Passive Fault-Tolerant Augmented Neural Lyapunov Control

A method to synthesise control functions for marine vehicles affected by actuators faults

Grande, Davide; Peruffo, Andrea; Salavasidis, Georgios; Anderlini, Enrico; Fenucci, Davide; Phillips, Alexander B.; Kosmatopoulos, Elias B.; Thomas, Giles

DOI
10.1016/j.conengprac.2024.105935

Publication date
2024

Document Version
Final published version

Published in
Control Engineering Practice

Citation (APA)
Grande, D., Peruffo, A., Salavasidis, G., Anderlini, E., Fenucci, D., Phillips, A. B., Kosmatopoulos, E. B., & Thomas, G. (2024). Passive Fault-Tolerant Augmented Neural Lyapunov Control: A method to synthesise control functions for marine vehicles affected by actuators faults. *Control Engineering Practice*, *148*, Article 105935. https://doi.org/10.1016/j.conengprac.2024.105935

Control Engineering Practice 148 (2024) 105935

Contents lists available at ScienceDirect

Control Engineering Practice

journal homepage: www.elsevier.com/locate/conengprac

Passive Fault-Tolerant Augmented Neural Lyapunov Control: A method to synthesise control functions for marine vehicles affected by actuators faults ☆

Davide Grande [a,b,*], Andrea Peruffo [c], Georgios Salavasidis [b], Enrico Anderlini [a], Davide Fenucci [b], Alexander B. Phillips [b], Elias B. Kosmatopoulos [d], Giles Thomas [a]

[a] University College London, Department of Mechanical Engineering, Gower St, WC1E 6BT London, UK
[b] National Oceanography Centre, SO14 3ZH Southampton, UK
[c] Delft University of Technology, Mekelweg 5, 2628 CD Delft, Netherlands
[d] Democritus University of Thrace & CERTH, Xanthi, 67100, Greece

ARTICLE INFO

Keywords:
Passive Fault-Tolerant Control
Lyapunov methods
Computer-aided control synthesis
Formal verification
Neural networks
Machine-learning

ABSTRACT

Closed-loop stability of control systems can be undermined by actuator faults. Redundant actuator sets and Fault-Tolerant Control (FTC) strategies can be exploited to enhance system resiliency to loss of actuator efficiency, complete failures or jamming. Passive FTC methods entail designing a fixed-gain control law that can preserve the stability of the closed-loop system when faults occur, by compromising on the performance of the faultless system. The use of Passive FTC methods is of particular interest in the case of underwater autonomous platforms, where the use of extensive sensoring to monitor the status of the actuator is limited by strict space and energy constraints. In this work, a machine learning-based method is formulated to systematically synthesise control laws for systems affected by actuator faults, encompassing *partial* and *total* loss of actuator efficiency and control surfaces jamming. Differently from other methods in this category, the closed-loop stability is formally certified. The learning architecture encompasses two Artificial Neural Networks, one representing the control law, and the other resembling a Control Lyapunov Function (CLF). Periodically, a Satisfiability Modulo Theory solver is employed to verify that the synthesised CLF formally satisfies the theoretical Lyapunov conditions associated to both the nominal and faulty dynamics. The method is applied to three marine test cases: first, an Autonomous Underwater Vehicle performing planar motion and subjected to full loss of actuator efficiency is investigated. Next, a study is conducted on a hybrid Underwater Glider with a pair of independent twin stern planes jamming at a fixed position. Finally, partial loss of effectiveness is considered. In all three scenarios, the system is able to synthesise stabilising control laws with performance degradation prescribed by the user. Unlike other machine-learning based techniques, this method offers formal stability certificates and relies on limited computational resources rendering it possible to be run on unassuming office laptops. An open-source software tool is developed and released at: https://github.com/grande-dev/pFT-ANLC.

1. Introduction

Faults are defined as an undesired abrupt change in the dynamics of a signal of a sensor or of an actuator (Willsky, 1976). Fault-Tolerant Control (FTC) aims at preserving the plant operation and closed-loop stability when faults occur (Patan, 2019). During Autonomous Underwater Vehicles (AUVs) and Underwater Gliders (UGs) deployments, a large variety of unplanned and undesired conditions can occur and jeopardise the mission success. In the case of AUVs, typically employing thrusters as the main source of actuation, malfunctions are often identified as thruster obstruction, thruster flooding and rotor failure (Caccia et al., 2001). External solid objects such as ice, seaweed or other marine detrita entering between the blades, the propeller and the enclosing, can also either damage the propeller blades or significantly increase the torque required to spin the motor shaft. Further, water

☆ This work was supported by the National Oceanography Centre, Southampton (UK), by the University College London, London (UK), and by the European Research Council through the SENTIENT project (ERC-2017-STG #755953).

* Corresponding author at: University College London, Department of Mechanical Engineering, Gower St, WC1E 6BT London, UK.
E-mail addresses: ucemdg0@ucl.ac.uk, grande.rdev@gmail.com (D. Grande), a.peruffo@tudelft.nl (A. Peruffo), georgios.salavasidis@noc.ac.uk (G. Salavasidis), e.anderlini@ucl.ac.uk (E. Anderlini), davfen@noc.ac.uk (D. Fenucci), abp@noc.ac.uk (A.B. Phillips), kosmatop@iti.gr (E.B. Kosmatopoulos), giles.thomas@ucl.ac.uk (G. Thomas).

https://doi.org/10.1016/j.conengprac.2024.105935
Received 1 November 2023; Received in revised form 29 March 2024; Accepted 1 April 2024
Available online 24 April 2024

ingress inside sealed electronic compartments can modify the internal electrical connections, leading to shorts or current dispersion. In the case of UGs, the long nature of their deployment, lasting up to several months, renders these platforms particularly subjected to unpredictable conditions, potentially leading to faults. UGs are oftentimes deployed in challenging environments, such as when conducting missions under the ice caps (Webster et al., 2015). Several factors can affect the successful vehicle reentry when deployed in such demanding operating domains. Brito et al. (2014) provides a survey of 58 gliders operating for a period of 4 years, with glider faults reported as mechanical, logistical and environmental causes (Queste et al., 2012). Diverse causes of faults are identified, ranging from logistic-driven faults, conventionally driven by human errors as incorrect ballasting and trimming conditions, to mechanical ones, as incorrect design, leaks and malfunctioning components. While most of the listed faults can be mitigated with sufficient planning and preliminary mission tests, environment-linked failures are usually more severe and intrinsically difficult to forecast due to their abrupt and chaotic nature. Environmental disruptions comprise getting tangled in drifting fishing nets, colliding with other vessels during the communication phase on surface (Queste et al., 2012) or getting attacked by large ocean predators, as white sharks (Stanway et al., 2015). Additionally, UGs deployed in shallow tropical waters nearby the equator are particularly prone to biofouling events (Anderlini et al., 2020). The growth of marine organisms such as barnacles over the control surfaces can lead to a loss of drag to lift ratio performance or to jamming, in the worst case scenarios. All the aforementioned events can affect the correct functionality of onboard actuators for both AUVs and UGs and require devising FTC schemes to prevent the catastrophic loss of the vehicle.

FTC methods are conventionally split in *Active* and *Passive* techniques. Active FTC (AFTC) methods exploit Fault Detection and Isolation diagnosis systems (Boem et al., 2019) and cover a plethora of diverse control architectures. AFTC architectures encompass various strategies, such as switching between a set of pre-computed control laws, online retuning of control gain, or even a complete redesign of the controller structure, among other approaches. Active methods can generally ensure satisfactory control performance following a fault, but are usually computationally expensive, depend on precise model information and suffer from a period of delay associated to the estimation of the fault location and severity. In opposition, *Passive* FTC (PFTC) architectures entail designing a unique set of static gain that guarantees stability in both nominal and faulty scenarios (Zhang & Jiang, 2008). With respect to AFTC, PFTC techniques results in more conservative control performance in the nominal (faultless) scenario, whilst being easier to design and needing lower computational requirements. Choosing between AFTC and PFTC methods depends on the application of interest and, in turn, on the tradeoff between stability requirements and control performance (Verhaegen et al., 2010).

Underwater platforms, such as AUVs and UGs, represent a category of exceptionally expensive vehicles operating in highly uncertain conditions, where reliability and safety are often preferred to higher performance. In the specific case of UGs, where energy saving determines the possibility 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 oftentimes entirely turned off and the whole control systems module is only intermittently switched on to perform periodic corrective actions (Graver, 2005). More broadly, in every field where the widespread sensoring and algorithms use cannot be assumed due to cost, power or complexity constraints, PFTC represents the option with the most significant potential impact. Thence, the PFTC class of controllers is the focus of this study.

Robust Control (RC) techniques are oftentimes employed to design PFTCs, by minimising, for instance, the $\mathscr{H}_2$ or $\mathscr{H}_\infty$-norms between exogenous inputs and desired performances (Blanke et al., 2006). Nonlinear techniques can also be employed via leveraging the Lyapunov theory to extend the nominal control law with additional terms compensating for partial loss of actuator efficiency (Benosman & Lum, 2009). Among several options, in the underwater domain, RCs represent the most widespread fault-tolerant class of technique (Katebi & Grimble, 1999). A limiting factor of employing RCs is represented by the necessity to define sensible operating points for linearisation, e.g. in Kaminer et al. (1991), as the underwater domain is characterised by highly nonlinear and coupled dynamics.

In recent times, machine-learning methods employing Artificial Neural Network (ANN) were employed in (faultless) AUV control applications (Anderlini et al., 2019; Carlucho et al., 2018; Thanh & Anh, 2022). ANN-based methods can also be adapted to design PFTC systems (Dooraki & Lee, 2020). In the general formulation, however, the rigorous stability of machine-learning-based controllers over a continuous domain is not certified, as the controllers are synthesised based on a finite training set. Hence, recently, a new trend focuses on neural controllers equipped with a formal proof of stability, based on *Satisfiability Modulo Theories* (SMT)-solving. SMTs are automated reasoning tools devised to deal with the problem of deciding whether a mathematical formula is *satisfiable* over a bounded domain of real numbers. SMTs can be used to verify if a function abides prescribed properties, e.g. if it is positive definite. The intersection between formal verification techniques and machine-learning is thus an interesting field where Lyapunov-based techniques find applications, e.g. Abate et al. (2021, 2020), Ahmed et al. (2020), Edwards et al. (2023). One such architecture relying on SMT-solving is the *Augmented Neural Lyapunov Control* (ANLC): two ANNs are employed, one representing a control law, and a second one embodying a Control Lyapunov Function (CLF) (Chang et al., 2019; Grande, Peruffo, Anderlini, & Salavasidis, 2023). This method exploits a loop between two main modules, a *Learner* and a *Falsifier*, with a third supporting module, referred to as *Translator*, employed as interface between the former two. The Learner is tasked with training the two ANNs starting from a finite small set of initial samples (sparse), that grows in size based on a CounterExample-Guided Inductive Synthesis (CEGIS) (Solar-Lezama et al., 2006). The Falsifier instead, formally verifies that the ANN represents a CLF for the considered dynamics by taking as input the symbolic expression of the CLF provided by the Translator, and verifies the CLF over a domain of real numbers (dense). One typical choice of the SMT solver is represented by *dReal*, due to its capability to handle nonlinear expressions (Gao et al., 2013). The ANLC was shown capable to synthesise controllers for nonlinear and unstable dynamics within minimal computational requirements, showing attractive potential to mix the learning ability of ANNs with formal certificate of closed-loop stability. The interested reader may find a recent survey on neural-based Lyapunov techniques in Dawson et al. (2023).

Contributions. This work is built upon the recent (Grande, Fenucci, et al., 2023), as it exploits the CEGIS loop to synthesise Lyapunov functions. Hereby, an extension of the method is devised, allowing the design of a controller for a larger variety of faults: from purely binary faults in Grande, Fenucci, et al. (2023), the updated procedure is now able to provide a controller for the entire range of loss of effectiveness, alongside considering jamming events. The method entails the automatic synthesis of linear or nonlinear control functions with static gain that can be computed offline and deployed in real-time closed-loop systems. To the best of the authors knowledge, this work represents the first effort providing neural-based, passive fault-tolerant control design, that deals with the entire range of actuator's loss of effectiveness. To showcase the reliability and strength of the procedure, three realistic benchmarks are presented, indicative of typical faults on diverse underwater platforms. As a further step towards devising control laws practically usable in the field, a strategy to account for actuator saturation is proposed, whilst still certifying the closed-loop stability. The newly devised technique is compared against a robust

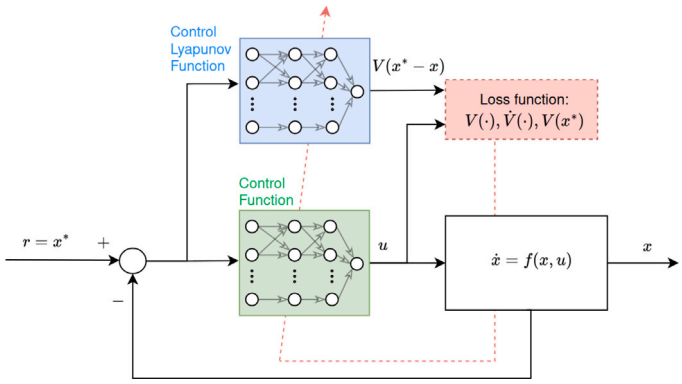


Fig. 1. Augmented Neural Lyapunov Control architecture with Lyapunov function ANN (blue box), control function ANN (green box) and loss function backpropagation (red dashed line). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$\mathscr{H}_\infty$ approach: the latter results in more energy-expensive control effort and it does not satisfy saturation constraints. Additionally, given the Lyapunov-based nature of the proposed method, key properties of the closed-loop system such as the Region Of Attraction can not only be investigated, but also tuned via dedicated gain parameters. The proposed method allows for an intuitive understanding of the tolerated performance degradation, defined as a maximum deviation from the desired equilibrium. Finally, the algorithmic logic is illustrated and a software tool is released open-source https://github.com/grande-dev/pFT-ANLC.

2. Control Lyapunov function synthesis via CEGIS

The aim of this work is the design of passive fault-tolerant control laws for a nonlinear dynamic system

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \phi_i), \tag{1}$$

where $\boldsymbol{x} \in \mathscr{D} \subseteq \mathbb{R}^n$ is the system's state, $\boldsymbol{u} \in \mathscr{U} \subseteq \mathbb{R}^m$ is the control input, and ϕ_i denotes possible system faults. It is assumed that each one of the p 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, whilst $\boldsymbol{f}_{\phi_j}(\boldsymbol{x}, \boldsymbol{u})$ denotes the dynamics characterised by the fault of the jth actuator (with $\phi_j \in \boldsymbol{\Phi}$). Aim of this work is the design of a control law that drives the system to an equilibrium $\boldsymbol{x}^\star \in \mathscr{D}$. Without loss of generality, in the following it is assumed that the system exhibits an equilibrium $\boldsymbol{x}^\star = 0$. Along with the design of a control law, a certificate of the closed-loop stability via a CLF is provided. The synthesis of a CLF is built upon (Grande, Peruffo, Anderlini, & Salavasidis, 2023) and it employs a neural control framework, depicted in Fig. 1, in which two ANNs represent the CLF and the control law, respectively. The control network can devise both linear and nonlinear control laws, based on the user preference. Clearly, the use of nonlinear control laws improves the learning capability of the framework, at the expense of the computational complexity, as it increases both the burden of the training algorithms, e.g. Stochastic Gradient Descent (SGD), and of the verification step (carried out by the SMT solver).

2.1. Method overview

The CEGIS paradigm evolves based on the information flow between three modules, the *Learner*, the *Translator* and the *Falsifier*, as outlined in Fig. 2. The first module trains the CLF and the control gains by iteratively minimising a loss function, expression of the three theoretical Lyapunov conditions, as follows.

Definition 1 (*Control Lyapunov Function Tedrake, 2023*)**.** Given a domain $\mathscr{D}$ and a system model $f : \mathscr{D} \times \mathscr{U} \to \mathscr{D}$ with unique equilibrium point $\boldsymbol{x}^\star \in \mathscr{D}$, such that $f(\boldsymbol{x}^\star, \boldsymbol{u}) = 0$; consider a function $V : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}, V \in C^1$. V is a (Control) Lyapunov Function if there exists $\boldsymbol{u}$ such that

$$V(\boldsymbol{x}^\star) = 0, \tag{2a}$$

$$V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\}, \tag{2b}$$

$$\dot{V}(\boldsymbol{x}, \boldsymbol{u}) = \langle \boldsymbol{\nabla}(\boldsymbol{V})_{\boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \rangle < 0, \quad \exists \boldsymbol{u}\ \forall \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\}, \tag{2c}$$

where $\langle a, b \rangle$ denotes the inner product of the terms a and b. The Learner, via the training of the neural networks, shall ensure that the three conditions (2) are satisfied over a finite dataset. Once (2) is valid over the finite size dataset, the training is halted and the ANNs are passed to the Translator, which symbolically evaluates the candidate controller (u_c) and candidate CLF (V_c). The resulting expressions are then transmitted to the Falsifier, where the Lyapunov stability conditions are formally evaluated over a bounded domain of *real numbers*. Following, either the candidate CLF is verified to be valid and the procedure terminates, or a set of points violating the Lyapunov conditions are generated. Such instances of points violating (2) are called *counterexamples* (CEs). If any CE is found, it is added to the dataset, which is fed back to the Learner, restarting the training procedure with additional information.

2.2. Learner

Given a nominal dynamical system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ and a target equilibrium $\boldsymbol{x}^\star$, the training procedure starts from a (small) initial sample set S composed of randomly selected states ($\boldsymbol{s_i}$) generated within a domain $\mathscr{D}$ (containing $\boldsymbol{x}^\star$). At each learning iteration, a cost function is evaluated and the ANN parameters (η) are updated according to the SGD algorithm. A detailed description of the cost functions is outlined in the following Section 3.

At the end of the training, this procedure returns a candidate control law and a candidate CLF that satisfy the Lyapunov conditions (2) over the *finite* sample set, 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 $\boldsymbol{s_i}$.

2.3. 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 *Translator* (Abate et al., 2021), hereby derived in the most general formulation for a feedforward ANN encompassing bias. First, the output of a feedforward ANN layer i is recalled to be:

$$\boldsymbol{z_i} = \sigma_i(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i}),\ i = 1, \dots, k \tag{3}$$

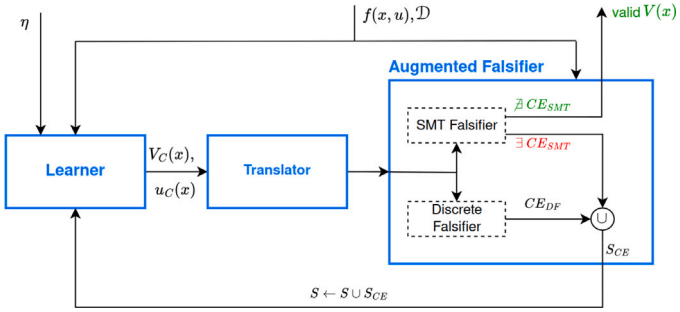


Fig. 2. Learner-Falsifier CEGIS loop: the Learner trains the ANNs, the Translator transforms the ANN equations into symbolic expressions while the Falsifier verifies the formal validity of the CLF (V) (Grande, Peruffo, Anderlini, & Salavasidis, 2023).

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

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

$$V(x) = \sigma_k(\boldsymbol{W_k}\ \boldsymbol{z_{k-1}} + \boldsymbol{B_k}). \tag{4}$$

Next, to symbolically evaluate the Lie derivative, the formal definition is recalled first: $\dot{V} = \langle \nabla(\boldsymbol{V})_{\boldsymbol{x}}, \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\rangle$, with $\nabla(\boldsymbol{V})_{\boldsymbol{x}} := \frac{\partial V(x)}{\partial \boldsymbol{x}} = [\frac{\partial V}{\partial x_1} \cdots \frac{\partial V}{\partial x_n}]^T$.

This gradient can be evaluated as the following chain rule:

$$\frac{\partial V}{\partial \boldsymbol{x}} = \frac{\partial z_i}{\partial \boldsymbol{z_{i-1}}} \frac{\partial \boldsymbol{z_{i-1}}}{\partial \boldsymbol{z_{i-2}}} \cdots \frac{\partial \boldsymbol{z_1}}{\partial \boldsymbol{z_0}} \tag{5}$$

with $\boldsymbol{z_0} = \boldsymbol{x}$ and $z_i = V$. The partial derivative of the output of a generic layer i with respect to the output of the previous layer can be evaluated by means of (3) as:

$$\frac{\partial z_i}{\partial \boldsymbol{z_{i-1}}} = \frac{\partial \sigma(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial \boldsymbol{z_{i-1}}} = \frac{\partial \sigma(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial (\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})} \frac{\partial (\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial \boldsymbol{z_{i-1}}}. \tag{6}$$

It is thus possible to compute the first factor of Eq. (6) as:

$$\frac{\partial \sigma(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial (\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})} = \mathrm{diag}[\sigma'(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})], \tag{7}$$

where $\mathrm{diag}[\boldsymbol{a}]$ represents a diagonal matrix whose entries are the elements of vector $[\boldsymbol{a}]$ and with h_i denoting the number of neurons of the ith layer, such that $\boldsymbol{W_i} \in \mathbb{R}^{h_i \times h_{i-1}}$, $\boldsymbol{B_i} \in \mathbb{R}^{h_i}$ and $\boldsymbol{z_i} \in \mathbb{R}^{h_i}$. Next, the second factor of Eq. (6) can be calculated as:

$$\frac{\partial (\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial \boldsymbol{z_{i-1}}} = \boldsymbol{W_i}. \tag{8}$$

Overall, Eq. (6) can be expressed as:

$$\frac{\partial \sigma(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})}{\partial \boldsymbol{z_{i-1}}} = \mathrm{diag}[\sigma'(\boldsymbol{W_i}\ \boldsymbol{z_{i-1}} + \boldsymbol{B_i})]\ \boldsymbol{W_i}. \tag{9}$$

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

$$\dot{V} = \left(\prod_{i=1}^{k} \mathrm{diag}[\sigma'_{k-i+1}(\boldsymbol{W_{k-i+1}}\ \boldsymbol{z_{k-i}} + \boldsymbol{b_{k-i+1}})]\ \boldsymbol{W_{k-i+1}} \right) \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \tag{10}$$

where $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$ embeds the symbolic expression of $\boldsymbol{u}(\boldsymbol{x})$, in turn obtained via Eq. (3).

2.4. Falsifier

Given a candidate CLF $V_c(\boldsymbol{x})$ and its corresponding Lie derivative $\dot{V}_c(\boldsymbol{x},\boldsymbol{u}_c)$, the Falsifier ought to prove that conditions (2) are satisifed over the entire dense domain $\mathscr{D}$ via SMT solving. As verifying complex nonlinear expressions, such as the ones stemming from the computation of the Lie derivative, represent the computational bottleneck of this procedure, the number of callbacks to the SMT solvers should be minimised. To this end, a *Discrete Falsifier* (DF) module is employed as support, tasked with numerically evaluating the correctness of (2) over a grid of prescribed precision (Grande, Peruffo, Anderlini, & Salavasidis, 2023). The DF device slims the overall computational burden by reducing the number of callbacks to the SMT Falsifier, that gets invoked only when no CE is found by the DF. Additionally, the DF allows to return several CEs at each callback, speeding up the overall learning capability of the framework while limiting issues linked to dataset overfitting.

Upon invoking the SMT Falsifier, the proposed procedure searches for an instance where conditions (2) do *not* hold. Formally, the Falsifier seeks for an occurrence of $\boldsymbol{x}$:

$$\exists \boldsymbol{x} : \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\} \implies \Big(V(\boldsymbol{x}) \leq 0 \ \vee \dot{V}_n(\boldsymbol{x},\boldsymbol{u}) \geq 0\Big). \tag{11}$$

As previously mentioned, such an occurrence $\boldsymbol{x}$ violates the Lyapunov conditions and is defined as a CE. If the Falsifier finds no CEs, the candidate CLF is indeed valid over $\mathscr{D}$, and the overall procedure terminates. Alternatively, it provides a CE point, where either Lyapunov condition is invalidated. This point is then added to the training dataset, and the learning restarts. Notice that condition $V(\boldsymbol{x}^\star) = 0$ is omitted from the Falsifier constraints (11): this condition is verified separately, since it simply represents a pointwise evaluation.

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*, thus spurious CEs might be returned in the neighbourhood of the origin (within a precision δ). Therefore, a small neighbourhood of 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 $\|\boldsymbol{x}\|_2 \leq \epsilon$ (Gao et al., 2019) (with $\|\boldsymbol{a}\|_2 = \sqrt{a_1^2 + \cdots + a_n^2}$ denoting the 2-norm of vector $\boldsymbol{a}$). The latter property is of great importance in real world applications, as even bounding a dynamics to oscillate sufficiently near the target equilibrium is a desired outcome of a control system (La Salle & Lefschetz, 1961). Especially within the fault-tolerant framework, this feature, rather than being a limitation, represents a useful additional tuning parameter. In 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-stability bound. The validity domain of the CLF is thus defined as: $\boldsymbol{x} \in \mathscr{D} : \epsilon \leq \|\boldsymbol{x}\|_2 \leq \overline{\gamma}$, where $\epsilon, \overline{\gamma}$ 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 tuned during the control system design.

3. Synthesis of passive fault-tolerant control laws

The introductory ANLC method presented in Grande, Peruffo, Anderlini, and Salavasidis (2023) proposes the synthesis of controllers for nominal (faultless) dynamics. In this section, the ANLC method is extended by devising tailored modification to both the *Learner* and

the *Falsifier*, to guarantee resilient fault-tolerant properties. A crucial assumption should be stated first.

Assumption 1. In this work, at most only one fault is assumed to be present at each time. A situation where multiple faults materialise at the same time is oftentimes 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.

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 negative definite. Hereby, the idea is illustrated in the case of complete loss of effectiveness, and, in Section 8, the framework is generalised to the case when partial losses of actuator effectiveness are accounted for. It is assumed that each one of the p actuators can be affected by complete faults, hence a total of $(p+1)$ dynamics (i.e. models with a fault on the jth actuator in addition to the nominal model) can be used to describe the nominal and faulty scenarios. To guarantee that $\boldsymbol{x}^\star$ is stable for all the $(p+1)$ dynamics, the corresponding $(p+1)$ Lie derivatives need to be negative definite; formally:

$$(\dot{V}_n(\boldsymbol{x},\boldsymbol{u}) < 0) \wedge (\forall j \in \boldsymbol{\Phi} : \{\dot{V}_{\phi_j}(\boldsymbol{x},\boldsymbol{u}) < 0\}) \tag{12}$$

where $\dot{V}_i(\boldsymbol{x},\boldsymbol{u}) = \langle \boldsymbol{\nabla V}(\boldsymbol{x}), \boldsymbol{f}_i(\boldsymbol{x},\boldsymbol{u})\rangle$, for $i = n$ (nominal dynamics) or $i \in \boldsymbol{\Phi}$, where $\boldsymbol{\Phi}$ represents the set of faults.

The training is set up in order to synthesise a CLF abiding (12).

3.1. Fault-tolerant learner

As proposed in Grande, Peruffo, Anderlini, and Salavasidis (2023), two loss functions are employed in this work. The first one, denoted as *Strict Lyapunov Risk Loss* ($\mathrm{L_{SLR}}$), is employed to detect when, over every sample ($\boldsymbol{s_i}$) within the training set S, the theoretical Lyapunov conditions are verified, namely occurring when:

$$\forall \boldsymbol{s_i} \in S : \{V(\boldsymbol{s_i}) > 0, \dot{V}(\boldsymbol{s_i},\boldsymbol{u_i}) < 0\}. \tag{13}$$

The $\mathrm{L_{SLR}}$, is therefore defined as:

$$L_{SLR} = \sum_{i=1}^{N} \mathscr{R}(-V(\boldsymbol{s_i})) + \sum_{i=1}^{N} \mathscr{R}(\dot{V}(\boldsymbol{s_i},\boldsymbol{u_i})) + V(0)^2 \tag{14}$$

where $\mathscr{R}(a) = ReLU(a) = \max(0,a)$ for a generic input a and N 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 (2) 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 the Region Of Attraction (ROA) of the closed-loop system around $\boldsymbol{x}^\star$. To this end, an improved loss function is defined, referred to as *Empirical Lyapunov Risk Loss* (L_{ELR}), whilst L_{SLR} is employed as a logical condition to stop the learning procedure.

The Empirical Risk Loss extends the definition of $\mathrm{L_{SLR}}$ by accounting for a fourth term regulating the size of the ROA and CLF shape, and for further properties of the Lie derivative as:

$$\begin{aligned} L_{ELR} = \alpha_1 \sum_{i=1}^{N} \mathscr{R}(-V(\boldsymbol{s_i})) + \alpha_2 L_{\dot{V}} + \alpha_3 V(0)^2 \\ + \alpha_4 \frac{1}{N} \sum_{i=1}^{N} (\|\boldsymbol{s_i}\|_2 - \alpha_{ROA}\ V(\boldsymbol{s_i}))^2, \end{aligned} \tag{15}$$

with $\alpha_1, \ldots, \alpha_4, \alpha_{ROA}$ tuning coefficients that can be selected as discussed in Grande, Peruffo, Anderlini, and Salavasidis (2023). The loss term associated with the Lie derivative ($L_{\dot{V}}$), capturing both nominal and faulty dynamics is:

$$L_{\dot{V}} = \sum_{i=1}^{N} \mathscr{R}(\boldsymbol{\nabla V}(\boldsymbol{s_i}) \cdot \boldsymbol{f_n}(\boldsymbol{s_i},\boldsymbol{u_i}) + \alpha_{\text{off}}) +$$

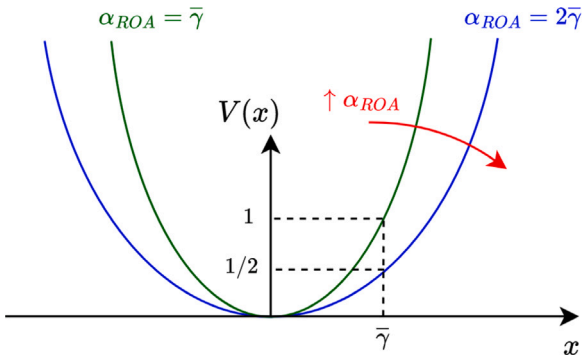


Fig. 3. CLFs with different α_{ROA} tuning, where $\bar{\gamma}$ denotes the upper boundary of the domain $\mathscr{D}$.

$$\sum_{j=1}^{p} \sum_{i=1}^{N} \mathscr{R}(\boldsymbol{\nabla V}(\boldsymbol{s_i}) \cdot \boldsymbol{f}_{\boldsymbol{\phi}_j}(\boldsymbol{s_i},\boldsymbol{u_i}) + \alpha_{\text{off}}) \tag{16}$$

where α_{off} is an additional tuning term used to enforce more negative Lie derivatives (Chang et al., 2019).

This loss function at the same time enforces the positiveness of V, the negativeness of the Lie derivatives $\dot{V}_i$, 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 *approximately* the CLF resembles a paraboloid of revolution, whilst verifying $L_{SLR} = 0$ strictly. The lower α_{ROA} the steeper the CLF, with the effect of tuning α_{ROA} illustrated in Fig. 3.

3.2. Fault-tolerant falsifier

Once the training procedure reaches the end of its operations, the Learner returns 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 $\mathscr{D}$. If the CLF is found not abiding the theoretical Lyapunov conditions, a CE is returned, added to the training set S and the training is restarted. The verification check (11) can be modified to accommodate faulty dynamics as follows:

$$\begin{aligned} \exists \boldsymbol{x} : \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\} \implies \Big(V(\boldsymbol{x}) \leq 0 \ \vee\ \dot{V}_n(\boldsymbol{x},\boldsymbol{u}) \geq 0 \ \vee \\ \{\dot{V}_{\phi_j}(\boldsymbol{x},\boldsymbol{u}) \geq 0\}_{j=1}^{p} \Big), \end{aligned} \tag{17}$$

where the latter term denotes the sequence of the p Lie derivatives associated to the faulty systems. Similarly to the nominal case, if the Falsifier cannot find an instance satisfying (17), the CLF 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 is hereby referred to as *passive Fault-Tolerant Augmented Neural Lyapunov Control* (pFT-ANLC).

4. Software

Following the description of the proposed pFT-ANLC method, in Algorithm 1 the pseudocode underlying the open-source software tool is introduced. The algorithm covers the Learner, Discrete and SMT falsifiers, whilst 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 size $\bar{\gamma}$. 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. The learning steps are iterated until the Lyapunov conditions are verified $\forall \boldsymbol{s_i} \in S$, i.e. when $L_{SLR} = 0$. Once the latter relationship holds, 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 ($\dot{V}^S_{\phi_{i,c}}$) 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. If CEs are found, the latter are added to S and the learning stage is restarted. When no CEs are obtained instead, the *SMT Falsifier* is tasked with verifying the Lyapunov conditions over the dense domain of the Reals. If a CE is obtained via SMT-solving (CE_{SMT}) the latter is added to S and the learning stage restarted. Otherwise, the candidate CLF V^S_c is formally verified to be correct, 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 $\overline{\tau}$ 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 verification of first-order logical formulae with generic nonlinear functions over the theory of nonlinear arithmetic is an undecidable problem. The SMT solver chosen for this study solves a δ-complete falsification constraint stemming into an NP-hard task (Chang et al., 2019). Such a SMT choice guarantees that the problem can be formally solved provided the exclusion of the ϵ-stability bound in the neighbourhood of the origin, as detailed in Section 2.4. As the verification step represents the computational bottleneck of the procedure, a *timeout* is introduced, defining a maximum time threshold for the SMT Falsifier to compute the CEs. If the timeout is exceeded, the training run is halted, flagged as not successful and a new run initialised with a different seed started.

Finally, the use of a *selective sliding window* is recalled (Grande, Peruffo, Anderlini, & Salavasidis, 2023), ensuring the dataset size to remain bounded as the training proceeds and new CEs are generated. Further algorithmic details regarding the use of a learning rate scheduler and about the sampling step of the Discrete Falsifier are discussed in Grande, Peruffo, Anderlini, and Salavasidis (2023).

The software tool aims at synthesising a *unique* set of gain that guarantees closed-loop stability in both the nominal and in the faulty-case scenarios. The control function, 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.

Algorithm 1 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
  until ($L_{SLR} > 0$)
  return $V_\eta(s_i), u_\eta(s_i)$

function DISCRETE FALSIFIER($V^S_c, \dot{V}^S_{n_c}, \dot{V}^S_{\phi_{i,c}}, \mathscr{D}$)
  Discretise $\mathscr{D}$ and numerically evaluate ($V^S_c \leq 0$, $\dot{V}^S_{n_c} \geq 0, \dot{V}^S_{\phi_{i,c}} \geq 0$)
  $CE_{DF} \leftarrow$ Violations points
  return CE_{DF}

function SMT FALSIFIER($V^S_c, \dot{V}^S_{n_c}, \dot{V}^S_{\phi_{i,c}}, \mathscr{D}$)
  Using SMT solver to verify conditions
  return `sat` or CE_{SMT}

function MAIN()
  Input: dynamics (f_n, f_{ϕ_i}), initial dataset (S), Falsifier domain $\mathscr{D}$ ($\epsilon, \overline{\gamma}$), loss function ($\alpha_{(\cdot)}$, $\overline{\tau}$), learning rate, optional initial linear control gains (q^{lqr})
  Initialise ANN size (optional: initialise linear control gains with q^{lqr})
  repeat
    if (size(S) $\geq S_{max}$): Apply sliding window
    $V_\eta(x), u_\eta(x) \leftarrow$ LEARNER(S, f, ANN_η)
    Compute symbolic values: $f^S_{n_c}, f^S_{\phi_{i,c}}, u^S_c, V^S_c, \dot{V}^S_{n_c}, \dot{V}^S_{\phi_{i,c}}$
    if ($L_{SLR} == 0$ and $L_{ELR} \leq \overline{\tau}$) **then**
      $CE_{DF} \leftarrow$ DISCR. FALSIFIER($V^S_c, \dot{V}^S_{n_c}, \dot{V}^S_{\phi_{i,c}}, \mathscr{D}$)
      if CE_{DF} is `None` **then**
        $CE_{SMT} \leftarrow$ SMT FALSIFIER($V^S_c, \dot{V}^S_{n_c}, \dot{V}^S_{\phi_{i,c}}, \mathscr{D}$)
      $S_{CE} \leftarrow (CE_{DF} \cup CE_{SMT})$
      if (not `sat`**):** $S \leftarrow (S \cup S_{CE})$
  until not (*converged* **or** *timeout*)

5. Case study A: Control of an autonomous underwater vehicle

In this section, the proposed method is applied to a case study encompassing the control of an AUV. Differently from the neural-Lyapunov studies (Chang et al., 2019; Grande, Peruffo, Anderlini, & Salavasidis, 2023), the goal is to synthesise a control to stabilise the system around a non-zero equilibrium, that, in this case, coincides with maintaining the AUV at a desired target speed. The AUV actuator configuration is inspired to the hover-capable AUV developed at the National Oceanography Centre,[1] as reported in Fig. 4. Each thruster is capable of generating force both in the positive and negative direction.

For this preliminary study, a two-dimensional dynamics accounting for surge speed (u) and angular velocity around the vertical axis (r), is used to describe the planar motion of the AUV, whilst the sway speed (v) is neglected. Each one of the three thrusters F_i is oriented at an angle α_i with respect to the y-body axis (y_B), considered positive clockwise, and is located at distance l_i from the CoG, with $l_{i,x}$ and $l_{i,y}$ indicating the projections of l_i along the x_b and y_b-axis, respectively. By denoting with m the mass of the vehicle and with J_z the moment of inertia around the vertical axis, the AUV dynamics, characterised by $\boldsymbol{x} = [u, r]^T$ and $\boldsymbol{u} = [F_1, F_2, F_3]^T$, is described as:

$$\begin{cases} \dot{x}_1 = \dfrac{-X_u x_1 - X_{uu} x_1^2 + h_1 F_{1,x} + h_2 F_{2,x} + h_3 F_{3,x}}{m} \\ \dot{x}_2 = \dfrac{-N_r x_2 - N_{rr} x_2^2 + (-F_{1,x} l_{1,y} + F_{1,y} l_{1,x}) h_1}{J_z} + \\ \qquad \dfrac{(-F_{2,x} l_{2,y} + F_{2,y} l_{2,x}) h_2 + (-F_{3,x} l_{3,y} + F_{3,y} l_{3,x}) h_3}{J_z} \end{cases} \tag{18}$$

where $F_{i,x} = F_i \sin(\alpha_i)$ and $F_{i,y} = F_i \cos(\alpha_i)$ represent the projections of F_i along the x_B and y_B-axis, respectively; X_u, X_{uu} denote the linear and quadratic surge drag coefficients, whilst N_r, N_{rr} the linear and quadratic yaw drag coefficients.

The proposed case study investigates the capability to synthesise control laws in scenarios involving progressively more faults. Starting with a possible fault occurring on the first thruster F_1 (aft port) only, faults on F_1 or on F_3 (bow) are considered, and finally, on F_1 or on F_2 (aft starboard) or on F_3. The aim is to maintain the AUV at $\boldsymbol{x}^\star = [0.5, 0.0]$ both in the nominal and faulty scenarios. For this application, a nonlinear control law is employed. The selected ANN architecture is

[1] https://noc.ac.uk/technology/technology-development/marine-autonomous-robotic-systems

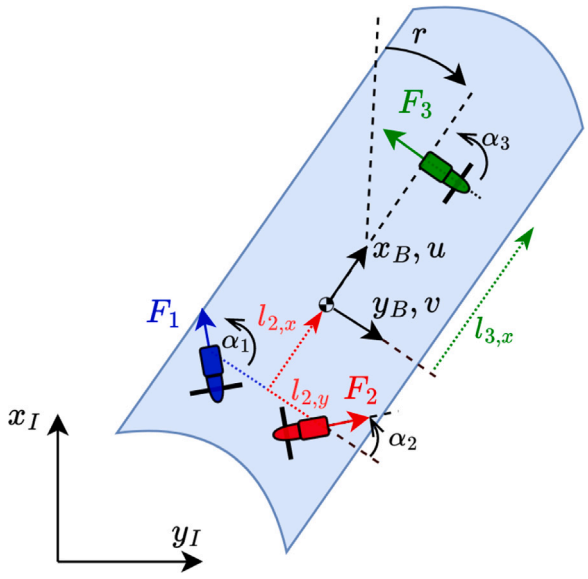


Fig. 4. AUV vehicle model with three (fixed) thrusters moving over the horizontal plane, characterised by surge speed (u), sway speed (v) and angular rate (r).

Table 1
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]

reported in Table 1, where the input, hidden, and output layers' sizes are outlined, along with the presence of the bias and the activation functions.

The training is carried out on an unassuming office laptop without GPU: the machine features an Intel Core i7-8665U CPU with 4 cores running at 1.90 GHz and 16 GB of RAM. During the training, 4 threads are generated and executed over 2 CPU cores, whilst 0.7 GB of RAM is requested. Table 2 reports the results of three simulation campaigns run for different faults 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 (how many 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 on F_1 or on F_2 or on F_3, all the 9 converged tests terminate within a maximum time of 39 [s]. The results reported in Table 2 illustrate how the required computational effort increases as more faults are accounted for. Additionally, fine-tuning the hyperparameters of the control ANN becomes more decisive 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_2) faulty, without significant differences in the success rate values. On the other hand, ANN architectures with limited number of neurons fail to systematically synthesise CLFs when faults can occur on all the three thrusters, highlighting the need to select sufficiently expressive nonlinear control functions as the problem complexity rises. One resulting CLF, obtained after 772 training iterations is reported in Fig. 5.

Finally, closed-loop performance of the resulting fault-tolerant dynamics is illustrated and discussed. Fig. 6 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 failing. The dynamics are initialised at 0.4 [m/s], and are shown converging to the desired ϵ-stability bound (0.5 ± 0.01 [m/s]). When a fault occurs at t=50 [s] on thruster F_1, the surge speeds undergo a drop, that, anyhow, always remain within the desired ϵ-stability threshold. This behaviour is achieved as the pFT-ANLC learns *automatically* to 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 on F_2 with respect to F_1 and a non-zero F_3, as illustrated in Fig. 7. An analogous behaviour is noticed in the angular rate dynamics, reported in Fig. 8, as the controllers learn to converge to an offset value with respect to $x_2^\star$ during nominal operations, to mitigate for possible faults.

6. Case study B: Underwater glider with saturated control

The next 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 (Leonard & Graver, 2001). Gliders follow a saw tooth pattern adopting steady gliding conditions on the ascent and descent. The UGs alternate a positive buoyancy and a nose-up configuration during the climbing phase to negative buoyancy combined with nose-down during dives. For this case study, only the vertical dynamics is considered as the sagittal plane is demonstrated to be invariant: when no initial linear or angular out-of-plane accelerations are provided, the vehicle remains indefinitely on the plane (Graver, 2005). A North-East-Down (NED) frame with origin $\{O_i\}$ is chosen as *inertial reference* since the distances involved in the simulations proposed are of small scale as compared to the Earth radius. Next, a *body-fixed reference* frame of origin $\{O_b\}$ and axes x_b-z_b is fixed at the centre of buoyancy of the glider, which in turn coincides with the centroid of the hull. The orientation of the latter 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. Following, a *flow reference frame* is aligned with the direction of the glider speed (V), 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. 9, is convenient to express forces according to standard hydrodynamic theory (Graver, 2005). In detail, 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 and, finally, inertial forces (accounting for added mass terms) are expressed in the body-fixed reference.

The aim of this study is 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 for the velocity along z_b. The onboard system leverages three actuators, a Variable Buoyancy Device (VBD) and a pair of independent twin stern planes (δ_1 and δ_2). This vehicle concept is inspired by AUVs with redundant movable surfaces designed in "+" or "×" configurations, such as the ones onboard the *Autosub Long Range* series (Phillips et al., 2020) and to hybrid AUVs as the *Sea-Whale 2000* (Huang et al., 2019). These AUV configurations allow to exploit redundant actuators when one of the two stern surfaces fail. Additionally, as per standard gliders design, 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: as neither the pitch angle nor the associated angular rate 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 as $G_s = m_s g$ and $G_p = m_p g$, respectively. The buoyancy force, accounting both for the constant volume of the hull (∇_h) and for the variable volume associated to the VBD, is computed as $B = B(\nabla_h) + B(u_{VBD}) = \rho g \nabla_h +$

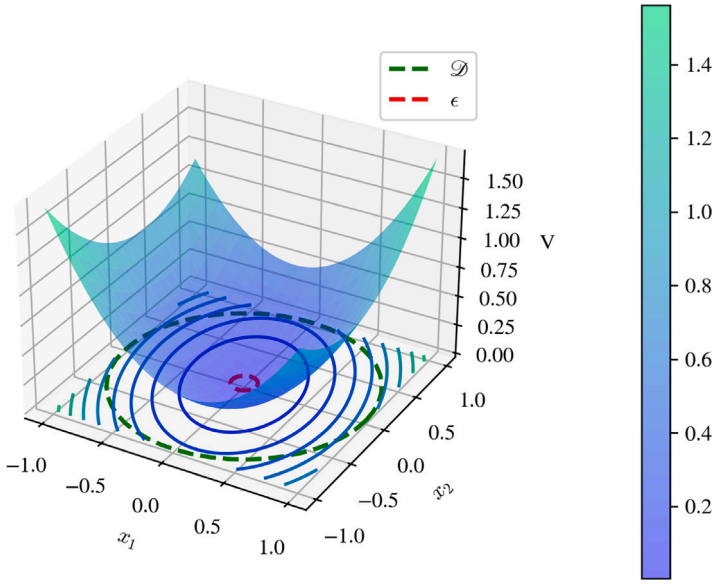


Fig. 5. Synthesised CLF for the AUV system.

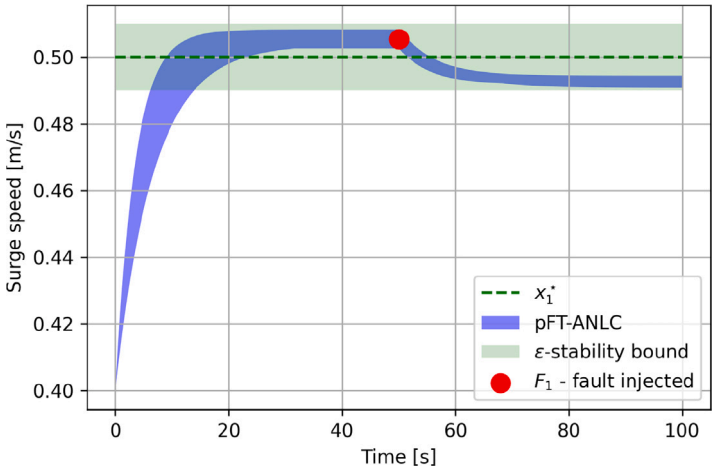


Fig. 6. Closed-loop test AUV system: range of surge dynamics associated to 10 synthesised controllers (blue interval) — fault on F_1 injected at $t = 50$ [s]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

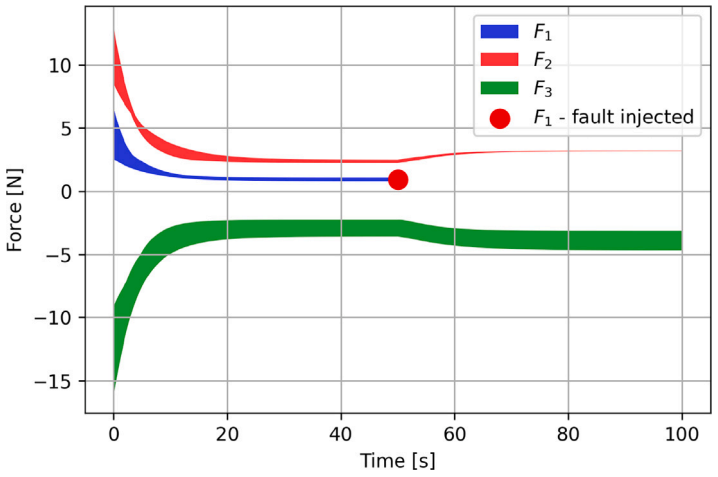


Fig. 7. Closed-loop test AUV system: range of control efforts associated to 10 synthesised controllers (blue, red and green intervals) — fault on F_1 injected at $t = 50$ [s]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
AUV campaign – Synthesis result statistics.

Faulty actuators	Learning 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

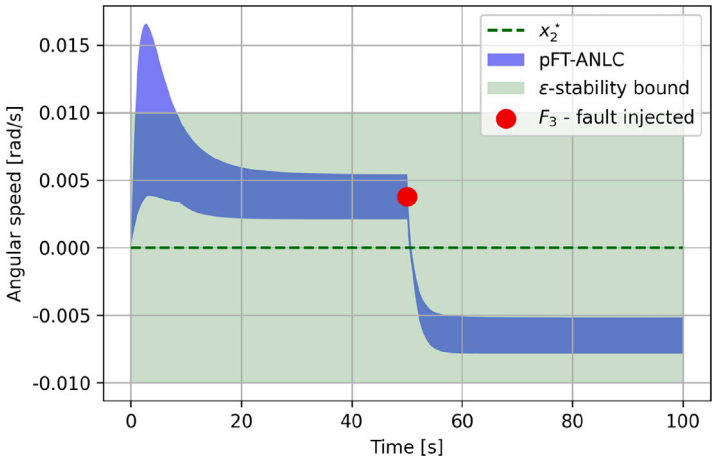


Fig. 8. Closed-loop test AUV system: range of angular rate dynamics associated to 10 synthesised controllers (blue interval) — fault on F_3 injected at $t = 50$ [s]. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

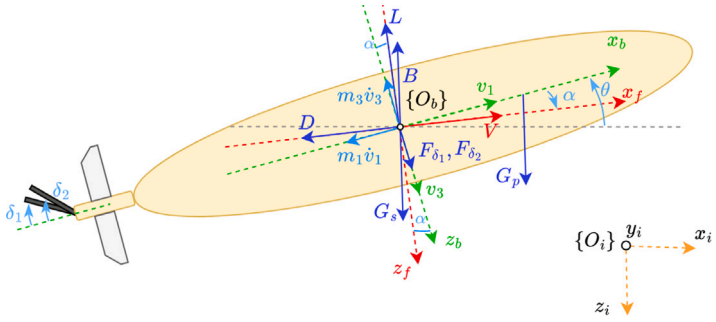


Fig. 9. Underwater Glider dynamics in the sagittal plane with restoring forces ($G_s, G_p, B(\nabla_h)$), hydrodynamic forces (L, D), control forces ($B(u_{VBD})$, F_{δ_1}, F_{δ_2}) and inertial forces (including added mass). The forces are modelled in the inertial frame (origin $\{O_i\}$, in orange), body-fixed frame (origin $\{O_b\}$, in green) and flow-fixed frame (origin $\{O_b\}$, in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$\rho g u_{VBD}$, with ρ the density of the surrounding fluid. In this preliminary study, the hull glider volume is assumed constant and not depending on the vehicle depth. More advanced formulations embedding hull compression due to pressure and temperature can be considered when advancing this application (Anderlini et al., 2020; Bennett et al., 2021). According to Zhang and Tan (2015), each stern plane j generates an additional lift force parallel to z_b, computed as $F_{\delta_j} = K_{F_\delta} K_{u_\delta} V^2 \delta_j$, with K_i representing wing-specific hydrodynamic coefficients and δ_j the deflection angle, as shown in Fig. 9. 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 (Grande et al., 2021).

This model leverages three assumptions: in profiling operations, as the VBD is actuated over long time span of 30 [s] or above in order to save energy, the VBD dynamics is neglected (Bennett et al., 2021). Next, it is assumed that the body-fixed velocities are measured, for instance by means of a Doppler Velocity Logger. Finally, pitch angle and angle of attack changes are considered negligible, as it is standard practice for the pilots to tune flight parameters to ensure a constant glide slope angle.

The corresponding dynamic model, derived based on previous works (Bennett et al., 2021; Graver, 2005; Zhou et al., 2020), is described as:

$$\begin{cases} \dot{v}_1 = \dfrac{1}{m_1}(-D\cos\alpha + L\sin\alpha + \sin\theta(B - G_s - G_p)) \\ \dot{v}_3 = \dfrac{1}{m_3}(-D\sin\alpha - L\cos\alpha + \cos\theta(-B + G_s + G_p) + F_{\delta_1} + F_{\delta_2}) \end{cases} \tag{19}$$

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 glides, hydrodynamic forces are computed through the quasi-steady state approximation as:

$$\begin{aligned} D &\approx (K_{D0} + K_D \alpha^2) V^2 \\ L &\approx (K_{L0} + K_L \alpha) V^2 \end{aligned} \tag{20}$$

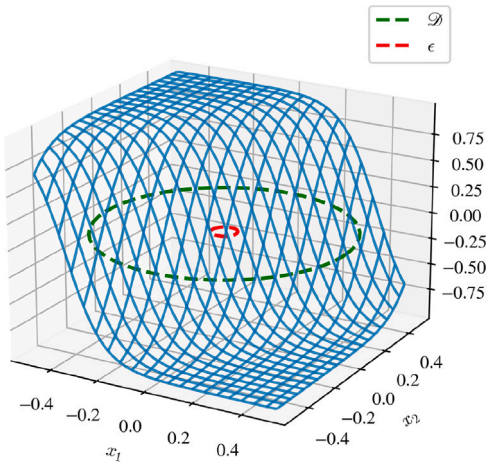


Fig. 10. Synthesised control function for the Underwater Glider encompassing actuator saturation (u_1 with $\overline{\sigma}_1 = 1.0$).

where $V = \sqrt{v_1^2 + v_3^2}$, and K_{D0}, K_D, K_{L0}, K_L represent the hydrodynamic coefficients, estimated through model identification or CFD analyses.

For this study, the Seaglider (ogive model) geometric and hydrodynamic parameters derived in Grande et al. (2021) are employed as follows: $m_s = 44.9$ kg, $m_p = 11.0$ kg, $m_1 = 64.84$ kg, $m_3 = 99.43$ kg, $K_{D_0} = 0.04$ kg/m, $K_D = 9.44$ kg/(m rad^2), $K_{L_0} = 2.16$ kg/m, $K_L = 4.88$ kg/(m rad), $\nabla_h = 0.054$ m^3, $\rho = 1027.5$ kg/m^3. Finally, $K_{F_\delta} = 10.0$ kg/(m rad), $K_{u_\delta} = 1.0$ are calculated as per standard glider stern planes design (Zhang & Tan, 2015).

For this case study, interest lies in investigating the effect of a stern plane jamming at a specific angle during the ascent phase, due to a frozen hinge or due to the presence of detritus. Additionally, a more advanced control system accounting for saturated control inputs is designed. The control function with saturation is synthesised as follows:

$$\boldsymbol{u} = \begin{bmatrix} \overline{\sigma}_1 \tanh(\cdot) & & \\ & \ddots & \\ & & \overline{\sigma}_m \tanh(\cdot) \end{bmatrix} (\boldsymbol{K}\boldsymbol{e} + \boldsymbol{B}) \tag{21}$$

where $\overline{\sigma}_j$ represents the saturation limits of the j actuator, $\boldsymbol{K}$ the ANN weight, $\boldsymbol{B}$ the ANN bias and $\boldsymbol{e}$ the state-space error vector.

The software tool synthesises 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. 10. The control function is obtained as $u_1 = \tanh(0.014 - 6.571x_1 + 4.912x_2)$, while the resulting CLF is synthesised as: $V = 0.264975x_1x_2 + 0.522009x_1^2 + 0.288912x_2^2$.

The evolution of the learning process is showcased by reporting the changing shape of the Lie derivative function for the nominal dynamics (note that the evolution of the Lie derivatives associated to the system with faults follow the same trend). In Fig. 11(a) (learning iteration #1) and Fig. 11(b) (learning iteration #5) the contour lines of the Lie derivatives can be appreciated exhibiting positive values within $\mathscr{D}$. Upon convergence, the Lie derivative becomes negative definite over the entire domain, as reported in Fig. 11(c). Eventually, a closed-loop system verification test is reported in Fig. 12, showing the capability to track $\boldsymbol{x}^\star$ within the target bounds despite the jamming of δ_2 happening at t = 1.8 [s].

7. Control law selection

The output of a pFT-ANLC *training campaign* is not a unique control law, but rather a series of synthesised laws, all certified to hold stabilising properties, but with performance slightly different from one another. As each run is generated via a different (random) seed, every training starts from a different initialisation of the ANN parameters and it might converge to different minima. Scope of this section is to provide tools to select which control law among the ones generated better fits target operating objectives, such as prescribed values of steady-state tracking accuracy, or lower energy consumption.

A Linear-Quadratic criterion is conventionally employed to assess control-system performance (Jiang & Yu, 2012). In this work, a criterion encompassing both a term related to the reference tracking error and a factor associated to the control effort is proposed as:

$$J = \int_0^\infty (\boldsymbol{e}^T \boldsymbol{Q}\boldsymbol{e} + \boldsymbol{u}_f^T \boldsymbol{R}\boldsymbol{u}_f) dt \tag{22}$$

where $\boldsymbol{u}_f$ represents the control signal embedding the faults, $\boldsymbol{e}$ the reference tracking error with $\boldsymbol{e}(t) = \boldsymbol{r}(t) - \boldsymbol{x}(t)$ and $\boldsymbol{r}(t)$ the reference tracking value, while $\boldsymbol{Q}$ and $\boldsymbol{R}$ (symmetric positive semi-definite) are the weighting matrixes. For convenience, the performance index J is split into the sum of two terms:

$$J = J_e + J_u \tag{23}$$

where J_e denotes the performance related to the tracking error and J_u the one associated to the control input. J_e can be designed to evaluate the tracking ability of a controller, namely:

$$J_e = \bar{\boldsymbol{e}}^T \boldsymbol{Q_1} \bar{\boldsymbol{e}} + \bar{\boldsymbol{e}}_{\boldsymbol{bf}}^T \boldsymbol{Q_2} \bar{\boldsymbol{e}}_{\boldsymbol{bf}} + \bar{\boldsymbol{e}}_{\boldsymbol{af}}^T \boldsymbol{Q_3} \bar{\boldsymbol{e}}_{\boldsymbol{af}} + e_{st}^T Q_4 e_{st} \tag{24}$$

where:

- $\bar{e}$ 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 subscript *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 matrixes.

Note that the first three terms of (24) express how accurate the dynamics tracks the target value, whilst the last one indicates how fast the dynamics reaches the steady-state value. The settling time is calculated as the time required to reach and to remain bounded within a ±2% error from the target setpoint value.

To evaluate the performance of the control signals instead, the following cost function J_u is defined:

$$J_u = \max(\boldsymbol{u})^T \boldsymbol{R_1} \max(\boldsymbol{u}) + \max\left(\frac{d\boldsymbol{u}}{dt}\right)^T \boldsymbol{R_2} \max\left(\frac{d\boldsymbol{u}}{dt}\right) + R_3 \int_0^T P(t) dt \tag{25}$$

where the first and second terms penalise the maximum values of the control input and of the actuator rates, respectively, while the latter factor accounts for the power consumption over time. If the force generated by each thruster (F_i) is know, the overall power consumption at time $t = k_t$ can be computed as $P_{k_t} = (\sum_{i=0}^{m} |F_{i,k_t}|^{(3/2)})$, with m the total number of available thrusters (Fossen, 2011). With such a convention, it follows that:

$$\boldsymbol{e} \in \mathbb{R}^n \tag{26}$$

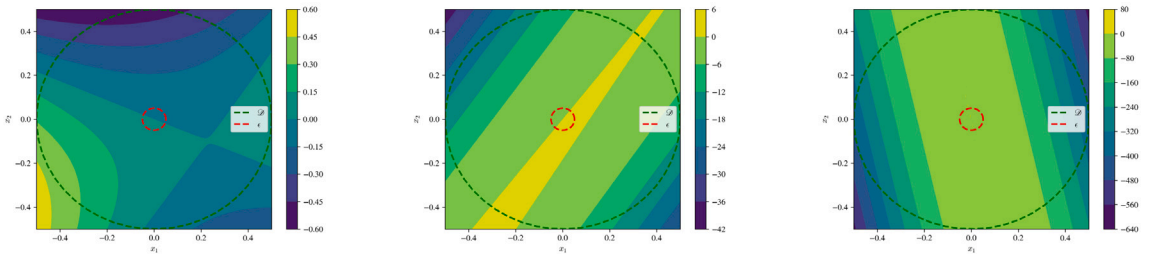
$$\boldsymbol{u} \in \mathbb{R}^m \tag{27}$$

$$\boldsymbol{Q_1}, \boldsymbol{Q_2}, \boldsymbol{Q_3} \in \mathbb{R}^{n \times n} \tag{28}$$

$$\boldsymbol{R_1}, \boldsymbol{R_2} \in \mathbb{R}^{m \times m} \tag{29}$$

$$R_3, Q_4 \in \mathbb{R}. \tag{30}$$

Hereby, quantitative examples of choices of performance criteria are provided. Given system (18), a simulation campaign consisting of 10 runs is executed. Two cost function tunings are devised. The first tuning, hereby denoted as $\overline{J}^a$, prioritises reference tracking performance after the faults over the control effort, i.e. the tracking error penalisation terms are the highest. On the other hand, the second tuning aims at limiting the overall power consumption: tracking goals are



(a) Training iteration #1 (initial random value). (b) Training iteration #5 (during the training). (c) Training iteration #1944 (upon convergence).

Fig. 11. Underwater Glider training: Lie derivative associated to the nominal dynamics at successive training iterations.

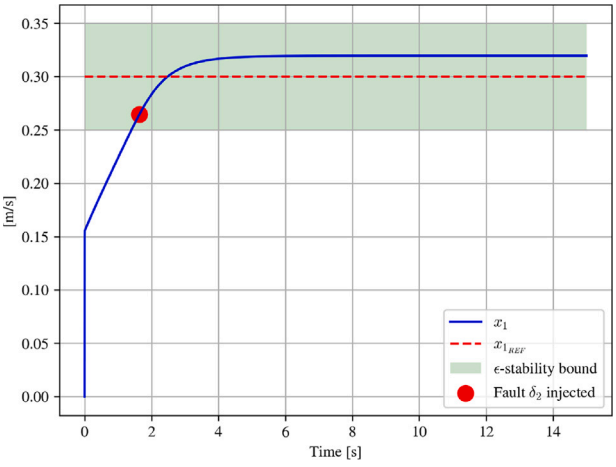


Fig. 12. Underwater Glider verification test: surge speed when the control surface δ_2 jams at t=1.8 [s].

relaxed while the power consumption penalisation gain is the highest. For instance, possible weighting terms associated with the first tuning choice are as follows:

$$\boldsymbol{Q}_1^a = \boldsymbol{Q}_2^a = \text{diag}(1.0,\ 1.0)$$
$$\boldsymbol{R}_1^a = \boldsymbol{R}_2^a = \text{diag}(1.0,\ 1.0,\ 1.0)$$
$$Q_4^a = R_3^a = 1.0$$
$$\boldsymbol{Q}_3^a = \text{diag}(100.0,\ 100.0),$$

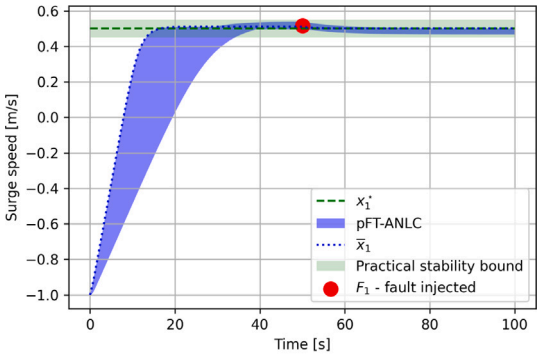
where the term $\boldsymbol{Q}_3$ penalises the tracking error following the occurrence of the faults and Q_4 penalises the settling time.

On the other hand, the control tuning that minimises the overall power consumption, denoted as $\overline{J}^b$, can be selected by choosing the following factors:

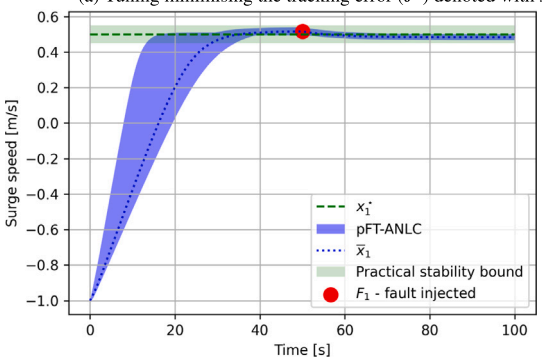
$$\boldsymbol{Q}_1^b = \boldsymbol{Q}_2^b = \boldsymbol{Q}_3^b = \text{diag}(1.0,\ 1.0)$$
$$\boldsymbol{R}_1^b = \boldsymbol{R}_2^b = \text{diag}(1.0,\ 1.0,\ 1.0)$$
$$Q_4^b = 1.0$$
$$R_3^b = 100.0.$$

The closed-loop dynamics resulting from the two different tunings are compared hereby. Fig. 13(a) reports the vehicle surge speed associated with the tuning prioritising the reference tracking error following the fault, where $\overline{x}_1$ (dotted line) denotes the controller minimising $\overline{J}^a$. Similarly, Fig. 13(b) illustrates the resulting dynamics of the more energy-conservative tuning $\overline{J}^b$. A zoomed view of the surge dynamics following the injection of a fault on F_1 is reported in Fig. 14: the chosen control law associated with $\overline{J}^a$ minimises the tracking error following the occurrence of a fault, as desired.

Finally, the forces generated by the three thrusters are illustrated in Figs. 15(a) and 15(b) for the tuning $\overline{J}^a$ and $\overline{J}^b$, respectively. As



(a) Tuning minimising the tracking error ($\overline{J}^a$) denoted with $\overline{x}_1$.



(b) Tuning minimising the overall power consumption ($\overline{J}^b$) denoted with $\overline{x}_1$.

Fig. 13. Range of possible surge dynamics associated to the converged controllers with fault on F_1 at 50 [s].

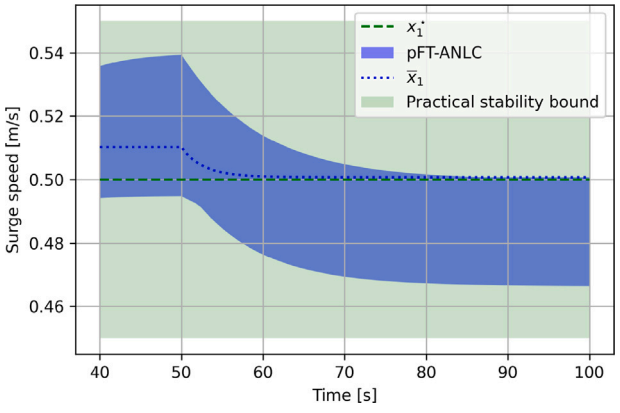
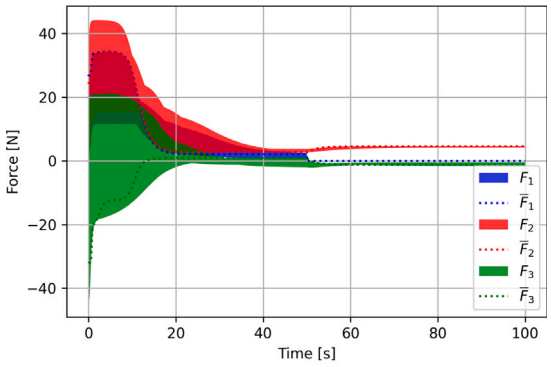
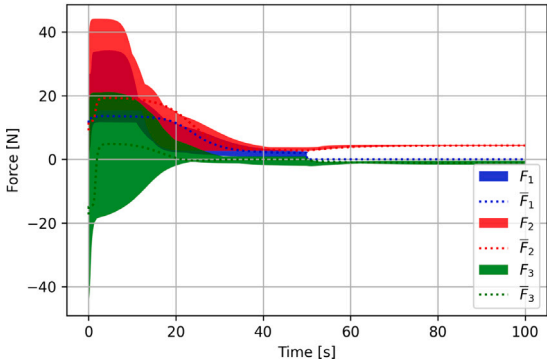


Fig. 14. Zoomed view of Fig. 13(a): the selected controller minimises the tracking error ($x_1^\star - \overline{x}_1$) following a fault on F_1 at 50 [s] (tuning $\overline{J}^a$).



(a) Tuning minimising the tracking error ($\overline{J}^a$) highlighted with dashed lines.



(b) Tuning minimising the overall power consumption ($\overline{J}^b$) highlighted with dashed lines.

Fig. 15. Range of control efforts associated to the converged controllers with different tuning following a fault on F_1.

expected, tuning $\overline{J}^a$, devised to lead to fast converging dynamics, requires an excess of forces when compared to $\overline{J}^b$.

8. Actuator loss of efficiency

In the previous sections, the cases of a jammed control surface and of a complete loss of one or more actuators were investigated. Whilst the case of the complete loss of actuator effectiveness represents an interesting case study, oftentimes, in real-life applications, faults appear as partial loss of actuator efficiency. This is the case when minor electric or mechanical issues arise, or when fishing debris or seaweed get tangled in the underwater thrusters, increasing the spinning resistance and limiting the overall revolving speed. In the specific case of gliders with missions lasting for weeks or months, biofouling of marine growth on the control surfaces can reduce the efficiency of the lift generated by a stern plane or the thrust generated by a propeller. These occurrences do not lead to a complete actuator loss altogether, but rather limit the delivered actuator effort, and can be modelled as a factor multiplying the nominal control effort.

In line with the rest of this work, the aim of the present section remains 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 modelling introduced in Section 5 can be extended by accounting for the *efficiency* of the jth actuator, modelled as a real variable $\mu_j \in [0, 1]$. In practical terms, the Lyapunov conditions (2) can be extended to include non-binary faults. This formally results in:

$$V(\boldsymbol{x}^\star) = 0, \tag{31a}$$

$$V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\}, \tag{31b}$$

$$\dot{V}_n(\boldsymbol{x}) < 0 \wedge \{\dot{V}_{\mu_j}(\boldsymbol{x}, \boldsymbol{u}) < 0\}_{j=1}^{p} \quad \forall \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\}, \tag{31c}$$

where $\dot{V}_{\mu_j}(\boldsymbol{x}, \boldsymbol{u})$ represents the Lie derivative in presence of a fault on the jth actuator with actuator efficiency μ_j. In a similar spirit, the Falsifier conditions (17) can be modified accordingly, as follows:

$$\exists(\boldsymbol{x} : \boldsymbol{x} \in \mathscr{D} \setminus \{\boldsymbol{x}^\star\}, \boldsymbol{\mu} : \boldsymbol{\mu} \in [0, 1]) \implies \left(V(\boldsymbol{x}) \leq 0 \ \vee \ \dot{V}_n(\boldsymbol{x}, \boldsymbol{u}) \geq 0 \ \vee \{\dot{V}_{\mu_j}(\boldsymbol{x}, \boldsymbol{u}) \geq 0\}_{j=1}^{p}\right), \tag{32}$$

where any instance $(\boldsymbol{x}, \boldsymbol{\mu})$ that satisfies these conditions would falsify the candidate CLF, and hence the proposed control law, triggering a restart of the training procedure.

8.1. Case study C: Autonomous underwater vehicle with loss of efficiency

The concept is once more illustrated in the case of the two-dimensional AUV introduced in Section 5. The model including a possible loss of efficiency on the first actuator (F_1) can be expressed as:

$$\begin{cases} \dot{x}_1 = \dfrac{-X_u x_1 - X_{uu} x_1^2 + \mu_1 F_{1,x} + F_{2,x}}{m} \\ \dot{x}_2 = \dfrac{-N_r x_2 - N_{rr} x_2^2 + (-F_{1,x} l_{1,y} + F_{1,y} l_{1,x})\mu_1}{J_z} + \\ \qquad \dfrac{(-F_{2,x} l_{2,y} + F_{2,y} l_{2,x}) + (-F_{3,x} l_{3,y} + F_{3,y} l_{3,x})}{J_z} \end{cases} \tag{33}$$

where the same coefficients introduced in (18) are employed. The result of a successful training run with target $\mathbf{x}^\star = [0.5, 0.0]^T$ and $\epsilon = 0.025$ is illustrated hereby. To verify the correctness of the method, the efficiency μ_1 is varied over time according to the profile reported in Fig. 16. The corresponding forces applied by the controller during a closed-loop test can be viewed in Fig. 17, with the resulting dynamics illustrated in Fig. 18(a) and Fig. 18(b).

It is possible to notice how, despite the efficiency of the actuator is arbitrarily varied over different values in the range (100% to 0%), the control systems is able to bound the dynamic response within the prescribed ϵ-stability threshold.

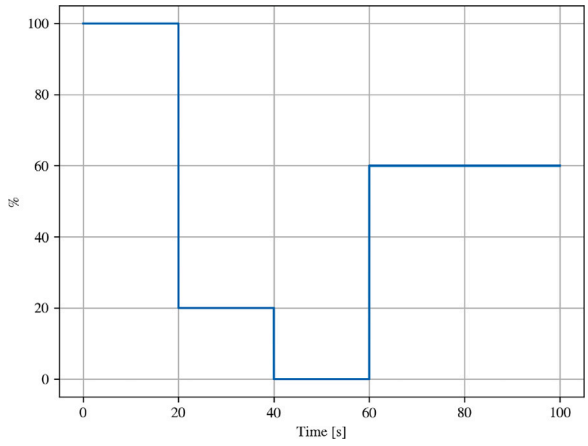


Fig. 16. Partial loss of efficiency for the AUV case study: efficiency of the first actuator (μ_1) over time.

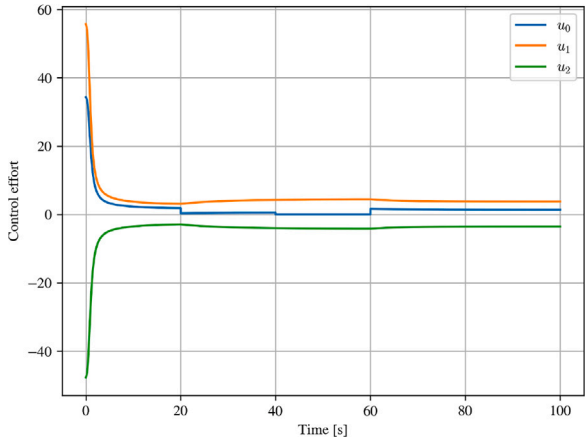
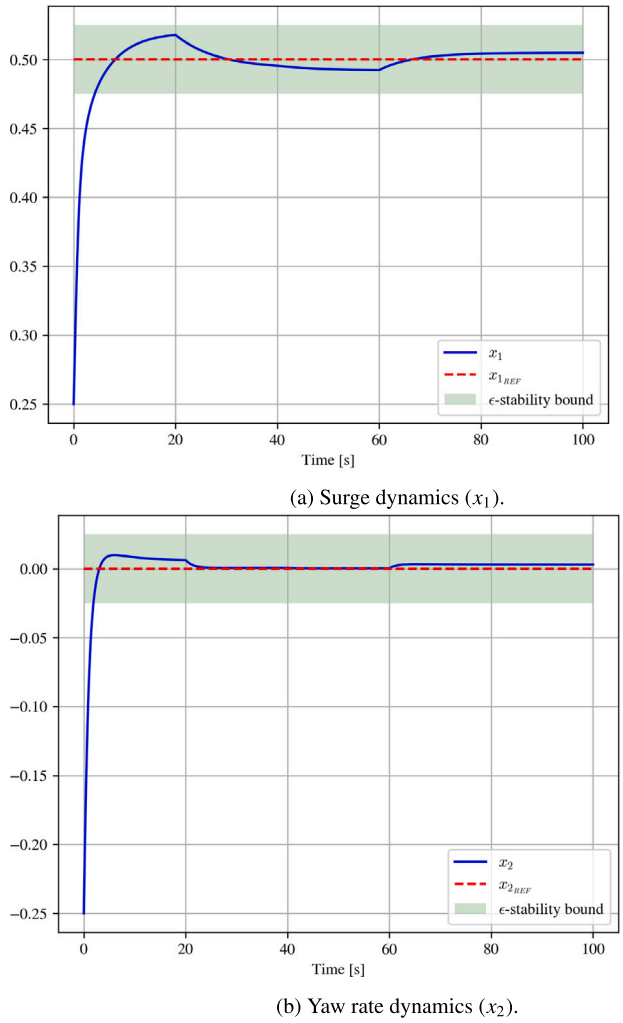


Fig. 17. Partial loss of efficiency for the AUV case study: control input forces.



(a) Surge dynamics (x_1).

(b) Yaw rate dynamics (x_2).

Fig. 18. Partial loss of efficiency for the AUV case study: dynamic response to varying thruster efficiency (μ_1).

9. Control techniques comparison and region of attraction shaping

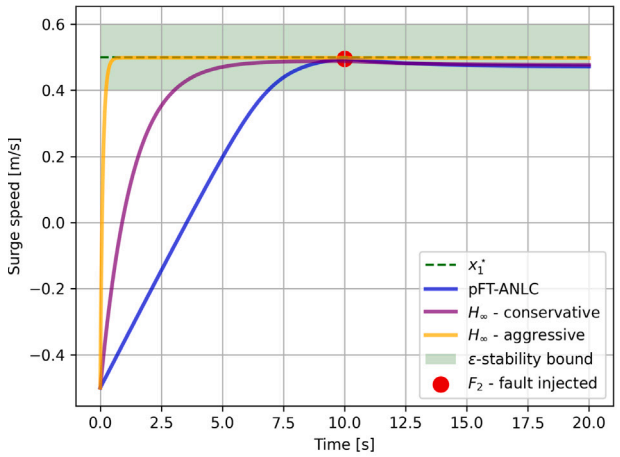
In this section, results of the synthesis of pFT-ANLC laws for the two-dimensional AUV are compared to the state-of-the-art technique within the frame of Passive Fault-Tolerant Control for Autonomous Underwater Vehicles, namely the $\mathscr{H}_\infty$ robust controller. $\mathscr{H}_\infty$-control synthesis aims at finding a stabilising controller that minimises the infinity norm of the transfer matrix T_{zw} between exogenous inputs w and desired performances z, namely $\|T_{zw}\|_\infty$ (Blanke et al., 2006). The problem entails the simultaneous stabilisation of a plant under both nominal and multiple failure modes, each one associated to the failure of one actuator. As the pFT-ANLC aims at optimising the tunable parameters of a fixed ANN structure, the $\mathscr{H}_\infty$ control problem is equivalently formulated as an optimisation procedure of a fixed-structure control subjected to a set of design requirements. Design requirements are specified in terms of target response time, maximum steady-state error, and, at times, in terms of the maximum effort of an actuator. Additionally, actuator saturation is accounted for in this control design problem. Realistic saturation values are proposed as taken from the *BlueRobotics T200* thruster, chosen as an example due its ubiquitous presence in underwater robotics applications.

Three $\mathscr{H}_\infty$ optimisation problems with different requirements are formulated as reported in Table 3, stemming in controllers with significantly different performance. All the controllers are computed after linearising the dynamics around the target operating point $\mathbf{x}^\star = [0.5, 0.0]^T$, coinciding with the prescribed setpoint equilibrium. To tune the gain, a target response time of 10 [s] is set (a realistic value in the considered application) while the maximum tolerated steady-state error is varied from 40% in the conservative scenarios, to a more stringent 5% in the aggressive counterpart. In the conservative scenarios, an additional constraint to limit the control effort is considered. To this aim, *MATLAB systune* toolbox is employed to synthesise a control gain that stabilises the closed-loop whilst subjected to design requirements expressed as hard optimisation goals. The aggressive tuning problem is promptly solved within 1 [s] of computational time. On the other hand, the synthesis fails to converge when the saturation limit of *BlueRobotics T200* thruster (namely 37.1 [N]) is imposed on the control values of the three thrusters, despite 10 successive rounds of optimisation. The conservative solution is thus obtained by iteratively increasing the control effort saturation limit from 37.1 [N] until 880.0 [N], found to be the first value that prompts the $\mathscr{H}_\infty$ synthesis to successfully converge. The conservative synthesis scenarios are solved within a maximum of 208 iterations, completed within 10 [s] of computational time.

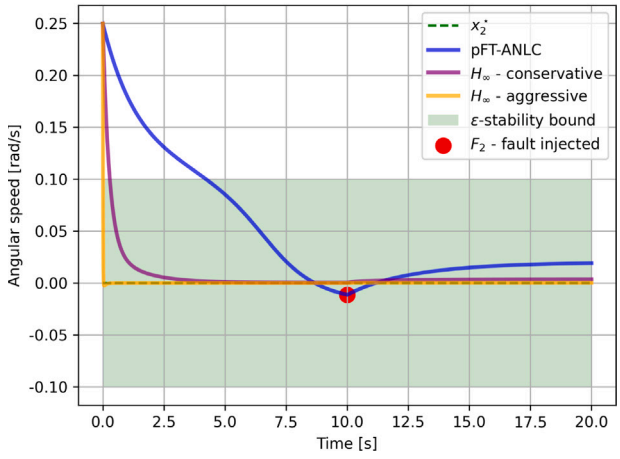
Next, the $\mathscr{H}_\infty$ laws are compared to one instance of converged pFT-ANLC. The pFT-ANLC solution is obtained by selecting the architecture hyperparameters described in Table 1, whilst adding the control saturation value of 37.1 [N]. The resulting closed-loop systems when a fault on F_2 occurs are compared: Fig. 19(a) reports the closed-loop surge

Table 3
$\mathscr{H}_\infty$-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



(a) Surge dynamics (x_1).



(b) Yaw rate dynamics (x_2).

Fig. 19. Control laws comparison for AUV case study: pFT-ANLC (blue) vs multiple $\mathscr{H}_\infty$ tuning (purple and orange). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

dynamics, while Fig. 19(b) illustrates the corresponding angular rate dynamics. It can be appreciated how the synthesised pFT-ANLC law (in blue) stems into dynamics that show qualitatively comparable steady-state errors to the $\mathscr{H}_\infty$ ones, whilst slower in terms of convergence time. As expected, the *aggressive* $\mathscr{H}_\infty$ tuning (in orange) shows faster convergence rate and improved tracking performance both before and after the occurrence of the fault at the price of an increased control effort, reported in the case of F_1 in Fig. 20. The *aggressive* $\mathscr{H}_\infty$ tuning in fact requires control effort two orders of magnitude higher than the pFT-ANLC one (F_1 and F_3 follow similar trends). Additionally, Fig. 20 highlights how pFT-ANLC is the only controller that guarantees feasible control actions by strictly respecting the actuator saturation limit.

These results highlight how the proposed framework can synthesise control laws that are qualitatively comparable with well established robust control techniques, whilst providing additional benefits from a formal guarantees perspective. Despite $\mathscr{H}_\infty$ techniques benefit from highly optimised commercial tools resulting in faster tuning of control laws (10 [s] for the $\mathscr{H}_\infty$ tuning, against the 92 [s] of the pFT-ANLC, in this study), pFT-ANLC provides benefits from both 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 the ones characterising the underwater domain. Second, as the pFT-ANLC synthesises a CLF, additional information regarding the properties of the nonlinear closed-loop system is available, such as the shape and dimension of the ROA. Conventionally, in control design applications, interest lies in providing a control law stemming into a ROA as large as possible. To this aim, pFT-ANLC allows shaping the ROA by calibrating α_4 and α_{ROA} in (15). To showcase the ROA shaping procedure, a first training is run with $\alpha_4 = 0.0$ (disabling the ROA tuning factor), and, following, a second training encompassing $\alpha_4 = 1.0$ and $\alpha_{ROA} = 100.0$ is carried out. Upon training completion, the resulting ROAs associated with the two CLFs are calculated by means of a line search algorithm. The results are reported in Figs. 21(a) and 21(b) for the ROA without tuning factor, and with tuning factor, respectively. It is noteworthy how, in the latter case, the ROA approximately overlaps the whole domain $\mathscr{D}$ as circular level sets are enforced during the training.

This analysis highlights how the joined synthesis of nonlinear control laws and CLFs proposed in this manuscript can be employed to generate control laws capable of catering for practical nonlinear effects, such as actuator saturation. As additional benefit, the resulting ROA can not only be estimated, but can be shaped via dedicated tuning parameters during the synthesis stage. These elements represent key advantages in further assessing closed-loop dynamic properties when compared to the robust techniques traditionally employed in the field.

10. Further discussions and conclusions

In this paper, a novel, automated method to design Passive Fault-Tolerant Control laws is presented. The proposed approach relies on the ANLC architecture to synthesise a control function with fixed gain that guarantees the ϵ-stability of the target equilibrium in both the nominal and faulty conditions. Both linear and nonlinear control laws can be synthesised based on user preference. The approach is shown 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 is developed and released open-source. The proposed tool can be run on standard office laptops without demanding performance requirements. The approach is of relevance for every control application where extensive sensoring or fault monitoring algorithms cannot be assumed, as in the case of autonomous underwater platforms. Faults of different nature, covering control surfaces jamming, full and partial loss of actuator efficiency are analysed, with encouraging outcomes. The approach devised in this work represents a first step towards the inclusion of *correct-by-design* machine-learning based techniques, leveraging formal guarantees via SMT solvers, to navigation problems. As the integration of artificial intelligence within both common and niche applications is becoming ubiquitous, modern control designs can be devised to guarantee desirable dynamical behaviours, without compromising on formal stability guarantees. Far from being a universal answer to nontrivial control problems, this work exploits the flexibility of neural networks to compose complex control laws accounting for actuator saturation and allowing to shape the Region Of Attraction, proposing out-of-the-box solutions to an expert control engineer. Future efforts encompass improving the approach's scalability to higher-dimensional systems, along with the inclusion of noisy measurements and disturbances.

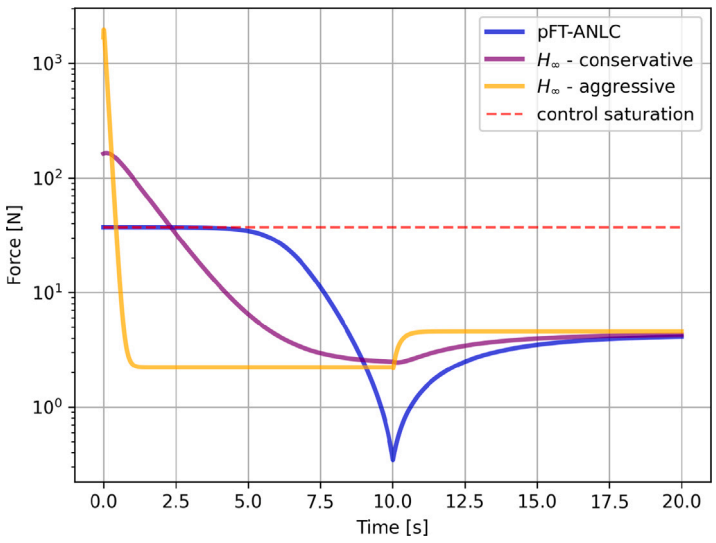
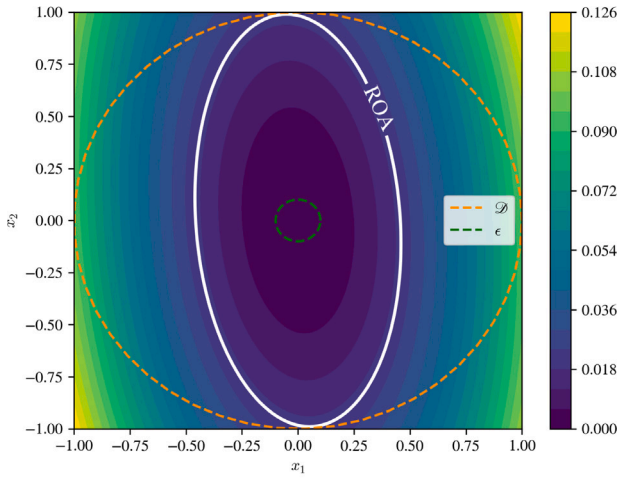
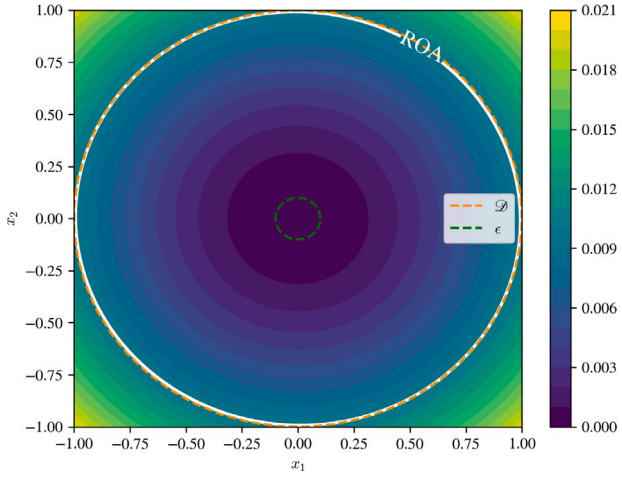


Fig. 20. Control laws comparison for AUV case study: F_1 control effort with pFT-ANLC strictly respecting the desired actuator saturation.



(a) CLF synthesised *without* ROA tuning factor.



(b) CLF synthesised *with* ROA tuning factor.

Fig. 21. pFT-ANLC controller — different CLF tuning factors. Domain boundaries in dashed lines and ROA in white solid line.

CRediT authorship contribution statement

Davide Grande: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Andrea Peruffo:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Georgios Salavasidis:** Writing – review & editing, Supervision, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Enrico Anderlini:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Davide Fenucci:** Writing – review & editing, Visualization, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Alexander B. Phillips:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Funding acquisition, Conceptualization. **Elias B. Kosmatopoulos:** Writing – review & editing, Visualization, Validation, Supervision, Resources, Methodology, Formal analysis. **Giles Thomas:** Writing – review & editing, Validation, Supervision, Project administration, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abate, A., Ahmed, D., Edwards, A., Giacobbe, M., & Peruffo, A. (2021). 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* (pp. 1–11).

Abate, A., Ahmed, D., Giacobbe, M., & Peruffo, A. (2020). Formal synthesis of Lyapunov neural networks. *IEEE Control Systems Letters*, *5*(3), 773–778.

Ahmed, D., Peruffo, A., & Abate, A. (2020). Automated and sound synthesis of Lyapunov functions with SMT solvers. In *Tools and algorithms for the construction and analysis of systems, proceedings, part i 26* (pp. 97–114). Springer.

Anderlini, E., Parker, G. G., & Thomas, G. (2019). Docking control of an autonomous underwater vehicle using reinforcement learning. *Applied Sciences*, *9*(17), 3456.

Anderlini, E., Thomas, G., Woodward, S. C., Real-Arce, D. A., Morales, T., Barrera, C., & Hernández-Brito, J. (2020). Identification of the dynamics of biofouled underwater gliders. In *2020 IEEE/OES autonomous underwater vehicles symposium* (pp. 1–6). IEEE.

Bennett, J. S., Stahr, F. R., Eriksen, C. C., Renken, M. C., Snyder, W. E., & Van Uffelen, L. J. (2021). Assessing seaglider model-based position accuracy on an acoustic tracking range. *Journal of Atmospheric and Oceanic Technology*, *38*(6), 1111–1123.

Benosman, M., & Lum, K.-Y. (2009). Passive actuators' fault-tolerant control for affine nonlinear systems. *IEEE Transactions on Control Systems Technology*, *18*(1), 152–163.

Blanke, M., Kinnaert, M., Lunze, J., Staroswiecki, M., & Schröder, J. (2006). *Diagnosis and fault-tolerant control: vol. 2*, Springer.

Boem, F., Gallo, A. J., Raimondo, D. M., & Parisini, T. (2019). Distributed fault-tolerant control of large-scale systems: An active fault diagnosis approach. *IEEE Transactions on Control of Network Systems*, *7*(1), 288–301.

Brito, M., Smeed, D., & Griffiths, G. (2014). Underwater glider reliability and implications for survey design. *Journal of Atmospheric and Oceanic technology*, *31*(12), 2858–2870.

Caccia, M., Bono, R., Bruzzone, G., Bruzzone, G., Spirandelli, E., & Veruggio, G. (2001). Experiences on actuator fault detection, diagnosis and accomodation for ROVs. *International Symposiyum of Unmanned Untethered Sub-mersible Technol*, 3–21.

Carlucho, I., De Paula, M., Wang, S., Petillot, Y., & Acosta, G. G. (2018). Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. *Robotics and Autonomous Systems*, *107*, 71–86.

Chang, Y.-C., Roohi, N., & Gao, S. (2019). Neural lyapunov control. *Advances in Neural Information Processing Systems*, *32*.

Dawson, C., Gao, S., & Fan, C. (2023). Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods. *IEEE Transactions on Robotics*.

Dooraki, A. R., & Lee, D.-J. (2020). 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* (pp. 161–166). IEEE.

Edwards, A., Peruffo, A., & Abate, A. (2023). A General Verification Framework for Dynamical and Control Models via Certificate Synthesis. arXiv:2309.06090.

Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons.

Gao, S., Kapinski, J., Deshmukh, J., Roohi, N., Solar-Lezama, A., Aréchiga, N., & Kong, S. (2019). Numerically-robust inductive proof rules for continuous dynamical systems. In *Computer aided verification: 31st international conference* (pp. 137–154). Springer.

Gao, S., Kong, S., & Clarke, E. M. (2013). Dreal: An SMT solver for nonlinear theories over the reals. In *International conference on automated deduction* (pp. 208–214). Springer.

Grande, D., Fenucci, D., Peruffo, A., Anderlini, E., Philips, A. B., Thomas, G., & Salavasidis, G. (2023). Systematic synthesis of passive fault-tolerant augmented neural Lyapunov control laws for nonlinear systems. In *2023 62nd IEEE conference on decision and control*. IEEE.

Grande, D., Huang, L., Harris, C. A., Wu, P., Thomas, G., & Anderlini, E. (2021). Open-source simulation of underwater gliders. In *OCEANS 2021: san diego–porto* (pp. 1–8). IEEE.

Grande, D., Peruffo, A., Anderlini, E., & Salavasidis, G. (2023). Augmented neural Lyapunov control. *IEEE Access*, *11*, 67979–67986.

Graver, J. G. (2005). *Underwater gliders: Dynamics, control and design* (Ph.D. thesis), Princeton University.

Huang, Y., Qiao, J., Yu, J., Wang, Z., Xie, Z., & Liu, K. (2019). Sea-whale 2000: A long-range hybrid autonomous underwater vehicle for ocean observation. In *Oceans 2019-marseille* (pp. 1–6). IEEE.

Jiang, J., & Yu, X. (2012). Fault-tolerant control systems: A comparative study between active and passive approaches. *Annual Reviews in Control*, *36*(1), 60–72.

Kaminer, I., Pascoal, A. M., Silvestre, C. J., & Khargonekar, P. P. (1991). Control of an underwater vehicle using h/sub infinity/synthesis. In *[1991] proceedings of the 30th IEEE conference on decision and control* (pp. 2350–2355). IEEE.

Katebi, M., & Grimble, M. J. (1999). Integrated control, guidance and diagnosis for reconfigurable autonomous underwater vehicle control. *International Journal of Systems Science*, *30*(9), 1021–1032.

La Salle, J., & Lefschetz, S. (1961). *Stability by Liapunov's direct method with applications*. Academic Press, RIAS.

Leonard, N. E., & Graver, J. G. (2001). Model-based feedback control of autonomous underwater gliders. *IEEE Journal of Oceanic Engineering*, *26*(4), 633–645.

Patan, K. (2019). *Robust and fault-tolerant control: neural-network-based solutions*. Springer.

Phillips, A. B., Kingsland, M., Linton, N., Baker, W., Bowring, L., Soper, S., Roper, D. T., Johnson, A., Morrison, R., Ciaramella, K., Matterson, D., Pebody, M., Marlow, R., Consensi, A., Williams, V., Fanelli, F., Fenucci, D., Martin, A., & Ó hÓbáin, E. (2020). Autosub 2000 under ice: Design of a new work class AUV for under ice exploration. In *2020 IEEE/OES autonomous underwater vehicles symposium* (pp. 1–8). IEEE.

Queste, B. Y., Heywood, K. J., Kaiser, J., Lee, G. A., Matthews, A., Schmidtko, S., Walker-Brown, C., & Woodward, S. W. (2012). Deployments in extreme conditions: Pushing the boundaries of seaglider capabilities. In *2012 IEEE/OES autonomous underwater vehicles* (pp. 1–7). IEEE.

Solar-Lezama, A., Tancau, L., Bodik, R., Seshia, S., & Saraswat, V. (2006). Combinatorial sketching for finite programs. In *Proceedings of the 12th international conference on architectural support for programming languages and operating systems* (pp. 404–415).

Stanway, M. J., Kieft, B., Hoover, T., Hobson, B., Klimov, D., Erickson, J., Raanan, B. Y., Ebert, D. A., & Bellingham, J. (2015). White shark strike on a long-range AUV in Monterey bay. In *OCEANS 2015-genova* (pp. 1–7). IEEE.

Tedrake, R. (2023). Underactuated robotics. URL https://underactuated.csail.mit.edu.

Thanh, P. N. N., & Anh, H. P. H. (2022). Advanced neural control technique for autonomous underwater vehicles using modified integral barrier Lyapunov function. *Ocean Engineering*, *266*, Article 112842.

Verhaegen, M., Kanev, S., Hallouzi, R., Jones, C., Maciejowski, J., & Smail, H. (2010). Fault tolerant flight control - a survey. In C. Edwards, T. Lombaerts, & H. Smaili (Eds.), *Fault tolerant flight control: a benchmark challenge* (pp. 47–89). Berlin, Heidelberg: Springer Berlin Heidelberg, ISBN: 978-3-642-11690-2.

Webster, S. E., Freitag, L. E., Lee, C. M., & Gobat, J. I. (2015). Towards real-time under-ice acoustic navigation at mesoscale ranges. In *2015 IEEE international conference on robotics and automation* (pp. 537–544). IEEE.

Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, *12*(6), 601–611.

Zhang, Y., & Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, *32*(2), 229–252.

Zhang, F., & Tan, X. (2015). Passivity-based stabilization of underwater gliders with a control surface. *Journal of Dynamic Systems, Measurement, and Control*, *137*(6), Article 061006.

Zhou, H., Wei, Z., Zeng, Z., Yu, C., Yao, B., & Lian, L. (2020). Adaptive robust sliding mode control of autonomous underwater glider with input constraints for persistent virtual mooring. *Applied Ocean Research*, *95*, Article 102027.