Systematic Synthesis of Passive Fault-Tolerant Augmented Neural Lyapunov Control Laws for Nonlinear Systems

Davide Grande[1,2,⋆], Davide Fenucci[2], Andrea Peruffo[3],
Enrico Anderlini[1], Alexander B. Phillips[2], Giles Thomas[1], Georgios Salavasidis[2]

***Abstract*— Performance and closed-loop stability of control systems can be jeopardised by actuator faults. Actuator redundancy in combination with appropriate control laws can increase the resiliency of a system to both loss of efficiency or jamming. Passive Fault-Tolerant Control (FTC) systems aim at designing a unique control law with guaranteed stability in both nominal and faulty scenarios. In this work, a novel machine learning-based method is devised to systematically synthesise control laws for systems affected by actuator faults, whilst formally certifying the closed-loop stability. The learning architecture trains two Artificial Neural Networks, one representing the control law, and the other resembling a Control Lyapunov Function (CLF). In parallel, a Satisfiability Modulo Theory solver is employed to certify that the obtained CLF formally guarantees the Lyapunov conditions. The method is showcased for two scenarios, one encompassing the stabilisation of an inverted pendulum with redundant actuators, whilst the other covers the control of an Autonomous Underwater Vehicle. The framework is shown capable of synthesising both linear and nonlinear control laws with minimal hyperparameter tuning and within limited computational resources.**

I. Introduction

A fault is usually defined as a change in the characteristics or in the performances of a component that does not compromise the entire functionality of the parent system [1]. Fault Tolerant Control (FTC) aims at preserving the plant operation and guaranteeing the system stability even when unexpected faults occur [2], [3]. The FTC is classically split between *Active* and *Passive* methods. Active FTC (AFTC) methods rely on the presence of a Fault Detection and Isolation system (FDI) and involve either controller redesign or a scheduler of different pre-computed control laws. These are typically computationally expensive, rely on precise information about the model, and suffer from a period of delay due to the FDI system. On the other hand, *Passive* FTC (PFTC) architectures encompass control laws that do not change when a fault occurs. Within the PFTC framework, the goal is the design of controllers that preserve stability in case of a pre-determined set of faults and their anticipated behaviour [3]. With respect to AFTC, PFTC is an attractive option due to the lower computational requirements and complexity. However, designing robust controls results in conservative controllers with low nominal performances [4].

Underwater vehicles are the reference application of this work. In every field where the widespread sensoring and algorithms use cannot be assumed, such as onboard underwater vehicles, PFTC represents the option with the most significant potential impact. Therefore, this class of controllers is the focus of this study. PFTC are oftentimes designed through Robust Control techniques aimed, e.g., at minimising the $\mathcal{H}_2$ or $\mathcal{H}_\infty$-norms between exogenous inputs and desired performances [5]. On the other hand, nonlinear control techniques encompassing extending the nominal control law with an additional Lyapunov function-based term can be employed to guarantee closed-loop stability upon fault occurrence [6]. The former are the most widespread solution in the underwater domain, but they rely on linearisation of highly nonlinear dynamics [7]. The latter instead, do not require any linearisation but cannot be applied when a full loss of actuator occurs, case that will be studied in this work.

In recent times Artificial Neural Network (ANN)-based controllers trained with the Deep Reinforcement Learning (DRL) paradigm have been employed to control different types of Autonomous Underwater Vehicles (AUVs) [8]–[10]. ANN-based controllers are also starting to be employed to design Passive FTC systems [11]. These works illustrate the potential of applying neural controllers to the complex underwater domain, characterised by highly nonlinear and coupled dynamics and unstructured disturbances.

Generally however, machine-learning-based controllers are generated over a finite training set and they offer no guarantees of validity over a continuous domain. Hence recently, several works focus on neural controllers equipped with a formal proof of stability, based on *Satisfiability Modulo Theories* (SMT)-solving. This class of architectures, namely the *Augmented Neural Lyapunov Control* (ANLC), relies on two ANNs, one representing the control law, and the second a Control Lyapunov Function (CLF) [12], [13]. This approach relies on a loop between two modules, a *Learner* and a *Falsifier*. The first is tasked with training the ANNs starting from a small set of initial samples, growing in size based on a Counter Example-Guided Inductive Synthesis (CEGIS) [14], [15]. The latter formally verifies that the ANN represents a CLF for the considered dynamics: the verification step is carried out with SMT solvers, that are designed to evaluate

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).

[1]University College London, Gower St, WC1E 6BT London, UK({e.anderlini,giles.thomas}@ucl.ac.uk)

[2]National Oceanography Centre, SO14 3ZH Southampton, UK ({davfen, abp, geosal}@noc.ac.uk)

[3]Delft University of Technology, Mekelweg 5, 2628 CD Delft, NL (a.peruffo@tudelft.nl)

⋆Corresponding author: grande.rdev@gmail.com

the validity of a symbolic expression over a domain of real numbers. One typical choice is represented by *dReal*, for its capability to handle nonlinear expressions [16]. The ANLC was shown capable to synthesise controllers for unstable dynamics within minimal computational requirements, showing attractive potential to design control laws for complex dynamics such as the ones characterising autonomous vehicles.

Contributions. This work represents, to the best of our knowledge, the first attempt at designing passive fault-tolerant control laws, in an automated way, whilst formally guaranteeing the closed-loop stability. We show how the flexibility of neural networks can be integrated and exploited in safety-critical control scenarios, as underwater autonomous vehicles. Our method requires solely the definition of the dynamic system of interest and the expected set of faults. The proposed approach is based on the ANLC method to compute a control function and a CLF, without needing expert knowledge input to initialise the weights of the networks. The method formally guarantees the (ϵ-) stability of the equilibrium of interest, via SMT-solving.

II. CLF Synthesis via CEGIS

In this work, we aim at designing passive fault-tolerant control laws for a dynamical system

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

where $\boldsymbol{x} \in \mathcal{D} \subseteq \mathbb{R}^n$ is the system's state, $\boldsymbol{u} \in \mathbb{R}^p$ is the control input, and ϕ_i denotes possible system faults. Let us assume each of the p actuators can be affected by complete faults (collected in a set Φ). For brevity, we use the shorthand $\boldsymbol{f_n}(\boldsymbol{x}, \boldsymbol{u})$ for 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 j-th actuator (with $\phi_j \in \Phi$). Along with the design of a control law, we provide a certificate of the closed-loop stability via a CLF.

The synthesis of a CLF is built upon [13] and it employs a neural CEGIS framework, depicted in Fig. 1, in which two NN represent the Lyapunov function and the control law, respectively. Notice that the control network is equipped with a linear and a nonlinear branch. Either one can be trained during the learning process, based on the required complexity specified by the user (defined by a Boolean parameter β denoted *control-branch training selector*, see Fig.1). 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).

Method overview. The CEGIS paradigm evolves based on the information flow between two modules, the *Learner* and the *Falsifier*, as outlined in Fig.2 (see [13] for further details). The former trains the CLF and the control gains by iteratively minimising a loss function, expression of the three theoretical Lyapunov conditions, namely the CLF being positive-definite with its Lie derivative negative-definite over a specified domain, and being zero at the equilibrium. Given an initial finite sample of points, the learning endures until the loss function reaches zero, guaranteeing that, on the sample set, all points respect the theoretical stability conditions. At this stage, the CLF is passed to the *Falsifier*, where the Lyapunov stability conditions are formally evaluated over a bounded domain of *real numbers*. Following, either the CLF is verified to be valid and the procedure is terminated, or new counterexamples (CEs) are generated. In the latter case, the CEs are added to the dataset, which is fed back to the *Learner*, which restarts the training procedure.

a) Learner: Given a 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 $\mathcal{D}$ (containing $\boldsymbol{x}^\star$). At each learning iteration, a cost function is evaluated and the ANN weight is updated according to the SGD algorithm. At the end of the training, this procedure returns a control law and a valid CLF over the *finite* sample set, i.e. $V(\boldsymbol{s_i}) > 0$, $\dot{V}(\boldsymbol{s_i}, \boldsymbol{u_i}) < 0$, $\forall\, \boldsymbol{s_i} \in S$ and $\boldsymbol{u_i}$ the sample control set.

b) Falsifier: Given a candidate CLF $V(\boldsymbol{x})$ and its corresponding Lie derivative $\dot{V}(\boldsymbol{x}, \boldsymbol{u}) = \nabla V(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$, our procedure ought to prove its negative definiteness $\forall x \in \mathcal{D} \setminus \{\boldsymbol{x}^\star\}$ via SMT solving. If the SMT verifies that the candidate CLF is indeed valid, the 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. 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 counterexample might be returned in the neighborhood of the origin (within a precision δ). Therefore, we exclude a small neighborhood of the origin 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 $\|\boldsymbol{x}\| \leq \epsilon$ [17]. The latter

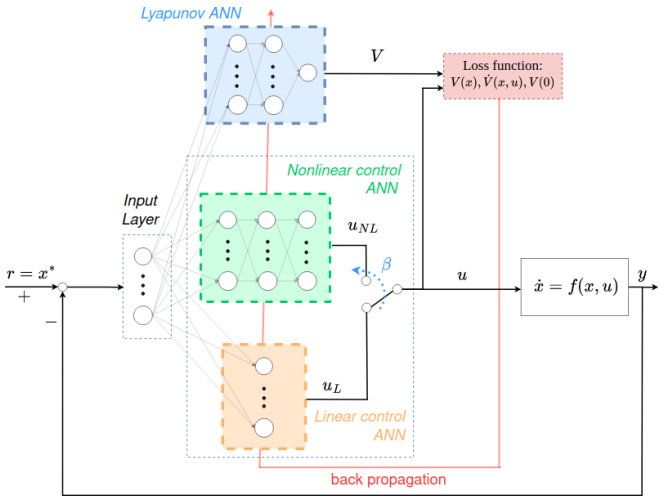

Fig. 1. Augmented Neural Lyapunov Control architecture with Lyapunov ANN (blue box), nonlinear and linear control ANN (green and orange boxes, respectively) and β the control-branch training selector.

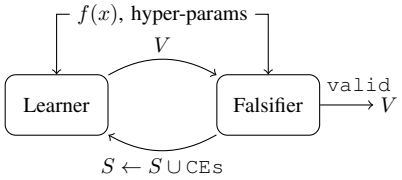

Fig. 2. Learner-Falsifier CEGIS loop: the Learner trains the ANNs, while the Falsifier verifies the formal validity of the CLF V.

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 [18]. Additionally, the architecture relies on an *Augmented Falsifier* module, that was shown speeding up the CE generation whilst mitigating issues linked to dataset overfitting [13].

III. Synthesis of Passive FTC laws

The ANLC CEGIS-based procedure can be extended to guarantee FTC properties by devising tailored modification to both the *Learner* and the *Falsifier*.

Assumptions. We leverage two assumptions regarding the system and fault behaviour: i) the actuators exhibit binary faults, i.e. they are either fully working or fully broken; ii) at most only one fault is present at each time. Condition i) can be assumed as, in the case of underwater actuators, a thruster can become mechanically stuck when solid objects (such as debris or ice) fall between the blades and the propeller, or a thruster can be switched off by the onboard computer when any problem is detected (such as when anomalous currents are drawn). Finally, a situation where multiple faults happen at the same time is 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. We assume that each 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 j-th 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

$$(\dot{V}_n(\boldsymbol{x},\boldsymbol{u}) < 0) \wedge (\forall j \in \Phi : \{\dot{V}_{\phi_j}(\boldsymbol{x},\boldsymbol{u}) < 0\}), \quad (2)$$

where $\dot{V}_i(\boldsymbol{x},\boldsymbol{u}) = \boldsymbol{\nabla V}(\boldsymbol{x}) \cdot \boldsymbol{f_i}(\boldsymbol{x},\boldsymbol{u})$, for $i = n$ (nominal dynamics) or $i \in \Phi$.

Let us now set up the training in order to synthesise a CLF abiding (2).

a) FTC Learner: Recalling the structure of the loss function proposed in [13], referred to as *Empirical Lyapunov Risk Loss* (L_{ELR}), four terms are defined: three are linked to the theoretical Lyapunov conditions (α_i), whilst one regulates the size of the Region Of Attraction (ROA) (α_{ROA}). The loss function is defined as

$$L_{ELR} = \alpha_1 \sum_{i=1}^{N} \max(0, -V(\boldsymbol{s_i})) + \alpha_2 L_{\dot{V}} + \alpha_3 V(0)^2 + \alpha_4 \frac{1}{N} \sum_{i=1}^{N} (\|\boldsymbol{s_i}\|_2 - \alpha_{ROA}\ V(\boldsymbol{s_i}))^2, \quad (3)$$

where N is the cardinality of the training dataset and where the tuning coefficients can be selected as discussed in [13]. The first term of (3) enforces the positiveness of V, the second one is responsible for the negativeness of the Lie derivatives $\dot{V}_i$, the third one represents the condition $V(0) = 0$, whilst the the latter constraint fosters circular level sets of the CLF. Note however that the *shape* of the CLF is accounted for as a side feature, as we do not strictly enforce the loss L_{ELR} to be equal to 0 [13]. The loss term associated with the Lie derivative ($L_{\dot{V}}$), captures both nominal and faulty dynamics as

$$L_{\dot{V}} = \sum_{i=1}^{N} \max(0, \boldsymbol{\nabla V}(\boldsymbol{s_i}) \cdot \boldsymbol{f_n}(\boldsymbol{s_i}, \boldsymbol{u_i})) + \sum_{j=1}^{p} \sum_{i=1}^{N} \max(0, \boldsymbol{\nabla V}(\boldsymbol{s_i}) \cdot \boldsymbol{f_{\phi_j}}(\boldsymbol{s_i}, \boldsymbol{u_i})). \quad (4)$$

b) FTC Falsifier: Once the training finds a suitable candidate Lyapunov function, this is checked by the verification engine, in order to formally certify that the Lyapunov conditions are satisfied over the whole continuous domain $\mathcal{D}$ – or, if these are not satisfied, a counterexample is returned and the training is restarted with an augmented dataset. The verification check is expressed with the formula

$$\forall \boldsymbol{x} : (\boldsymbol{x} \in \mathcal{D} \Rightarrow \Big(V(\boldsymbol{x^\star}) = 0 \wedge V(\boldsymbol{x}) > 0 \wedge \dot{V}_n(\boldsymbol{x},\boldsymbol{u}) < 0 \wedge \{\dot{V}_{\phi_j}(\boldsymbol{x},\boldsymbol{u}) < 0\}_{j=1}^{p}\Big), \quad (5)$$

where the latter term denotes the sequence of the p Lie derivatives associated to the faulty systems. Finally, these conditions guarantee the system stability even when an actuator within the set Φ fails. The proposed method is hereby referred to as *passive Fault Tolerant-Augmented Neural Lyapunov Control* (pFT-ANLC).

IV. Control of an inverted pendulum with actuator redundancy

Our method is compared against two LQR laws and a general not fault tolerant ANLC (*vanilla* ANLC or vANLC) [13]), in order to show the capabilities of a passive FTC, for an inverted pendulum with redundant actuators model.

The nominal dynamics $\boldsymbol{f_n}$ of an inverted pendulum with a redundant actuator set can be written as

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{(mgL\sin x_1 - bx_2 + h_1 u_1 - h_2 u_2)}{(mL^2)}, \end{cases} \quad (6)$$

where x_1 and x_2 represent the pendulum position and velocity, respectively; u_1 and u_2 represent the torque generated

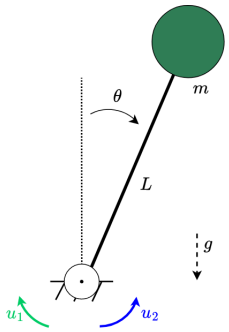

Fig. 3. Inverted pendulum with redundant actuator set.

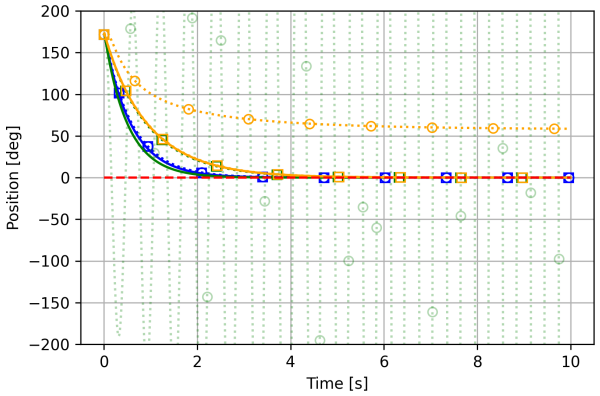

Fig. 4. Inverted pendulum closed-loop tests. Color code: blue lines (pFT-ANLC), orange lines (LRQ_2), green lines (vANLC). Line style: solid (nominal dynamics), dashed with square markers (fault 1), dashed with round markers (fault 2).

by two separate motors; m, L the pendulum mass and arm length; g the gravity constant and b the drag coefficient. Also, h_i denotes the *health status* of the i-th actuator, namely $h_i = 1$ when the actuator is functioning nominally, and $h_i = 0$ when faulty. It follows that in the nominal scenario $h_1 = h_2 = 1$. A schematic depiction is illustrated in Fig. 3.

LQR control: The LQR method is chosen due to its intrinsic robustness properties. First, we linearize the model (6) around the target unstable equilibrium point. Two LQR laws are tuned based on different weights assigned to the actuators (denoted LQR_1 and LQR_2). In the first case, we consider u_1 and u_2 equally important; in the second case, u_2 is the preferred control source.

Augmented Neural Lyapunov Control: Following, a linear control law is synthesised with the vANLC. To resemble the case of actuators with different weights, the loss function (3) is modified as

$$L_{vANLC} = L_{ELR} + \alpha_{u_1} \sum_{i=1}^{N} u_{1,i}^2 + \alpha_{u_2} \sum_{i=1}^{N} u_{2,i}^2, \quad (7)$$

where the extra two terms add, as further objective, the minimisation of the energy spent by the actuators. By tuning α_{u_1} and α_{u_2} the effort of the two actuators can be penalised differently. As we compare against LQR controllers, we select the linear control architecture (see Section II). We set $\alpha_{u_1} = 0.7$, $\alpha_{u_2} = 0$ for this training scenario.

Passive Fault Tolerant ANLC: The pFT-ANLC encompasses, besides the nominal model, both faulty systems, namely derived from (6) as $f_{\phi_1}(h_1 = 0, h_2 = 1)$ and $f_{\phi_2}(h_1 = 1, h_2 = 0)$.

We run 10 tests with the same hyperparameters (with the exception of the seed), and we select the same learning parameters shared with the vANLC, including the choice of linear control laws. Each of the 7 converged training runs stems into a controller with slightly different performance, such as settling time and steady-state error, but all are certified to stabilise the closed-loop. In real applications, control engineers may select one of these controllers according to their preferred performance criteria. The definition of such an indicator is beyond the scope of this work and will be object of future investigation.

Control laws comparison: Our analysis includes the nominal case along with two faulty scenarios, for the four control schemes, amounting to 12 different case studies. We report the closed-loop eigenvalues in Table I.

TABLE I

REDUNDANT INVERTED PENDULUM: CLOSED-LOOP POLES LOCATION.

Controller	Dynamics		
	Nominal (f_n)	**Fault 1 (f_{ϕ_1})**	**Fault 2 (f_{ϕ_2})**
LQR_1	-1.29, -22.06	-0.37, -12.92	-0.37, -12.92
LQR_2	-1.01, -160.39	-0.99, -145.98	0.18, -17.28
vANLC	-1.91, -128.18	-1.02, -132.29	$0.28 \pm 9.50j$
pFT-ANLC	-1.68, -208.76	-1.59, -101.87	-1.58, -108.08

Every control scheme guarantees the stability of the nominal and of the first faulty scenario, but not when u_2, the main actuator, is faulty. When the first fault occurs, notice that the dominant poles shift towards the imaginary axis, hinting towards a reduced stability margin. When the fault is injected on the second motor instead, two of the closed-loop systems become unstable, while both the first LQR and our pFT methods can retain stability. The first LQR is computed with much stricter constraints with respect to the second, and it provides stability margins considerably worse than our method under all scenarios.

The closed-loop dynamics of the LQR_2, vANLC, and pFT-ANLC are illustrated in Fig. 4 (LQR_1 is not shown for readability, but the dynamics qualitatively resemble the pFT-ANLC ones). All the three control laws can stabilise the pendulum in the nominal case (solid lines) and first faulty model (square markers). When a fault on the second motor occurs, only the pFT-ANLC can stabilise the system; both the green and orange lines with round markers do not converge to the desired equilibrium point. The corresponding graphical animations of the pendulum motion with the three control laws are provided[1]. This analysis illustrates qualitatively how state-feedback control laws cannot always guarantee stability when the underlying dynamics undergoes a fault, such as the loss of one *redundant* actuator, whereas our method provides sufficient stability margin under all fault scenarios.

[1]https://github.com/grande-dev/pFT-ANLC-preview

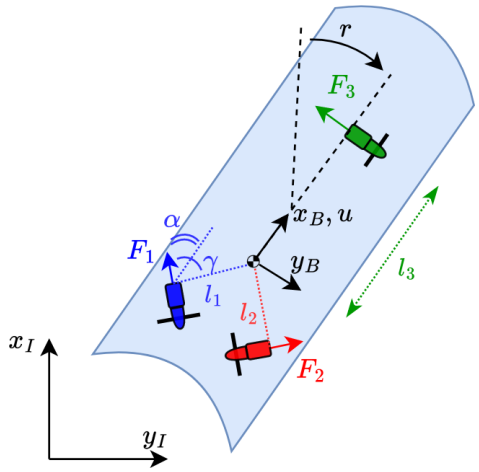

Fig. 5. AUV vehicle model with three fixed thrusters.

V. 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 [12], [13], 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 Southampthon[2], UK. A two dimensional dynamics accounting for surge speed (x_1) and angular velocity around the vertical axis (x_2), is used to describe the planar motion of the AUV. Thrusters F_1 and F_2 are oriented at an angle α wrt the x-body axis (x_B) and form an angle γ with the segment connecting them to the centre of gravity (l_i). By denoting with m the mass of the vehicle and with J_z the moment of inertia around the vertical axis, the AUV dynamics can be described as

$$\begin{cases} \dot{x}_1 = \dfrac{-X_u x_1 + F_1 \cos\alpha + F_2 \cos\alpha}{m} \\ \dot{x}_2 = \dfrac{-N_r x_2 + F_1 l_1 \sin\gamma - F_2 l_2 \sin\gamma - F_3 l_3}{J_z}, \end{cases} \tag{8}$$

where F_i represents the force generated by the i-th thruster, X_u and N_r denote the surge and yaw drag coefficients, respectively. For slow moving vehicles, simple linear drag terms are sufficiently descriptive and no Coriolis-related effects appear due to the sway dynamics being neglected.

The proposed case study investigates the occurrence of two possible faults, namely the fault on the first (aft port) and third (bow) thruster. 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 we employ a nonlinear control law. The selected ANN architecture is reported in Table II, where we outline the input, hidden, and output layers' sizes, along with the presence of the bias and the activation functions σ.

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

TABLE II
AUV campaign: ANN architecture.

Parameter	Lyapunov NN	Control NN
layer size	[2, 10, 10, 1]	[2, 10, 10, 3]
bias	[No, No, No]	[Yes, Yes, No]
σ	[x^2, linear, linear]	[softplus, softplus, softplus]

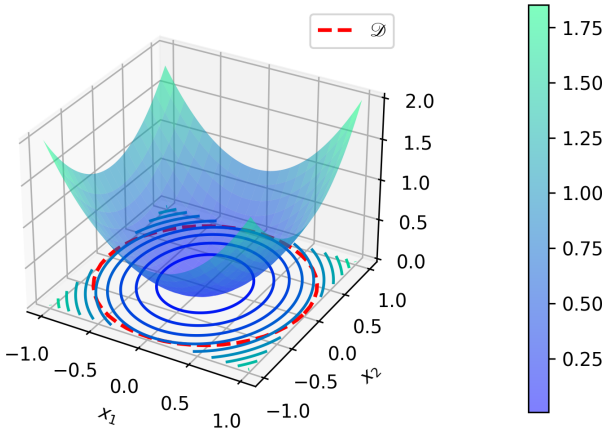

Fig. 6. Synthesised CLF for the AUV system.

The training is carried out on an unassuming office laptop without GPU (8 CPUs at 1.90GHz). 10 tests (with different seeds) are run, of which 9 converge within an average time of about 4 min. One resulting CLF, obtained after 883 training iterations is reported in Fig. 6.

To conclude, closed-loop results are hereby illustrated and discussed. Fig. 7 reports the AUV surge dynamics, shown as per bounds spanned by the 9 controllers with stability certificate. 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 behavior is achieved as the pFT-ANLC learns *automatically* to set the steady-state target higher to compensate for the possible faults. This is in turn accomplished by applying an excess of force on F_2 wrt to F_1 and a non-zero F_3, as illustrated in Fig. 8. An analogous behaviour is noticed in the angular rate dynamics, reported in Fig.9, as the controllers learn to converge to an offset value wrt $x_2^\star$ to mitigate the possible occurrence of a fault.

VI. 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 unique control function that guarantees ϵ-stability of the target equilibrium. Both linear and nonlinear control laws are synthesised for two benchmark systems of increasing complexity. The approach is shown capable of maintaining the system dynamics within the target stability boundaries, without relying on external information flow from a Fault Detection and Isolation block. The architecture requires minimal hyperparameter tuning and can be run on standard

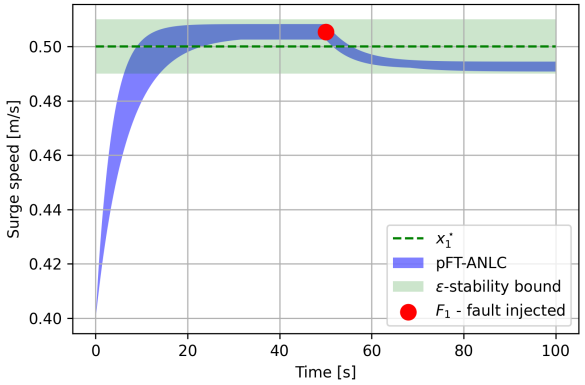

Fig. 7. Closed-loop test AUV system: surge dynamics — F_1 faulty.

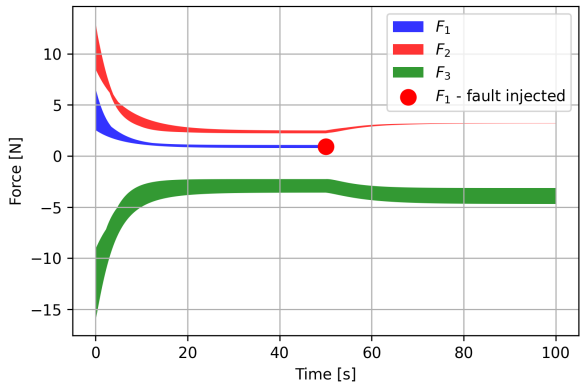

Fig. 8. Closed-loop test AUV system: control effort allocation — F_1 faulty.

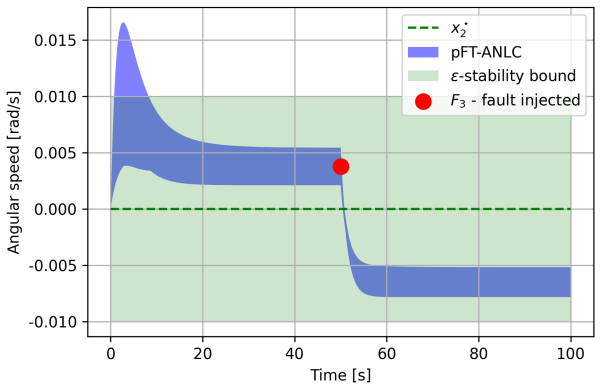

Fig. 9. Closed-loop test AUV system: angular rate dynamics — F_3 faulty.

office laptops. The approach is of relevance for every control application where extensive sensoring or fault monitoring algorithms cannot be assumed, such as in the case of autonomous underwater vehicles. Despite the benefits mentioned, limitations encompassing the scalability to higher order systems and the introduction of noisy measurements and disturbances will require dedicated assessment. A thorough comparison against additional state-of-the-art fault tolerant methods, is matter of ongoing work. Future work focusses both on testing the control method onto real vehicles, on the definition of a criterion to select one of the synthesised controllers, and on comparing the performance with diverse Fault Tolerant Control techniques.

References

[1] G. J. Ducard, *Fault-tolerant flight control and guidance systems: Practical methods for small unmanned aerial vehicles*. Springer Science & Business Media, 2009.

[2] K. Patan, "Robust and fault-tolerant control: neural-network-based solutions," 2019.

[3] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[4] M. Verhaegen, S. Kanev, R. Hallouzi, C. Jones, J. Maciejowski, and H. Smail, *Fault Tolerant Flight Control - A Survey*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 47–89.

[5] M. Blanke, M. Kinnaert, J. Lunze, M. Staroswiecki, and J. Schröder, *Diagnosis and fault-tolerant control*. Springer, 2006, vol. 2.

[6] M. Benosman and K.-Y. Lum, "Passive actuators' fault-tolerant control for affine nonlinear systems," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 152–163, 2009.

[7] M. Katebi and M. J. Grimble, "Integrated control, guidance and diagnosis for reconfigurable autonomous underwater vehicle control," *International Journal of Systems Science*, vol. 30, no. 9, pp. 1021–1032, 1999.

[8] P. N. N. Thanh and H. P. H. Anh, "Advanced neural control technique for autonomous underwater vehicles using modified integral barrier Lyapunov function," *Ocean Engineering*, vol. 266, p. 112842, 2022.

[9] E. Anderlini, G. G. Parker, and G. Thomas, "Docking control of an autonomous underwater vehicle using reinforcement learning," *Applied Sciences*, vol. 9, no. 17, p. 3456, 2019.

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

[11] A. R. Dooraki and D.-J. Lee, "Reinforcement learning based flight controller capable of controlling a quadcopter with four, three and two working motors," in *2020 20th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2020, pp. 161–166.

[12] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," *Advances in neural information processing systems*, vol. 32, 2019.

[13] D. Grande, A. Peruffo, E. Anderlini, and G. Salavasidis, "Augmented Neural Lyapunov Control," *IEEE Access*, vol. 11, pp. 67 979–67 986, 2023.

[14] A. Abate, D. Ahmed, M. Giacobbe, and A. Peruffo, "Formal synthesis of Lyapunov neural networks," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 773–778, 2020.

[15] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks," in *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, 2021, pp. 1–11.

[16] S. Gao, S. Kong, and E. M. Clarke, "dReal: An SMT solver for nonlinear theories over the reals," in *International conference on automated deduction*. Springer, 2013, pp. 208–214.

[17] S. Gao, J. Kapinski, J. Deshmukh, N. Roohi, A. Solar-Lezama, N. Aréchiga, and S. Kong, "Numerically-robust inductive proof rules for continuous dynamical systems," in *Computer Aided Verification: 31st International Conference*. Springer, 2019, pp. 137–154.

[18] J. La Salle and S. Lefschetz, "Stability by Liapunov's Direct Method with Applications," *Academic Press, RIAS*, 1961.