# GUIDING REACHABILITY LEARNING WITH PONTRYAGIN'S MAXIMUM PRINCIPLE

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hamilton–Jacobi (HJ) reachability provides a framework for analyzing the safety of autonomous systems under worst-case disturbances. Specifically, its value function identifies states from which failure is unavoidable, while its spatial gradient determines how a controller should act to preserve safety. Computing this function with traditional grid-based solvers is prohibitively expensive in high dimensions. Neural methods such as DeepReach improve scalability, but training on the HJ partial differential equation residual alone provides limited direct supervision for the gradient, while existing methods that generate additional value labels can be costly. We propose a sample-efficient method that supplements residual training with value and gradient labels generated using Pontryagin's Maximum Principle (PMP). At intervals aligned with the training curriculum, we initialize state–costate pairs from the network's current level sets and integrate the PMP characteristic equations over a short horizon to generate supervision beyond the curriculum frontier. At the same time, since the network provides the initial conditions, generating these labels avoids solving the expensive two-point boundary-value problems associated with PMP, thereby enabling efficient training. In experiments with a 13-dimensional quadrotor and an eight-dimensional racing vehicle, the method reduces training time and improves the accuracy of the learned safety value function in challenging settings. We further deploy the learned value function as a safety filter on a full-scale electric vehicle driven near the friction limit and against an adversarial human driver instructed to leave the track.

## 1 INTRODUCTION

Safety-critical autonomous systems must operate under external disturbances, modeling errors, and the behavior of other agents that cannot be reliably predicted. Near the physical limits of the platform—a vehicle at the friction limit, an aerial vehicle in gusts—even small deviations from a nominal plan can become unrecoverable. In these settings it is not enough to optimize a nominal trajectory. One would like to know, before acting, which states admit a safety-preserving strategy, and to have a feedback law that exploits that knowledge online to maintain safety.

Hamilton–Jacobi (HJ) reachability provides a principled framework for this problem. It formulates safety as a zero-sum differential game between a controller seeking to avoid failure and a bounded disturbance acting against it (Evans & Souganidis, 1984). The resulting value function is obtained by solving a Hamilton–Jacobi–Isaacs partial differential equation (PDE) and serves two complementary roles: its sign identifies states from which failure cannot be avoided despite the controller's best efforts, while its spatial gradient determines the control action that best preserves safety Bansal et al. (2017). Together, these quantities allow a reachability-based safety filter to decide when to intervene and how to modify the nominal control (Borquez et al., 2024). Learning both accurately is therefore essential: errors in the value can misclassify unsafe states as safe, while errors in the gradient can lead to ineffective interventions even when the need to act is correctly identified.

The main obstacle is computational. Classical solvers discretize the state space to solve the PDE, with computation and memory requirements growing exponentially in the number of state variables, typically limiting direct grid-based computation to five or six dimensions. Neural methods such as DeepReach (Bansal & Tomlin, 2021) improve scalability by representing the value function with a neural network trained to satisfy the governing HJI equation at sampled states. Subsequent work



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

enforces the terminal condition exactly through the network's construction (Singh et al., 2025). However, minimizing the equation residual provides only indirect supervision for the spatial gradient: it constrains how the gradient interacts with the system dynamics without directly determining all of its components. Even theoretical guarantees of value-function convergence under suitable assumptions do not, by themselves, ensure convergence of the gradients used for control (Hofgard, 2024). This motivates additional supervision that improves both the learned value and its gradient. Reinforcement-learning formulations offer another approach by learning control policies directly, but face the challenges of adversarial policy optimization (Fisac et al., 2019; Hsu et al., 2023).

Recent work has therefore incorporated additional optimal-control information into neural HJ learning. Nakamura-Zimmerer et al. (2021) solve two-point boundary-value problems for finite-horizon HJB equations to generate supervision for both the value and its spatial gradient. They use the partially trained network to warm-start these solves and select new initial conditions where validation error is large. However, solving two-point boundary problems can be expensive. Moreover, the generated supervision is not combined with the HJ residual so the learned solution is only locally optimal. Feng et al. (2025) and MADR (Teoh et al., 2025) generate approximate value labels using sampling-based MPC and refresh them as training progresses to longer time horizons, as does the reach-avoid method of Thorup et al. (2026). Other approaches use trajectories to guide where the governing equation is enforced. However, sampling-based MPC is computationally expensive, especially for high-dimensional systems, leading to longer training times. Dallas et al. (2026) generate boundary trajectories using Pontryagin's Maximum Principle (PMP) (Pontryagin, 1987) before training and sample along them when learning a control-barrier-value function. Park & Tu (2026) use the current value function to guide forward rollouts and adapt the residual sampling distribution without introducing additional labels. Finally, Feng et al. (2026) avoid spatial gradients altogether through a discrete-time Bellman–Isaacs recursion.

Our approach combines direct supervision of both the value and its gradient with HJ PDE residual training. The key observation is that PMP describes how value and gradient information propagate along particular trajectories, known as characteristics. Starting with estimates of these quantities at one point, we can integrate the PMP equations to obtain estimates at additional points. We use the partially trained network to provide this initial information and generate supervision just beyond the time horizon covered by the current training curriculum. The resulting trajectories provide approximate value and gradient labels at longer time horizons, which supplement the standard HJI residual during training. This construction makes label generation efficient while significantly improving the quality of the learned safety value function and the corresponding safety policy. Repeating this procedure as the curriculum advances refreshes the supervision using the network's evolving solution.

To summarize, our key contributions are: (1) we propose a method that supplements HJ residual training with value and gradient labels generated using Pontryagin's Maximum Principle for improving the quality of the learned safety value function and safety policy; (2) we demonstrate that the proposed approach is highly sample-efficient and achieves competitive or improved accuracy compared to state-of-the-art baselines with substantially lower total training time; and (3) we deploy the learned value function as a safety filter on a full-scale electric vehicle and demonstrate its efficacy in maintaining track containment during autonomous driving near the friction limit and under a human driver instructed to leave the track.

## 2 Problem Formulation

Consider a continuous-time nonlinear dynamical system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}), \quad \boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \boldsymbol{u} \in \mathcal{U} \subset \mathbb{R}^m, \quad \boldsymbol{d} \in \mathcal{D} \subset \mathbb{R}^k \tag{1}$$

over a finite horizon $T < \infty$, where $\boldsymbol{x}$ is the state, $\boldsymbol{u}$ is the control, and $\boldsymbol{d}$ is a bounded adversarial disturbance. $\boldsymbol{d}$ might represent model uncertainty or an exogenous uncertainty that we want to safeguard against. Throughout, we assume that $\mathcal{U}$ and $\mathcal{D}$ are compact sets. The vector field $f : \mathcal{X} \times \mathcal{U} \times \mathcal{D} \to \mathbb{R}^n$ is both continuous and continuously differentiable in its first argument. We are given a Lipschitz continuous function $\ell : \mathcal{X} \to \mathbb{R}$ whose zero sub-level set encodes the *failure set* $\mathcal{L}$,

$$\mathcal{L} = \{\boldsymbol{x} \in \mathcal{X} : \ell(\boldsymbol{x}) \leq 0\}. \tag{2}$$

**Objective.** Our goal is to determine the set of states from which failure for the system in equation 1 can be avoided despite the worst-case disturbance, and to obtain a feedback policy that preserves safety.

### 2.1 HAMILTON-JACOBI REACHABILITY

Let $\boldsymbol{\xi}_{\boldsymbol{x},t}^{\boldsymbol{u},\boldsymbol{d}}(s)$ be the state at time $s \in [t, T]$ of a trajectory that starts at $\boldsymbol{x}$ at time $t$ under $\boldsymbol{u}$ and $\boldsymbol{d}$, such that $\boldsymbol{\xi}_{\boldsymbol{x},t}^{\boldsymbol{u},\boldsymbol{d}}(t) = \boldsymbol{x}$. The *backward reachable tube* (BRT) for avoidance tasks collects the states from which a failure over a remaining horizon is unavoidable,

$$\mathcal{B}(t) = \left\{ \boldsymbol{x} \in \mathcal{X} : \forall \boldsymbol{u}, \exists \boldsymbol{d} \in \mathcal{D}, \exists s \in [t, T], \text{ s.t. } \boldsymbol{\xi}_{\boldsymbol{x},t}^{\boldsymbol{u},\boldsymbol{d}}(s) \in \mathcal{L} \right\}. \tag{3}$$

It is characterized by the zero sub-level set of the value function of a zero-sum differential game,

$$V(\boldsymbol{x}, t) = \sup_{\boldsymbol{u} \in \mathcal{U}} \inf_{\boldsymbol{d} \in \mathcal{D}} \min_{s \in [t,T]} \ell \left( \boldsymbol{\xi}_{\boldsymbol{x},t}^{\boldsymbol{u},\boldsymbol{d}}(s) \right), \tag{4}$$

where the controller tries to maximize and the disturbance tries to minimize the smallest value $\ell(\boldsymbol{\xi}(s))$ over the remaining horizon. Hence, the BRT we would like to find is the tube $\mathcal{B}(t) = \{\boldsymbol{x} \in \mathcal{X} : V(\boldsymbol{x}, t) \leq 0\}$. The value function equation 4 satisfies the Hamilton-Jacobi-Isaacs variational inequality Bansal et al. (2017)

$$\min \left\{ \frac{\partial V(\boldsymbol{x}, t)}{\partial t} + H\left(\boldsymbol{x}, \nabla_{\boldsymbol{x}} V(\boldsymbol{x}, t)\right), \ell(\boldsymbol{x}) - V(\boldsymbol{x}, t) \right\} = 0, \tag{5}$$

with Hamiltonian

$$H(\boldsymbol{x}, \boldsymbol{p}) = \max_{\boldsymbol{u} \in \mathcal{U}} \min_{\boldsymbol{d} \in \mathcal{D}} \langle \boldsymbol{p}, f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \rangle \tag{6}$$

where $\boldsymbol{p} \in \mathbb{R}^n$ and $V(\boldsymbol{x}, T) = l(\boldsymbol{x})$. The optimal input and disturbance are $\boldsymbol{u}^\star(\boldsymbol{x}, \boldsymbol{p}) = \arg\max_{\boldsymbol{u} \in \mathcal{U}} \min_{\boldsymbol{d} \in \mathcal{D}} \langle \boldsymbol{p}, f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d}) \rangle$ and $\boldsymbol{d}^\star(\boldsymbol{x}, \boldsymbol{p}) = \arg\min_{\boldsymbol{d} \in \mathcal{D}} \langle \boldsymbol{p}, f(\boldsymbol{x}, \boldsymbol{u}^\star, \boldsymbol{d}) \rangle$. The optimal controller $\boldsymbol{u}^\star$ preserves system safety as long as the system starts outside the BRT.

### 2.2 NEURAL HAMILTON JACOBI REACHABILITY

To scale HJ reachability to high-dimensional systems, the value function in equation 4 can be approximated with a neural network $V_\theta$, as in (Bansal & Tomlin, 2021; Singh et al., 2025). In particular, the terminal condition can be enforced exactly through the parametrization:

$$V_\theta(\boldsymbol{x}, \tau) = \ell(\boldsymbol{x}) + \tau \cdot f_\theta(\boldsymbol{x}, \tau), \tag{7}$$

where $\tau = T - t$ denotes the time-to-go, so that $V_\theta(\cdot, T) \equiv \ell$ by construction. Neural HJ reachability methods typically train $V_\theta$ by minimizing the residual of the HJI-VI at sampled collocation points (Bansal & Tomlin, 2021; Singh et al., 2025). Starting from the terminal condition at $\tau = 0$, training can be carried out with a curriculum that progressively expands the time horizon.

## 3 METHOD

Our key idea is to use the partially learned value function to generate additional supervision for neural HJ reachability during training. At selected points in the time curriculum, we identify states on the level sets of the current value function and use Pontryagin's Maximum Principle (PMP) (Pontryagin, 1987) to propagate information from these states further along the time horizon. This procedure produces additional labels for both the value function and its spatial gradient, which are used alongside the standard HJI-VI residual during training.

The main advantage is that the current neural value function provides the information needed to initialize the PMP computation directly. This avoids the expensive boundary-value problems that arise in a conventional application of PMP and allows new supervision to be generated efficiently throughout training. We first explain the connection between HJ reachability and PMP, and then describe how we use this connection to guide neural reachability learning.

### 3.1 PONTRYAGIN'S MAXIMUM PRINCIPLE

Our goal is to generate additional supervision for both the value function $V$ in equation 4 and its spatial gradient $\nabla_{\boldsymbol{x}} V$. To do so, we exploit a classical property of Hamilton–Jacobi equations: rather than reasoning about the PDE everywhere in the state space at once, one can follow particular curves in state–time along which the evolution of the solution is described by ordinary differential equations. These curves are known as *characteristics*.

This is useful for our setting because a characteristic does not only describe how the state evolves. It also describes how the spatial gradient of the value function evolves along that state trajectory. Thus, if the value and its gradient are known at one point, the characteristic equations provide a way to propagate this information to other points in state–time.

For the reachability problem in Section 2.1, these characteristics arise whenever the obstacle constraint in the HJI-VI is inactive. In this region, i.e., when $V(\boldsymbol{x}, t) < \ell(\boldsymbol{x})$, the variational inequality reduces to the Hamilton–Jacobi equation

$$\frac{\partial V(\boldsymbol{x}, t)}{\partial t} + H\left(\boldsymbol{x}, \nabla_{\boldsymbol{x}} V(\boldsymbol{x}, t)\right) = 0. \tag{8}$$

To describe a characteristic, we track the state $\boldsymbol{x}(s)$ together with the spatial gradient of the value function along the trajectory. At points where $V$ is differentiable, we denote this gradient by

$$\boldsymbol{p}(s) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}(s), s). \tag{9}$$

The characteristic equations describe how $\boldsymbol{x}(s)$ and $\boldsymbol{p}(s)$ evolve together:

$$\dot{\boldsymbol{x}}(s) = \frac{\partial H(\boldsymbol{x}(s), \boldsymbol{p}(s))}{\partial \boldsymbol{p}} = f\left(\boldsymbol{x}(s), \boldsymbol{u}^\star(s), \boldsymbol{d}^\star(s)\right), \tag{10a}$$

$$\dot{\boldsymbol{p}}(s) = -\frac{\partial H(\boldsymbol{x}(s), \boldsymbol{p}(s))}{\partial \boldsymbol{x}} = -\left[\frac{\partial f}{\partial \boldsymbol{x}}\left(\boldsymbol{x}(s), \boldsymbol{u}^\star(s), \boldsymbol{d}^\star(s)\right)\right]^\top \boldsymbol{p}(s) \tag{10b}$$

These characteristic equations coincide with the state and adjoint equations of Pontryagin's Maximum Principle (PMP) (Pontryagin, 1987). In PMP terminology, $\boldsymbol{p}$ is called the *costate*. Readers familiar with constrained optimization may view the costate as analogous to a dual variable associated with the system dynamics. In our setting, the key interpretation is $\boldsymbol{p}(s) = \nabla_{\boldsymbol{x}} V(\boldsymbol{x}(s), s)$: propagating the costate therefore propagates the spatial-gradient information that we seek to supervise.

The remaining question is how to initialize these equations. In a conventional application of PMP, the state is specified at the initial time, while a condition on the costate is specified at the terminal time. This leads to a two-point boundary-value problem, which generally requires a shooting method to find a trajectory satisfying both boundary conditions.

Neural HJ reachability (Sec.2.2) provides a different way to initialize the same equations. At a curriculum frontier $\tau_c$, the current neural value function provides both its value and spatial gradient at a state $\boldsymbol{x}_c$: $V_\theta(\boldsymbol{x}_c, \tau_c)$ and $\boldsymbol{p}_c = \nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}_c, \tau_c)$. Thus, both the state and costate are available at the same point in state–time. This turns the PMP propagation into an initial-value problem, which can be integrated directly without solving a shooting problem. This gives us a way to propagate gradient information, but we still need to determine the value associated with the propagated states. The following Lemma 1 shows that along a valid characteristic, this value remains constant.

**Lemma 1** (Value is constant)**.** *Suppose that the value function $V$ is differentiable at $(\boldsymbol{x}(s), s)$, that $V(\boldsymbol{x}(s), s) < \ell(\boldsymbol{x}(s))$, and that the characteristic attains the optimum in equation 4. Then the value along a characteristic is constant*

$$\frac{d}{ds} V(\boldsymbol{x}(s), s) = 0. \tag{11}$$

To determine if a label is valid and has not crossed the obstacle, we utilize the Proposition 2 below.

**Proposition 2** (Obstacle domination). *Let the value function $V$ be as in equation 4. For all $\boldsymbol{x} \in \mathcal{X}$ and $t \in [0, T]$, it holds that*

$$V(\boldsymbol{x}, t) \leq \ell(\boldsymbol{x}), \tag{12}$$

*regardless of whether the outer optimization is $\inf_{\boldsymbol{u} \in \mathcal{U}}$ or $\sup_{\boldsymbol{u} \in \mathcal{U}}$.*

Both proofs for Lemma 1 and Proposition 2 can be found in the Appendix A.5, A.6.

### 3.2 Guiding Reachability Learning with PMP

The previous subsection 3.1 shows that, given a state with a known value and spatial gradient, PMP can propagate both quantities to additional points in state–time. We now describe how we obtain such initialization points form the partially trained neural value function and use the resulting PMP labels to guide reachability learning.

At selected milestones of the time curriculum, let $\tau_c$ denote the current curriculum frontier. We refer to a state used to initialize PMP propagation as *seed*. Each seed is associated with a target value $q_i$ and, once the seed has been located on the corresponding level set of $V_\theta$, an initial costate

$$\boldsymbol{p}_i = \nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}_i, \tau_c). \tag{13}$$

Starting from these seeds, PMP generates additional state–time points together with value and gradient labels using equation 10. These labels are stored in a replay buffer

$$\mathcal{E} = \{(\boldsymbol{x}_i, \tau_i, \boldsymbol{p}_i, V_i^c)\}_i, \tag{14}$$

and are used as additional supervision during training.

**Selecting informative level sets.** The zero level set $V_\theta(\boldsymbol{x}, \tau_c) = 0$ represents the current estimate of the BRT boundary and is therefore a natural location at which to generate supervision. However, the spatial gradient is also used in a neighborhood of the boundary. We therefore generate $\rho N$ seeds on the zero level set and distribute the remaining $(1-\rho)N$ seeds across nearby level sets $q_i \in [q_-, q_+]$. Here, $N$ is the total number of seeds. This scheme provides gradient supervision over a band of relevant values rather than only a single surface.

**Projecting states onto level sets.** One challenge is how to efficiently sample states within a given level $q_i$, since a randomly sampled state will generally not satisfy $V_\theta(\boldsymbol{x}, \tau_c) = q_i$. We therefore move each candidate state sample toward the desired level set. Starting from $\boldsymbol{x}^{(0)}$, we apply a Newton Projection algorithm to converge to the desired level. We retain candidates that successfully reach their target level set. We also discard states for which $\ell$ Proposition 2 implies that such a state cannot have the value $q_i$. To avoid running into non-smooth areas, where the gradient can explode, we require the norm of its gradient to be bounded, $\gamma_{\min} < \|\nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}, \tau_c)\| < \gamma_{\max}$.

**Compounding error.** Since the projected seeds are obtained from the partially trained network itself, an incorrect value or gradient estimate could produce incorrect PMP trajectory and reinforce the network's own error. We therefore use a consistency check before using a seed for supervision. Starting from a projected seed, we roll out the control and disturbance induced by the current value function and compute the smallest failure margin encountered along the trajectory, $\zeta(\boldsymbol{x}_i) = \min_k \ell(\hat{\boldsymbol{x}}_k)$. We retain seeds only when this observed margin is consistent with its assigned value $q_i$, i.e., $|\zeta(\boldsymbol{x}_i) - q_i| < \varepsilon_w$.

**Integration.** For every accepted seed, we initialize the PMP equations with $\boldsymbol{x}(\tau_c) = \boldsymbol{x}_i$ and $\boldsymbol{p}(\tau_c) = \nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}_i, \tau_c)$ and propagate equation 10 forward in time-to-go $\tau$ over the window $(\tau_c, \tau_c + h]$ with $h = \min(h_{\max}, T - \tau_c)$. Each step $k$ of the integration yields a label $(\boldsymbol{x}_k, \tau_k, \boldsymbol{p}_k, V_i^c)$ with $\tau_k = \tau_c + k\Delta\tau$. The costate $\boldsymbol{p}_k$ is the gradient label and, by Lemma 1, the seed's level $V_i^c = q_i$ is the value label. Since the equations 10 have no notion of the obstacle, a segment can pass through $\ell < q_i$ and come back, which would carry an invalid label. By Proposition 2, we discard every pair from the first time instant with $\ell(\boldsymbol{x}_k) < q_i - \varepsilon_\ell$ and add a random subset of size $k$ of the remaining labels to the replay buffer $\mathcal{E}$.

**Optimization.** At every step we minimize the residual of equation 5 on collocation points $(\boldsymbol{x}_j, \tau_j)$ with $\tau_j \sim \mathcal{U}[0, \tau_c]$,

$$\mathcal{L}_{\text{PDE}}(\theta) = \sum_{j}^{N_{\text{PDE}}} \left| \max\left( \frac{\partial}{\partial \tau_j} V_\theta(\boldsymbol{x}_j, \tau_j) - H(\boldsymbol{x}_j, \nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}_j, \tau_j)), V_\theta(\boldsymbol{x}_j, \tau_j) - \ell(\boldsymbol{x}_j) \right) \right|. \tag{15}$$

As elaborated in Section 1, the PDE loss leaves $n-1$ directions unsupervised, so driving it to zero does not imply convergence of the gradient (Hofgard, 2024). We therefore supervise the value and its gradient on labels drawn from $\mathcal{E}$, in the manner of Sobolev training (Czarnecki et al., 2017). In equation 16, we define the new loss function, which draws a batch uniformly from the buffer and regresses the current network's gradient and value with the batch labels. We use an L1 norm for better interpretability of each term's gradient to the total optimizer update.

$$\mathcal{L}_{\text{Sobolev}}(\theta) = \sum_{(\boldsymbol{x}_i, \tau_i, \boldsymbol{p}_i, V_i^c) \sim \mathcal{E}}^{N_{\text{Sobolev}}} w_g \left\| \nabla_{\boldsymbol{x}} V_\theta(\boldsymbol{x}_i, \tau_i) - \boldsymbol{p}_i \right\|_1 + w_v \left| V_\theta(\boldsymbol{x}_i, \tau_i) - V_i^c \right|, \tag{16}$$

We minimize the total objective $\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{Sobolev}}$ with the Adam optimizer (Kingma & Ba, 2014) with a constant learning rate $\alpha$ over the curriculum schedule.

The replay buffer $\mathcal{E}$ can grow without bound, so that every curriculum stage stays represented by the uniform draw from the buffer. This way, we can prevent catastrophic forgetting. Since both terms are batch sums, the weights $w_g$ and $w_v$ have to be balanced against each other, so that the loss equation 16 does not overpower the residual, or the reverse.

**Role of PMP supervision.** PMP provides only necessary conditions for optimality, so trajectories generated from the current network may reflect local rather than globally optimal behavior. To reduce the influence of such errors, we do not rely on a fixed set of PMP labels throughout training. Instead, we periodically reseed from the updated value function as the curriculum progresses, allowing the supervision to evolve together with the network, as described in the steps above.

**False-positive fine-tuning.** For control synthesis, the value function may classify a state as safe when the induced policy fails to keep the system safe. For challenging long-horizon tasks where such errors can compound, we optionally follow training with a short fine-tuning phase focused on such false positives. We sample states with $|V_\theta(\boldsymbol{x}_j, \tau_i)| \leq b$, evaluate the network-induced policy, and use its achieved safety margin $V_i = \zeta(\boldsymbol{x}_i)$ as the target. We minimize

$$\mathcal{L}_{\text{FT}}(\theta) = \frac{1}{n} \sum_i \omega_i \left| V_i - V_\theta(\boldsymbol{x}_i, \tau_i) \right| + \frac{1}{N_{\text{PDE}}} \mathcal{L}_{\text{PDE}}(\theta), \quad \omega_i = \begin{cases} \lambda_{\text{FP}} \max\left(V_\theta, 0\right), & V_\theta \geq 0 > V_i \\ w & \text{otherwise} \end{cases}. \tag{17}$$

We fine-tune for $N_{\text{FT}}$ steps with a learning rate of $\alpha_{\text{FT}}$, taking only a few minutes on a single GPU.

## 4 Simulation Experiments

We evaluate our method on two systems: a 13-dimensional quadrotor navigating a cluttered environment and an 8-dimensional high-speed racing vehicle operating near the limit of handling. Both present unique safety challenges: the quadrotor system often needs to navigate through narrow gaps, requiring very high accuracy in the safety value function. On the other hand, the highly nonlinear tire dynamics near the handling limit make value function computation challenging for the racing vehicle.

**Baselines.** We compare against several state-of-the-art baselines: **ExactDeepReach** (Singh et al., 2025) uses the same parametrization, architecture, and curriculum as our method but trains only on the HJI residual, isolating the effect of PMP supervision. **DeepReachPMP** (Dallas et al., 2026) uses PMP trajectories only to place collocation points, testing PMP for residual sampling rather than direct value and gradient supervision. **DeepReachMPC** (Feng et al., 2025) uses refreshed labels from massively sampling-based MPC, providing the strongests prior label-based comparison. **MADR** (Teoh et al., 2025) extends MPC supervision to adversarial disturbance and serves as the corresponding baseline

for disturbance-driven problems. **Gradient-Free HJR** avoids spatial gradients via discrete-time Bellman–Isaacs recursion, contrasting with our continuous-time formulation. Finally, **ISAACS** (Hsu et al., 2023) represents reinforcement learning based methods that bypass that HJI PDE by jointly learning a safety critic and adversarial policies.

**Evaluation.** The evaluation metrics vary across experiments. In all settings, we report the true positive rate (TPR) and false positive rate (FPR), estimated using $500,000$ rollouts under the policy induced by each learned value function. We additionally perform conformal prediction on the learned value function using the method proposed in (Lin & Bansal, 2024). This ultimately leads to a calibrated, high-confidence safe set for the system. We compare the volume of the calibrated safe set across different methods. We report n/a if no volume can be certified with $500{,}000$ samples.

For each model, we report the mean and standard deviation across five different seeds.

### 4.1 13 Dimensional Quadrotor

We use the 13-dimensional quadrotor dynamics as described in Appendix A.2.1. We use two different setups: (a) ***Single cylinder obstacle:*** the quadrotor needs to safely navigate around a single cylindrical obstacle. This is a standard toy problem often used in reachability studies to assess and compare methods. (b) ***Multiple cylinder obstacles in close vicinity:*** we place four cylindrical obstacles very close to each other, creating a narrow traversal gap. To pass through the gap, the quadrotor must roll by at least $54^\circ$ from hover, thereby requiring aggressive avoidance maneuvers. Consequently, a state approaching the cage at high speed is safe only if the quadrotor can brake sufficiently or execute an evasive maneuver around the obstacle in time. The resulting safe set is therefore shaped by aggressive avoidance maneuvers, which the induced controller must recover from $\nabla_{\boldsymbol{x}} V$.

Table 1 shows the quantitative results for the single cylinder obstacle case comparing our method against DeepReachMPC. Results show that both perform the same, up to noise, while our method is about 40% faster in total training time.

| **Method** | **TPR** $\uparrow$ | **FPR** $\downarrow$ | $\mu_\epsilon \uparrow$ | $\delta_{\textbf{cal}}$ | **Time (h)** $\downarrow$ |
|---|---|---|---|---|---|
| DeepReachMPC | **0.999±0.000** | 0.021±0.001 | **95.272±0.044** | **0.008±0.009** | 5.164±0.037 |
| Ours | 0.996±0.001 | **0.019±0.010** | 94.918±0.031 | **0.008±0.009** | **3.16±0.026** |

Table 1: **Single cylinder.** TPR and FPR against rollouts of each method's own policy, certified safe volume $\mu_\varepsilon$ with calibration level $\delta_{\mathrm{cal}}$, and training time. Mean $\pm$ std over five seeds.

Table 2 summarizes the results for the multi-cylinder obstacle setting. DeepReachMPC and our method achieve the same TPR, while FPR of DeepReachMPC is twice that of our method. Crucially, our probabilistic safety guarantee yields a certified safe volume of 78.98±0.24%, compared with 57.94±19.47% for DeepReachMPC. The most important finding here is the extremely high variance in the certified safe volume. In Figure 1, we show how the false positives increase as the speed towards the cylinders increases. In the given problem, the optimal avoidance control set is very small, especially as the speed increases. The maneuvers that avoid a collision are near-saturated maneuvers, i.e., full brake or full swerve, that must start immediately. The chance that a random shooting finds this thin set shrinks as speed increases is rather low, which is exactly backed by the results in Fig. 1, where it shows that for DeepReachMPC the false positives grow rapidly, whereas for our method this metric stays flat, since instead of random search, we derive the optimal control in one shot from Pontryagin's Maximum Principle. Most of the RL baseline's false positives are sttes that could have been saved: from 94% of them, our controller keeps the quadrotor safe. The failure lies in its controller, which when approaching the cage at high speed, rarely commands the full thrust, maximum rotation maneuvers needed to escape.

**Qualitative Results.** Figure 2 shows components of $\nabla_{\boldsymbol{x}} V_\theta$ on the $(x, y)$ plane for DeepReachMPC's worst performing seed and for ours. DeepReachMPC's gradient exhibits high-frequency ripples, most prominently in $\partial V_\theta / \partial \omega_x$, whereas ours remains smooth. The ripples follow from the supervision: DeepReachMPC fits the value function using sparse, noisy value labels, allowing the network to introduce high-frequency components between labeled points while still matching the observations. Such components may be barely visible in $V_\theta$ itself, but differentiation amplifies them in proportion

to their frequency, causing them to dominate the gradient. In contrast, our labels are deterministic and densely sampled along the level sets. Because we also directly supervise the gradient, high-frequency components are penalized increasingly strongly as their frequency grows.

| Method | TPR ↑ | FPR ↓ | $\mu_\epsilon$ [%] ↑ | $\delta_{\text{cal}}$ | Time (h) ↓ |
|---|---|---|---|---|---|
| ExactDeepReach | 0.996±0.002 | 0.369±0.143 | 34.66±12.81 | 1.262±0.288 | 3.28±0.08 |
| RL baseline | 0.933±0.004 | 0.164±0.022 | n/a | n/a | **1.72±0.01** |
| DeepReachMPC | 0.988±0.004 | 0.051±0.027 | 57.94±19.47 | 0.660±0.498 | 5.81±0.05 |
| Ours | 0.987±0.004 | **0.024±0.014** | **78.98±0.24** | **0.073±0.020** | **4.45±0.05** |

Table 2: **Four cylinders.** TPR and FPR against rollouts of each method's own policy, certified safe volume $\mu_\varepsilon$ with calibration level $\delta_{\text{cal}}$, and training time. Mean ± std over five seeds.

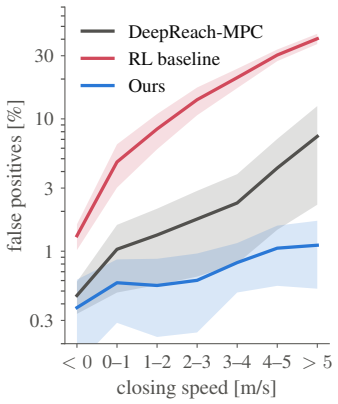


Figure 1: **False positives vs. closing speed.**

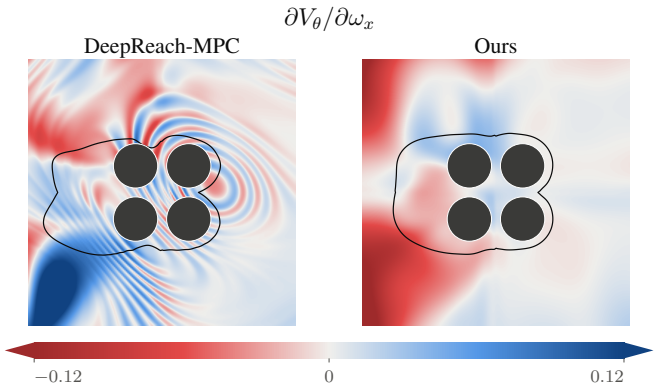


Figure 2: **Gradient of the value function.** $\partial V_\theta/\partial \omega_x$ on the $(x, y)$ plane at $v_x = 3$ m/s.

### 4.2 Vehicle at the handling limit

We consider an eight-state single-track vehicle model in the Frenet frame – described in Appendix A.2.2 – of a closed track as seen in Fig. 4. We learn the value function over a horizon of 8 seconds. We compare against MADR (Teoh et al., 2025), the gradient-free discrete-time solver of (Feng et al., 2026), the reinforcement learning baseline ISAACS (Hsu et al., 2023). In Table 3 MADR leads in TPR and matches our FPR at four times the training time. The gradient-free method and the RL baseline train for the same time to the same TPR, but the RL critic, fitted by temporal difference regression without dynamics information, has twice the FPR. Appendix A.7 shows that MADR trained under our label budget, plain ExactDeepReach, and the boundary-sampling method of Dallas et al. (2026) all fail to learn a usable tube on this system.

| Method | TPR ↑ | FPR ↓ | $\mu_\epsilon$ | $\delta_{\text{cal}}$ | Time (h) ↓ |
|---|---|---|---|---|---|
| RL Baseline | 0.868±0.011 | 0.0187±0.0003 | n/a | n/a | 8.20±0.28 |
| Gradient Free | 0.865 | 0.0089 | n/a | n/a | 8.50 |
| MADR | **0.940±0.004** | **0.0032±0.0003** | n/a | n/a | 12.23±0.23 |
| Ours | 0.907±0.011 | 0.0034±0.0006 | n/a | n/a | **3.08±0.10** |

Table 3: **Vehicle, offline.** TPR and FPR against rollouts of each method's own policy under the worst-case disturbance, certified volume $\mu_\varepsilon$ with calibration level $\delta_{\text{cal}}$, and training time. Mean ± std over seeds.

**Simulation.** We deploy each value function using a sampling-based control-barrier filter (Appendix A.3) around a controller that tracks the centerline at the grip limit, with the results reported in

Tab. 4. All filters prevent collisions. The gradient free method is fastest on the straights but induces sliding, exceeding $\mu g$ in every trial and producing three times the sideslip of our method. MADR slides but is the slowest. Our method achieves the fastest cornering while matching the gradient free methods lap time, with the slowest jerk and sideslip.

| **Method** | **Coll.** ↓ | **Interv.** % ↓ | **Speed** $[\frac{m}{s}]$ ↑ | **Corner S.** $[\frac{m}{s}]$ ↑ | **Lap** $[s]$ ↓ | **Jerk** ↓ | $\min \ell(x)$ $[m]$ ↑ | $\max \lvert e \rvert$ $[m]$ ↓ | $\max \lvert \beta \rvert$ $[rad]$ ↓ |
|---|---|---|---|---|---|---|---|---|---|
| RL baseline | 0/100 | 97.8±0.8 | 9.91±0.20 | 9.73±0.43 | >20 | 0.015±0.001 | 2.06±0.05 | 1.34±0.05 | 0.059±0.031 |
| MADR | 0/100 | **69.9±2.8** | 11.47±0.15 | 10.95±0.38 | 19.47±0.22 | 0.013±0.001 | 2.58±0.37 | 0.82±0.37 | 0.052±0.046 |
| Gradient Free | 0/100 | 70.4±2.8 | **11.66±0.16** | 11.11±0.39 | **19.18±0.27** | 0.013±0.001 | 2.46±0.32 | 0.94±0.32 | 0.135±0.053 |
| Ours | 0/100 | 71.9±2.8 | 11.64±0.15 | **11.21±0.41** | 19.25±0.24 | **0.012±0.001** | **2.72±0.39** | **0.68±0.39** | **0.044±0.037** |

Table 4: **Vehicle, simulation.** 100 trials of 20 s from shared initial states (mean±std). Shading: best, second, third.

## 5 Hardware Experiments

**Hardware.** The same three tubes are evaluated over eight laps each on a full-scale electric vehicle using the same controller and filter (Table 5, Appendix A.4), and the separation persists on the real vehicle: the gradient-free tube slides at 1.10 g, MADR is 0.3 s slower per lap, and ours corners fastest with the least jerk and sideslip. Its higher intervention rate reflects time spent at the speed cap on the straights, while its larger $\max |e|$ results from the tracker unwinding the steering at corner exit, 2.8 m from the wall. The car spends 6.6 % of the time steps outside our safe set, compared with 12.1 and 16.9 % for the baselines, meaning that the real trajectory crosses our boundary roughly half as often.

| **Method** | **Coll.** ↓ | **Interv.** % ↓ | **Speed** $[\frac{m}{s}]$ ↑ | **Corner S.** $[\frac{m}{s}]$ ↑ | **Lap** $[s]$ ↓ | **Jerk** ↓ | $\min \ell(x)$ $[m]$ ↑ | $\max \lvert e \rvert$ $[m]$ ↓ | $\max \lvert \beta \rvert$ $[rad]$ ↓ |
|---|---|---|---|---|---|---|---|---|---|
| MADR | 0/8 | 72.8±1.5 | 11.77±0.26 | 11.28±0.28 | 19.0±0.5 | 0.051±0.002 | **2.94±0.09** | **0.46±0.09** | 0.040±0.005 |
| Gradient Free | 0/8 | **69.7±1.6** | **12.08±0.09** | 11.54±0.08 | 18.7±0.5 | 0.055±0.004 | 2.87±0.05 | 0.53±0.05 | 0.069±0.006 |
| Ours | 0/8 | 74.6±0.4 | 12.01±0.16 | **11.60±0.17** | **18.7±0.3** | **0.045±0.002** | 2.82±0.00 | 0.58±0.00 | **0.031±0.001** |

Table 5: **Vehicle, hardware.** Eight consecutive laps per network, (mean±std). Shading: best, second, third.

**Adversarial driving.** Finally, we replace the autonomous controller with a human driver instructed to leave the track, while using our model as the safety filter. Despite the driver's attempts, the vehicle never left the track. To quantify the aggressiveness of the driver's inputs, we roll out the vehicle model from each logged state while holding the driver's requested control constant (Fig. 3). Under this counterfactual evaluation, 78% of the requests would have caused the vehicle to leave the track within 2 s, and 59% within 1.2 s, compared with 40% for the autonomous controller held over the same horizon.

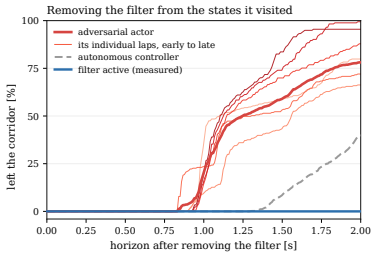


Figure 3: **Adversarial driving.**

## 6 Conclusion

**Contributions.** We learn Hamilton–Jacobi reachable tubes by supervising the value and its gradient with labels transported along characteristics from the network's own level sets, which costs one integration per label instead of a sampling-based solve. On a 13-D quadrotor and an 8-D vehicle the tubes match the accuracy of the best baseline at a quarter of its training time, and the vehicle tube kept a full-scale car inside its corridor at the friction limit and against a driver trying to leave it.
**Limitations.** The labels are seeded on the network's own level sets, so early errors can propagate.

### AI USE STATEMENT

AI was used to proofread the manuscript for typographical and grammatical errors, support the literature search, and assist with writing and reviewing portions of the experimental and analytical code. All methods, experiments, results, interpretations, and claims were conceived, conducted, verified, and written by the authors, who take full responsibility for the content.

### ETHICS STATEMENT

The authors do not feel that this paper raises questions regarding the Code of Ethics.

### REPRODUCIBILITY STATEMENT

All models for the quadrotor experiments (Sec. 4.1) were trained on a machine with three NVIDIA RTX A5000 GPUs, and all models for the vehicle experiments (Section 4.2) on a machine with two NVIDIA RTX 3090 GPUs. Each machine has 24 physical CPU cores, and each run uses a single GPU. Because the experiments also involve substantial CPU processing, we limit each run to eight CPU threads to prevent concurrent runs on the same machine from competing for CPU resources. The code used to train our models can be accessed anonymously here.

## REFERENCES

Ayush Agrawal and Koushil Sreenath. Discrete control barrier functions for safety-critical control of discrete systems with application to bipedal robot navigation. In *Robotics: Science and systems*, volume 13, pp. 1–10. Cambridge, MA, USA, 2017.

Somil Bansal and Claire J Tomlin. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1817–1824. IEEE, 2021.

Somil Bansal, Mo Chen, Sylvia Herbert, and Claire J Tomlin. Hamilton-Jacobi Reachability: A brief overview and recent advances. In *IEEE Conference on Decision and Control (CDC)*, 2017.

Javier Borquez, Kaustav Chakraborty, Hao Wang, and Somil Bansal. On safety and liveness filtering using hamilton-jacobi reachability analysis. *IEEE Transactions on Robotics (T-RO)*, 2024.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in neural information processing systems*, 30, 2017.

James Dallas, Thomas Lew, John Talbot, Jonathan DeCastro, Somil Bansal, and John Subosits. Boundary sampling to learn predictive safety filters via pontryagin's maximum principle. *arXiv preprint arXiv:2604.13325*, 2026.

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, 1984.

Zeyuan Feng, Le Qiu, and Somil Bansal. Bridging model predictive control and deep learning for scalable reachability analysis. *arXiv preprint arXiv:2505.03830*, 2025.

Zeyuan Feng, Ali Fuat Sahin, Santiago Thorup, and Somil Bansal. Gradient-free neural hamilton-jacobi reachability for scalable safety-critical control. *arXiv preprint arXiv:2609.14087*, 2026.

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)*, pp. 8550–8556. IEEE, 2019.

William Hofgard. Convergence guarantees for neural network-based hamilton-jacobi reachability. *arXiv preprint arXiv:2410.02904*, 2024.

Kai-Chieh Hsu, Duy Phuong Nguyen, and Jaime Fernandez Fisac. Isaacs: Iterative soft adversarial actor-critic for safety. In *Learning for Dynamics and Control Conference*, pp. 90–103. PMLR, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Albert Lin and Somil Bansal. Verification of neural reachable tubes via scenario optimization and conformal prediction. In *6th Annual Learning for Dynamics & Control Conference*, pp. 719–731. PMLR, 2024.

Tenavi Nakamura-Zimmerer, Qi Gong, and Wei Kang. Adaptive deep learning for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(2):A1221–A1247, 2021.

Sungje Park and Stephen Tu. Forward trajectory steering for hamilton-jacobi reachability analysis. *arXiv preprint arXiv:2608.11480*, 2026.

Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

Aditya Singh, Zeyuan Feng, and Somil Bansal. Exact imposition of safety boundary conditions in neural reachable tubes. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5489–5495. IEEE, 2025.

Ryan Teoh, Sander Tonkens, William Sharpless, Aijia Yang, Zeyuan Feng, Somil Bansal, and Sylvia Herbert. Madr: Mpc-guided adversarial deepreach. *arXiv preprint arXiv:2510.18845*, 2025.

Santiago Thorup, Luca Castelletto, Zeyuan Feng, and Somil Bansal. Neural backward reach-avoid tubes with mpc supervision for high-dimensional systems: An application to safe spacecraft docking. *arXiv preprint arXiv:2605.02021*, 2026.

# CONTENTS

**1 Introduction** **1**

**2 Problem Formulation** **2**

2.1 Hamilton-Jacobi Reachability . . . . . 3

2.2 Neural Hamilton Jacobi Reachability . . . . . 3

**3 Method** **3**

3.1 Pontryagin's Maximum Principle . . . . . 4

3.2 Guiding Reachability Learning with PMP . . . . . 5

**4 Simulation Experiments** **6**

4.1 13 Dimensional Quadrotor . . . . . 7

4.2 Vehicle at the handling limit . . . . . 8

**5 Hardware Experiments** **9**

**6 Conclusion** **9**

**A Appendix** **14**

A.1 Configuration Parameters . . . . . 14

A.2 Dynamics . . . . . 14

A.2.1 Quadrotor . . . . . 14

A.2.2 Vehicle Dynamics . . . . . 15

A.3 Sampling-based CBF Filter and Autonomous Controller . . . . . 15

A.4 Hardware Experiment . . . . . 16

A.4.1 Skidpad . . . . . 16

A.5 Proof of Lemma 1 . . . . . 16

A.6 Proof of Proposition 2 . . . . . 16

A.7 Further Results on the Skidpad . . . . . 17

## List of Tables


1 **Single cylinder.** TPR and FPR against rollouts of each method's own policy, certified safe volume $\mu_\varepsilon$ with calibration level $\delta_{\mathrm{cal}}$, and training time. Mean $\pm$ std over five seeds. . . . . 7

2 **Four cylinders.** TPR and FPR against rollouts of each method's own policy, certified safe volume $\mu_\varepsilon$ with calibration level $\delta_{\mathrm{cal}}$, and training time. Mean $\pm$ std over five seeds. . . . . 8

3 **Vehicle, offline.** TPR and FPR against rollouts of each method's own policy under the worst-case disturbance, certified volume $\mu_\varepsilon$ with calibration level $\delta_{\mathrm{cal}}$, and training time. Mean $\pm$ std over seeds. . . . . 8

4 **Vehicle, simulation.** 100 trials of 20 s from shared initial states (mean±std). Shading: best, second, third. . . . . 9

5 **Vehicle, hardware.** Eight consecutive laps per network, (mean±std). Shading: best, second, third. . . . . 9

6 **Config parameters** . . . . . 14

7 **Vehicle, sample efficiency.** MADR$^\star$: MADR with its label generation capped to 19 s per regeneration; ExactDeepReach: no labels; DeepReachPMP: boundary characteristics generated once. TPR, FPR and soundness against rollouts of each method's own policy; n/a where no state is claimed safe. . . . . 17


## List of Figures


1 **False positives vs. closing speed.** . . . . . 8

2 **Gradient of the value function.** $\partial V_\theta / \partial \omega_x$ on the $(x, y)$ plane at $v_x = 3$ m/s. . . . 8

3 **Adversarial driving.** . . . . . 9

4 The skidpad is a 223 m oval with two corners each measuring 20 m in radius and a corridor measuring 3.4 m either side of the centerline. . . . . 16

# A Appendix

## A.1 Configuration Parameters

| Method | SC Quadrotor (4.1) | MC Quadrotor (4.1) | Vehicle (4.2) |
|---|---|---|---|
| $\gamma_{\min}$ | 0.05 | 0.05 | 0.05 |
| $\gamma_{\max}$ | 100 | 100 | 100 |
| $\varepsilon_V$ | 0.01 | 0.01 | 0.01 |
| $\varepsilon_w$ | 0.1 | 0.1 | 0.1 |
| $q_-$ | -0.45 | -0.45 | -3.0 |
| $q_+$ | 3.0 | 2.0 | 2.5 |
| $\varepsilon_\ell$ | $10^{-3}$ | $10^{-3}$ | $10^{-3}$ |
| $w_v$ | 30 | 30 | 9.7 |
| $w_g$ | 0.16 | 0.16 | 0.21 |
| $\lambda_{\text{FP}}$ | – | – | 100 |
| $w$ | – | – | 1 |
| $h_{\max}$ | 0.2 | 0.2 | 0.2 |
| $k$ | 100'000 | 100'000 | 300'000 |
| $N_{\text{Sobolev}}$ | 5'000 | 5'000 | 5'000 |
| $N_{\text{PDE}}$ | 65'000 | 65'000 | 65'000 |
| $N_{\text{FT}}$ | – | – | 10'000 |
| $\alpha$ | $2 \times 10^{-5}$ | $2 \times 10^{-5}$ | $2 \times 10^{-5}$ |
| $\alpha_{\text{FT}}$ | – | – | $2 \times 10^{-6}$ |

Table 6: **Config parameters**

## A.2 Dynamics

### A.2.1 Quadrotor

We model the dynamics of the quadrotor with the 13-dimensional state

$$\boldsymbol{x} = (\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\omega}) \tag{18}$$

where $\boldsymbol{p}, \boldsymbol{v}, \boldsymbol{\omega} \in \mathbb{R}^3$ denote the position, translational velocity, and body angular velocity respectively, and $\boldsymbol{q} \in \mathbb{S}^3$ is the quaternion with unit-norm constraint $\|\boldsymbol{q}\|_2 = 1$. The control input is $u = (f, a)$, with $a = (a_x, a_y, a_z) \in \mathbb{R}^3$, where $f$ is the collective thrust and $a$ contains the angular-acceleration control inputs. The dynamics are

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{19}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{20}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{g} + \frac{C_T}{m} f \boldsymbol{R}(\boldsymbol{q}) \boldsymbol{e}_3 \tag{21}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{a} + \boldsymbol{c}(\boldsymbol{\omega}) \tag{22}$$

where $\otimes$ denotes the quaternion multiplication, $\boldsymbol{g} = [0 \quad 0 \quad G_z]^\top$, $\boldsymbol{e}_3 = [0 \quad 0 \quad 1]^\top$, and $\boldsymbol{R}(\boldsymbol{q})$ is the rotation matrix associated with $\boldsymbol{q}$. The gyroscopic coupling term used in our model is $\boldsymbol{c}(\boldsymbol{\omega}) = \left[-\frac{5}{9}\omega_y\omega_z \quad \frac{5}{9}\omega_x\omega_z \quad 0\right]^\top$. We use $m = 1$, $C_T = 1$, $G_z = -9.8$. Hence the translational dynamics simplify to $\dot{\boldsymbol{v}} = [0 \quad 0 \quad -9.8]^\top + f\boldsymbol{R}(\boldsymbol{q})\boldsymbol{e}_3$.

**Failure set.**

The quadrotor is modeled as a disk of radius $r = 0.17$ m centered at the position and orthogonal to the body axis $\boldsymbol{n} = R(\boldsymbol{q})\boldsymbol{e}_3$. For a vertical cylinder with axis $\boldsymbol{c} = (c_x, c_y)$ and radius $R_c$, let $\Delta$ be the

horizontal offset from the axis to the quadrotor, $\rho = \|\Delta\|$, and $\boldsymbol{n}_{xy} = (n_x, n_y)$. The clearance is

$$d_c(\boldsymbol{x}) = \max\left\{\rho - \frac{r|n_z|\rho}{\sqrt{\left(\Delta^\top \boldsymbol{n}_{xy}\right)^2 + n_z^2\rho^2}}, 0\right\} - R_c, \tag{23}$$

where the subtracted term is the horizontal half-length of the disk's intersection with the vertical plane through the quadrotor and the cylinder axis. The single-cylinder task uses $\ell(\boldsymbol{x}) = d_{\boldsymbol{c}_1}(\boldsymbol{x})$ with $\boldsymbol{c}_1 = (0, 0)$. The four-cylinder task uses $\ell(\boldsymbol{x}) = \min_i d_{\boldsymbol{c}_i}(\boldsymbol{x})$ with centers $\boldsymbol{c}_{1,\dots,4} = (0.6, 0.6)$ m, so that adjacent cylinders are separated by a gap of $0.2$ m. In both tasks, $R_c = 0.5$ m.

### A.2.2 Vehicle Dynamics

We model the vehicle in a Frenet frame that is attached to the centerline of the oval skid pad of Fig. 4. The dynamic state is

$$\boldsymbol{x} = [s, e, \Delta\varphi, v, r, \beta, \delta, \tau] \in \mathbb{R}^8, \tag{24}$$

where $s$ is the arc length, $e$ the lateral offset (positive to the left of the centerline), $\Delta\varphi$ the heading error relative to the centerline tangent, $v$ the speed, $r$ the yaw rate, $\beta$ the body slip angle, $\delta$ the road-wheel steering angle, and $\tau$ the longitudinal tire force. The control inputs are the rates $\boldsymbol{u} = [\dot{\delta}, \dot{\tau}]$. The disturbance $\boldsymbol{d} = [d_0, d_1]$ collects the additive yaw acceleration $d_0$ and lateral drift $d_1$ that acts as an unmodeled lateral velocity $w = d_s(|v| + 1)d_1$. The friction coefficient $\mu$ and the disturbance scale $d_s$ are treated as a context: they are held constant along every trajectory, sampled uniformly and appended to the network input, so one network represents the value function for every grip level.

**Dynamics.** With the course angle $\chi = \Delta\varphi + \beta$ and the centerline curvature $\kappa(s)$,

$$\dot{s} = \frac{v\cos(\chi) - w\sin(\chi)}{1 - \kappa(s)e} \tag{25}$$

$$\dot{e} = v\sin(\chi) + w\cos(\chi) \tag{26}$$

$$\dot{\Delta\varphi} = r - \kappa(s)\dot{s} \tag{27}$$

$$\dot{v} = \frac{1}{M}\left[F_{xf}\cos(\delta - \beta) - F_{yf}\sin(\delta - \beta) + F_{xr}\cos(\beta) + F_{yr}\sin(\beta)\right] \tag{28}$$

$$\dot{r} = \frac{1}{I_{zz}}\left[\ell_f\left(F_{xf}\sin(\delta) + F_{yf}\cos(\delta)\right) - \ell_r F_{yr}\right] + d_s d_0 \tag{29}$$

$$\dot{\beta} = \frac{F_{xf}\sin(\delta - \beta) + F_{yf}\cos(\delta - \beta) - F_{xr}\sin(\beta) + F_{yr}\cos(\beta)}{Mv} - r \tag{30}$$

where $w = d_s\left(|v| + 1\right)d_1$.

**Failure set.** The car counts as off the track once its center of gravity crosses a corridor wall, $\ell(\boldsymbol{x}) = 3.4 - |e|$.

### A.3 Sampling-based CBF Filter and Autonomous Controller

At runtime the autonomous controller or the human driver proposes a rate command $\boldsymbol{u}_{\text{nom}}$. The filter queries the learned value at the current state, $V_\theta(\boldsymbol{x}, \tau_q)$ with $\tau_q = 7.2$s on the deployed slice $\mu = 0.92$, $d_s = 1.0$, and applies the closest admissible input that satisfies a discrete-time barrier decrease condition (Agrawal & Sreenath, 2017),

$$V_\theta\left(F_{\Delta t}(\boldsymbol{x}, \boldsymbol{u}), \tau_q\right) \geq \max\left\{(1 - \gamma\Delta t)V_\theta(\boldsymbol{x}, \tau_q), 0\right\}, \qquad \gamma = 3\text{s}^{-1}, \Delta t = 20 \tag{31}$$

where $F_{\Delta t}$ is the state predicted $\Delta t$ ahead under $\boldsymbol{u}$ (five Euler sub-steps of the vehicle model at $\mu = 0.92$, no disturbance). Because $V_\theta$ is not differentiated at runtime, the condition is evaluated over a sampled batch of 128 candidates in one forward pass: the nominal command, the four vertices of the control box, 16 Gaussian perturbations of the nominal with standard deviation equal to the actuator bound, 107 uniform samples from the box, and one policy slot, which for the gradient-free baseline holds the action of its co-trained policy and for the other networks the zero-rate command. Let $\mathcal{U}_{\text{safe}}$ be the feasible subset. The applied control is

$$\boldsymbol{u}^\star = \begin{cases} \arg\min_{\boldsymbol{u} \in \mathcal{U}_{\text{safe}}} \|\Lambda(\boldsymbol{u} - \boldsymbol{u}_{\text{nom}})\|_2, & \mathcal{U}_{\text{safe}} \neq \emptyset \\ \arg\max_{\boldsymbol{u} \in \mathcal{U}_{\text{batch}}} V_\theta(F_{\Delta t}(\boldsymbol{x}, \boldsymbol{u}), \tau_q), & \text{otherwise} \end{cases} \tag{32}$$

where the weighting $\Lambda = \mathrm{diag}(1/\dot{\delta}_{\max}, 1/\dot{\tau}_{\max})$ measures the deviation in fractions of each actuator's range, so that a steering and a torque deviation of the same fraction count the same. The filter runs at 200 Hz on the vehicle computer; the same code path is used in simulation and on the car, and the same filter and parameters are used for every value function. We apply no calibration offset to $V_\theta$.

### A.4 Hardware Experiment

All experiments were conducted on the track shown in Figure 4 using a full-scale, custom steer-by-wire electric vehicle. Vehicle state estimates were provided at 200 Hz by a dual-antenna RTK GPS/INS system. The filter was executed onboard using an NVIDIA GeForce RTX 4070 Ti SUPER GPU and a 13th-generation Intel Core i9-13900E processor.

#### A.4.1 Skidpad

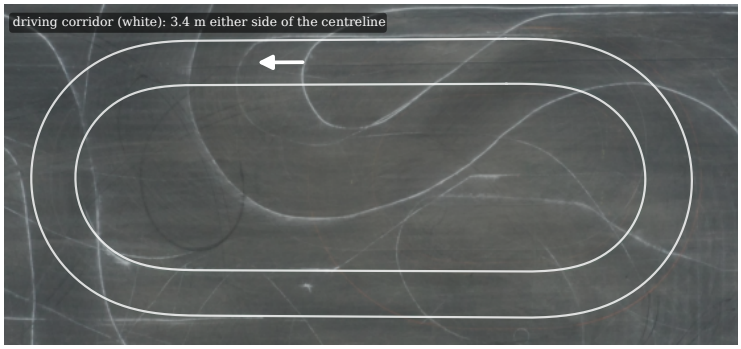


Figure 4: The skidpad is a 223 m oval with two corners each measuring 20 m in radius and a corridor measuring 3.4 m either side of the centerline.

### A.5 Proof of Lemma 1

*Proof.* The Hamiltonian function in equation 6 is positively homogeneous of degree one in the costate: for any $\lambda > 0$,

$$H(\boldsymbol{x}, \lambda \boldsymbol{p}) = \max_{\boldsymbol{u}\in\mathcal{U}} \min_{\boldsymbol{d}\in\mathcal{D}} \langle \lambda \boldsymbol{p}, f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d})\rangle = \lambda \max_{\boldsymbol{u}\in\mathcal{U}} \min_{\boldsymbol{d}\in\mathcal{D}} \langle \boldsymbol{p}, f(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{d})\rangle = \lambda H(\boldsymbol{x}, \boldsymbol{p}). \tag{33}$$

Whenever $V(\boldsymbol{x}, t) < \ell(\boldsymbol{x})$, the constraint $V(\boldsymbol{x}, t) = \ell(\boldsymbol{x})$ in equation 5 is inactive, so there it holds that $\frac{\partial}{\partial t} V(\boldsymbol{x}, t) = -H(\boldsymbol{x}, \nabla_{\boldsymbol{x}} V)$. Taking the total derivative of the value function $V$ along a characteristic, we get

$$\frac{d}{ds} V(\boldsymbol{x}(s), s) = \partial_t V + \langle \boldsymbol{p}, \dot{\boldsymbol{x}} \rangle \overset{(1)}{=} -H(\boldsymbol{x}, \boldsymbol{p}) + \langle \boldsymbol{p}, \nabla_{\boldsymbol{p}} H(\boldsymbol{x}, \boldsymbol{p}) \rangle \overset{(2)}{=} 0 \tag{34}$$

where $(1)$ follows from both the Hamilton-Jacobi equation and equation 10a, and $(2)$ from Euler's identity for functions positively homogeneous of degree one, $\langle \boldsymbol{p}, \nabla_{\boldsymbol{p}} H \rangle = H$. □

### A.6 Proof of Proposition 2

*Proof.* Fix an admissible pair of $\boldsymbol{u}$ and $\boldsymbol{d}$. Let $\boldsymbol{\xi}(\cdot)$ denote the resulting trajectory. Since $t \in [t, T]$ and $\boldsymbol{\xi}(t) = \boldsymbol{x}$, the running minimum includes the initial state,

$$\min_{s\in[t,T]} \ell(\boldsymbol{\xi}(s)) \le \ell(\boldsymbol{x}). \tag{35}$$

The bound holds for every pair and the right-hand side is independent of either player, so it is preserved by taking $\inf_{\boldsymbol{d}}$ and then $\sup_{\boldsymbol{u}}$, in either order. Hence $V(\boldsymbol{x}, t) \le \ell(\boldsymbol{x})$. □

### A.7 Further Results on the Skidpad

**Sample efficiency and ablations.** The baselines in Table 3 require two to four times longer training than our method. To disentangle the effect of the labels from that of the training budget, Table 7 retrains three of these methods under the same conditions as ours. MADR$^\star$ is MADR (Teoh et al., 2025) with its label-generation budget capped, at every regeneration, to the wall-clock time of one of our reseeding stages ($19\,\mathrm{s}$), using a batch of $5{,}000$ states instead of $10{,}000$ to match our setting. Under this budget it learns no usable tube, claiming $0.02\,\%$ of the state space as safe. ExactDeepReach (Singh et al., 2025) is the residual-only method on which our implementation is built, trained with the same network and curriculum but without labels, thereby isolating the contribution of the labels themselves; its zero sub-level set covers the entire state space. DeepReachPMP (Dallas et al., 2026) generates boundary characteristics once before training and uses them as additional collocation points. On this system, the characteristics leave the state box within half a second of the boundary, and therefore cover only a thin layer of the eight-second horizon; consequently, the tube does not converge either. Where no state is claimed safe, FPR and soundness are undefined.

| **Method** | **TPR** $\uparrow$ | **FPR** $\downarrow$ | **Sound [%]** $\uparrow$ | $\mu_\epsilon$ | $\delta_{\mathrm{cal}}$ | **Time (h)** $\downarrow$ |
|---|---|---|---|---|---|---|
| MADR$^\star$ | 0.001 | 0.000 | 63.64 | 0.000 | n/a | 3.9 |
| ExactDeepReach | 0.000 | n/a | n/a | 0.000 | n/a | 2.1 |
| DeepReachPMP | 0.000 | n/a | n/a | 0.000 | n/a | 2.1 |
| Ours | 0.907±0.011 | 0.0034±0.0006 | 95.97±0.66 | 0.218 | 0.29 | 3.08±0.10 |

Table 7: **Vehicle, sample efficiency.** MADR$^\star$: MADR with its label generation capped to $19\,\mathrm{s}$ per regeneration; ExactDeepReach: no labels; DeepReachPMP: boundary characteristics generated once. TPR, FPR and soundness against rollouts of each method's own policy; n/a where no state is claimed safe.