Proceedings of Machine Learning Research vol 242:1–18, 2024

Linear Supervision for Nonlinear, High-Dimensional Neural Control and Differential Games

William Sharpless WSHARPLESS@UCSD.EDU
Department of Mechanical and Aerospace Engineering, University of California San Diego

Zeyuan Feng ZEYUANF@STANFORD.EDU
Department of Aeronautics and Astronautics, Stanford University

Somil Bansal SOMIL@STANFORD.EDU
Department of Aeronautics and Astronautics, Stanford University

Sylvia Herbert SHERBERT@UCSD.EDU
Department of Mechanical and Aerospace Engineering, University of California San Diego

Abstract

As the dimension of a system increases, traditional methods for control and differential games rapidly become intractable, making the design of safe autonomous agents challenging in complex or team settings. Deep-learning approaches avoid discretization and yield numerous successes in robotics and autonomy, but at a higher dimensional limit, accuracy falls as sampling becomes less efficient. We propose using rapidly generated *linear* solutions to the partial differential equation (PDE) arising in the problem to accelerate and improve learned value functions for guidance in high-dimensional, *nonlinear* problems. We define two programs that combine supervision of the linear solution with a standard PDE loss. We demonstrate that these programs offer improvements in speed and accuracy in both a 50-D differential game problem and a 10-D quadrotor control problem.

1. Introduction

The goal of this work is to solve differential games and optimal controllers for high-dimensional, nonlinear problems. These problems arise in many fields, including multi-agent robotics, medicine, and finance (Bansal et al. (2017)). High-dimensional nonlinearity renders most approaches to autonomy in these domains infeasible, making this a difficult challenge. Ultimately, we aim to generate a value function that an agent or team may follow to safely guide itself amidst disturbances or antagonists.

A popular method for the analysis of differential games is to solve its corresponding Hamilton-Jacobi (HJ) partial differential equation (PDE) (Evans and Souganidis (1984a)). The sub-zero level set of the solution represents the *Backwards Reachable Tube* (BRT) of states that may reach or avoid a target despite any bounded disturbance (Mitchell et al. (2005); Altarovici et al. (2013)). Moreover, the gradient of the solution defines the corresponding control law. This approach is widely successful (Huang et al. (2011); Chen et al. (2015); Jeong and Bajcsy (2024)). However, numerical integration of the PDE with dynamic programming requires a grid, leading to an exponential growth in computation with dimension, becoming infeasible beyond seven. To overcome this "curse of dimensionality", many directions have been explored. Decomposition of the system offers a conservative value when feasible (Chen et al. (2018); He et al. (2023)). Multiple methods directly approximate the BRT, largely

for linear and polynomial dynamics (Althoff et al. (2008); Majumdar and Tedrake (2017); Kousik et al. (2018); Yang et al. (2022); Kochdumper and Althoff (2023)). These formal methods offer strong guarantees but can suffer from conservativeness.

For convex games with linear time-varying dynamics, several have recognized that a solution to the HJ-PDE is given by the Hopf formula (Darbon and Osher (2016); Chow et al. (2017); Kurzhanski (2014)). By computing the value independently in space and time, this approach offers orders of magnitude in acceleration. Some have used this to approximate the value in nonlinear settings (Kirchner et al. (2018); Lee and Tomlin (2019); Sharpless et al. (2023)), but few offer guarantees (Kurzhanski (2014); Liu et al. (2024); Sharpless et al. (2024b)), typically at the cost of being overly conservative.

With the advent of deep learning, many have sought to learn the value function. Typically, a residual based PDE loss (Raissi et al. (2017); Han et al. (2018)) is used to train the approximation to satisfy the HJ equation (Djeridane and Lygeros (2006); Darbon et al. (2020)). These methods have explored this in reinforcement frameworks (Fisac et al. (2019); Ganai et al. (2024)). One may also learn the dynamic programming procedure itself (Esteve-Yagüe et al. (2024)). Amongst these, we build on Bansal and Tomlin (2021), which demonstrated that a sinusoidal architecture (Sitzmann et al. (2020)) was well-suited for the problem. Learning the HJ-PDE solution has proved to increase the feasible dimension, however, many of the works demonstrate a nebulous limit beyond which learned solutions deteriorate.

We seek to combine the scalability of linear HJ-PDE's with the efficacy of learned methods to obtain accurate value functions for nonlinear, high-dimesnional games. Namely, we introduce supervision with a linear solution to the PDE loss program to learn nonlinear solutions. This shifts the learning process from generation to refinement, and results in a faster and more accurate learning process with tighter probablistic guarantees. In summary,

1. We propose two semi-supervision programs for learning high-dimensional, nonlinear value functions.
2. We demonstrate their potential with a 50-D game and 10-D quadrotor control.
3. We provide theoretical insights behind the effectiveness of the proposed programs.

2. Problem Statement

We aim to guide the agent state $x \in \mathbb{X} \triangleq \mathbb{R}^n$ that evolves by nonlinear dynamics f such that we optimize the cost $J_{\mathcal{T}} : \mathbb{X} \to \mathbb{R}$ of the trajectory at some time $\tau \in \mathbb{T} \triangleq [t, t_f]$. Let the cost be defined such that its sub-zero level set represents a compact, convex target $\mathcal{T} \subset \mathbb{X}$. While the agent controls the evolution, assume it plays against an opponent who is seeking the opposite score of $J_{\mathcal{T}}$ in a zero-sum fashion. Hence, if the agent seeks to reach/avoid the target, the opponent seeks to avoid/reach it. Note, if the disturbance action is null, the game is an optimal control problem. In this section, let the goal be of reaching the target.

Formally, we seek to solve the differential game,

$$\begin{cases} \underset{\mathfrak{d}\in\mathfrak{D}(t)}{\text{maximize}} \ \underset{\mathsf{u}\in\mathbb{U}(t)}{\text{minimize}} \ \underset{\tau\in\mathbb{T}}{\text{minimize}} & J_{\mathcal{T}}(\mathsf{x}(\tau)) \\ \text{subject to} & \dot{\mathsf{x}}(\tau) = f(\mathsf{x}(\tau), \mathsf{u}(\tau), \mathsf{d}(\tau), \tau) \\ & \mathsf{u}(\tau) \in \mathcal{U}, \ \ \mathsf{d}(\tau) = \mathfrak{d}[\mathsf{u}](\tau) \in \mathcal{D}, \\ & \mathsf{x}(t) = x. \end{cases} \tag{1}$$

Here, $\mathsf{x} : \mathbb{T} \to \mathbb{X}$ is the unique state trajectory beginning at $\mathsf{x}(t) = x$. The trajectory arises from the effect of the agent's control signal $\mathsf{u} : \mathbb{T} \to \mathcal{U}$ and the opponent's disturbance signal $\mathsf{d} : \mathbb{T} \to \mathcal{D}$, which are in sets of measurable functions $\mathbb{U}(t)$ and $\mathbb{D}(t)$ that output actions in the compact, convex sets $\mathcal{U} \subset \mathbb{R}^{n_u}$ and $\mathcal{D} \subset \mathbb{R}^{n_d}$ respectively. This disturbance signal is the output of the opponent's strategy $\mathfrak{d} : \mathbb{U}(t) \to \mathbb{D}(t)$, which is in the set of non-anticipative strategies $\mathfrak{D}(t)$ s.t. for $\mathfrak{d} \in \mathfrak{D}(t)$, $\forall s \in [t, \tau], u(s) = \hat{u}(s) \implies \mathfrak{d}[u](s) = \mathfrak{d}[\hat{u}](s)$ Mitchell et al. (2005). Let f be Lipschitz continuous in $(\mathsf{x}, \mathsf{u}, \mathsf{d})$ and continuous in τ.

For an initial condition (x, t), the optimal value of this game is given by

$$V(x,t) = \sup_{\mathfrak{d} \in \mathfrak{D}(t)} \inf_{u(\cdot) \in \mathbb{U}(t)} \inf_{\tau \in \mathbb{T}} J_{\mathcal{T}}(\mathsf{x}(\tau)). \tag{2}$$

Notably, this value describes where the agent may win or lose (Mitchell et al. (2005)) as

$$V(x,t) < 0 \iff x \in \mathcal{R}_{\mathcal{T}}(t) \triangleq \{x \in \mathbb{X} \mid \forall \mathfrak{d} \in \mathfrak{D}(t),\ \exists u(\cdot) \in \mathbb{U}(t) \text{ s.t. } \mathsf{x}(\tau) \in \mathcal{T},\ \tau \in \mathbb{T}\}. \tag{3}$$

$\mathcal{R}_{\mathcal{T}}$ is the BRT that contains all states from which the agent can reach the target despite any disturbance strategy. Better yet, this value also offers the optimal policy for any point in time or space, $u^*(t; x) = \arg\min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \langle \nabla_x V, f(x, u, d, t) \rangle$.

2.1. Hamilton-Jacobi-Bellman Solution for V

Applying Bellman's principle of optimality to the value in (2), Evans and Souganidis (1984c) proved that V is equivalently the viscosity solution to the HJ-PDE

$$\begin{aligned} \dot{V} + \min\{0,\ H(x, \nabla_x V, \tau)\} &= 0 \qquad \text{on } \mathbb{X} \times \mathbb{T}, \\ V(x, t_f) &= J_{\mathcal{T}}(x) \quad \text{on } \mathbb{X}, \end{aligned} \tag{4}$$

where the Hamiltonian is defined by $H(x, p, t) = \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \langle p, f(x, u, d, t) \rangle$.

Thus, to solve the game, we may solve this HJ-PDE. In the years following this result, dynamic programming proved to yield high-fidelity solutions for a broad class of nonlinear systems and nonconvex games, making this a popular framework *when the system dimension is low*. Due to the need for a discrete grid, dynamic programming methods suffer from the *curse of dimensionality* and thus are infeasible for systems of dimension $n \geq 7$.

2.2. Deep Learning of the HJ-PDE Solution

To avoid computation over a discrete grid, one may also use deep learning to approximate a PDE solution (Raissi et al. (2017)). Namely, let the neural approximator V_θ be defined by

$$\begin{aligned} V_\theta(x,t) &\triangleq J_{\mathcal{T}}(x) + (t - t_f)\left(W_Y\left[\phi_{Y-1} \circ \phi_{Y-2} \circ \ldots \circ \phi_0\right]\left([x,t]^\top\right) + b_Y\right), \\ \phi_i(v) &\triangleq \sin(W_i v + b_i),\ \ i \in [1, Y] \end{aligned} \tag{5}$$

where $\phi_i : \mathbb{R}^{M_i} \to \mathbb{R}^{N_i}$ is the i-th layer of the neural network with weight matrix $W_i \in \mathbb{R}^{N_i \times M_i}$ and bias $b_i \in \mathbb{R}^{N_i}$ (Sitzmann et al. (2020)). To train this approximator, the PDE loss

$$\mathcal{L}_{PDE}(\theta; k) \triangleq \mathbb{E}_{\mathbb{X}, \mathbb{T}_k}[L_{PDE}(\theta)], \quad L_{PDE}(\theta) \triangleq \|\dot{V}_\theta + H(x, \nabla_x V_\theta, t)\|, \tag{6}$$

has proved succesful, where $\mathbb{T}_k \triangleq [t, t_k]$ is an increasing time range with $t_K \triangleq t_f$. This approach has proved to offer high-fidelity approximations of the solution V for many systems beyond the dynamic programming limit (Bansal and Tomlin (2021)). However, in the limit of dimension, this method (like other learned PDE solutions) suffers from the complexity of optimizing the PDE loss, and the approximations can deteriorate.

2.3. The Hopf Solution to HJ-PDE's with Linear Dynamics

Recently, Darbon and Osher (2016) recognized that when the dynamics in (1) are given by a linear time-varying function ℓ, an alternative to brute force grid-solving is given by the Hopf formula, a solution to specific HJ-PDEs. Namely, if the linear game is given by

$$V_\ell(x,t) = \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{u(\cdot)\in\mathbb{U}} \inf_{\tau\in\mathbb{T}} J_{\mathcal{T}}(\mathsf{x}_\ell(\tau)), \tag{7}$$

where x_ℓ is the trajectory of $\dot{\mathsf{x}}_\ell = \ell(\mathsf{x}_\ell, \mathsf{u}, \mathfrak{d}[u], \tau)$, $J_{\mathcal{T}}$ is convex, and the linear Hamiltonian $H_\ell(p,t) \triangleq \min_u \max_d \langle p, \Phi(t_f - t)(\nabla_u \ell(t_f - t)u + \nabla_d \ell(t_f - t)d)\rangle$ is convex (where Φ is the fundamental matrix of ℓ), then the value V_ℓ is given by

$$V_\ell(x,t) = -\min_{\tau\in\mathbb{T}} \min_{p\in\mathbb{R}^n} \left\{ J^\star_{\mathcal{T}}(p) - \langle \Phi(t_f - \tau)x, p\rangle + \int_0^{t_f-\tau} H_\ell(p,s)\, ds \right\} \triangleq \mathcal{H}[J_{\mathcal{T}}, H_\ell](x,t), \tag{8}$$

where $J^\star(p) : \mathbb{X} \to \mathbb{R}\cup\{+\infty\}$ is the convex-conjugate of J. This Hopf formula $\mathcal{H}$ may be *independently* solved over space with proximal algorithms as it is a non-smooth convex optimization problem in the space of $\mathbb{R}^n$. This offers orders of magnitude of acceleration and memory efficiency over dynamic programming by skirting the need to grid the space Sharpless et al. (2024a), but is *limited to linear systems* or conservative linearizations.

3. Approach

We propose two methods for marrying the ability to solve the value V_ℓ for simplified linear dynamics with the learned approach to approximating V for the true nonlinear dynamics, and discuss the theoretical validity of these approaches.

3.1. On the Relationship Between V_ℓ and V

We begin by considering a few simple results which contextualize the linear form of the game. As is true in most dynamical systems theory, the game value for a nonlinear system is intimately related to the game value with a corresponding linearized version of the system. One perspective on this is given by the following results.

Theorem 1 *Let $\mathcal{S}(t)$ be a set containing any $\mathsf{x}(\tau) \in \mathbb{X}$ s.t. $\mathsf{x}(s) \in \mathcal{T}, s \in \mathbb{T}$. If*

$$H^{\pm}_{\ell+\varepsilon}(x,p,t) \triangleq H_\ell(x,p,t) \pm \max_{\varepsilon\in\mathcal{E}(t)} \pm\langle p, \varepsilon\rangle, \tag{9}$$

where $\mathcal{E}(s) \triangleq \{\|\varepsilon\| \le \delta^*(s)\}$ and $\delta^*(s) = \max\|f - \ell\|$ is the maximum difference over $\mathcal{S}(t)\times\mathcal{U}\times\mathcal{D}\times[s,t_f]$, then for any $\tau\in\mathbb{T}, x\in\mathcal{S}(t)$,

$$\left|V - V_\ell\right| \le \mathcal{H}[J_{\mathcal{T}}, H^+_{\ell+\varepsilon}] - \mathcal{H}[J_{\mathcal{T}}, H^-_{\ell+\varepsilon}] \triangleq \epsilon^*. \tag{10}$$

See the Appendix in Sec. 6 for the proof.

Thm 1 offers a finite bound between the value of the linear and nonlinear games for any fixed time in a local region. This knowledge justifies the use of the linear value in various scenarios, including its application in the current context for learning the nonlinear value. The following Corollary highlights an intuitive and desirable property of this bound: it diminishes in the limit of the operating point.

Corollary 1 *With the linear game in (7), if ℓ is defined to be*

$$\ell(x,u,d,t) \triangleq \nabla_x f|_{m_0}(t)x + \nabla_u f|_{m_0}(t)u + \nabla_d f|_{m_0}(t)d + \nabla_t f|_{m_0} t, \tag{11}$$

with operating point $m_0 \triangleq (\mathsf{x}_0, \mathsf{u}_0, \mathsf{d}_0, \tau_0)$ and finite error $\delta^*(\tau) \sim O(\|m - m_0\|^2)$, then

$$V(x,t) = V_\ell(x,t) + \epsilon, \;\; \epsilon \in [0, \epsilon^*] \quad s.t. \quad \lim_{m \to m_0} \epsilon = 0. \tag{12}$$

Thm. 1 and Cor. 1 imply that computing V_ℓ provides a portion of the solution that is dominant near the operating point. For these reasons, we propose using the linear solution in the nonlinear program, specifically with a least-squares supervision loss.

Definition 1 (Linear Supervision Loss) *Let the linear supervision loss be defined by*

$$L_{LS}(\theta) \triangleq \rho\|V_\theta - V_\ell\| + \rho_g\|\nabla_x V_\theta - \nabla_x V_\ell\| \tag{13}$$

where $\rho, \rho_g \in \mathbb{R}$ are hyperparameters.

The least-squares loss satisfies the local Polyak-Łojasiewicz (PL) condition for wide models with smooth activations (Liu et al. (2022)). This property guarantees exponential convergence to a global optimum with (stochastic) gradient descent, an increasingly beneficial fact as the dimension rises.

3.2. Decayed Linear Semi-Supervision for Learning V

In light of the relation between V & V_ℓ in Cor. 1, and the simplicity of learning V_ℓ with supervision, we propose the following simple semi-supervision loss, combining linear supervision with the PDE loss term to learn the nonlinear value.

Definition 2 (Linear Semi-Supervision Loss, Decayed) *Let a loss be defined by*

$$\mathcal{L}_{LSS-D}(\theta; k) \triangleq (1 - \lambda_k)\mathbb{E}_{\mathbb{X},\mathbb{T}}[L_{LS}(\theta)] + \lambda_k \mathbb{E}_{\mathbb{X},\mathbb{T}}[L_{PDE}(\theta)], \quad \textit{where} \quad \lambda_0 \triangleq 0. \tag{14}$$

Here k is the program iteration and λ_k is a monotonically increasing value starting at $\lambda_0 \triangleq 0$.

The key idea is to use λ_k to gradually transition the neural value from approximating the linear solution V_ℓ, a global minimizer of L_{LS}, to approximating the nonlinear solution V, a global minimizer of L_{PDE}. We note there is no guarantee of a globally minimizing path between L_{LS} and L_{PDE} for $\lambda_0 \to \lambda_K$, but empirically we find that this works well nonetheless, as shown in Sec. 4. Rather than learn the solution from scratch, one might note that the program only needs to correct a partially true solution and, thus, has a simpler task at hand. This also allows one to avoid the gradual time-curriculum approach that previous authors have found necessary for learned PDE solutions (Bansal and Tomlin (2021)), yielding a highly accelerated and performant program.

3.3. Linear Semi-Supervision in an Augmented Game for Learning V

In addition to the decayed LSS program, we propose a more sophisticated method that learns the solution of an augmentation of the game. By defining a continuous spectrum of systems between the nonlinear and linear systems, we show that the augmented game offers a setting where L_{LS} and L_{PDE} are not in conflict. Consider the following augmented dynamics.

Definition 3 Nonlinear Spectrum Augmentation
Let an augmented system be defined by state $\tilde{\mathsf{x}} = [\mathsf{x}, \lambda]^\top$ *in* $\tilde{\mathbb{X}} \triangleq \mathbb{X} \times \mathbb{R}$ *with dynamics*

$$\dot{\tilde{\mathsf{x}}} \triangleq \begin{bmatrix} (1-\lambda)\ell(\mathsf{x}, u, d, t) + \lambda f(\mathsf{x}, u, d, t) \\ 0 \end{bmatrix} \triangleq \tilde{f}_\lambda(\tilde{\mathsf{x}}, u, d, t). \tag{15}$$

Let the augmented game be defined with the same inputs and strategies as V*, s.t.*

$$V_\lambda(x, \lambda, t) \triangleq \sup_{\mathfrak{d} \in \mathfrak{D}(t)} \inf_{\mathfrak{u} \in \mathbb{U}(t)} \inf_{\tau \in \mathbb{T}} J_{\mathcal{T}}(P\tilde{\mathsf{x}}(\tau)). \tag{16}$$

where $P : \tilde{\mathbb{X}} \to \mathbb{X}$ *represents projection from the augmented space given by* $P([x, \lambda]^\top) = x$*.*

Intuitively, this system represents a continuous spectrum of nonlinear systems between any given linear and nonlinear dynamics. Notably, this game has a few simple properties that will make it valuable for learning.

Theorem 2 *V_λ is the viscosity solution of*

$$\begin{aligned} \dot{V}_\lambda + \min\{0, H_\lambda(\tilde{x}, \nabla_x V_\lambda, t)\} = 0 \qquad & \text{on } \tilde{\mathbb{X}} \times \mathbb{T}, \\ V_\lambda(\tilde{x}, t_f) = J_{\mathcal{T}}(x) \qquad & \text{on } \tilde{\mathbb{X}}, \end{aligned} \tag{17}$$

where $H_\lambda(x, \lambda, p, t) \triangleq \min_{u \in \mathcal{U}} \max_{d \in \mathcal{D}} \left\langle p, (1-\lambda)\ell(x, u, d, t) + \lambda f(x, u, d, t) \right\rangle$*. Moreover,*

$$V_\lambda(x, \lambda, t) = \begin{cases} V_\ell(x, t) & \text{if } \lambda = 0, \\ V(x, t) & \text{if } \lambda = 1, \end{cases} \tag{18}$$

and V_λ *Lipschitz continuous w.r.t.* λ*.*

We provide the formal proof in the Appendix (Sec. 6) and a demonstration in Fig. 1. Due to the invariance of the λ trajectory with respect to the inputs, the game is independent for different values of λ. By this, we mean that the players cannot affect λ, and hence the optimal strategies are preserved on λ slices. Yet there is a continuity in the game value across λ that will be valuable for simplifying the learning problem.

The purpose of the special augmented game is to host a program in which the linear supervision and PDE losses are not in conflict. The work of Evans and Souganidis (1984b) certifies that V_λ is the solution of the HJ-PDE in (17), and hence may be learned with L_{PDE}. With (18), we propose limiting the supervision loss to the $\lambda = 0$ subspace where $V_\lambda = V_\ell$, while applying the PDE loss to the entire augmented state space to solve a continuous solution across λ. This is given explicitly by the following loss.

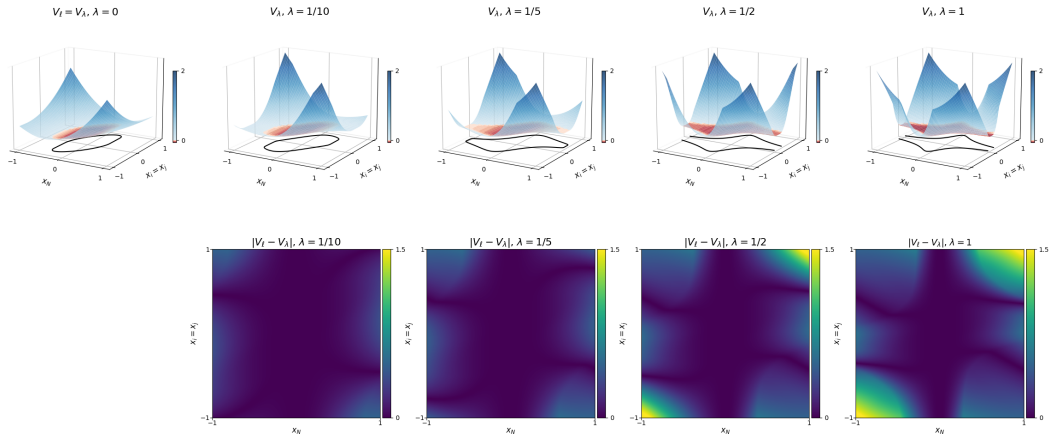

Figure 1: Demonstration of Thm. 2 [V_λ] and Cor. 1 On top, the true value of V_λ at $t = 1$ for the problem posed in (16) with $N = 7$ along the range of λ is given. Note the smooth change from $\lambda = 0$, where $V_\lambda = V_\ell$, to $\lambda = 1$, where $V_\lambda = V$. In the bottom row, the error between V_λ and V_ℓ is plotted as λ increases. Note the gradual increase in error and the large regions of V with low error.

Definition 4 (Linear Semi-Supervision Loss, Nonlinear Spectrum) *Let a loss be*

$$\mathcal{L}_{LSS-NS}(\theta; k) \triangleq \mathbb{E}_{\tilde{\mathbb{X}}_{\lambda=0}, \mathbb{T}_k}[L_{LS}(\theta)] + \mathbb{E}_{\tilde{\mathbb{X}}, \mathbb{T}_k}[L_{PDE}(\theta)] \tag{19}$$

where $\mathbb{T}_k \triangleq [t, t_k]$ *and* t_k *is a monotonically increasing time range with* $t_0 \triangleq t$ *and* $t_K \triangleq t_f$.

Note, in this case, V_θ has an input of one increased dimension to accommodate λ. In some sense, this program adds structure to the problem by incorporating another boundary condition (corresponding to V_ℓ at $\lambda = 0$) that allows us to ultimately better approximate V, which lives on the $\lambda = 1$ subspace of V_λ.

3.4. Generating the Linear Supervisor V_ℓ

Here we introduce two options for constructing the linear supervisor V_ℓ. First, the Hopf formula (7) can generate samples of the linear value, which are then used to train a neural network via supervision, employing the PDE loss L_{PDE} (6) to encourage smooth solutions. Alternatively, one can train the neural network directly with L_{PDE} without the Hopf formula. This approach tends to be slower, but similar in accuracy. The learned model V_ℓ is then fixed and queried across the state space in the following programs to learn V.

4. Demonstration

4.1. N-Dimensional, Differential Game Benchmark

For demonstration, we first introduce a "publisher-subscriber" game, chosen such that the ground truth solution can be directly computed (via dynamic programming) for comparison. Consider a system with a "publisher" state x_0 which unidirectionally influences "subscriber" states x_i, such that

$$\begin{bmatrix} \dot{\mathsf{x}}_0 \\ \dot{\mathsf{x}}_i \end{bmatrix} \triangleq \begin{bmatrix} a & 0 \\ -1 & a \end{bmatrix} \begin{bmatrix} \mathsf{x}_0 \\ \mathsf{x}_i \end{bmatrix} + \begin{bmatrix} 0 \\ b \end{bmatrix} u_i + \begin{bmatrix} 0 \\ c \end{bmatrix} d_i + \begin{bmatrix} \alpha \sin(\mathsf{x}_0)\mathsf{x}_0^2 \\ -\beta \mathsf{x}_0 \mathsf{x}_i^2 \end{bmatrix}, \tag{20}$$

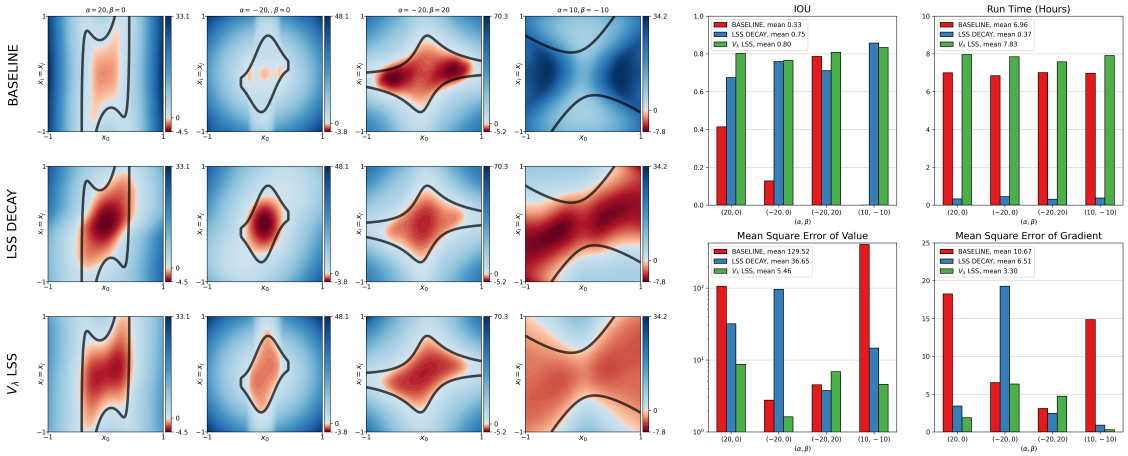

Figure 2: 50-D Benchmark Result Comparison On the left, a slice of the learned solution for four variations of (21) where $(\alpha, \beta) \in \{(20, 0), (-20, 0), (-20, 20), (10, -10)\}$ is shown for each proposed method, and the ground truth zero-level set is overlaid in black. On the right, the IOU, MSEs, and run time are given for each of the variations and methods.

with $u_i \in \{|u_i| \leq 1\}$, $d_i \in \{|d_i| \leq 1\}$ and $a, b, c, \alpha, \beta \in \mathbb{R}$. In this game, let the agent's goal be to attenuate the subscribers while the opponent will seek to amplify them. Hence, let the target set be a ball of radius r with $J_{\mathcal{T},i}(x_0, x_i) \triangleq \frac{1}{2}(x_0^2 + x_i^2 - r^2)$ and let the goals of the agent and the opponent be to minimize and maximize $J_{\mathcal{T},i}$ respectively.

With $N-1$ subscribers, the composed system takes the form

$$\dot{\mathsf{x}} = \left(e_1 e_1^\top - \mathbf{1}_N e_1^\top + aI_N\right)\mathsf{x} + \begin{bmatrix}\mathbf{0}_{N-1}^\top \\ bI_{N-1}\end{bmatrix} u + \begin{bmatrix}\mathbf{0}_{N-1}^\top \\ cI_{N-1}\end{bmatrix} d + \begin{bmatrix}\alpha \sin(\mathsf{x}_0) \\ -\beta \mathsf{x}_0 \mathbf{1}_{N-1}\end{bmatrix} \circ (\mathsf{x} \circ \mathsf{x}), \tag{21}$$

where $\mathsf{x} = [\mathsf{x}_0, ..., \mathsf{x}_{N-1}]^\top \in \mathbb{R}^N$, $u \in \{u \in \mathbb{R}^{N-1} \mid \|u\|_\infty \leq 1\}$, $d \in \{d \in \mathbb{R}^{N-1} \mid \|d\|_\infty \leq 1\}$, and here $\circ$ represents element-wise multiplication. Moreover, the composed target may be written as $J_{\mathcal{T}}(x) = \frac{1}{2}\left((N-1)x_0^2 + \sum x_i^2 - (N-1)r^2\right) = \sum J_{\mathcal{T},i}(P_i x)$ where $P_i : \mathbb{R}^N \to \mathbb{R}^2$ is projection from the composed space to each 2-D publisher-subscriber space.

Remark 1 *The value of this game has the special properties,*

$$V(x,t) = \sum_{}^{N-1} V_i(P_i x, t), \quad \& \quad \mathcal{R}(t) \cap \tilde{\mathcal{X}} = \mathcal{R}_i(t), \tag{22}$$

where $\tilde{\mathcal{X}} \triangleq \{\tilde{x} \in \mathbb{R}^N \mid \forall i, j > 0, \tilde{x}_i = \tilde{x}_j\}$ *is a diagonal. For proof, see the Appendix (Sec. 6).*

Hence, we may solve the value of the N-D game value by summing the values of the $N-1$ decomposed 2-D games, which are solved with dynamic-programming. This is beneficial for a high-dimensional assay as we may naively solve the game in the full system (21) with a given method and score it on the high-fidelity dynamic programming solution.

To interrogate the proposed methods, let $N = 50$ and consider four nonlinear parameter variations of (21). For each, we train a 3-layer V_θ with the baseline program ($\mathcal{L}_{PDE}$, (6)), the decayed linear semi-supervision program ($\mathcal{L}_{LSS-D}$, Def. (2)), and the augmented game V_λ program ($\mathcal{L}_{LSS-NL}$, Def. (4)). See the Appendix in Sec. 6 for training details. We score

them against the dynamic programming solution with the Intersection-Over-Union (IOU) ($|\mathcal{R}_1 \cap \mathcal{R}_2|/|\mathcal{R}_1 \cup \mathcal{R}_2|$) and the Mean-Squared Error (MSE) of both the value and the value gradient. The results are displayed in Fig. 2.

The results demonstrate that both proposed approaches offer significant improvement over the baseline. The augmented program with V_λ achieves the highest mean IOU and lowest mean MSEs with a slight cost in run time over the baseline. Alternatively, the decayed program yields fairly accurate IOU and MSEs with a nearly 20-fold acceleration over the baseline, since no time curriculum is required during training. Both these programs, particularly the latter, underscore the value of the supervision loss in learning the HJ-PDE solution in high dimensions.

4.2. 10D Quadrotor Optimal Control for Collision Avoidance

Let a drone be flying with high-velocity toward an obstacle that it would like to avoid. Consider a 10-D quadrotor dynamic model (Gong and Herbert (2024)) with state defined as $x = [p_x, v_x, \theta, \omega_y, p_y, v_y, \phi, \omega_x, p_z, v_z]^T$. Here, (p_x, p_y, p_z) represent the position, (v_x, v_y, v_z) are the velocities in the world frame, (ϕ, θ) denote the roll and pitch angles, and (ω_x, ω_y) are the corresponding angular rates. The full dynamics can be found in the Appendix (Sec. 6.6). The target set is given by $\mathcal{T} = \{x \mid p_x^2 + p_y^2 \leq 0.5^2\}$, representing a cylindrical obstacle.

The drone dynamics are linearized around the trivial operating point and then used to learn the linear value function V_ℓ with only the PDE loss $\mathcal{L}_{PDE}$. With V_ℓ, we learn the nonlinear solutions through both proposed semi-supervised approaches. We compare these to the baseline method proposed in Bansal and Tomlin (2021), which solely uses $\mathcal{L}_{PDE}$. As tuning ((1)) can be time-consuming, we also introduce an adaptive-weighting scheme (see Appendix, Sec. 6.7) that may work for either approach but is applied here to the decay method. The performance gain is quantified with three metrics: the volume of the corrected safe set through probabilistic conformal expansion (referred to as recovered volume), roll-out false-negatives (FN%) of the solution (without conformal expansion), and roll-out false-positives (FP%). Intuitively, $FP\%$ measures the proportion of overly optimistic classifications while $FN\%$ measures the proportion of overly conservative classifications. Detailed descriptions of these metrics are provided in the Appendix (Sec. 6.8).

The results are summarized in Fig. 3 and Table 1. The proposed variants significantly outperform the baseline model in terms of recovered volume and false positives. All the methods maintain a reasonable false-negative rate, as FN% is less critical from a safety perspective — false negatives correspond to issuing false alarms, whereas false positives can lead to system failures.

Table 1: 10D Quadrotor Results

Method	Recovered Volume	FP%	FN%	Time
Baseline (Bansal and Tomlin (2021))	81.09%	1.864%	0.005%	2.5h
LSS Decay (Def. 2)	94.89%	0.725%	0.049%	0.5h
LSS Decay Adp. (App. 6.7)	94.96%	0.234%	0.326%	0.8h
V_λ LSS (Def. 4)	88.92%	0.908%	0.041%	4.5h

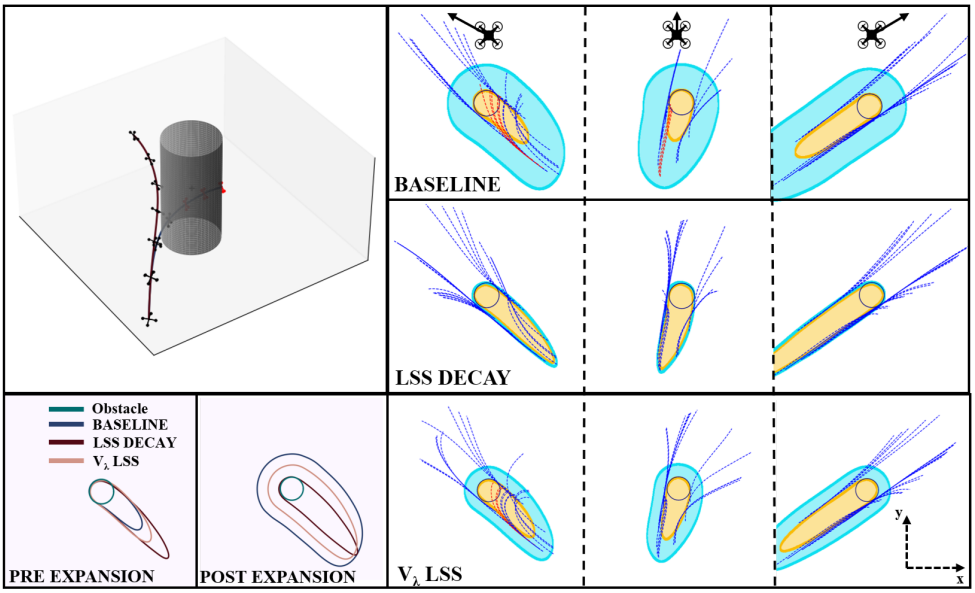

Figure 3: 10-D Quadrotor Result Comparison In the upper-left, the problem in which the drone is flying toward an obstacle is depicted with two trajectories demonstrating success and failure. On the right, slices of the sub-zero level set of the learned value that approximate the unsafe set are shown (gold), along with the 99.9%-confidence conformal expansion of the learned set (teal) and a sample of the roll-outs (blue if safe, else red). In the lower-left, a slice of the learned sets is shown before and after the conformal expansion.

With all learned methods, one may note the higher number of FP% to FN%, corresponding to an underapproximation of the unsafe set. This is likely due to the "concentration of measure" phenomenon of uniformly sampled high-dimensional spaces, which would here bias the learned solution towards safe, positive value, and potentially exacerbated in a solely self-supervised program. Considering the pre-expansion plots in Fig. 3, it seems linear semi-supervision protects against this bias, grounding the program in some sense.

Interestingly, the faster LSS decay methods demonstrate superior performance compared to V_λ LSS. We hypothesize that this is because the true solution is not particularly nonlinear. As a result, the decay schemes perform effectively by "polishing" the linear solution, while the augmented system introduces additional complexity. In scenarios where the nonlinear solution deviates significantly from V_ℓ, however, the augmented variant may be more robust. In addition, the innacuracy may be because V_λ LSS assumes the linear solution to be completely correct for $\lambda = 0$. In reality, as discussed in Sec. 3.4, this linear solution is generated from an optimization process itself and thus can be noisy.

5. Conclusion

In this work, we propose and demonstrate the introduction of the HJ-PDE *linear* solution to the problem of learning high-dimensional, *nonlinear* HJ-PDE solutions for control and differential games. This proves to be a significant insight, offering improvements in speed and accuracy. We believe this to be a valuable step for high-dimensional approaches to autonomy, and plan to extend this to more complicated games and real world applications.

Aknowledgements

This work was supported by ONR grant N00014-24-1-2661 and NSF CAREER (Award # 2240163). We would like to thank Albert Lin, Dylan Hirsch, Feng Yi, Nikhil Shinde, Sander Tonkens, and Yat Tin Chow for helpful discussions related to the paper.

References

Albert Altarovici, Olivier Bokanowski, and Hasnaa Zidani. A general hamilton-jacobi framework for non-linear state-constrained control problems. *ESAIM: Control, Optimisation and Calculus of Variations*, 19(2):337–357, 2013.

Matthias Althoff, Olaf Stursberg, and Martin Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *2008 47th IEEE Conference on Decision and Control*, pages 4042–4048. IEEE, 2008.

S. Bansal, M. Chen, J. F. Fisac, and C. J. Tomlin. Safe Sequential Path Planning of Multi-Vehicle Systems Under Presence of Disturbances and Imperfect Information. *Proc. American Control Conference*, 2017.

Somil Bansal and Claire Tomlin. DeepReach: A deep learning approach to high-dimensional reachability. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

William E Boyce, Richard C DiPrima, and Douglas B Meade. *Elementary differential equations and boundary value problems*. John Wiley & Sons, 2021.

Mo Chen and Claire Tomlin. Exact and Efficient Hamilton-Jacobi Reachability for Decoupled Systems. In *Conference on Decision and Control*, 12 2015.

Mo Chen, Qie Hu, Casey Mackin, Jaime Fisac, and Claire J Tomlin. Safe Platooning of Unmanned Aerial Vehicles via Reachability. In *Conference on Decision and Control*, 2015.

Mo Chen, Sylvia L Herbert, Mahesh S Vashishtha, Somil Bansal, and Claire J Tomlin. Decomposition of reachable sets and tubes for a class of nonlinear systems. *IEEE Transactions on Automatic Control*, 63(11):3675–3688, 2018.

Yat Tin Chow, Jérôme Darbon, Stanley Osher, and Wotao Yin. Algorithm for overcoming the curse of dimensionality for time-dependent non-convex Hamilton–Jacobi equations arising from optimal control and differential games problems. *Journal of Scientific Computing*, 73: 617–643, 2017.

Jérôme Darbon and Stanley Osher. Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere. *Research in the Mathematical Sciences*, 3(1):19, 2016.

Jérôme Darbon, Gabriel P Langlois, and Tingwei Meng. Overcoming the curse of dimensionality for some hamilton–jacobi partial differential equations via neural network architectures. *Research in the Mathematical Sciences*, 7(3):20, 2020.

Badis Djeridane and John Lygeros. Neural approximation of pde solutions: An application to reachability computations. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 3034–3039. IEEE, 2006.

Carlos Esteve-Yagüe, Richard Tsai, and Alex Massucco. Finite-difference least square methods for solving hamilton-jacobi equations using neural networks, 2024.

L. C. Evans and P. E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations. *Indiana Univ. Math. J.*, 33(5):773–797, 1984a.

Lawrence C Evans and Panagiotis E Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-isaacs equations. *Indiana University mathematics journal*, 33(5):773–797, 1984b.

L.C. Evans and Panagiotis E. Souganidis. Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations, 1984c. ISSN 0022-2518.

Jaime F Fisac, Neil F Lugovoy, Vicenç Rubies-Royo, Shromona Ghosh, and Claire J Tomlin. Bridging hamilton-jacobi safety analysis and reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8550–8556. IEEE, 2019.

Milan Ganai, Sicun Gao, and Sylvia Herbert. Hamilton-jacobi reachability in reinforcement learning: A survey. *IEEE Open Journal of Control Systems*, 2024.

Zheng Gong and Sylvia Herbert. Robust control lyapunov-value functions for nonlinear disturbed systems, 2024. URL https://arxiv.org/abs/2403.03455.

Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34): 8505–8510, 2018.

Chong He, Zheng Gong, Mo Chen, and Sylvia Herbert. Efficient and guaranteed hamilton-jacobi reachability via self-contained subsystem decomposition and admissible control sets. *IEEE Control Systems Letters*, 2023.

Haomiao Huang, J Ding, Wei Zhang, and C J Tomlin. A differential game approach to planning in adversarial scenarios: A case study on capture-the-flag. In *International Conference on Robotics and Automation (ICRA)*, pages 1451–1456, 2011.

Hyun Joe Jeong and Andrea Bajcsy. Robots that suggest safe alternatives. *arXiv preprint arXiv:2409.09883*, 2024.

Matthew R. Kirchner, Robert Mar, Gary Hewer, Jerome Darbon, Stanley Osher, and Y. T. Chow. Time-optimal collaborative guidance using the generalized Hopf formula. *IEEE Control Systems Letters*, 2(2):201–206, apr 2018. doi: 10.1109/lcsys.2017.2785357. URL https://doi.org/10.11092Flcsys.2017.2785357.

Niklas Kochdumper and Matthias Althoff. Constrained polynomial zonotopes. *Acta Informatica*, 60(3):279–316, 2023.

Shreyas Kousik, Sean Vaskov, Fan Bu, Matthew Johnson-Roberson, and Ram Vasudevan. Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots. *arXiv*, 2018. URL http://arxiv.org/abs/1809.06746.

Alexander B Kurzhanski. Dynamics and control of trajectory tubes. theory and computation. In *2014 20th International Workshop on Beam Dynamics and Optimization (BDO)*, pages 1–1. IEEE, 2014.

Donggun Lee and Claire J Tomlin. Iterative method using the generalized Hopf formula: Avoiding spatial discretization for computing solutions of Hamilton-Jacobi equations for nonlinear systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1486–1493. IEEE, 2019.

Albert Lin and Somil Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. *arXiv preprint arXiv:2312.08604*, 2023.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022.

Vincent Liu, Chris Manzie, and Peter M Dower. A hamilton-jacobi-bellman approach to ellipsoidal approximations of reachable sets for linear time-varying systems. *arXiv preprint arXiv:2401.06352*, 2024.

A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *Int. J. Robotics Research*, pages 947–982, Jun. 2017. doi: 10.1177/0278364917712421.

I.M. Mitchell, A.M. Bayen, and C.J. Tomlin. A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games. *IEEE Trans. Automatic Control*, 50(7): 947–957, 2005. doi: 10.1109/TAC.2005.851439.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

Will Sharpless, Yat Tin Chow, and Sylvia Herbert. Conservative linear envelopes for high-dimensional, Hamilton-Jacobi reachability for nonlinear systems via the Hopf formula, 2024a.

Will Sharpless, Yat Tin Chow, and Sylvia Herbert. State-augmented linear games with antagonistic error for high-dimensional, nonlinear hamilton-jacobi reachability. *arXiv preprint arXiv:2403.16982*, 2024b.

William Sharpless, Nikhil Shinde, Matthew Kim, Yat Tin Chow, and Sylvia Herbert. Koopman-Hopf Hamilton-Jacobi reachability and control. *arXiv preprint arXiv:2303.11590*, 2023.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Liren Yang, Hang Zhang, Jean-Baptiste Jeannin, and Necmiye Ozay. Efficient backward reachability using the minkowski difference of constrained zonotopes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(11):3969–3980, 2022.

6. Appendix

6.1. Proof of V_ℓ, V Theorem 1

Proof

By Theorem 1 of Sharpless et al. (2024a), it is possible to define a linear game with "antagonistic" error V_{ℓ,δ^*} with the special property

$$V^+_{\ell,\delta^*}(x,t) \triangleq \sup_{\mathfrak{e}\in\mathfrak{E}(t)} \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J_{\mathcal{T}}(\mathsf{x}_{\ell+\varepsilon}(t_f)), \quad \implies \quad V, V_\ell \le V^+_{\ell,\delta^*},$$

when the strategies $\mathfrak{e} \in \mathfrak{E}$ are defined s.t. $\|\mathfrak{e}[u](\tau)\| \le \delta^*$ (see Sharpless et al. (2024a) for details). We may also define a game where the error assists the player, which would be unsafe but by the proof of Theorem 1 of Sharpless et al. (2024a) offers a lower bound s.t.

$$\begin{aligned} V^-_{\ell,\delta^*}(x,t) \triangleq \inf_{\mathfrak{e}\in\mathfrak{E}(t)} \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J_{\mathcal{T}}(\mathsf{x}_{\ell+\varepsilon}(t_f)) &\implies V, V_\ell \ge V^-_{\ell,\delta^*}, \\ &\implies |V - V_\ell| \le (V^+_{\ell,\delta^*} - V^-_{\ell,\delta^*}). \end{aligned}$$

Finally, since V_{ℓ,δ^*} and V^-_{ℓ,δ^*} are games with linear dynamics and the hamiltonians in (9), they may be solved with the Hopf formula, yielding the relation in (1).

■

6.2. Proof of Corollary 1

Proof Let ℓ be defined as in (11), then by Taylor's theorem, $\delta^* = \max_{\mathcal{S}\times\mathcal{U}\times\mathcal{D}\times\mathbb{T}} \|[f - \ell](x,u,d,t)\| \sim O(\|m - m_0\|^2)$. By Thm. 1, it must be that $V_\ell(x,t) \in [0, \epsilon^*]$ for any x in a local region $\mathcal{S}$ (where ϵ^* is a function of the maximum error δ^* on $\mathcal{S}$), and thus there must exist an $\epsilon \in \mathbb{R}$ s.t. $V - V_\ell = \epsilon$. Considering this, one may note for any $\tau \in \mathbb{T}$

$$\begin{aligned} \|\mathsf{x}(\tau) - \mathsf{x}_\ell(\tau)\| &= \left\| x + \int_t^s f(\mathsf{x}(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s))\, ds - \left(x + \int_t^s \ell(\mathsf{x}_\ell(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s))\, ds \right) \right\| \\ &= \left\| \int_t^s f(\mathsf{x}(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s)) - f(\mathsf{x}_\ell(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s)) + O(\|m - m_0\|^2)\, ds \right\| \\ &\le L_\ell \int_t^s \|P\tilde{\mathsf{x}}(\tau) - \mathsf{x}_\ell(\tau)\|\, ds \end{aligned}$$

and thus,

$$\begin{aligned} \lim_{m\to m_0} \|\mathsf{x}(\tau) - \mathsf{x}_\ell(\tau)\| &= \left\| \int_t^s f(\mathsf{x}(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s)) - f(\mathsf{x}_\ell(s), \mathsf{u}(s), \mathfrak{d}[\mathsf{u}](s))\, ds \right\| \\ &\le L_f \int_t^s \|\mathsf{x}(\tau) - \mathsf{x}_\ell(\tau)\|\, ds. \end{aligned}$$

This relation implies (Boyce et al. (2021)) that $\lim_{m\to m_0} \|\mathsf{x}(\tau)-\mathsf{x}_\ell(\tau)\| = 0 \implies \lim_{m\to m_0} \mathsf{x}_\ell(\tau) = \mathsf{x}(\tau)$. It then follows,

$$\lim_{m\to m_0} V_\ell(x,t) = \lim_{m\to m_0} \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J(\mathsf{x}_\ell(t_f)) = \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J(\mathsf{x}(t_f)) = V(x,t).$$

■

6.3. Proof of Nonlinear Spectrum Augmetation Theorem 2

Proof The proof of the λ boundary conditions follows from the fact that for initial condition (x,λ,t) at any time $\tau\in[t,t_f]$,

$$P\tilde{\mathsf{x}}(\tau;x,\lambda,t,u(\cdot),\mathfrak{d}[u](\cdot)) = \begin{cases} \mathsf{x}_\ell(\tau;x,t,u(\cdot),\mathfrak{d}[u](\cdot)) & \text{if } \lambda=0, \\ \mathsf{x}(\tau;x,t,u(\cdot),\mathfrak{d}[u](\cdot)) & \text{if } \lambda=1. \end{cases} \tag{23}$$

The proof of this fact for each case is analogous so we give that of $\lambda=0$. Consider

$$\begin{aligned} \|P\tilde{\mathsf{x}}(\tau)-\mathsf{x}_\ell(\tau)\| &= \left\| x + \int_t^s P\tilde{f}_\lambda(P\tilde{\mathsf{x}}(s),\mathsf{u}(s),\mathfrak{d}[\mathsf{u}](s))\,ds - \left(x+\int_t^s \ell(\mathsf{x}_\ell(s),\mathsf{u}(s),\mathfrak{d}[\mathsf{u}](s))\,ds\right)\right\| \\ &= \left\| \int_t^s \ell(P\tilde{\mathsf{x}}(s),\mathsf{u}(s),\mathfrak{d}[\mathsf{u}](s)) - \ell(\mathsf{x}_\ell(s),\mathsf{u}(s),\mathfrak{d}[\mathsf{u}](s))\,ds\right\| \\ &\le L_\ell \int_t^s \|P\tilde{\mathsf{x}}(\tau)-\mathsf{x}_\ell(\tau)\|\,ds \end{aligned}$$

where L_ℓ is a Lipschitz constant. This relation implies that $\|P\tilde{\mathsf{x}}(\tau)-\mathsf{x}_\ell(\tau)\| = 0$; see a standard trajectory uniqueness proof for further details (Boyce et al. (2021)). Note, the invariance of λ sufficed to fix the projection of $P\tilde{f}_\lambda$. With (23), the value defined in (16) is then trivially,

$$V_\lambda(x,\lambda,t) = \begin{cases} \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J_{\mathcal{T}}(\mathsf{x}_\ell(t_f)) & \text{if } \lambda=0, \\ \sup_{\mathfrak{d}\in\mathfrak{D}(t)} \inf_{\mathsf{u}\in\mathbb{U}(t)} J_{\mathcal{T}}(\mathsf{x}(t_f)) & \text{if } \lambda=1, \end{cases} \tag{24}$$

giving the boundary properites in (18). Given the assumptions on the original game and that $\tilde{f}_\lambda$ is Lipschitz continuous w.r.t. λ, it must be that V_λ is the solution of the HJ-PDE given in (17) and Lipschitz continuous w.r.t. λ (Thm 2., Evans and Souganidis (1984b)). ■

6.4. Proof of Benchmark Property 1

Proof Proof. Note the unidirectional influence ensures trajectories are "decomposable" Chen and Tomlin (2015) s.t.

$$P_i\mathsf{x}_{x,t}^{u(\cdot),d(\cdot)}(\tau) = \mathsf{x}_{i,P_ix,t}^{u_i(\cdot),d_i(\cdot)}(\tau), \quad \tau\in[t,t_f]$$

where x_i is a trajectory of (20) and x is a trajectory of (21). It follows that

$$V(x,t) = \sup_{\mathfrak{d}} \inf_{u(\cdot)} J_{\mathcal{T}}(\mathsf{x}_{x,t}^{u(\cdot),\mathfrak{d}}(t_f)) = \sup_{\{\mathfrak{d}_i\}} \inf_{\{u_i(\cdot)\}} \sum^{N-1} J_{\mathcal{T}i}(\mathsf{x}_{i,P_ix,t}^{u_i(\cdot),\mathfrak{d}_i}(t_f)) = \sum^{N-1} V_i(P_ix,t).$$

It immediately follows that for $\tilde{x} \in \tilde{\mathcal{X}}$ and $i > 0$,

$$V(\tilde{x}, t) = \sum^{N-1} V_i(P_i\tilde{x}, t) = (N-1)V_i(P_i\tilde{x}, t) \implies \Big(V(\tilde{x}, t) = 0 \iff V_i(P_i\tilde{x}, t) = 0\Big).$$

■

6.5. Training Details

In this work, we introduced two new training programs as augmentations of the well-known existing approach proposed in Bansal and Tomlin (2021), where the DeepReach software was proposed. For this reason, we adopted the parameters found in that work and our fork of the existing DeepReach software may be found here here. As such, all neural nets used in this work consist of 3 layers with 512 neurons and sinsoidal activations.

For the benchmark problem, we used the same training parameters for all variations of the system. For the baseline and the V_λ programs, we found that experiments with 300k iterations, a batch size of 60k and a learning rate of 5e-6 performed best on the 50-D case with our compute budget. For the decay program, we found that the experiments with 10k iterations, a batch size of 10k and a learning rate 1e-6 achieved the given results. For both LSS programs, the linear supervisors generated with or without Hopf data ultimately yielded equivalent accuracy but required five and ten minutes respectively; note, this run time is added to the final run time in all results.

In the quadrotor problem, we used a learning rate of 1e-5 for all variations of the system. For the baseline and the V_λ programs, we found that experiments with 100k iterations, and a batch size of 65k performed best. For both the linear and adaptive decay program, we used a batch size of 10k and 60k iterations for training. We further conducted a coarse parameter search for the LSS Decay program, and found that the experiments with $\lambda_K = 0.6$, $\rho = 0.1$, $\rho_g = 0.2$ achieved the given results. For the LSS Decay Adp. method, we set $\mathcal{I}_{start} = 10$ and $\mathcal{I}_{end} = 1$.

6.6. 10-D Quadrotor Dynamics

The quadrotor dynamics is given by

$$\begin{aligned}
\dot{p_x} &= v_x \\
\dot{p_y} &= v_y \\
\dot{p_z} &= v_z \\
\dot{\phi} &= -d_1\phi + \omega_x \\
\dot{\theta} &= -d_1\theta + \omega_y \\
\dot{v_x} &= g\ tan(\theta) \\
\dot{v_y} &= g\ tan(\phi) \\
\dot{v_z} &= u_3 \\
\dot{\omega_x} &= -d_0\phi + n_0u_1 \\
\dot{\omega_y} &= -d_0\theta + n_0u_2,
\end{aligned} \tag{25}$$

where $(u_1, u_2, u_3) \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]^2 \times [-1, 1]$ is the control input and $(d_0, d_1, n_0) = (7, 4, 12)$ are constants. The state space $\mathbb{X}$ is chosen as $[-4, 4]^2 \times [-2, 2] \times [-1.5, 1.5]^2 \times [-3, 3]^2 \times [-2, 2] \times [-6, 6]^2$.

6.7. Adaptive Weighting Scheme

The overall idea of the adaptive weighting scheme is to gradually decrease the contribution of the supervision loss terms during training. To achieve this, we propose the following algorithm:

Algorithm 1 Adaptive Weighting training

Require: N_k, $\mathcal{I}_{end}$, $\mathcal{I}_{start}$
Ensure: $\mathcal{I}_{end} < \mathcal{I}_{start}$
  for $k = 0, 1, \ldots, K$ **do**
    Compute importance of supervision loss terms $\mathcal{I}_k = \mathcal{I}_{end} \cdot exp\left\{ln(\frac{\mathcal{I}_{start}}{\mathcal{I}_{end}}) * (1 - \frac{k}{K})\right\}$
    Compute loss terms $L_{LS}(\theta)$ and $\mathcal{L}_{PDE}(\theta)$
    Compute the relative weight $\lambda_k = 0.9\lambda_k + 0.1\mathcal{I}_k \frac{||\nabla_\lambda L_{LS}(\theta)||}{||\nabla_\lambda \mathcal{L}_{PDE}(\theta)||}$
    Compute overall loss $\mathcal{L}_{LSS-A}(\theta; k) \triangleq \lambda_k \mathbb{E}_{\mathbb{X},\mathbb{T}}[L_{LS}(\theta)] + \mathbb{E}_{\mathbb{X},\mathbb{T}}[L_{PDE}(\theta)]$
    Step the optimizer
  end

Empirically, choosing $\mathcal{I}_{start} = 10.0$ and $\mathcal{I}_{end} = 1.0$ works well.

6.8. Metrics

6.8.1. Probabilistic expansion

We leverage method proposed in Lin and Bansal (2023) to generate a probabilistic expansion from the learned value function using conformal prediction. To do this, we determine with high-confidence δ, the maximum value for the empirically derived unsafe set under the learned policy, and then solve the corrected unsafe set $\mathcal{R}_\delta = \{x : x \in \mathcal{X}, V_\theta(x, t) < \delta\}$. Then with $1 - 10^{-16}$ confidence, we can assert that 99.9% of the states within the compliment $\mathcal{R}_\delta^C$ are safe under the learned policy. Finally, let the sampled volume of the recovered safe set be

$$\%v_{\mathcal{S}} = \mathbb{P}_{x \sim Uniform(\mathbb{X})}(x \in \mathcal{R}_\delta^C). \tag{26}$$

Note, the higher the volume the better the quality of learned solution.

6.8.2. False positive rate & False negative rate

The false positive rate represents the likelihood of a random state $x \in \mathbb{X}$ being predicted as safe when it is unsafe under the learned policy, given by:

$$FP\% = \mathbb{P}_{x \sim Uniform(\mathbb{X})}(x \in \mathcal{S} \ \& \ J_{\mathcal{T}}(\mathsf{x}(t_f)) < 0). \tag{27}$$

Similarly, the false negative rate is defined as the likelihood of a random state being classified as unsafe when it is indeed safe:

$$FN\% = \mathbb{P}_{x \sim Uniform(\mathbb{X})}(x \in \mathcal{S}^C \ \& \ J_{\mathcal{T}}(\mathsf{x}(t_f)) \geq 0). \tag{28}$$

Intuitively, $FP\%$ indicates the proportion of overly optimistic classifications while $FN\%$ represents the proportion of overly conservative classifications.