Reinforcement-learning-inspired Autonomous Flight Envelope Protection

Aristeidis Antonakis Research Engineer, ONERA, DTIS, 31055, Toulouse, France.
aristeidis.antonakis@onera.fr

Sofiane Kraïem Research Engineer, ONERA, DTIS, 31055, Toulouse, France.
sofiane.kraiem@onera.fr

ABSTRACT

In aerospace engineering, many systems exhibit strongly nonlinear, divergent, or unstable behavior when driven outside their nominal operating regimes: Their control design has to rely upon conservative assumptions and dedicated envelope protection logics restricting operation within regions with substantially linear dynamics. This often leads to a laborious, expensive, and time-consuming development process. In this paper, we propose a reinforcement-learning-inspired technique for envelope protection aimed explicitly at reducing the required designer workload by removing the requirement to explicitly define envelope limits. Our method comprises three key elements: (1) Neural nets for data-only dynamic identification, (2) a novel method for exploiting the neural model's parametric uncertainty to generate gradients that contain the dynamics within the known, safe envelope and, finally, (3) a Temporal Backpropagation (TB) calculation which converts the resulting optimal control problem to one of training of a deep, recurrent neural net architecture. The method's performance is assessed in simulated experiments: A simple numerical case demonstrates the algorithm's key characteristics. Finally, testing on a 6-DoF aircraft simulator evaluates the effectiveness of a TB-based Flight Envelope Protection (FEP) in flight sequences including aggressive pilot inputs and highly nonlinear post-stall aerodynamics.

Keywords: Flight Envelope Protection; Reinforcement Learning; Uncertainty Propagation; Neural networks

Nomenclature

AoA	=	Angle-of-Attack
c_i	=	total cost/reward at step i
$C = \{(X_i, U_i, X_{i+1})\}$	=	dataset of sampled state transitions
C^*	=	admissible domain defined by ellipsoid containment
ΔS	=	difference matrix $S^*_{Y\|\theta} - \tilde{S}_{Y_i\|\theta}$
D	=	number of system states
dl, dm, dn	=	aileron, elevator, rudder command
E_{U_i}	=	trajectory-dependent error term in gradient wrt U_i
$E_{\tilde{X}_i}$	=	error contribution from state $\tilde{X}_i$ to future outputs
F	=	performance objective function
G	=	quadratic penalty constraint function

λ_i, q_i	=	eigenvalue and corresponding unit eigenvector of ΔS
M	=	surrogate model (emulator) of system dynamics
N	=	prediction horizon length (number of steps)
S_θ	=	covariance of parameters θ
$S^*_{Y\|\theta}$	=	bounding covariance ellipsoid (reference)
$\tilde{J}_{Y_i\|\theta}$	=	Jacobian of $\tilde{Y}_i$ w.r.t. θ
$\tilde{S}_{Y_i\|\theta}$	=	output covariance due to parameter uncertainty
$\tilde{X}_i, \tilde{Y}_i$	=	emulator-predicted state and output
U_i	=	system input at time step i
$\vec{V}_b$	=	body velocity vector
w_d	=	discount factor for planning horizon
X_i	=	system state at discrete time step i
$\vec{x}_b$	=	x-body axis vector
Y_i	=	system output at time step i
$\vec{z}_b$	=	z-body axis vector
β	=	sideslip angle
θ	=	parameters of the surrogate model M
ψ, θ, ϕ	=	Euler angles (heading, pitch, roll)
$\vec{\omega}_b$	=	body angular rate vector $\vec{\omega}_b = [\omega_x, \omega_y, \omega_z]$

1 Introduction

In aerospace engineering, many systems exhibit strongly nonlinear, divergent, or unstable behavior when driven outside their nominal operating regimes. Examples include stall and post-stall phenomena in fixed-wing aircraft, loss-of-control incidents involving departure from controlled flight and aerodynamic degradation effects such as icing which change lift, drag, and stability derivatives significantly. Under these conditions, system dynamics can diverge rapidly and small errors in state or inputs may lead to large deviations in the subsequent trajectories [1, 2].

System identification in these divergent regions is difficult because: (a) experiments in those regimes are often unsafe or infeasible; (b) the system may behave in a manner not seen in nominal conditions, so extrapolation from nominal-regime models is unreliable; (c) uncertainty (aerodynamic, structural, sensor/actuator) tends to be larger; (d) linearization around trim points fails to capture essential nonlinearities or the underlying bifurcation behavior.

Consequently, given the strict regulatory environment of the aerospace/aviation sector [3, 4], the control design of such systems has to rely upon conservative assumptions to ensure stability under modeling uncertainty, which often translates in reduced operational envelopes. To manage safety, industrial practice mandates defining operating limits, and designing envelope protection or limit functions for key parameters (e.g., limiting angle of attack, load factor, airspeed, bank angles, angular rates) [5]. These protections are often implemented manually: An analysis of available experimental or test data is performed to define parameter boundaries, followed by the tuning of the required protection laws, typically comprising saturations, limiters, filters, command governors, or dedicated protection logics [6]. This process is laborious, expensive, and time-consuming, especially for new aircraft with complex, unconventional configurations.

With a view to improving the modeling of non-linearities that are usually the cause of divergent behavior outside the nominal operating domain, neural network (NN) regressors –known for their ability to act as global function approximators– have been introduced in the system identification/control design process: Stachiw et al. [7] present a physics-based NN model capable of approximating forces and moments over the entire flight envelope, for a wide range of angle of attack and sideslip, using flight

test data and second-order Taylor expansions augmented by NNs. In a related study by Yue et al. [8], NNs in recurrent architectures are applied to detect flight environment changes and to adjust envelope protection control laws in UAVs. Despite the benefits of the above approach in terms of modeling error, it is accepted that one has to deal with some degree of uncertainty when trying to expand the operating envelope of such a system.

When attempting to impose constraints on the state trajectories of an uncertain system, different control solutions are available to the control designer: From the field of optimal control, covariance steering is a technique whereby one designs controls that "steer" both the mean and the covariance (uncertainty) of a system's state [9, 10]. Ridderhof and Tsiotras [11] demonstrate this approach in a powered descent and landing problem, where the mean and variance of the descent paths must be constrained within predefined targets. Autenrieb [12] formulates flight-envelope protection (FEP) as an explicit state-constrained control problem and solves it online with a safety filter built from Control Barrier Functions (CBFs). The safety filter minimally modifies a nominal command by solving a quadratic program that enforces the High-Order CBF inequalities in addition to actuator/input limits; this guarantees forward-invariance of the flight-safe set, subject to the modeling/tuning assumptions, while keeping the nominal controller in charge whenever it is safe to do so. The method is demonstrated on a nonlinear missile longitudinal flight control model: angle-of-attack, load factors, actuator deflection and rate limits are enforced successfully. Yi et al. [13] use offline Monte Carlo simulation methods to construct a probabilistic safe envelope, store it as a database which is then implemented in an online, graded predictive protection function via generalized multi-loop pseudo-control hedging (PCH) integrated into a nonlinear dynamic inversion (NDI) flight controller.

Albeit well-established from a theoretical standpoint, the above techniques require a more-or-less specific model structure to function, typically comprising linear, linearizable or time-invariant dynamics. To overcome this limitation, recent work has begun to experiment with reinforcement learning (RL)-based methods for FEP: For example, Catak et al. [14] propose a longitudinal FEP RL algorithm that limits variables such as angle of attack, load factor, and pitch rate; their design reduces manual tuning by using learned approximations to protecting functions under nonlinear flight control. Grillo et al. apply RL to an optimal stall recovery task, to manage control through highly nonlinear behavior near stall departure [15]. The RL approach is advantageous in that it provides more flexibility in terms of the admissible model structure and can function quasi-autonomously, partly relieving the control designer from the need to define dedicated logics and controller structures.

The absence of theoretical safety guarantees is currently a blocking point for industrial-level applications of this methodology in the aerospace field, however, there exists a clear trend towards its progressive introduction: Roadmaps for the introduction of Artificial Intelligence (AI)- based guidance and control systems are currently being established [16], benefiting from recent work on the validation of autonomous systems such as self-driving cars [17]. Nonetheless, work on RL-derived control protections is still rather rare in the open literature.

In this paper, building up on previous work by the authors [18], we propose a RL-inspired technique for FEP aimed explicitly at reducing the required designer workload: Radial Basis Function Networks (RBFNs) are employed for data-only identification of dynamics in the nominal operating envelope. A novel method for exploiting the system model's inherent parametric/structural uncertainty –a byproduct of the system identification process– to generate state/output gradients that contain the dynamics within the known, safe envelope is introduced. This is integrated to a temporal backpropagation (TB) calculation [19], which converts a finite-horizon discrete-time optimal control problem to one of training of a deep, recurrent NN architecture.

Section 2 details the theoretical background of the proposed technique. Section 3 presents example applications, starting with a simple numerical test case, followed by a full-scale demonstration on a 6-DoF aircraft simulator. Conclusions and related findings are discussed in section 4.

2 Methodology

2.1 System Dynamic Emulator

Let X_i denote the state of a Markovian system S at discrete time step i, and U_i the corresponding system input. Consider the collection of observed transitions

$$C = \{(X_i, U_i, X_{i+1}) \mid i \in I\},$$

with I being the index set of sampled time steps. This dataset serves to train a surrogate model M, which emulates the effect of inputs on the system's dynamics, producing the approximate state evolution

$$\tilde{X}_{i+1} = M(X_i, U_i).$$

A variety of surrogate architectures may be employed for M, hereafter referred to as "emulator", including physics-inspired or purely data-driven approximators. Analytic formulations such as neural networks [19] are particularly well-suited to the proposed framework, since they facilitate the computation of derivatives and reduce computational cost. For the demonstrations in this work, Radial Basis Function networks (RBFNs) [20] are adopted, though the methodology is not limited to this choice.

Given an input sequence $U = \{U_i \mid i \in [0, N]\}$ and an initial state $X_0 = \tilde{X}_0$, the system emulator estimates the system state trajectory recursively as per following:

$$\tilde{X}_{i+1} = M(\tilde{X}_i, U_i), \quad i = 0, 1, \ldots, N. \tag{1}$$

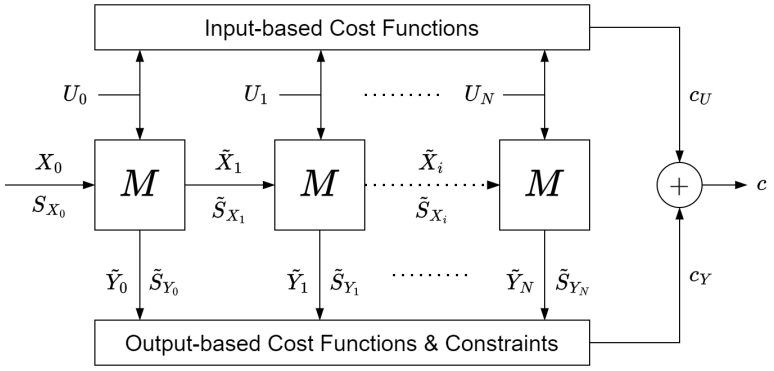


Fig. 1 Forward propagation of the surrogate model M for trajectory approximation [18].

From an optimal control perspective, the model is assumed to generate not only state predictions $\tilde{X}_i$, but also corresponding outputs $\tilde{Y}_i$. These, combined with the applied inputs U_i, enable the formulation of cost functions and constraints across the trajectory:

$$(\tilde{X}_{i+1}, \tilde{Y}_i) = M(\tilde{X}_i, U_i), \quad i = 0, 1, \ldots, N. \tag{2}$$

2.2 Modeling Uncertainty

The statistical characterization of M, as derived during training, may further be used to produce estimates of parameter uncertainty $\tilde{S}_{Y|\theta}$, originating from the imperfect estimation of the model coefficients θ. For nonlinear models, the parameter uncertainty of $\tilde{Y}_i$ is obtained as

$$\tilde{S}_{Y_i|\theta} = \tilde{J}_{Y_i|\theta} S_\theta \tilde{J}_{Y_i|\theta}{}^T \tag{3}$$

where $\tilde{J}_{Y_i|\theta}$ is the Jacobian of $\tilde{Y}_i$ with respect to parameters θ, and S_θ denotes their covariance. The associated calculations being rather straightforward, both quantities are assumed to be known or readily

estimated. The interested reader may refer to reference [21] for additional details. In this study, the parametric uncertainty of RBF models was obtained through a bootstrapping scheme [22].

Non-parametric regressors such as neural networks are known for their poor performance in tasks involving any form of extrapolation: Once outside its training domain, the output of such a model rapidly diverges away from the true output. Nonetheless, this behavior may be predicted if one tracks the relative variation of $\tilde{S}_{Y_i|\theta}$ which, by default, will showcase a similar divergence. In this case, the mean of $\tilde{S}_{Y_i|\theta}$ over the training dataset, or, alternatively, its value at the center of the training domain may be used to provide a reference signal, in order to identify divergence due to extrapolation. In other words, the relative size of the output confidence intervals compared to the interpolation case may be employed to identify extrapolation and, hence, define the limits of a model's validity domain (Fig. 2).

The above property is of particular interest for the identification and control of systems experiencing divergent dynamics when moved away from their nominal operating regime; such systems are widespread in the aerospace field, ranging from aircraft with spin-prone departure characteristics to aero-structural limit-cycle oscillations, rotorcraft retreating blade stall and jet engine compressor stall/surge dynamics. In these cases, the behavior outside the normal operating envelope does not allow for proper identification of the system dynamics under these conditions; thus, most usable models are limited to nominal operation and abnormal conditions need to be extrapolated.

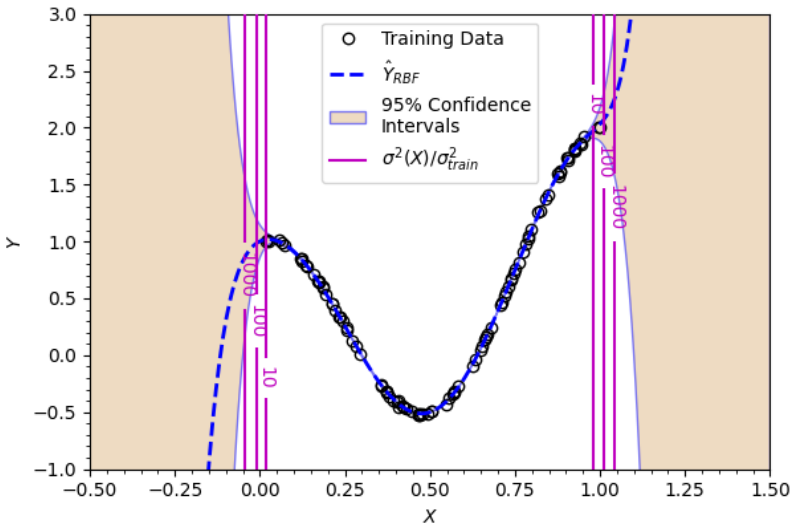


Fig. 2 RBFN output on a 1-D regression task, with training data $X \in (0, 1)$. Divergence of predictions outside the training domain is captured by confidence intervals. Magenta lines show the ratio of local output variance to training error variance.

As per standard industrial practice, a control designer needs to manually define system operating limits and, based on the latter, conceive and tune suitable envelope protection functions that will contain system dynamics within the target regime.

In the following sections, we demonstrate how the local parametric uncertainty estimates $\tilde{S}_{Y_i|\theta}$ can be leveraged within a reinforcement-learning-inspired temporal backpropagation calculation to automatically generate Multiple-Input-Multiple-Output (MIMO) envelope protection control signals without explicitly defining the system's operating limits.

2.3 Parametric Covariance Constraints

Previous work by the authors has introduced probabilistic/chance constraints in the temporal backpropagation calculation to generate control sequences respecting predefined output limits while accounting for various sources of modeling uncertainty [18]. In this context, action gradients were calculated based on the distance of time-propagated model output covariance ellipsoids from hyperplanes representing a convex constraint set.

In this study, we assume that the bounds of the nominal operating domain are unknown to the designer, yet included in the covariance S_θ of the parameters θ of a system emulator M, trained on a finite

set C of sampled state/output trajectories. Let $\tilde{S}_{Y_i|\theta}$ represent the output covariance at instance i due to θ and $S^*_{Y|\theta}$ a bounding ellipsoid defining the upper limit of the admissible covariance output. In the absence of system-specific covariance information, the latter may be taken to be a simple scaled-up version of the emulator's output covariance over its training dataset. The difference between $\tilde{S}_{Y_i|\theta}$ and $S^*_{Y|\theta}$ is used to establish an expression defining the limits of the latter, in the form of a linear matrix inequality:

$$C^* = \left\{ X \in \mathbb{R}^D \;\middle|\; \Delta S(X) := S^*_{Y|\theta} - \tilde{S}_{Y_i|\theta}(X) \succeq 0 \right\} \tag{4}$$

Eq. 4 defines domain C^* as a subset of $\mathbb{R}^D$ containing all state vectors X such that the matrix quantity $\Delta S(X)$ is positive semidefinite, i.e. the local parametric covariance ellipsoid $\tilde{S}_{Y_i|\theta}(X)$ is contained within the respective bounding covariance ellipsoid $S^*_{Y|\theta}$. The level of conservatism applied to the scaling of the latter controls the tightness of the C^* bound around C: The scaling factor is the only tuning parameter required for the proposed envelope protection to function. In addition, the performance of the control law was found to be rather insensitive to the amount of scaling applied, thanks to the rapid divergence of neural network regressors' parametric covariance outside C (see Fig. 2): This suggests that, in practice, the C^* bound in X is only slightly shifted for different values of the scaled covariance parameter, provided the latter be such that the C^* limit lies close to the "edge" of the training domain. Our numerical experiments use a scaling factor of 10 applied to training set covariance which represents a good compromise between non-intrusiveness in nominal operation and timely activation of the protection logic.

Eq. 4 sets the basis for writing a constraint function G that forces X to remain within C^*. Among various possibilities and without loss of generality, we hereby adopt a quadratic penalty formulation for G:

$$G(X) = \frac{1}{2} w \sum_i \lambda_i^2 \quad , \quad i : \lambda_i < 0 \tag{5}$$

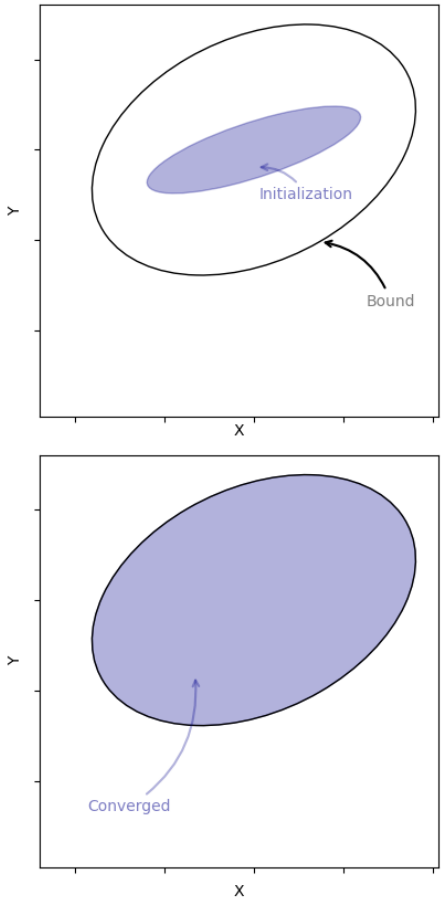


Fig. 3 Demonstration of ellipsoid containment constraint (Eq. 5).

where λ_i are the negative eigenvalues of ΔS and w is a weighting factor. The gradient of G with respect to the entries of $\tilde{S}_{Y_i|\theta}(X)$ is:

$$\frac{\partial G}{\partial \tilde{S}_{Y_i|\theta}} = -w \; \text{vec}\left[\sum_i \lambda_i \frac{\partial \Delta S}{\partial \lambda_i} \right] =$$

$$-w \; \text{vec}\left[\sum_i \lambda_i (q_i q_i^T) \right] \tag{6}$$

with q_i being the unit eigenvector corresponding to eigenvalue λ_i. The vec() operator is used to flatten the square array to a column vector. A toy numerical example demonstrating the function of constraint G is shown in Fig 3: Given an outer elliptical 2-dimensional boundary representing $S^*_{Y|\theta}$ and a random

initialization of $\tilde{S}_{Y_i|\theta} \subset S^*_{Y|\theta}$, the entry-wise ℓ_1 norm $|\tilde{S}_{Y_i|\theta}\|_1$ is maximized subject to constraint G, leading to the –known– optimal solution $\tilde{S}_{Y_i|\theta} \equiv S^*_{Y|\theta}$.

Eq. 3 defines the relationship between output parameter covariance $\tilde{S}_{Y_i|\theta}$ and the local Jacobian $\tilde{J}_{Y_i|\theta}$: For a neural network regressor, the latter is an analytic function of input state X and can be calculated using error backpropagation. The gradient of $\tilde{S}_{Y_i|\theta}$ with respect to a component X_k of X is given by:

$$\frac{\partial \tilde{S}_{Y_i|\theta}}{\partial X_k} = \text{vec}\left[\frac{\partial \tilde{J}_{Y_i|\theta}}{\partial X_k} S_\theta \tilde{J}_{Y_i|\theta}{}^T + \tilde{J}_{Y_i|\theta} S_\theta \left(\frac{\partial \tilde{J}_{Y_i|\theta}}{\partial X_k}\right)^T\right] \tag{7}$$

Applying the chain rule to Eqs. 6- 7, the derivative $\frac{\partial G}{\partial X_k}$ may be obtained:

$$\frac{\partial G}{\partial X_k} = \left(\frac{\partial G}{\partial \tilde{S}_{Y_i|\theta}}\right)^T \frac{\partial \tilde{S}_{Y_i|\theta}}{\partial X_k} \tag{8}$$

We assemble the complete gradient $\nabla_X G$ by repeating the calculation for all k:

$$\nabla_X G = \left[\frac{\partial G}{\partial X_1}, \cdots, \frac{\partial G}{\partial X_k}, \cdots \frac{\partial G}{\partial X_N}\right] \quad , \quad k = 1, \cdots, D \tag{9}$$

The same process can be applied to obtain the gradient of G with respect to system input U. Constraint function G is combined with an objective function $F(X_i, U_i)$ (e.g. a minimum control energy criterion) to form the complete cost reward function c for the emulator chain of Fig. 1.

$$c_i = F(X_i, U_i) + G(X_i, U_i) = F_i + G_i \tag{10}$$

2.4 Temporal Backpropagation

The backpropagation-through-time framework, introduced by Nguyen and Widrow [19], is a model-based alternative to black-box reinforcement learning methods such as Q-learning or actor–critic schemes [23]. Unlike these, it leverages a system emulator to compute both rewards and their gradients with respect to candidate policies or action sequences. Training thus proceeds in two stages: first, the emulator identifies the system dynamics; second, the extracted model is exploited to optimize control policies. This decoupling of system identification from control optimization affords the designer finer control over the training process.

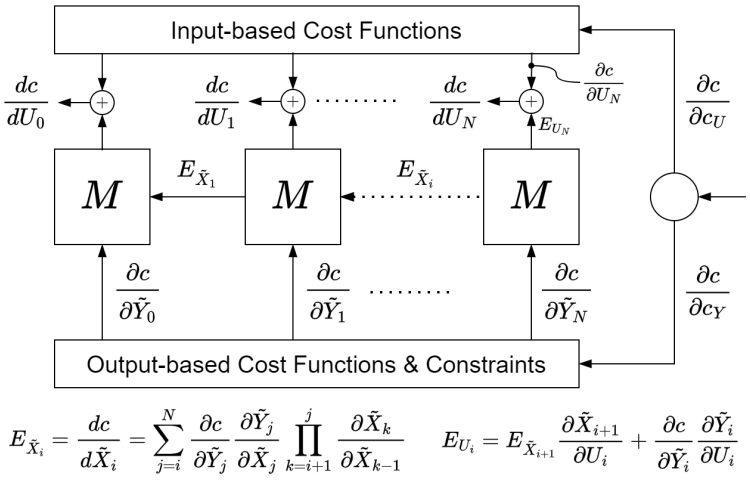


Fig. 4 Backward pass: computation of reward/cost gradients with respect to inputs U_i. [18]

Based on the above, the proposed methodology broadly classifies as "data-driven model-based" learning, sharing elements with both optimal control and reinforcement learning algorithms. For instance,

TB can be used to simultaneously train the controller and the system emulator itself: In that case, the method reduces to a "standard" reinforcement learning algorithm. If, on the other hand, the system emulator is replaced by a linear/linearized model, the algorithm is analogous to an LQR or model predictive controller (MPC). As shown in [19], the NN-based emulator leverages the global approximation capabilities of NNs to convert complex system models to analytic form and simplify action gradient calculation.

As in other reinforcement learning methods, the computation of optimal policies follows an iterative two-step cycle until convergence: a forward pass, which updates reward values, and a backward pass, which propagates gradients to adjust the actions. Figures 1 and 4 illustrate these computations.

As shown in Figure 4, the gradient of the cost/reward function c with respect to each action U_i has two contributions: the direct derivative from explicit input terms, and an indirect derivative reflecting the influence of U_i on subsequent states and outputs. Formally,

$$\frac{dc}{dU_i} = \frac{\partial c}{\partial U_i} + E_{U_i}, \tag{11}$$

where E_{U_i} denotes the error term arising from trajectory-dependent effects:

$$E_{U_i} = \frac{\partial c}{\partial \tilde{Y}_i}\frac{\partial \tilde{Y}_i}{\partial U_i} + E_{\tilde{X}_{i+1}}\frac{\partial \tilde{X}_{i+1}}{\partial U_i}. \tag{12}$$

The first term captures the direct sensitivity of output $\tilde{Y}_i$ to U_i; the second accounts for the propagated effect through future states. The latter depends on

$$E_{\tilde{X}_i} = \sum_{j=i}^{N} \frac{\partial c}{\partial \tilde{Y}_j}\frac{\partial \tilde{Y}_j}{\partial \tilde{X}_j}\prod_{k=i+1}^{j} w_d \frac{\partial \tilde{X}_k}{\partial \tilde{X}_{k-1}}. \tag{13}$$

which represents the chain of dependencies linking state $\tilde{X}_i$ to all future states $\tilde{X}_j$. A discount factor w_d is introduced to limit the effective planning horizon and prevent numerical issues such as exploding gradients.

Overall, as stated above, TB shares similarities with Nonlinear Model Predictive Control (NMPC) [24], in that present decisions are optimized with foresight of their long-term effects. This is achieved through recurrent stacking of surrogate model layers, forming deep recurrent-like networks with a depth proportional to the planning horizon. Such architectures facilitate efficient evaluation of derivatives and error backpropagation across the entire trajectory.

3 Results

3.1 A simple numerical example

Consider the following non-linear state-space system with state X and input U [18]:

$$X = (x, v)^T \equiv Y, \quad U = u \in [-2, 2]$$

$$dX = \begin{cases} v \\ -0.14 - 1.14x - 0.29x^2 - 1.43v - 0.43v^2 + 0.6u \end{cases}$$

$$X_{i+1} = X_i + 0.01dX \tag{14}$$

Define the following learning problem: Starting from a random initial condition $X_0 \in C^*$, calculate an optimal sequence of $N = 300$ actions U_i that minimizes the value of the cost function c:

$$c = \sum_{i=1}^{N} \left(w_u U_i^2 + G_i \right) \tag{15}$$

Using the methodology described in section 2, an RBF network with 20 hidden units was employed, initially to identify a model $X_{i+1} = M(X_i, U_i)$ emulating the system dynamics and, following this, to extract an optimal sequence of actions minimizing the value of Eq. 15. The training of the RBF emulator was performed on a database of simulated time series data of the true model of Equation 14 comprising 1,000 sample instances i. The following filtering condition was applied to the training samples so as to artificially create "empty" areas in the sampling domain, representing regions of unknown dynamics, as shown in Fig. 5:

$$0.35x + 0.94v + 0.2 > 0 \tag{16}$$

The calculation of the optimal sequence U was performed by means of a gradient descent algorithm based on the analytic derivatives $\frac{dc}{dU_i}$ extracted from the TB scheme (Eq. 11) with a discount factor $w_d = 0.999$.

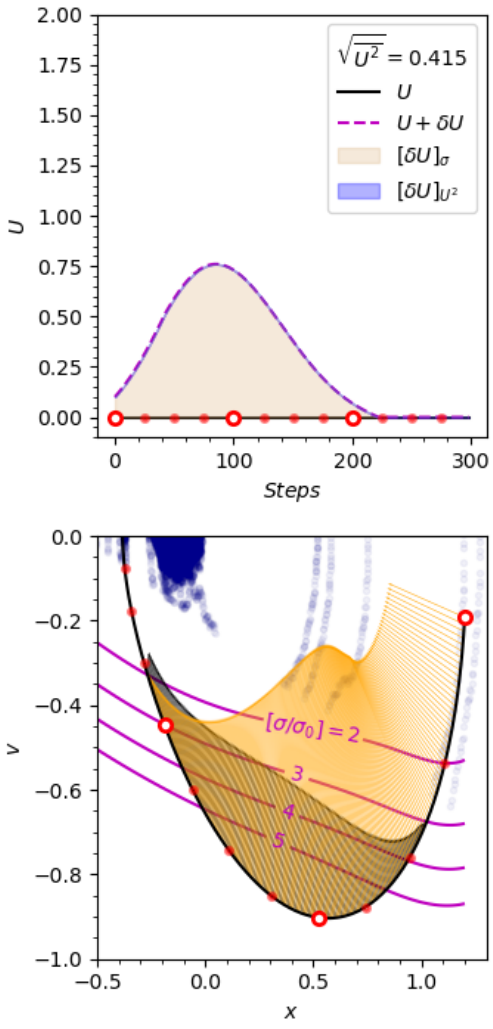
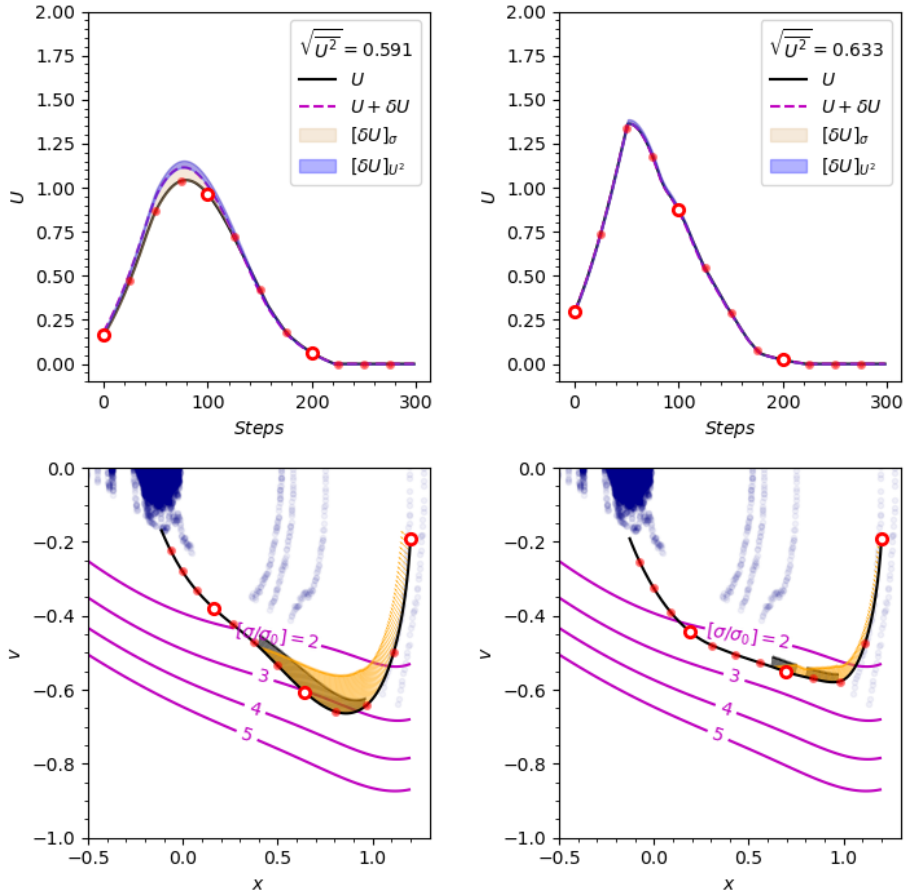


Fig. 5 Section 3.1–Learning iteration 1. Control Sequence [top] VS State Trajectory [bottom], over training data [blue] and parametric uncertainty contours [magenta]. Black- and orange-colored arrows represent local and backpropagated gradients of G respectively.

Figs. 5-6 demonstrate the convergence process of the learning algorithm over 40 iterations. Initially (Fig. 5), the zero-input system state trajectory (black curve) moves away from the limits of the "known" domain C^*, represented by blue dots accounting for training datapoints and contours of relative local covariance σ/σ_0 of the emulator M (in Figs. 5-6, the effect of U on σ/σ_0 is neglected, shown values correspond to $U = 1$) . This results in locally non-zero values of constraint function G_i which translate into local state gradients (black arrows); the latter are subsequently backpropagated in time by the learning algorithm (orange arrows) to affect past states and introduce anticipatory action in the control sequence. Consequently, whereas the first non-zero local gradients of G_i appear after timestep 50, the respective backpropagated gradients transfer the information to all previous steps.

A correction δU_i is calculated by the gradient descent update rule for each time step, comprising two subcomponents $[\delta U_i]_\sigma$, $[\delta U_i]_{U^2}$ associated to the covariance constraint and minimum control energy objective respectively. Initial control gradients (Fig. 5) are driven by δU_σ, later the δU_{U^2} term gradually limits the amplitude of the control update δU (Fig. 6a) until the two correction signals mutually cancel out to obtain convergence (Fig. 6b).



(a) Iteration 3 **(b) Iteration 40**

Fig. 6 Section 3.1–Convergence

As a result of the calculated control input, the final state trajectory moves parallel to the contours of σ/σ_0, staying closer to the training data than the initial trajectory, at a distance defined by the relative weighting of the minimum control energy and covariance objectives.

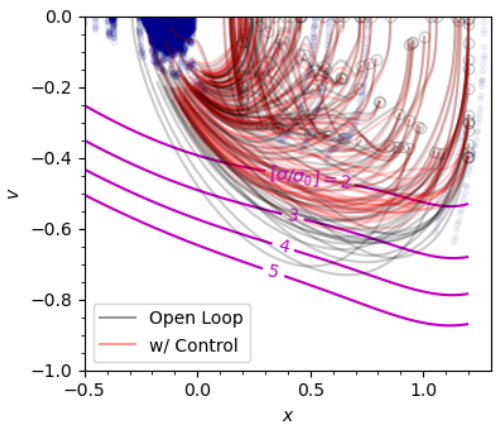


Fig. 7 Section 3.1–Traces of 100 randomly-initialized control-free state trajectories (black) VS "closed-loop" results (red). Circle markers denote initial conditions.

Overall, as was observed in [18], despite the large dimensionality ($N = 300$) of the underlying optimal control problem, the algorithm is numerically well-behaved and convergence is obtained within a few iterations.

The global effect of the proposed learning algorithm on system dynamics is demonstrated in Fig. 7, comparing traces of 100 randomly-initialized control-free state trajectories with the respective closed-loop results: It can be observed that the control action globally bounds state trajectories within the $\sigma/\sigma_0 = 3$ limit, without this being explicitly defined by the user, say, in the form of a planar constraint similar to Eq. 16.

Finally, we note the role of the minimum control energy objective in Eq. 15 which serves, on the one hand to reduce the amplitude of control inputs, on the other to avoid unnecessary interference of the controller with the natural system dynamics when no risk of crossing

the C^* boundary is identified. The latter feature is essential for a non-intrusive envelope protection function, as shown in the -equivalent- quadratic formulation adopted for CBFs in reference [12].

3.2 Aircraft Envelope Protection

In addition to the numerical example of section 3.1, with a view to testing the performance of the proposed methodology in a challenging use case from the aerospace field, a demonstration of a notional online stall departure protection function was performed in a series of simulated experiments with a representative test vehicle.

The main idea behind the above demonstration being a proof of the feasibility of the proposed control solution, the test strategy was adapted to represent an extreme test environment for the latter: A limited dataset was made available to the system emulator for the identification of a system with divergent behavior outside its nominal operating envelope, in addition to rapid overall dynamics.

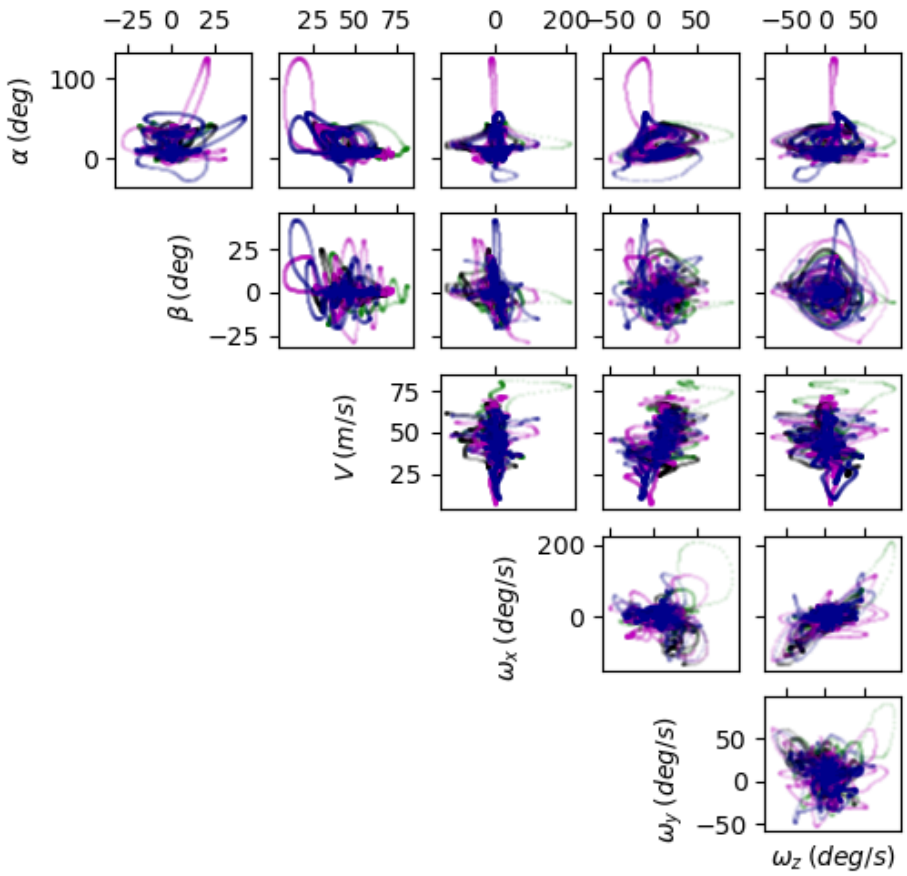


Fig. 8 Training Data for the RBF aircraft emulator. Different colors correspond to individual recordings.

The test vehicle selected for the experiments is based on an in-house 6-DoF aerobatic aircraft simulator [25], employing strip-theory aerodynamics [26] augmented with the pressure gradient correction factor proposed by McCormick [27] for more accurate prediction of aerodynamic forces and moments in the post-stall regime. The simulator can realistically model a practically unlimited angle of attack (AoA) and sideslip (β); during validation experiments it was noted that transients close to the stall AoA lead to unpredictable departure dynamics which are very sensitive to the stall entry conditions. As such, depending on pilot inputs, the modeled stall behavior frequently leads to a spin condition and is hard to identify with a surrogate model.

The following setup was used for the experiments performed: First, a series of four man-in-the-loop simulations were recorded including purposely aggressive piloting so as to induce departures: In a total

duration of 362 sec of simulated time, sampled at 50Hz, the maximum recorded values of AoA and angular rates exceeded 100 deg and 200 deg/s respectively. The resulting data, summarized in Fig. 8, were employed to train an RBFN emulator of the aircraft dynamics. A network with 400 hidden units was selected, with the following input/output configuration:

$$X \equiv Y = [\vec{V}_b, \vec{\omega}_b, \vec{x}_b, \vec{z}_b] \ , \ U = [dl, dm, dn] \tag{17}$$

where $\vec{V}_b, \vec{\omega}_b$ are the body-axis velocity and angular rates, $\vec{x}_b, \vec{z}_b$ are the vectors of the aircraft x, z body axes in earth coordinates and dl, dm, dn account for aileron, elevator and rudder commands respectively. The aircraft dynamic emulator thus comprises a "black-box" model with 15 input and 12 output dimensions.

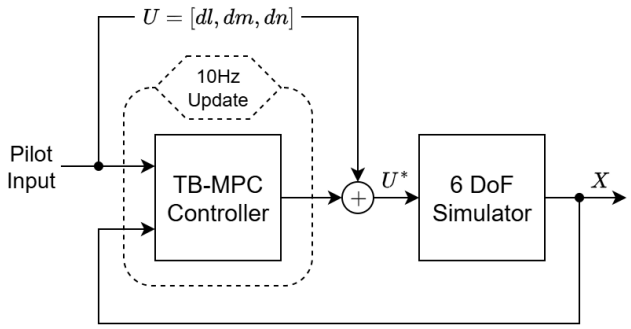


Fig. 9 Controller schematic for the aircraft envelope protection experiments.

Following model identification, additional man-in-the-loop sequences were recorded and used for testing a pseudo-real-time implementation of the proposed algorithm. Motivated by the rapid convergence of the algorithm in the experiments of section 3.1, an online implementation was targeted for the FEP controller:

The selected control configuration, shown in Fig. 9, resembles that of a non-linear MPC algorithm, with the neuro-emulator of Section 2 being employed to predict the aircraft state over a 4-second time horizon, and the temporal backpropagation scheme generating an updated control sequence. The calculated controller output is added to pilot aileron, elevator and rudder commands in the form of an online envelope protection function: In Fig. 9, an asterisk (*) is used to denote the joint command signal sent to the aircraft simulator. Prediction and control updates were based on a zero-order-hold (ZOH) measurement of the pilot commands and the aircraft state, both updated at a frequency of 10 Hz. Only one update of the temporal backpropagation scheme of Section 2 was allowed between successive state/pilot input measurements (i.e. only one gradient descent step per 0.1 sec), to assess the feasibility of the proposed technique from a computational standpoint in a real-time-like setup. The above-mentioned update rate was achievable on a standard-spec personal computer; given that the proposed algorithm does not involve matrix inversions or other complex operations, its computational cost scales linearly with longer planning horizons and aircraft model complexity. Nonetheless, integration aspects associated to a real-world flight control application have not been thoroughly analyzed, but were instead left for future work on the project.

Figures 10 - 13 depict an extract of four test timeseries obtained in the frame of these experiments, superimposing the initial recorded data (black) with timeseries data obtained with the controller in the loop: The respective control output corresponds to the difference between dl, dm, dn signals in open- and closed-loop mode.

As far as the specific features of the simulated test sequences are concerned, Sequence 1 (Fig. 10) comprises gentle control inputs at low-moderate AoA and was used to evaluate the non-intrusiveness of the tested envelope protection control law within the nominal envelope; the very small differences between open- and closed-loop command signals confirm this characteristic. Of course, as is the case with most envelope protection laws, even minor protection-related changes in dynamic derivatives cumulate over time, particularly in states resulting from kinematic integration (in Fig 10, ϕ, θ, ψ and V, the latter linked to the vehicle's altitude and mechanical energy). The effect becomes more pronounced in the presence of dynamic non-linearities and high angular rates, but, in reality, is not perceived by the pilot

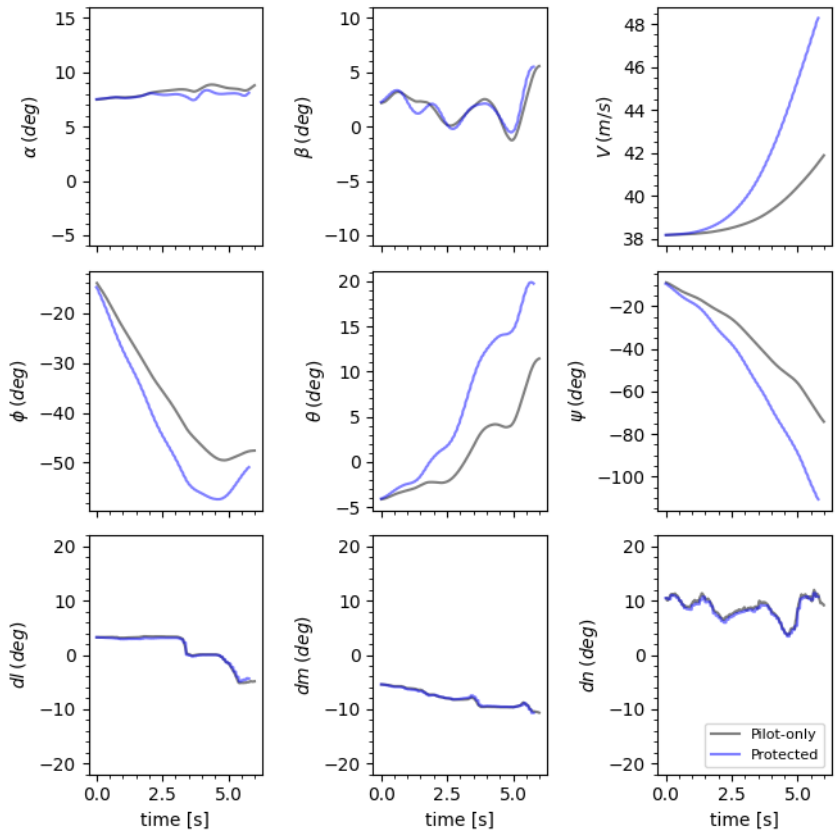


Fig. 10 Aircraft Test Sequence 1: Pilot-only (black) VS envelope-protected (blue) results.

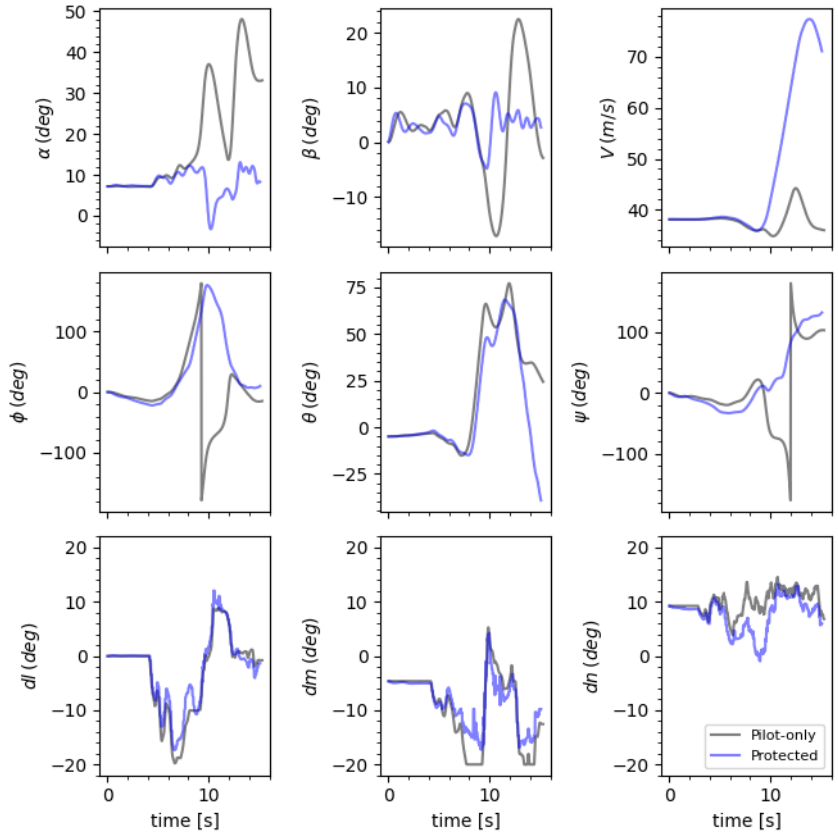


Fig. 11 Aircraft Test Sequence 2: Pilot-only (black) VS envelope-protected (blue) results.

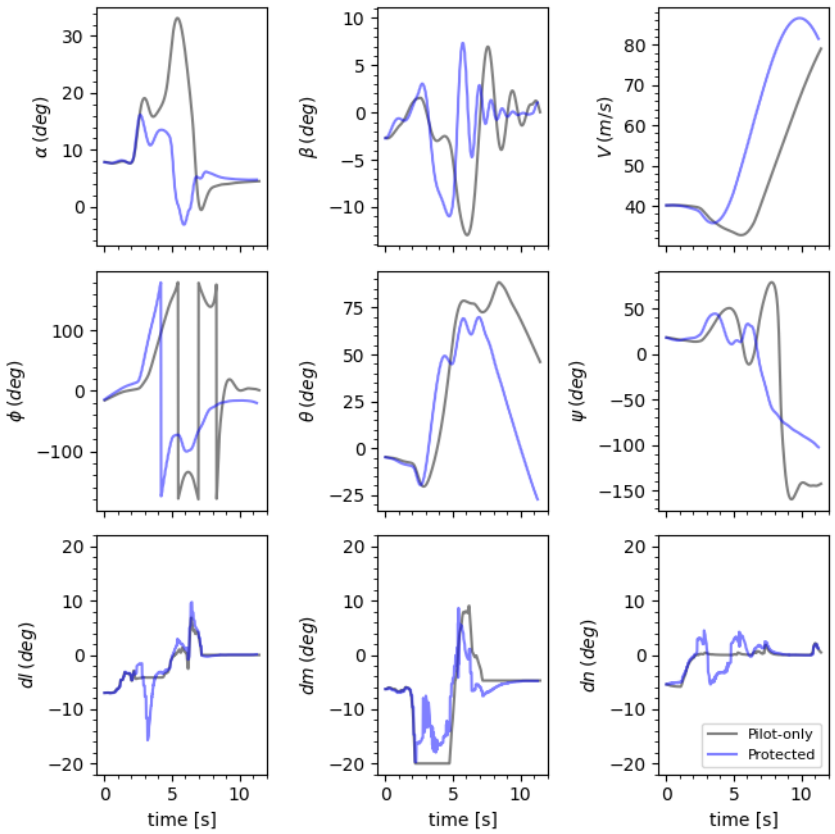


Fig. 12 Aircraft Test Sequence 3: Pilot-only (black) VS envelope-protected (blue) results.

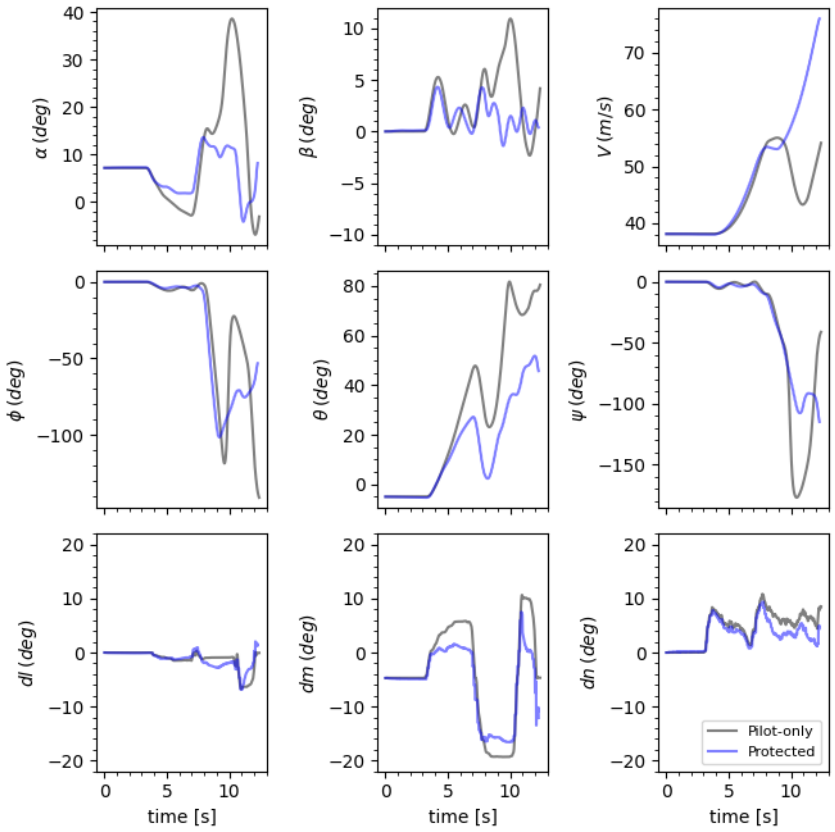


Fig. 13 Aircraft Test Sequence 4: Pilot-only (black) VS envelope-protected (blue) results.

whose focus is on the short-term variation of the α, β states: The latter are practically identical between pilot-only and protection-ON results.

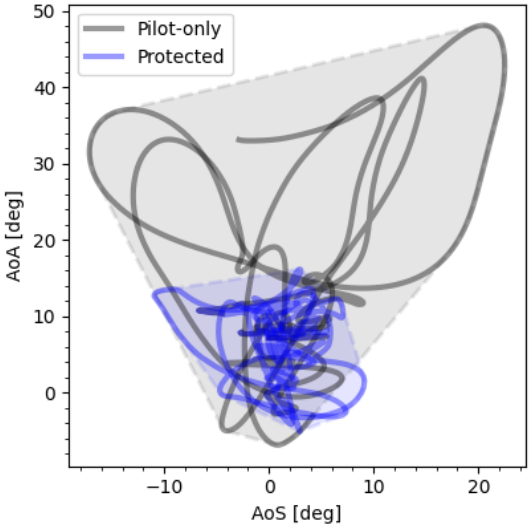


Fig. 14 AoA-Sideslip trajectories of test sequences 1-4, in pilot-only (black) and protection-ON mode (blue). Shaded areas represent the convex hulls of the respective distributions.

Sequences 2-4 (Figs. 11-13) include different combinations of simultaneous 3-axis excitation at high-AoA, aimed at inducing random, asymmetric stall departures as confirmed by the somewhat unpredictable aircraft behavior observed in the respective time response curves. In Sequence 2, stall departure is induced by elevator (dm) input, combined with aggressive aileron (dl) commands and simultaneous rudder (dn) pulses. Test Sequence 3 combines elevator and negative pre-stall rudder commands to cause departure under negative sideslip β. Finally, Sequence 4 employs a similar strategy, but with positive post-stall rudder inputs. In all 3 experiments, through coordinated 3-axis control actions, the TB-MPC controller manages to limit the aircraft AoA and β within the stall-free flight domain ($|\alpha|, |\beta| < 15$), in spite off the the unaugmented aircraft's AoA reaching values as high as 50 deg in the original recordings. Controller action is more prominent in time instances preceding stall departures of the unprotected aircraft and remains or rapidly converges to zero if no risk is identified (e.g. Figs. 10-13 when $\alpha \in [0, 10]$ and $\beta \approx 0$), as desired for a non-intrusive FEP function. Fig. 14 presents a superposition of the AoA-β trajectories of test sequences 1-4 in pilot-only and protected mode. The protection has a clear effect on the resulting distributions, which are contained well within the limits of the nominal operating envelope. The above results demonstrate a satisfactory level of robustness to large-amplitude pilot inputs at the departure limit and to the inherent modeling error (i.e. "reality gap") of the RBFN emulator used for calculating the control output.

To conclude the above discussion, it needs to be clarified that the selected online control configuration does not constitute the only possible use of the method itself, which can be equally used within a more "standard" offline reinforcement learning environment to train neural controllers on the same task. In such a case, having access to the full state trajectories, the learning algorithm is expected to showcase better performance overall, thus the interest of testing the real-time version of the controller. Moreover, since the scope of this work is limited to a proof of feasibility of the control architecture itself, testing only comprised a finite number of recorded sequences; a more formal validation of global stability was left for future work, deemed to be case-dependent and out of the scope of the intended demonstration.

4 Conclusions

In this article, a new reinforcement-learning-inspired methodology for active envelope protection was presented: The proposed technique exploits the uncertainty in the estimation of the internal parameters of a neuro-emulator of a system's dynamics to identify instances where the system exits its domain of nominal operation. A method for deriving control gradients out of parametric uncertainty output was equally detailed; this was integrated to a temporal backpropagation calculation, used to generate control sequences that protect the system against excursions of the normal operating boundaries. Numerical experiments demonstrated the effectiveness of the proposed approach in bounding dynamics within a safe operating regime, without need for an explicit definition of the latter by the designer.

In line with the authors' original intention, very little human intervention was required for setting up the MIMO FEP logic, limited to the tuning of the gradient descent update gain and the scale factor applied to the reference covariance signal, without any vehicle-specific information passed to the TB solver. Thus, the method is particularly suitable for ensuring safe operation while testing in a rapid-prototyping-type of operation. In the same context, the protection law is tightly linked to the system emulator itself and automatically adapts to new test data being injected to the latter, as would be the case during envelope-expansion testing.

Finally, the vehicle-agnostic nature of the method allows for its direct application to vehicles with exotic, unexplored configurations, a fact that would be otherwise impossible with more traditional techniques where the designer often sets the envelope boundaries based on experience-derived criteria. Our results, albeit limited to a finite number of recorded flight sequences, demonstrate satisfactory performance in a demanding, highly nonlinear aircraft 3-axis control application.

References

[1] Frank R Brown William Gato, John T Tran. Stall departure identification, recognition, and recovery, 2018. DOT/FAA/TC-17/56.

[2] Jeffrey Keller, Robert McKillip, and Sungwan Kim. Aircraft flight envelope determination using upset detection and physical modeling methods. In *AIAA Guidance, Navigation, and Control Conference*, page 6259, 2009.

[3] EASA. *Certification specifications and acceptable means of compliance for large aeroplanes (CS-25), Amendment 27*. EASA, Annex to ED Decision 2021/015/R, November 2021.

[4] Arp4761a – guidelines for conducting the safety assessment process on civil aircraft, systems, and equipment, 2023.

[5] Gary J Balas. Flight control law design: An industry perspective. *European Journal of Control*, 9(2-3):207–226, 2003.

[6] Philippe Goupil. Airbus state of the art and practices on fdi and ftc in flight control system. *Control Engineering Practice*, 19(6):524–539, 2011.

[7] Terrin Stachiw, Alexander Crain, and Joseph Ricciardi. A physics-based neural network for flight dynamics modelling and simulation. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):13, 2022.

[8] Ting Yue, Xianlong Wang, Bo Wang, Shang Tai, Hailiang Liu, Lixin Wang, and Feihong Jiang. Design of ice tolerance flight envelope protection control system for uav based on lstm neural network for detecting icing severity. *Drones*, 9(1):63, 2025. DOI: 10.3390/drones9010063.

[9] Fengjiao Liu, George Rapakoulias, and Panagiotis Tsiotras. Optimal covariance steering for discrete-time linear stochastic systems. *IEEE Transactions on Automatic Control*, 2024.

[10] Joshua Pilipovsky and Panagiotis Tsiotras. Data-driven covariance steering control design. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 2610–2615. IEEE, 2023.

[11] Jack Ridderhof and Panagiotis Tsiotras. Uncertainty quantication and control during mars powered descent and landing using covariance steering. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0611, 2018.

[12] Johannes Autenrieb. Quadratic programming approach to flight envelope protection using control barrier functions. *Journal of Guidance, Control, and Dynamics*, pages 1–12, 2025.

[13] Mingzhou Yin, Q Ping Chu, Ye Zhang, Michael A Niestroy, and Coen C de Visser. Probabilistic flight envelope estimation with application to unstable overactuated aircraft. *Journal of Guidance, Control, and Dynamics*, 42(12):2650–2663, 2019.

[14] Akin Catak, Ege C. Altunkaya, Mustafa Demir, Emre Koyuncu, and Ibrahim Ozkol. Enhanced flight envelope protection: A novel reinforcement learning approach. *arXiv preprint*, 2024. arXiv:2406.05586.

[15] Agustin Grillo, Gabriel Torre, and Roberto Bunge. Optimal stall recovery via deep reinforcement learning for a general aviation aircraft. In *AIAA SCITECH 2024 Forum*, page 2408, 2024.

[16] G Soudain, F Triboulet, and A Leroy. Easa concept paper: Guidance for level 1 & 2 machine learning applications. Technical report, Cologne: European Union Aviation Safety Agency, 2023.

[17] Anthony Corso, Robert Moss, Mark Koren, Ritchie Lee, and Mykel Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *Journal of Artificial Intelligence Research*, 72:377–428, 2021.

[18] Aristeidis Antonakis. Reinforcement-learning-based aircraft handling qualities testing with uncertainty modeling. *CEAS Aeronautical Journal*, pages 1–17, 2025.

[19] Derrick Nguyen and Bernard Widrow. The truck backer-upper: An example of self-learning in neural networks. In *Advanced neural computers*, pages 11–19. Elsevier, 1990.

[20] David Lowe and D Broomhead. Multivariable functional interpolation and adaptive networks. *Complex systems*, 2(3):321–355, 1988.

[21] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[22] Achilleas Zapranis and Efstratios Livanis. Prediction intervals for neural network models. In *Proceedings of the 9th WSEAS International Conference on Computers*, page 76. World Scientific and Engineering Academy and Society (WSEAS) Stevens Point . . . , 2005.

[23] Richard S Sutton. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

[24] Lars Grüne, Jürgen Pannek, Lars Grüne, and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

[25] Aristeidis Antonakis. Développement d'un environnement de simulation homme-dans-la-boucle pour l'évaluation des qualités de vol d'aéronefs à voilure fixe. Rapport interne, ONERA/DTIS, juillet 2025.

[26] Bandu N Pamadi. Semiempirical method for prediction of aerodynamic forces and moments on a steadily spinning light airplane. Technical report, NASA-TM-4009, 1987.

[27] Barnes W McCormick. The prediction of normal force and rolling moment coefficients for a spinning wing. Technical report, NASA-CR-165680, 1981.