Evolutionary Reinforcement Learning: A Hybrid Approach for Safety-informed Intelligent Fault-tolerant Flight Control

Gavra, V.; van Kampen, E.

DOI
10.2514/6.2024-0954

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the AIAA SCITECH 2024 Forum

AIAA SciTech Forum
8-12 January 2024, Orlando, FL
AIAA SCITECH 2024 Forum

10.2514/6.2024-0954

Evolutionary Reinforcement Learning: A Hybrid Approach for Safety-informed Intelligent Fault-tolerant Flight Control

V. Gavra[*] and E. van Kampen[†]
Delft University of Technology, P.O. Box 5058, 2600GB Delft, The Netherlands

Recent research in artificial intelligence potentially provides solutions to the challenging problem of fault-tolerant and robust flight control. The current work proposes a novel Safety-informed Evolutionary Reinforcement Learning (SERL) algorithm, which combines Deep Reinforcement Learning (DRL) and neuro-evolution to optimize a population of non-linear control policies. Using SERL, the work has trained agents to provide attitude tracking on a high-fidelity non-linear fixed-wing aircraft model. Compared to a state-of-the-art DRL solution, SERL achieves better tracking performance in nine out of ten cases, remaining robust against faults and changes in flight conditions, while providing smoother actions.

Keywords: **Fault-tolerant Flight Control, Deep Reinforcement Learning, Evolutionary Algorithms, Intelligent Control, Control Policy Noise.**

Nomenclature

B	=	Batch of transition tuples
F	=	Fitness function value
H, V_{tas}	=	Altitude, true airspeed [m/s]
N	=	Population size
Q_ϕ, μ_θ	=	Critic function parameterised by ϕ, deterministic actor parameterised by θ
Sm	=	Smoothness of the action [rad · Hz]
T, f_s	=	Duration of simulation trail [s], sampling frequency [Hz]
p, q, r	=	Roll, pitch, yaw rate [rad/s]
$\tilde{r}, R$	=	Reward, future return value
$\mathbf{a}, \mathbf{s}, \mathbf{x}$	=	Vectors for the agent's action, state, and aircraft state
$\alpha, \beta, \phi, \theta, \psi$	=	Angle of attack, sideslip, roll, pitch and yaw [rad]
$\delta_e, \delta_a, \delta_r$	=	Elevator, aileron, rudder deflection [rad]
$\alpha_{\text{a.,c.}}, \gamma$	=	Learning rate of actor, critic learning, temporal discount rate of future rewards
ϵ	=	Variable with Gaussian distribution
κ	=	Genetic buffer size
λ	=	Regularization coefficient
σ	=	Standard deviation of a Gaussian distribution
$\mathcal{L}$	=	Loss function

Sets and tuples

[*]MSc, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.
[†]Assistant Professor, Faculty of Aerospace Engineering, Control and Simulation Division, Delft University of Technology.

$\mathcal{A}, \mathcal{S}$	=	Sets of all possible actions, states
$\mathcal{B}$	=	Memory buffer of previous experiences
$\mathcal{D}_x$	=	Domain in which variable x resides
$\mathcal{T}_t$	=	Transition tuple experienced by the agent at time t

I. Introduction

Aviation has experienced an excellent safety record in recent decades, its ever-lowering rate of accidents making it the safest means of long-distance transportation in terms of the number of casualties. Despite this, the number of accidents involving "loss of control in flight" is not decreasing at a comparable rate [1]. Improving the autonomy capabilities of aerospace systems in general and of aircraft controllers, in particular, could further reduce the number of air crashes.

Modern techniques in intelligent control incorporate bio-inspired Artificial Intelligence (AI) could enhance the safety, durability, autonomy, and performance of aerospace vehicles. Ultimately, a milestone in achieving these goals concerns improving the way simulation-based flight control systems could remain robust to real-life conditions such as faults, unmodeled dynamics, and external disturbances [2–4].

Reinforcement Learning (RL) uses repeated rewarded interactions between an intelligent agent and its environment to optimize its control policy [5]. Its extension, Deep Reinforcement Learning (DRL) harvests the potential of deep neural networks (NNs) as data-driven models that approximate and control non-linear dynamics [6, 7]. These algorithms belong to the class gradient-based optimization as the neural networks are trained via gradient descent [6].

Recently, DRL has shown above-human performance in complex tasks such as strategy games [8, 9]. Applied to flying systems, it has obtained state-of-the-art performance in tasks such as guidance [10], low-level attitude control in the presence of faults [11] or disturbances [12], identification of worst-case maneuvers causing aircraft departures [13] and agile drone racing [14]. Offline model-free DRL methods prioritize performance which can lead to policies commanding actions with unconstrained noise. This aggravates the gap between simulation and reality, increasing the already-challenging deployment on flight hardware [15]. Whereas regularization methods are developed to inhibit the action noise [14, 15], the root cause of the noise has yet to be extensively discussed.

Evolutionary Algorithms (EAs) are meta-heuristics that optimize a repertoire of control strategies, or population, by selecting the best-performing individuals in each generation, combining and varying them [16, 17]. Genetic Algorithms (GAs), the earliest, most popular, and versatile EAs mimic natural selection to solve optimization problems [18]. Neural evolution (NE) specifically trains non-linear mappings parameterized as neural networks by using an EA to iteratively update their weights. In optimal control, NE is an episode-based alternative to DRL [19, 20].

When evolving randomly initialized non-linear policies, the intrinsic difference in the parameters of individuals translates into novel behaviors [21]. Throughout this work, novelty always describes the difference in the current individual's behavior (i.e., set of actions) with respect to the previous one or in comparison to the behaviors of other actors. Qualitatively, *diversity* denotes a population with an arbitrarily large number of novel individuals.

The achievable action-space diversity makes NE attractive for high-dimensional non-linear control tasks, such as fault-tolerant legged robots [21, 22] and robot-arm manipulation [23]. Within flight control, GAs have been tasked to derive flight control laws [24] or schedule gains linear controllers such as Proportional Integral Derivative (PID) [25] but, until now, the applications of NE are limited as reviewed by the authors of [26].

Despite their achievements, both the aforementioned bio-inspired frameworks have their drawbacks. Hybrid algorithms, incorporating evolutionary loops and reinforcement learning updates, show more stable learning and increased performance in complex learning environments [27–29]. Evolutionary Reinforcement

Learning (ERL), proposed by the authors of [30], trains a population of control policies in a GA loop, periodically infusing it with policies optimized by a DDPG actor-critic learning from the population's shared memories. Whereas the algorithm aims at the best of both worlds, the average population performance remains unstable due to spontaneous catastrophic forgetting and is sensitive to the chosen hyperparameters [27, 29]. Frameworks such as Proximal Distilled ERL (PDERL) incorporate numerically stable policy mutations [27, 31] to counteract catastrophic forgetting and distillation crossover to enhance sample-efficiency [29], thus achieving state-of-the-art performance in continuous control [27]. Lastly, the adaptation to flight control tasks is facilitated by its modular structure.

This paper has a threefold contribution. First, we develop a novel fixed-wing aircraft attitude controller by extending a state-of-the-art ERL algorithm. Second, the paper shows that optimizing a population of controllers via DRL and GAs provides significantly better fault tolerance and robustness to unseen flight conditions compared to a state-of-the-art DRL-only approach. Lastly, we demonstrate that the evolutionary mechanisms can remove the root causes of the noise phenomena and thus hybrid frameworks can balance the controller performance and smoothness [*].

The paper outlines in Sec. II the theoretical background of ERL algorithms and then describes in Sec. III the design of SERL as a flight controller. Sec. IV compares the performance of the trained agents and discusses the effect of the SERL mechanism on the control policy smoothness. Finally, the article concludes by summarizing the achievements in Sec. V.

II. Background

This section introduces the mathematical formalism of ERL by summarizing the actor-critic RL, population-based policy search concepts, and the hybrid algorithm combining the two. Additionally, it presents the attitude control task converted to a learning problem suitable for the ERL framework.

A. Deep Reinforcement Learning

Reinforcement Learning is a bio-inspired framework that uses repeated rewarded interactions between an intelligent agent and its environment to learn an optimal control policy. The sequential decisions made by the agent within the environment are modeled as a Markov Decision Process (MDP) formalized by $\mathcal{P}\left\{\mathbf{s}_{t+1}, \tilde{r}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t, \ldots, \mathbf{s}_0, \mathbf{a}_0\right\} = \mathcal{P}\left\{\mathbf{s}_{t+1}, \tilde{r}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right\}$, with state space $\mathcal{S} \subset \mathbb{R}^n$, action space $\mathcal{A} \subset \mathbb{R}^m$, reward signal $\tilde{r} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and probability distribution of the stochastic state transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. The MDP, therefore, assumes collapsed state-action history as the current state and action solely determines the next state and received reward [5, 32].

The policy of a deterministic agent commands at each time step an action $\mathbf{a}_t \in \mathcal{A}$ according to a deterministic function of its state $\mu(\mathbf{s}_t)$ and retrieves from its observation the transition tuple $\mathcal{T}_t = \langle \mathbf{s}_t, \mathbf{a}_t, \tilde{r}_t, \mathbf{s}_{t+1} \rangle$. Off-policy learning methods employ a form of *memory buffer* $\mathcal{B}$ to store the transition data for subsequent updates of the agent. In episodic tasks, the sequence of transitions continues until the agent has reached a terminal state or a maximum number of steps. The *return*, defined as $R_t = \lim_{n\to\infty} \sum_{k=0}^{n} \gamma^k \tilde{r}_{t+k}$, represents the accumulated future reward expected from time t and discounted by $\gamma \in (0, 1]$. The action-value function $Q(\mathbf{s}, \mathbf{a})$ predicts the value of the return from the given state $\mathbf{s}$ when the agent selects action $\mathbf{a}$ thereafter following policy μ [5].

Actor-critic methods combine policy and value function learning [5]. Whereas the first category frames the goal of maximizing return as the minimization of a control policy loss function $\mathcal{L}(\theta)$, the value-based frameworks approximate the state-action value function as $Q(\mathbf{s}, \mathbf{a}) \approx Q_\phi(\mathbf{s}, \mathbf{a})$ with $\phi \in \mathcal{D}_\phi$, iteratively updating it. The parametric policy $\mu_\theta(\mathbf{s})$ (with $\theta \in \mathcal{D}_\theta$) subsequently trains on the estimated Q-values, moving towards the optimal function $\mu^*(\mathbf{s})$.

[*]The source code used for training and evaluation is available at `https://github.com/VladGavra98/SERL`

A state-of-the-art method, Deep Deterministic Policy Gradient employs NNs as function approximators for both the critic and deterministic actor due to their intrinsic ability to learn complex non-linear dynamics [6]. The Twin-Delayed DDPG (TD3), developed by Fujimoto et al. [33] improves the sample complexity and learning stability of DDPG in offline continuous control tasks [34]. Adding to its relatively uncomplicated structure, these made TD3 the candidate for the DRL part of the SERL flight control framework.

During training, the networks are updated by randomly sampling batches B from the buffer $\mathcal{B}$. The critic part trains using recursive temporal difference updates, minimizing the Mean Squared Bellman error (MSBE) from Eq. (1). TD3 learns two Q-functions (hence its name) and it takes the minimum of the two Q-values as targets in the MSBE loss - the *dual critics* trick. By taking the less optimistic Q-value as the optimization target, TD3 is less prone to the overestimation bias of Q-learning [35] and thus has stable learning [33].

$$\mathcal{L}_Q(\phi, B) := \frac{1}{|B|} \sum_{\mathcal{T}_t \in B} \left[Q_\phi(\mathbf{s}_t, \mathbf{a}_t) - \left(\tilde{r}_t + \gamma \min_{i=1,2} Q_{\phi_{i,\text{targ}}} \left(\mathbf{s}_t, \mathbf{a}_{\text{targ}_t}\right) \right) \right]^2, \quad \phi \in \mathcal{D}_\phi, \theta \in \mathcal{D}_\theta \tag{1}$$

Concurrently, the actor learning translates into the minimization of policy loss, defined by Eq. (2) as the negative of the value estimate over one batch of states.

$$\mathcal{L}_\mu(\theta, B) := -\frac{1}{|B|} \sum_{\mathbf{s}_t \in B} \min_{i=1,2} Q_{\phi,i}\left(\mathbf{s}_t, \mu_\theta(\mathbf{s}_t)\right), \quad \theta \in \mathcal{D}_\theta \tag{2}$$

The target networks are separate copies of the actor and critic networks, parametrized within the same $\mathcal{D}_\phi, \mathcal{D}_\theta$ spaces. To ensure stability during learning, they are synchronized using a Polyak moving average of the current network's parameters [7, 36].

Sustained exploration within the state-action space is a necessary condition for convergence toward the optimal policy. The TD3 agent explores during training by corrupting the actor's actions with off-policy noise sampled from an unbiased Gaussian distribution [7], as shown by Eq. (3). Additionally, to train the target critic, TD3 clips the noise added to the action of the target policy as a form of regularisation. For both the behavioral policy and the target one, the resulting actions are clipped to lie within the interval admissible by the environment $\left[\mathbf{a}_{\text{low}}, \mathbf{a}_{\text{high}}\right]$.

$$\mathbf{a}_t = \text{clip}\left(\mu_\theta(\mathbf{s}_t) + \boldsymbol{\epsilon}, \mathbf{a}_{\text{low}}, \mathbf{a}_{\text{high}}\right), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma I) \tag{3}$$

Lastly, the target policy network updates less frequently than the target critic network. Adding to the smoothening effect of the Polyak average, delaying the update of the target policy further reduces the risk of divergence.

DRL in general optimizes the networks via a form of stochastic gradient descent (SGD), which requires the backpropagation of the loss functions' gradients with respect to the networks' parameters [6]. By doing so, the batch SGD updates drive sample-efficient experience-based training. Despite this, local optima and saddle points in the policy loss landscape alongside noisy or sparse gradient estimates impede long-term exploration and therefore hinder learning [37].

B. Evolutionary Algorithms

EAs are another bio-inspired alternative which, as a black-box local-search technique, iteratively improve a *population* of control policies or individuals targeting a higher *fitness* value [38].

EAs are characterized by a standard underlying architecture: evaluate the episodic performance of each individual, select the best-behaving individuals (fittest policies), and apply variational operators. A sub-class, Genetic Algorithms (GAs) are derivative-free optimization algorithms that aim to deliver a sufficiently accurate solution within a relatively low computational time [39]. Having a modular structure, traditional GAs explore the solution space via two separate means: *crossover* and *mutation*. Crossover combines a fraction of elites and mutation alters the non-elites, both generating new offsprings to replace the least fit individuals.

The selection operation is probabilistic according to the tournament selection method [40]. Solutions with higher fitness values have a higher probability of being selected. One assumes higher fitness values to represent better solution quality. With each generation, the average quality of the population improves, as well as the reward of the best actor (the *champion*).

C. Hybrid Frameworks

Learning emerges from the trade-off between exploitation or quality-seeking and exploration, or novelty-seeking [21, 41]. ERL, introduced by [30], is a class of hybrid methods that implements this trade-off as a combination of off-policy DRL and GA. It evolves a population of N control policies, applying selective pressure based on episodic fitness defined according to Eq. (4).

First, the GA part generates novel individuals, or *offsprings* through crossover and Gaussian mutation. They both act on the *genome*, the NN weights tensor flattened to a one-dimensional sequence. The standard Gaussian mutation, showed by Eq. (5) samples the offspring policy weights θ_o from a Gaussian distribution centered at the parent's weights θ_p. Crossover combines the genomes of two *elites* (i.e., policies from the first e fitness percentile). Sec. III will describe how SERL performs crossover operator and safety-inform mutation.

$$F_i := \sum_{t=0}^{T} \tilde{r}_t, \quad i = \overline{1, N} \tag{4}$$

$$\theta_o \sim \mathcal{N}\left(\theta_p, \sigma_{mut} I\right) \tag{5}$$

Second, parallel to the genetic population, an actor-critic agent based on DDPG learns via gradient updates by randomly sampling batches from the common memory buffer. Training is performed off-policy, as the actor and critic are updated with information gathered by all the other policies. At a given number of generations, the RL actor is injected into the population by cloning its weights in place of the least-fit genetic actor. Thus, ERL establishes a bidirectional transfer of information [27, 30]. Whereas the shared memory buffer enables off-policy exploration, the actor synchronization aims to decrease the sample complexity of the neuro-evolution as the newly injected NN has been updated more often via gradient-based optimization [29, 30].

The hybrid agent sequentially evaluates all actors and stores the experienced transition tuples in a *shared memory buffer* $\mathcal{B}^{(N)} := \left\{\mathcal{T}_t^{(i)} \mid i = \overline{1, N}\right\}$. The superscript (N) refers to the size of the genetic population and will be later dropped for brevity. Thus, the ERL agent's shared buffer contains unordered exploratory traces of both GA and DRL loops.

Whereas the DDPG actor uses zero-mean additive Gaussian noise to explore in the $\mathcal{S} \times \mathcal{A}$ space, ERL combines this with indirect time-independent exploration in the policy parameter space $\mathcal{D}_\theta$ [37, 42]. Crucially, since the distribution of the weight-magnitude is not uniform across the network, the output of each layer is normalized to more evenly scale the effect of the applied perturbations [42].

Proximal Distilled Evolutionary Reinforcement Learning (PDERL) developed by Bodnar et. al. [27] uses the same structure as ERL and a DDPG agent, but it proposes two novel genetic operators, distillation crossover and proximal mutation which sample experiences from individual buffers of the genetic actors [27]. They aim to counteract catastrophic forgetting caused by Gaussian mutation and K-point crossover applied to a direct encoding of the network, a problem of the off-policy hybrid methods [27, 29, 31].

1. Genetic memory buffer

To integrate both operators, PDERL uses a set of memory buffers with each actor possessing *genetic memory* $\mathcal{B}^G$. The individual buffers store the κ most recent experiences of each actor. Moreover, each personal buffer can store experiences that span multiple generations which will be spread through the population by the variational operators. The common buffer $\mathcal{B}$ shares experience with the individual genetic memories but it does not equal their union.

2. Q-filtered distillation crossover

The operator selectively merges the behaviors of two elite policies based on their estimated Q-values such that offsprings inherit optimized behaviors from their parents and not just the weights of the networks [27].

Consider two distinct parent policies μ_x and μ_y. An offspring policy μ_o is created and its buffer $\mathcal{B}_o$ is filled with $\kappa/2$ experiences from both parents. Its parameters θ_o are initialized based on one of the parents. Then, the offspring actor is optimized to selectively imitate (or distill) its parents' actions for the sampled states.

Genetic crossover combines two networks instead of one. This introduces the problematic possibility of the offspring's behavior diverging from both parents. To account for it, PDERL introduces a cloning loss $\mathcal{L}^{(C)}$ which trains the offspring μ_o to minimize the deviation from the parents' behaviors over a batch B:

$$\begin{aligned} \mathcal{L}^{(C)}_{\mu}\left(\theta_o, B_C\right) &:= \frac{1}{|B_C|}\sum_{i}^{B_C} \|\mu_o\left(\mathbf{s}_i\right)\|^2 + \left\|\mu_o\left(\mathbf{s}_i\right) - \mu_p\left(\mathbf{s}_i\right)\right\|^2; \\ p &= \begin{cases} x & Q\left(\mathbf{s}_i, \mu_x\left(\mathbf{s}_i\right)\right) > Q\left(\mathbf{s}_i, \mu_y\left(\mathbf{s}_i\right)\right) \\ y & else \end{cases} \quad \text{and} \quad B_C \sim \mathcal{B}^o \end{aligned} \tag{6}$$

The operator $\mathbb{I}$ selects the policy with a higher Q-value, biasing the offspring towards the estimated best-acting parent. The last term of Eq. (6) performs L_2-norm regularisation to optimize for less aggressive behaviors (i.e., actions changing rapidly for marginally different states) [27].

Lastly, the pairing metric from Eq. (7) decides which elites are going to be crossed. It uses the Euclidean distance between the average actions taken by the agents [27]. It samples a batch of states from each parent genetic memory with $\mathcal{P}_{x,y}$ the corresponding state-visitation probability.

$$d\left(\mu_x, \mu_y\right) := \mathbb{E}_{\mathbf{s}\sim\mathcal{P}_x}\left[\left\|\mu_x(\mathbf{s}) - \mu_y(\mathbf{s})\right\|^2\right] + \mathbb{E}_{\mathbf{s}\sim\mathcal{P}_y}\left[\left\|\mu_x(\mathbf{s}) - \mu_y(\mathbf{s})\right\|^2\right] \tag{7}$$

The probability of selecting a pair increases with the distance $d\left(\mu_x, \mu_y\right)$ between them. This distance-based selection strategy incentivizes behavioral *novelty*, developing a diverse population. Given the distance metric definition, the achieved novelty acts across the action space.

3. Proximal mutation

Let one consider the parent to be mutated μ_p, parametrized by θ_p with genetic memory $\mathcal{B}_p$. The mutation operator samples a batch of transitions B_M and sums the policy gradient with respect to its parameters over the states within B_M. Eq. (8) computes the policy sensitivity s_{μ_p} as the root of the squared gradient summed over the action space:

$$s_{\mu_p} := \sqrt{\sum_{k}^{|\mathcal{A}|}\left(\sum_{i}^{|B_M|} \nabla_{\theta}\mu_{\theta}\left(\mathbf{s}_i\right)\right)_k^2}, \quad B_M \sim \mathcal{B}^p \tag{8}$$

Then, the sensitivity is used to scale the zero-mean Gaussian perturbation, the policy parameters being updated according to $\theta_o \sim \mathcal{N}\left(\theta_p, \frac{\sigma_{mut}}{s} I\right)$. The higher the magnitude of the policy gradient with respect to a parameter, the smaller the mutation step becomes. Essentially, sensitive NN parameters are perturbed less by the scaled update. The offspring policy μ_o has its parameters θ_o within the proximity of its predecessor, thus limiting behavioral divergence [31, 43].

The current framework incorporates the memory and variational operators, extending the mutation towards a safety-informed approach.

III. Methodology

To show how SERL can improve the autonomy and safety of flight controllers, the work employs an offline attitude-tracking task introduced in this section. Then, it presents the SERL as an ERL algorithm that accounts for safety information in the mutation operator. Lastly, the training and evaluation scenarios are described.

A. Attitude Tracking

The simulation environment uses the model of a fixed-wing aircraft and considers continuous attitude control split into episodes of maximum duration T and with sampling rate $f_s = 100\,\text{Hz}$. The high-fidelity six-degree of freedom model combines the nonlinear translational and rotational equations of motion for the rigid body aircraft, trimmed for specific altitude, airspeed, and climb angle (kept at zero in this work) [44, 45]. It has been validated through system identification on flight test data recorded onboard a Cessna Citation II [†].

Eq. (9) depicts the complete modeled aircraft state with the longitudinal and lateral coordinates X_e and Y_e measured with respect to the trim location. The agent state vector $\mathbf{s} \in \mathbb{R}^7$, defined by Eq. (11) retrieves observed information from the aircraft state augmenting it with the current tracking error. The actions $\mathbf{a}_t = \mu(\mathbf{s}_t) \in \mathbb{R}^3$ outputted by the behavioral policy are fed as the input vector at the same frequency f_s. Defined by Eq. (10), the actor's action corresponds to the deflection of the elevator, aileron, and rudder. The thrust control is delegated to an inner auto-throttle [11, 44] and the trim tab and flap deflections are kept at zero. The control diagram from Fig. 1 shows the feedback loop between the non-linear intelligent agent and the controlled plant.

$$\mathbf{x} = [p, q, r, V_{tas}, \alpha, \beta, \theta, \phi, \psi, H, X_e, Y_e]^\top \in \mathrm{R}^{12} \tag{9}$$

$$\mathbf{a} := [\delta_e, \delta_a, \delta_r]^\top \tag{10}$$

$$\mathbf{s} := [\theta_e, \phi_e, \beta_e, p, q, r, \alpha]^\top \tag{11}$$

To account for realistic physical limits of the actuators, the space $\mathcal{A}$ is restricted for each channel bounded to $\left[\mathbf{a}_{\text{low}} = -10°, \mathbf{a}_{\text{high}} = 10°\right]$. The ranges are within the admissible deflections of the modeled control surfaces [46].

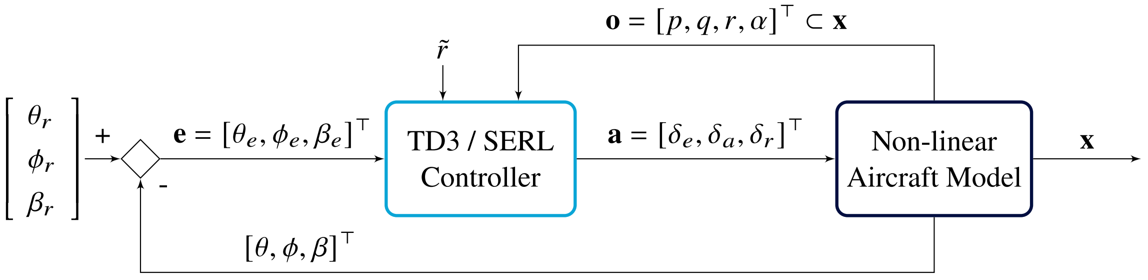
The reference signals correspond to a random sequence of coordinated pitch-roll maneuvers. Following from [11] and [47], the θ_r and ϕ_r references are sequences of cosine-smoothed step signals with amplitudes uniformly sampled from the ranges: $[-25°, 25°]$ for pitch and $[-45°, 45°]$ for roll. The latter is motivated by the values specified in the CS-25 [48]. The range for pitch is based on the previous work of [11, 47, 49]. The sideslip reference remains at zero.

As part of optimal control MDP formulation, Eq. (12) defines the reward signal. Thought the episode (i.e, first case), it penalizes the L_1 norm of the clipped and scaled attitude tracking error, similar to the reward implemented by [11]. The scaling factor $\mathbf{c}_r = \frac{6}{\pi}[1, 1, 4]$ accounts for the smaller magnitude of sideslip error. At the end of the trial, a sparse negative reward $\tilde{r}_T$ penalizes early crashes proportionally to the time left before the prescribed episode duration. Empirically, a value of $C_P = 20$ is deemed appropriate for the proportionally

[†]The PH-LAB research aircraft jointly owned between the TU Delft Faculty of Aerospace Engineering and Netherlands Aerospace Centre (NLR) `https://cs.lr.tudelft.nl/citation/` [10/01/2023]

constant.

$$\tilde{r}_t := \begin{cases} -\frac{1}{3} \left\| \text{clip}\left[\mathbf{c}_r \odot \mathbf{e}_t, -1, 0\right] \right\|_1 & t \in [0, T) \\ -\frac{C_P}{\Delta t} \left(T_{max} - T\right) & t = T \end{cases} \tag{12}$$



$\mathbf{o} = [p, q, r, \alpha]^\top \subset \mathbf{x}$
$\tilde{r}$
$[\theta_r, \phi_r, \beta_r]$
+
-
$\mathbf{e} = [\theta_e, \phi_e, \beta_e]^\top$
TD3 / SERL Controller
$\mathbf{a} = [\delta_e, \delta_a, \delta_r]^\top$
Non-linear Aircraft Model
$\mathbf{x}$
$[\theta, \phi, \beta]^\top$

Fig. 1 Control diagram for the attitude control task using TD3 / SERL.

B. Learning Framework

The Safety-informed Evolutionary Reinforcement Learning referred to as SERL, extends PDERL from [27] by replacing the DDPG actor-critic with Twin Delayed Deep Deterministic Gradient (TD3) and the proximal mutation operator with the safety-informed mutation operator.

1. Framework overview

SERL has a modular design, where each component can be developed separately. The additional argument (N) denotes the number of actors of the genetic population, with SERL(N) having a total count of $N + 1$ policy due to the extra TD3 actor with an equal footprint (i.e., the number of learnable parameters). Figure 2 depicts the components of SERL and the interaction between the RL agent (blue), the GA mechanisms (green), and the simulation environment (grey).

SERL uses two types of replay buffers. The shared buffer trains the RL agent in an off-policy manner but each actor within the population has its own genetic buffer. The critics are feedforward neural networks with different shapes and sizes than the genetic actors and TD3 policy, which have the same footprint.

Empirically, it has been observed that N is the hyperparameter with the highest impact on training, directly altering the balances between the GA and TD3 learning. A larger population increases the number of frames experienced between RL updates and it reduces the number of transitions corrupted with exploratory noise. Thus, increasing the genetic population limits the RL effort. At the same time, both the crossover and mutation operators become more effective at developing novel behaviors once the genetic pool increases.

2. Safety-informed mutation

The hybrid algorithm structure offers the opportunity to develop either of the optimization loops with overall potential gains. However, to take into account existing safety-related domain knowledge, a safety-informed mutation operator is adapted from [31]. Similarly to the proximal mutation, it scales the parameter noise proportionally to the inverse of the norm of the policy gradient (see Eq. (8)). This time, the gradients are estimated on batches B_C sampled from the parent's *critical buffer* $\mathcal{B}_C \subset \mathcal{B}$. It contains the transition tuples associated with a high value of a cost function, directly related to safety knowledge on the maneuver.

Eq. (13) defines the SERL critical buffer as a sub-set of transitions that cause the next-state angle of α and ϕ approach a set of simplified flight envelope bounds. These are marked by entering pre-stall dynamics at an angle of attack higher than 11° or by having a roll angle larger than 60°, empirically observed to degenerate in an unstable spiral motion.

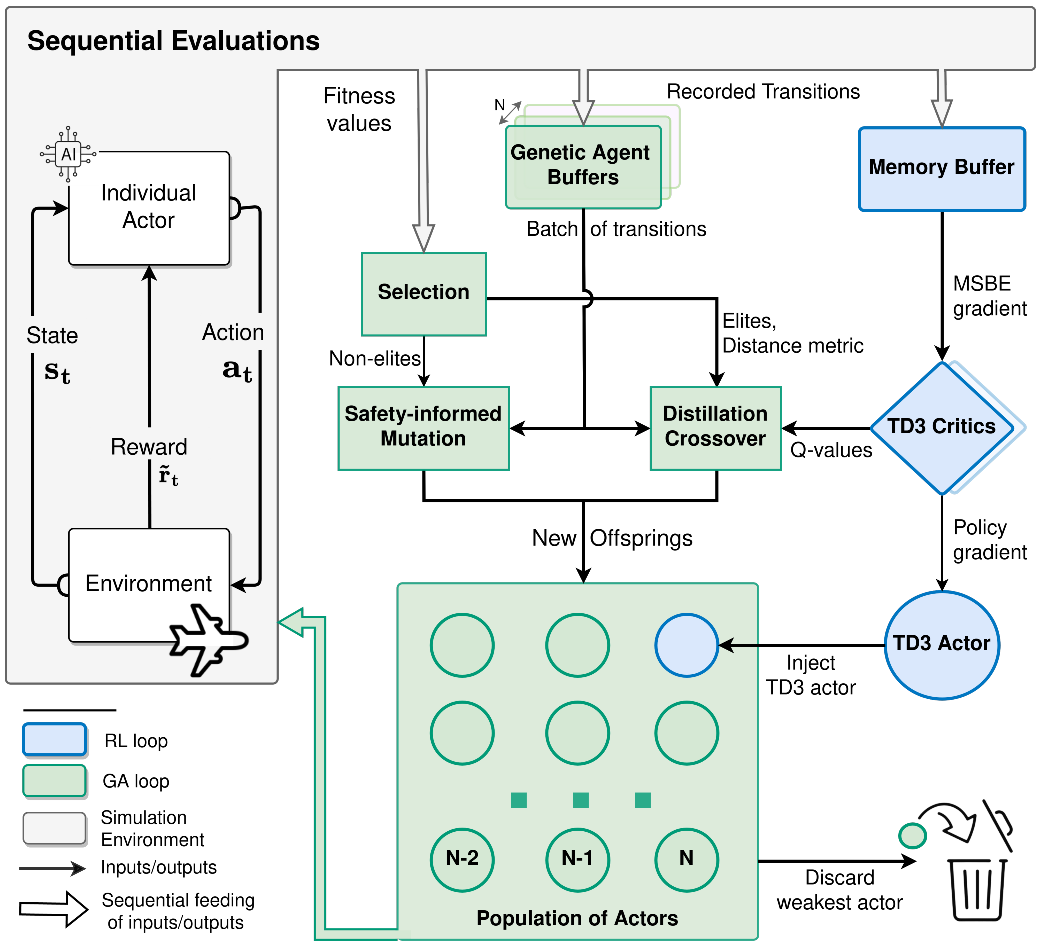


Fig. 2 High-level overview of the data flows through the SERL(N) framework. Inspired from [27, 30, 31].

$$\mathcal{B}_C := \{\langle \mathbf{s}_t, \mathbf{a}_t, \tilde{r}_t, \mathbf{s}_{t+1}\rangle \in \mathcal{B} | \quad \alpha_{t+1} \geq 11^\circ \quad || \quad \phi_{t+1} \geq 60^\circ\} \tag{13}$$

The safety-informed mutation overrides the effect of the proximal operator and directs exploration towards the less-sensitive regions of the parameters space $\mathcal{D}_\theta$. Essentially, the offspring's genome will evolve less in the directions that could result in unsafe behaviors. Nevertheless, since the operator scales the additive parameter noise, unsafe exploration is only discouraged and not stopped so the resulting policies do not benefit from safety guarantees.

C. Optimization for Policy Smoothness

To combat policy noise, the Conditioning for Action Policy Smoothness (CAPS) regularizes the policy loss function according to Eq. (14). Whereas the term $\mathcal{L}_T$ aims for Lipschitz temporal continuity by minimizing the L_2 norm between the current action and the next action, the spatial term $\mathcal{L}_S$ penalizes noise in system dynamics by ensuring a locally consistent policy [15]. The regularization weights, λ_T, and λ_S, control the strength of the corresponding terms and can be tuned. During training, CAPS adds their weighted sum to the policy loss function from Eq. (2) for SGD to propagate their effect and direct the RL updates towards smoother control actions.

$$\mathcal{L}_{\mu}^{(\text{CAPS})} := \lambda_S \cdot \underbrace{\|\mu_\theta(\mathbf{s}) - \mu_\theta(\tilde{\mathbf{s}})\|_2}_{\mathcal{L}_S} + \lambda_T \cdot \underbrace{\|\mu_\theta(\mathbf{s}) - \mu_\theta\left(\mathbf{s}_{t+1}\right)\|_2}_{\mathcal{L}_T}, \quad \tilde{\mathbf{s}} \sim \mathcal{N}(\mathbf{s}, \sigma I) \tag{14}$$

Due to their inherent non-linearity, neural network policies output action signals with spectral components in which the input information is not readily visible. To measure and compare the smoothness of different policies following the same references, the Sm metric from Eq. (15) takes the frequency-weighted sum of the spectral amplitudes of all actuating signals. The metric is inspired by the smoothness defined in [15] but with two key differences. Dividing by the episode duration T penalizes shorter trails and the square root scales down the value, making the unit of the Sm metric equal to rad · Hz. Lastly, the negative sign ensures that higher Sm corresponds to smoother actuating signals and the positive scaling constant C_{S_m} (set to 10) makes the metric comparable to the reward-based fitness term from Eq. (12).

$$Sm := -\frac{C_{S_m}}{T} \cdot \sqrt{\frac{2}{n} \sum_{a=1}^{|\mathcal{A}|} \sum_{k=1}^{n/2} \left(\frac{1}{n} |A[k]|^2 \cdot f_k \right)_a}, \quad n = \frac{T}{f_s} \tag{15}$$

The metric uses the spectral amplitude estimate of the Fourier-transformed actuating signal $A[k]$ calculated via a Fast Fourier Transform. The spectral components are summed up to the discrete Nyquist frequency index, $f_s/2$, sampled at f_s.

While CAPS indirectly optimizes spatial and temporal continuity, it is more difficult to objectively quantify action smoothness in multi-dimensional spaces. Moreover, whereas RL learns from frames, SERL performs updates after episodes are completed. Therefore, Sm is not only used to evaluate the trained actors of both TD3 and SERL but also as an additional fitness term for evolutionary selection.

For the training scenario described by Sec. III.D, the fitness is defined according to Eq. (4). Additionally, an ablation study investigates the effect of CAPS regularisation in comparison to Sm optimization. The latter is obtained by adding the Sm term from Eq. (15) to the fitness function.

D. Training Task

For the training environment, the aircraft model is trimmed for steady straight symmetric flight at $V_{tas} = 90\,\text{m/s}$ and $H = 2,000$ m. The learning phase concerns repeated training epochs alternated with testing epochs. During learning, the maximum duration of one training episode is $T_{max} = 20\,\text{s}$. The training algorithms of both TD3 and SERL consider five agents, each with its own seed used to randomly initiate the learnable parameters of its actor(s) and critics.

For SERL, choosing the episode length is a trade-off between actors receiving enough samples to explore and the training computational complexity. Longer episodes would increase the time required for the neuro-evolutionary loop [‡].

Most of the hyperparameters configure the TD3 agent, apart from the last five parameters which trace back to PDERL settings. Among them, the mutation magnitude σ_{mut} has the highest impact on exploration. The synchronization rate controls the generational frequency of copying the TD3 policy weights. Also, the outcome of population evaluations dictates the champion so a large number of evaluations is more informative but also time-consuming. For SERL(50), using fewer evaluations helps reduce the training time while the number of training steps has been proportionally increased to account for the additional learning actors.

[‡]Training either TD3 or SERL(10) for 10^6 frames required approximately 200 min on 8 CPU cores, Intel Xeon E5-1620 3.50 GHz. For SERL(50), the wall-clock time scales proportionally with the genetic population size.

E. Evaluation Tests

The policies optimized by each agent are tested on multiple scenarios, listed in Table 2. They are selected from the surveyed literature based on their relevance to flight systems, expected difficulty, and possibility to be modeled within the DASMAT framework with the described attitude control architecture.

The cosine-smoothed step references follow similar coordinated pitch-roll maneuvers but the evaluation episodes are longer, lasting $T_{max} = 80\,\text{s}$. Each test case loops through the same list of pseudo-random references generated using three different seeds. Thus, each intelligent controller is evaluated on the same tasks to enable a fair comparison between their tracking performance and smoothness.

In Table 2, the nominal scenario considers the same flight conditions as the training task and it serves as the benchmark for the other cases to be compared against. The next six fault cases are based on the work of [11]. Simulating the agent's response with the controlled plant altered according to each of these cases investigates their fault-tolerant capabilities.

Cases R1 and R2 consider two flight conditions with different dynamic pressures by changing the airspeed and altitude from the trim setting used during training. Since the actuators and aircraft dynamics are affected by the dynamic pressure, these tests benchmark the agent's robustness to unseen flight regimes. We test the new TD3 and SERL agent's robustness on two more extreme cases than the ones previously investigated in [11, 49].

The last case tests the capacity to reject an external disturbance in the presence of realistic Gaussian noise (modeled from [50]) added to the sensor observation. The simplified disturbance is a vertical gust of wind whose velocity is specified by the MIL-F-8785C specification [51].

The normalized Mean Absolute Error (nMAE) evaluates the tracking performance by averaging pitch, roll, and sideslip. The normalization interval is $[-25°, 25°]$ for the pitch and yaw channels whereas the sideslip error is normalized by $[-1°, 1°]$ as its response is expected to remain in the neighborhood of zero.

Table 1 Hyperparameters of the DRL and GA, used training the TD3, SERL(10) and SERL(50).

Parameter Name	Symbol	TD3	SERL(10)	SERL (50)
Population size	N	-	10	50
Learning rate (actor, critics)	$\alpha_{a,c}$	4.33×10^{-4}	4.82×10^{-5}	1.86×10^{-5}
Discount factor	γ	0.99	0.99	0.99
Batch size	$\lvert B \rvert$	64	86	256
Memory buffer size	$\lvert \mathcal{B} \rvert$	100,000	800,000	2,000,000
Actor hidden layers	-	3	3	3
Actor hidden sizes	-	(96, 96,96)	(72,72,72)	(72,72,72)
Critics hidden layers	-	2	2	2
Critics hidden size	-	(200,300)	(200,300)	(200,300)
Non-linear activation	-	ReLU	tanh	tanh
Exploratory noise magnitude	σ	0.33	0.29	0.23
Mutation magnitude	σ_{mut}	N.A.	2.47×10^{-2}	6.27×10^{-2}
Elite Fraction	e	N.A.	0.3	0.2
Genetic memory size	κ	N.A.	8,000	8,000
Actor synchronization rate	-	N.A.	1	1

Table 2 Evaluation cases for fault-tolerance and robustness.

Case	Title	Description
-	Nominal	Unchanged from training: $H = 2.000\,\text{m}$, $V_{tas} = 90\,\text{m/s}$
F1	Iced Wings	Maximum angle of attack reduced by 30% and the drag coefficient increased with 0.06
F2	Shifted Centre of Gravity	Aircraft centre of gravity shifted aft by 0.25 m
F3	Saturated Aileron	Aileron deflection clipped at $\pm 1°$
F4	Saturated Elevator	Elevator deflection clipped at $\pm 2.5°$
F5	Broken Elevator	Elevator effectiveness coefficients multiplied by 0.3
F6	Jammed Actuator	Rudder stuck at a deflection of 15°
R1	High Dynamic Pressure	Trim setting at: $H = 2.000\,\text{m}$, $V_{tas} = 150\,\text{m/s}$
R2	Low Dynamic Pressure	Trim setting at: $H = 10.000\,\text{m}$, $V_{tas} = 90\,\text{m/s}$
R3	Disturbance & Biased Sensor-noise	Additive sensor noise & vertical wind gust of 15 ft/s acting for 3 s

IV. Results and Discussion

This section presents the results of learning, fault tolerance, and initial condition robustness for three distinct agents: SERL(10), SERL(50), and TD3. It also discusses the effects of involving direct Sm optimization. If not specified otherwise, the sample average and standard deviation are computed over 30 random evaluations generated from three seeds. To verify whether the nMAE averages differed significantly, the t-test is used under the assumption of normally distributed nMAE samples.

A. Learning Curves

The training curves from Fig. 3 depict the effect of policy learning by plotting the average episodic reward obtained by each policy alongside the total number of time steps the policies expired interacting with the environment. Ultimately, all agents converge towards returns, comparable to values reported by [11, 47, 49].

TD3 shows an initial advantage but suffers from unstable learning progress marked by spikes in standard deviation followed by sudden drops in the return, referred to as *catastrophic forgetting*. In contrast, both SERL agents are less sample efficient whereas their training progresses less erratically than TD3, and with a lower and monotonically decreasing deviation.

SERL(50) requires five times more samples to reach a similar return - a direct drawback of the poorer sample efficiency of genetic algorithms. SERL(10) learns with sample complexity comparable to the TD3 method as its gradient-updated actor is more often selected among the elites. Thus within SERL, TD3 learning drives early-stage learning, its elitism rate increasing up to 25% after 5×10^5 frames. Then, the rate of RL selection slowly decreases, marking the switch to genetic mechanisms that allow for long-term performance.

B. Fault-tolerancy analysis

Despite validating the effect of training, learning curves do not paint a full picture of the control capabilities. Figure 4 shows the fault-case evaluation performance of the champion policy of SERL(10) and SERL(50) alongside the TD3-only agents trained in parallel to the population by the hybrid loop and the separately trained TD3. The best-behaving SERL(50) actor significantly outperforms ($p < 0.05$) both the SERL(10) champion and the TD3-only actor for all cases apart from the last two fault cases. Looking at F5 where

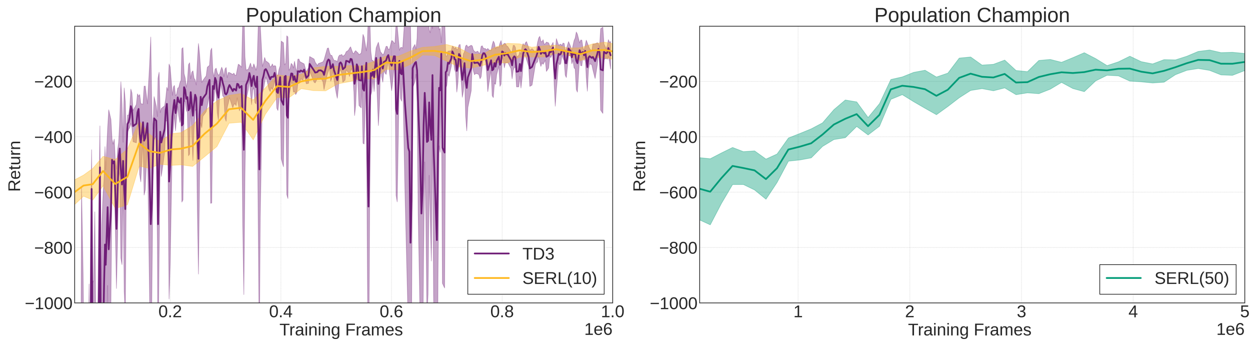


Fig. 3 Learning curves of the average champion reward (solid) and its standard deviation (shading) for TD3, SERL(10), and SERL(50). Statistics are computed over five evaluations.

SERL(50) obtains an nMAE score lower by 6.4% with $p = 0.1 > 0.05$ whereas for F6, the TD3-only actor tracks the references better than the best of SERL(10) and SERL(50) by 16.5% ($p = 2 \times 10^{-35} < 0.05$) and by 6.1% ($p = 8 \times 10^{-13} < 0.05$) respectively. In the last fault case, the nMAE scores are significantly higher than the rest due to the large and relatively constant sideslip bias Sec. IV.B.2 will further detail this case.

For both SERL configurations, the identified champion for each case obtains an average tracking error at least as good as the TD3 actor trained alongside. In the case of a small population, the RL part drives the policy optimization so the TD3 actor is more often selected among the elites. This happens for the nominal scenario and the faults F3, F4, and F6. One can argue that, in the context of the current control tasks, training multiple randomly initialized non-linear controllers benefits from the emerging diversity and thus achieves higher tracking performance in all but one case.

Next, F3 and F6 are discussed, signifying the opposite ends on the spectrum of relative tracking performance.

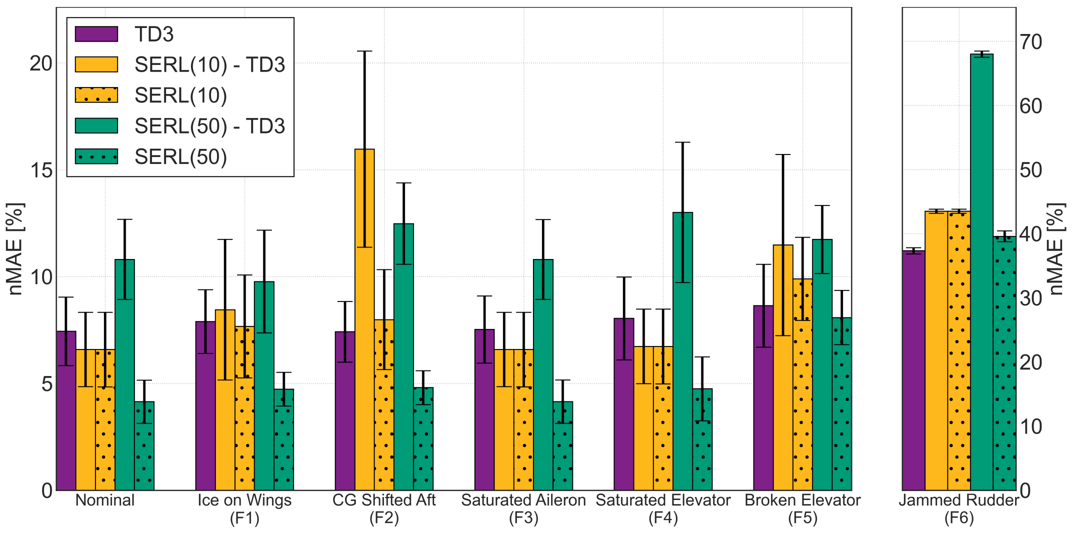


Fig. 4 Tracking nMAE and its standard deviation for each fault case. The hatched bars denote the SERL champion and the solid-colored bars the TD3 actors trained alone or within SERL.

1. Saturated Aileron (F3): downside of aggressive actions

Fig. 5 depicts the responses of the SERL(50) champion and the TD3 actor when tasked to control a plant with a clipped actuator (case F3 from Table 2).

Large actuating bounds during training allow the policies to exploit them and control the system aggressively. In Fig. 5, comparing the responses computed by the two actors, the δ_a commanded by the SERL(50) champion remains within the saturation bounds. With a relatively calm response (i.e., low frequency and magnitude), the champion follows the roll and sideslip references with the same accuracy as in the nominal case. It obtains an nMAE tracking error lower by 45% ($p = 4 \times 10^{-10} < 0.05$) than its TD3-only counterpart.

The TD3 actor behaves more aggressively across all actuating channels, with high-frequency commands and large deflections. At $t = 20\,\mathrm{s}$, the roll angle reference shifts from $3°$ to $-3°$, prompting the TD3 actor to command a high aileron deflection. Since the simulated actuator is clipped, the TD3 actor pushes further, unaware of the fault. Thus, the commanded δ_a spikes, approaching the lower bound of $\delta_{a_{low}} = -10°$. Arguably, this unnecessary and sub-optimal behavior stems from a preference for aggressive actions learned by the TD3 agent.

2. Jammed Rudder (F6): lost novelty

When dealing with a rudder stuck at $\delta_r = 15°$, the genetic champions cannot match the error of the TD3-only agent. Figure 6 shows the responses of the aircraft when controlled by the SERL(50) champion (top plots) and the TD3 actor (bottom plots). One can argue that, when one channel is completely blocked, a significant amount of the behavioral diversity, encoded by the evolved population, has been lost. Here, training multiple NNs is less advantageous than concentrating the frame-based learning updates on one single policy, as done by the TD3-only agent.

Also, while the SERL(50) champion tries to correct for the induced sideslip deviation, the TD3 actor does not. A potential explanation stems from the stronger coupling between the rudder and aileron channel of the TD3-only policy.

C. Robustness analysis

Previous works have shown robustness to the flight condition of a stochastic policy trained within a hierarchical architecture on one trim condition [11]. Despite this, their outer loop controller observed the altitude which offers partial information with respect to the dynamic pressure which is missing from the current attitude tracking task. The bar chart from Fig. 7 shows the control performance in the last four cases from Table 2, corresponding to previously unencountered conditions.

High dynamic pressures would increase the effectiveness of all three control surfaces considered. This benefits both of the SERL controllers, with the SERL(50) error deviation decreasing and the SERL(10) tracking the reference slightly better (14.2% lower error with $p = 0.06 > 0.05$). The TD3 actor experience the opposite, its R1 nMAE being higher by 35.4% ($p = 1 \times 10^{-7} < 0.05$) than the nominal case. Again, the TD3 policy is more aggressive, a sub-optimal trait when the controlled actuators become more effective.

Flying in low dynamic pressure makes the responses sluggish. Reacting to it proves problematic for all controllers and translates into a higher average tracking error and standard deviation. The SERL(50) champion sees an increase of 60.1% which is lower than the one reported by [49] for a less difficult task. In this case, the RL-only agent proves more robust as its nMAE increases by 31.4%, the least among all three.

Lastly, adding biased noise does not significantly influence tracking performance. Moreover, the policies can reject a 15 ft/s vertical wind gust. The SERL champion remains the same for the nominal case, hinting that, in contrast to the faulty scenarios, genetic diversity is not necessary to achieve robustness to this atmospheric disturbance.

Overall, both SERL agents achieve relatively small changes in nMAE, comparable to the ones previously

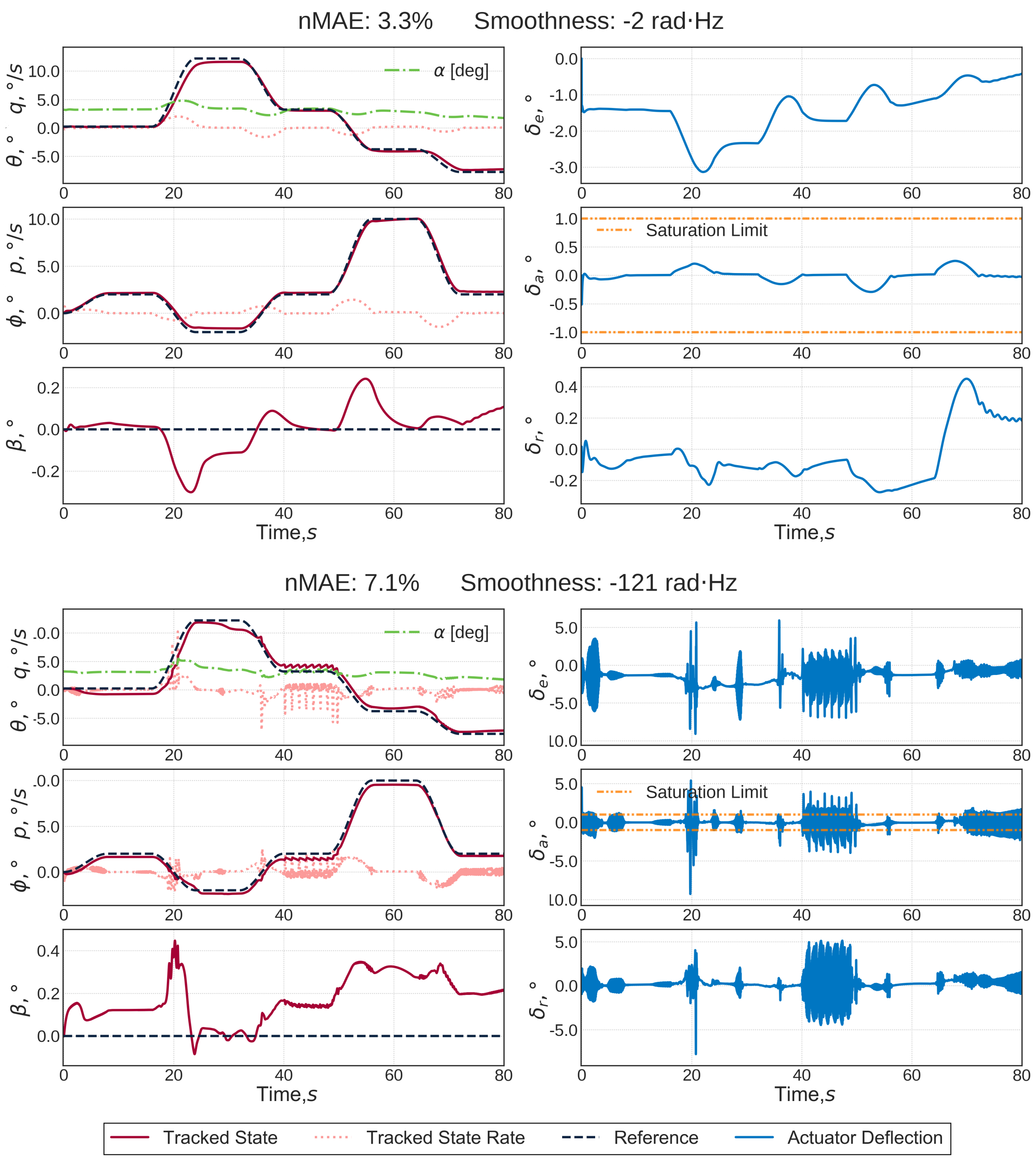


Fig. 5 Evaluation time traces of the SERL(50) champion (top) and TD3 policy (bottom) for the damaged aileron (F3). The interrupted line shows the aileron saturation limit.

reported in the literature. Thus, they remain robust to the change in initial flight conditions and external disturbances. The TD3 agent is less capable of reproducing its nominal performance once the conditions change. Nevertheless, the low dynamic pressure case remains the hardest for all controllers, pointing towards the need for more development.

D. Smoothness of the Control Policy Action Signals

In the time traces shown by the previous sub-sections, a key difference surfaces when comparing the SERL and the RL-only actors. The control policy smoothness is always lower for the TD3-trained controllers by at least one order of magnitude. This is not only visible via the printed *Sm* metric but also from the

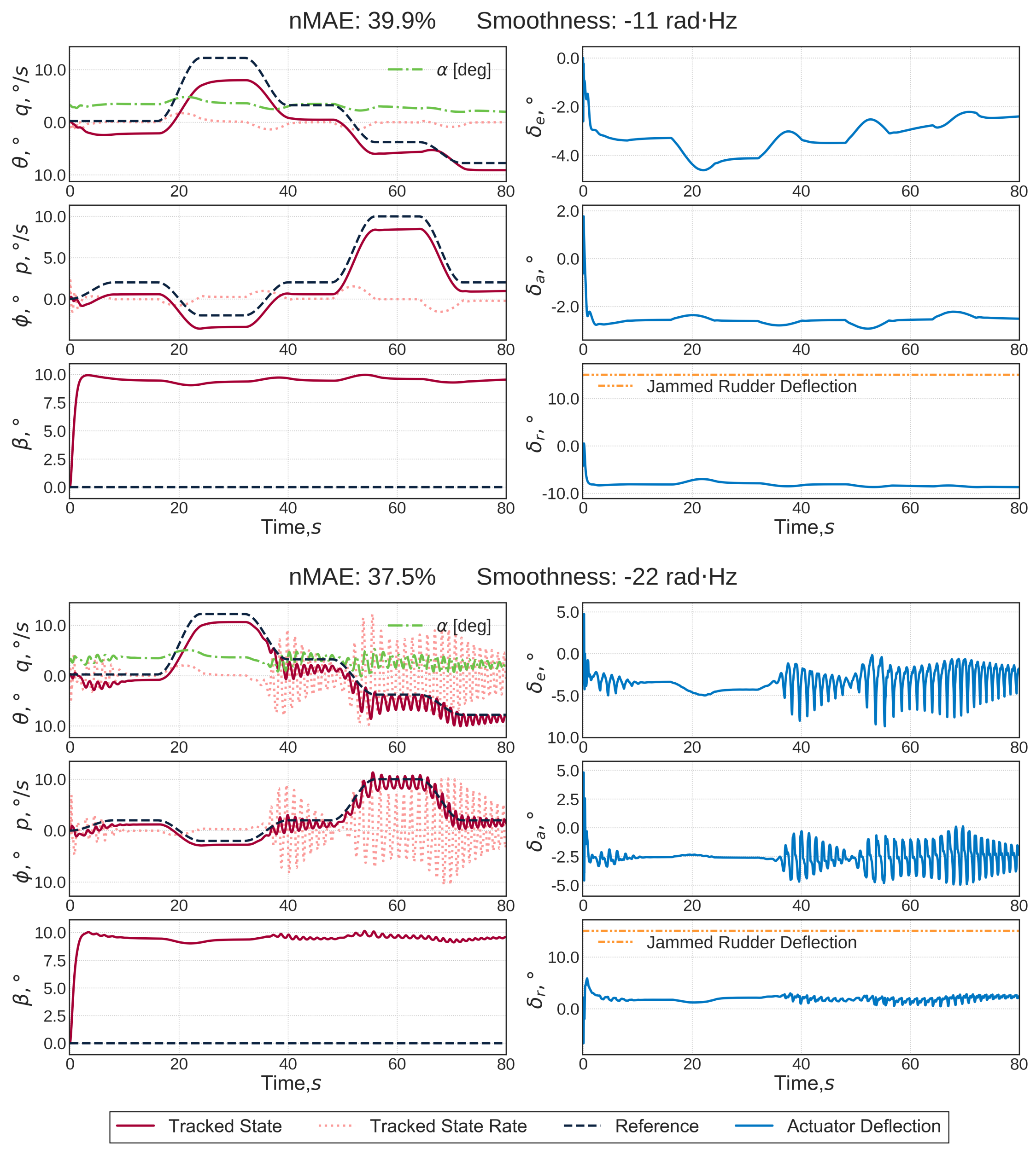


Fig. 6 Evaluation time traces of the SERL(50) champion (top) and TD3 policy (bottom) for the jammed rudder fault (F6). The interrupted line shows the real rudder deflection.

high-frequency components present in the actuating signals.

The possibility of improving the population smoothness is investigated by doing an ablation study on fitness and objective function. One has two options to optimize for a smooth control policy, namely using CAPS and adding the *Sm* term to the fitness function. Spectral components of the tracked reference signals are also present in the actuating signal but simply subtracting their effect from the computed *Sm* value is not readily possible. Thus, only using the same set of references for all agents enables a valid comparison.

The scatter plot from Fig. 8 compares the episodic return of the champion with the population average smoothness. It depicts the three intelligent agents with and without CAPS regularisation and smoothness terms added to the fitness function. For SERL(50), omit the CAPS configuration as, in the large population

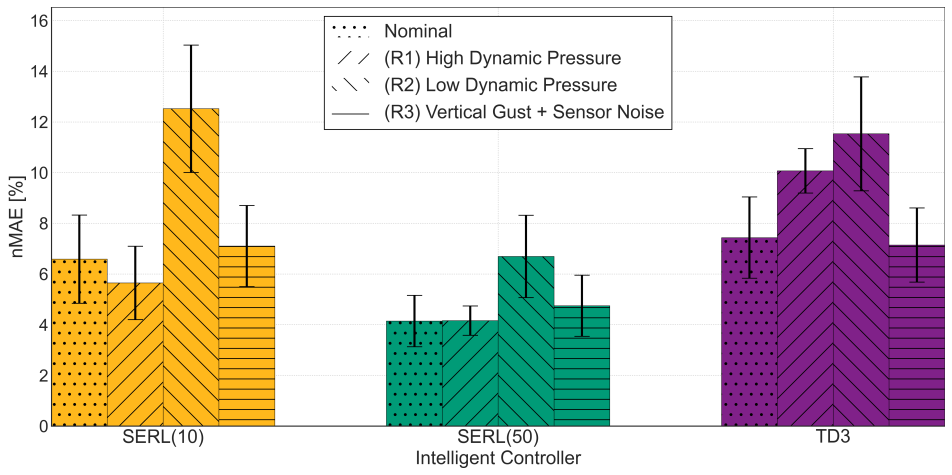


Fig. 7 Average nMAE and standard deviation for SERL(10), SERL(50) and TD3 controller. Hatching denotes each of the evaluation cases (i.e., nominal and R1-3).

scenario, its impact of TD3 is insignificant.

Increasing the population size affects the performance and control smoothness at different scales. The SERL(50) agent obtains a reward comparable with the CAPS-regularised TD3 and SERL(10) but it maintains a higher and almost constant policy smoothness at $Sm = -5 \pm 1$ rad · Hz. With the highest performance at $R = -78.9 \pm 21.3$, TD3 suffers from the high-frequency action noise already visible in the time traces, with an $Sm = -1500 \pm 123$ rad · Hz. The effect of both CAPS and smooth fitness are visible, significantly shifting the Sm metric towards the right.

As expected, adding the Sm term to the fitness function improves the control smoothness but it reduces the selective pressure for performance. Hence, when the TD3 actor is not regularised, it does not pass the evolutionary selection step. In practice, this results in a significantly higher spread of the SERL(10)-SF compared to the SERL(10)-CAPS+SF. While SERL(50) overcomes this issue, the small population of SERL(10) does not so its top performance deteriorates. For SERL(10), the selection for smooth policies removes the injected TD3 policies which should drive the optimization search. Thus, the loss in performance can only be partially mitigated by CAPS.

Lastly, the standard deviation is an imperfect estimator of the population distribution. The Sm distribution shows skewness towards large smoothness, indicating that, at the end of the training, only a few noisy outliers remain. These outliers can be traced back to TD3-trained weights injected during synchronization. The performance metric distribution is more symmetrically spread through the population.

Causes for lack of action smoothness

To explain the control noise phenomena, the current work presents the following three factors which have been inferred from the aforementioned empirical ablation study and the surveyed literature:

1) *Using (deep) neural networks as optimal policy function approximators*: Multi-layered NNs have a significant number of degrees of freedom, justifying their popularity in learning complex non-linear dynamics [6]. Depending on how and for how long they are trained, NNs can learn any frequency component within the training signal. Without being biased towards them, this also includes the high-frequency components. Nevertheless, the frequency responses and Fourier analysis of neural networks are not yet a well-researched field.
2) *Exploiting environmental dynamics*: The high-frequency components in the actuating signals are damped out and saturated by the simulated aircraft dynamics [44]. Hence, they are not present in

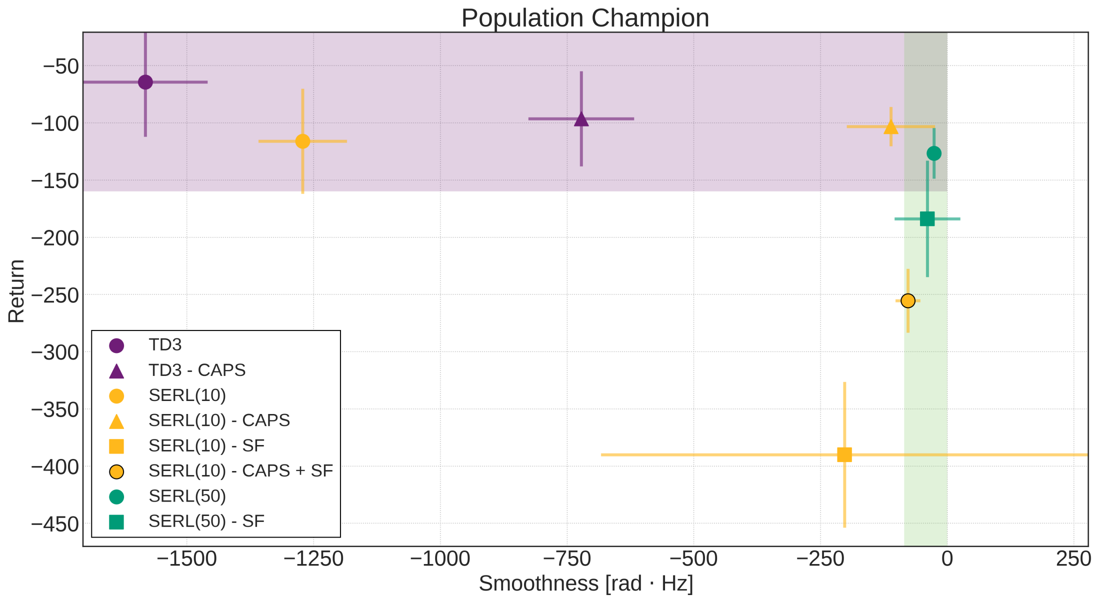


Fig. 8 Champion return versus average smoothness for three agents trained with and without CAPS regularisation and a smoothness optimization term.

the tracked states. Meanwhile, a real-life system with unideal actuators and friction would further reduce the high-frequency comments. By not modeling such phenomena or penalizing the noise via the reward or fitness signals, the agent could optimize the policy towards containing it for local incremental in the loss landscape. In turn, this would eventually aggravate the already-present gap between offline training and deployment on real hardware.

3) *Exploratory strategy*: As an off-policy learning framework, TD3 explores during training by adding zero-mean Gaussian noise to the selected actions. Thus, the sampled transitions used by the SGD updates are already corrupted by noise. The NNs could learn to overfit these noisy dynamics. Empirical evidence suggests that the absence or damping of additive state-action exploratory noise can benefit the action smoothness. The analysis presented by [15], has empirically shown that on-policy methods, e.g. PPO, or stochastic policy algorithms, such as SAC, are less prone to train noisy control policies.

Whereas the first two factors do not directly generate action noise, they enable it. Thus, one can argue that the developed TD3 controller learns less smooth policies in comparison to the SERL as a consequence of combining the three aspects.

Previous works in DRL-based flight control have circumvented the noise problem by diminishing the impact of the first two factors. To reduce noise, one can use simpler control architectures, such as a PID, regularizing the NN with, for example, CAPS [15], or rewarding smooth actions (a method used by the authors of [14]). Similarly, incremental control such as the methods from [11, 49], low-pass filtering or discretizing the output signals obtain the smoothing effect [13]. These methods change the learning to partially remove the second factor.

Consequently, another hypothesis states that neural evolution mitigates the impact of the third factor. While state-action noise is required for off-policy exploration of the TD3 agent, SERL can explore the policy parameter space, learning without generating noisy transitions. In parallel, the safe mutation operator biases exploratory learning towards the non-critical regions of the state space. The resulting behaviors propagate through the genetic population via distillation crossover which also inhibits the development of aggressive actions via the L_2-norm. Thus, SERL can train in the large-population configuration without deteriorating the average smoothness.

The smoothness-reward trade-off

Understating the trade-off between performance and action smoothness is relevant for bridging the gap between simulation training and the deployment of hardware. Arguably, online low-pass filtering of the control commands can mitigate the effects of noise. Despite this, they can have unpredicted consequences on the neural network controllers due to their highly non-linear structure [15]. Thus, tuning the filters for multiple operating conditions might require real-life testing [15]. Similarly, changing the training task to incremental control achieves a filtering effect. Unfortunately, it also inherently increases the dimensionality of the learning space, hindering convergence and increasing the rate of failed trials [11].

Looking back at Fig. 8, two distinct regions emerge. Applications such as drone racing prioritize short-term performance and the motor actuators can be more easily replaced in case of damage to the high-frequency commands. These vehicles might therefore benefit more from controllers trained within the top-left configurations, accepting a higher degree of action noise. In contrast, when the aircraft carries sensitive payloads, such as scientific instruments or passengers, performance might be sacrificed to ensure that smooth commands are followed. For these applications is more appropriate to train control agents in the right-hand region.

In both cases, the designer should have the opportunity to decide, but using SERL(50) achieved the best of both worlds at the cost of proportionally longer training time.

V. Conclusion

We combined two bio-inspired frameworks, DRL and EA, culminating in SERL, a safety-informed Evolutionary Reinforcement Learning method that trains flight controllers on a non-linear fixed-wing aircraft model to provide optimal attitude tracking. It uses an actor-critic RL structure with GA mechanisms. The TD3 algorithm improves sample efficiency and stability of the learning. The GA population evolves through a novelty-seeking crossover operator and safety-informed mutation that uses high-level domain knowledge about the flight envelope. With them, SERL obtains parameter-space exploration adding to the off-policy state space exploration of TD3.

The current work empirically proved that, by adjusting the population size, SERL can balance performance and control smoothness. Following previous research, off-policy state-space exploration using unregularised multi-layer neural networks was shown to aggravate the control action noise. This happens in simulation environments that only prioritize tracking performance but can be alleviated via parameter-space exploration performed via neuro-evolutionary operations.

Training a large controller population required proportionally more samples but achieved significantly higher fault tolerance than a comparable TD3 agent in five out of six evaluation cases. It also remained robust to flight conditions unencountered during training and external disturbances in the presence of realistic observation noise. This demonstrated the generalization benefit of evolving a diverse population of controllers through distillation crossover and safety-informed mutations. Nevertheless, when system faults collapse the offline-obtained diversity, the TD3 remains marginally superior.

To build on top of the current research, it is recommended to address the issue of sample inefficiency. Ideally, such issues could be targeted by improving the sample efficiency of either or both learning frameworks through the use of distributional critics to enhance the RL convergence rate [47] or replace the GA loop with a variant of the Covariance Matrix Adaptation [52]. Moreover, an adaptation scheme shall be designed to select online the behavioral policy from the SERL-trained pool of actors. Lastly, future work has to push for control system validation through flight testing as the empirically proven offline robustness is a valuable but insufficient step towards showing that the SERL policies can provide attitude control on real systems. Generally, validating NNs as low-level controllers is cumbersome due to their black-box nature, which lacks explainability. The latter has been addressed by intensive research in the eXplainable AI (XAI) field [53], also transferred to DRL applied to flight control problems [54].

With the increasing popularity of artificial intelligence, research in control systems has the challenge and opportunity to adapt and synchronize its state-of-the-art algorithms with novel AI methods. Directly balancing the performance-smoothness trade-off and benefiting from its proven robustness, the offline trained SERL agents can be more flexibly deployed on real hardware applications. This represents a step towards including artificial intelligence algorithms within autonomous fault-tolerant controllers, making them a safety net applicable to safety-critical systems in general and to aircraft in particular.

References

[1] ICAO, "Safety Report," Tech. rep., International Civil Aviation Organization, 2020. URL `https://www.icao.int/safety/iStars/Pages/Accident-Statistics.aspx`.

[2] Lu, P., van Kampen, E.-J., De Visser, C., and Chu, Q., "Aircraft fault-tolerant trajectory control using Incremental Nonlinear Dynamic Inversion," *Control Engineering Practice*, Vol. 57, 2016, pp. 126–141.

[3] Edwards, C., Lombaerts, T., Smaili, H., et al., "Fault tolerant flight control," *Lecture Notes in Control and Information Sciences*, Vol. 399, 2010, pp. 1–560.

[4] Lavretsky, E., and Wise, K. A., "Robust adaptive control," *Robust and adaptive control*, Springer, 2013, pp. 317–353.

[5] Sutton, R. S., and Barto, A. G., *Reinforcement learning: An introduction*, MIT press, 2018.

[6] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, 2016. `http://www.deeplearningbook.org`.

[7] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., "Human-level control through deep reinforcement learning," *nature*, Vol. 518, No. 7540, 2015, pp. 529–533. https://doi.org/https://doi-org.tudelft.idm.oclc.org/10.1038/nature14236.

[9] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., "Mastering the game of Go with deep neural networks and tree search," *nature*, Vol. 529, No. 7587, 2016, pp. 484–489.

[10] Choi, M., Filter, M., Alcedo, K., Walker, T. T., Rosenbluth, D., and Ide, J. S., "Soft Actor-Critic with Inhibitory Networks for Retraining UAV Controllers Faster," *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 1561–1570.

[11] Dally, K., and Van Kampen, E.-J., "Soft Actor-Critic Deep Reinforcement Learning for Fault Tolerant Flight Control," *AIAA SCITECH 2022 Forum*, 2022, p. 2078.

[12] Bøhn, E., Coates, E. M., Moe, S., and Johansen, T. A., "Deep reinforcement learning attitude control of fixed-wing uavs using proximal policy optimization," *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2019, pp. 523–533.

[13] Braun, D., Marb, M. M., Angelov, J., Wechner, M., and Holzapfel, F., "Worst-Case Analysis of Complex Nonlinear Flight Control Designs Using Deep Q-Learning," *Journal of Guidance, Control, and Dynamics*, 2023, pp. 1–13.

[14] Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D., "Champion-level drone racing using deep reinforcement learning," *Nature*, Vol. 620, No. 7976, 2023, pp. 982–987.

[15] Mysore, S., Mabsout, B., Mancuso, R., and Saenko, K., "Regularizing action policies for smooth control with reinforcement learning," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 1810–1816.

[16] Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. G., “Evolutionary Robotics: What, Why, and Where to,” *Frontiers in Robotics and AI*, Vol. 2, 2015. https://doi.org/10.3389/frobt.2015.00004.

[17] Dianati, M., Song, I., and Treiber, M., “An introduction to genetic algorithms and evolution strategies,” Tech. rep., University of Waterloo, Waterloo, Ontario, Canada, 2002.

[18] Simon, D., *Evolutionary optimization algorithms*, John Wiley & Sons, 2013.

[19] Papavasileiou, E., Cornelis, J., and Jansen, B., “A Systematic Literature Review of the Successors of “NeuroEvolution of Augmenting Topologies”,” *Evolutionary Computation*, Vol. 29, No. 1, 2021, pp. 1–73.

[20] Stanley, K. O., and Miikkulainen, R., “Evolving neural networks through augmenting topologies,” *Evolutionary computation*, Vol. 10, No. 2, 2002, pp. 99–127.

[21] Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B., “Robots that can adapt like animals,” *Nature*, Vol. 521, No. 7553, 2015, pp. 503–507.

[22] Allard, M., Smith, S. C., Chatzilygeroudis, K., Lim, B., and Cully, A., “Online damage recovery for physical robots with hierarchical quality-diversity,” *ACM Transactions on Evolutionary Learning*, Vol. 3, No. 2, 2023, pp. 1–23.

[23] Fontaine, M. C., Togelius, J., Nikolaidis, S., and Hoover, A. K., “Covariance matrix adaptation for the rapid illumination of behavior space,” *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 94–102.

[24] Krishnakumar, K., and Goldberg, D. E., “Control system optimization using genetic algorithms,” *Journal of Guidance, Control, and Dynamics*, Vol. 15, No. 3, 1992, pp. 735–740.

[25] Ghiglino, P., Forshaw, J. L., and Lappas, V. J., “Online evolutionary swarm algorithm for self-tuning unmanned flight control laws,” *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 4, 2015, pp. 772–782.

[26] Emami, S. A., Castaldi, P., and Banazadeh, A., “Neural network-based flight control systems: Present and future,” *Annual Reviews in Control*, Vol. 53, 2022, pp. 97–137.

[27] Bodnar, C., Day, B., and Lió, P., “Proximal distilled evolutionary reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, 2020, pp. 3283–3290.

[28] Pourchot, A., and Sigaud, O., “CEM-RL: Combining evolutionary and gradient-based methods for policy search,” *arXiv preprint arXiv:1810.01222*, 2018.

[29] Sigaud, O., “Combining Evolution and Deep Reinforcement Learning for Policy Search: a Survey,” *ACM Transactions on Evolutionary Learning*, 2022.

[30] Khadka, S., and Tumer, K., “Evolution-guided policy gradient in reinforcement learning,” *Advances in Neural Information Processing Systems*, Vol. 31, 2018.

[31] Marchesini, E., Corsi, D., and Farinelli, A., “Exploring safer behaviors for deep reinforcement learning,” *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 2022, pp. 7701–7709. URL https://ojs.aaai.org/index.php/AAAI/article/view/20737.

[32] Bertsekas, D., *Reinforcement Learning and Optimal Control*, Athena Scientific optimization and computation series, Athena Scientific, 2019. URL http://www.mit.edu/~dimitrib/RLbook.html.

[33] Fujimoto, S., Hoof, H., and Meger, D., “Addressing function approximation error in actor-critic methods,” *International conference on machine learning*, PMLR, 2018, pp. 1587–1596.

[34] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S., “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[35] Thrun, S., and Schwartz, A., "Issues in using function approximation for reinforcement learning," *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, Vol. 6, 1993, p. 263.

[36] Polyak, B. T., "Some methods of speeding up the convergence of iteration methods," *Ussr computational mathematics and mathematical physics*, Vol. 4, No. 5, 1964, pp. 1–17.

[37] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I., "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[38] Zhang, B.-T., Muhlenbein, H., et al., "Evolving optimal neural networks using genetic algorithms with Occam's razor," *Complex systems*, Vol. 7, No. 3, 1993, pp. 199–220.

[39] Koza, J. R., and Poli, R., "Genetic programming," *Search methodologies*, Springer, 2005, pp. 127–164.

[40] Khadka, S., Majumdar, S., Nassar, T., Dwiel, Z., Tumer, E., Miret, S., Liu, Y., and Tumer, K., "Collaborative evolutionary reinforcement learning," *International conference on machine learning*, PMLR, 2019, pp. 3341–3350.

[41] Pugh, J. K., Soros, L. B., and Stanley, K. O., "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, Vol. 3, 2016, p. 40.

[42] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M., "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.

[43] Lehman, J., Chen, J., Clune, J., and Stanley, K. O., "Safe mutations for deep and recurrent neural networks through output gradients," *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 117–124.

[44] Linden, V. D., "DASMAT-Delft University aircraft simulation model and analysis tool: A Matlab/Simulink environment for flight dynamics and control analysis," *Series 03: Control and Simulation 03*, 1998. URL http://resolver.tudelft.nl/uuid:9bfb1f68-2f7c-4f32-91b4-79297d295f84.

[45] Van den Hoek, M., de Visser, C., and Pool, D., "Identification of a Cessna Citation II model based on flight test data," *Advances in Aerospace Guidance, Navigation and Control*, Springer, 2018, pp. 259–277. URL http://resolver.tudelft.nl/uuid:c0a57513-38b7-4d3a-898c-fa57c3e7ac2e.

[46] Konatala, R., Van Kampen, E.-J., and Looye, G., "Reinforcement Learning based Online Adaptive Flight Control for the Cessna Citation II (PH-LAB) Aircraft," *AIAA Scitech 2021 Forum*, 2021, p. 0883. URL http://resolver.tudelft.nl/uuid:b37cefbf-e353-43cf-8d9d-98f2653216c6.

[47] Seres, P., Liu, C., and van Kampen, E.-J., "Risk-sensitive Distributional Reinforcement Learning for Flight Control," *IFAC*, Vol. 56, No. 2, 2023, pp. 2013–2018.

[48] EASA, "Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes (CS-25)," Tech. rep., European Union, June 2021. URL https://www.easa.europa.eu/en/document-library/certification-specifications/cs-25-amendment-27.

[49] Teirlinck, C., and Van Kampen, E.-J., "Reinforcement Learning for Flight Control: Hybrid Offline-Online Learning for Robust and Adaptive Fault-Tolerance," Tech. rep., Delft University of Technology, 2022. URL http://resolver.tudelft.nl/uuid:dae2fdae-50a5-4941-a49f-41c25bea8a85.

[50] Grondman, F., Looye, G., Kuchar, R. O., Chu, Q. P., and Van Kampen, E.-J., "Design and flight testing of incremental nonlinear dynamic inversion-based control laws for a passenger aircraft," *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, p. 0385.

[51] Moorhouse, D. J., and Woodcock, R. J., "Background information and user guide for MIL-F-8785C, military specification-flying qualities of piloted airplanes," Tech. rep., Air Force Wright Aeronautical Labs Wright-Patterson AFB OH, 1982.

[52] Hansen, N., "The CMA evolution strategy: a comparing review," *Towards a new evolutionary computation*, 2006, pp. 75–102.

[53] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information fusion*, Vol. 58, 2020, pp. 82–115.

[54] de Haro Pizarroso, G., and Van Kampen, E.-J., "Explainable Artificial Intelligence Techniques for the Analysis of Reinforcement Learning in Non-Linear Flight Regimes," *AIAA SCITECH 2023 Forum*, 2023, p. 2534.