5.2.1 Shifting the equilibrium state

The control design problem is formulated as the stabilisation of the dynamics about a generic desired equilibrium state $\boldsymbol{x}^\star$. When the desired $\boldsymbol{x}^\star$ does not coincide with the origin, it is at times convenient to shift the equilibrium to the origin. This consideration is particularly relevant in the method proposed, as both the Learner and Falsifier, described so far, assume the equilibrium to be located at the origin.

Table 6.4: AUV dynamical parameters.

Parameters	Values	Units	Explanation
m	500.0	kg	Vehicle mass including added mass
J_z	300.0	kg m^2	Vehicle inertia (around vertical axis) including added inertia
X_u	6.106	kg s^{-1}	Surge linear drag coefficient
X_{uu}	5.0	kg m^{-1}	Surge quadratic drag coefficient
N_r	210.0	kg m^2 s^{-1}	Yaw linear drag coefficient
N_{rr}	3.0	kg m^2	Yaw quadratic drag coefficient
$l_{1,x}$	-1.01	m	Distance of F_1 projected along x_B
$l_{1,y}$	-0.353	m	Distance of F_1 projected along y_B
β_1	110.0	deg	Orientation of F_1 with respect to (wrt) y_B
$l_{2,x}$	-1.01	m	Distance of F_2 projected along x_B
$l_{2,y}$	0.353	m	Distance of F_2 projected along y_B
β_2	70.0	deg	Orientation of F_2 wrt y_B
$l_{3,x}$	0.75	m	Distance of F_3 projected along x_B
$l_{3,y}$	0.0	m	Distance of F_3 projected along y_B
β_3	180.0	deg	Orientation of F_3 wrt y_B

To shift the equilibrium to the origin, a new set of coordinates, described by the pair $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ can be defined and a controller to drive the dynamics to the origin of $\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{f}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ can be devised.

Given a nonlinear system with exogenous inputs in the form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{6.16}$$

it is possible to analyse the stability of the equilibrium of interest by defining a new set of coordinates by extending [92] (pp. 45) as:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}^\star \tag{6.17a}$$

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}^\star, \tag{6.17b}$$

transforming the dynamics into:

$$\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{f}(\tilde{\boldsymbol{x}} + \boldsymbol{x}^\star, \tilde{\boldsymbol{u}} + \boldsymbol{u}^\star). \tag{6.18}$$

It should be noted that the AUV system proposed in this section shows a trivial equilibrium at the origin, corresponding to the vehicle having zero surge speed (u) and zero angular rate (r). To control the dynamics around a target (generic) operating point $\boldsymbol{x}^\star = [u^\star, r^\star]^T$, the corresponding shifted dynamical model is defined as:

$$\begin{cases} \dot{\tilde{x}}_1 = \dfrac{-X_u(\tilde{x}_1 + x_1^\star) - X_{uu}(\tilde{x}_1 + x_1^\star)^2 + h_1(\tilde{F}_{1,x} + F_{1,x}^\star)}{m} + \\ \quad \dfrac{h_2(\tilde{F}_{2,x} + F_{2,x}^\star) + h_3(\tilde{F}_{3,x} + F_{3,x}^\star)}{m} \end{cases} \tag{6.19a}$$

$$\begin{cases} \dot{\tilde{x}}_2 = \dfrac{-N_r(\tilde{x}_2 + x_2^\star) - N_{rr}(\tilde{x}_2 + x_2^\star)^2 + (-(\tilde{F}_{1,x} + F_{1,x}^\star)l_{1,y} + (\tilde{F}_{1,y} + F_{1,y}^\star)l_{1,x})h_1}{J_z} + \\ \quad \dfrac{(-(\tilde{F}_{2,x} + F_{2,x}^\star)l_{2,y} + (\tilde{F}_{2,y} + F_{2,y}^\star)l_{2,x})h_2}{J_z} + \\ \quad \dfrac{(-(\tilde{F}_{3,x} + F_{3,x}^\star)l_{3,y} + (\tilde{F}_{3,y} + F_{3,y}^\star)l_{3,x})h_3}{J_z} \end{cases} \tag{6.19b}$$

By inserting the target equilibrium value $\boldsymbol{x}^\star$, it is possible to design a control law to stabilise the origin of system in Eq. (6.19) to correspondingly stabilise the system in Eq. (6.15) around $\boldsymbol{x}^\star$. Care must be taken during the synthesis of such a control law: for a system with an equilibrium at the origin, the control law is designed as $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x}$, with $\boldsymbol{K}$ being a generic state-feedback control gain. In addition, for the shifted system expressed in coordinates $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$, the control law needs to be adapted to the new coordinate frame as $\tilde{\boldsymbol{u}} = \boldsymbol{K}\tilde{\boldsymbol{x}}$. Upon devising such a control law $\tilde{\boldsymbol{u}}$, it might be desirable to apply the control law in the original coordinate system $(\boldsymbol{x}, \boldsymbol{u})$. By substitution, it is possible to obtain:

$$\tilde{\boldsymbol{u}} = (\boldsymbol{u} - \boldsymbol{u}^\star) = \boldsymbol{K}\tilde{\boldsymbol{x}}, \tag{6.20}$$

and, finally:

$$\boldsymbol{u} = \boldsymbol{K}\tilde{\boldsymbol{x}} + \boldsymbol{u}^\star. \tag{6.21}$$

Intuitively, this means that to control the dynamical system around a target non-zero equilibrium, a constant input $\boldsymbol{u}^\star$ needs to be applied. Obtaining such a value $\boldsymbol{u}^\star$ is not trivial, and would require manual calculations, defeating the purpose of the automatic control synthesis proposed in this work. To this aim, the learning architecture can be

modified to automatically compute the value of $\boldsymbol{u}^\star$ while synthesising the usual control gain $\boldsymbol{K}$. Thus, for a non-zero target equilibrium, the control law needs to encompass a bias term, resulting in the form $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x} + \boldsymbol{b}$, where both the terms $\boldsymbol{K}$ and $\boldsymbol{b}$ will be automatically computed by the software tool.

First, as with the case of the inverted pendulum system (Section 6.5.1), the controllability assumption was checked. Recalling that for the system under consideration $\boldsymbol{x} \in \mathbb{R}^{n=2}$ and $\boldsymbol{u} \in \mathbb{R}^{m=3}$, the Kalmann controllability matrices corresponding to the nominal and fault modes were obtained as $\boldsymbol{C_o} = \begin{bmatrix} \mathbf{B} & \mathbf{AB} \end{bmatrix} \in \mathbb{R}^{n \times mn}$. The local controllability was assessed by checking the rank of the controllability matrices obtained by setting $\mathbf{B} = \{\mathbf{B_n}, \mathbf{B}_{\phi_1}, \mathbf{B}_{\phi_2}, \mathbf{B}_{\phi_3}\}$, resulting in full rank in each operational mode.

6.5.2.2 Results

The proposed case study investigates the capability to synthesise control laws in scenarios involving progressively more faults, while maintaining the AUV at $\mathbf{x}^\star = [0.5, 0.0]^T$, in both the nominal and fault modes.

Three scenarios of possible faults are reported in Table 6.6. First, a scenario considering a possible fault occurring only at thruster F_1 (aft port) was considered. Next, possible faults at F_1 or at F_3 (bow) were considered. Finally, possible faults at F_1 or at F_2 (aft starboard) or on F_3 were investigated. For this application, a nonlinear control law was employed. The selected ANN architecture is reported in Table 6.5, where the input, hidden, and output layers' sizes are presented, along with the presence of the bias and the choice of activation functions.

Table 6.5: AUV campaign – ANN architecture.

Parameter	**Lyapunov ANN**	**Control ANN**
Layer size	[2, 10, 10, 1]	[2, 30, 3]
Bias	[No, No, No]	[Yes, Yes]
σ	[x^2, linear, linear]	[tanh, linear]

The training was carried out on an unassuming office laptop without GPU module. The machine featured an Intel Core i7-8665U CPU with 8 cores running at 1.90GHz and 16 GB of RAM. During the training, 4 threads are generated and executed over 2 CPU cores, while 0.7 GB of RAM was utilised.

Table 6.6: AUV campaign – Synthesis statistics.

Faulty actuators	Iterations			Time [s]			Success rate [%]
	min	mean(std.)	max	min	mean(std.)	max	
F_1	18	96(67)	253	1	3(2)	5	100
F_1 or F_3	18	116(85)	300	1	4(2)	9	100
F_1 or F_2 or F_3	64	562(343)	1000	4	10(10)	39	90

Table 6.6 reports the results of three simulation campaigns run for different fault scenarios. Each simulation campaign is composed of 10 tests, all sharing the same hyperparameters with the exception of the seed (within each campaign, the seeds are cycled from 1 to 10). Results are reported in terms of: number of learning iterations (as minimum, maximum, mean and standard deviation), computational time and success rate (what percentage of the 10 tests successfully find a valid CLF). Note that if a CLF is not found within 1000 learning iterations, the run is flagged as unsuccessful.

For the most demanding test scenario, i.e. the one with faults occurring at F_1 or F_2 or F_3, all the 9 tests that converged terminated within a maximum time of 39 [s]. The results reported in Table 6.6 hints towards a required computational effort increasing as more faults are accounted for. Additionally, fine-tuning the hyperparameters of the control ANN became more challenging as the number of faults increases. Smaller architectures were tested for the case of faults on F_1 alone and of (F_1 or F_3) being faulty, without significant differences in the success rate values. ANN architectures with a limited number of neurons failed to systematically synthesise CLFs when faults can occur at all three thrusters, highlighting the need to select sufficiently expressive nonlinear control functions as the complexity of the problem increases. One resulting CLF for the case of F_1 or F_3 faulty, obtained after 142 training iterations is reported in Fig. 6.5, with the Lie derivative associated to the system with a fault at F_1 illustrated in Fig. 6.6.

Finally, closed-loop performance of the resulting fault-tolerant dynamics is illustrated and discussed. Fig. 6.7 reports the AUV surge dynamics, showing the range of different dynamics emerging from different control laws (depicted with the blue interval), stemming from the 10 converged runs synthesised for the case of (F_1 or F_3) thrusters possibly at fault. The surge dynamics are initialised at 0.4 [m/s], and are

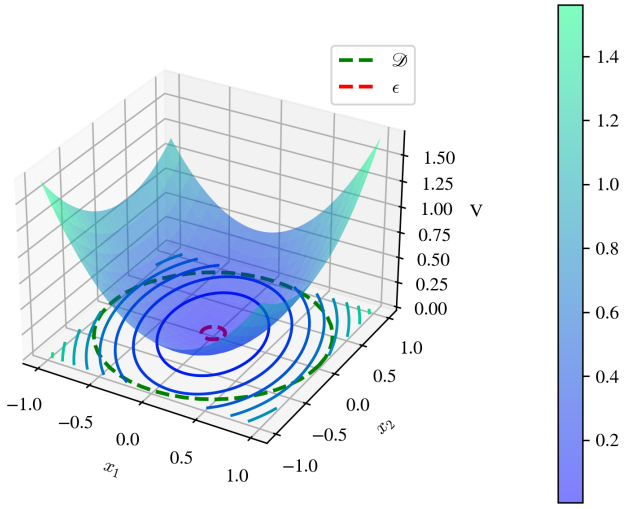


Figure 6.5: Synthesised CLF for the AUV system, © 2024 Elsevier [23].

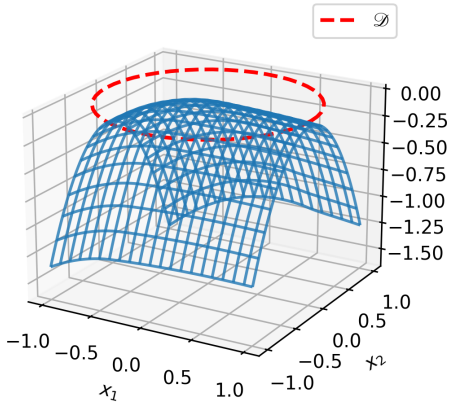


Figure 6.6: Lie derivative associated to the AUV system with fault at F_1.

shown converging to the desired ε-stability bound (0.5 ± 0.01 [m/s]). When a fault occurs at t=50 [s] at thruster F_1, the surge speeds undergo a drop that always remain within the desired ε-stability threshold (regardless of the initial conditions, or seed, of the test). This behaviour is achieved through the pFT-ANLC learning to *automatically* set the steady-state target higher to compensate for the possible occurrence of faults. This is in turn accomplished by applying an excess of force at F_2 with respect to F_1 and a non-zero force F_3, as illustrated in Fig. 6.8. Analogous behaviour is noticed in the angular rate dynamics, reported in Fig. 6.9 for the case of a fault at F_3: the controllers learn to converge to an offset value with respect to $x_2^\star$ during nominal operations to mitigate for possible faults.

It should be noted that some values of the angular velocity dynamics in Fig. 6.9

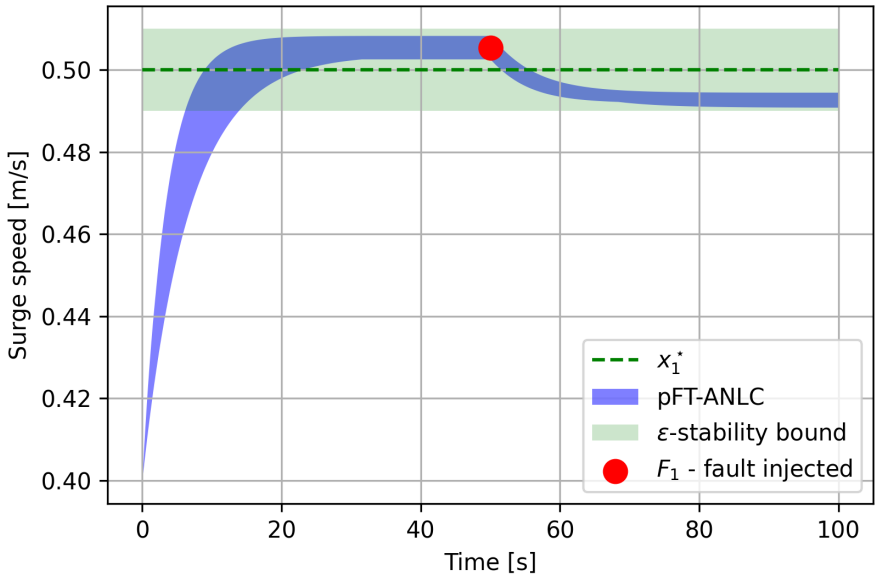


Figure 6.7: Closed-loop test AUV system: range of surge dynamics associated to 10 synthesised controllers (blue interval) — fault at F_1 injected at t=50 [s], © 2024 IEEE [24].

seem misleading exceeding the ε-stability bound during the time span between t=0.0 [s] and t=10.0 [s]. The system exhibits this behaviour because the dynamics have in fact not yet converged to within the ε-stability bound. To further illustrate such behaviour, in Fig. 6.10 the dynamics in the (x_1, x_2) plane is reported. The dynamics are initialised as $\boldsymbol{x}_0 = [0.4, 0.0]^T$ and converge towards $\boldsymbol{x}^\star = [0.5, 0.0]^T$. A magnified view of the dynamics in the neighbourhood of the origin is presented in Fig. 6.11. The x_2 dynamics reach the desired stability threshold, by temporarily assuming values $x_2(t) > |\varepsilon|$, during the transient phase. Once the dynamics enter the ε-stability bound, they no longer exceed the boundaries of the region. Interestingly, it can be noticed how, in this case scenario, the faultless dynamics converge to the outer bound of the ε-stability domain, to then transition to another equilibrium state (still within the ε-stability bound) upon the injection of a fault.

Finally, recalling that these results are associated to training runs carried out by considering faults at F_1 or F_3 only, it would be possible to wonder what happens when a fault at F_2 is injected into the system. In fact, due to the symmetry of actuators F_1 and F_2, one could surmise that the system is already robust to faults at F_2. Evaluating

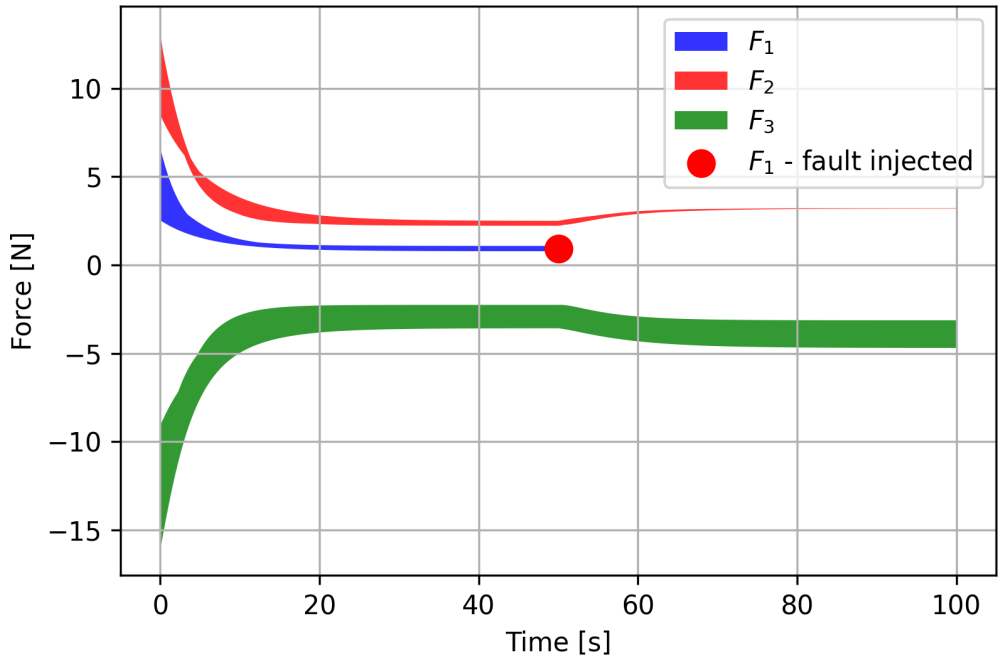


Figure 6.8: Closed-loop test AUV system: range of control efforts associated to 10 synthesised controllers (blue, red and green intervals) — fault at F_1 injected at t=50 [s], © 2024 IEEE [24].

this scenario is equivalent to test a beyond-design fault, namely consisting in injecting a fault which the control system was not designed to compensate for [97].

To this aim, Fig. 6.12 reports a similar case scenario as discussed above but for a fault at F_2. The dynamics are initialised as $\boldsymbol{x}_0 = [0.4, 0.0]^T$ converging in the faultless scenario to the outer bound of the ε-stability domain. Once the fault is introduced at F_2, the dynamics leave such ε-stability region, in turn revealing that the control law is not able to control such fault mode. Naturally, this result is to be expected as the fault at F_2 is beyond-design and no assumption of further robustness shall be made without an appropriate stability verification analysis.

6.5.3 Case study 3: Underwater Glider with saturated control

This following case study focuses on an UG during the most common operating condition, i.e. a profiling steady dive, covering the majority of the deployment time [178]. Gliders follow a saw tooth pattern adopting steady gliding conditions on both the ascent and descent manoeuvring phases, such as the glide previously illustrated in Fig. 4.5a. The UG alternates a positive buoyancy and nose-up attitude during the climbing phase

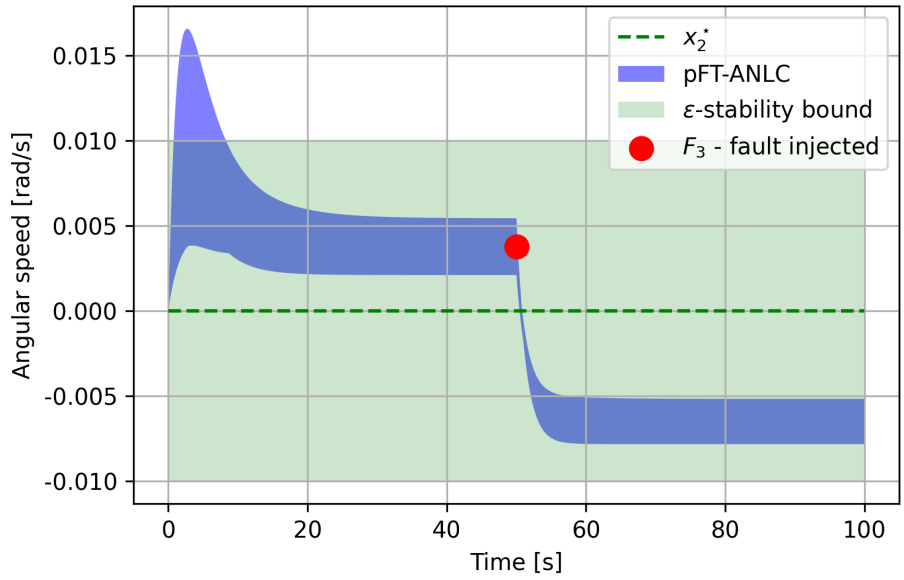


Figure 6.9: Closed-loop test AUV system: range of angular rate dynamics associated to 10 synthesised controllers (blue interval) — fault at F_3 injected at t=50 [s], © 2024 IEEE [24].

with a negative buoyancy combined with a nose-down attitude during the diving phase. For this case study, only the vertical dynamics were considered, as the sagittal plane is proved to be invariant [20]. When no initial linear or angular out-of-plane accelerations are provided, the vehicle remains on that plane indefinitely.

The modelling of the UG vehicle employed for this case study follows the derivation provided in Chapter 4, with the main elements recalled hereby. A NED frame with origin $\{O_i\}$ was chosen as *inertial reference* since the distances involved in the simulations proposed are of small scale when compared to the Earth's radius. Next, a *body-fixed reference* frame of origin $\{O_b\}$ and with axes x_b-z_b was fixed at the centre of buoyancy of the glider, which in turn coincides with the centroid of the hull. The orientation of the hull is obtained by applying a rotation of θ (pitch angle) around y_i (positive nose-up). The x_b-axis of the body frame is aligned with the longitudinal axis of the vehicle, while the z_b-axis points downward. Next, a *flow reference frame* was aligned with the direction of the glider velocity (V_r), with its origin coinciding with $\{O_b\}$, and the axes x_f and z_f are obtained by applying a rotation of α (angle of attack) from the body-fixed axes. This choice of the terns, reported in Fig. 6.13, is

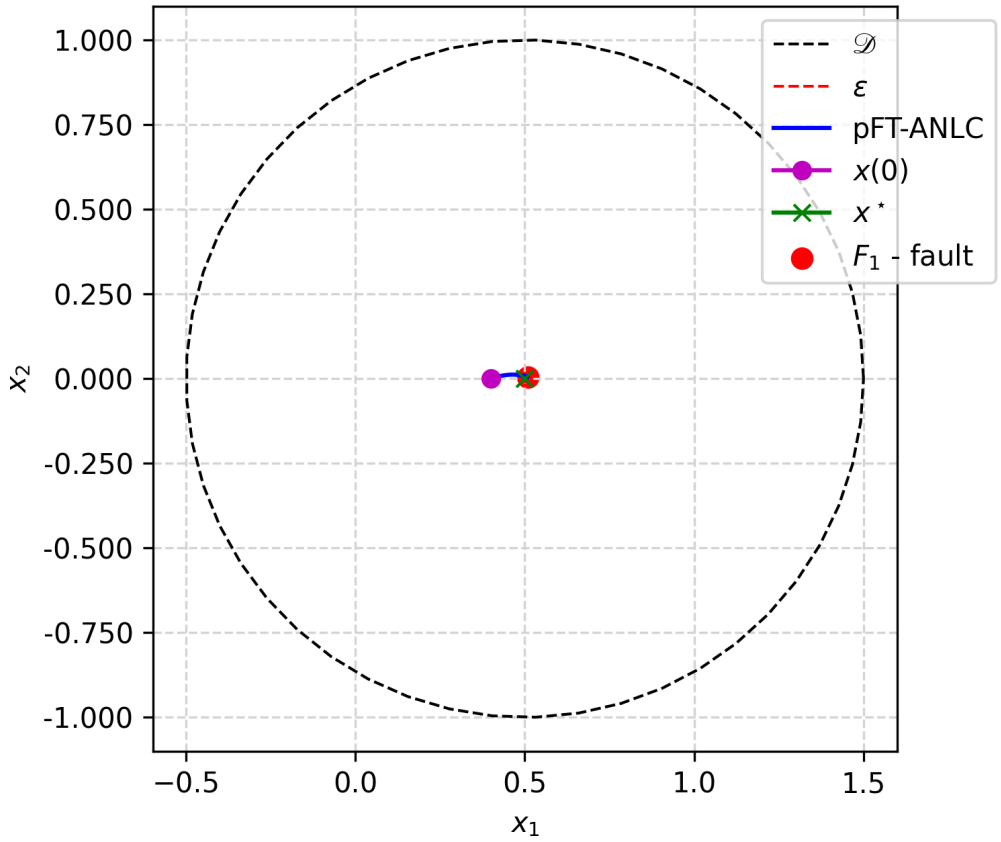


Figure 6.10: Nonlinear trajectories for the AUV dynamics, illustrating the significance of the ε-stability bound.

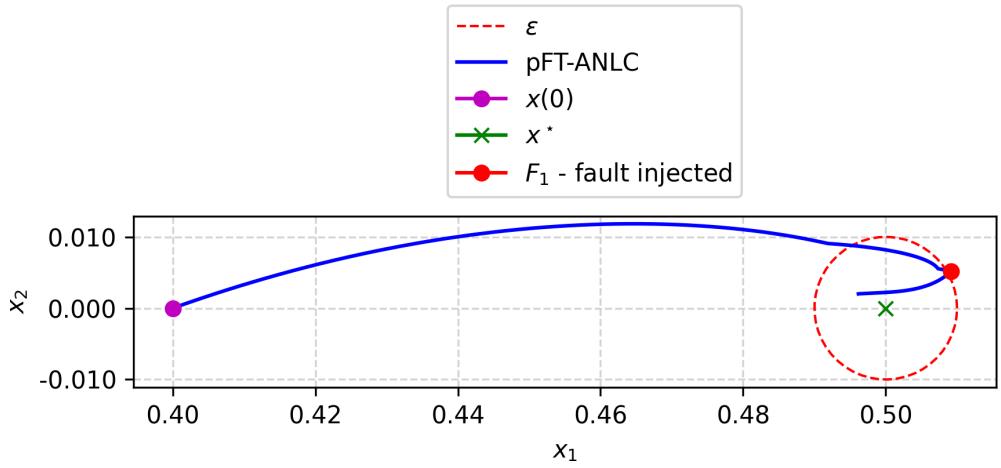


Figure 6.11: Magnified view of Fig. 6.10. Upon occurrence of the fault at F_1, the dynamics remain bounded within the ε-stability bound.

convenient when expressing forces according to standard hydrodynamic theory [20]. Hydrodynamic forces, namely lift (L) and drag (D), are aligned with the flow-tern axes. Restoring forces, namely gravity (G) and buoyancy (B) act parallel to the inertial axes.

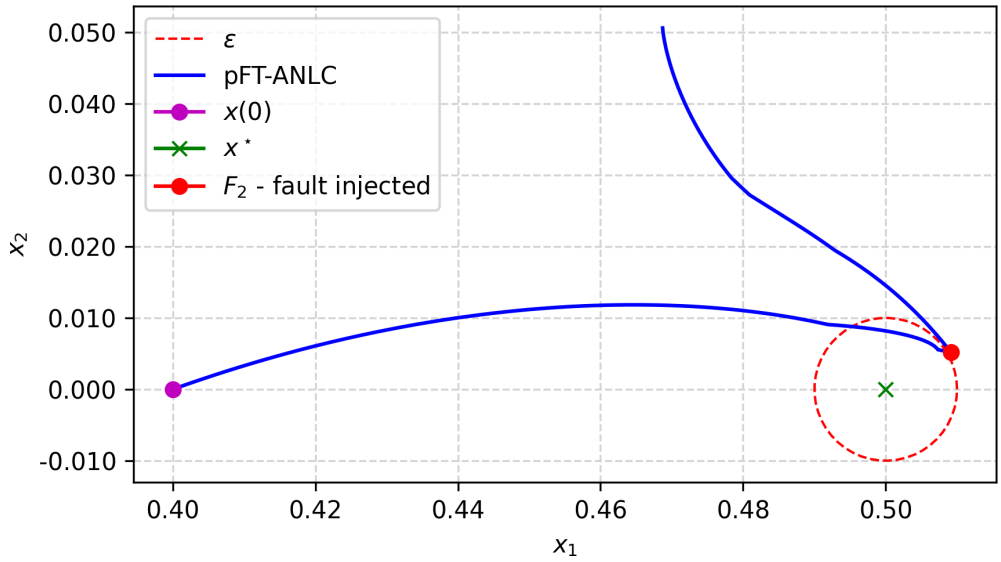


Figure 6.12: Test of the dynamics after injecting a fault at F_2, not covered during the training. Upon the occurrence of the fault, the dynamics leave the ε-stability bound.

The inertia forces (accounting for added mass terms) are expressed in the body-fixed reference.

The aim of this study was to devise a control law that maintains the body-fixed velocities of the UG within prescribed bounds when potential faults occur. The body-fixed velocity along the x_b-axis is hereby denoted as v_1, while v_3 is used to denote the velocity along z_b. The onboard system employs three actuators, a VBD and a pair of independent stern-planes (δ_1 and δ_2). This vehicle concept is inspired by AUVs with redundant movable surfaces designed in "+" (or cruciform-stern) or "×" (or X-stern) configurations, such as the ones on board the *Autosub Long Range* series [240] and hybrid AUVs such as the *Sea-Whale 2000* [156]. These AUV configurations allow the exploiting of redundant actuators when one of the two stern surfaces fail. Additionally, as per standard glider modelling, two masses can be identified: a uniformly distributed static mass, lumped within a single term (m_s) located at $\{O_b\}$, and an internal shifting mass (m_p). The position of m_p is typically used to control the angular momentum of the vehicle, in turn affecting the vehicle pitch angle. Since neither the pitch angle nor the associated angular rate of the vehicle are controlled in this study, m_p is modelled within the system dynamics, but not included in the control vector.

Hydrostatic forces accounting for the gravity effect of the static mass (m_s) and of

the shifting mass (m_p) are computed respectively as:

$$G_s = m_s g \tag{6.22a}$$

$$G_p = m_p g. \tag{6.22b}$$

The buoyancy force, accounting for both the constant volume of the hull (∇_h) and for the variable volume associated with the VBD, is computed as:

$$B = B(\nabla_h) + B(u_{VBD}) = \rho g \nabla_h + \rho g u_{VBD}. \tag{6.23}$$

In this preliminary study, the volume of the glider hull was assumed to be constant and independent of the vehicle's depth. More advanced formulations that can accommodate hull compression due to pressure and temperature can be considered when enhancing this application [187, 72].

Next, control forces were considered. Each stern-plane j generates an additional lift force parallel to z_b [191], computed as:

$$F_{\delta_j} = K_{F_\delta} K_{u_\delta} V_r^2 \delta_j, \tag{6.24}$$

with K_i representing wing-specific hydrodynamic coefficients and δ_j the control surface deflection angle, with $j \in [1,2]$. Forces and terns are reported in Fig. 6.13. Finally, m_1 and m_3 represent the sum of the overall dry glider mass ($m_s + m_p$) and the added mass along the longitudinal and vertical body axes, computed approximating the glider hull to a prolate spheroid.

This model leverages three assumptions, as follows. In profiling operations, as the VBD is actuated over a time span of 30 [s] or more to save energy, the actuator dynamics are neglected [187]. It is assumed that the body-fixed velocities are measured, (for instance) by means of a DVL. Finally, pitch angle and angle of attack changes are negligible, as it is standard practice for the pilots to tune flight parameters such as to ensure a constant glide slope angle.

The corresponding dynamical model, derived based on previous works [20, 187,

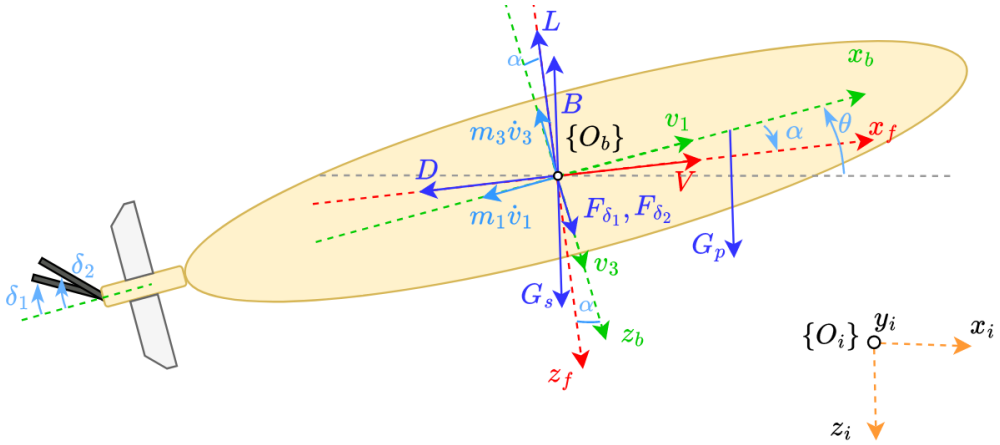


Figure 6.13: Underwater Glider dynamics in the sagittal plane with restoring forces $(G_s, G_p, B(\nabla_h))$, hydrodynamic forces (L, D), control forces $(B(u_{VBD}), F_{\delta_1}, F_{\delta_2})$ and inertial forces (including added mass). The forces are modelled in the inertial frame (origin $\{O_i\}$, in orange), body-fixed frame (origin $\{O_b\}$, in green) and flow-fixed frame (origin $\{O_b\}$, in red), © 2024 Elsevier [23].

192], is described as:

$$\begin{cases} \dot{v}_1 = \dfrac{1}{m_1}(-D\cos\alpha + L\sin\alpha + \sin\theta(B - G_s - G_p)) & \text{(6.25a)} \\ \dot{v}_3 = \dfrac{1}{m_3}(-D\sin\alpha - L\cos\alpha + \cos\theta(-B + G_s + G_p) + F_{\delta_1} + F_{\delta_2}) & \text{(6.25b)} \end{cases}$$

with the state-space vector defined as $\boldsymbol{x} = [v_1, v_3]^T$, and $\boldsymbol{u} = [u_{VBD}, \delta_1, \delta_2]^T$ denoting the control vector. The control signals are in turn embedded in the corresponding restoring and hydrodynamic forces as $u_1 = B(u_{VBD})$, $u_2 = F_{\delta_1}(\delta_1)$ and $u_3 = F_{\delta_2}(\delta_2)$. Given the steady-state nature of the gliding phases, hydrodynamic forces are computed through the quasi-steady state approximation as:

$$D \approx (K_{D0} + K_D\alpha^2)V_r^2 \quad \text{(6.26a)}$$

$$L \approx (K_{L0} + K_L\alpha)V_r^2 \quad \text{(6.26b)}$$

where $V_r = \sqrt{v_1^2 + v_3^2}$, and K_{D0}, K_D, K_{L0}, K_L represent the hydrodynamic coefficients, estimated through model identification or Computational Fluid Dynamics (CFD) anal-

yses.

For this study, the Seaglider (ogive model) geometric and hydrodynamic parameters derived in Chapter 4 were employed, as reported in Table 6.7. Finally, the values K_{F_δ} and K_{u_δ} were set as per standard glider stern-planes design [191].

Table 6.7: UG dynamical parameters.

Parameters	Values	Units	Explanation
m_s	44.9	kg	Vehicle hull mass
m_p	11.0	kg	Shifting mass
m_1	64.84	kg	Mass along x_b including added mass
m_3	99.43	kg	Mass along z_b including added mass
K_{D_0}	0.04	kg/m	Quasi-steady state drag (offset) coefficient
K_D	9.44	kg/(m rad)	Quasi-steady state drag coefficient
K_{L_0}	2.16	kg/m	Quasi-steady state lift (offset) coefficient
K_L	4.88	kg/(m rad)	Quasi-steady state lift coefficient
∇_h	0.054	m^3	Vehicle volume
K_{F_δ}	10.0	kg/(m rad)	Control surfaces coupling factor
K_{u_δ}	1.0	adimensional	Control surfaces scale constant
ρ	1027.5	kg/m^3	Seawater density

For this case study, interest lies in investigating the effect of a stern-plane jamming at a specific angle during the glide ascent phase, simulating the case of a frozen hinge or the presence of detritus. A more advanced control system accounting for saturated control inputs was designed. The control function with saturation was synthesised as follows:

$$\boldsymbol{u} = \begin{bmatrix} \overline{\sigma}_1 \tanh(\cdot) & & \\ & \ddots & \\ & & \overline{\sigma}_m \tanh(\cdot) \end{bmatrix} (\boldsymbol{K}\boldsymbol{e} + \boldsymbol{B}) \tag{6.27}$$

where $\overline{\sigma}_j$ represents the saturation limits of the j-th actuator, $\boldsymbol{K}$ the ANN weight, $\boldsymbol{B}$ the ANN bias (as justified in Section 6.5.2) and $\boldsymbol{e}$ the state-space error vector.

Given the complexity of this dynamical model, short training runs (i.e. composed of 1000 iterations) were observed leading to unsuccessful tests. After gradually increasing the number of iterations, the software tool synthesised a control function for the selected equilibrium $\boldsymbol{x}^\star = [0.300, -0.018]$ after 1944 training iterations (completed within 103 [s]). The resulting u_1, saturated to the desired actuator limit $\overline{\sigma}_1 = 1.0$, is illustrated in Fig. 6.14. The control function was obtained as $u_1 = \tanh(0.014 - 6.571x_1 + 4.912x_2)$, while the resulting CLF was synthesised as: $V = 0.264975x_1x_2 + 0.522009x_1^2 + 0.288912x_2^2$.

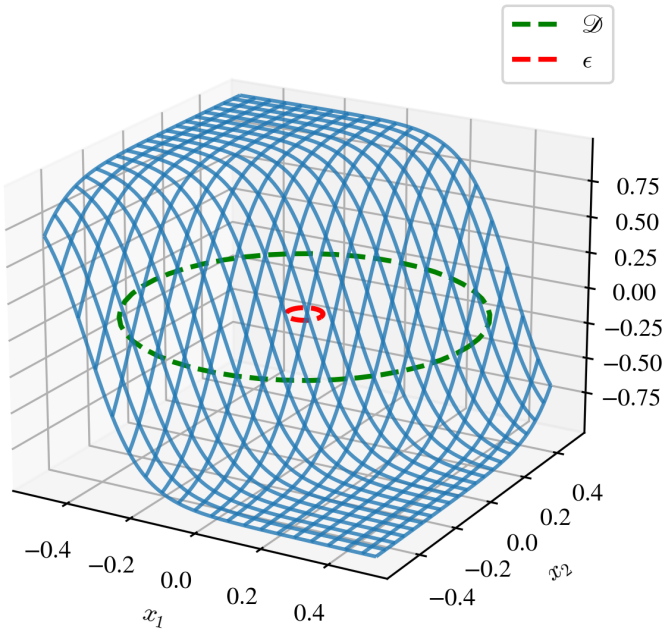
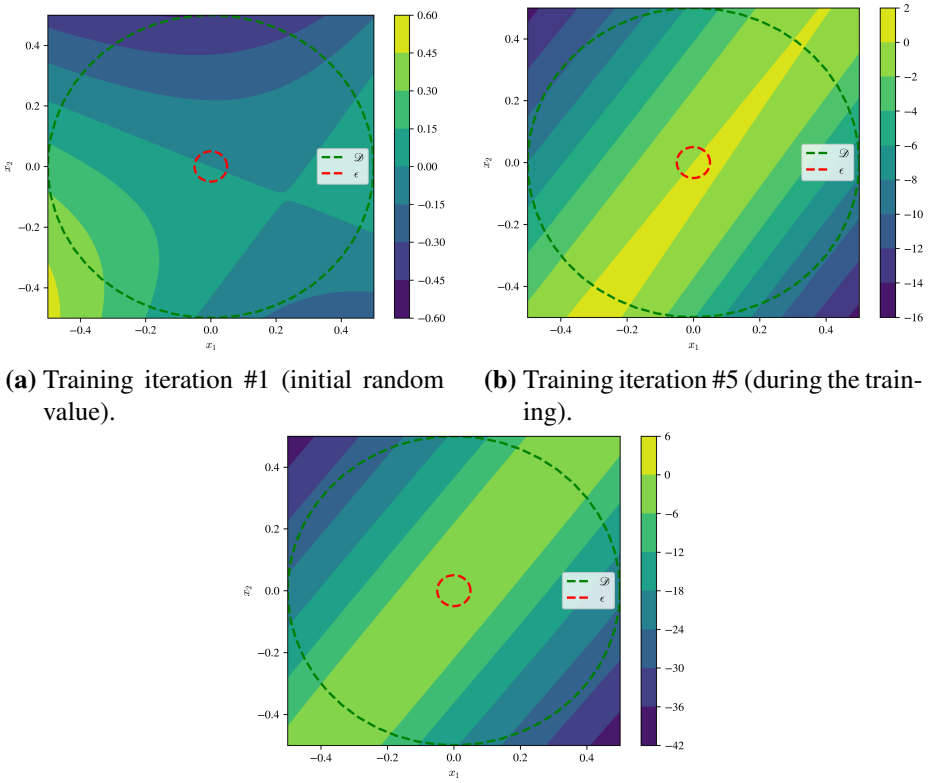


Figure 6.14: Synthesised control function for the Underwater Glider encompassing actuator saturation (u_1 with $\overline{\sigma}_1 = 1.0$), © 2024 Elsevier [23].

The evolution of the learning process is showcased by reporting the changing shape of the Lie derivative function for the nominal (faultless) dynamics (note that the evolution of the Lie derivatives associated to the fault modes follow the same trend). In Fig. 6.15a (learning iteration #1) and Fig. 6.15b (learning iteration #5) the contour lines of the Lie derivatives can been seen exhibiting positive values within $\mathcal{D}$. Upon convergence, the Lie derivative became negative definite over the entire domain, as shown in Fig. 6.15c. Finally, a closed-loop system verification test is reported in Fig. 6.16, showing the capability to track $\boldsymbol{x}^\star$ within the target bounds despite the jamming of δ_2 happening at t = 1.8 [s].



(a) Training iteration #1 (initial random value).

(b) Training iteration #5 (during the training).

(c) Training iteration #1944 (upon convergence).

Figure 6.15: Underwater Glider training: Lie derivative associated to the nominal dynamics at successive training iterations, © 2024 Elsevier [23].

6.6 Control law selection

The output of a pFT-ANLC training campaign is not a unique control law, but rather a series of synthesised control laws, all guaranteed to hold stabilising properties but with performances slightly different from one another. As each run is generated using a different seed, every training starts from a different initialisation of the ANN parameters and might converge to different minima. In other words, the pFT-ANLC method solves a feasibility problem, attempting to find any controller that stabilise a closed-loop fault-tolerant problem, and the solution might not be unique.

The scope of this section is to provide tools to select which control law, from among those generated, best fits the target operating objectives, such as prescribed values of steady-state tracking accuracy, or lower energy consumption.

A Linear-Quadratic (LQ) criterion is conventionally employed to assess control-system performance [97]. In this section, a criterion encompassing both a term related

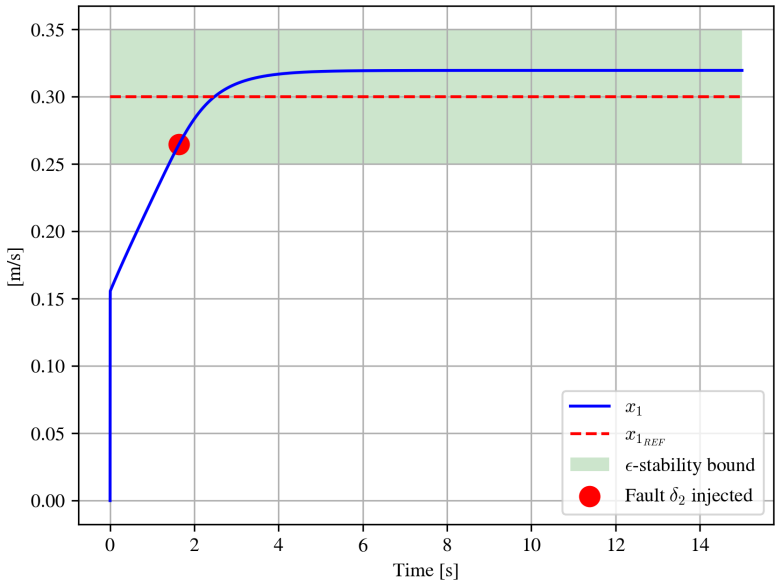


Figure 6.16: Underwater Glider verification test: surge speed when the control surface δ_2 jams at t=1.8 [s], © 2024 Elsevier [23].

to the reference tracking error and a factor associated to the control effort is proposed as:

$$J = \int_0^\infty (\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{e} + \boldsymbol{u}_{\boldsymbol{f}}^T \boldsymbol{R} \boldsymbol{u}_{\boldsymbol{f}}) dt \tag{6.28}$$

where $\boldsymbol{u_f}$ represents the control signal embedding the faults, $\boldsymbol{e}$ the reference tracking error with $\boldsymbol{e} = \boldsymbol{r} - \boldsymbol{x}$ and $\boldsymbol{r}$ the reference tracking value, while $\boldsymbol{Q}$ and $\boldsymbol{R}$ (symmetric positive semi-definite) are the weighting matrices. For convenience, the performance index J is split into the sum of two terms:

$$J = J_e + J_u \tag{6.29}$$

where J_e denotes the performance related to the tracking error and J_u the performance associated with the control input. J_e can be designed to evaluate the tracking ability of a controller, namely:

$$J_e = \bar{\boldsymbol{e}}^T \boldsymbol{Q_1} \bar{\boldsymbol{e}} + \bar{\boldsymbol{e}}_{bf}^T \boldsymbol{Q_2} \bar{\boldsymbol{e}}_{bf} + \bar{\boldsymbol{e}}_{af}^T \boldsymbol{Q_3} \bar{\boldsymbol{e}}_{af} + e_{st}^T Q_4 e_{st} \tag{6.30}$$

where:

- $\bar{\boldsymbol{e}}$ denotes the Root Mean Square Error (RMSE) of $\boldsymbol{e}$, computed as: $\bar{\boldsymbol{e}} = \text{RMSE}(\boldsymbol{e}) = \sqrt{\frac{\sum_{i=0}^{T}(r_i - x_i)^2}{T}}$, with T the number of available samples;
- the subscripts *bf* stands for *before fault* and *af* for *after fault*;
- the subscript *st* indicates the *settling time*;
- $\boldsymbol{Q_1}, \boldsymbol{Q_2}, \boldsymbol{Q_3}, Q_4$ are the weighting matrices.

Note that the first three terms of Eq. (6.30) express how accurately the control tracks the target value, while the last term indicates how fast the control reaches the steady-state value. The settling time is calculated as the time required to the dynamics to reach, and to remain bounded within a $\pm 2\%$ error from the target setpoint value.

To evaluate the performance of the control signals, the following cost function J_u is defined as:

$$J_u = \max(\boldsymbol{u})^T \boldsymbol{R_1} \max(\boldsymbol{u}) + \max\left(\frac{d\boldsymbol{u}}{dt}\right)^T \boldsymbol{R_2} \max\left(\frac{d\boldsymbol{u}}{dt}\right) + R_3 \int_0^T P(t)dt \tag{6.31}$$

where the first and second terms penalise the maximum values of the control forces and of the actuator rates, respectively, while the third factor accounts for the power consumption over time. In an application using thrusters as actuators, if the force generated by each thruster (F_i) is known, the overall power consumption at time $t = k_t$ can be computed as [175]:

$$P_{k_t} = (\sum_{i=0}^{m} |F_{i,k_t}|^{(3/2)}) \tag{6.32}$$

with m being the total number of available thrusters.

It follows that:

$$\boldsymbol{e} \in \mathbb{R}^n \tag{6.33a}$$

$$\boldsymbol{u} \in \mathbb{R}^m \tag{6.33b}$$

$$\boldsymbol{Q_1}, \boldsymbol{Q_2}, \boldsymbol{Q_3} \in \mathbb{R}^{n \times n} \tag{6.33c}$$

$$\boldsymbol{R_1}, \boldsymbol{R_2} \in \mathbb{R}^{m \times m} \tag{6.33d}$$

$$R_3, Q_4 \in \mathbb{R}. \tag{6.33e}$$

A quantitative example for the definition of a performance criterion is provided here. Given the system defined by Eq. (6.15), a simulation campaign consisting of 10 runs was executed. Two different cost function tunings were defined, and denoted as $\bar{J}^a$ and $\bar{J}^b$. The first tuning, denoted as $\bar{J}^a$, prioritised reference tracking performance, after the faults have occurred, over the control effort, i.e. the tracking error penalty terms were the highest. On the contrary, the second tuning priorities limiting of the overall power consumption, therefore tracking goals were relaxed while the power consumption penalty gain was the highest.

For instance, possible weighting terms associated with the first tuning choice were selected as follows:

$$\begin{aligned}
&\boldsymbol{Q_1^a} = \boldsymbol{Q_2^a} = \text{diag}(1.0,\, 1.0)\\
&\boldsymbol{R_1^a} = \boldsymbol{R_2^a} = \text{diag}(1.0,\, 1.0,\, 1.0)\\
&Q_4^a = R_3^a = 1.0\\
&\boldsymbol{Q_3^a} = \text{diag}(100.0,\, 100.0),
\end{aligned}$$

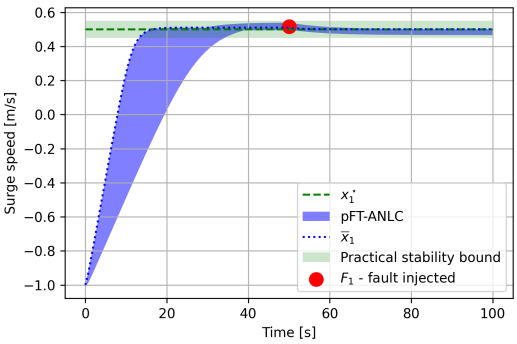
where the term $\boldsymbol{Q_3}$ denotes the tracking error penalty following the occurrence of the faults and Q_4 the settling time penalty.

In contrast, the control tuning that prioritises the minimisation of the overall power consumption, denoted with $\bar{J}^b$, can be selected by imposing the following tuning factors:

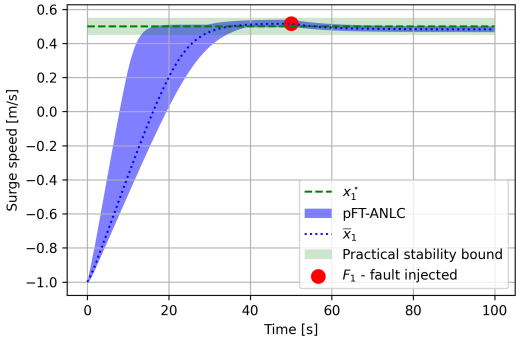
$$\begin{aligned}
&\boldsymbol{Q_1^b} = \boldsymbol{Q_2^b} = \boldsymbol{Q_3^b} = \text{diag}(1.0,\, 1.0)\\
&\boldsymbol{R_1^b} = \boldsymbol{R_2^b} = \text{diag}(1.0,\, 1.0,\, 1.0)\\
&Q_4^b = 1.0\\
&R_3^b = 100.0.
\end{aligned}$$

The closed-loop dynamics resulting from the two different tunings are compared here. Fig. 6.17a reports the vehicle surge speed associated with the tuning that prioritises minimisation of the reference tracking error following the fault, where $\bar{x}_1$ (dotted

line) denotes the controller minimising $\overline{J}^a$. Similarly, Fig. 6.17b illustrates the resulting dynamics of the more energy-conservative tuning $\overline{J}^b$. A magnified view of the surge dynamics following the injection of a fault at F_1 is reported in Fig. 6.18. The chosen control law associated with $\overline{J}^a$ minimises the tracking error following the occurrence of a fault, as desired.



(a) Tuning minimising the tracking error ($\overline{J}^a$) denoted with $\bar{x}_1$.



(b) Tuning minimising the overall power consumption ($\overline{J}^b$) denoted with $\bar{x}_1$.

Figure 6.17: Range of possible surge dynamics associated to the converged controllers with fault at F_1 at 50 [s], © 2024 Elsevier [23].

Finally, the forces generated by the three thrusters are illustrated in Fig. 6.19a and Fig. 6.19b for the tuning $\overline{J}^a$ and $\overline{J}^b$, respectively. As expected, tuning $\overline{J}^a$, leading to a faster converging dynamics, requires an excess of forces when compared to $\overline{J}^b$.

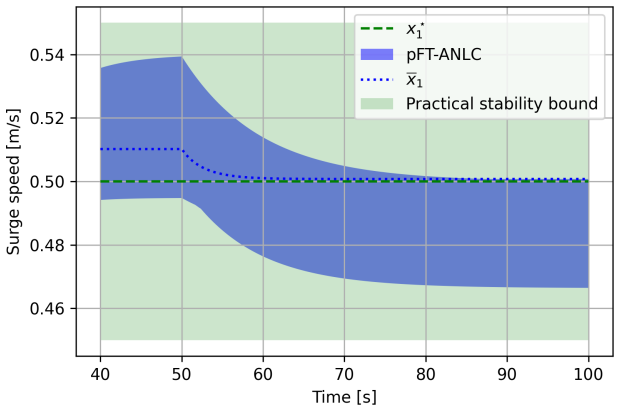
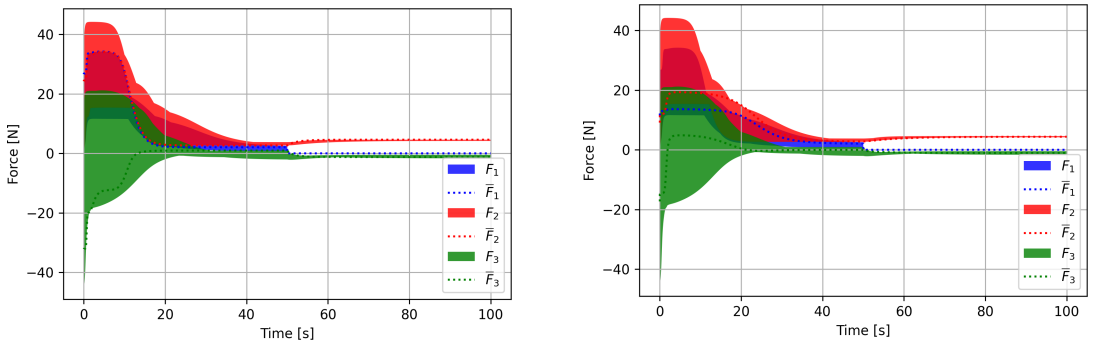


Figure 6.18: Magnified view of Fig. 6.17a: the selected controller minimises the tracking error ($x_1^\star - \bar{x}_1$) following a fault at F_1 at 50 [s] (tuning $\overline{J}^a$), © 2024 Elsevier [23].



(a) Tuning minimising the tracking error ($\bar{J}^a$) highlighted with dashed lines.

(b) Tuning minimising the overall power consumption ($\bar{J}^b$) highlighted with dashed lines.

Figure 6.19: Range of control efforts associated to the converged controllers with different tuning following a fault at F_1, © 2024 Elsevier [23].

6.7 Actuator loss of efficiency

In the previous sections, the case of a jammed control surface and of a complete loss of one or more actuators were investigated. While the case of a complete loss of actuator efficiency represents an interesting case study, at times, in real-life applications, faults appear as partial loss of actuator efficiency. This could be the case if minor electrical or mechanical issues arise, or if fishing debris or seaweed foul the thrusters, increasing the resistance to rotation and limiting the overall rate of rotation. In the specific case of gliders with missions extending for weeks or months, biofouling of marine growth on the control surfaces can reduce the efficiency of the lift generated by the stern-planes or the thrust generated by a propeller. These occurrences do not lead to a complete loss of actuator functionality but rather limit the delivered actuator effect, and can be modelled as a factor that multiplies the nominal control effort.

In line with the remainder of this work, the aim of this section is the design of a control law that stabilises the system around a target equilibrium when the loss of efficiency of an actuator occurs. To this end, the method introduced in Section 6.3.1 and Section 6.3.2 can be extended by accounting for the *efficiency* of the j-th actuator, modelled as a real (continuous) variable $\mu_j \in [0,1]$. In practical terms, the Lyapunov conditions Eq. (5.4) can be extended to include non-binary faults. This formally results

in:

$$V(\boldsymbol{x}^\star) = 0, \tag{6.34a}$$

$$V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}, \tag{6.34b}$$

$$\dot{V}_n(\boldsymbol{x}) < 0 \wedge \{\dot{V}_{\mu_j}(\boldsymbol{x},\boldsymbol{u}) < 0\}_{j=1}^{d} \quad \forall \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}, \tag{6.34c}$$

where $\dot{V}_{\mu_j}(\boldsymbol{x},\boldsymbol{u})$ represents the Lie derivative in the presence of a fault at the actuator j, with actuator efficiency μ_j (with $\mu_j \in \boldsymbol{\mu}$). The Falsifier conditions (6.9) can therefore be modified as follows:

$$\exists(\boldsymbol{x} : \boldsymbol{x} \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}, \boldsymbol{\mu} : \boldsymbol{\mu} \in [0,1]) \implies \Big(V(\boldsymbol{x}) \leq 0 \ \vee \ \dot{V}_n(\boldsymbol{x},\boldsymbol{u}) \geq 0 \ \vee \{\dot{V}_{\mu_j}(\boldsymbol{x},\boldsymbol{u}) \geq 0\}_{j=1}^{d}\Big), \tag{6.35}$$

where any instance of ($\boldsymbol{x}$, $\boldsymbol{\mu}$) that satisfies these conditions would falsify the candidate CLF, and hence the proposed control law, triggering a restart of the training procedure.

6.7.1 Case Study 4: Autonomous Underwater Vehicle with reduction in thruster efficiency

The concept of how to tackle the loss of actuator efficiency is illustrated in the case of the two dimensional AUV introduced in Section 6.5.2. The model includes a possible loss of efficiency (μ_1) on the first actuator (F_1), and can be expressed as:

$$\begin{cases} \dot{x}_1 = \dfrac{-X_u x_1 - X_{uu} x_1^2 + \mu_1 F_{1,x} + F_{2,x}}{m} & \text{(6.36a)} \\ \dot{x}_2 = \dfrac{-N_r x_2 - N_{rr} x_2^2 + (-F_{1,x} l_{1,y} + F_{1,y} l_{1,x})\mu_1}{J_z} + \dfrac{(-F_{2,x} l_{2,y} + F_{2,y} l_{2,x}) + (-F_{3,x} l_{3,y} + F_{3,y} l_{3,x})}{J_z} & \text{(6.36b)} \end{cases}$$

where the same model coefficients listed in Table 6.4 are employed.

The result of a successful training run with target $\mathbf{x}^\star = [0.5, 0.0]^T$ and $\varepsilon = 0.025$ is presented here. To verify the correctness of the method, the efficiency μ_1 was modified over time in line with the profile reported in Fig. 6.20, with μ_1 set to arbitrary values

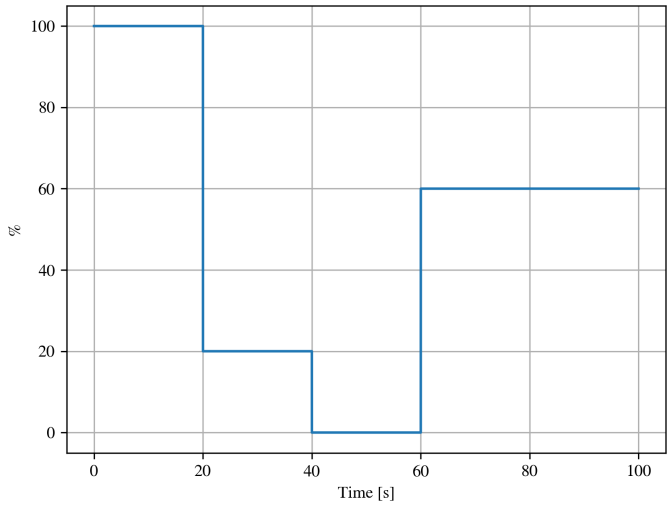


Figure 6.20: Partial loss of efficiency for the AUV case study: efficiency of the first actuator (μ_1) over time, © 2024 Elsevier [23].

ranging from 100% (fully functioning) to 0% (fully compromised). The corresponding forces applied by the controller during a closed-loop test can be viewed in Fig. 6.21, with the resulting dynamics illustrated in Fig. 6.22a and Fig. 6.22b.

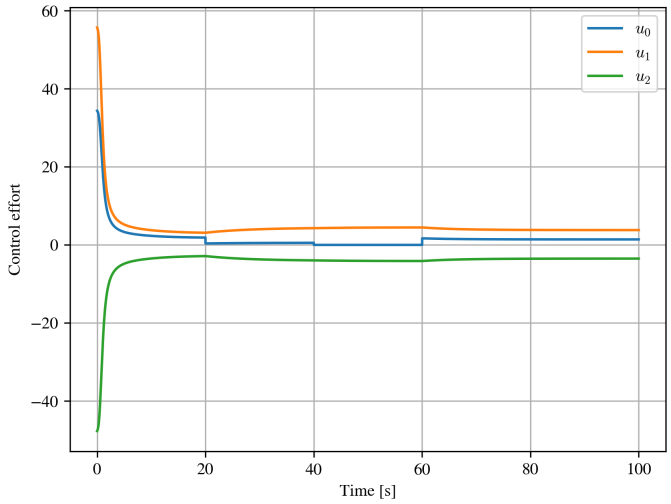
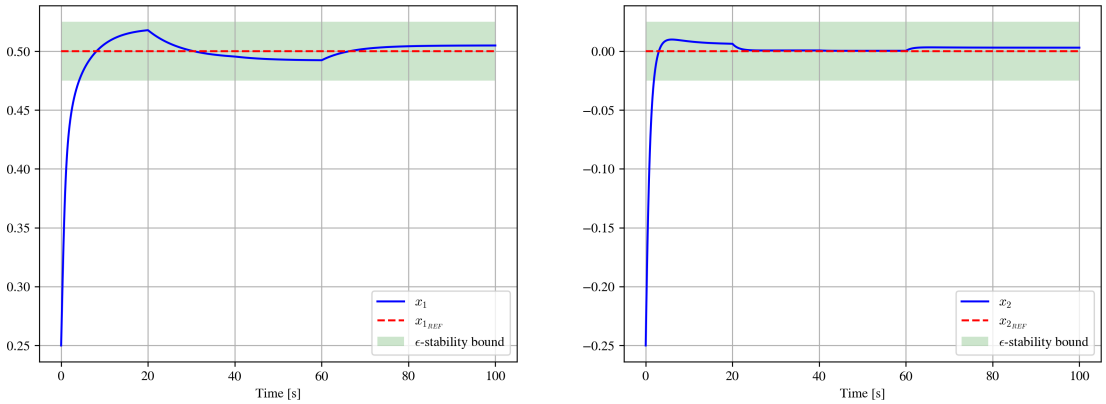


Figure 6.21: Partial loss of efficiency for the AUV case study: control input forces, © 2024 Elsevier [23].

It can be noted that despite the efficiency of the actuator is being arbitrarily varied over different values in the range (100% to 0%), the control system is able to bound the dynamic response within the prescribed practical stability threshold.



(a) Surge dynamics (x_1). **(b)** Yaw rate dynamics (x_2).

Figure 6.22: Partial loss of efficiency for the AUV case study: dynamic response to varying thruster efficiency (e_1), © 2024 Elsevier [23].

6.8 Comparison with $\mathcal{H}_\infty$ control method

In this section, results of the synthesis of pFT-ANLC laws for the two dimensional AUV are compared to the state-of-the-art method within the frame of Passive fault-tolerant Control for AUVs, namely the $\mathcal{H}_\infty$ robust controller.

As was previously introduced in Section 2.5.8, the $\mathcal{H}_\infty$ method applied to FTC problems aims at finding a unique control law for a linear (or linearised) system affected by uncertainties and/or multiple operating modes. In more detail, given the state-space vector $\boldsymbol{x} \in \mathbb{R}^n$, the control vector $\boldsymbol{u} \in \mathbb{R}^m$, the measurement output $\boldsymbol{y} \in \mathbb{R}^r$, the exogenous input vector $\boldsymbol{w} \in \mathbb{R}^k$ and the performance output vector $\boldsymbol{z} \in \mathbb{R}^l$, a state-space representation of the generalised plant (P_g) can be formulated as:

$$\begin{cases} \dot{x} = Ax + B_1 w + B_2 u \\ z = C_1 x + D_{11} w + D_{12} u \\ y = C_2 x + D_{21} w + D_{22} u \end{cases} \tag{6.37}$$

The $\mathcal{H}_\infty$ problem synthesis aims at minimising the $\mathcal{H}_\infty$ norm of the transfer matrix $\boldsymbol{T_{zw}}$ (from w to z), namely $\|\boldsymbol{T_{zw}}\|_\infty$, where $\|\boldsymbol{T_{zw}}\|_\infty = \sup\{\sigma_{\max}(\boldsymbol{T_{zw}}(j\omega)) : \omega \; in \; \mathbb{R}\}$, with $\sigma_{\max}(\cdot)$ denotes the maximum singular value. Intuitively, the $\|\boldsymbol{T_{zw}}\|_\infty$ norm can be viewed as the maximum energy gain that the controller attempts to minimise [118]. Given the generalised plant P_g, the goal is to design an (output) feedback controller in the form $\boldsymbol{u} = \boldsymbol{K}(s)\boldsymbol{y}$, where the dependency from the Laplace operator s indicates that

the function can be frequency-dependent. The block diagram used for the synthesis of the control law is reported in Fig. 6.23.

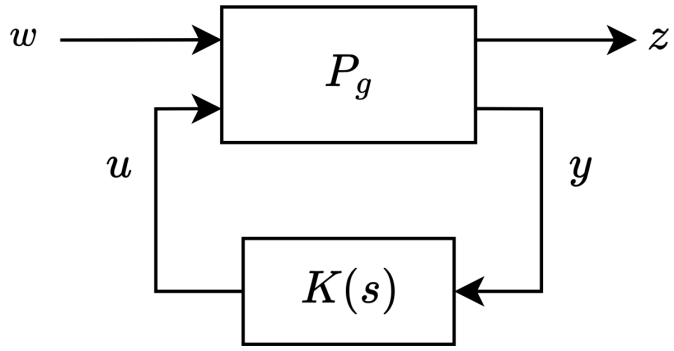


Figure 6.23: $\mathcal{H}_\infty$ control scheme.

The control synthesis process starts by selecting a fixed-structure control law (e.g. a PID, or a static gain or a state-space model), with the parameters updated iteratively until a stabilising solution is obtained or, optionally, until further design requirements are met. Typical examples of these design requirements are obtaining a prescribed settling time to a step response input, having a maximum response overshoot, or placing the closed-loop poles in a specific region of the complex plane. Design requirements can be formulated as hard or soft optimisation constraints, with the solution being subjected to satisfying of the hard tuning constraints while trying to additionally satisfy the soft tuning constraints. Thus, given a controller structure with a vector of free control parameters to tune $\boldsymbol{\theta}$, by providing the design goals as soft requirements (expressed as a normalised value $\boldsymbol{f}(\boldsymbol{\theta})$) and hard constraints (expressed as a normalised value $\boldsymbol{g}(\boldsymbol{\theta})$), the optimizer iteratively solves the following problem:

$$\begin{aligned} &\min(\max \boldsymbol{f}(\boldsymbol{\theta}))) \\ &\text{subject to:} \\ &\max \boldsymbol{g}(\boldsymbol{\theta}) < 1 \\ &\boldsymbol{\theta}_{min} < \boldsymbol{\theta} < \boldsymbol{\theta}_{max} \end{aligned} \tag{6.38}$$

where $\boldsymbol{\theta}_{min}$ and $\boldsymbol{\theta}_{max}$ are the minimum and maximum values of the free control parameters. In other words, the optimal solution is returned when all the hard constraints (including closed-loop stability) are strictly verified by a choice of the free variable $\boldsymbol{\theta}$,

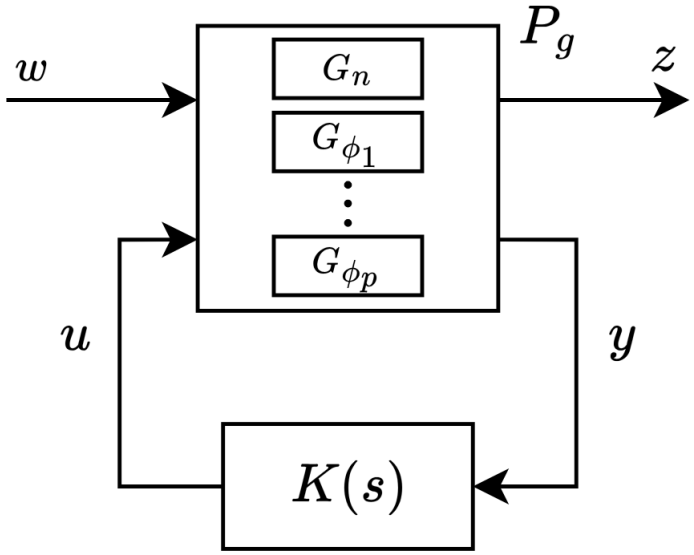


Figure 6.24: $\mathcal{H}_\infty$ control scheme with multiple plant models, with $K(s)$ denoting a generic control law.

with optional verification of the additional soft constraints.

Similar to the pFT-ANLC formulation, the $\mathcal{H}_\infty$ problem is set up as the simultaneous minimisation of the $\mathcal{H}_\infty$-norm of $\|\boldsymbol{T}_{\boldsymbol{zw}}\|_\infty$ of a set of plants, each corresponding to one operating mode, i.e. nominal mode or one fault modes [100]. In detail, given a plant (G_n) corresponding to the nominal faultless system and given d plants, each one corresponding to one admissible fault (ϕ_i), the controller aims at minimising the worst case scenario of $\|\boldsymbol{T}_{\boldsymbol{zw}_j}\|_\infty$, with $j = n, 1, ..., d$. A corresponding illustration of the control scheme is reported in Fig. 6.24, with P$= \{G_n \cup G_{\phi_i}\}$ with $i = 1, ..., d$.

In the specific case of the AUV dynamics described in Eq. (6.15), the control synthesis was formulated as an optimisation problem with the objective of minimising the tracking error (**e**) between the reference signal (**r**) and the output of the augmented plant (**y**). In this study, the exogenous input vector $\boldsymbol{w}$ coincides with the reference signal $\boldsymbol{r}$, the control vector $\mathbf{u} = [F_1, F_2, F_3]^T$, the measured output $\mathbf{y} = [y_1, y_2]^T$ and $\mathbf{z} = [\mathbf{y}, \mathbf{u}, \mathbf{e}]^T$. The system state and control Jacobians (**A** and **B**, respectively) were obtained through linearisation as:

$$\mathbf{A}|_{\mathbf{x}=\mathbf{x}^\star} = \begin{bmatrix} \frac{-X_u - 2X_{uu}x_1^\star}{m} & 0 \\ 0 & \frac{-N_r - 2N_{rr}x_2^\star}{J_z} \end{bmatrix} \tag{6.39}$$

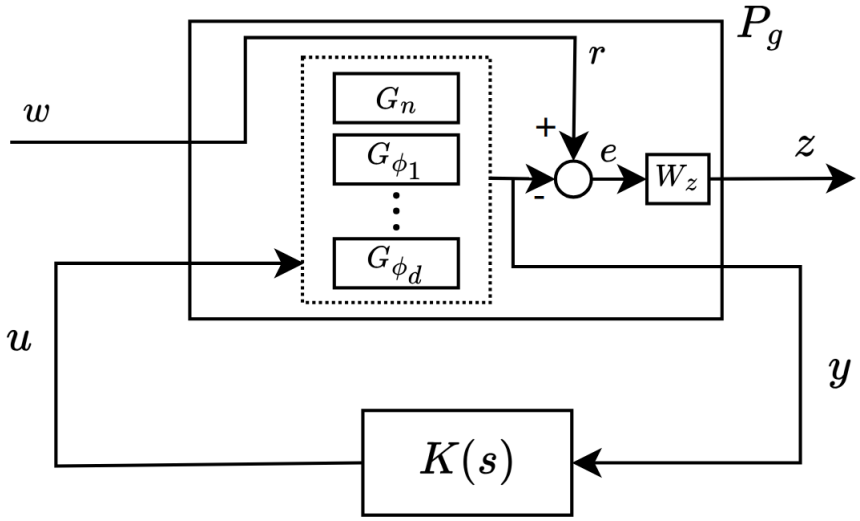


Figure 6.25: $\mathcal{H}_\infty$ for AH1 control scheme, with $K(s)$ denoting a generic control law.

$$\mathbf{B} = \begin{bmatrix} \frac{\sin\alpha_1}{m} & \frac{\sin\alpha_2}{m} & \frac{\sin\alpha_3}{m} \\ \frac{-l_{1y}\sin\alpha_1 + l_{1x}\cos\alpha_1}{J_z} & \frac{-l_{2y}\sin\alpha_2 + l_{2x}\cos\alpha_2}{J_z} & \frac{-l_{3y}\sin\alpha_3 + l_{3x}\cos\alpha_3}{J_z} \end{bmatrix} \tag{6.40}$$

with the state-to-output and control-to-output matrices selected as:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.41}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{6.42}$$

For this study, the case of three faults at thrusters were considered (at most one at any one time), leading to the definition of four plants, namely $P_g = \{G_n, G_{\phi_1}, G_{\phi_2}, G_{\phi_3}\}$. Each of the four systems G_j was then evaluated around the target operating point $\boldsymbol{x}^\star = [0.5, 0.0]^T$. Finally, the overall control scheme is reported in Fig. 6.25.

For this analysis, three $\mathcal{H}_\infty$ optimisation problems with different requirements were formulated and are detailed in Table 6.8. The requirements were formulated to devise controllers with different performance characteristics.

To tune the gain, a target response time of 10 [s] was set (a realistic value for the application being considered) while the maximum tolerated steady-state error was

varied from 40% in the conservative tuning scenario, to a more stringent 5% in the aggressive tuning scenario.

Additionally, in the conservative scenarios, an extra constraint to limit the control effort was considered. Actuator saturation was set at a limit of 37.1 [N], chosen as the maximum force generated by a *BlueRobotics T200* thruster. This thruster model was selected in this example due to its widespread use in underwater robotics applications.

The aggressive scenario was solved within 1 [s] of computational time. However, the synthesis failed to converge when the actuator saturation limit of 37.1 [N] was imposed, despite 10 rounds of optimisation were attempted. Each attempt started from a different seed, to ensure that the optimisation problem comprised different initial values of the free control parameters. The conservative solution was thus obtained by incrementally increasing the control effort saturation limit from 37.1 [N]. A saturation of 880.0 [N] was found to be the first value that enabled the $\mathcal{H}_\infty$ synthesis to successfully converge. The conservative synthesis scenarios were solved within a maximum of 208 iterations and terminated within 10 [s] of computational time.

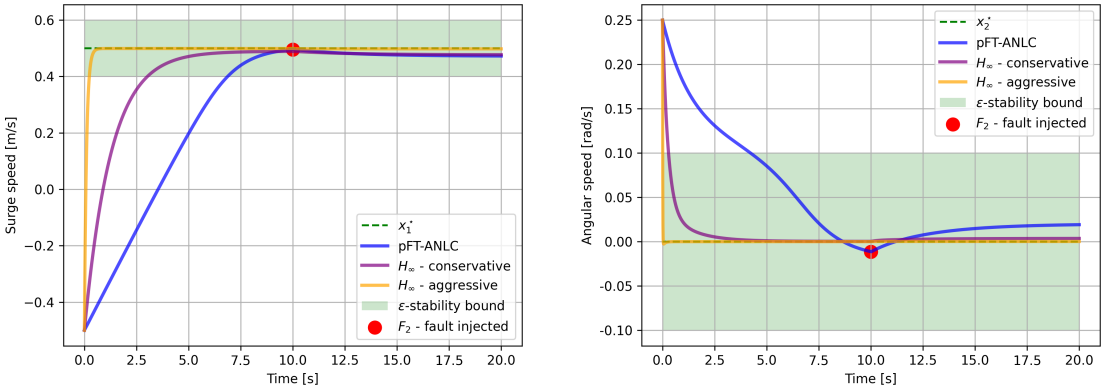
Table 6.8: $\mathcal{H}_\infty$-control tuning design specifications and synthesis results.

Control tuning	Response time [s]	Max. steady-state error [%]	Control Saturation	Controller found
aggressive	10	5	None	Yes
conservative	10	40	37.1 [N]	**No**
conservative	10	40	880.0 [N]	Yes

Next, the devised $\mathcal{H}_\infty$ laws were compared with control laws synthesised via pFT-ANLC. The pFT-ANLC solution was obtained by selecting the same architecture hyperparameters previously described in Table 6.5, while adding the output saturation value as 37.1 [N].

The resulting closed-loop responses when a fault at F_2 occurs can now be compared. Fig. 6.26a reports the closed-loop surge dynamics, while Fig. 6.26b reports the corresponding angular rate dynamics. It can be appreciated how the synthesised pFT-ANLC law (in blue) are linked to dynamics that show qualitatively comparable steady-state errors to those of the $\mathcal{H}_\infty$ dynamics, while they are slower in terms of convergence time. As expected, the *aggressive* $\mathcal{H}_\infty$ tuning (in orange) shows faster convergence rate and improved tracking performance both before and after the occur-

rence of the fault at the cost of an increased control effort, with F_1 reported as an example in Fig. 6.27. The *aggressive* $\mathcal{H}_\infty$ tuning in fact requires a control effort that is two orders of magnitude higher than that of pFT-ANLC (F_2 and F_3 follow comparable trends). Additionally, Fig. 6.27 highlights how pFT-ANLC is the only controller that guarantees feasible control actions by strictly respecting the actuator saturation limit.



(a) Surge dynamics (x_1). **(b)** Yaw rate dynamics (x_2).

Figure 6.26: Control laws comparison for AUV case study: pFT-ANLC (blue) vs multiple $\mathcal{H}_\infty$ tuning (purple and orange), © 2024 Elsevier [23].

These results highlight how the proposed framework can synthesise control laws that are qualitatively comparable with well established robust control methods, while providing additional benefits from the perspective of formal guarantees. Despite $\mathcal{H}_\infty$ method benefiting from highly optimised commercial tools resulting in faster tuning of

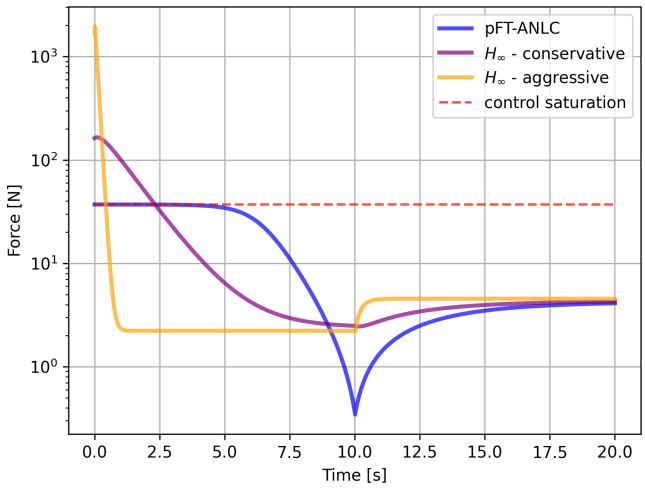


Figure 6.27: Control laws comparison for AUV case study: F_1 control effort with pFT-ANLC strictly respecting the desired actuator saturation, © 2024 Elsevier [23].

control laws (10 seconds for the $\mathcal{H}_\infty$ tuning, against the 92 seconds of the pFT-ANLC, in this study), pFT-ANLC provides benefits from both the theoretical and practical perspectives. First, pFT-ANLC does not rely on the linearisation of the system dynamics, representing an advantage in every application catering for highly nonlinear and coupled dynamics, such as those characterising the underwater environment. Second, as the pFT-ANLC synthesises a CLF, additional information regarding the properties of the nonlinear closed-loop system, such as the shape and dimensions of the ROA is available. An investigation about the properties of the ROA of an equilibrium controlled with a pFT-ANLC-derived control laws is outlined in the following section.

6.8.1 Region of Attraction shaping and estimation

Conventionally, in control design applications, interest lies in providing a control law with a ROA that is as large as possible. To this aim, pFT-ANLC allows shaping the ROA by calibrating α_4 in Eq. (6.7). To showcase the ROA shaping procedure, two distinct tunings of the loss function parameters Eq. (6.7) were trialled, and will be compared and discussed in this section.

An initial training was performed with $\alpha_4 = 0.0$ (disabling the ROA tuning factor), and, following, a second training encompassing $\alpha_4 = 1.0$ and $\alpha_{ROA} = 100.0$ was carried out. Upon completion of the training, the resulting ROAs associated with the two CLFs were calculated. It is worth recalling that the exact calculation of the ROA for arbitrarily high dimensional systems exhibiting complex nonlinear phenomena is not a trivial task [197]. In this work, calculating an estimate of the ROA, i.e. an inner approximation (conservative), is deemed a sufficient solution. Such an approximation can be calculated as a set $\Omega_\beta \subset \mathcal{D}$ such that:

$$\Omega_\beta = \{V(\boldsymbol{x}) \leq \beta_\rho\} \tag{6.43}$$

where, $\mathcal{D} = \varepsilon \leq \|\boldsymbol{x}\|_2 \leq \overline{\gamma}$, and β_ρ defines the maximum level set of $V(\boldsymbol{x})$, coinciding with the estimate the ROA.

To calculate such a value β_ρ, SMT solvers can be employed. A recursive SMT formula can be evaluated, as:

$$\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{D}) \Rightarrow (V(\boldsymbol{x}) > \beta_\rho). \tag{6.44}$$

To this aim, a simple algorithm recursively increasing the value β_ρ was devised and it is here reported as Algorithm 3. In such algorithm, $\beta_s \in \mathbb{R}$ represents the constant incremental step. The procedure terminates when a CE of Eq. (6.44) is returned, or when the upper limit of the domain is reached, namely $\beta_\rho = \beta_{max}$. When $\beta_\rho = \beta_{max}$ holds, it equates to the case when the ROA approximates $\mathcal{D}$, as it will be shown in the following analysis.

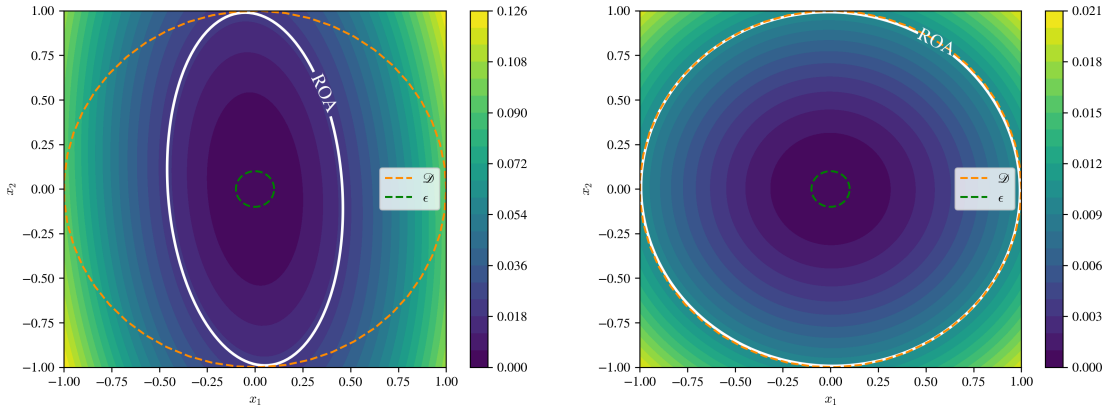
Algorithm 3 Estimation of the Region Of Attraction

```
1:  function LYAPUNOV CHECK(V_c^S, D, β_ρ)
2:      Check condition Eq. (6.44)
3:      return unsat or CE
4:
5:  function MAIN()
6:      Input: V_c^S, D, β_s
7:      β_ρ : V(‖x‖_2 = ε) = β_ρ
8:      β_max : V(‖x‖_2 = γ̄) = β_max
9:      repeat
10:         CE ← LYAPUNOV CHECK(V_c^S, D, β_ρ)
11:         if unsat then β_ρ = β_ρ + β_s
12:     until CE or β_ρ = β_max
```

Algorithm 3 is trialled over two CLFs synthesised with different loss function tunings. The ROA corresponding to a CLF synthesised without tuning factor (i.e. $\alpha_4 = 0$) is reported in Fig. 6.28a, while the ROA associated to the use of a tuning factor (i.e. $\alpha_4 = 1$, $\alpha_{ROA} = 100$) is shown in Fig. 6.28b. It should be noted that, in the second tuning, the ROA approximately overlaps the whole domain $\mathcal{D}$, as circular level sets were enforced during the training.

The analyses illustrated thus far highlighted how the joint synthesis of nonlinear control laws and CLFs proposed in this study can be employed to generate control laws capable of accounting for practical nonlinear effects, such as actuator saturation. As an additional benefit, the resulting ROA can not only be estimated, but can be shaped by means of a dedicated tuning parameter during the synthesis stage. These elements represent key advantages in further assessing closed-loop dynamic properties when

compared to the robust methods traditionally employed in the field.



(a) CLF synthesised *without* ROA tuning factor.

(b) CLF synthesised *with* ROA tuning factor.

Figure 6.28: pFT-ANLC controller — different CLF tuning factors. Domain boundaries in dashed lines and ROA in white solid line, © 2024 Elsevier [23].

6.9 Conclusions

In this chapter, a novel, automated method to design pFTC control laws was presented. The proposed approach relies on the ANLC architecture described in Chapter 5 to synthesise a unique control function with fixed gain that guarantees the ε-stability of the target equilibrium. Both linear and nonlinear control laws were synthesised for benchmark systems of increasing complexity. The approach was shown to be capable of maintaining the system dynamics within prescribed stability bounds, without relying on external information flow from a Fault Detection and Isolation system. The method further benefits from an intuitive understanding of the tolerated performance degradation, expressed as the maximum state-space deviation from the target equilibrium value.

A dedicated software tool, used to generate the results presented, was described and released open-source. The proposed tool was run on standard office laptops without demanding performance requirements.

Faults of different natures, ranging from full to partial loss of actuator efficiency, to control surfaces jamming were analysed, with encouraging outcomes. Finally, a method to account for actuator saturation was illustrated, and a comparison with different control laws tuned through a robust $\mathcal{H}_\infty$ technique was presented. As a result, the

pFT-ANLC was the only method shown capable to successfully account for actuator saturation.

After having completed the design of the desired pFTC method, in the following chapter, a complete case study entailing the control of a faulty AUV will be investigated.

Chapter 7

OpenMAUVe and pFT-ANLC: an AUV case study

7.1 Introduction

This chapter outlines the modelling-simulation-control process aimed at controlling an underwater vehicle affected by actuator faults. First, the vehicle dynamics are implemented within the OpenMAUVe simulator introduced in Chapter 4. Next, the design of a control law using the pFT-ANLC method is carried out, in accordance with the early investigations reported in Chapter 6. Two simulation scenarios are performed and the results obtained are analysed. The analysis of the results and the limitations of the proposed methods and tools are presented, with proposals for and the direction of future research work reported discussed.

7.2 Simulator architecture definition

The *OpenMAUVe* simulator, introduced in Chapter 4, represents a quick prototyping tool that can be employed to analyse the dynamics of a wide range of underwater vehicles. Additionally, the *OpenMAUVe* simulator can be used in control applications to verify the stability of closed-loop control laws, or to support the fine-tuning of control gains. To showcase the versatility of the simulator, this section outlines the design steps required to simulate the dynamics of an AUV.

For this case study, a neutrally buoyant AUV moving in an horizontal plane was considered. The AUV employed for this study is inspired by both the hover-capable

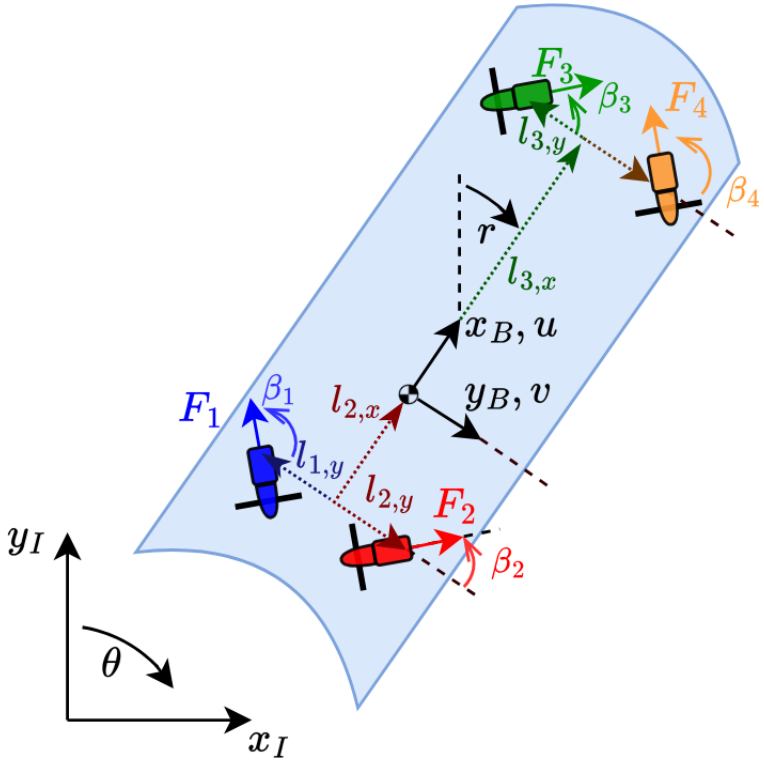


Figure 7.1: AUV vehicle model with four (fixed) thrusters moving over the horizontal plane.

AUV introduced in Section 6.5.2, and the ROVs with fixed thrusters positioned at the extremities of the vehicle [241, 242]. The vehicle, featuring four fixed thrusters, is illustrated in Fig. 7.1.

To investigate the AUV dynamics, the simulator architecture was designed as detailed below. First, the *ModelAUV* class was created, encompassing dynamic elements as in the restoring forces, the hydrodynamic forces and the control forces. Additionally, Coriolis and centripetal terms were accounted for within the *Modelica.Mechanical.Multibody.Body* class from the MSL. Moreover, four input interfaces were included to allow the flow of the control forces generated by the four thrusters F_1, F_2, F_3 and F_4. Finally, virtual sensors were introduced to measure the vehicle's linear velocities, angular velocities and attitude angles. The overall definition of the *ModelAUV* class is shown in Fig. 7.2.

Next, the *ModelAUV* class was embedded within a closed-loop architecture, allowing control of the surge speed, the sway speed, the angular rate around the z-body axis, and the yaw angle. The closed-loop architecture was defined in a class named

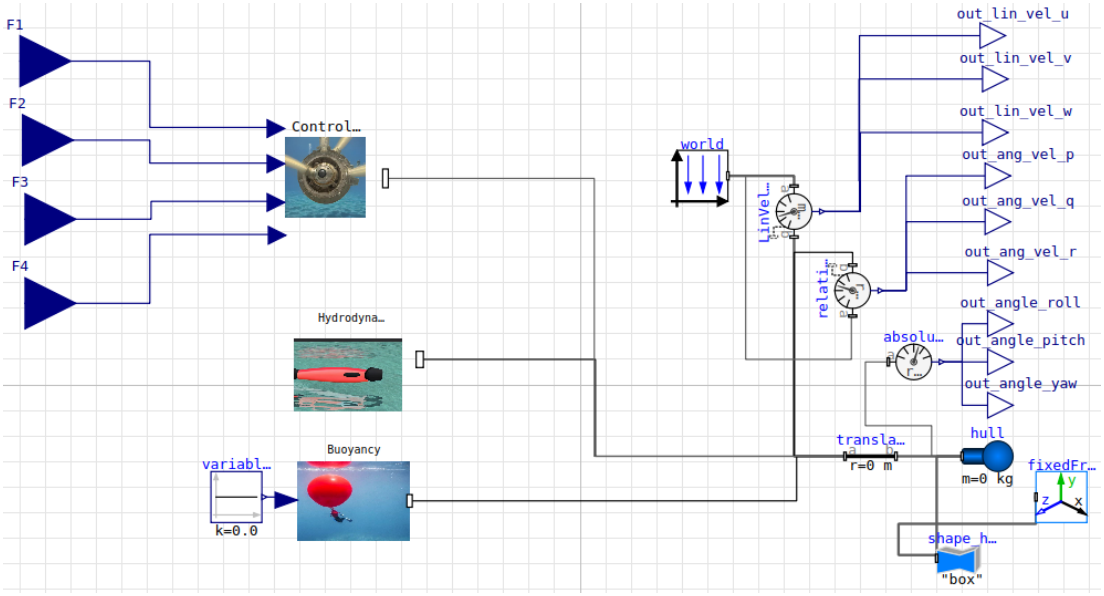


Figure 7.2: *OpenMAUVe* simulator architecture: *ModelAUV* class.

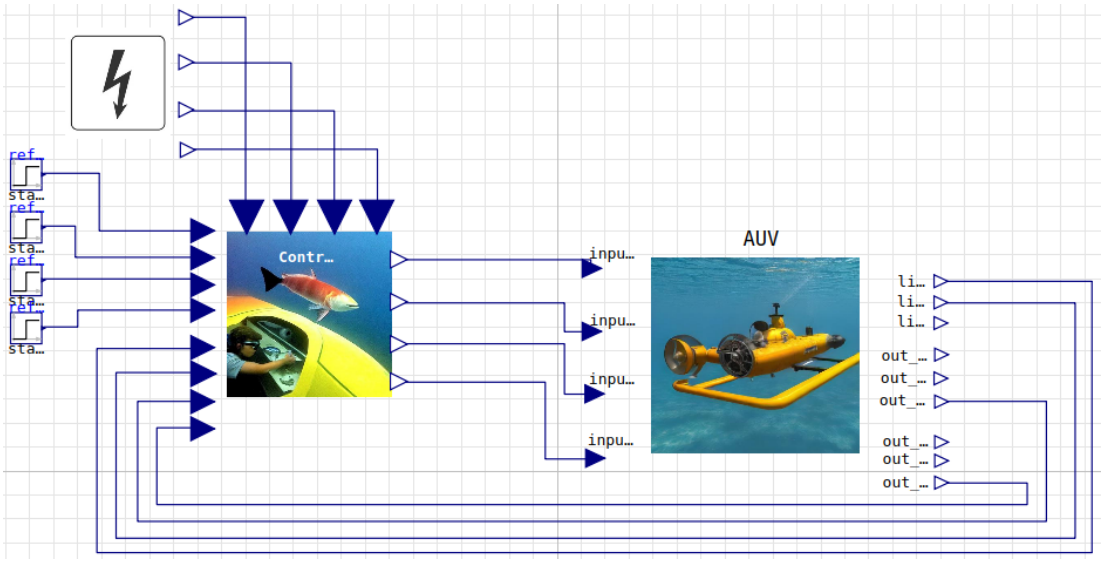


Figure 7.3: *OpenMAUVe* simulator architecture: *SimulateAUV* class encompassing the AUV dynamics (*ModelAUV* class, right hand-side), the control laws (*ControlAUV* class, lower left-hand side) and the signals of the faults (*FaultsAUV* class, top left-hand side).

SimulateAUV, illustrated in Fig. 7.3. The *SimulateAUV* class encompasses three elements: the class implementing the AUV dynamics (*ModelAUV*), the class dedicated to the definition of the control laws (*ControlAUV*) and the class required to manage the fault signals (*FaultsAUV*).

The class defining the control laws, named *ControlAUV*, receives the target mea-

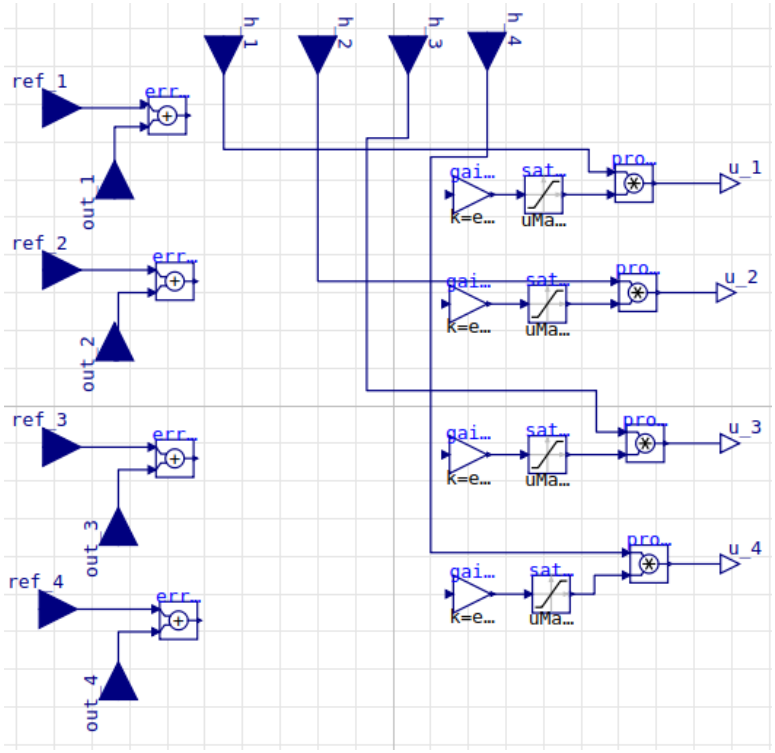


Figure 7.4: *OpenMAUVe* simulator architecture: *ControlAUV* class.

sured states and the measured states as inputs, and outputs the force commands to the four thrusters. Additionally, the saturation of the actuators was included as a limit to the thruster control signal. Finally, a signal h_i, representing the health status of the i-th actuator is input to the control class. The control class outputs each control input u_i as $u_i = h_i u_i^{des}$ where u_i^{des} represents the *desired* control input, i.e. the ideal control input without the factoring in of any faults. The resulting *ControlAUV* class is reported in Fig. 7.4.

Finally, the *FaultsAUV* class, accounting for the definition of the fault signals, was created. This class outputs four signals (one signal per actuator). Each signal $h_i \in [0,1]$ denotes the efficiency of the i-th actuator, namely $h_i = 1$ when the actuator is functioning nominally, and $h_i = 0$ when the actuator is completely faulty.

In the following sections, the control function is derived and, following that, two examples of closed-loop control scenarios are simulated and discussed.

7.3 Synthesis of the control law

To control the AUV's surge speed, sway speed, angular rate around the z-axis and yaw angle, a pFT-ANLC law was first derived. The dynamical model was selected by extending the AUV dynamics described in Eq. (6.15) to include the sway speed and yaw angle. The vehicle 4-dimensional dynamics, characterised by $\boldsymbol{x} = [u, v, r, \psi]^T$ and $\boldsymbol{u} = [F_1, F_2, F_3, F_4]^T$, are defined as:

$$
\begin{cases}
\dot{x}_1 = \dfrac{-X_u x_1 - X_{uu} x_1^2 + m x_2 x_3 + h_1 F_{1,x} + h_2 F_{2,x} + h_3 F_{3,x} + h_4 F_{4,x}}{m} & \text{(7.1a)} \\
\dot{x}_2 = \dfrac{-Y_v x_2 - Y_{vv} x_2^2 - m x_1 x_3 + h_1 F_{1,y} + h_2 F_{2,y} + h_3 F_{3,y} + h_4 F_{4,y}}{m} & \text{(7.1b)} \\
\dot{x}_3 = \dfrac{-N_r x_3 - N_{rr} x_3^2 + (-F_{1,x} l_{1,y} + F_{1,y} l_{1,x}) h_1 + (-F_{2,x} l_{2,y} + F_{2,y} l_{2,x}) h_2}{J_z} + \\
\qquad \dfrac{(-F_{3,x} l_{3,y} + F_{3,y} l_{3,x}) h_3 + (-F_{4,x} l_{4,y} + F_{4,y} l_{4,x}) h_4}{J_z} & \text{(7.1c)} \\
\dot{x}_4 = x_3, & \text{(7.1d)}
\end{cases}
$$

where $F_{i,x} = F_i \sin(\beta_i)$ and $F_{i,y} = F_i \cos(\beta_i)$ represent the projection of F_i along the x_B and y_B-axes, respectively; X_u, X_{uu} denote the linear and quadratic surge drag coefficients, Y_v, Y_{vv} the linear and quadratic sway drag coefficients, while N_r, N_{rr} the linear and quadratic yaw drag coefficients. Additionally, the angle β_i represents the orientation of thruster i-th with respect to y_B, and l_i the distance from the COG, with $l_{i,x}$ and $l_{i,y}$ indicating the projections along the x_b and y_b-axis, respectively. Finally, m represents the mass of the vehicle (including the added mass) and with J_z the vehicle's moment of inertia around the vertical axis (including the added inertia). For this study, geometric and hydrodynamic parameters are selected as reported in Table 7.1, where such parameters are inspired to a design iteration of the hover-capable AUV developed at the NOC.

To synthesise the control law, training insights gained from the investigations discussed in Chapter 5 and Chapter 6 were employed. First, the hyperparameters of the Lyapunov ANN and of the Control ANN were chosen as reported in Table 7.2.

Having selected the ANN architectures, a control law that can accommodate ac-

Table 7.1: AUV dynamical parameters.

Parameters	Values	Units	Explanation
m	500.0	kg	Vehicle mass including added mass
J_z	300.0	kg m^2	Vehicle inertia (around vertical axis) including added inertia
X_u	6.106	kg s^{-1}	Surge linear drag coefficient
X_{uu}	5.0	kg m^{-1}	Surge quadratic drag coefficient
Y_v	6.106	kg s^{-1}	Sway linear drag coefficient
Y_{vv}	5.0	kg m^{-1}	Sway quadratic drag coefficient
N_r	210.0	kg m^2 s^{-1}	Yaw linear drag coefficient
N_{rr}	3.0	kg m^2	Yaw quadratic drag coefficient
$l_{1,x}$	-1.01	m	Distance of F_1 projected along x_B
$l_{1,y}$	-0.353	m	Distance of F_1 projected along y_B
β_1	135	deg	Orientation of F_1 wrt y_B
$l_{2,x}$	-1.01	m	Distance of F_2 projected along x_B
$l_{2,y}$	0.353	m	Distance of F_2 projected along y_B
β_2	45	deg	Orientation of F_2 wrt y_B
$l_{3,x}$	1.01	m	Distance of F_3 projected along x_B
$l_{3,y}$	-0.353	m	Distance of F_3 projected along y_B
β_3	45	deg	Orientation of F_3 wrt y_B
$l_{4,x}$	1.01	m	Distance of F_4 projected along x_B
$l_{4,y}$	0.353	m	Distance of F_4 projected along y_B
β_4	135	deg	Orientation of F_4 wrt y_B

Table 7.2: AUV campaign – ANN architecture.

Parameter	Lyapunov ANN	Control ANN
Layer size	[2, 10, 1]	[4, 4]
Bias	[No, No]	[Yes]
σ	[x^2, linear]	[linear]

tuator saturation was selected to better replicate a realistic control design process. The actuator saturation was set at a limit of 37.5 [N], a value chosen within the usual range of underwater robotic vehicle applications (once again selected as the maximum force generated by a *BlueRobotics T200* thruster).

In line with previous analyses, the target setpoint was chosen as $\boldsymbol{x}^{\star} = [0.5, 0.0, 0.0, 0.0]^{T}$. This choice of setpoint targets the condition of the AUV moving at prescribed surge speed, while controlling the sway speed, the angular rate and the yaw angle at $0 \pm \varepsilon$.

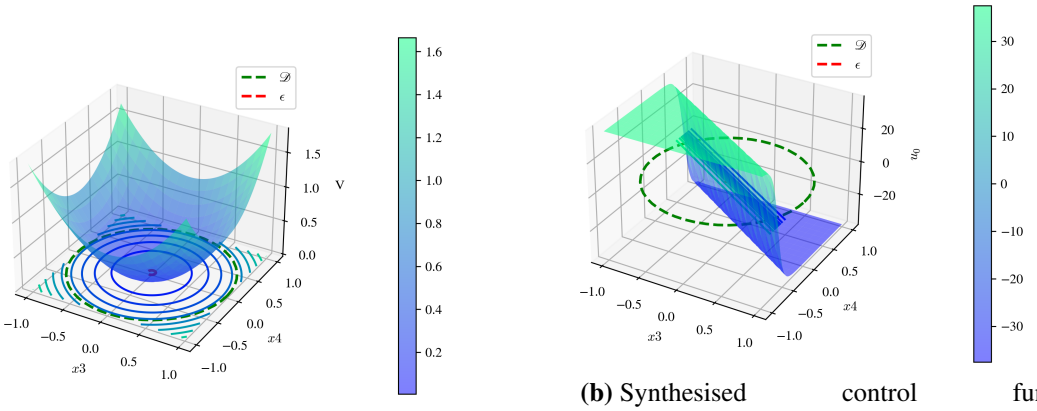
For this study, the CLF domain was selected as $\mathcal{D} = 1.0$ and $\varepsilon = 0.05$. The value of ε was determined iteratively: starting from a *relaxed* value of $\varepsilon = 0.3$, the value was gradually reduced until $\varepsilon = 0.05$, empirically found to be the smallest value that would lead to the training terminating successfully.

By selecting the faults as total loss of actuator efficiency of either F_1, F_2 F_3 or F_4, one nominal fault free mode and four fault modes are defined. With this definition of the faults, one converging training run was completed within 716 learning iterations. The CLF V is obtained as:

$$\begin{aligned} V = (&0.878x_1^2 + 0.861x_2^2 + 3.568x_3^2 + 1.599x_4^2 + \\ &0.067(x_1x_2) + 0.711(x_1x_3) + 0.301(x_1x_4) + 0.0179(x_2x_3) + \\ &0.429(x_2x_4) + 3.549(x_3x_4)). \end{aligned} \tag{7.2}$$

A projection of $V(x_1, x_2, x_3, x_4)$ over the plane defined by $x_1 = x_2 = 0$ is reported in Fig. 7.5a.

The control functions, encompassing the thruster saturation set at 37.5 [N], are synthesised as follows:



(a) Synthesised CLF $V(x1,x2,x3,x4)$.

(b) Synthesised control function $F_1(x1,x2,x3,x4)$ with saturation.

Figure 7.5: Results of the pFT-ANLC synthesis for the AUV case study, plotted for $x_1 = x_2 = 0$.

$$F_1 = 37.5\tanh(0.208 - 4.826x_1 - 7.430x_2 + 4.724x_3 + 3.484x_4) \tag{7.3a}$$

$$F_2 = 37.5\tanh(-0.133 + 5.510x_1 - 4.630x_2 + 9.201x_3 + 4.962x_4) \tag{7.3b}$$

$$F_3 = 37.5\tanh(-0.165 + 4.351x_1 - 3.627x_2 - 7.429x_3 - 6.505x_4) \tag{7.3c}$$

$$F_4 = 37.5\tanh(0.177 - 3.689x_1 - 3.990x_2 - 8.501x_3 - 6.153x_4). \tag{7.3d}$$

After having synthesised the control functions, the expressions of Eq. (7.3) are applied in a closed-loop within the *OpenMAUVe.ControlAUV* class, as:

$$F_1 = 37.5\tanh(0.208 - 4.826e_1 - 7.430e_2 + 4.724e_3 + 3.484e_4) \tag{7.4a}$$

$$F_2 = 37.5\tanh(-0.133 + 5.510e_1 - 4.630e_2 + 9.201e_3 + 4.962e_4) \tag{7.4b}$$

$$F_3 = 37.5\tanh(-0.165 + 4.351e_1 - 3.627e_2 - 7.429e_3 - 6.505e_4) \tag{7.4c}$$

$$F_4 = 37.5\tanh(0.177 - 3.689e_1 - 3.990e_2 - 8.501e_3 - 6.153e_4), \tag{7.4d}$$

where e_i represents the error of the i-th state, namely $e_i = x_i^m - x_i^\star$ with x_i^m the measured value of x_i and with $x_i^\star$ the target state value.

7.4 Numerical Evaluation

After having synthesised the control law and implemented it within the *OpenMAUVe* simulator, closed-loop tests were devised to verify the efficacy of the control strategy. To verify that the control law can cope with the faulty actuators, two simulated scenarios were devised and are presented hereby. First, a constant target setpoint was requested and faults at different thrusters were injected. Second, a simulation encompassing the tracking of a time-varying reference yaw angle was carried out. Finally, the results of these two simulated scenarios will be analysed to determine whether or not the control strategy is verified.

7.4.1 Simulation A: simulation with four faults injected

The first simulated scenario was with the AUV initially set at rest, namely $\boldsymbol{x}_0 = [0.0, 0.0, 0.0, 0.0]^T$, controlled to move to a target setpoint $\boldsymbol{x}^\star = [0.5, 0.0, 0.0, 0.0]^T$. This setpoint represents a condition where the AUV is commanded to move with a constant surge speed of 0.5 [m/s], with the other states controlled at 0.

During the simulation, faults were injected on either F_1, F_2, F_3 or F_4 (only a maximum of one fault present at any time). The efficiency profile of the actuators are reported in Fig. 7.6, with each fault injected for a time period of 200 [s], during which time the other three actuators are functioning nominally.

A simulation lasting for 1700 [s] was run. The thruster forces were analysed to verify that the control commands did not result in the actuator saturation limits being exceeded. As an example, F_1 is illustrated in Fig. 7.7, with a magnified view of the first 20 [s] of the simulation reported in Fig. 7.8 (F_2, F_3 and F_4 follow similar trends). The plots highlight how the requested control force remains bounded within the saturation limits set at 37.5 [N]. Additionally, it is worth noting that F_1 outputs null force in the time range from 100 [s] to 300 [s], when the fault at F_1 is injected.

Finally, the AUV's dynamics were inspected to verify the efficacy of the control law. First, the surge dynamics (u) are reported in Fig. 7.9, with the ε-stability bound highlighted in green and with the target setpoint $u^\star = 0.5$ [m/s] reported with dashed lines. A magnified view of the surge dynamics over the first 20 [s] of the simulation is reported in Fig. 7.10. Fig. 7.9 and Fig. 7.10 illustrate how the surge speed enters the

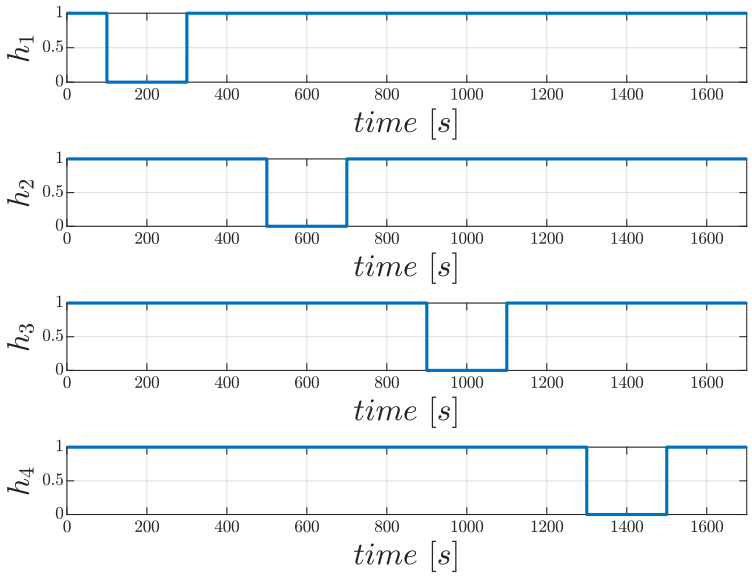


Figure 7.6: *OpenMAUVe* AUV test case: actuator efficiency profiles.

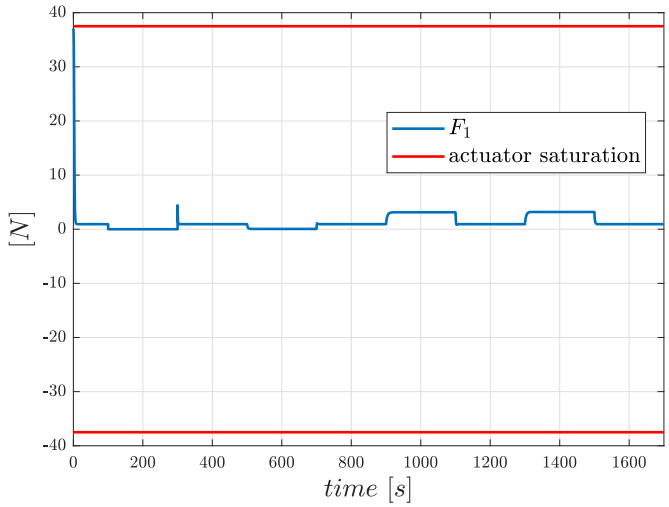


Figure 7.7: *OpenMAUVe* AUV test case A: control force generated by the first thruster (F_1).

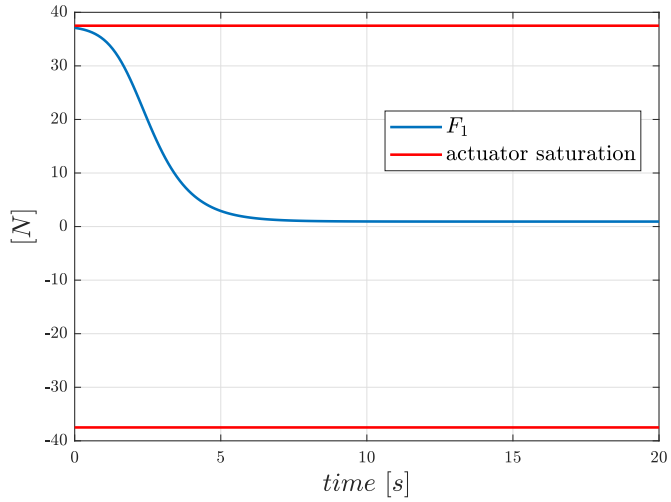


Figure 7.8: *OpenMAUVe* AUV test case A: magnified view of Fig. 7.7 in the first 20 [s] of the simulation.

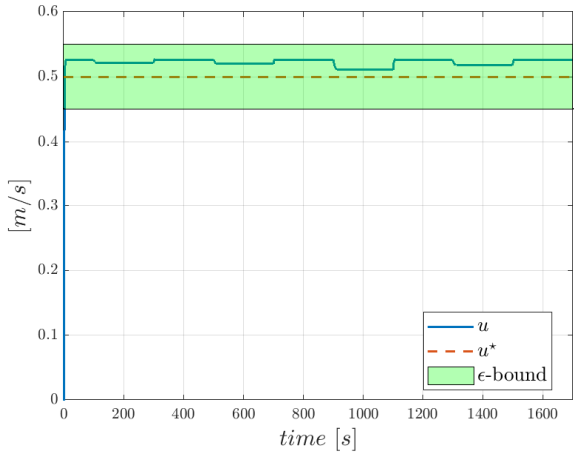


Figure 7.9: *OpenMAUVe* AUV test case A: surge dynamics (u).

ε-stability bound after 3.11 [s], and remains within the region defined by $u^\star \pm \varepsilon$ for the remainder of the simulated scenario, even when faults occur.

Similar to the surge speed dynamics, the sway speed dynamics (v), the angular rate dynamics (r) and the yaw angle dynamics (ψ) are illustrated in in Fig. 7.11a, Fig. 7.11b, and Fig. 7.11c, respectively. As expected, all the dynamic responses remain within the ε-stability bound for the duration of the simulation, both during the nominal (faultless) operation and during operation under fault mode conditions.

After having confirmed that the synthesised control law can control the AUV around $\boldsymbol{x}^\star$ even when faults occur, a more advanced simulation scenario is set up.

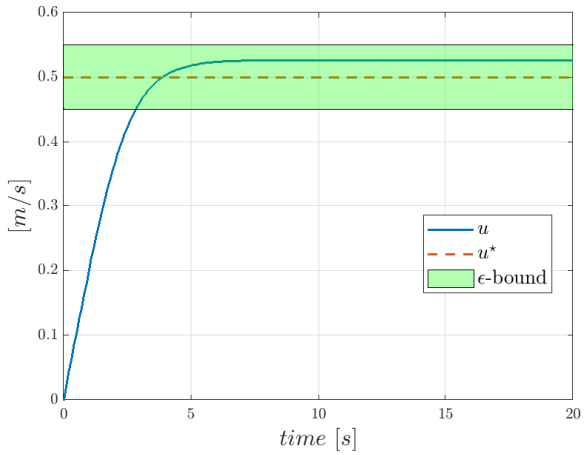
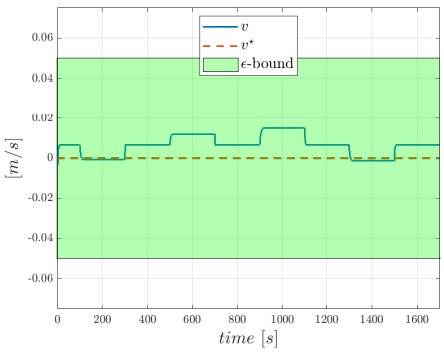
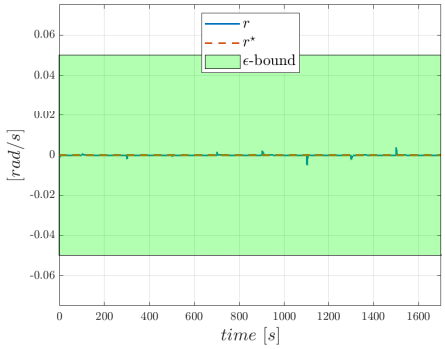


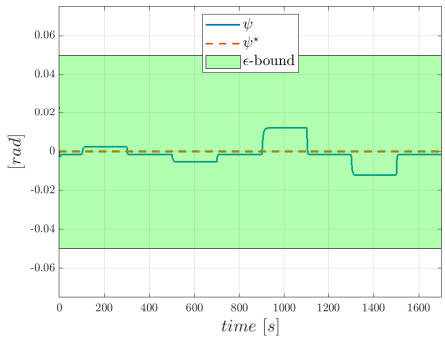
Figure 7.10: *OpenMAUVe* AUV test case A: magnified view of Fig. 7.9 in the first 20 [s] of the simulation.



(a) Sway speed dynamics (v).



(b) Angular rate around the z-axis dynamics (r).



(c) Yaw angle dynamics (ψ).

Figure 7.11: *OpenMAUVe* AUV test case A.

7.4.2 Simulation B: tracking a time-varying yaw angle reference

In this second scenario, the AUV was required to track a time-varying yaw angle reference. This simulation study recreates a scenario where the AUV is requested to

perform a manoeuvre encompassing time-varying heading, such as in lawnmower applications for sonar search [25, 243]. In this scenario, the AUV moves at constant depth, with the surge speed set to the desired cruise speed, and where the reference yaw angle is used to control the desired heading.

For this study, the overall simulation time is set to 200 [s], with the target cruise speed selected as $u^\star = 0.5$ [m/s], with $v^\star = 0.0$ [m/s] and $r^\star = 0.0$ [rad/s]. The control system is tasked with tracking a time-varying yaw angle reference $\psi^\star(t)$.

It is of paramount importance to recall that the pFT-ANLC is designed to ensure tracking of a unique setpoint value, and not tracking of a generic time-varying reference value $\boldsymbol{x}^\star$. Nonetheless, certain reference tracking capabilities are retained in specific dynamical models, such is the case for the 4-dimensional AUV model Eq. (7.1), selected for the present study.

In model Eq. (7.1), the fourth state (namely the yaw angle ψ), does not appear in the dynamics of the other states. For this reason, the shifted model $\dot{\tilde{\boldsymbol{x}}} = \boldsymbol{f}(\tilde{\boldsymbol{x}}+\boldsymbol{x}^\star, \tilde{\boldsymbol{u}}+\boldsymbol{u}^\star)$ (discussed in Section 6.5.2.1), used to synthesise the control law, is independent from the value of $\psi^\star$ (i.e. $x_4^\star$). Model Eq. (7.1), shifted such that $\tilde{\boldsymbol{x}}$ coincides with the origin, shows the following coupling:

$$\begin{cases} \dot{\tilde{x}}_1 = f(x_1, x_1^\star, x_2, x_2^\star, x_3, x_3^\star + \boldsymbol{u}) & \text{(7.5a)} \\ \dot{\tilde{x}}_2 = f(x_1, x_1^\star, x_2, x_2^\star, x_3, x_3^\star + \boldsymbol{u}) & \text{(7.5b)} \\ \dot{\tilde{x}}_3 = f(x_1, x_1^\star, x_2, x_2^\star, x_3, x_3^\star + \boldsymbol{u}) & \text{(7.5c)} \\ \dot{\tilde{x}}_4 = f(x_3, x_3^\star). & \text{(7.5d)} \end{cases}$$

From a physical perspective, this means that stabilising model Eq. (7.5) around $x_4^\star = 0$, or around any other value $x_4^\star(t) \in \mathbb{R}$, does not entail any change in the control function design. This clearly represents a beneficial feature from an analytical perspective, that well couples with the yaw angle being a perfect candidate for a conventional time-varying heading tracking application. For the sake of this numerical example, the target $x_4^\star(t)$ will be slowly varying; the associated limitations are discussed in more detail in Section 7.5.

Once again, to evaluate the capability to track $\psi^\star(t)$ both during nominal and fault

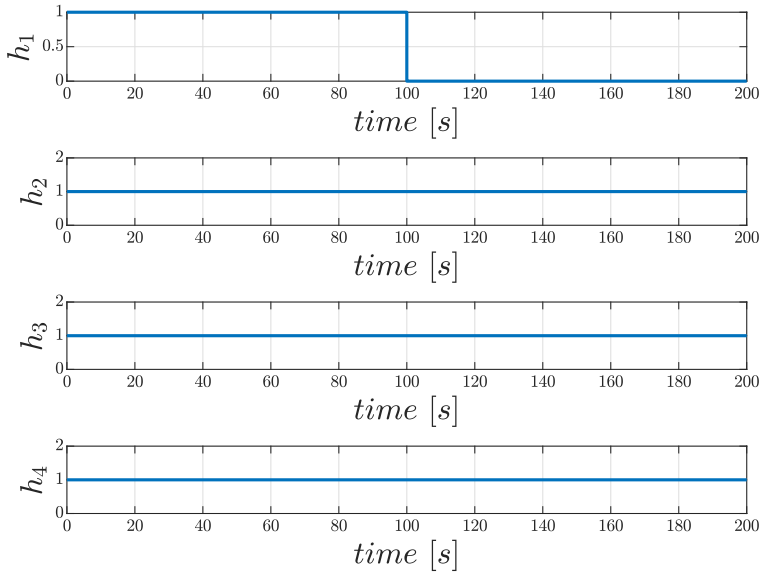


Figure 7.12: *OpenMAUVe* AUV test case B: actuator efficiency profiles.

modes, faults at different thrusters are injected. As an example, the case of F_1 being faulty is showcased here (with the other fault modes following comparable trends). The efficiency profiles of the actuators are reported in Fig. 7.12, with the fault at F_1 injected halfway through the simulation at 100 [s], while the other three actuators continue to function nominally.

To demonstrate the system's ability to track a varying yaw angle target profile $\psi^{\star}(t)$, the desired angle is initially set to 0 [deg], then set to +10 [deg] at 50 [s], and finally set to -10 [deg] after 150 [s]. The yaw dynamic response is illustrated in Fig. 7.13. As it can be seen, the yaw dynamic response tracks the reference value within the ε-stability bound both during nominal operations (all four thrusters functioning nominally) and during faulty operations (after 100 [s]).

In line with the previous case study, surge speed (u), sway speed (v) and angular rate (r) remain within the ε-stability bound during the simulation, with the surge speed reported as an example in Fig. 7.14.

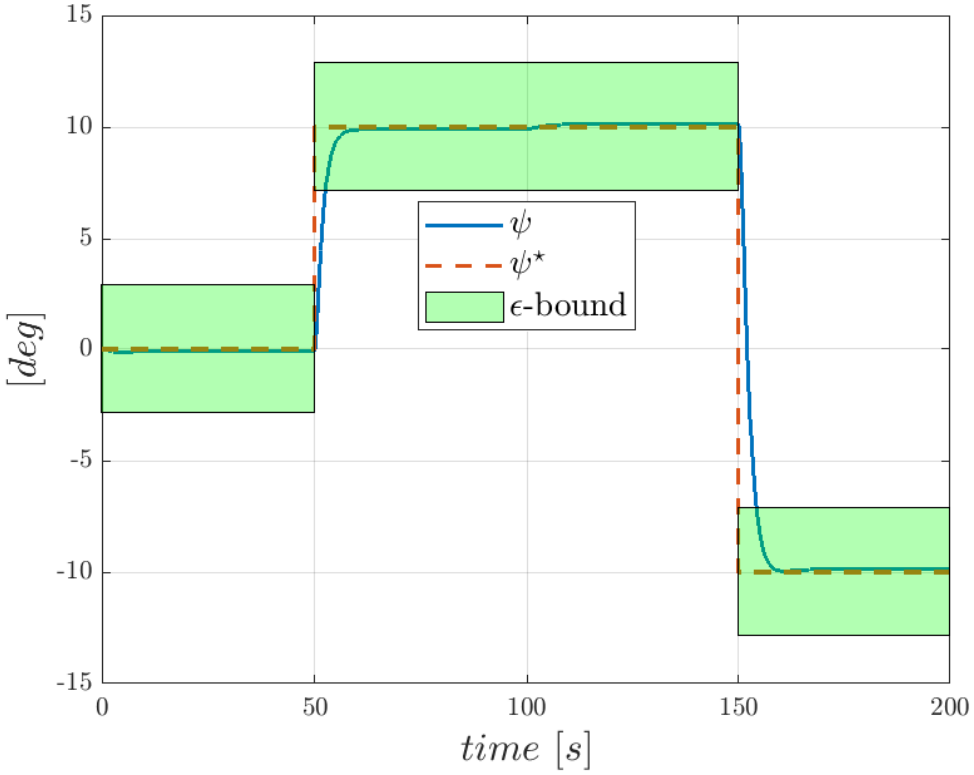


Figure 7.13: *OpenMAUVe* AUV test case B: yaw angle dynamics with time-varying reference and fault injected at *time*=100 [s].

7.5 Discussion on limitations

During the development of the numerical experiments reported in Section 7.4, two limitations were highlighted. Both theoretical and practical limitations are discussed in more detail hereby.

1. With respect to the case studies reported earlier in Chapter 5 and Chapter 6, this investigation focuses on the case of a 4-dimensional dynamical system. In accordance with findings discussed in the earlier chapters, the difficulty of synthesising control laws and CLFs grows as the dimension of the system increases or as the complexity of the nonlinearities rises.

   More specifically, a key limiting element observed is the tuning of the hyperparameters. Deeper and wider ANN architectures benefit the Learner, while, at the same time, render the CLFs's formal verification stage more convoluted. The synthesis of the control law illustrated in Section 7.3 required over 20 tuning

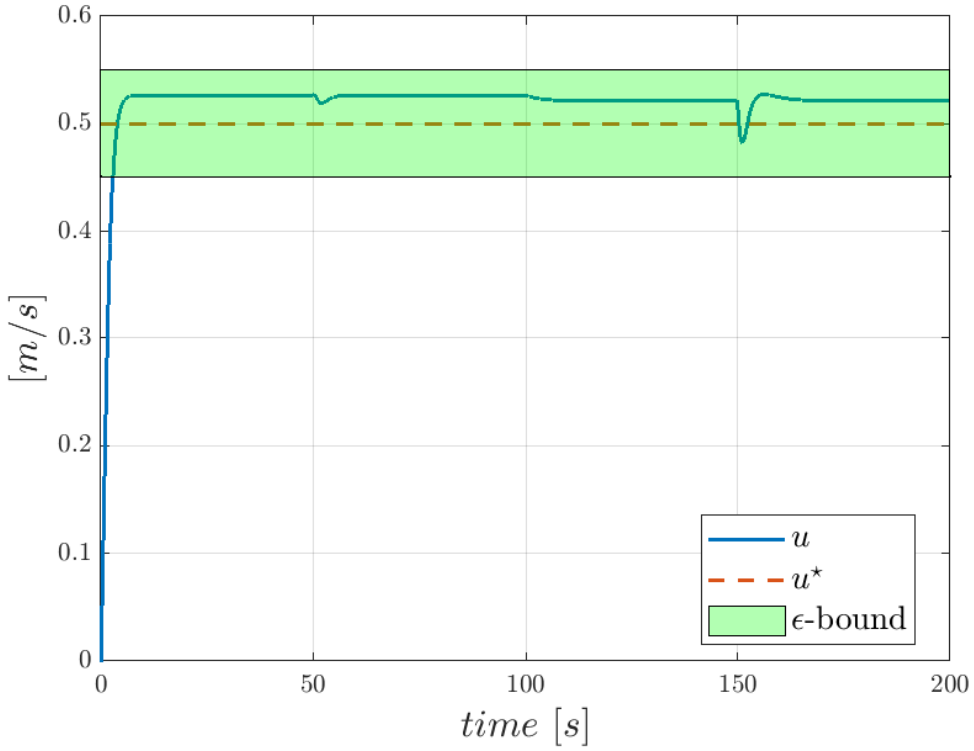


Figure 7.14: *OpenMAUVe* AUV test case B: yaw angle tracking response.

attempts, with the associated work ranging over a working week. These findings confirm that the proposed control synthesis approach, in the present form, is not suited for high-dimensional systems.

2. The pFT-ANLC method is designed to solve a problem of setpoint-tracking. From a broader operational perspective, this represents the major limitation. Nonetheless, as shown in the case study reported in Section 7.4.2, specific properties of the dynamical systems can be leveraged to extend the framework to time-varying reference tracking applications.

7.6 Conclusions

In this chapter, a complete example of how to exploit the techniques and tools developed in this thesis to complete the modelling-simulation-control stages for a faulty AUV was detailed. First, a dynamical model of an AUV was defined within the *OpenMAUVe* simulator framework. Next, four possible faults were defined, and a dedicated

control law was automatically synthesised with the pFT-ANLC method. The control law was then implemented in a closed-loop architecture within the *OpenMAUVe* simulator and tested in nominal and faulty operational modes, demonstrating the ability to control the vehicle dynamics within the prescribed stability bounds. In the second simulation scenario, a time-varying reference yaw angle was requested, resembling a conventional lawnmower operational mode, such as during sonar searches.

The results presented highlight how the methods and tools developed and presented within this thesis are an effective suite to devise control laws for complex nonlinear dynamics such as those characterising AUVs applications. In line with results illustrated in the previous chapters, limitations are noticed. An example on how to leverage the physical dynamical properties is illustrated as the base for future studies.

In the current industrial landscape characterised by the development and use of ever more sophisticated AUVs, automating the synthesis of control laws along while streamlining the test and verification of closed-loop dynamics, represents a necessary step in the future of AUVs operational control. The pFT-ANLC methods represents the first step towards increasing the reliability of AUVs to actuator faults, and to ensure the capability to successfully and safely complete the allocated tasks.

Chapter 8

Concluding remarks

8.1 Overview statements

In this thesis, novel ML-based technologies were exploited to tackle unresolved control problems without compromising on the classical formal theories characterising the control engineering discipline. Specifically, ML-based technologies were employed to tackle non-trivial control problems, such as the computation of pFTC solutions for AUVs, eliminating the need to use sensors to confirm the existence of faults.

Three research questions were initially formulated, and are recapped here together with the related answers that were derived as result of this investigation.

Research Question RQ1

Are multibody object-oriented simulators suitable for simulating the dynamics of autonomous underwater vehicles?

Answer Yes, multibody object-oriented simulators are a particularly suitable tools to investigate the dynamics of AUVs. The associated study, reported in Chapter 4 and which resulted in the design of the OpenMAUVe simulator, illustrated how exploiting the classes-object paradigm to define the rigid-bodies composing an AUV significantly simplifies the construction of complex multibody dynamics. Employing object-oriented simulation tools well fits the increasing need to efficiently test and simulate an ever growing number of different vehicles. To this aim, it was shown how ROVs, AUVs, UGs and hybrid vehicles obey the same underlying hydrodynamic characteristics, that can be captured by using standard blocks to model the hydrodynamic inter-

actions with the surrounding environment. When non-standard dynamics arise, such as where shifting and rolling internal masses are employed, the multibody modelling approach offers a clear advantage over the traditional causal simulators, that require the equations of motions to be derived ad hoc. As movable masses are increasingly being used as control inputs in combination with the more traditional actuators such as thrusters and control surfaces, multibody simulators allow prompt re-design of the vehicle and testing the different actuator configurations and arrangements. In the FTC context, where actuator redundancy plays a crucial role, the ability to promptly test a vehicle with different actuators can provide key insights regarding safety and reliability of target design choices. To this end, the OpenMAUVe environment was used in Chapter 7 to verify the use of control laws, highlighting how such simulation tool can be used to verify the behaviour and performance of a control law in a safe manner. Moreover, the OpenMAUVe simulator allows for efficient simulation of long-lasting missions, owing to the enhanced capabilities to solve systems of DAEs when compared to standard causal simulators. This can allow testing the effect of slow dynamical phenomena, such as the growth of biofouling agents on the hull. Finally, given the capability to assemble vehicles through the combination of mechanical components, the simulator can be used as a fast prototyping tool for novel vehicle design, allowing the understanding of how the dynamics change when subjected to different control inputs.

Research Question RQ2 (initial)

Is the Neural Lyapunov Control method an efficient approach for the design of stable control laws for nonlinear systems?

Answer Experiments have shown that the Neural Lyapunov Control, in its original formulation, can not be used to design stable control laws for nonlinear systems without proving sensible initialisation parameters. This conclusion was reached after running over 1800 training scenarios where the hyperparameters of the methods were systematically permuted. It must be noted that the Neural Lyapunov Control was not originally proposed as a method to synthesise control laws, but rather as a method to certify the closed-loop stability and extend the Region of Attraction of an equilibrium of a nonlinear system by starting from a candidate stabilising control law. The systematic analysis

highlighted areas of improvements that were deemed worth investigating to render the method fit for the purpose of systematically synthesising control laws. Based on these findings, Research Question RQ2 was reformulated as follows.

Research Question RQ2 (reformulated)

Can an upgraded version of the Neural Lyapunov Control method be devised to systematically design stable control laws for continuous-time nonlinear dynamical systems?

Answer Yes, the Neural Lyapunov Control can be modified to design control laws for continuous-time nonlinear dynamical systems. The modified version of the method proposed in this thesis is denoted as Augmented Neural Lyapunov Control. Such a method was shown capable to automatically synthesise both linear and nonlinear control laws for benchmark systems, without requiring the provision of an initial sensible control candidate solution to successfully converge.

Research Question RQ3

How can the Neural Lyapunov Control method be extended to a passive fault-tolerant control approach to ensure closed-loop stability for platforms affected by actuator faults?

Answer The (Augmented) Neural Lyapunov Control method can be extended to fit such a purpose in the form of the passive Fault-Tolerant Augmented Neural Lyapunov Control. The method relies on the idea that, given a nominal dynamical system and a finite number of fault modes, each one associated with the fault of one actuator, the Lie derivatives of each mode must be negative definite at the same time. The method allows the derivation of a unique control law that stabilises both the nominal and fault modes. Such a method is robust to a set of predefined faults, and does not require confirmatory information as to the status of the actuators that would normally be provided by sensors and detection algorithms.

The tools and methods developed in this thesis can serve as a foundation for further academic research and for implementation by industrial practitioners. Three open-source software tools in the field of simulation and control of AUVs were developed

as part of this research. These contributions are summarised as follows.

- Upon validation, the OpenMAUVe simulator can be used to test, tune and verify different control laws and navigation algorithms in AUV applications. Additionally, due to the possibility to effectively simulate long-lasting missions, the tool can be used to investigate the effect of gradually changing dynamical phenomena, such as biofouling growth on the vehicle hull.

- Resulting from the ability to promptly compose vehicles as assemblies of mechanical subcomponents, OpenMAUVe can assist the design of novel AUVs concepts, allowing the investigation of the vehicle dynamics and the response to prescribed control inputs.

- The ANLC software tool can assist every control application that can benefit from the definition of an associated CLF. Despite the fact that synthesis of state-feedback control functions is achieved conventionally with standard analytical techniques, a control law synthesised through the ANLC adds additional information regarding the ROA, in turn providing insights regarding the nonlinear dynamics, rather than just on the usual linearised behaviour in the neighbourhood of an equilibrium. Additionally, the ANLC software tool allows the synthesis of nonlinear state-feedback control laws, allowing enhanced flexibility when compared to the conventional linear counterpart.

- The pFT-ANLC software tool represents a first in the synthesis of pFTC accounting for the complete loss of actuator efficiency, without requiring linearising (i.e. simplifying) the system dynamics. Additionally, this tool allows to factor in actuator saturation. These features facilitate the enhancing of the onboard control software reliability, requiring fewer assumptions and accounting for more advanced real nonlinear phenomena.

Overall, these tools aim at supporting academic and industrial applications in the maritime field, increasing the reliability of the vehicles and their ability to respond to unforeseeable scenarios. In turn, these research outputs increase the confidence

in autonomous vehicles' ability to successfully complete the assigned missions, by autonomously responding to disruptive events.

Nevertheless, the applicability of the work illustrated in this thesis is subject to certain limitations. First, the OpenMAUVe simulator was verified against theoretical models and compared against the results of other simulators. Complete validation of the simulator is not possible at this stage, due to a lack of publicly available datasets encompassing the time series commands to the actuator, the measured and estimated position, velocities and attitude of the vehicle, and the details regarding the vehicle trimming configuration. Even more than a conventional caveat, the results of the OpenMAUVe simulator need to be critically analysed and not used as a source of ground truth at this stage. Additionally, current limitations encompass environmental disturbances, depth-related hull deformation phenomena and non-idealised behaviours linked to the presence of hull appendages, that can not yet be accounted for.

Second, the ANLC and pFT-ANLC exhibit common limitations, owing to their underlying shared methodological basis. The major limitation is linked to the lack of scalability to high dimensional systems. In this work, it was possible to systematically obtain control laws for dynamical systems of up to four states. When analysing higher-dimensional dynamics of six states and above, such as those characterising a rigid-body moving unconstrained in space, the difficulty in formally verifying the solution increases. Issues of scalability are intrinsically linked to the nature of the SMT solvers that, with current available technologies, do not have a trivial solution. To mitigate such a limitation, careful tuning of the hyperparameters needs to be carried out, especially when dealing with dynamical models encompassing transcendental functions.

Third, the ANLC and pFT-ANLC methods are currently designed to solve the problem of setpoint stabilisation. Within the specific application of autonomous navigation, interest instead lies in solving problems of reference-tracking. In the current formulation, the proposed methods cannot generically solve a problem of reference-tracking. However, in Chapter 7, it was shown how favourable properties of certain dynamical systems can be leveraged to solve the problem of tracking a time-varying reference.

8.2 Recommendations for future research

This section suggests a set of research directions that can further progress the work presented in this thesis. Naturally, certain research directions would follow from addressing the aforementioned limitations.

- A validation of the OpenMAUVe simulator is necessary to increase the trustworthiness of the output results. Given the lack of open-access data from repositories and other publications, collaborations with a research centre that is remotely operating and deploying AUVs (even more possibly UGs) is necessary to retrieve the necessary hydrodynamic parameters of a vehicle, together with the trimming conditions and the associated deployment data.

- To increase the fidelity of the OpenMAUVe simulator, models of environmental disturbances need to be included, as well as more advanced formulations accommodating for hull deformation phenomena due to pressure and temperature and, where fitted, the effects of hull appendages.

- In order to scale the ANLC and pFT-ANLC methods to higher dimensional systems, alternatives to SMT solvers need to be considered as stability certificates. As of the summer of 2024, there is no alternative technology to provide the same certificate over the dense domain of the Reals. Nonetheless, a possible relaxation entails using probabilistic verification methods, rather than exact methods, in order to reduce the associated complexity burden and render the problem tractable within limited computational means.

- Further tests of the application of pFT-ANLC to AUV studies are suggested. In more detail, a proposed set of tests will encompass wrapping the pFTC control law within a complete Guidance, Navigation and Control (GNC) architecture. For instance, testing a classical cascade control architecture where the AUV desired heading is provided by a top level planner (or Guidance-law), is deemed the most straightforward follow-up stage.

- Another suggested research direction encompasses veering away from the clas-

sical cascade GNC architectures where possible faults are typically managed at the innermost control loop, e.g. the force and moment control loop, and addressing the faults at a higher level, namely at the Guidance level. To this regard, the concept of fault-tolerant guidance rather than fault-tolerant control is highlighted as a valuable research direction.

Bibliography

[1] A Evangelio, O Nyaas, G Yuzichuck, S Sweeney, M Karagoz, M Coffman, and J Fox. Guidance for developing maritime unmanned systems (MUS) capability. *Combined Joint Operations from the Sea Centre of Excellence Report*, 12:029–02, 2012.

[2] European Defence Agency. Best practice guide for unmanned maritime systems handling, operations, design and regulations. https://eda.europa.eu/docs/default-source/documents/eda_ums-bpg-edition-2022_public.pdf, (Accessed: 16/09/2024).

[3] International Maritime Organisation. Msc.1/Circ.1638 - outcome of the regulatory scoping exercise for the use of Maritime Autonomous Surface Ships (MASS). https://wwwcdn.imo.org/localresources/en/MediaCentre/PressBriefings/Documents/MSC.1-Circ.1638%20-%20Outcome%20Of%20The%20Regulatory%20Scoping%20ExerciseFor%20The%20Use%20Of%20Maritime%20Autonomous%20Surface%20Ships...%20(Secretariat).pdf, (Accessed: 16/09/2024).

[4] Maritime & Coastguard Agency. The Workboat Code, edition 3, 2024. https://assets.publishing.service.gov.uk/media/667c2220aec8650b10090087/Workboat_Code_Edition_3.pdf, (Accessed: 16/09/2024).

[5] Tobias Hofmann and Alexander Proelss. The operation of gliders under the international law of the sea. *Ocean Development & International Law*, 46(3):167–187, 2015.

[6] Committees UK Parliament. National Oceanography Centre (NOC) – written evidence (AUV0056). https://committees.parliament.uk/writtenevidence/74298/html/, (Accessed: 16/09/2024).

[7] Anja Diez. Liquid water on Mars. *Science*, 361(6401):448–449, 2018.

[8] Margaret G Kivelson, Krishan K Khurana, Christopher T Russell, Martin Volwerk, Raymond J Walker, and Christophe Zimmer. Galileo magnetometer measurements: A stronger case for a subsurface ocean at Europa. *Science*, 289(5483):1340–1343, 2000.

[9] Olivier Grasset, MK Dougherty, A Coustenis, EJ Bunce, C Erd, D Titov, M Blanc, A Coates, P Drossart, LN Fletcher, et al. JUpiter ICy moons Explorer (JUICE): An ESA mission to orbit Ganymede and to characterise the Jupiter system. *Planetary and Space Science*, 78:1–21, 2013.

[10] Wayne Zimmerman, Robert Bonitz, and Jason Feldman. Cryobot: an ice penetrating robotic vehicle for Mars and Europa. In *2001 IEEE Aerospace Conference Proceedings (Cat. No. 01TH8542)*, volume 1, pages 1–311. IEEE, 2001.

[11] Rodrigo Perez Fernandez and Antonio Sanchez-Torres. Underwater Exploration Mission on Europa Jovian Moon. *International Journal of Engineering Research & Science*, 1, 2015.

[12] Peter W Kimball, Evan B Clark, Mark Scully, Kristof Richmond, Chris Flesher, Laura E Lindzey, John Harman, Keith Huffstutler, Justin Lawrence, Scott Lelievre, et al. The ARTEMIS under-ice AUV docking system. *Journal of field robotics*, 35(2):299–308, 2018.

[13] Dirk Heinen, Jan Audehm, Fabian Becker, Georg Boeck, Clemens Espe, Marco Feldmann, Gero Francke, Pia Friend, Niklas Haberberger, Klaus Helbin, et al. The TRIPLE melting probe-an electro-thermal drill with a forefield reconnaissance system to access subglacial lakes and oceans. In *OCEANS 2021: San Diego–Porto*, pages 1–7. IEEE, 2021.

[14] University of California San Diego. Argo float map. https://www.ocean-ops.org/board, (Accessed: 16/09/2024).

[15] Web Of Science. List of publications containing the kewords "control systems", "machine-learning" within the engineering field. `https://www.ocean-ops.org/board`, (Accessed: 16/09/2024).

[16] Souther Ocean Carbon and Climate Observatory. Ocean gliders battle southern ocean in order to capture the seasonal cycle. `https://socco.org.za/news/ocean-gliders-battle-southern-ocean-in-order-to-capture-the-seasonal-cycle/attachment/figure-2/`, (Accessed: 16/09/2024).

[17] Michel Verhaegen, Stoyan Kanev, Redouane Hallouzi, Colin Jones, Jan Maciejowski, and Hafid Smail. *Fault Tolerant Flight Control - A Survey*, pages 47–89. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[18] Rosana Cibely Batista Rego. *Lyapunov-based intelligent control*. PhD thesis, Universidade Federal do Rio Grande do Norte, 2022.

[19] Davide Grande, Luofeng Huang, Catherine A Harris, Peng Wu, Giles Thomas, and Enrico Anderlini. Open-source simulation of underwater gliders. In *OCEANS 2021: San Diego–Porto*, pages 1–8. IEEE, 2021.

[20] Joshua Grady Graver. *Underwater gliders: Dynamics, control and design*. PhD thesis, Princeton University, 2005.

[21] Gursel Serpen. Empirical approximation for Lyapunov functions with artificial neural nets. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 735–740. IEEE, 2005.

[22] Davide Grande, Enrico Anderlini, Andrea Peruffo, and Georgios Salavasidis. Augmented Neural Lyapunov Control. *IEEE Access*, 2023.

[23] Davide Grande, Andrea Peruffo, Georgios Salavasidis, Enrico Anderlini, Davide Fenucci, Alexander B Phillips, Elias B Kosmatopoulos, and Giles Thomas. Passive Fault-Tolerant Augmented Neural Lyapunov Control: A method to synthesise control functions for marine vehicles affected by actuators faults. *Control Engineering Practice*, 148:105935, 2024.

[24] Davide Grande, Davide Fenucci, Andrea Peruffo, Enrico Anderlini, Alexander B Phillips, Giles Thomas, and Georgios Salavasidis. Systematic Synthesis of Passive Fault-Tolerant Augmented Neural Lyapunov Control Laws for Nonlinear Systems. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 5851–5856. IEEE, 2023.

[25] Davide Fenucci, Francesco Fanelli, Alberto Consensi, Georgios Salavasidis, Miles Pebody, and Alexander B Phillips. A multi-platform Guidance, Navigation and Control system for the autosub family of Autonomous Underwater Vehicles. *Control Engineering Practice*, 146:105902, 2024.

[26] Zachary Bedja-Johnson, Peng Wu, Davide Grande, and Enrico Anderlini. Smart anomaly detection for slocum underwater gliders with a variational autoencoder with long short-term memory networks. *Applied Ocean Research*, 120:103030, 2022.

[27] Stefano Farì and Davide Grande. Vector Field-based Guidance Development for Launch Vehicle Re-entry via Actuated Parafoil. In *Proceedings of the International Astronautical Congress, IAC*, 2021.

[28] Asher Bender, Daniel Matthew Steinberg, Ariell Lee Friedman, and Stefan B Williams. Analysis of an autonomous underwater glider. In *Proceedings of the Australasian conference on robotics and automation*, pages 1–10, 2008.

[29] NOAA Office of Ocean Exploration and Research. Why do we explore the ocean? `https://oceanexplorer.noaa.gov/backmatter/whatisexploration.html#:~:text=Information%20from%20ocean%20exploration%20can,%2C%20tsunamis%2C%20and%20other%20hazards`, (Accessed: 16/09/2024).

[30] NASA Jet Propulsion Laboratory. Why study the ocean? `https://sealevel.jpl.nasa.gov/ocean-observation/why-study-the-ocean/overview/`, (Accessed: 16/09/2024).

[31] Henry Stommel. The slocum mission. *Oceanography*, 2(1):22–25, 1989.

[32] Dean Roemmich, Gregory C Johnson, Stephen Riser, Russ Davis, John Gilson, W Brechner Owens, Silvia L Garzoli, Claudia Schmid, and Mark Ignaszewski. The

Argo Program: Observing the global ocean with profiling floats. *Oceanography*, 22(2):34–43, 2009.

[33] European Global Ocean Observing System. Tide gauges. `https://eurogoos.eu/tide-gauge-task-team/`, (Accessed: 16/09/2024).

[34] NOAA National Ocean Service. What drifters are. `https://www.aoml.noaa.gov/phod/gdp/faq.php#drifter1`, (Accessed: 16/09/2024).

[35] Bastien Y Queste, Karen J Heywood, Jan Kaiser, Gareth A Lee, Adrian Matthews, Sunke Schmidtko, Christopher Walker-Brown, and Stephen W Woodward. Deployments in extreme conditions: Pushing the boundaries of Seaglider capabilities. In *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pages 1–7. IEEE, 2012.

[36] Anton Zhilenkov. The study of the process of the development of marine robotics. *Vibroengineering Procedia*, 8:17–21, 2016.

[37] Alexander Brian Phillips. *Simulations of a self propelled autonomous underwater vehicle*. PhD thesis, University of Southampton, 2010.

[38] Daniel T Roper, Alexander B Phillips, Catherine A Harris, Georgios Salavasidis, Miles Pebody, Robert Templeton, Sriram Vikraman Sithalashmi Amma, Micheal Smart, and Stephen McPhail. Autosub long range 1500: An ultra-endurance AUV with 6000 km range. In *OCEANS 2017-Aberdeen*, pages 1–5. IEEE, 2017.

[39] M Caccia, R Bono, Ga Bruzzone, Gi Bruzzone, E Spirandelli, and G Veruggio. Experiences on actuator fault detection, diagnosis and accomodation for ROVs. *International Symposiyum of Unmanned Untethered Sub-mersible Technol*, pages 3–21, 2001.

[40] Enrico Anderlini, Catherine A Harris, Georgios Salavasidis, Alvaro Lorenzo, Alexander B Phillips, and Giles Thomas. Autonomous detection of the loss of a wing for underwater gliders. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)(50043)*, pages 1–6. IEEE, 2020.

[41] Sarah E Webster, Lee E Freitag, Craig M Lee, and Jason I Gobat. Towards real-time under-ice acoustic navigation at mesoscale ranges. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 537–544. IEEE, 2015.

[42] KW Nicholls, EP Abrahamsen, JJH Buck, PA Dodd, C Goldblatt, G Griffiths, KJ Heywood, NE Hughes, A Kaletzky, GF Lane-Serff, et al. Measurements beneath an Antarctic ice shelf using an autonomous underwater vehicle. *Geophysical Research Letters*, 33(8), 2006.

[43] Gianluca Antonelli. A survey of fault detection/tolerance strategies for AUVs and ROVs. In *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*, pages 109–127. Springer, 2003.

[44] Mouhacine Benosman and K-Y Lum. Passive actuators' fault-tolerant control for affine nonlinear systems. *IEEE Transactions on Control Systems Technology*, 18(1):152–163, 2009.

[45] Nick Bostrom. *Superintelligence: Paths, dangers, strategies*. Oxford University Press., 2016.

[46] Brett A Halperin and Stephanie M Lukin. Artificial dreams: Surreal visual storytelling as inquiry into AI 'hallucination'. In *Proceedings of the 2024 ACM Designing Interactive Systems Conference*, pages 619–637, 2024.

[47] Gary Marcus. Alphaproof, Alphageometry, ChatGPT, and why the future of AI is neurosymbolic. `https://garymarcus.substack.com/p/alphaproof-alphageometry-chatgpt`, (Accessed: 16/09/2024).

[48] Dean C Wardell. *Deep Learning-Based, Passive Fault Tolerant Control Facilitated by a Taxonomy of Cyber-Attack Effects*. PhD thesis, AIR FORCE INSTITUTE OF TECHNOLOGY, 2020.

[49] Deric P Jones. *Biomedical sensors*. Momentum press, 2010.

[50] Alan S Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611, 1976.

[51] Guillaume JJ Ducard. *Fault-tolerant flight control and guidance systems: Practical methods for small unmanned aerial vehicles*. Springer Science & Business Media, 2009.

[52] Mogens Blanke, Michel Kinnaert, Jan Lunze, Marcel Staroswiecki, and Jochen Schröder. *Diagnosis and fault-tolerant control*, volume 2. Springer, 2006.

[53] Ali Zolghadri, David Henry, Jérôme Cieslak, Denis Efimov, and Philippe Goupil. *Fault Diagnosis and Fault-tolerant Control and Guidance for Aerospace Vehicles*, volume 236. Springer, 2014.

[54] Mogens Blanke, Michel Kinnaert, Jan Lunze, and Marcel Staroswiecki. *Diagnosis and fault-tolerant control*, volume 3. Springer, 2015.

[55] Mario Brito, David Smeed, and Gwyn Griffiths. Underwater glider reliability and implications for survey design. *Journal of Atmospheric and Oceanic technology*, 31(12):2858–2870, 2014.

[56] Koorosh Aslansefat, Gholamreza Latif-Shabgahi, and Mojtaba Kamarlouei. A strategy for reliability evaluation and fault diagnosis of Autonomous Underwater Gliding Robot based on its Fault Tree. *International Journal of Advances in Science Engineering and Technology*, 2(4):83–89, 2014.

[57] Gwyn Griffiths, Mario Brito, Ian Robbins, and Mark Moline. Reliability of two REMUS-100 AUVs based on fault log analysis and elicited expert judgment. 2009.

[58] Fuqiang Liu, Zuxing Ma, Bingxian Mu, Chaoqun Duan, Rui Chen, Yi Qin, Huayan Pu, and Jun Luo. Review on fault-tolerant control of unmanned underwater vehicles. *Ocean Engineering*, 285:115471, 2023.

[59] Takeshi Nakatani, Tamaki Ura, Yuzuru Ito, Junichi Kojima, Kenkichi Tamura, Takashi Sakamaki, and Yoshiaki Nose. AUV TUNA-SAND and its exploration of hydrothermal vents at Kagoshima Bay. In *OCEANS 2008-MTS/IEEE Kobe Techno-Ocean*, pages 1–5. IEEE, 2008.

[60] Alfredo Martins, José Almeida, Carlos Almeida, Bruno Matias, Stef Kapusniak, and Eduardo Silva. EVA a hybrid ROV/AUV for underwater mining operations support. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–7. IEEE, 2018.

[61] Ramon A Suarez Fernandez, Zorana Milošević, Sergio Dominguez, and Claudio Rossi. Motion control of underwater mine explorer robot UX-1: field trials. *IEEE Access*, 7:99782–99803, 2019.

[62] Woods Hole Oceanographic Institution. Pioneering deep-sea robot lost at sea. https://www.whoi.edu/press-room/news-release/pioneering-deep-sea-robot-lost-at-sea/, (Accessed: 16/09/2024).

[63] Stephen McPhail, Rob Templeton, Miles Pebody, Daniel Roper, and Richard Morrison. Autosub long range AUV missions under the Filchner and Ronne ice shelves in the Weddell sea, Antarctica-an engineering perspective. In *OCEANS 2019-Marseille*, pages 1–8. IEEE, 2019.

[64] Rixia Qin, Xiaohong Zhao, Wenbo Zhu, Qianqian Yang, Bo He, Guangliang Li, and Tianhong Yan. Multiple receptive field network (mrf-net) for autonomous underwater vehicle fishing net detection using forward-looking sonar images. *Sensors*, 21(6):1933, 2021.

[65] BBC News. Why are chinese fishermen finding so many 'submarine spies'? https://www.bbc.com/news/world-asia-china-51130644, (Accessed: 16/09/2024).

[66] BBC News. Fisherman catches underwater drone with chinese characteristics. https://www.maritime-executive.com/article/fisherman-catches-underwater-drone-with-chinese-characteristics, (Accessed: 16/09/2024).

[67] Forbes. China discovers underwater spy drones in its waters. https://www.forbes.com/sites/hisutton/2020/01/15/china-discovers-underwater-spy-drones-in-its-waters/?sh\unhbox\voidb@x\bgroup\let\unhbox\voidb@x\setbox\@tempboxa\hbox{7\global\mathchardef\accent@spacefactor\spacefactor}\let\begingroup\let\typeout\protect\begingroup\def\MessageBreak{˙(Font)}\let\protect\immediate\write\m@ne{LaTeXFontInfo:\def{}oninputline336.}\endgroup\endgroup\relax\let\ignorespaces\relax\accent97\egroup\spacefactor\accent@spacefactorbef482f6990, (Accessed: 16/09/2024).

[68] Charitha Pattiaratchi, L Mun Woo, Paul G Thomson, Kah Kiat Hong, and Dennis Stanley. Ocean glider observations around Australia. *Oceanography*, 30(2):90–91, 2017.

[69] M Jordan Stanway, Brian Kieft, Thomas Hoover, Brett Hobson, Denis Klimov, Jon Erickson, Ben Yair Raanan, David A Ebert, and James Bellingham. White shark strike on a long-range AUV in Monterey Bay. In *OCEANS 2015-Genova*, pages 1–7. IEEE, 2015.

[70] University of Western Australia. Shark tries to take chunk out of 250k underwater glider off Perth coast. `https://www.abc.net.au/news/2015-09-29/shark-tries-to-eat-uwa-underwater-glider-off-perth-yanchep-coast/6813166`, (Accessed: 16/09/2024).

[71] Woods Hole Oceanographic Institution. REMUS SharkCam: The hunter and the hunted. `https://www.youtube.com/watch?v=faZw3IFJOXs`, (Accessed: 16/09/2024).

[72] Enrico Anderlini, Giles Thomas, Stephen CA Woodward, Daniel A Real-Arce, Tania Morales, Carlos Barrera, and JJ Hernández-Brito. Identification of the dynamics of biofouled underwater gliders. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pages 1–6. IEEE, 2020.

[73] Rolf Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2005.

[74] Krzysztof Patan. Robust and fault-tolerant control: neural-network-based solutions. 2019.

[75] Janos J Gertler. Survey of model-based failure detection and isolation in complex plants. *IEEE Control systems magazine*, 8(6):3–11, 1988.

[76] Paul M Frank and Birgit Köppen-Seliger. New developments using AI in fault diagnosis. *Engineering Applications of Artificial Intelligence*, 10(1):3–14, 1997.

[77] Zhiwei Gao, Carlo Cecati, and Steven X Ding. A survey of fault diagnosis and fault-tolerant techniques—Part i: Fault diagnosis with model-based and signal-based approaches. *IEEE transactions on industrial electronics*, 62(6):3757–3767, 2015.

[78] Alireza Abbaspour, Sohrab Mokhtari, Arman Sargolzaei, and Kang K Yen. A Survey on Active Fault-Tolerant Control Systems. *Electronics*, 9(9):1513, 2020.

[79] Younghwan An. *A design of fault tolerant flight control systems for sensor and actuator failures using on-line learning neural networks.* PhD thesis, West Virginia University, 2001.

[80] Francesca Boem, Alexander J Gallo, Davide M Raimondo, and Thomas Parisini. Distributed fault-tolerant control of large-scale systems: An active fault diagnosis approach. *IEEE Transactions on Control of Network Systems*, 7(1):288–301, 2019.

[81] Afef Fekih. Fault-tolerant flight control design for effective and reliable aircraft systems. *Journal of Control and Decision*, 1(4):299–316, 2014.

[82] Kumpati S Narendra and Jeyendran Balakrishnan. Adaptive control using multiple models. *IEEE transactions on automatic control*, 42(2):171–187, 1997.

[83] Thomas Steffen. *Control reconfiguration of dynamical systems: linear approaches and structural tests*, volume 320. Springer Science & Business Media, 2005.

[84] Het Joshi and Nandan K Sinha. Adaptive fault tolerant control design for stratospheric airship with actuator faults. *IFAC-PapersOnLine*, 55(1):819–825, 2022.

[85] X-J Li and G-H Yang. Robust adaptive fault-tolerant control for uncertain linear systems with actuator failures. *IET control theory & applications*, 6(10):1544–1551, 2012.

[86] Basil Kouvaritakis and Mark Cannon. Model predictive control. *Switzerland: Springer International Publishing*, 38, 2016.

[87] Lalo Magni and Riccardo Scattolini. *Advanced and multivariable control*. Società Editrice Esculapio, 2023.

[88] Fang Nan, Sihao Sun, Philipp Foehn, and Davide Scaramuzza. Nonlinear MPC for quadrotor fault-tolerant control. *IEEE Robotics and Automation Letters*, 7(2):5047–5054, 2022.

[89] Tarun Kanti Podder, Gianluca Antonelli, and Nilanjan Sarkar. Fault tolerant control of an autonomous underwater vehicle under thruster redundancy: Simulations and experiments. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 1251–1256. IEEE, 2000.

[90] Tor A Johansen and Thor I Fossen. Control allocation—a survey. *Automatica*, 49(5):1087–1103, 2013.

[91] Albert N Andry, Eliezer Y Shapiro, and JC Chung. Eigenstructure assignment for linear systems. *IEEE transactions on aerospace and electronic systems*, (5):711–729, 1983.

[92] Jean-Jacques E Slotine et al. *Applied nonlinear control*. 1991.

[93] Sarah Spurgeon. Sliding mode control: a tutorial. In *2014 European Control Conference (ECC)*, pages 2272–2277. IEEE, 2014.

[94] Tao Wang, Wenfang Xie, and Youmin Zhang. Sliding mode fault tolerant control dealing with modeling uncertainties and actuator faults. *ISA transactions*, 51(3):386–392, 2012.

[95] Vadim Utkin and Hoon Lee. Chattering problem in sliding mode control systems. In *International Workshop on Variable Structure Systems, 2006. VSS'06.*, pages 346–350. IEEE, 2006.

[96] Serdar Soylu, Bradley J Buckham, and Ron P Podhorodeski. A chattering-free sliding-mode controller for underwater vehicles with fault-tolerant infinity-norm thrust allocation. *Ocean Engineering*, 35(16):1647–1659, 2008.

[97] Jin Jiang and Xiang Yu. Fault-tolerant control systems: A comparative study between active and passive approaches. *Annual Reviews in control*, 36(1):60–72, 2012.

[98] Youmin Zhang and Jin Jiang. Bibliographical review on reconfigurable fault-tolerant control systems. *Annual reviews in control*, 32(2):229–252, 2008.

[99] Stoyan Kamenov Kanev. *Robust fault-tolerant control*. PhD thesis, University of Twente, 2004.

[100] Shashikanth Suryanarayanan, Masayoshi Tomizuka, and Tatsuya Suzuki. Design of simultaneously stabilizing controllers and its application to fault-tolerant lane-keeping controller design for automated vehicles. *IEEE Transactions on Control Systems Technology*, 12(3):329–339, 2004.

[101] Pierre Apkarian and Dominikus Noll. Nonsmooth H_∞ synthesis. *IEEE Transactions on Automatic Control*, 51(1):71–86, 2006.

[102] Daniel Ankelhed. *On design of low order H-infinity controllers*. PhD thesis, Linköping University Electronic Press, 2011.

[103] Tom M Mitchell and Tom M Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[104] Mahbubul Alam, Manar D Samad, Lasitha Vidyaratne, Alexander Glandon, and Khan M Iftekharuddin. Survey on deep neural networks in speech and vision systems. *Neurocomputing*, 417:302–321, 2020.

[105] Amir Ramezani Dooraki and Deok-Jin Lee. Reinforcement learning based flight controller capable of controlling a quadcopter with four, three and two working motors. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pages 161–166. IEEE, 2020.

[106] Melrose Roderick. *Ensuring the Safety of Reinforcement Learning Algorithms at Training and Deployment*. PhD thesis, Carnegie Mellon University, 2023.

[107] Gianluca Antonelli and Gianluca Antonelli. *Modelling of underwater robots*. Springer, 2014.

[108] Tarun Kanti Podder and Nilanjan Sarkar. Fault-tolerant control of an autonomous underwater vehicle under thruster redundancy. *Robotics and Autonomous Systems*, 34(1):39–52, 2001.

[109] Jin-Kyu Choi and Hayato Kondo. On fault-tolerant control of a hovering AUV with four horizontal and two vertical thrusters. In *OCEANS'10 IEEE SYDNEY*, pages 1–6. IEEE, 2010.

[110] Edin Omerdic and Geoff Roberts. Thruster fault diagnosis and accommodation for open-frame underwater vehicles. *Control engineering practice*, 12(12):1575–1598, 2004.

[111] Nilanjan Sarkar, Tarun Kanti Podder, and Gianluca Antonelli. Fault-accommodating thruster force allocation of an AUV considering thruster redundancy and saturation. *IEEE Transactions on Robotics and Automation*, 18(2):223–233, 2002.

[112] Dana R Yoerger, John G Cooke, and J-JE Slotine. The influence of thruster dynamics on underwater vehicle behavior and their incorporation into control system design. *IEEE Journal of Oceanic Engineering*, 15(3):167–178, 1990.

[113] Xing Liu, Mingjun Zhang, and Feng Yao. Adaptive fault tolerant control and thruster fault reconstruction for autonomous underwater vehicle. *Ocean Engineering*, 155:10–23, 2018.

[114] Xuelian Ding, Daqi Zhu, and Mingzhong Yan. Research on static fault-tolerant control method of thruster based on MPC. *Journal of Marine Science and Technology*, 26:861–871, 2021.

[115] Rafael Meireles Saback, Andre Gustavo Scolari Conceicao, Tito Luís Maia Santos, Jan Albiez, and Marco Reis. Nonlinear model predictive control applied to an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 45(3):799–812, 2019.

[116] Tu Lv, Junliang Zhou, Yujia Wang, Wei Gong, and Mingjun Zhang. Sliding mode based fault tolerant control for autonomous underwater vehicle. *Ocean Engineering*, 216:107855, 2020.

[117] MR Katebi and Michael J Grimble. Integrated control, guidance and diagnosis for reconfigurable autonomous underwater vehicle control. *International Journal of Systems Science*, 30(9):1021–1032, 1999.

[118] Isaac Kaminer, Antonio M Pascoal, Carlos J Silvestre, and Pramod P Khargonekar. Control of an underwater vehicle using H_∞ synthesis. In *[1991] Proceedings of the 30th IEEE Conference on Decision and Control*, pages 2350–2355. IEEE, 1991.

[119] Cedric L Logan. A comparison between H-infinity/mu-synthesis control and sliding-mode control for robust control of a small autonomous underwater vehicle. In *Proceedings of IEEE Symposium on Autonomous Underwater Vehicle Technology (AUV'94)*, pages 399–416. IEEE, 1994.

[120] P Varzandeh and Amir Hooshang Mazinan. Neural network-based fault-tolerant control approach considering a submarine system. *Evolving Systems*, 12:913–922, 2021.

[121] Peng Wu. *Decarbonising coastal shipping using fuel cells and batteries*. PhD thesis, UCL (University College London), 2020.

[122] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[123] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[124] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.

[125] Mark R Baker and Rajendra B Patil. Universal approximation theorem for interval neural networks. *Reliable Computing*, 4(3):235–239, 1998.

[126] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[127] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[128] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[129] Prashant Bhopale, Faruk Kazi, and Navdeep Singh. Reinforcement learning based obstacle avoidance for autonomous underwater vehicle. *Journal of Marine Science and Application*, 18:228–238, 2019.

[130] Jianya Yuan, Hongjian Wang, Honghan Zhang, Changjian Lin, Dan Yu, and Chengfeng Li. Auv obstacle avoidance planning based on deep reinforcement learning. *Journal of Marine Science and Engineering*, 9(11):1166, 2021.

[131] Zhuo Wang, Shiwei Zhang, Xiaoning Feng, and Yancheng Sui. Autonomous underwater vehicle path planning based on actor-multi-critic reinforcement learning. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 235(10):1787–1796, 2021.

[132] Enrico Anderlini, Gordon G Parker, and Giles Thomas. Docking control of an autonomous underwater vehicle using reinforcement learning. *Applied Sciences*, 9(17):3456, 2019.

[133] Khalid Isa and Mohd Rizal Arshad. Neural networks control of hybrid-driven underwater glider. In *2012 Oceans-Yeosu*, pages 1–7. IEEE, 2012.

[134] Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural Lyapunov Control. *Advances in Neural Information Processing Systems*, 32, 2019.

[135] Alessandro Abate, Daniele Ahmed, Mirco Giacobbe, and Andrea Peruffo. Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, 5(3):773–778, 2020.

[136] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pages 1–11, 2021.

[137] Rosana CB Rego and Fábio MU de Araújo. Learning-based robust neuro-control: A method to compute control Lyapunov functions. *International Journal of Robust and Nonlinear Control*, 32(5):2644–2661, 2022.

[138] Hongkai Dai, Benoit Landry, Lujie Yang, Marco Pavone, and Russ Tedrake. Lyapunov-stable neural-network control. *arXiv preprint arXiv:2109.14152*, 2021.

[139] Junlin Wu, Andrew Clark, Yiannis Kantaros, and Yevgeniy Vorobeychik. Neural Lyapunov Control for discrete-time systems. *Advances in Neural Information Processing Systems*, 36, 2024.

[140] Andrea Caffaz, Andrea Caiti, Giuseppe Casalino, and Alessio Turetta. The hybrid glider/AUV folaga. *IEEE Robotics & Automation Magazine*, 17(1):31–44, 2010.

[141] Surasak Phoemsapthawee, Marc Le Boulluec, Jean-Marc Laurens, and François Deniset. An underwater glider flight simulator. 2010.

[142] Lei Kan, Yuwen Zhang, Hui Fan, Wugang Yang, and Zhikun Chen. MATLAB-based simulation of buoyancy-driven underwater glider motion. *Journal of Ocean University of China*, 7(1):113–118, 2008.

[143] Musa Morena Marcusso Manhães, Sebastian A Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *Oceans 2016 Mts/Ieee Monterey*, pages 1–8. Ieee, 2016.

[144] Manlio Oddone, Agostino Bruzzone, Emanuel Coelho, Daniele Cecchi, and Bartolome Garau. An underwater buoyancy-driven glider simulator with Modelling & Simulation as a Service architecture. In *Proceedings of Defense and Homeland Security Simulation Workshop, Barcelona*, 2017.

[145] Mabel M Zhang, Woen-Sug Choi, Jessica Herman, Duane Davis, Carson Vogt, Michael McCarrin, Yadunund Vijay, Dharini Dutia, William Lew, Steven Peters, et al. DAVE aquatic virtual environment: Toward a general underwater robotics simulator. In *2022 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pages 1–8. IEEE, 2022.

[146] John Nicholas Newman. *Marine hydrodynamics*. MIT press, 1977.

[147] Thor I Fossen. Guidance and control of ocean vehicles. *University of Trondheim, Norway, Printed by John Wiley & Sons, Chichester, England, ISBN: 0 471 94113 1, Doctors Thesis*, 1999.

[148] Timothy Prestero. *Verification of a six-degree of freedom simulation model for the REMUS autonomous underwater vehicle*. PhD thesis, Massachusetts institute of technology, 2001.

[149] Stephen Wood and AV Inzartsev. *Autonomous underwater gliders*. Citeseer, 2009.

[150] Ramon A Suarez Fernandez, Davide Grande, Alfredo Martins, Luca Bascetta, Sergio Dominguez, and Claudio Rossi. Modeling and control of underwater mine explorer robot UX-1. *IEEE Access*, 7:39432–39447, 2019.

[151] Douglas C Webb, Paul J Simonetti, and Clayton P Jones. SLOCUM: An underwater glider propelled by environmental energy. *IEEE Journal of oceanic engineering*, 26(4):447–452, 2001.

[152] Jeff Sherman, Russ E Davis, WB Owens, and J Valdes. The autonomous underwater glider" spray". *IEEE Journal of Oceanic Engineering*, 26(4):437–446, 2001.

[153] Charles C Eriksen, T James Osse, Russell D Light, Timothy Wen, Thomas W Lehman, Peter L Sabin, John W Ballard, and Andrew M Chiodi. Seaglider: A long-range autonomous underwater vehicle for oceanographic research. *IEEE Journal of oceanic Engineering*, 26(4):424–436, 2001.

[154] T James Osse and Charles C Eriksen. The Deepglider: A full ocean depth glider for oceanographic research. In *OCEANS 2007*, pages 1–12. IEEE, 2007.

[155] Khalid Isa, MR Arshad, and Syafizal Ishak. A hybrid-driven underwater glider model, hydrodynamics estimation, and an analysis of the motion control. *Ocean Engineering*, 81:111–129, 2014.

[156] Yan Huang, Jianan Qiao, Jiancheng Yu, Zhenyu Wang, Zongbo Xie, and Kai Liu. SeaWhale 2000: A long-range hybrid autonomous underwater vehicle for ocean observation. In *Oceans 2019-Marseille*, pages 1–6. IEEE, 2019.

[157] Aa Alvarez, Ab Caffaz, Andrea Caiti, Giuseppe Casalino, Lavinio Gualdesi, Alessio Turetta, and R Viviani. Folaga: A low-cost autonomous underwater vehicle combining glider and AUV capabilities. *Ocean engineering*, 36(1):24–38, 2009.

[158] J Kofránek, M Mateják, P Privitzer, and M Tribula. Causal or acausal modeling: labour for humans or labour for machines. *Technical computing prague*, pages 1–16, 2008.

[159] Gianni Ferretti, GianAntonio Magnani, and Paolo Rocco. Virtual prototyping of mechatronic systems in Modelica. *IFAC Proceedings Volumes*, 35(2):791–796, 2002.

[160] Ezril Hisham Mat Saat, Rosbi Mamat, and O Yaakob. Design and implementation of electronic control system for UTM-AUV. In *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No. 00CH37119)*, volume 2, pages 332–337. IEEE, 2000.

[161] João Lucas Dozzi Dantas and Ettore Apolonio de Barros. A real-time simulator for AUV development. In *ABCM symposium series in mechatronics*, volume 4, pages 538–549, 2010.

[162] Simon A Watson and Peter N Green. Depth control for micro-autonomous underwater vehicles (μAUVs): Simulation and experimentation. *International Journal of Advanced Robotic Systems*, 11(3):31, 2014.

[163] Mohammad Hedayati Khodayari and Saeed Balochian. Modeling and control of autonomous underwater vehicle (AUV) in heading and depth attitude via self-adaptive fuzzy pid controller. *Journal of Marine Science and Technology*, 20(3):559–578, 2015.

[164] Sriharsha Bhat, Chariklia Panteli, Ivan Stenius, and Dimos V Dimarogonas. Nonlinear model predictive control for hydrobatics: Experiments with an underactuated AUV. *Journal of Field Robotics*, 40(7):1840–1859, 2023.

[165] Tom Egel. Real time simulation using non-causal physical models. Technical report, SAE Technical Paper, 2009.

[166] Michael M. Tiller. Modelica by example. `https://mbe.modelica.university/components/components/component_models/#acausal-modeling`, (Accessed: 16/09/2024).

[167] Francesco Casella et al. Simulation of large-scale models in modelica: State of the art and future perspectives. In *Linköping electronic conference proceedings*, pages 459–468, 2015.

[168] Hilding Elmqvist. *A structured model language for large continuous systems*. PhD thesis, Department of Automatic Control, Lund Institute of Technology (LTH), 1978.

[169] Günther Zauner, Daniel Leitner, and Felix Breitenecker. Modeling Structural-Dynamics Systems in MODELICA/Dymola, MODELICA/Mosilab and AnyLogic. In *EOOLT*, pages 99–110, 2007.

[170] Peter Fritzson, Adrian Pop, Adeel Asghar, Bernhard Bachmann, Willi Braun, Robert Braun, Lena Buffoni, Francesco Casella, Rodrigo Castro, Alejandro Danós, et al. The

OpenModelica integrated modeling, simulation and optimization environment. In *Proceedings of the 1st American Modelica Conference*, pages 8–10. Modelica Association, 2018.

[171] The Functional Mock-up Interface beginners' tutorial. `https://github.com/modelica/fmi-beginners-tutorial-2023/blob/main/part1/Introduction-to-FMI.pdf`, (Accessed: 16/09/2024).

[172] Stefano Farì, Marco Sagliano, José Alfredo Macés Hernández, Anton Schneider, Ansgar Heidecker, Markus Schlotterer, and Svenja Woicke. Physical Modeling and Simulation of Reusable Rockets for GNC Verification and Validation. *Aerospace*, 11(5):337, 2024.

[173] Johan Andreasson. VehicleDynamics library. In *Proceedings of the 3rd International Modelica Conference, Linköping*, 2003.

[174] Gertjan Looye, Simon Hecker, Thiemo Kier, and Christian Reschke. FlightDynLib: An object-oriented model component library for constructing multi-disciplinary aircraft dynamics models. 2005.

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

[176] Millard F Beatty and Millard F Beatty. Kinematics of rigid body motion. *Principles of Engineering Mechanics: Kinematics—The Geometry of Motion*, pages 85–149, 1986.

[177] Thor Inge Fossen. *Nonlinear modelling and control of underwater vehicles*. PhD thesis, Universitetet i Trondheim (Norway), 1991.

[178] Naomi Ehrich Leonard and Joshua G Graver. Model-based feedback control of autonomous underwater gliders. *IEEE Journal of oceanic engineering*, 26(4):633–645, 2001.

[179] Shaowei Zhang, Jiancheng Yu, Aiqun Zhang, and Fumin Zhang. Spiraling motion of underwater gliders: Modeling, analysis, and experimental results. *Ocean Engineering*, 60:1–13, 2013.

[180] Slocum G2 - technical specification. `http://gliderfs.coas.oregonstate.edu/gliderweb/docs/slocum_manuals/Slocum_G2_Glider_Operators_Manual.pdf`, (Accessed: 16/09/2024).

[181] Slocum G3 - technical specification. `http://gliderfs.coas.oregonstate.edu/gliderweb/docs/slocum_manuals/Slocum_G3_Operator_Manual_20171219.pdf`, (Accessed: 16/09/2024).

[182] Society of Naval Architects, Marine Engineers (U.S.). Technical, and Research Committee. Hydrodynamics Subcommittee. Nomenclature for treating the motion of a submerged body through a fluid. *T&R Bulletin*, pages 1–5, 1950.

[183] Sighard F Hoerner. Fluid Dynamic Drag, published by the author. *Midland Park, NJ*, pages 16–35, 1965.

[184] Roy Burcher and Louis J Rydill. *Concepts in submarine design*, volume 2. Cambridge university press, 1995.

[185] Edoardo I Sarda, Huajin Qu, Ivan R Bertaska, and Karl D von Ellenrieder. Station-keeping control of an unmanned surface vehicle exposed to current and wind disturbances. *Ocean Engineering*, 127(3):305–324, 2016.

[186] Frederick H Imlay. The complete expressions for" added mass" of a rigid body moving in an ideal fluid. Technical report, United States Department of the Navy, David Taylor Model Basin, 1961.

[187] James S Bennett, Frederick R Stahr, Charles C Eriksen, Martin C Renken, Wendy E Snyder, and Lora J Van Uffelen. Assessing seaglider model-based position accuracy on an acoustic tracking range. *Journal of Atmospheric and Oceanic Technology*, 38(6):1111–1123, 2021.

[188] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Force control*. Springer, 2009.

[189] Feitian Zhang, Jianxun Wang, John Thon, Cody Thon, Elena Litchman, and Xiaobo Tan. Gliding robotic fish for mobile sampling of aquatic environments. In *Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control*, pages 167–172. IEEE, 2014.

[190] Peter Ridley, Julien Fontan, and Peter Corke. Submarine dynamic modeling. In *Proceedings of the Australian Conference on Robotics and Automation*, pages 1–8. Australian Robotics and Automation Association, 2003.

[191] Feitian Zhang and Xiaobo Tan. Passivity-based stabilization of underwater gliders with a control surface. *Journal of Dynamic Systems, Measurement, and Control*, 137(6):061006, 2015.

[192] Hexiong Zhou, Zhaoyu Wei, Zheng Zeng, Caoyang Yu, Baoheng Yao, and Lian Lian. Adaptive robust sliding mode control of autonomous underwater glider with input constraints for persistent virtual mooring. *Applied Ocean Research*, 95:102027, 2020.

[193] Jun-liang Cao, Bao-heng Yao, and Lian Lian. Nonlinear pitch control of an underwater glider based on adaptive backstepping approach. *Journal of Shanghai Jiaotong University (Science)*, 20:729–734, 2015.

[194] Davide Grande. Modelling and simulation of a spherical vehicle for underwater surveillance. Master's thesis, Politecnico di Milano, 2018.

[195] Davide Grande, Luca Bascetta, and Alfredo Martins. Modeling and simulation of a spherical vehicle for underwater surveillance. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–7. IEEE, 2018.

[196] JG Graver, Jonathan Liu, Craig Woolsey, and Naomi Ehrich Leonard. Design and analysis of an underwater vehicle for controlled gliding. In *Proc. 32nd Conference on Information Sciences and Systems*, pages 801–806, 1998.

[197] Hassan K Khalil. *Nonlinear control*, volume 406. Pearson New York, 2015.

[198] Fabio Dercole, Sergio Rinaldi, et al. Dynamical systems and their bifurcations. *Advanced methods of biomedical signal processing*, pages 291–325, 2011.

[199] Navid Noroozi, Paknoosh Karimaghaee, Fatemeh Safaei, and Hamed Javadi. Generation of Lyapunov functions by neural networks. In *Proceedings of the World Congress on Engineering*, volume 2008, 2008.

[200] Jean Mawhin. Alexandr Mikhailovich Lyapunov, thesis on the stability of motion (1892). In *Landmark Writings in Western Mathematics 1640-1940*, pages 664–676. Elsevier, 2005.

[201] Eduardo D Sontag. *Mathematical control theory: deterministic finite dimensional systems*, volume 6. Springer Science & Business Media, 2013.

[202] Russ Tedrake. *Underactuated Robotics*. 2023.

[203] James Anderson and Antonis Papachristodoulou. Advances in computational lyapunov analysis using sum-of-squares programming. *Discrete & Continuous Dynamical Systems-Series B*, 20(8), 2015.

[204] Antonis Papachristodoulou and Stephen Prajna. On the construction of lyapunov functions using the sum of squares decomposition. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 3482–3487. IEEE, 2002.

[205] Mohammad Sabouri, Peyman Setoodeh, and Mohammad Hassan Asemani. Construction of Lyapunov functions using multi-objective genetic algorithm. In *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pages 1–5. IEEE, 2020.

[206] CF Verdier and M Mazo Jr. Formal controller synthesis via genetic programming. *IFAC-PapersOnLine*, 50(1):7205–7210, 2017.

[207] Danil V Prokhorov. A lyapunov machine for stability analysis of nonlinear systems. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 2, pages 1028–1031. IEEE, 1994.

[208] Charles Dawson, Sicun Gao, and Chuchu Fan. Safe Control With Learned Certificates: A Survey of Neural Lyapunov, Barrier, and Contraction Methods for Robotics and Control. *IEEE Transactions on Robotics*, 2023.

[209] Roberto Sebastiani. Lazy satisfiability modulo theories. *Journal on Satisfiability, Boolean Modeling and Computation*, 3(3-4):141–224, 2007.

[210] Daniel Kroening and Ofer Strichman. *Decision procedures*. Springer, 2016.

[211] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. Combinatorial sketching for finite programs. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, pages 404–415, 2006.

[212] Alessandro Abate, Cristina David, Pascal Kesseli, Daniel Kroening, and Elizabeth Polgreen. Counterexample guided inductive synthesis modulo theories. In *International Conference on Computer Aided Verification*, pages 270–288. Springer, 2018.

[213] Hongkai Dai, Benoit Landry, Marco Pavone, and Russ Tedrake. Counter-example guided synthesis of neural network Lyapunov functions for piecewise linear systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1274–1281. IEEE, 2020.

[214] Daniele Masti, Filippo Fabiani, Giorgio Gnecco, and Alberto Bemporad. Counter-example guided inductive synthesis of control Lyapunov functions for uncertain systems. *IEEE Control Systems Letters*, 2023.

[215] Sicun Gao, Jeremy Avigad, and Edmund M Clarke. δ-complete decision procedures for satisfiability over the reals. In *International Joint Conference on Automated Reasoning*, pages 286–300. Springer, 2012.

[216] Sicun Gao, Soonho Kong, and Edmund M Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *International conference on automated deduction*, pages 208–214. Springer, 2013.

[217] Soonho Kong, Armando Solar-Lezama, and Sicun Gao. Delta-decision procedures for exists-forall problems over the reals. In *International Conference on Computer Aided Verification*, pages 219–235. Springer, 2018.

[218] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[219] Aaron Stump, Clark W Barrett, and David L Dill. CVC: A cooperating validity checker. In *Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings 14*, pages 500–504. Springer, 2002.

[220] Sicun Gao, James Kapinski, Jyotirmoy Deshmukh, Nima Roohi, Armando Solar-Lezama, Nikos Aréchiga, and Soonho Kong. Numerically-robust inductive proof rules for continuous dynamical systems. In *Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II 31*, pages 137–154. Springer, 2019.

[221] Joseph La Salle and Solomon Lefschetz. Stability by Liapunov's Direct Method with Applications. *Academic Press, RIAS*, 1961.

[222] Lars Grüne. Computing Lyapunov functions using deep neural networks. *arXiv preprint arXiv:2005.08965*, 2020.

[223] Shunichi Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, (3):299–307, 1967.

[224] Michael Sipser. Introduction to the Theory of Computation. *ACM Sigact News*, 27(1):27–29, 1996.

[225] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[226] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[227] Grégoire Montavon, Geneviève Orr, and Klaus-Robert Müller. *Neural networks: tricks of the trade*, volume 7700. springer, 2012.

[228] Đorđe Žikelić, Mathias Lechner, Krishnendu Chatterjee, and Thomas A Henzinger. Learning stabilizing policies in stochastic control systems. *arXiv preprint arXiv:2205.11991*, 2022.

[229] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[230] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[231] R Choroszucha. Control of the Lorenz equations. *Department of Naval and Marine Engineering, University of Michigan*, 2000.

[232] Pedro Pablo Cardenas Alzate, German Correa Velez, and Fernando Mesa. Chaos control for the Lorenz system. *Advanced Studies in Theoretical Physics*, 12(4):181–188, 2018.

[233] Xiang-qin Cheng, Jing-yuan Qu, Zhe-ping Yan, and Xin-qian Bian. H_∞ robust fault-tolerant controller design for an autonomous underwater vehicle's navigation control system. *Journal of Marine science and Application*, 9(1):87–92, 2010.

[234] Daniele Ahmed, Andrea Peruffo, and Alessandro Abate. Automated and sound synthesis of Lyapunov functions with SMT solvers. In *Tools and Algorithms for the Construction and Analysis of Systems: 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25–30, 2020, Proceedings, Part I 26*, pages 97–114. Springer, 2020.

[235] Alec Edwards, Andrea Peruffo, and Alessandro Abate. A General Verification Framework for Dynamical and Control Models via Certificate Synthesis, 2023. arXiv:2309.06090.

[236] Rudolf E Kalman. On the general theory of control systems. In *Proceedings First International Conference on Automatic Control, Moscow, USSR*, pages 481–492, 1960.

[237] Poh Wong and Michael Athans. Closed-loop structural stability for linear-quadratic optimal systems. *IEEE Transactions on Automatic Control*, 22(1):94–99, 1977.

[238] Michael Safonov and Michael Athans. Gain and phase margin for multiloop LQG regulators. *IEEE Transactions on Automatic Control*, 22(2):173–179, 1977.

[239] Arthur E Bryson. Applied optimal control: Optimization. *Estimization and Control*, 2, 1975.

[240] Alexander B Phillips, Matthew Kingsland, Nick Linton, Will Baker, Leon Bowring, Scott Soper, Daniel T Roper, Alexis Johnson, Richard Morrison, Konrad Ciaramella, Daniel Matterson, Miles Pebody, Rachel Marlow, Alberto Consensi, Val Williams, Francesco Fanelli, Davide Fenucci, Achille Martin, and Eoin Ó hÓbáin. Autosub 2000 under ice: Design of a new work class AUV for under ice exploration. In *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pages 1–8. IEEE, 2020.

[241] Alessandro Baldini, Antonio Fasano, Riccardo Felicetti, Alessandro Freddi, Sauro Longhi, and Andrea Monteriù. A model-based active fault tolerant control scheme for a remotely operated vehicle. *IFAC-PapersOnLine*, 51(24):798–805, 2018.

[242] Josué González-García, Néstor Alejandro Narcizo-Nuci, Luis Govinda García-Valdovinos, Tomás Salgado-Jiménez, Alfonso Gómez-Espinosa, Enrique Cuan-Urquizo, and Jesús Arturo Escobedo Cabello. Model-free high order sliding mode control with finite-time tracking for unmanned underwater vehicles. *Applied Sciences*, 11(4):1836, 2021.

[243] Jianguang Shi and Mingxi Zhou. A data-driven intermittent online coverage path planning method for auv-based bathymetric mapping. *Applied Sciences*, 10(19):6688, 2020.

Appendix A

OpenMAUVe class implementation example

The implementation of the elevator control surface class, defined within *OpenMAUVe.Actuators.Elevator*, is reported with the code snippet presented hereby.

```
within OpenMAUVe.Actuators;
model Elevator "Force and moment generated by one elevator."
  Modelica.Mechanics.MultiBody.Interfaces.Frame_b frame_b
    annotation (...);

  parameter Real K_F_delta(unit="kg/(m.rad)") = 0.0 "control
    surface coupling factor";
  parameter Real K_u_delta = 0.0 "adimensional scaling constant";
  parameter Real K_M_delta(unit="kg/rad") = 0.0 "control surface
    moment coefficient";
  Real flowspeed(unit="m/s");
  Modelica.Mechanics.MultiBody.Sensors.AbsoluteVelocity
    absoluteVelocity(resolveInFrame =
    Modelica.Mechanics.MultiBody.Types.ResolveInFrameA.frame_a)
    annotation(...);
  Modelica.Blocks.Interfaces.RealInput delta_in annotation(...);
  Modelica.Mechanics.MultiBody.Forces.WorldForce force_stern_plane(
    resolveInFrame =
    Modelica.Mechanics.MultiBody.Types.ResolveInFrameB.frame_b)
    annotation(...);
```

```
12   Modelica.Mechanics.MultiBody.Forces.WorldTorque
      torque_stern_plane(resolveInFrame =
      Modelica.Mechanics.MultiBody.Types.ResolveInFrameB.frame_b)
      annotation(...);
13 equation
14   flowspeed = Modelica.Math.Vectors.norm(absoluteVelocity.v, 2);
15   force_stern_plane.force = {0.0, 0.0, K_F_delta * K_u_delta *
      delta_in * flowspeed ^ 2};
16   torque_stern_plane.torque = {0.0, -K_M_delta * delta_in *
      flowspeed ^ 2, 0.0};
17   connect(absoluteVelocity.frame_a, frame_b) annotation(...);
18   connect(absoluteVelocity.frame_a, frame_b) annotation(...);
19   connect(force_stern_plane.frame_b, frame_b) annotation(...);
20   connect(torque_stern_plane.frame_b, frame_b) annotation(...);
21 end Elevator;
```

Listing A.1: *OpenMAUVe.Actuators.Elevator* class code snippet.

Lines 1 and 2 of the code snippet define the structure of the package and the overall class description.

With the keyword *parameter*, the three scalar coefficients involved in Eq. (4.22a) and Eq. (4.22b) are defined, namely K_{F_δ}, K_{u_δ} and K_M. The associated units and default values are provided (in this case arbitrarily set to 0.0).

The expression of the *flowspeed* from Eq. (4.3), of the lift force from Eq. (4.22a) and lift moment from Eq. (4.22b) are implemented in the *equation* section (lines 14-16).

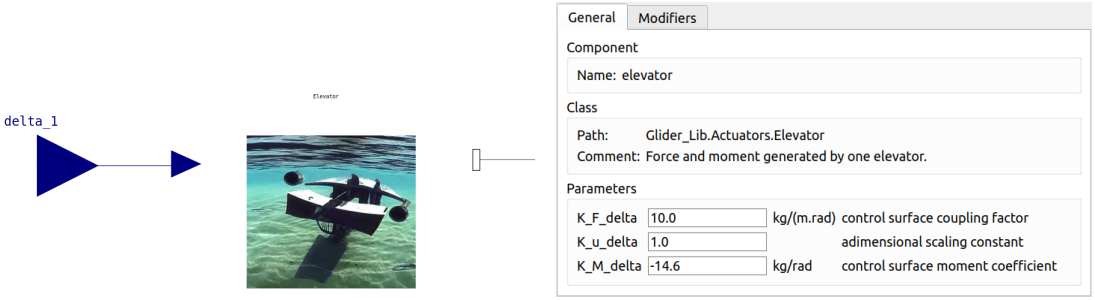
The keyword *annotation* provides detail as regards the graphic elements used and the position of the connectors.

The resulting class *OpenMAUVe.Actuators.Elevator* exhibits one real scalar input, namely the deflection angle (δ_1), and a set of forces and moments as outputs. The final icon view is reported in Fig. A.1a[1].

The class *OpenMAUVe.Actuators.Elevator* can now be imported by *dragging-and-dropping* the corresponding icon. The parameters of the instantiated object can

[1]The images used for the icons are AI-generated: https://deepai.org/machine-learning-model/text2img

be modified by selecting the icon view and updating the values in the corresponding interface, as illustrated in Fig. A.1b.



(a) Drag and drop icon. **(b)** Parameters interface.

Figure A.1: *Elevator* component interfaces.

Appendix B

Example of a CLF synthesised with the ANLC method

Example of the synthesised CLF associated to Fig. 5.11 for the inverted pendulum case, as returned by dReal:

$$V(\boldsymbol{x}) = (-0.1951 * pow((-0.173523 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.0572846 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.0733083 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0905027 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.267289 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0372416 * (0.115603 * x_1 - 0.559943 * x_2) - 0.175913 * (0.178242 * x_1 + 0.147661 * x_2) - 0.306655 * (0.189229 * x_1 - 0.212302 * x_2) + 0.310249 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0936387 * (0.327156 * x_1 - 0.407098 * x_2)), 2) + 0.150784 * pow((-0.157337 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.150808 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.206556 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.176699 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.125142 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.00291987 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0495719 * (0.178242 * x_1 + 0.147661 * x_2) - 0.331928 * (0.189229 * x_1 - 0.212302 * x_2) - 0.445404 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0962845 * (0.327156 * x_1 - 0.407098 * x_2)), 2) + 0.237683 * pow((-0.148784 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0132134 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.228939 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0529093 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.185947 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0333771 * (0.115603 * x_1 - 0.559943 * x_2) -$$

$$0.0496193*(0.178242*x_1+0.147661*x_2)-0.337578*(0.189229*x_1-0.212302*x_2)-0.206439*(0.228434*x_1-0.0207164*x_2)-0.191168*(0.327156*x_1-0.407098*x_2)),2)-0.0845882*pow((-0.141529*(-0.293793*x_1+0.62935*x_2)-0.00379008*(-0.261837*x_1+0.474018*x_2)+0.050696*(-0.163165*x_1-0.536536*x_2)-0.107278*(0.0589481*x_1+0.0207023*x_2)+0.164167*(0.0736858*x_1+0.156368*x_2)+0.112903*(0.115603*x_1-0.559943*x_2)-0.28331*(0.178242*x_1+0.147661*x_2)+0.251034*(0.189229*x_1-0.212302*x_2)-0.0055419*(0.228434*x_1-0.0207164*x_2)-0.221779*(0.327156*x_1-0.407098*x_2)),2)+0.204022*pow((-0.102796*(-0.293793*x_1+0.62935*x_2)-0.245426*(-0.261837*x_1+0.474018*x_2)+0.0271749*(-0.163165*x_1-0.536536*x_2)+0.149373*(0.0589481*x_1+0.0207023*x_2)-0.20195*(0.0736858*x_1+0.156368*x_2)+0.187741*(0.115603*x_1-0.559943*x_2)+0.18074*(0.178242*x_1+0.147661*x_2)-0.0441101*(0.189229*x_1-0.212302*x_2)-0.237967*(0.228434*x_1-0.0207164*x_2)-0.216235*(0.327156*x_1-0.407098*x_2)),2)-0.149335*pow((-0.0391913*(-0.293793*x_1+0.62935*x_2)+0.123921*(-0.261837*x_1+0.474018*x_2)+0.17003*(-0.163165*x_1-0.536536*x_2)+0.0757615*(0.0589481*x_1+0.0207023*x_2)+0.201078*(0.0736858*x_1+0.156368*x_2)+0.252316*(0.115603*x_1-0.559943*x_2)-0.326691*(0.178242*x_1+0.147661*x_2)+0.00698547*(0.189229*x_1-0.212302*x_2)-0.00301863*(0.228434*x_1-0.0207164*x_2)-0.194609*(0.327156*x_1-0.407098*x_2)),2)-0.147767*pow((-0.0107183*(-0.293793*x_1+0.62935*x_2)+0.0956865*(-0.261837*x_1+0.474018*x_2)+0.162321*(-0.163165*x_1-0.536536*x_2)+0.108168*(0.0589481*x_1+0.0207023*x_2)-0.0291888*(0.0736858*x_1+0.156368*x_2)+0.100121*(0.115603*x_1-0.559943*x_2)+0.0900326*(0.178242*x_1+0.147661*x_2)-0.0554088*(0.189229*x_1-0.212302*x_2)-0.3618*(0.228434*x_1-0.0207164*x_2)+0.0837*(0.327156*x_1-0.407098*x_2)),2)-0.24489*pow((0.141513*(-0.293793*x_1+0.62935*x_2)-0.0752214*(-0.261837*x_1+0.474018*x_2)+0.00765232*(-0.163165*x_1-0.536536*x_2)-0.0884482*(0.0589481*x_1+0.0207023*x_2)+0.256175*(0.0736858*x_1+0.156368*x_2)-0.170602*(0.115603*x_1-0.559943*x_2)-0.166361*$$

$$(0.178242 * x_1 + 0.147661 * x_2) + 0.322339 * (0.189229 * x_1 - 0.212302 * x_2) - 0.197938 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0139234 * (0.327156 * x_1 - 0.407098 * x_2)), 2) + 0.206889 * pow((0.159732 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.107909 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.306389 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.0325058 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.30328 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.276108 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0709101 * (0.178242 * x_1 + 0.147661 * x_2) + 0.132233 * (0.189229 * x_1 - 0.212302 * x_2) - 0.410339 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.112674 * (0.327156 * x_1 - 0.407098 * x_2)), 2) + 0.237191 * pow((0.296381 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0594621 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.176919 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0108246 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.193618 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0796573 * (0.115603 * x_1 - 0.559943 * x_2) + 0.246453 * (0.178242 * x_1 + 0.147661 * x_2) + 0.179855 * (0.189229 * x_1 - 0.212302 * x_2) + 0.277703 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0339893 * (0.327156 * x_1 - 0.407098 * x_2)), 2))$$

Appendix C

Examples of a Lie derivative function synthesised with the ANLC method

Example of the Lie derivative function associated to Fig. 5.12 for the inverted pendulum system, as returned by dReal:

$$\dot{V}(\boldsymbol{x}) = ((x_2 * (0.017088 * (-0.173523 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.0572846 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.0733083 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0905027 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.267289 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0372416 * (0.115603 * x_1 - 0.559943 * x_2) - 0.175913 * (0.178242 * x_1 + 0.147661 * x_2) - 0.306655 * (0.189229 * x_1 - 0.212302 * x_2) + 0.310249 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0936387 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0704116 * (-0.157337 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.150808 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.206556 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.176699 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.125142 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.00291987 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0495719 * (0.178242 * x_1 + 0.147661 * x_2) - 0.331928 * (0.189229 * x_1 - 0.212302 * x_2) - 0.445404 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0962845 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0889163 * (-0.148784 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0132134 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.228939 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0529093 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.185947 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0333771 * (0.115603 * x_1 - 0.559943 * x_2) - 0.0496193 * (0.178242 * x_1 + 0.147661 * x_2) - 0.337578 * (0.189229 * x_1 - 0.212302 *$$

$$x_2) - 0.206439 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.191168 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0040081 * (-0.141529 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.00379008 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.050696 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.107278 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.164167 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.112903 * (0.115603 * x_1 - 0.559943 * x_2) - 0.28331 * (0.178242 * x_1 + 0.147661 * x_2) + 0.251034 * (0.189229 * x_1 - 0.212302 * x_2) - 0.0055419 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.221779 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.00180469 * (-0.102796 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.245426 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.0271749 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.149373 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.20195 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.187741 * (0.115603 * x_1 - 0.559943 * x_2) + 0.18074 * (0.178242 * x_1 + 0.147661 * x_2) - 0.0441101 * (0.189229 * x_1 - 0.212302 * x_2) - 0.237967 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.216235 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0362855 * (-0.0391913 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.123921 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.17003 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0757615 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.201078 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.252316 * (0.115603 * x_1 - 0.559943 * x_2) - 0.326691 * (0.178242 * x_1 + 0.147661 * x_2) + 0.00698547 * (0.189229 * x_1 - 0.212302 * x_2) - 0.00301863 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.194609 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0243201 * (-0.0107183 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.0956865 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.162321 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.108168 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.0291888 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.100121 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0900326 * (0.178242 * x_1 + 0.147661 * x_2) - 0.0554088 * (0.189229 * x_1 - 0.212302 * x_2) - 0.3618 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0837 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0188589 * (0.141513 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0752214 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.00765232 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.0884482 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.256175 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.170602 * (0.115603 * x_1 - 0.559943 * x_2) - 0.166361 * (0.178242 * x_1 + 0.147661 * x_2) + 0.322339 * (0.189229 * x_1 - 0.212302 * x_2) - 0.197938 * (0.228434 * x_1 - 0.0207164 *$$

$$x_2) + 0.0139234 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0870816 * (0.159732 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.107909 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.306389 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.0325058 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.30328 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.276108 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0709101 * (0.178242 * x_1 + 0.147661 * x_2) + 0.132233 * (0.189229 * x_1 - 0.212302 * x_2) - 0.410339 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.112674 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0274396 * (0.296381 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0594621 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.176919 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0108246 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.193618 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0796573 * (0.115603 * x_1 - 0.559943 * x_2) + 0.246453 * (0.178242 * x_1 + 0.147661 * x_2) + 0.179855 * (0.189229 * x_1 - 0.212302 * x_2) + 0.277703 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0339893 * (0.327156 * x_1 - 0.407098 * x_2)))) + ((0.0271708 * (-0.173523 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.0572846 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.0733083 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0905027 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.267289 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0372416 * (0.115603 * x_1 - 0.559943 * x_2) - 0.175913 * (0.178242 * x_1 + 0.147661 * x_2) - 0.306655 * (0.189229 * x_1 - 0.212302 * x_2) + 0.310249 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0936387 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.01116 * (-0.157337 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.150808 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.206556 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.176699 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.125142 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.00291987 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0495719 * (0.178242 * x_1 + 0.147661 * x_2) - 0.331928 * (0.189229 * x_1 - 0.212302 * x_2) - 0.445404 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.0962845 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0406836 * (-0.148784 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0132134 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.228939 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0529093 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.185947 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0333771 * (0.115603 * x_1 - 0.559943 * x_2) - 0.0496193 * (0.178242 * x_1 + 0.147661 * x_2) - 0.337578 * (0.189229 * x_1 - 0.212302 * x_2) - 0.206439 * (0.228434 * x_1 - 0.0207164 *$$

$$x_2) - 0.191168 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0275022 * (-0.141529 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.00379008 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.050696 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.107278 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.164167 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.112903 * (0.115603 * x_1 - 0.559943 * x_2) - 0.28331 * (0.178242 * x_1 + 0.147661 * x_2) + 0.251034 * (0.189229 * x_1 - 0.212302 * x_2) - 0.0055419 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.221779 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0816945 * (-0.102796 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.245426 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.0271749 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.149373 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.20195 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.187741 * (0.115603 * x_1 - 0.559943 * x_2) + 0.18074 * (0.178242 * x_1 + 0.147661 * x_2) - 0.0441101 * (0.189229 * x_1 - 0.212302 * x_2) - 0.237967 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.216235 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0405768 * (-0.0391913 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.123921 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.17003 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0757615 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.201078 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.252316 * (0.115603 * x_1 - 0.559943 * x_2) - 0.326691 * (0.178242 * x_1 + 0.147661 * x_2) + 0.00698547 * (0.189229 * x_1 - 0.212302 * x_2) - 0.00301863 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.194609 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0320322 * (-0.0107183 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.0956865 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.162321 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.108168 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.0291888 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.100121 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0900326 * (0.178242 * x_1 + 0.147661 * x_2) - 0.0554088 * (0.189229 * x_1 - 0.212302 * x_2) - 0.3618 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0837 * (0.327156 * x_1 - 0.407098 * x_2)) - 0.0433391 * (0.141513 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0752214 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.00765232 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.0884482 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.256175 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.170602 * (0.115603 * x_1 - 0.559943 * x_2) - 0.166361 * (0.178242 * x_1 + 0.147661 * x_2) + 0.322339 * (0.189229 * x_1 - 0.212302 * x_2) - 0.197938 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0139234 * (0.327156 * x_1 - 0.407098 *$$

$$x_2)) - 0.0739191 * (0.159732 * (-0.293793 * x_1 + 0.62935 * x_2) + 0.107909 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.306389 * (-0.163165 * x_1 - 0.536536 * x_2) - 0.0325058 * (0.0589481 * x_1 + 0.0207023 * x_2) - 0.30328 * (0.0736858 * x_1 + 0.156368 * x_2) + 0.276108 * (0.115603 * x_1 - 0.559943 * x_2) + 0.0709101 * (0.178242 * x_1 + 0.147661 * x_2) + 0.132233 * (0.189229 * x_1 - 0.212302 * x_2) - 0.410339 * (0.228434 * x_1 - 0.0207164 * x_2) - 0.112674 * (0.327156 * x_1 - 0.407098 * x_2)) + 0.0555686 * (0.296381 * (-0.293793 * x_1 + 0.62935 * x_2) - 0.0594621 * (-0.261837 * x_1 + 0.474018 * x_2) + 0.176919 * (-0.163165 * x_1 - 0.536536 * x_2) + 0.0108246 * (0.0589481 * x_1 + 0.0207023 * x_2) + 0.193618 * (0.0736858 * x_1 + 0.156368 * x_2) - 0.0796573 * (0.115603 * x_1 - 0.559943 * x_2) + 0.246453 * (0.178242 * x_1 + 0.147661 * x_2) + 0.179855 * (0.189229 * x_1 - 0.212302 * x_2) + 0.277703 * (0.228434 * x_1 - 0.0207164 * x_2) + 0.0339893 * (0.327156 * x_1 - 0.407098 * x_2))) * ((-4.65223 * x_1 - 4.29965 * x_2 + 0.73575 * sin(x_1))/0.037499999999999999)))$$