Evolutionary Reinforcement Learning
Hybrid Approach for Safety-Informed Fault-Tolerant Flight Control

Gavra, Vlad; van Kampen, Erik Jan

DOI
10.2514/1.G008112

Publication date
2024

Document Version
Final published version

Published in
Journal of Guidance, Control, and Dynamics

JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 47, No. 5, May 2024

Evolutionary Reinforcement Learning: Hybrid Approach for Safety-Informed Fault-Tolerant Flight Control

Vlad Gavra* and Erik-Jan van Kampen†
Delft University of Technology, 2600 GB Delft, The Netherlands

https://doi.org/10.2514/1.G008112

Recent research in artificial intelligence potentially provides solutions to the challenging problem of fault-tolerant and robust flight control. This paper proposes a novel Safety-Informed Evolutionary Reinforcement Learning algorithm (SERL), which combines Deep Reinforcement Learning (DRL) and neuroevolution to optimize a population of nonlinear control policies. Using SERL, the work has trained agents to provide attitude tracking on a high-fidelity nonlinear 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 action signals.

Nomenclature

$\mathcal{A}, \mathcal{S}$	=	sets of all possible actions, states
B	=	batch of transition tuples
$\mathcal{B}$	=	memory buffer of previous experiences
$\mathcal{D}_x$	=	domain in which variable x resides
E	=	policy evaluations per epoch/generation
F	=	fitness function value
H, V_{tas}	=	altitude, true airspeed, m/s
$\mathcal{L}$	=	loss function
N	=	population size
p, q, r	=	roll, pitch, and yaw rates, rad/s
Q_ϕ, μ_θ	=	critic function parameterized by ϕ, deterministic actor parameterized by θ
$\tilde{r}, R$	=	reward, future return value
Sm	=	smoothness value of action signal, rad/s
T	=	duration of simulation trail, s
f_s	=	sampling frequency, Hz
$\mathcal{T}_t$	=	transition tuple experienced by agent at time t
$\boldsymbol{a}, \boldsymbol{s}, \boldsymbol{x}$	=	vectors for agent's action, state, and aircraft state
$\alpha, \beta, \phi, \theta, \psi$	=	angle of attack, side slip, roll, pitch, and yaw, rad
$\alpha_{\text{a.,c.}}, \gamma$	=	learning rate of actor, critic learning, and temporal discount rate of future rewards
$\delta_e, \delta_a, \delta_r$	=	elevator, aileron, and rudder deflection, rad
ϵ	=	random variable with Gaussian distribution
λ	=	policy loss regularization coefficient
σ	=	standard deviation of Gaussian distribution

I. Introduction

AVIATION has recently experienced an unprecedented safety record, its ever-lowering rate of accidents making it the safest means of long-distance transportation in terms of human 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, will further reduce the number of air crashes.

Modern techniques in intelligent control incorporate artificial intelligence, aiming to enhance the safety, durability, autonomy, and performance of aerospace vehicles. A milestone along the way 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 a control policy [5,6]. Its extension, Deep Reinforcement Learning (DRL) harvests the potential of deep neural networks (NNs) as data-driven models that approximate high-dimensional and nonlinear dynamics [7–9]. These algorithms belong to the class gradient-based optimization as the neural networks are trained via gradient descent [7].

Applied to flying systems, DRL has recently obtained state-of-the-art performance in complex tasks. These include flight guidance [10], low-level attitude control in the presence of faults [11] or external 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 metaheuristics 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 nonlinear mappings parameterized as neural networks by using an EA to iteratively update their weights [19]. In optimal control, NE is an episode-based alternative to DRL [20,21].

When evolving randomly initialized nonlinear policies, the intrinsic difference in the parameters of individuals translates into novel behaviors [22]. 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 nonlinear control tasks, such as fault-tolerant legged robots [22,23] and robot-arm manipulation [24]. Within flight control, GAs have been tasked to derive symbolic control laws directly [25] or schedule the gains of linear controllers such as Proportional Integral Derivative (PID) [26]. Nevertheless, the applications of NE in flight control are, until now, limited as reviewed by the authors of [27].

Despite their achievements, both the aforementioned biologically-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 [28–30]. Evolutionary Reinforcement Learning (ERL),

Received 24 March 2023; accepted for publication 14 December 2023; published online 14 February 2024. Copyright © 2024 by Delft University of Technology. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the ISSN 1533-3884 to initiate your request. See also AIAA Rights and Permissions www.aiaa.org/randp.

*M.Sc. Graduate, Control and Simulation Section, Faculty of Aerospace Engineering, P.O. Box 5058; vladgavra98@gmail.com (Corresponding Author).

†Assistant Professor, Control and Simulation Section, Faculty of Aerospace Engineering, P.O. Box 5058; E.vanKampen@tudelft.nl. Member AIAA.

proposed by Khadka and Turner [31], trains a population of control policies in a GA loop, periodically infusing it with policies optimized by 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 [28,30]. Frameworks such as Proximal Distilled ERL (PDERL) incorporate numerically stable policy mutations [28,32] to counteract catastrophic forgetting and distillation crossover to enhance sample efficiency [30], thus achieving state-of-the-art performance in continuous control [28].

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 GA 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. Section IV compares the performance of the trained agents and discusses the effect of the SERL mechanism on the control policy smoothness. Finally, the paper concludes by summarizing the achievements in Sec. V.

II. Background

This section highlights the mathematical formalism of intelligent control using ERL. It introduces the sequential decision making for control and then summarizes actor/critic RL, population-based policy search concepts, and the hybrid algorithm combining the two.

A. Sequential Decision Making

The learning agent encapsulates a control policy that sequentially takes rewarded decisions within the environment. This is formalized by a Markov Decision Process (MDP), which states that $\mathcal{P}\{\boldsymbol{s}_{t+1}, \tilde{r}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t, \ldots, \boldsymbol{s}_0, \boldsymbol{a}_0\} = \mathcal{P}\{\boldsymbol{s}_{t+1}, \tilde{r}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t\}$, 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 [6,33]. It will form the basis for the flight control task presented in Sec. III.

B. Deep Reinforcement Learning

In DRL, the policy of a deterministic agent commands at each time step an action $\boldsymbol{a}_t \in \mathcal{A}$ according to a deterministic function of its state $\mu: \mathcal{S} \to \mathcal{A}$ and retrieves from its observation the transition tuple $\mathcal{T}_t = \langle \boldsymbol{s}_t, \boldsymbol{a}_t, \tilde{r}_t, \boldsymbol{s}_{t+1} \rangle$. 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 $Q(\boldsymbol{s}, \boldsymbol{a})$ predicts the value of the return from the given state $\boldsymbol{s}$ when the agent selects action $\boldsymbol{a}$ thereafter following policy μ [6].

Actor/critic methods combine policy and value function learning [6]. 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 $Q(\boldsymbol{s}, \boldsymbol{a}) \approx Q_\phi(\boldsymbol{s}, \boldsymbol{a})$ with $\phi \in \mathcal{D}_\phi$, iteratively updating it. The parametric policy $\mu_\theta(\boldsymbol{s})$ (with $\theta \in \mathcal{D}_\theta$) subsequently trains on the estimated Q values, moving toward the optimal function $\mu^*(\boldsymbol{s})$.

A state-of-the-art method, Deep Deterministic Policy Gradient (DDPG) employs NNs as function approximators for both the critic and deterministic actor due to their intrinsic ability to learn complex nonlinear dynamics [7]. The Twin-Delayed DDPG (TD3), developed by Fujimoto et al. [34], improves the sample-complexity and learning stability of DDPG in offline continuous control tasks [35]. 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 learn from data batches B randomly sampled from the memory buffer $\mathcal{B}$. The critic part trains using recursive temporal difference updates, minimizing the mean squared Bellman error (MSBE) from Eq. (1). TD3 updates 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 [36] and thus has stable learning [34],

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

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,

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

The target networks are separate copies of the actor and critic networks, parameterized 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 [8,37].

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 [8], 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 regularization. For both the behavioral policy and the target one, the resulting actions are clipped to lie within the interval admissible by the environment $[\boldsymbol{a}_{\mathrm{low}}, \boldsymbol{a}_{\mathrm{high}}]$,

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

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 employs a form of stochastic gradient descent (SGD) to optimize the actor and critic, which requires the backpropagation of the loss functions' gradients with respect to the networks' parameters [7]. By doing so, the batch SGD updates drive sample-efficient experience-based training. As a downside, local optima and saddle points in the policy loss landscape alongside noisy gradient estimates impede long-term parameter-space exploration and therefore hinder learning [38].

C. Evolutionary Algorithms for Control

EA is a biologically-inspired alternative which, as a local stochastic heuristic, iteratively improves a population of individuals targeting a higher fitness value [19]. Opposing gradient-based methods, an EA is not bound to the neighborhood of a local optimum but it searches over a multi-dimensional space for the global optimal fitness [18].

In an optimal control framework, the evolving individuals are parameterized control policies. Their fitness function values, defined as $F_{\mu_i}: \mathcal{D}_{\mu_i} \to \mathbb{R}$, are based on the control objective(s) computation over one or multiple trials on metrics in both the parameter and phenotype spaces.

[‡]The source code used for training and evaluation is available at https://github.com/VladGavra98/SERL [retrieved 10 March 2023].

All EAs are characterized by a standard underlying architecture. First, the individuals are initialized, usually sampling from a random distribution. Then, the EA loops over multiple generations until a termination criterion has been reached. Such criteria are based on fitness convergence, number of generations, or a specific fitness value being reached. During each generation, it evaluates the episodic performance of each individual, selects the best-behaving or fittest individuals, applies variational and recombination operators, and discards a sub-set of the under-performers. Because of the generational updates happening on an episode basis, an EA optimizes controllers without a mandatory Markovian property. Despite this, when the evolving policies act according to an MDP setting, the fitness value of a policy can be a mapping of the rewards accumulated during its evaluation [31].

The broad spectrum of EA paradigms inherits the basic bare-bone structure and comprises algorithms like Evolution Strategies [29], GA [39,40], and Genetic Programming (GP). In contrast to the first two, the search space of a GP spans both the encoded policy structure and its parameters [18]. Whereas all classes can offer derivative-free optimization of real-valued genomes, due to their wide-scale popularity and early adoption in ERL, the genetic algorithm subclass remains the focus for the rest of this section.

Benefiting from a modular architecture, GAs can be described by their inner operators: selection, mutation, and recombination or crossover. During selection, solutions with higher fitness values have a higher probability of being maintained in the population. The possible implementations of it are stochastic universal sampling, overselection, rank based (simple or linear), and tournament [17,18]. The latter possesses the advantage of computational efficiency over universal sampling and overselection, while its selective pressure scales proportionally to the fitness values and not to individual rank. Thus, tournament selection is attractive for ERL methods, especially when dealing with a large populations of policies.

Over multiple generations, selective pressure filters out the least fit individuals, perpetuating the behaviors associated with high fitness. The discarded policies are replaced with the new generation of offsprings generated by the two variational operators: mutation and crossover.

A GA explores the solution space via mutation, which stochastically alters the individual's genome and ultimately translates into the introduction of new behaviors. Possible implementations of the mutation consider sampling the parameters from a uniform or a Gaussian distribution, centered around the parent's genome or at another, problem-specific, location. In parallel, the crossover operator combines a randomly selected fraction of genes of two or more individuals, resulting in one or multiple offsprings. Different types of crossover traditionally used for GAs and applicable to the NE setting are single point, M point (with $M \geq 2$), segment, uniform, and multiparent [17]. The effect of the two operators can also be combined by, for example, mutating the crossover offsprings. Lastly, in spite of the variations taking place on the genome level, they can be biased by providing information from the phenotypic space of behaviors and their fitness values. Section III will detail the current design's implementation of such biased operators.

Whereas crossover and mutation can be applied to any subset of the population, elitism practices may be considered for enhanced sample efficiency [17]. The percentile of policies with the highest fitness, the elites, contains an objectively better pool of genes for recombination. At the same time, mutating the nonelites would pose a lower risk of negatively impacting the average achievable fitness.

D. Hybrid Frameworks

Learning emerges from the tradeoff between exploitation (or quality seeking) and exploration (or novelty seeking) [22,41]. ERL, introduced by [31], implements this tradeoff as a combination of off-policy DRL and GA.

First, the genetic individual's genome is the flattened array of the agent's policy NN. ERL creates new individuals, or offsprings, through the M-point crossover of the genomes of two elites (i.e., within the fittest percentile) [31]. In ERL, the mutation operator samples the vector of policy weights from an isotropic Gaussian distribution centered at the parent's weights. The original algorithm and its future adaptions use a form of the Gaussian mutation due to its simplicity and flexibility [30,32].

The hybrid agent evaluates all actors sequentially 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.

Parallel to the genetic population, an actor/critic DDPG agent 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 [28,31]. Whereas the shared memory buffer enables off-policy exploration, the actor synchronization aims to decrease the sample complexity of the neural evolution as the newly injected NN has been updated more often via backpropagated gradients [30,31].

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 parameter space $\mathcal{D}_\theta$ of the policy [38,42]. Crucially, the distribution of the weight magnitude is not uniform across the network; hence, the output of each layer is normalized to evenly scale the effect of mutation [42].

PDERL developed by Bodnar et al. [28] 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 [28]. They aim to counteract catastrophic forgetting caused by Gaussian mutation and M-point crossover applied to a direct encoding of the network, a problem of the off-policy hybrid methods [28,30,32]. The current framework adapts both the crossover and mutation using dedicated memory buffers, mechanisms described in the next section.

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 learning framework centered around SERL and its specific crossover and mutation operators. 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 $f_s = 100$ Hz. The high-fidelity six-degrees-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) [43,44]. It has been validated through system identification on flight-test data recorded onboard a Cessna Citation II.[§]

Equation (4) depicts the complete modeled aircraft state with the longitudinal and lateral coordinates X_e and Y_e measured with respect to the trim location. To build the MDP formulation, the agent state vector $\boldsymbol{s} \in \mathbb{R}^7$, defined by Eq. (6), retrieves observed information from the aircraft state augmenting it with the current tracking error. Given that the agent state is a subset of the modeled state, the Markovian assumption shall be relaxed to a partially observable MDP (POMDP). The actions $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t) \in \mathbb{R}^3$ output by the behavioral policy are fed as the input vector at the same frequency f_s. Defined by Eq. (5), the actor's action corresponds to the deflection of the elevator, aileron, and rudder. The thrust control is delegated to an inner auto-throttle [11,43] while the trim tab and flap deflection stay zero. The control diagram from Fig. 1 shows the feedback loop between the nonlinear intelligent agent and the controlled plant:

[§]The PH-LAB research aircraft jointly owned between the Delft University of Technology Faculty of Aerospace Engineering and Netherlands Aerospace Centre (NLR) (https://cs.lr.tudelft.nl/citation/) [retrieved 11 January 2023].

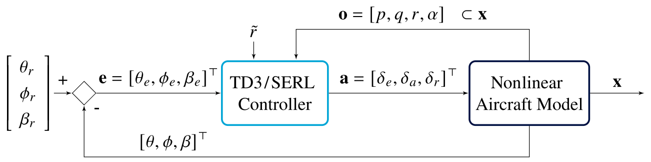

$\mathbf{o} = [p, q, r, \alpha] \subset \mathbf{x}$
$\tilde{r}$
$[\theta_r, \phi_r, \beta_r]$
+
-
$\mathbf{e} = [\theta_e, \phi_e, \beta_e]^\top$
TD3/SERL Controller
$\mathbf{a} = [\delta_e, \delta_a, \delta_r]^\top$
Nonlinear Aircraft Model
$\mathbf{x}$
$[\theta, \phi, \beta]^\top$

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

$$\boldsymbol{x} = [p, q, r, V_{tas}, \alpha, \beta, \theta, \phi, \psi, H, X_e, Y_e]^\top \tag{4}$$

$$\boldsymbol{a} \coloneqq [\delta_e, \delta_a, \delta_r]^\top \in \mathcal{A} \tag{5}$$

$$\boldsymbol{s} \coloneqq [\theta_e, \phi_e, \beta_e, p, q, r, \alpha]^\top \in \mathcal{S} \tag{6}$$

To account for realistic physical limits of the actuators, the space $\mathcal{A}$ is restricted for each channel bounded to $[\boldsymbol{a}_{\text{low}} = -10$ deg, $\boldsymbol{a}_{\text{high}} = 10$ deg]. The ranges are within the admissible deflections of the modeled control surfaces [45].

The goal of the attitude controller is to track reference signals. Following from [11] and [46], the θ_r and ϕ_r references are sequences of cosine-smoothed step signals with amplitudes uniformly sampled from the ranges: [−25 deg, 25 deg] for pitch and [−45 deg, 45 deg] for roll angle. The latter is motivated by the values specified in the CS-25 regulations [47]. The range for pitch is based on the previous work of [11,46,48]. The side-slip reference remains at zero.

Completing the POMDP formulation of optimal control, Eq. (7) defines the reward signal, and Eq. (8) defines the fitness value $F_i^{(T)}$ as the total reward the ith policy accumulates during one episode. It penalizes the L_1 norm of the clipped and scaled attitude tracking error, similar to the reward implemented by [11]. The scaling factor $\boldsymbol{c}_r = \frac{6}{\pi}[1, 1, 4]^\top$ accounts for the smaller magnitude of side-slip error. At the end of the trial, a sparse negative reward $\tilde{r}_T$ penalizes early crashes proportionally to the time left before the maximum episode length $T_{\max}$. Empirically, a value of $C_P = 20$ is deemed appropriate for the proportionally constant,

$$\tilde{r}_t \coloneqq \begin{cases} -\frac{1}{3}\left\|\text{clip}\left(\boldsymbol{c}_r \odot \boldsymbol{e}_t, -1, 0\right)\right\|_1 & t \in [0, T) \\ -\frac{C_P}{\Delta t}(T_{\max} - T) & t = T \end{cases} \tag{7}$$

$$F_i^{(T)} \coloneqq \sum_{t=0}^{T} \tilde{r}_t, \quad i = \overline{1, N} \tag{8}$$

B. Learning Framework

The Safety-Informed Evolutionary Reinforcement Learning referred to as SERL, extends PDERL from [28] by replacing the DDPG actor/critic with TD3 and the proximal mutation operator with the safety-informed mutation operator.

1. Architecture Overview

SERL(N) trains $N + 1$ controllers (a TD3 actor and the genetic population of N) with the same shape and size. Figure 2 depicts the components of SERL and the interaction between the RL agent (blue) and the GA operators (green). The environment (gray) simulates the repeated rewarded evaluations of the policies. Overall, the architecture boasts a modular design, where each component can be developed separately with overall potential gains.

The shared buffer trains the RL agent in an off-policy manner. The two critics are feedforward neural networks with different shapes and sizes than the genetic actors and TD3 policy, which have the same footprint.

Inheriting from ERL and PDERL, selection is done via repeated tournaments of size 3. The method is seen as appropriate due to its

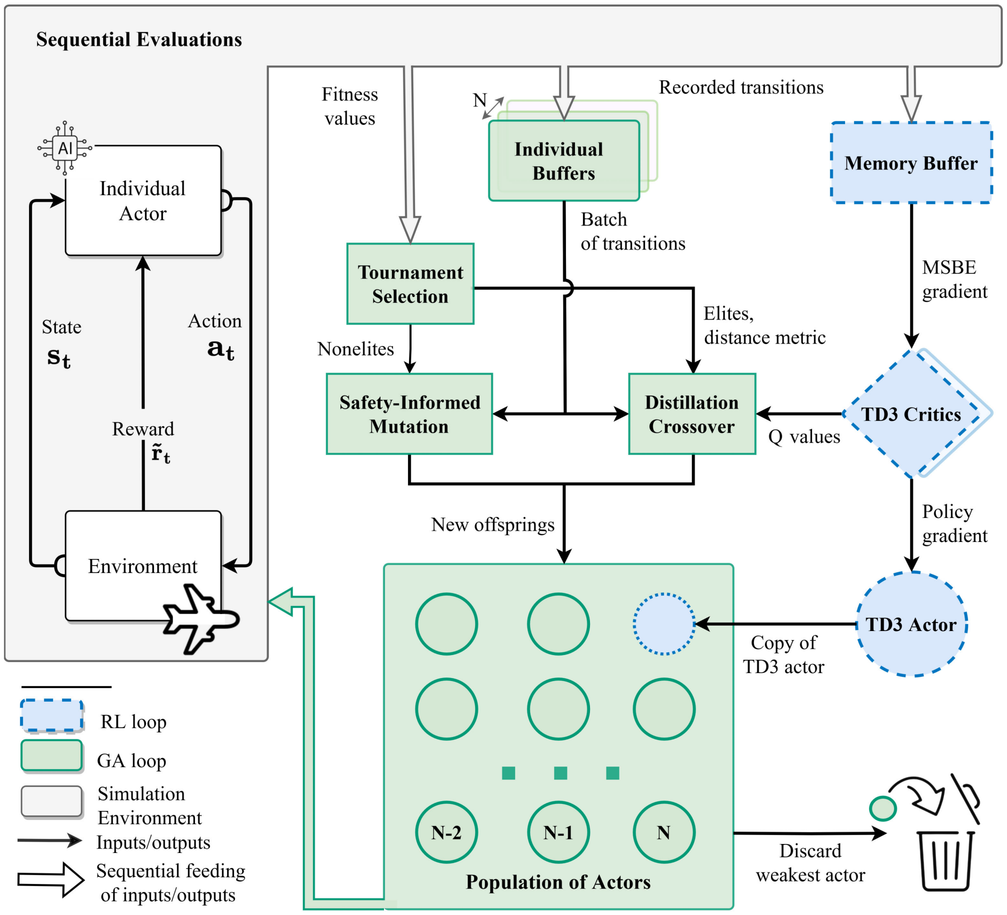


Fig. 2 High-level overview of the data flows through the SERL(N) framework. Inspired from similar diagrams shown in [28,31,32].

relatively low computational time and performance for small populations [18]. Genetic operators act once per generation, so the number of agent-environment interactions (steps) was chosen as the smallest common unit of training progress, and the termination criterion is based on the accumulated number of steps. In the beginning, episodes end prematurely, and so the number of steps per generation cannot be constant, but it is upper-bounded by the maximum episode length multiplied by the population size.

2. *Set of Genetic Memory Buffers*

To integrate both operators, SERL 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. Each genetic buffer can then store experiences that span multiple generations which are used by the distillation crossover and safety-informed mutation operators. The common buffer $\mathcal{B}$ shares experience with the individual genetic memories, but it does not equal their union.

3. *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 [28]. In contrast to M-point crossover, it acts on the phenotypical space and not on the genomes.

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 equal amounts of transitions 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, the authors of [28] introduce a cloning loss $\mathcal{L}^{(C)}$ which trains the offspring μ_o to minimize the L_2 loss from the parents' behaviors over a batch B:

$$\mathcal{L}_{\mu}^{(C)}(\theta_o, B) := \frac{1}{|B|}\sum_{i}^{|B|} \left\|\mu_o(\boldsymbol{s}_i)\right\|^2 + \left\|\mu_o(\boldsymbol{s}_i) - \mu_p(\boldsymbol{s}_i)\right\|^2,$$
$$\mu_p = \begin{cases} \mu_x & \text{if} \quad Q(\boldsymbol{s}_i, \mu_x(\boldsymbol{s}_i)) > Q(\boldsymbol{s}_i, \mu_y(\boldsymbol{s}_i)) \\ \mu_y & \text{else} \end{cases} \quad \text{and} \quad B \sim \mathcal{B}^o \tag{9}$$

The operator uses gradient updates to bias the offspring's phenotype toward the estimated best-performing parent. The term $\|\mu_o(\boldsymbol{s}_i)\|^2$ simply ensures that the backpropagated gradients of the policy loss will not saturate the nonlinear activation functions [28].

Lastly, the pairing metric from Eq. (10) decides which elites are going to be crossed. It uses the Euclidean distance between the average actions taken by the agents [28]. It samples a batch of states from each parent genetic memory with $\mathcal{P}_{x,y}$ the corresponding state-visitation probability,

$$d(\mu_x, \mu_y) := \mathbb{E}_{\boldsymbol{s}\sim\mathcal{P}_x}\left[\|\mu_x(\boldsymbol{s}) - \mu_y(\boldsymbol{s})\|^2\right] + \mathbb{E}_{\boldsymbol{s}\sim\mathcal{P}_y}\left[\|\mu_x(\boldsymbol{s}) - \mu_y(\boldsymbol{s})\|^2\right] \tag{10}$$

The probability of selecting a pair increases with the distance $d(\mu_x, \mu_y)$ between them. Distance-based selection strategy incentivizes the introduction of new behaviors, developing a phenotypically diverse population.

4. *Safety-Informed Proximal Mutation*

SERL retrieves the Gaussian mutation operator from the original ERL, but it adapts it for the flight control task. Let one consider the parent to be mutated μ_p, parameterized by θ_p with genetic memory $\mathcal{B}_p$. The genome of the offspring μ_o is sampled from an isotropic Gaussian distribution with the mean μ_p and variance σ_{mut} (mutation intensity).

Then, 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. Equation (11) computes the policy sensitivity s_{μ_p} as the root of the squared gradient summed over the action space:

$$\theta_o \sim \mathcal{N}\left(\theta_p, \frac{\sigma_{\text{mut}}}{s} I\right) \quad \text{with} \quad s := \sqrt{\sum_k^{|\mathcal{A}|}\left(\sum_i^{|B_M|} \nabla_{\boldsymbol{\theta}}\mu_{\boldsymbol{\theta}}(\boldsymbol{s}_i)\right)_k^2},$$
$$B_M \sim \mathcal{B}^p \tag{11}$$

The higher the magnitude of the policy gradient concerning a parameter, the smaller the mutation step becomes. Essentially, sensitive NN parameters are perturbed less by the scaled update. The offspring policy has its parameters within the proximity of its predecessor, thus limiting behavioral divergence, which was shown to lead to catastrophic forgetting [32,49].

However, for the flight control applications, to take into account existing safety-related domain knowledge, a safety-informed mutation operator is adapted from [32]. Subsequently, the gradients are estimated on batches B_C sampled from its parent *critical buffer* $\mathcal{B}_C \subset \mathcal{B}$. This buffer contains the transition tuples associated with a high value of a cost function, directly related to safety knowledge on the maneuver.

Equation (12) defines the SERL critical buffer as a subset of transitions that cause the next state to enter a nonsafe region of the flight envelope. These are marked by entering prestall dynamics at an angle of attack higher than 11 deg or by having the absolute value of the roll angle larger than 60 deg. The latter was empirically observed to degenerate in an unstable spiral motion,

$$\mathcal{B}_C := \{\langle \boldsymbol{s}_t, \boldsymbol{a}_t, \tilde{r}_t, \boldsymbol{s}_{t+1}\rangle \in \mathcal{B} | \quad \alpha_{t+1} \geq 11 \text{ deg} \quad || \quad |\phi_{t+1}| \geq 60 \text{ deg}\} \tag{12}$$

The safety-informed mutation hijacks the effect of the proximal operator and directs exploration toward the less-sensitive regions of the parameters space. Essentially, the offspring's genome will evolve less in the directions that could result in critical transitions, in other words, unsafe behaviors. Nevertheless, because the operator scales the additive parameter noise, unsafe exploration is only discouraged and not prohibited; hence, the offspring does 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. (13). 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 toward smoother control actions,

$$\mathcal{L}_{\mu}^{(\text{CAPS})} := \lambda_S \cdot \underbrace{\|\mu_\theta(\boldsymbol{s}) - \mu_\theta(\tilde{\boldsymbol{s}})\|_2}_{\mathcal{L}_S} + \lambda_T \cdot \underbrace{\|\mu_\theta(\boldsymbol{s}) - \mu_\theta(\boldsymbol{s}_{t+1})\|_2}_{\mathcal{L}_T},$$
$$\tilde{\boldsymbol{s}} \sim \mathcal{N}(\boldsymbol{s}, \sigma I) \tag{13}$$

Because of their inherent nonlinearity, multilayer NN 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. (14) 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

Table 1 Hyperparameters of the DRL (first group), GA (second), and their interaction (third), used training the TD3, SERL(10) and SERL(50)

Parameter Name	Symbol	TD3	SERL(10)	SERL (50)
Learning rate (actor and 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)
Nonlinear activation	—	ReLU	tanh	tanh
Exploratory noise magnitude	σ	0.33	0.29	0.23
Population size	N	N.A.	10	50
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
Policy evaluations per epoch	E	10	10	5
Steps for training	—	10^6	10^6	5×10^6

ReLU, Rectified Linear Unit.

Table 2 Evaluation cases for fault-tolerance (F) and robustness (R)

Case	Title	Description
-	Nominal	Unchanged from training: $H = 2{,}000$ m, $V_{tas} = 90$ m/s
F1	Iced wings	Maximum angle of attack reduced by 30% and the drag coefficient increased with 0.06
F2	Shifted center of gravity	Aircraft center of gravity shifted aft by 0.25 m
F3	Saturated aileron	Aileron deflection clipped at ± 1 deg
F4	Saturated elevator	Elevator deflection clipped at ± 2.5 deg
F5	Broken elevator	Elevator effectiveness coefficients multiplied by 0.3
F6	Jammed actuator	Rudder stuck at a deflection of 15 deg
R1	High dynamic pressure	Trim setting at $H = 2{,}000$ m, $V_{\text{tas}} = 150$ m/s
R2	Low Dynamic Pressure	Trim setting at $H = 10{,}000$ m, $V_{\text{tas}} = 90$ m/s
R3	Disturbance and biased cesensor noise	Additive sensor noise and vertical wind gust of 15 ft/s acting for 3 s

trails, and the square root scales down the value, making the unit of the Sm metric equal to rad/s. 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. (7),

$$Sm := -\frac{C_{S_m}}{T} \cdot \sqrt{\frac{2}{n} \sum_{a=1}^{|A|} \sum_{k=1}^{n/2} \left(\frac{1}{n} \left| A[k] \right|^2 \cdot f_k \right)_a}, \quad n = \frac{T}{f_s} \tag{14}$$

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 multidimensional spaces. In contrast, any contentious policy can be evaluated and optimized using the above-defined smoothness value.

For the training scenario described by Sec. III.D, the fitness is defined according to Eq. (8). Additionally, an ablation study investigates the effect of CAPS regularization in comparison to Sm optimization. The latter is obtained by adding the Sm term from Eq. (14) 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, using the same set of references for all agents enables a valid comparison.

D. Training Task

For the training environment, the aircraft model is trimmed for steady straight symmetric flight at $V_{\text{tas}} = 90$ 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 $T_{\max} = 20$ s, corresponding to at most 200 agent-environment interactions. One training epoch represents one generation during which all policies are evaluated over E episodes. The genetic operators act once per generation, while TD3 takes $E \times T$ gradient steps to compensate for the transitions accumulated over the epoch. At the end of the generation, the actor synchronization replaces the policy with the lowest fitness.

The training routine of both TD3 and SERL considers five distinct initializations, each with its own random seed used to randomly initialize the actors, critics, and reference signals.

For SERL, choosing the episode length is a tradeoff between actors receiving enough samples to explore and the training computational complexity. Longer episodes would increase the time required for the neuroevolutionary loop. TD3 or SERL(10) learns from 10^6 steps (i.e., interactions) which require approximately 200 min on eight CPU cores, Intel Xeon E5-1620 3.50 GHz.[¶] SERL(50) trains for 5×10^6 steps, proportionally higher to account for the additional learning actors. The wall-clock time also scales linearly with the genetic population size due to the larger number of evaluations. Tournament selection, mutation, and crossover add a small overhead relative to smaller population sizes, while the time needed for gradient backpropagation does not increase.

The first group of hyperparameters from Table 1 configures the TD3 agent, whereas the second sets the GA loop. Among them, the mutation magnitude σ_{mut} has the highest impact on exploration as it directly regulates the amount of noise added to the genome. The last group controls the interaction between the two optimization methods. The synchronization rate equals the generational frequency of copying the TD3 policy weights.

E. Evaluation Tests

The policies optimized by each agent are tested on multiple scenarios, listed in Table 2. They are selected from the surveyed

[¶]GPU parallelization was tested, but due to relative small NN and batch sizes, it did not speed up the training.

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 $T_{\max} = 80$ s. Each test case loops through the same list of pseudorandom 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. Because 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 [11,48].

The last case tests the capacity to reject an external disturbance in the presence of realistic zero-mean 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 side slip. The normalization interval is $[-25\ \deg, 25\ \deg]$ for the pitch and yaw channels and $[-1\ \deg, 1\ \deg]$ for side slip, as its response is expected to remain in the neighborhood of 0.

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 samples.

A. Learning Curves

The training curves from Fig. 3 depict learning progress by plotting the average episodic reward obtained by each policy alongside the total number of time steps the policies expired interacting with the environment. All agents converge toward returns, comparable to values reported by [11,46,48]. After training, none of the polices ends the episode prematurely.

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 selecting TD3 slowly decreases, marking the switch to genetic mechanisms that allow for long-term performance.

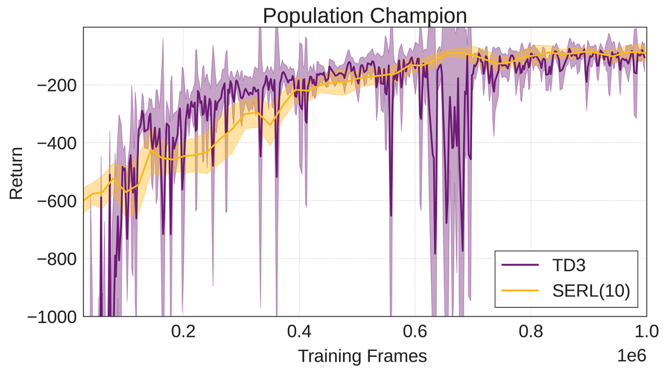
Empirically, it has been observed that N is the hyperparameter with the highest impact on training, directly altering the balances between the GA and TD3 learning. A larger population increases the number of frames experienced between RL updates, and it reduces the number of transitions corrupted with 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.

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 case F5, SERL(50) scores 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% ($p = 2 \times 10^{-35}$) and by 6.1% ($p = 8 \times 10^{-13}$), respectively. The F6 NMAE scores are significantly higher than the rest, so 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 it. 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 nonlinear 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.



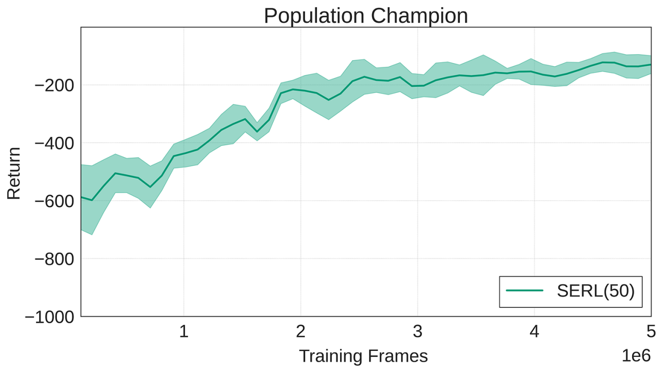


Fig. 3 Learning curves of the average champion reward (solid) and its standard deviation (shading) for TD3, SERL(10), and SERL(50). Statistics are computed over five evaluations.

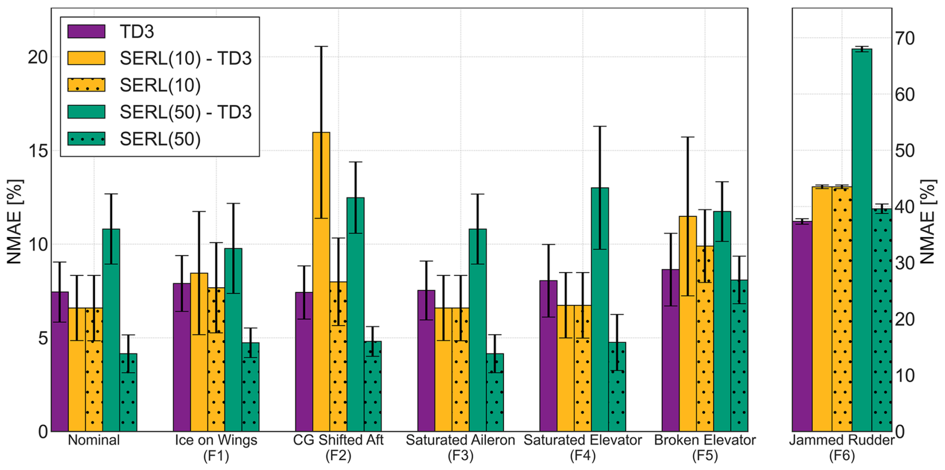


Fig. 4 Tracking NMAE and its standard deviation for each fault case. The hatched bars denote the SERL champion and the solid-colored bars the TD3 actors trained alone or within SERL (CG, Centre of gravity).

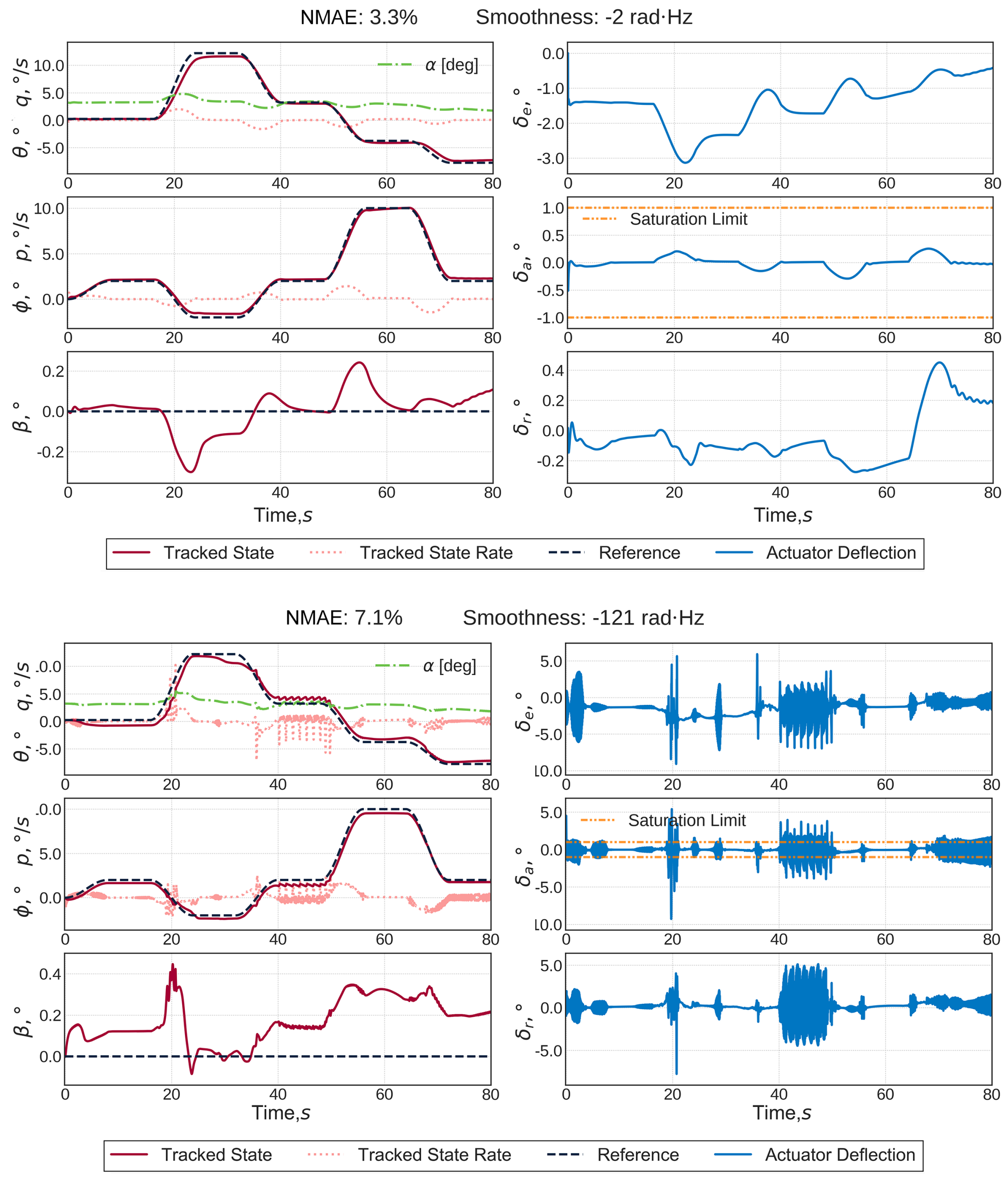


Fig. 5 Evaluation time traces of the SERL(50) champion (top) and TD3 policy (bottom) for the damaged aileron (F3). The interrupted line shows the aileron saturation limit.

1. *Saturated Aileron (F3): Downside of Aggressive Actions*

Figure 5 depicts the responses of the SERL(50) champion and the TD3 actor when tasked to control a plant with a clipped actuator.

Large actuating bounds during training allow the policies to exploit them and behave 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 side-slip references with the same accuracy as in the nominal case. It obtains an NMAE tracking error lower by 45% ($p = 4 \times 10^{-10}$) than its TD3-only counterpart.

The TD3 actor behaves more aggressively across all actuating channels, with high-frequency commands and large deflections. At $t = 20$ s, the roll angle reference shifts from 3 to -3 deg, prompting the TD3 actor to command a high aileron deflection. TD3 actor tries to push further the actuator bound, unaware of the fault. Thus, the commanded δ_a spikes nearly to its limit at $\delta_{a_{low}} = -10$ deg. Arguably, this unnecessary and suboptimal behavior stems from a preference for aggressive actions learned by the TD3 agent.

2. *Jammed Rudder (F6): Lost Achievable Novelty*

When the rudder is stuck at $\delta_r = 15$ deg, a large and relatively constant side-slip error raises the NMAE, and 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) and the TD3 actor (bottom). One can argue that, when one control channel is completely blocked, a significant amount of the behavioral diversity, encoded by the evolved population, has been lost. Here, training multiple controllers 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 side-slip deviation, the TD3 actor does not. The explanation stems from the stronger coupling between the rudder and aileron channel of the TD3-only policy.

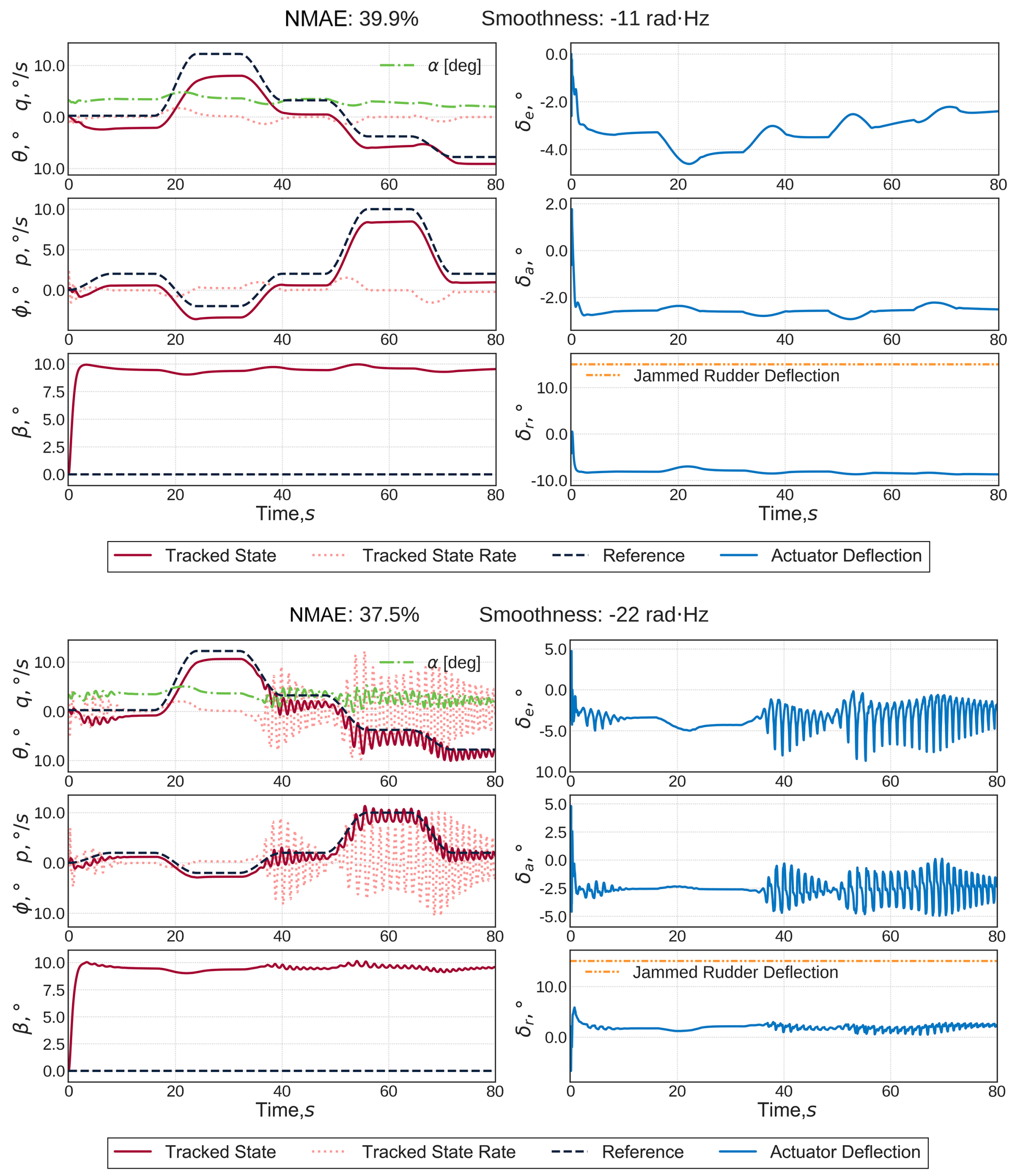


Fig. 6 Evaluation time traces of the SERL(50) champion (top) and TD3 policy (bottom) for the jammed rudder fault (F6). The interrupted line shows the real rudder deflection.

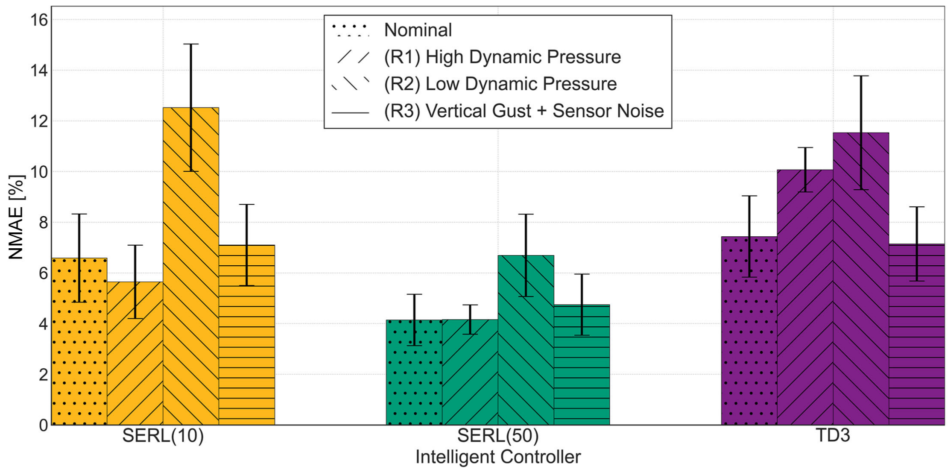


Fig. 7 Average NMAE and standard deviation for SERL(10), SERL(50), and TD3 controller. Hatching denotes each of the evaluation cases.

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 experiences 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 suboptimal 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 [48] for a comparable but 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. It contrasts with the effect it has on online RL methods that use locally identified dynamics models which suffer from poor robustness against [45]. Moreover, the policies can reject a 15 ft/s vertical wind gust. The SERL champion policy remains the same for the nominal case, hinting that, in contrast to the faulty scenarios, phenotypic diversity is not necessary to achieve superior robustness to atmospheric disturbance.

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

D. Smoothness of Control Action Signals

In the time traces shown by the previous subsections, 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 high-frequency components present in the actuating signals.

1. 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 study on robustness and the surveyed literature:

1) Using (deep) neural networks as optimal policy function approximators: Multilayered NNs have a significant number of degrees of freedom, justifying their popularity in learning complex nonlinear dynamics [7]. Thus, NNs can learn any frequency component within the training signal. Without being biased toward them, this also includes the high-frequency components. Nevertheless, the frequency-domain analysis of neural network controllers is 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 [43]. Hence, they are not present in 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 reward or fitness signals, the agent could optimize the policy toward containing it for small increments 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 action smoothness. The analysis presented by [15] has empirically shown that on-policy methods or stochastic policy algorithms are less prone to training noisy control policies. So far, methods using the NE mechanism have not been assessed in the field literature.

The presence or absence of one or more of these factors dictates the noise phenomenon. Subsequently, one can hypothesize that NN policies optimized via GAs output smoother control actions compared to their counterparts, trained via off-policy DRL.

Whereas the first two factors do not directly generate action noise, they enable it. Moreover, both SERL and TD3 could be affected by either of them; the only difference comes from their exploratory strategies. Nevertheless, the TD3 information is constantly copied into the genetic population, and individuals are therefore trained with its exploratory strategy. If tournament selection places it among the elites, its noise behavior would spread through the population.

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 [11,48], low-pass filtering, or discretizing the output signals obtains the smoothing effect [13]. These methods change the learning to partially remove the second factor.

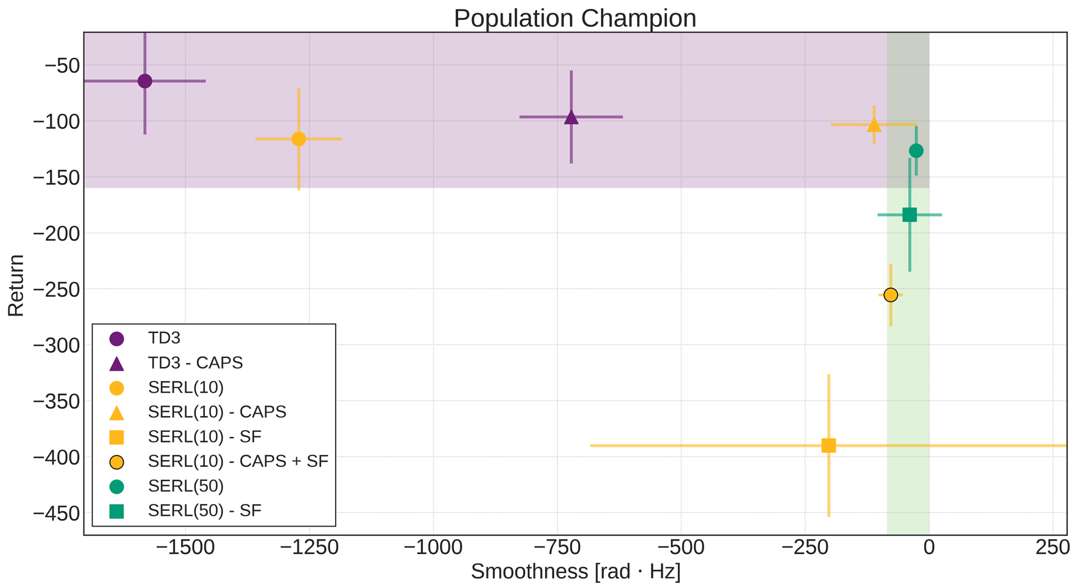


Fig. 8 Champion return vs average smoothness for three agents trained with and without CAPS regularization and a smoothness optimization term.

2. *Training for Smoothness Versus Performance*

The noisy-behavior hypothesis is investigated through an ablation study on the fitness and policy objective functions. Because the learning environment cannot be changed without modifying the task, one shall look at ways to adjust the first and third factors.

By design, SERL has two options to optimize for a smooth control policy, namely, using CAPS for the TD3 actors and adding the smoothness objective to the fitness function. The same three agents are considered: TD3, SERL(10), and SERL(50). Each learns with and without CAPS regularization and smoothness terms added to the objective and fitness function, resulting in ten training configurations. For SERL(50), we can omit the CAPS configuration as, in the large population scenario, the TD3 actor is mostly discarded during selection.

The study compares the average smoothest of the population in each training scenario to identify and isolate the effect of the NE mechanism on smoothest and performance. Figure 8 summarizes the gathered results as a scatter plot of the episodic return of the champion versus the population average smoothness.

Increasing the population size affects performance and control smoothness at different scales. The SERL(50) agent obtains a reward comparable with the CAPS-regularized TD3 and SERL(10), but it maintains a higher and almost constant policy smoothness at $Sm = -5 \pm 1$ rad $\cdot$ 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 shown in Sec. IV.B, with an $Sm = -1500 \pm 123$ rad $\cdot$ Hz. The effect of both CAPS and smooth fitness is visible, significantly shifting the Sm metric toward the right.

Adding the Sm term to the fitness function improves control smoothness, but it also reduces the selective pressure for performance. Hence, when the TD3 actor is not regularized, it rarely wins tournaments against other policies. In practice, this results in a significantly higher spread of the SERL(10)-SF compared to the SERL(10)-CAPS + SF. The smoothness of SERL(50) is not improved by either CAPS or smoothness-oriented fitness, hinting that the population already learns smooth behaviors via the NE mechanisms and benefits from TD3 injection only in the incipient part of the training. For SERL(10)-SF, the loss caused in performance by adding the smoothness term can only be partially mitigated by CAPS, the size of the population being too small for sample-efficient mutation and crossover.

Thus, based on the ablation study, one can argue that the developed TD3 controller learns noisier policies in comparison to the SERL as a direct consequence of combining the three factors. Meanwhile, the NE optimization 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, decoupling learning from the noisy transitions. The resulting behaviors propagate through the genetic population via distillation crossover. Thus, SERL trains performant controllers (controllers with high tracking permanence) in the large-population configuration without deteriorating the average smoothness, validating the hypothesis.

The standard deviation is an imperfect estimator of the population distribution. The Sm distribution shows skewness toward 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 copied during recent actor synchronizations. The performance metric distribution is more symmetrically spread through the population.

3. *Applications of Smoothness-Performance Tradeoff*

Understating the tradeoff between performance and action smoothness is relevant for bridging the gap between simulation training and the hardware. Low-pass filtering the commends of pretrained policies can mitigate the effects of noise, but it might unpredictably degrade the performance of nonlinear controllers. Subsequently, such filters will require extensive tuning for multiple operating conditions via 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 [11,46].

Looking back at Fig. 8, two distinct regions emerge. Applications such as drone racing prioritize short-term performance so they will 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. The right-hand training regime would therefore be preferred for these applications.

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

V. Conclusions

SERL, a safety-informed Evolutionary Reinforcement Learning method, combines DRL and neuroevolution via GA to train flight controllers on a nonlinear fixed-wing aircraft model.

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 unregularized multilayer neural networks was shown to aggravate the control action noise. This happens in simulation environments that only prioritize tracking performance but can be completely alleviated via parameter-space exploration through neuroevolutionary operations.

Training a large controller population required proportionally more samples but achieved significantly higher fault tolerance than a comparable state-of-the-art TD3 agent in five out of six evaluation cases. SERL's performance also remained robust to changes in flight conditions and external disturbances in the presence of realistic observation noise. This demonstrated the generalization benefit of evolving a phenotypically diverse population of controllers through NE combined with RL. Nevertheless, when system faults collapse the offline-obtained diversity, TD3 remains marginally superior.

To build on top of the current research, it is recommended to orient toward real-world applicability. As a first step, a heuristic should be designed to adapt or select online the behavioral policy from the SERL-trained pool of actors. The discrete decision problem can be formulated as short-term receding horizon optimization. Then, online system identification can model the local dynamics (e.g., an incremental model such as the one used in [52] to predict the one-step-ahead state or the agent's behavior such as done in [23]) and estimate each actor's performance over the receding horizon. A new champion policy is selected based on performance ranking or tournament selection, and the behavioral policy is shifted toward it. Ultimately, this represents a step in validating artificial intelligence within autonomous fault-tolerant controllers, making them a safety net applicable to safety-critical systems in general and to aircraft in particular.

Appendix: Effect of Mutation on Learning

This section's comparative study investigates the relative performance contribution of the safety-informed mutation. Three SERL(10) agents have trained on the same task and with the same hyperparameters described in Sec. III, each incorporating one of the operators: safety informed (ours), proximal mutation from [28], and the simple Gaussian mutation used by original ERL [31].

The performance variation across the population correlates with its robustness. In the left-hand side of Fig. A1, the learning history is compared for the three operators. Mutating via the safety-informed operator ultimately obtains a converged performance better than both proximal and Gaussian mutations ($p = 10^{-4}$). Its learning progress is more stable learning, with a lower chance of catastrophic forgetting due to less abrupt random parameter disturbances. A more performance-robust operator positively affects the maximum performance of the champion, as visible in the right-hand side plot where the safety-informed mutation outscores significantly the proximal one ($p = 3.6 \cdot 10^{-4}$). Safety-informed exploration directs learning away from hard-to-learn experiences, developing policies with smoother loss landscapes. These policies benefit from enhanced sample efficiency and converging toward better controllers.

References

[1] ICAO, "Safety Report," TR, International Civil Aviation Organization, 2020, https://www.icao.int/safety/iStars/Pages/Accident-Statistics.aspx [retrieved 5 March 2023].

[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, Dec. 2016, pp. 126–141. https://doi.org/10.1016/j.conengprac.2016.09.010

[3] Edwards, C., Lombaerts, T., and Smaili, H., *Fault Tolerant Flight Control*, Lecture Notes in Control and Information Sciences, Vol. 399, Springer–Verlag, Berlin, April 2010, pp. 1–560. https://doi.org/10.1007/978-3-642-11690-2

[4] Lavretsky, E., and Wise, K. A., "Robust Adaptive Control," *Robust and Adaptive Control: With Aerospace Applications*, Springer–Verlag, Berlin, 2012, pp. 317–353. https://doi.org/10.1007/978-1-4471-4396-3

[5] 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. https://doi.org/10.1038/nature16961

[6] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.

[7] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, MA, 2016, http://www.deeplearningbook.org [retrieved 5 March 2023].

[8] 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," 2019, https://arxiv.org/abs/1509.02971 [retrieved 5 March 2023].

[9] 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/10.1038/nature14236

[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)*, Inst. of Electrical and Electronics Engineers, New York, 2022, pp. 1561–1570. https://doi.org/10.1109/ICUAS54217.2022.9836052

[11] Dally, K., and Van Kampen, E.-J., "Soft Actor-Critic Deep Reinforcement Learning for Fault Tolerant Flight Control," *AIAA Scitech 2022 Forum*, AIAA Paper 2022-2078, 2022. https://doi.org/10.2514/6.2022-2078

[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)*, Inst. of Electrical and Electronics Engineers, New York, 2019, pp. 523–533. https://doi.org/10.1109/ICUAS46274.2019

[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*, Vol. 46, No. 7, 2023, pp. 1–13. https://doi.org/10.2514/1.G007335

[14] Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., and Scaramuzza, D., "Champion-Level Drone Racing Using Deep

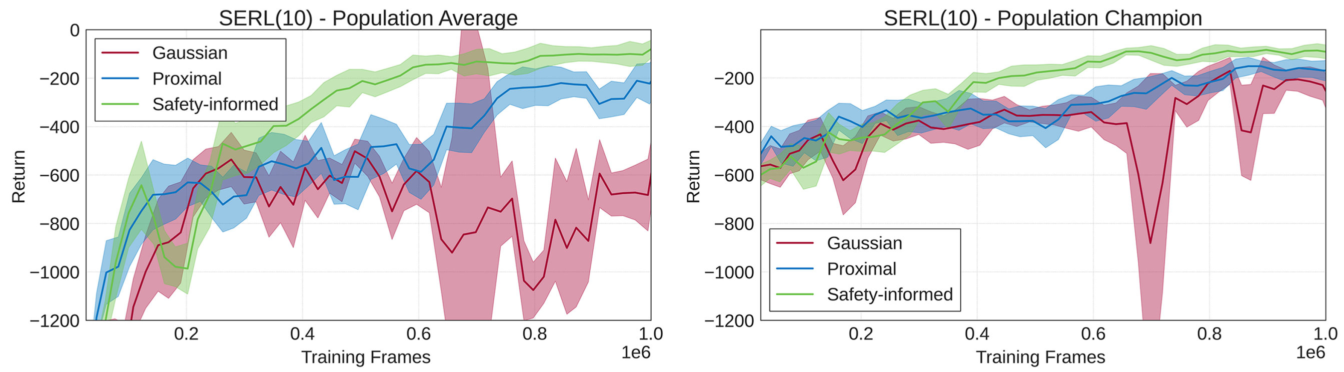


Fig. A1 Learning curves of the average over the actor population (left) and champion (right) trained with each of the three mutation operators. The reward statistics are calculated over 30 evaluation runs at each training step.

Reinforcement Learning," *Nature*, Vol. 620, No. 7976, 2023, pp. 982–987.
https://doi.org/10.1038/s41586-023-06419-4

[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)*, Inst. of Electrical and Electronics Engineers, New York, 2021, pp. 1810–1816.
https://doi.org/10.1109/ICRA48506.2021.9561138

[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, March 2015, p. 4.
https://doi.org/10.3389/frobt.2015.00004

[17] Simon, D., *Evolutionary Optimization Algorithms*, Wiley, Hoboken, NJ, 2013.

[18] Koza, J. R., and Poli, R., "Genetic Programming," *Search Methodologies*, Springer–Verlag, Berlin, 2005, pp. 127–164.

[19] Zhang, B.-T., and Muhlenbein, H., "Evolving Optimal Neural Networks Using Genetic Algorithms with Occam's Razor," *Complex Systems*, Vol. 7, No. 3, 1993, pp. 199–220.

[20] Papavasileiou, E., Cornelis, J., and Jansen, B., "A Systematic Literature Review of the Successors of 'Neuro Evolution of Augmenting Topologies'," *Evolutionary Computation*, Vol. 29, No. 1, 2021, pp. 1–73.
https://doi.org/10.1162/evco_a_00282

[21] Stanley, K. O., and Miikkulainen, R., "Evolving Neural Networks Through Augmenting Topologies," *Evolutionary Computation*, Vol. 10, No. 2, 2002, pp. 99–127.
https://doi.org/10.1162/106365602320169811

[22] 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.
https://doi.org/10.1038/nature14422

[23] 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.
https://doi.org/10.1145/3596912

[24] 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*, Assoc. for Computing Machinery, New York, 2020, pp. 94–102.
https://doi.org/10.1145/3377930

[25] 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.
https://doi.org/10.2514/3.20898

[26] 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.
https://doi.org/10.2514/1.G000376

[27] Emami, S. A., Castaldi, P., and Banazadeh, A., "Neural Network-Based Flight Control Systems: Present and Future," *Annual Reviews in Control*, Vol. 53, Jan. 2022, pp. 97–137.
https://doi.org/10.1016/j.arcontrol.2022.04.006

[28] Bodnar, C., Day, B., and Lió, P., "Proximal Distilled Evolutionary Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34, No. 4, April 2020, pp. 3283–3290, https://ojs.aaai.org/index.php/AAAI/article/view/5728.
https://doi.org/10.1609/aaai.v34i04.5728

[29] Pourchot, A., and Sigaud, O., "CEM-RL: Combining Evolutionary and Gradient-Based Methods for Policy Search," 2019.
https://doi.org/10.48550/arXiv.1810.01222

[30] Sigaud, O., "Combining Evolution and Deep Reinforcement Learning for Policy Search: A Survey," *ACM Transactions on Evolutionary Learning*, Vol. 3, No. 3, 2022, pp. 1–20.
https://doi.org/10.1145/3569096

[31] Khadka, S., and Tumer, K., "Evolution-Guided Policy Gradient in Reinforcement Learning," *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Vol. 31, Curran Assoc., 2018.
https://doi.org/10.48550/arXiv.1805.07917

[32] Marchesini, E., Corsi, D., and Farinelli, A., "Exploring Safer Behaviors for Deep Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, No. 7, June 2022, pp. 7701–7709, https://ojs.aaai.org/index.php/AAAI/article/view/20737.
https://doi.org/10.1609/aaai.v36i7.20737

[33] Bertsekas, D., *Reinforcement Learning and Optimal Control*, Athena Scientific Optimization and Computation Series, Athena Scientific, Belmont, MA, 2019, http://www.mit.edu/~dimitrib/RLbook.html [retrieved 5 March 2023].

[34] 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.
https://doi.org/10.48550/arXiv.1802.09477

[35] 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.
https://doi.org/10.48550/arXiv.1801.01290

[36] Thrun, S., and Schwartz, A., "Issues in Using Function Approximation for Reinforcement Learning," *Proceedings of the 1993 Connectionist Models Summer School*, Vol. 6, Lawrence Erlbaum, Hillsdale, NJ, 1993, p. 263.

[37] 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.
https://doi.org/10.1016/0041-5553(64)90137-5

[38] 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.
https://doi.org/10.48550/arXiv.1703.03864

[39] Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R., "Designing Neural Networks Through Neuroevolution," *Nature Machine Intelligence*, Vol. 1, No. 1, 2019, pp. 24–35.
https://doi.org/10.1038/s42256-018-0006-z

[40] Stanley, K., and Miikkulainen, R., "Efficient Evolution of Neural Network Topologies," *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02 (Cat. No. 02TH8600)*, Vol. 2, Inst. of Electrical and Electronics Engineers, New York, 2002, pp. 1757–1762.
https://doi.org/10.1109/CEC.2002.1004508

[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, July 2016, p. 40.
https://doi.org/10.3389/frobt.2016.00040

[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," 2018.
https://doi.org/10.48550/arXiv.1706.01905

[43] 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, TR, Delft Univ. of Technology, 1998, http://resolver.tudelft.nl/uuid:9bfb1f68-2f7c-4f32-91b4-79297d295f84

[44] 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–Verlag, Berlin, 2018, pp. 259–277, http://resolver.tudelft.nl/uuid:c0a57513-38b7-4d3a-898c-fa57c3e7ac2e [retrieved 5 March 2023].

[45] 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*, AIAA Paper 2021-0883, 2021, http://resolver.tudelft.nl/uuid:b37cefbf-e353-43cf-8d9d-98f2653216c6

[46] Seres, P., Liu, C., and van Kampen, E.-J., "Risk-Sensitive Distributional Reinforcement Learning for Flight Control," *International Federation of Automatic Control*, Vol. 56, No. 2, 2023, pp. 2013–2018.
https://doi.org/10.1016/j.ifacol.2023.10.1097

[47] EASA, "Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes (CS-25)," TR, European Union, June 2021, https://www.easa.europa.eu/en/document-library/certification-specifications/cs-25-amendment-27 [retrieved 5 March 2023].

[48] Teirlinck, C., and Van Kampen, E.-J., "Reinforcement Learning for Flight Control: Hybrid Offline-Online Learning for Robust and Adaptive Fault-Tolerance," TR, Delft Univ. of Technology, The Netherlands, 2022, http://resolver.tudelft.nl/uuid:dae2fdae-50a5-4941-a49f-41c25bea8a85 [retrieved 5 March 2023].

[49] 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.
https://doi.org/10.1145/3205455

[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*, AIAA Paper 2018-0385, 2018.
https://doi.org/10.2514/6.2018-0385

[51] Moorhouse, D. J., and Woodcock, R. J., Background Information and User Guide for MIL-F-8785C, Military Specification-Flying Qualities of Piloted Airplanes," Air Force Wright Aeronautical Lab., TR-ADA119421, Wright–Patterson AFB, 1982.

[52] Zhou, Y., Kampen, E.-J. V, and Chu, Q., "Nonlinear Adaptive Flight Control Using Incremental Approximate Dynamic Programming and Output Feedback," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2016, pp. 493–496.
https://doi.org/10.2514/1.G001762