Towards Bridging Data-Driven Control Synthesis with Hamilton-Jacobi-Bellman Theory

Claudio Vestini
Mechanical & Aerospace Engineering
Princeton University
Princeton, NJ, United States
claudio.vestini@keble.ox.ac.uk

***Abstract*—Synthesising robust control policies for uncertain dynamical systems is traditionally hindered by the computational intractability of the Hamilton-Jacobi-Bellman equation, particularly in high-dimensional or parametric settings. To address this, we propose a data-driven framework that bridges classical optimal control theory with certificate synthesis. Leveraging the Scenario Approach, we extend the subsurface descent algorithm to jointly learn a robust controller and a corresponding reachability certificate from a finite set of sampled parameters. We demonstrate that the certificate conditions function as a relaxation of the discrete-time Bellman equation, with the certificate approximating the optimal Value Function. Numerical validation confirms that the synthesised data-driven controller closely approximates the optimal policy, offering a scalable alternative for robust control synthesis without requiring explicit solutions to partial differential equations.**

I. Introduction

Formal verification of dynamical systems is essential for ensuring properties such as safety, stability, and reachability. Among current verification methodologies, *certificate synthesis* has emerged as a robust, scalable approach, wherein a function satisfying specific inequalities provides formal guarantees on system behaviour. The *subsurface descent* (SSD) algorithm [1], rooted in Scenario Approach Theory [2], has proven effective for synthesising such certificates from finite datasets of discrete trajectories. By leveraging recent advancements in Compression Learning [3], SSD provides probably approximately correct (PAC) probabilistic guarantees on the generalisation of these certificates to unseen trajectories. While SSD effectively verifies pre-existing controllers, a more fundamental challenge lies in *synthesising* control policies that enforce these properties on uncertain parametric models. In this setting, system dynamics depend on parameters governed by unknown probability distributions. We address this by extending the SSD algorithm to jointly learn a robust control policy and its verifying certificate. By nesting a modified SSD routine within an outer-loop trajectory generation process, we update the controller in response to certificate violations, thereby learning from open-loop, discrete-time data.

This methodology is intrinsically connected to optimal control theory, where policies typically minimise cost functionals via the Hamilton-Jacobi-Bellman (HJB) equation [4]. The solution to this partial differential equation (PDE), known as the Value Function, provides both a necessary and sufficient condition for optimality [4]. However, the "curse of dimensionality" often renders direct HJB solutions intractable when the system dynamics comprise a large number of states. In addition, these standard formulations struggle to handle non-deterministic dynamics, and are not robust to parametric uncertainty. Our data-driven synthesis technique can serve as a scalable approximation to the optimal control framework.

Extending the setup in [1], we consider a discrete-time dynamical system evolving in a bounded state space $X \subset \mathbb{R}^n$. The system dynamics are governed by function $f : X \times \mathcal{U} \times \mathcal{V} \to \mathbb{R}^n$, such that the state evolution is given by:

$$x[k+1] = f(x[k], u[k]; v), \tag{I.1}$$

where $x[k] \in X$ is the state at time step k, $u[k] \in \mathcal{U} \subset \mathbb{R}^m$ is the control input constrained to a hyperrectangular set $\mathcal{U}$, and $v \in \mathcal{V}$ represents an uncertain parameter vector. We assume the initial state $x[0]$ is drawn from an initial set $X_I \subset X$ according to a probability measure $\mathbb{P}_x$, and the parameter v is drawn from uncertainty set $\mathcal{V}$ according to an unknown distribution $\mathbb{P}_v$. Additionally, we define a (deterministic) compact goal set $X_G \subset X$, and we denote its boundary as ∂X_G. The sequence of states $\xi = \{x[k]\}_{k=0}^{T}$ constitutes a trajectory of the system over a finite horizon $T \in \mathbb{N}_{>0}$. With a slight abuse of notation, we define the trajectory set Ξ by denoting in turn the joint sets $\Xi_{v,\mathcal{U}} := \bigcup_{u\in\mathcal{U}} \Xi_{v,u}$ and $\Xi_{u,\mathcal{V}} := \bigcup_{v\in\mathcal{V}} \Xi_{u,v}$, joining the two together to obtain $\Xi = \bigcup_{v\in\mathcal{V}} \Xi_{v,\mathcal{U}} = \bigcup_{u\in\mathcal{U}} \Xi_{u,\mathcal{V}}$. This model allows us to incorporate knowledge about a system whilst allowing for uncertainty in some model parameters. Importantly, we seek to find a controller that is *robust* to different parameter instantiations, as opposed to one that depends on the parameters. In

this probabilistic setting, the classical optimal control objective transforms into minimising the expected cost over the distribution of parameters and initial conditions. We define a running cost $L(t, x, u)$ (the system's *lagrangian*) and a terminal cost $K(x[T])$, seeking a control policy $u : X \to \mathcal{U}$ that minimises the functional:

$$J(x_0, u) = \mathbb{E}_{\sim v}\left[\sum_{k=0}^{T-1} L(t_k, x[k], u(x[k])) + K(x[T])\right] \tag{I.2}$$

The optimal Value Function $V^*(x)$ is defined as the minimum expected cost-to-go from state x. In the deterministic case (given a fixed realisation $\bar{v}$), this function satisfies the discrete-time Bellman equation:

$$V^*(x[k]) = \min_{u_k \in \mathcal{U}} \left\{L(t_k, x[k], u_k) + V^*(x[k+1])\right\}. \tag{I.3}$$

Our primary contribution is to demonstrate that data-driven certificate synthesis techniques can be viewed as a scalable approximation to the optimal control framework. We extend the methodology proposed in [1] to synthesise both a certificate function (parameterised by a neural network) and a controller from a finite dataset of sampled trajectories. By interpreting the certificate conditions not merely as property constraints but as relaxations of the Bellman equation (I.3), we connect data-driven verification and HJB theory. Specifically, we show that minimising the residual of the Bellman equation within the *subsurface descent* (SSD) framework (for an appropriate choice of initial states X_I and goal sets yields a controller that is not only robustly stabilising but also optimal with respect to the underlying cost structure. The proposed methodology presents a potential bridge between Optimal Control Theory and Data-Driven Certificate Synthesis.

Notation. We use $\{\xi^i\}_{i=1}^N$ to denote a sequence indexed by $i \in \{1, 2, \ldots, N\}$. We adopt the notation $x[k]$ to indicate the system state at the discrete time step k, corresponding to the continuous time instance $t_k = kT_d$, where T_d is the sampling period. We denote by $\phi(\xi)$ the condition that a trajectory ξ satisfies the desired property, which evaluates to true if the property holds. The symbol $\neg\phi(\xi)$ represents the logical negation, such that $\neg\phi(\xi)$ is true if and only if the trajectory ξ violates property ϕ.

II. Reachability certificate

To verify that the uncertain dynamical system satisfies a desired specification, we seek a scalar function, termed a *certificate*, over the state space. The existence of such a function, subject to specific inequalities, provides a formal guarantee on the system's behaviour. While the framework supports various properties, including safety and reach-while-avoid [1], we focus here on *reachability* certificates. This choice is motivated by the direct theoretical connection between Reachability certificates (which resemble Lyapunov functions) and the Value Function in Optimal Control Theory. A trajectory $\xi \in \Xi_{u,\mathcal{V}}$ is said to satisfy the reachability property, denoted ϕ_{reach}, if it enters the goal set within the time horizon T:

$$\phi_{\text{reach}}(\xi) := \exists k \in \{0, \ldots, T\} : x[k] \in X_G. \tag{II.1}$$

Verifying this property directly over the infinite set of all possible trajectories is intractable. Instead, we employ a certificate function $V : \mathbb{R}^n \to \mathbb{R}$ and a controller $u : X \to \mathcal{U}$. We define the certificate conditions such that their satisfaction implies ϕ_{reach} holds for the controlled system. Fixing a scalar design parameter $\delta > -\inf_{y \in X_I} V(y) \geq 0$, we require the certificate to satisfy structural constraints on the level set:

$$V(x) \leq 0, \quad \forall x \in X_I, \tag{II.2}$$
$$V(x) \geq -\delta, \quad \forall x \in \partial X_G, \tag{II.3}$$
$$V(x) > -\delta, \quad \forall x \in X \setminus X_G, \tag{II.4}$$
$$V(x) > 0, \quad \forall x \in \mathbb{R}^n \setminus X. \tag{II.5}$$

These conditions ensure that the certificate is non-positive on the initial set (II.2), positive outside the domain (to guarantee we do not exit it (II.5)), while the sublevel set V less than $-\delta$ should be contained within the goal set (II.4) (with the inequality relaxed on the goal border (II.3)). To guarantee the system reaches the $-\delta$ sublevel set (and thus X_G), we impose an additional decrease condition along the system dynamics:

$$\begin{aligned} V\big(f(x[k], u(x[k]); v)\big) - V(x[k]) \\ < -\frac{1}{T}\Big(\sup_{y \in X_I} V(y) + \delta\Big), \\ k = 0, \ldots, k_G - 1, \quad \forall v \in \mathcal{V}, \\ k_G := \min\big\{k \in \{0, \ldots, T\} : V(x[k]) \leq -\delta\big\} \end{aligned} \tag{II.6}$$

i.e. the condition must hold for all x along the trajectory until the goal is reached. This condition acts as a relaxation of the standard Lyapunov decrease; rather than requiring asymptotic convergence, it enforces a minimum decay rate sufficient to drive the state into the target set X_G within finite time T. If a pair (V, u) satisfies conditions (II.2)–(II.6) for all parametric realisations $v \in \mathcal{V}$, then the controller u robustly steers the system to the goal set X_G.

III. Data-Driven Synthesis

We adopt a data-driven perspective where the initial state $x[0]$ and the uncertain parameter v are random variables drawn from a product probability space $(X_I \times \mathcal{V}, \mathcal{F}, \mathbb{P})$. We assume that N initial conditions and parameter pairs are drawn independently and identically distributed (i.i.d.) from $\mathbb{P}$. For a fixed controller u, using samples $\{x^i[0], v^i\}_{i=1}^N \sim \mathbb{P}^N$ we can unravel a set of trajectories $\{\xi^i\}_{i=1}^N \in \Xi_{u,\mathcal{V}}$. We parameterise the certificate $V_\theta(x)$ and the controller $u_\eta(x)$ using two separate neural networks with parameters θ and η,

respectively, which we wish to learn[1]. This setting is almost identical to that of [1], with the only difference being the introduction of a new neural network with output vector η used to parametrise the controller. We begin by unravelling trajectories according to some initialised fixed controller u_0, which may be warm-started for improved convergence using information about the system dynamics, or is otherwise initialised as uniform random across states. We iteratively update the controller and trajectories, ultimately learning controller $u_\eta(x)$ (see Section III-D for details). As a result, the trajectories are distributed according to the same probabilistic law distribution as the initial state[2], which we can denote as $\xi \sim \mathbb{P}_u$ with a slight abuse of notation. We seek to learn a controller u_η and a certificate V_θ such that, with high confidence, the certificate conditions hold for a new trajectory sampled from $\mathbb{P}_{u_\eta}$. This is a generalisation problem: ensuring that a controller optimised on a finite training set remains valid in unseen scenarios.

A. Probabilistic Guarantees

We frame the synthesis problem within the context of PAC compression learning. Given a user-specified confidence level $\beta \in (0,1)$, we aim to determine a risk level $\epsilon \in (0,1)$ such that the probability of the synthesised certificate failing on a new trajectory is bounded by ϵ. Formally, denoting as $\mathcal{D} = \{\xi^i\}_{i=1}^N$ the dataset of trajectories generated by an initial controller u_0, and as $\mathcal{A}$ our synthesis algorithm producing $(V_\theta, u_\eta) = \mathcal{A}(\mathcal{D})$, we require:

$$\mathbb{P}^N \left\{ \mathcal{D} \in \Xi^N_{u_0,\mathcal{V}} : \mathbb{P}_{u_\eta}\{\xi \in \Xi_{u_\eta,\mathcal{V}} : \neg\phi_{\text{reach}}(\xi)\} \leq \epsilon \right\} \geq 1-\beta \tag{III.1}$$

where $\neg\phi_{\text{reach}}(\xi)$ indicates that a trajectory ξ violates the reachability property. To establish this bound, we utilise the concept of a *compression set* $\mathcal{C}_N \subseteq \mathcal{D}$. A subset of the data is a compression set if the algorithm produces the same output when trained on this subset as it does when trained on the full dataset: $\mathcal{A}(\mathcal{C}_N) = \mathcal{A}(\mathcal{D})$. The cardinality of this set, $C_N = |\mathcal{C}_N|$, serves as a measure of the complexity of the solution relative to the data, and is known as the *compression*. We rely on the following assumptions regarding the distribution and the algorithm to retain the PAC bound from [1]:

Assumption III.1 (Non-concentrated Mass)**.** *For any fixed controller u, the induced probability measure satisfies $\mathbb{P}_u\{\xi\} = 0$ for any single trajectory $\xi \in \Xi_{u,\mathcal{V}}$.*

Assumption III.2 (Algorithmic Properties)**.** *The synthesis algorithm $\mathcal{A}$ satisfies the following properties:*

(i) ***Preference:*** *If a subset $\mathcal{C}_1 \subseteq \mathcal{C}_2 \subseteq \mathcal{D}$ is not a compression set for $\mathcal{C}_2$, it remains not a compression set of superset $\mathcal{C}_2 \cup \{\xi\}$ for any $\xi \in \Xi$.*

(ii) ***Non-associativity:*** *If $\mathcal{C}$ compresses a dataset augmented with any future sample, it must compress the dataset augmented with all future samples.*

(iii) ***Controller Evaluation:*** *The output of the algorithm remains unchanged if the input trajectories $\{\xi^i\}_{i=1}^N$ are re-generated using the synthesised controller u_η instead of the initial controller u_0.*

Note that these assumptions are equivalent to those in [1], with the addition of the last assumption required to ensure convergence with a varying controller. Under these assumptions, the following theorem provides the link between the compression C_N and the risk ϵ.

Theorem III.1 (Generalisation Bound)**.** *Consider any algorithm $\mathcal{A}$ satisfying Assumption III.2 and generating a valid certificate V_θ for property II.1 from i.i.d. trajectories $\{\xi^i\}_{i=1}^N$ satisfying Assumption III.1. Fix $\beta \in (0,1)$. Let $k \in \{0,\dots,N\}$. If $k < N$, let $\epsilon(k,\beta,N)$ be the unique solution to the polynomial equation in ϵ in the interval $[k/N, 1]$:*

$$\frac{\beta}{2N}\sum_{m=k}^{N-1}\frac{\binom{m}{k}}{\binom{N}{k}}(1-\epsilon)^{m-N} + \frac{\beta}{6N}\sum_{m=N+1}^{4N}\frac{\binom{m}{k}}{\binom{N}{k}}(1-t)^{m-N} = 1. \tag{III.2}$$

while for $k = N$, let $\epsilon(N,\beta,N) = 1$. Then, if the algorithm returns a compression set of size $C_N = |\mathcal{C}_N|$, the risk bound in (III.1) holds with $\epsilon = \epsilon(C_N,\beta,N)$.

A rudimentary proof is provided in Appendix A.

B. Controller constraints

To ensure the control inputs respect the hyperrectangular actuation limits of the dynamical system $u \in [u_{\min}, u_{\max}]$, we apply the following transformation to the network output $\hat{u}_\eta(x)$:

$$u_\eta(x) = \frac{u_{\max}-u_{\min}}{2}\tanh(\hat{u}_\eta(x)) + \frac{u_{\max}+u_{\min}}{2}, \tag{III.3}$$

where $\tanh(\cdot)$ is applied elementwise, and $u_{\max}, u_{\min}$ are vectors defining the limits of the hyperrectangular control region $\mathcal{U}$. This ensures the controller is valid by construction, removing the need to enforce input constraints via the loss function.

C. Loss function

The optimisation objective is defined via a loss function $\mathcal{L}(\theta,\eta,\xi,v) := l^{\Delta}(\theta,\eta,\xi,v) + l^s(\theta)$. The term $l^s(\theta)$ is sample-independent and enforces the structural requirements on the certificate (II.2)–(II.5). Without loss of generality, we can assume this term can be driven to zero. This occurs as, given a sufficiently

[1]It is relevant to highlight how both the learned certificate $V_\theta(x)$ and controller $u_\eta(x)$ carry a direct dependency on the number of samples N. This dependency has been dropped here to ease the notation.

[2]This occurs due to the absence of stochasticity in the system dynamics for a given realisation $\bar{v}$ of the uncertain parameter.

expressive function approximator, we can find a certificate which satisfies the above structural requirements. As in [1, Section 4.3], we approximate the loss spatial integrals via summation over dense grid sets $\mathcal{X}_{\overline{G}}$, $\mathcal{X}_I$, and $\mathcal{X}_\partial$:

$$\begin{aligned} l^s(\theta) := &\frac{1}{|\mathcal{X}_{\overline{G}}|}\sum_{x\in\mathcal{X}_{\overline{G}}}\max\{0,-\delta-V_\theta(x)\} \\ &+\frac{1}{|\mathcal{X}_I|}\sum_{x\in\mathcal{X}_I}\max\{0,V_\theta(x)\} \\ &+\frac{1}{|\mathcal{X}_\partial|}\sum_{x\in\mathcal{X}_\partial}\max\{0,-V_\theta(x)\}, \end{aligned} \tag{III.4}$$

where $\mathcal{X}_{\overline{G}}$ approximates the set of points inside the domain but outside the goal region $X \setminus X_G$, $\mathcal{X}_I$ approximates the initial set X_I, and $\mathcal{X}_\partial$ approximates the domain border ∂X. The sample-dependent loss l^Δ enforces the decrease condition (II.6) along the sampled trajectories. Crucially, this term depends on both the certificate and the controller parameters. For reachability, the loss is defined as:

$$\begin{aligned} l^\Delta(\theta,\eta,\xi,v) := \max\Big\{0, \max_{k=0,\dots,k_G-1}\Big(V_\theta(x[k+1]) \\ - V_\theta(x[k])\Big) - \frac{1}{T}\Big(\sup_{y\in\mathcal{X}_I}V_\theta(y)+\delta\Big)\Big\}, \end{aligned} \tag{III.5}$$

where the dependency on η and v emerge from the computation of $x[k+1] = f(x[k], u_\eta(x[k]), v)$, and v is the realisation of the uncertain parameter corresponding to trajectory ξ. Minimising this loss implies driving the system towards the goal set while satisfying the certificate's decrease requirement.

D. Algorithms

To solve the optimisation problem while identifying the compression set, we employ a two-stage iterative procedure. Due to constraints on the page count for this paper, the pseudocode for the algorithms has been included in Appendix D. The *Inner Loop* (Algorithm A.1) optimises the certificate parameters θ and controller parameters η for a fixed set of trajectories using a modified SSD scheme. This algorithm proceeds similarly to [1, Algorithm 1], with modified gradient computations that account for the controller parameters and an altered "jump" condition that yields improved performance. Specifically, rather than triggering a "jump" whenever a subgradient is misaligned, we continue the iterations until the parameters achieve zero loss on the current running compression set and only then "jump" to the sample attaining the current maximum loss. This mechanism provides an alternative way to balance exploitation and exploration: we first optimise with respect to the present sample until the controller satisfies the associated conditions (exploitation), and subsequently augment the compression set $\mathcal{C}_N$ with a new sample once this has been achieved (exploration). This behaviour is illustrated in Figure III.1.

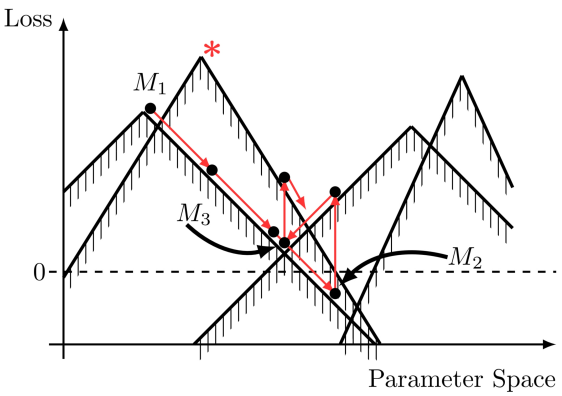


Fig. III.1: Graphical representation of Algorithm A.1.

The *Outer Loop* (Algorithm A.2) manages the trajectory generation. Since the trajectories ξ depend on the controller u_η, they must be updated as η evolves. The outer loop iteratively calls the inner loop to update parameters, then regenerates the trajectories $\mathcal{S}_k$ using the new controller u_{η_k} and the sampled parameters $\{v^i\}$. This process repeats until the parameters converge and the loss on the active trajectories is zero. The set of samples $\mathcal{R}_N$ discarded throughout this process forms the final compression set used to calculate the risk ϵ. The combination of Algorithms A.1–A.2 also satisfies Assumption III.2. A proof sketch for this is provided in Appendix B. Additionally, throughout numerical examples (see Section V), the synthesis procedure outlined above exhibited termination for all trial initial conditions. Although a proof is not provided in this paper, we hypothesise that, due to the similarity with [1, Algorithm 1–Algorithm 2], the nested Algortihm A.1–Algortihm A.2 procedure is guaranteed to terminate, driving loss function $\mathcal{L}(\theta,\eta,\xi,v)$ to zero[3].

IV. Control Perspective

For the discrete-time system (I.1) and cost functional (I.2), the optimal Value Function $V^*(x)$ satisfies the Bellman Equation (I.3). This equation can be directly obtained from the Principle of Optimality (using Theorem 20.1 in [4] as starting point, we detailed this derivation in Appendix C). The connection between $V^*(x)$ and the control certificate function $V_\theta(x)$ becomes clear when examining the structure of the decrease certificate condition (II.6), which requires the function V_θ to decrease along system trajectories. Fixing a realisation of the uncertain parameter $\bar{v}$, the decrease condition (II.6) is of the form

$$V_\theta\big(f_{\bar{v}}(x,u_\eta)\big) - V_\theta(x) \le \gamma, \tag{IV.1}$$

where γ depends on the *supremum* attained by $V_\theta(x)$ over the initial set X_I and design parameter δ. Re-

[3]Note that the sample independent loss function $l^s(\theta)$ is guaranteed to converge to zero since it is unchanged compared to its formulation in [1], thus the statement only depends on the convergence of sample-dependent loss function $l^\Delta(\theta,\eta,\xi,v)$.

arranging the Bellman equation (I.3), we obtain that the optimal value function $V^*(x)$ must satisfy the condition $V^*\big(f_{\bar{v}}(x,u^*)\big) - V^*(x) = -L(t,x,u^*)$, where $u^\star$ is the optimal control policy defined as $u^*(x) := \text{argmin}_u\{L(t,x,u) + V^*\big(f_{\bar{v}}(x,u)\big)\}$. Relaxing the condition, for an arbitrary controller u the value function $V(x)$ must satisfy:

$$V\big(f_{\bar{v}}(x,u)\big) - V(x) \leq -L(t,x,u). \tag{IV.2}$$

For a standard optimal control problem with a positive running cost $L(t,x,u) > 0$, identifying the certificate decay term γ in (IV.1) with the negative term $-L(t,x,u)$ in (IV.2), we observe that the certificate condition is structurally equivalent to the relaxation of the Bellman equation. Consequently, for a choice of the running cost L compatible with the reachability objective (e.g., a quadratic regulator cost that penalises distance from the centre of the goal set), the sample-dependent loss l^Δ in (III.5) acts as to minimise the upper bound of the *Bellman residual*:

$$R(\theta,\eta) := \left|V_\theta\big(f_{\bar{v}}(x,u_\eta(x))\big) - V_\theta(x) + L(t,x,u_\eta(x))\right|,$$

By driving l^Δ to zero, we force the learned certificate V_θ to behave like a Control Lyapunov Function that upper-bounds the optimal cost-to-go, whilst simultaneously learning a controller u_η. Invoking *inverse optimality* [5, Section 3.5] (which states that a stabilising control law derived from a Lyapunov-like function is optimal with respect to some meaningful cost functional), the resulting control law u_η approximates the optimal policy u^* for a given choice of initial set X_I. As demonstrated through numerical experiments (see Section V), the policy learned via this robust certificate synthesis closely mirrors the optimal policy derived from solving the HJB equation.

V. Numerical Experiments

To validate the theoretical connection between the synthesised certificates and optimal control, we apply our methodology to a benchmark unstable dynamical system. We compare the robust controller learned via the synthesis procedure of Algorithms A.1–A.2 against the optimal policy derived from solving the discrete-time Bellman equation on a grid. All experiments were implemented in `Python 3.12`. Simulations were carried out on the `baserver.cs.ox.ac.uk` server, which contains 80 2.5 GHz CPUs and 125 GB of RAM. The code used to generate the results presented in this section is made available at the following repository[4]:

https://github.com/ClouD-161803/fossil_scenario.git

[4]A `Linux` distribution is recommended for this repository.

A. Problem Setup

We consider the unstable planar system with state vector $x[k] = (x_1[k], x_2[k])^T$ evolving according to:

$$x[k+1] = \begin{bmatrix} 0.7 & 0.8 \\ v & 1.4 \end{bmatrix} x[k] + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u(x[k]), \tag{V.1}$$

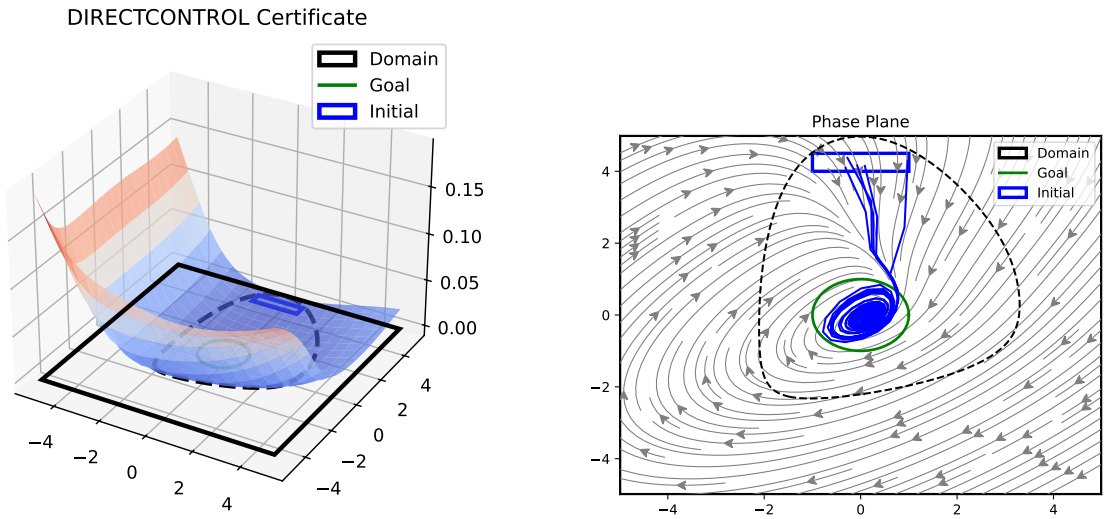
where the uncertain parameter v is distributed in the interval $\mathcal{V} := [-0.5, -0.3]$. The (univariate) control input is constrained to the set $\mathcal{U} := [-4, 4]$. The objective is to drive the system from an initial set $X_I = [-1,1] \times [4, 4.5]$ to the unit-radius Euclidean ball centred at the origin: we define the goal set as $X_G = \{x \in \mathbb{R}^2 : \|x\|_2 \leq 1\}$. The time horizon is chosen as $T = 100$, and the discretisation interval as $T_d = 0.1$ For the data-driven synthesis, we parameterise both the certificate V_θ and the pre-activation controller $\hat{u}_\eta$ using fully connected neural networks with 2 hidden layers of 5 neurons each and sigmoid activation functions. The final control output is passed through a regularising function as per (III.3) to satisfy the actuation limits. We choose both probability distributions $\mathbb{P}_x$ and $\mathbb{P}_v$ as uniform across their respective domains, and draw $N = 1000$ independent sample pairs for training. Using an initially random uniform controller u_0, we uncover the corresponding initial trajectories $\{\xi^i\}_{i=1}^N$. We set the required confidence level to $1 - \beta = 1 - 10^{-5}$. We implement Algorithms A.1–A.2 as a `PyTorch` solver, and set up the loss function $\mathcal{L}$ as the sum of a sample-independent component l^s as in III.4 and a sample-dependent component l^Δ as in III.5.

B. Certificate and Controller Synthesis Results

The synthesis procedure converged after $60,190$ seconds. The resulting compression set $\mathcal{C}_N$ yielded a PAC risk bound of $\epsilon = 0.068$ (computed using Theorem III.1). Figure V.1 visualises the learned certificate. The surface plot (Figure V.1a) demonstrates the function's global structure, while the phase plane (Figure V.1b) shows the zero-level set $V_\theta(x) = 0$ (dashed line) and the compression set trajectories (blue lines). As expected, the certificate is negative on the initial set (blue rectangle) and positive outsite the domain (black box), and the system trajectories (grey arrows) flow inwards across the level sets towards the goal set (green circle), with no recorded violations of certificate conditions (II.2)–(II.6).

C. Comparison with Optimal Control

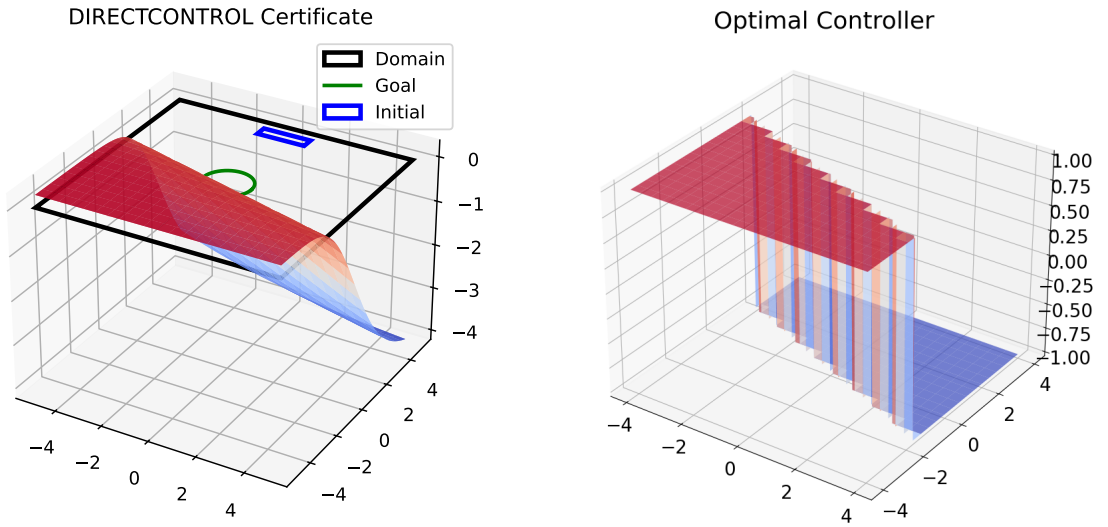
To investigate the optimality of the learned controller, we numerically solved the Bellman equation (I.3) for the nominal system via Dynamic Programming (DP), using a realisation of parameter value v as the expected value over its range, $\bar{v} = -0.4$. We employed a discrete 50×50 grid over the domain $X := [-4, 4]^2$. The running cost function was chosen as a quadratic regulator $L(t,x,u) := x^T Q x + u^T R u$ with Q and $R := 0.1$ and zero terminal cost $K(x[T]) = 0$, simultaneously



(a) Surface plot.

(b) Phase plane plot.

Fig. V.1: Reachability certificate synthesised from $N = 1000$ trajectories. The certificate acts as a Lyapunov function, decreasing along state trajectories from the initial set X_I (blue box) into the goal set X_G (green circle).



(a) Synthesised Robust Controller $u_\eta(x)$.

(b) Optimal Controller $u^*(x)$.

Fig. V.2: Comparison between the data-driven controller and the optimal controller. The learned policy (left) smooths the optimal switching surface (right) while maintaining the correct global structure required to stabilise the system (in spite of vertical shift and stretching due to the dependency on an initial set).

penalising distance from the origin and control effort (although priority was given to the former). Figure V.2 compares the data-driven controller (Figure V.2a) with the numerically derived optimal policy (Figure V.2b). Both controllers exhibit a "bang-bang" like structure, applying maximal negative effort in the region where $x_1, x_2 > 0$ to counteract the instability. However, several key differences illustrate the specific nature of our data-driven approach. First, the synthesised controller is much smoother than the optimal policy. This is a direct consequence of the $\tanh$ activation function used in the neural network parameterisation, which acts as a regulariser compared to the discrete minimisation in DP. Second, the data-driven controller is *robust* to uncertainty: it was trained to satisfy the decrease condition across the entire distribution of $v \in \mathcal{V}$, whereas the optimal policy is specific to the nominal case of $\bar{v}$. The conservative slope of the data-driven policy reflects the need to maintain stability even for the worst-case parameter realisations. Finally, the parametrised controller applies zero (and not positive) control efforts for stable states. This highlights the dependency of the data-driven framework on the choice of specific initial and goal sets, which direct the

optimiser towards specific portions of the state space. On the other hand, the optimal controller directly computes the best action for any initial sate and with a singleton goal set (the origin). Nonetheless, these results confirm our theoretical assertion: by minimising the certificate decrease violation, the synthesis procedure finds a controller that approximates the optimal policy, providing a data-driven, scalable, and robust alternative to directly solving the HJB equation.

VI. Conclusion

This paper has presented a framework that provides a first step towards bridging data-driven certificate and control synthesis and classical optimal control theory. By extending the SSD algorithm to the problem of joint controller and certificate synthesis, we have demonstrated that scalable, data-driven verification techniques can approximate optimal control policies. We established a connection between the certificate decrease condition (II.6) and the discrete-time Bellman equation (I.3), highlighted in conditions (IV.1)–(IV.2). The resulting certificate function V_θ approximates the optimal Value Function $V^*(x)$, and the synthesised controller u_η mirrors the optimal policy u^* that minimises the underlying cost functional (I.2). This theoretical insight was validated through numerical experiments on an unstable system. The data-driven controller approximated the "bang-bang" structure of the true optimal policy derived via DP. Furthermore, the use of neural network parameterisation provided a smooth, robust approximation that naturally handles parametric uncertainty, constituting a significant advantage over standard grid-based DP methods which struggle with the "curse of dimensionality" and model uncertainty. Future work will focus on further formalising the link, potentially by deriving explicit bounds on the suboptimality of the learned controller as a function of the compression set size. A complete connection between the data-driven certificate synthesis approach and optimal control theory might allow future learning-enabled control systems to be deployed with both optimality and strong probabilistic certificate satisfaction guarantees.

References

[1] L. Rickard, A. Abate, and K. Margellos. "Data-Driven Certificate Synthesis". In: *Automatica (to appear)* 1-18 (2025).

[2] M. Campi and S. Garatti. *Introduction to the Scenario Approach*. SIAM Series on Optimization, 2018.

[3] M. C. Campi and S. Garatti. "Compression, Generalization and Learning". In: *J. Mach. Learn. Res.* 24 (2023), 339:1–339:74.

[4] R. Beeson. *Optimal Control: MAE 546 Lecture Notes*. Princeton University. 2025.

[5] R. Sepulchre, M. Jankovic, and P. V. Kokotovic. *Constructive Nonlinear Control*. Berlin, Heidelberg: Springer-Verlag, 1997.

APPENDIX

A. Proof of Theorem III.1

This proof is largely adapted from [1] to include the new assumption on the learned controller.

Proof. Fix $\beta \in (0,1)$ and let $\mathcal{C}_N$ be the compression set returned by the algorithm. We interpret the algorithm $\mathcal{A}$ as a mapping from the dataset to a decision (V_θ, u_η). If the certificate conditions are violated on a new sample ξ, the algorithm must produce a different output when trained on $\mathcal{D} \cup \{\xi\}$ to satisfy the condition for ξ. This implies a change in the compression set. Specifically, let $\mathcal{C}_N^+$ be the compression set for the augmented dataset. We have the inclusion:

$$\{\xi \in \Xi : \neg\phi_{\text{reach}}(\xi)\} \subseteq \{\xi \in \Xi : \mathcal{C}_N \neq \mathcal{C}_N^+\}.$$

Furthermore, Assumption III.2 (iii) ensures that re-evaluating the algorithm with trajectories generated by the learnt controller u_η does not alter the compression set. Thus, the probability of failure on the true distribution $\mathbb{P}_{u_\eta}$ is bounded by the probability that the compression set changes upon observing a new sample:

$$\mathbb{P}_{u_\eta}\{\xi \in \Xi_{u_\eta,\mathcal{V}} : \neg\phi_{\text{reach}}(\xi)\} \leq \mathbb{P}_{u_\eta}\{\xi \in \Xi_{u_\eta,\mathcal{V}} : \mathcal{C}_N \neq \mathcal{C}_N^+\}.$$

Adopting [3, Theorem 7], this probability is bounded by $\epsilon(C_N, \beta, N)$ with confidence $1-\beta$, where ϵ is the solution to the polynomial equation provided in the theorem statement. □

B. Properties of the Synthesis Algorithms

We briefly verify that the synthesis procedure in Algorithms A.1–A.2 satisfies Assumption III.2.

Proof. We consider each of the properties in turn:

(i) Preference: The inner loop (Algorithm A.1) selects samples based on maximal loss. If a subset is not a compression set, it implies there exists a sample in the larger set with higher loss that would be selected. Adding more samples ($\mathcal{D} \cup \{\xi\}$) cannot resolve this conflict without including the problematic sample, preserving the non-compression property (and we cannot add repeated samples due to Assumption III.1).

(ii) Non-associativity: This holds by construction of the greedy addition strategy in the inner loop. If a set $\mathcal{C}$ suffices as a compression set for $\mathcal{D} \cup \{\xi\}$ for any ξ, it implies all other samples have non-positive loss relative to $\mathcal{C}$, which means $\mathcal{C}$ suffices as a compression set for the union of all samples.

(iii) Controller evaluation: The outer loop (Algorithm A.2) terminates only when the controller parameters η converge ($\eta_k \approx \eta_{k-1}$). At this fixed point, re-generating trajectories with u_{η_k} yields the same set of trajectories $\mathcal{S}_k$ used in the final optimisation step. Consequently, re-running the algorithm produces the exact same parameters and compression set, satisfying condition (iii).

□

C. Derivation of the Discrete-Time Bellman Equation

We derive the discrete-time Bellman equation starting from the fundamental Principle of Optimality provided in [4, Page 146].

Proof. Consider the value function $V(t,x)$, representing the minimum cost-to-go from state x at time t. According to [4, Theorem 20.1], the Principle of Optimality states that for any time increment $\Delta t > 0$:

$$V(t,x) = \inf_{u \in \mathcal{U}(t,t+\Delta t)} \left\{ \int_t^{t+\Delta t} L(s, X_s^{(t,x)}, u_s) ds + V(t+\Delta t, X_{t+\Delta t}^{(t,x)}) \right\}.$$

To transition to the discrete-time formulation used in our data-driven framework, we apply the following assumptions:

1) Zero-order hold: The control input u_s is constant over the interval $[t, t+\Delta t]$. Let $t_k = kT_d$ denote the current time step and $u[k]$ the constant control applied.

2) Euler approximation: For a small time step Δt, the integral of the running cost can be approximated as:
$$\int_{t_k}^{t_k+\Delta t} L(s, X_s^{(t,x)}, u_s)\, ds \approx L(t_k, x[k], u[k]) \cdot \Delta t.$$

3) Parameter-free dynamics: We fixed a realisation of the uncertain parameter v as $\bar{v}$, yielding a parameter-free version of control dynamics (I.1):
$$f(x[k], u[k]; \bar{v}) \equiv f_{\bar{v}}(x[k], u[k])$$

Substituting these approximations into the principle of optimality, absorbing the time step Δt into the definition of the discrete running cost L, and considering the update from control dynamics (I.1) we obtain the recurrence relation

$$V(x) = \min_{u_k \in \mathcal{U}} \left\{ L(t_k, x[k], u_k) + V\big(f_{\bar{v}}(x[k], u_k)\big) \right\}$$

and thus arrive at (I.3) by substituting optimal value function $V^*(x)$. □

Note that, in Section I, the dynamics have been hidden into $x[k+1]$ to ease the notation.

D. Pseudocode for algorithms

We provide pseudocode for the inner loop (Algorithm A.1) and outer loop (Algorithm A.2) below:

Algorithm A.1 Inner Loop Optimisation

function $\mathcal{A}(\theta, \eta, \mathcal{D})$
  Set $k \leftarrow 0$, $\mathcal{C} \leftarrow \varnothing$
  Fix $\mathcal{L}_1 < \mathcal{L}_0$ with $|\mathcal{L}_1 - \mathcal{L}_0| > \zeta$ ▷ ζ is a fixed tolerance
  while $l^s(\theta) > 0$ **do** ▷ Minimise sample-independent loss first via gradient descent
    $\theta \leftarrow \theta - \alpha \nabla_\theta l^s(\theta)$
  end while

---

  repeat
    $k \leftarrow k + 1$
    $\overline{\xi} \in \operatorname{argmax}_{\tilde{\xi} \in \mathcal{D}} \mathcal{L}(\theta, \eta, \tilde{\xi})$ ▷ Find worst-case sample in dataset
    $\overline{\xi}_{\mathcal{C}} \in \operatorname{argmax}_{\tilde{\xi} \in \mathcal{C}} \mathcal{L}(\theta, \eta, \tilde{\xi})$ ▷ Find worst-case sample in compression set
    $\overline{g}_{\mathcal{D}} \leftarrow \big(\nabla_\theta \mathcal{L}(\theta, \eta, \overline{\xi}), \nabla_\eta \mathcal{L}(\theta, \eta, \overline{\xi})\big)$ ▷ Compute subgradients of loss function for $\overline{\xi}$
    $\overline{g}_{\mathcal{C}} \leftarrow \big(\nabla_\theta \mathcal{L}(\theta, \eta, \overline{\xi}_{\mathcal{C}}), \nabla_\eta \mathcal{L}(\theta, \eta, \overline{\xi}_{\mathcal{C}})\big)$ ▷ Approximate subgradients of loss function for $\overline{\xi}_{\mathcal{C}}$

---

    if $\mathcal{L}(\theta, \eta, \overline{\xi}_{\mathcal{C}}) \leq 0$ **then** ▷ Subgradient descent & update compression set
      $\theta \leftarrow \theta - \alpha \overline{g}_{\mathcal{D}}[\theta]$
      $\eta \leftarrow \eta - \alpha \overline{g}_{\mathcal{D}}[\eta]$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\xi\}$ ▷ Add violating sample to compression set
    else ▷ Optimise on current compression set
      $\theta \leftarrow \theta - \alpha \overline{g}_{\mathcal{C}}[\theta]$
      $\eta \leftarrow \eta - \alpha \overline{g}_{\mathcal{C}}[\eta]$
    end if

    $\mathcal{L}_k \leftarrow \mathcal{L}_{k-1}$
    if $\max_{\xi \in \mathcal{C}} \mathcal{L}(\theta, \eta, \xi) < \mathcal{L}_{k-1}$ **then**
      $\mathcal{L}_k \leftarrow \max_{\xi \in \mathcal{C}} \mathcal{L}(\theta, \eta, \xi)$, $(\theta^*, \eta^*) \leftarrow (\theta, \eta)$
    end if
  until $|\mathcal{L}_k - \mathcal{L}_{k-1}| \leq \zeta$ ▷ Iterate until tolerance is met
  return $\theta^*, \eta^*, \mathcal{C}_N = \mathcal{C} \cup \operatorname{argmax}_{\xi \in \mathcal{D}} \mathcal{L}(\theta, \eta, \xi)$
end function

Algorithm A.2 Controller Synthesis (Outer Loop)

Initialise θ_0, η_0 randomly
Sample parameters and initial states $\mathcal{D}_0 \leftarrow \{x^i[0], v^i\}_{i=1}^N \sim \mathbb{P}^N$
Generate initial trajectories $\mathcal{S}_0 \leftarrow \{\xi^i\}_{i=1}^N \in \prod_{i=1}^N \Xi_{u_{\eta_0}, v^i}$ using u_{η_0}
Set $\mathcal{I} \leftarrow [1, \ldots, N]$, $k \leftarrow 0$, $\tilde{\mathcal{C}} \leftarrow \varnothing$
while $\max_{\xi \in \mathcal{S}_k} L(\theta_k, \eta_k, \xi) > 0$ **do**
  repeat
    $k \leftarrow k + 1$
    $\theta_k, \eta_k, \mathcal{C} \leftarrow \mathcal{A}(\theta_{k-1}, \eta_{k-1}, \mathcal{S}_{k-1})$
    $\tilde{\mathcal{C}} \leftarrow \tilde{\mathcal{C}} \cup \mathcal{C}$ ▷ Accumulate compression set
    Update indices $\mathcal{I}_{\tilde{\mathcal{C}}}$ corresponding to samples in $\tilde{\mathcal{C}}$
    Update trajectories $\mathcal{S}_k \leftarrow \{\xi^i\}_{i \in \mathcal{I}} \in \prod_{i \in \mathcal{I}} \Xi_{u_{\eta_k}, v^i}$ using new controller u_{η_k}
  until $\eta_k \approx \eta_{k-1}$ ▷ Controller parameters converged
  $\mathcal{S}_k \leftarrow \mathcal{S}_{k-1} \setminus \mathcal{C}$ ▷ Discard satisfied samples
  Update index set $\mathcal{I}$ to remove discarded indices
end while
return $\theta_k, \eta_k, \mathcal{R}_N$ (indices of all discarded samples)