**Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021**

Safe Reinforcement Learning via Statistical Model Predictive Shielding

Osbert Bastani
University of Pennsylvania
obastani@seas.upenn.edu

Shuo Li
University of Pennsylvania
lishuo1@seas.upenn.edu

Anton Xu
University of Pennsylvania
antonxue@seas.upenn.edu

Abstract—**Reinforcement learning is a promising approach to solving hard robotics tasks. An important challenge is ensuring safety—e.g., that a walking robot does not fall over or an autonomous car does not crash into an obstacle. We build on an approach that composes the learned policy with a backup policy—it uses the learned policy on the interior of the region where the backup policy is guaranteed to be safe, and switches to the backup policy on the boundary of this region. The key challenge is checking when the backup policy is guaranteed to be safe. Our algorithm, statistical model predictive shielding (SMPS), uses sampling-based verification and linear systems analysis to perform this check. We prove that SMPS ensures safety with high probability, and empirically evaluate its performance on several benchmarks.**

I. Introduction

Reinforcement learning (RL) has recently had a number of successes, including superhuman performance at Atari games [32] and Go [43]. A promising practical application of DRL is to automatically synthesize deep neural network (DNN) policies for robotics control. In these settings, we typically have a *nominal dynamics model*—i.e., a model of the dynamics that is a good approximation of the true dynamics. For instance, we have nominal dynamics models for cars, quadcopters, walking robots, and grasping robots [46]. Thus, we can use DRL in a simulator based on the nominal dynamics to learn a DNN policy, which can then be deployed on the real robot. For instance, this approach has been applied to (i) solving challenging planning problems, such as object manipulation [5, 22], multi-agent planning [24, 23, 47], and planning from partial observations (e.g., LIDAR or images) [33, 20], (ii) compressing a slow, model-predictive controller (MPC) into a much faster DNN policy [26, 16], and (iii) planning when the dynamics are stochastic or uncertain [14, 15, 25, 53, 34]. In these approaches, DRL is used to learn a deep neural network (DNN) policy in a simulator, which can then be deployed on the real robot.

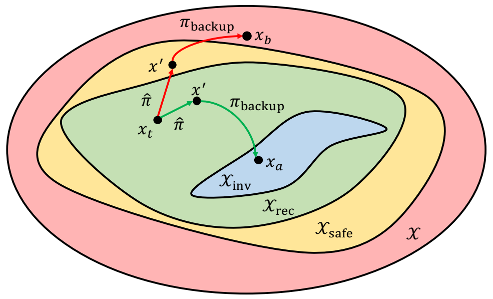
A key challenge is how to ensure safety when using the DNN policy on the real robot—e.g., ensuring that an autonomous car does not drive into obstacles, a walking robot does not fall, or a quadcopter does not crash [17, 4]. In general, it is impossible to ensure safety when no information about the robot dynamics is known initially. However, since we have a nominal dynamics model, we can model the true dynamics as the nominal dynamics plus a stochastic disturbance, and then ensure safety with respect to this model. In this setting, our goal is to provide probabilistic safety guarantees—i.e., ensure the robot is safe with high probability with respect to the randomness in the dynamics.

A number of approaches have been proposed for ensuring safety in this setting. A majority of these approaches focus on proving safety *ahead-of-time*—i.e., before the DNN policy is deployed on the real robot [48, 8, 20]. However, these approaches have a number of shortcomings. For instance, these approaches typically do not provide a way to fix a policy if they fail to prove it is safe. Furthermore, they often scale exponentially with the dimension of the state space, making them computationally intractable for high-dimensional state spaces. One scalable variant of this approach is to "over-approximate" the nominal dynamics (i.e., devise upper and lower bounds on the nominal dynamics for which verification is computationally tractable), but these techniques often fail to prove safety even when the policy is safe since the gap between the upper and lower bounds tends to be large.

An alternative approach is to compose the learned policy $\hat{\pi}$ with a *backup policy* π_{backup} in a way that guarantees safety [30, 41, 37, 18, 2, 10, 13, 3, 49, 50, 38, 56]. The idea is to maintain the invariant that π_{backup} can always be used to ensure safety. In particular, they use $\hat{\pi}$ on the interior of the region where π_{backup} is guaranteed to be safe, and switch to π_{backup} near the boundary of this region. The benefit is that we only need to verify safety for π_{backup}, which can be a simpler policy since it only needs to ensure safety and does not need to perform well. Nevertheless, most existing approaches applicable to general nonlinear dynamics face the same computational challenges even for verifying π_{backup} [18, 2, 10, 38].

A promising approach to ensuring safety is *model predictive shielding (MPS)* [51, 7, 27]. Rather than compute ahead-of-time the region where π_{backup} can ensure safety, this approach performs this check on-the-fly. In particular, it decomposes π_{backup} into (i) an *LQR policy* π_{LQR} used to ensure safety near *safe equilibrium points*, and (ii) a *recovery policy* π_{rec} that tries to transition the system to a safe equilibrium point. Intuitively, safe equilibrium points are states where π_{LQR} can keep the robot safely at rest. Such points exist for most robots of interest—e.g., an autonomous car that is at rest, a quadcopter that has landed, or a walking robot that is standing still. Importantly, the time complexity of MPS is independent of the state and action dimensions (except through $\hat{\pi}$ and π_{backup}).

Existing MPS algorithms have several shortcomings: [51]



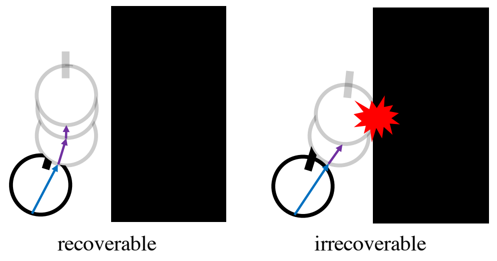


Fig. 1. Left: An overview of SMPS. At state x_t, SMPS simulates the learned policy $\hat{\pi}$ for one step, and then simulates the backup policy π_{backup} for N steps to see if it can drive the robot from $\mathcal{X}_{\text{rec}}$ to $\mathcal{X}_{\text{inv}}$ while staying in $\mathcal{X}_{\text{safe}}$. If so (green trajectory to x_a), it uses $\hat{\pi}$; otherwise (red trajectory to x_b), it uses π_{backup}. Right: Example of recoverable and irrecoverable states for an autonomous car, where the goal is to avoid the black obstacle. We show the simulated trajectory used to check recoverability; the simulation of $\hat{\pi}$ is blue, and the simulation of π_{backup} is purple.

only applies when the nominal dynamics are linear, and [7] only applies when the dynamics are deterministic. The approach proposed by [27], called *robust MPS (RMPS)*, applies more generally to stochatic nonlinear dynamics, but has two shortcomings: (i) it relies on heuristics to compute invariant sets for the LQR policy, and (ii) their strategy for checking π_{backup} is very slow, so they use heuristics to speed up computation that invalidate their safety guarantee.

a) Contributions: We propose a novel algorithm (depicted in Figure 1), *statistical MPS (SMPS)*, that addresses the shortcomings of RMPS as follows: (i) we use techniques from robust control [11] to analyze the safety of π_{LQR}, using Taylor's theorem to bound the approximation error due to linearization, and (ii) we use statistical verification [54, 40, 9], which leverages sampling in conjunction with concentation inequalities, to check whether π_{backup} can ensure safety. We prove that our approach is guaranteed to be safe as long as the nominal dynamics and its derivatives are Lipschitz continuous, and the stochastic disturbances are bounded. Our experiments show that at the same safety confidence level, our approach can be more than $100\times$ faster than RMPS.

b) Example: Consider the cart-pole, where the goal is to move the cart as fast as possible without letting the pole fall over. Here, $\hat{\pi}$ performs well but may be unsafe. Safe equilibrium points $x \in \mathcal{X}_{\text{eq}}$ are states where the pole is upright and the cart is at rest. Then, π_{backup} is composed of (i) π_{LQR}, wich maintains safety in an invariant set $\mathcal{X}_{\text{inv}}$ around $\mathcal{X}_{\text{eq}}$, and (ii) π_{rec} is trained to bring the moving cart-pole to $\mathcal{X}_{\text{inv}}$. SMPS uses $\hat{\pi}$ to move the cart as long as the invariant that π_{rec} can safely bring the cart-pole to $\mathcal{X}_{\text{inv}}$ holds; otherwise, it switches to π_{backup} to ensure safety.

II. Background & Related Work

There are three kinds of safe RL problems: (i) learn a policy while heuristically satisfying safety [1, 52, 35]; however, these approaches do not guarantee safety, (ii) prove that a policy learned in simulation is safe before deploying it on a real robot (our focus), and (iii) guarantee safety during learning. A standard approach to (iii) is to reduce it to (ii) by modeling the true dynamics as the known nominal dynamics plus a Gaussian process (GP) encoding the unknown portion [18, 2, 10]. Then, we can update the GP based on observed state transitions. Finally, we ensure safety with high probability with respect to the GP uncertainty, which is an instance of (ii). Our algorithm can solve (iii) using this approach; see Section IV.

For (ii), given dynamics $f : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, where $\mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the state space and $\mathcal{U} \subseteq \mathbb{R}^{n_U}$ is the action space, and DNN policy $\hat{\pi} : \mathcal{X} \to \mathcal{U}$, the goal is to prove the robot is safe starting from any initial state $x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}$—i.e., the trajectory $x_0, x_1, ...$ defined by $x_{t+1} = f(x_t, \hat{\pi}(x_t))$ satisfies $x_t \in \mathcal{X}_{\text{safe}}$ for all $t \in \mathbb{N}$, where $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$ is a given safe set. For now, we assume f is deterministic, and discuss the stochastic case below. Safety over a finite horizon T can typically be formulated as a constraint satisfaction problem [21]—e.g., if the dynamics are piecewise affine, it can be encoded as an integer program [8]. For an infinite horizon, a standard approach is to prove the existence of an *invariant set*—i.e., $\mathcal{X}_{\text{inv}} \subseteq \mathcal{X}$ such that (i) $f(x) \in \mathcal{X}_{\text{inv}}$ for all $x \in \mathcal{X}_{\text{inv}}$, and (ii) $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{inv}} \subseteq \mathcal{X}_{\text{safe}}$; then, the robot is safe by induction on t.

Invariant sets can be constructed from a *Lyapunov function* or a *control barrier function* [46], which can be computed by solving the Lyapunov equation for linear dynamics [11], a sum-of-squares program for polynomial dynamics [46], or the Hamilton-Jacobi equation for nonlinear dynamics with bounded Lipschitz constant [31, 18, 10]. However, the last approach scales exponentially with the dimension of $\mathcal{X}$.

In some cases, even if it is computationally intractable to compute $\mathcal{X}_{\text{inv}}$ for $\hat{\pi}$, we can do so for a simpler backup policy π_{backup}. Then, we can safely use $\hat{\pi}$ inside $\mathcal{X}_{\text{inv}}$ and use π_{backup} near its boundary—i.e.,

$$\pi_{\text{shield}}(x) = \begin{cases} \hat{\pi}(x) & \text{if } f(x, \hat{\pi}(x)) \in \mathcal{X}_{\text{inv}} \\ \pi_{\text{backup}} & \text{otherwise.} \end{cases}$$

If $\mathcal{X}_{\text{inv}}$ is invariant for π_{backup}, it is also invariant for π_{shield}. This approach is called the *simplex architecture* [41, 42] or *shielding* [3, 56]; it has been used for safe RL [37, 2, 13, 38]. However, computing $\mathcal{X}_{\text{inv}}$ for π_{backup} often remains intractable.

Rather than compute $\mathcal{X}_{\text{inv}}$ ahead-of-time, MPS checks whether $x \in \mathcal{X}_{\text{inv}}$ on-the-fly. Consider a *safe equilibrium point*—i.e., $(\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}} \subseteq \mathcal{X} \times \mathcal{U}$ such that (i) $f(\bar{x}, \bar{u}) = \bar{x}$, and (ii) $\bar{x} \in \mathcal{X}_{\text{safe}}$. At $\bar{x}$, we can use $\bar{u}$ to keep the robot safe for an infinite horizon. We assume that π_{backup} that satisfies $\pi_{\text{backup}}(\bar{x}) = \bar{u}$ for all $(\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}}$. Then, $\mathcal{X}_{\text{eq}} = \{\bar{x} \mid \exists \bar{u} \,.\, (\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}}\}$ is an invariant set for π_{backup}. Thus, in principle, we could use traditional shielding with π_{backup} and $\mathcal{X}_{\text{inv}} = \mathcal{X}_0$.

The shortcoming of this approach is that π_{backup} may have a much larger invariant set; thus, it may unnecessarily switch to π_{backup} even when it is safe to use $\hat{\pi}$. Instead, at $x \in \mathcal{X}$, MPS simulates π_{backup} and checks if π_{backup} safely transitions the robot to a state $\bar{x} \in \mathcal{X}_{\text{eq}}$. Since π_{backup} can ensure safety from $\bar{x}$, it can ensure safety from x as well. More precisely:

Definition II.1. A state $x \in \mathcal{X}$ is *recoverable* (denoted $x \in \mathcal{X}_{\text{rec}}$) if for the trajectory $x_0, x_1, ..., x_N$ generated using π_{backup} from x, there exists $t \in \{0, 1, ..., N\}$ such that (i) $x_t \in \mathcal{X}_{\text{eq}}$, and (ii) $x_0, x_1, ..., x_t \in \mathcal{X}_{\text{safe}}$.

Then, we have the following [7]:

Theorem II.2. *The set $\mathcal{X}_{rec}$ is invariant for π_{shield}.*

When $\hat{\pi}$ is an MPC, this idea is called *dual MPC* [30, 29]; for safe RL with linear dynamics, it is called *model predictive safety certification (MPSC)* [49, 50, 51].

Stochastic disturbances in f complicates this picture, since $\bar{u}$ may no longer keep the robot exactly at $\bar{x}$. Instead, we can typically use an LQR π^z_{LQR} to keep the robot near $\bar{x}$, since the dynamics are typically well-behaved near equilibrium points. Now, suppose that we (i) establish the existence of local invariant sets $\mathcal{X}^z_{\text{inv}}$ for π^z_{LQR} for each $z \in \mathcal{Z}_{\text{eq}}$, and (ii) check that π_{backup} safely transitions the robot to some $\mathcal{X}^z_{\text{inv}}$ with high probability. Then, we can obtain high probability safety guarantees similar to the deterministic case.

For linear dynamics, (i) and (ii) can be addressed using standard techniques [11]. For nonlinear dynamics, [27] uses sampling in conjunction with learning theory bounds to solve (ii); however, their approach is very sample inefficient. They also use sampling to solve (i), but their technique is a heuristic and does not guarantee safety. In contrast, our SMPS algorithm uses a much more sample-efficient strategy to solve (ii), and solves (i) using tools from linear control theory.

Finally, there has been work extending MPS to multi-agent settings [55] and to human-robot interactive settings [19], but both assume the robot dynamics are deterministic.

III. Problem Formulation

Consider a dynamical system $f : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$ with states $\mathcal{X}$, actions $\mathcal{U}$, and disturbances $\mathcal{W}$, which we asssume to be i.i.d. samples $w \sim \mathcal{P}_{\mathcal{W}}$. Given a policy $\pi : \mathcal{X} \to \mathcal{U}$, the trajectory $\zeta(x_0, \pi, \vec{w}) = (x_0, x_1, ...) \in \mathcal{X}^\infty$ generated using π from $x_0 \in \mathcal{X}$ for disturbances $\vec{w} \sim \mathcal{P}^\infty_{\mathcal{W}}$ is $x_{t+1} = f(x_t, \pi(x_t), w_t)$. For now, we assume given a set of invariant states $\mathcal{X}_{\text{inv}}$ from which π_{backup} can ensure safety; both π_{backup} and $\mathcal{X}_{\text{inv}}$ are described in Section V. Our goal is to ensure safety with high probability from initial states $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{inv}}$.

Definition III.1. Given safe states $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$, initial states $\mathcal{X}_0 \subseteq \mathcal{X}_{\text{inv}}$, and $\epsilon \in (0, 1]$, a policy π is *ϵ-safe* if

$$\mathbb{P}_{\vec{w} \sim \mathcal{P}^\infty_{\mathcal{W}}}(\zeta(x_0, \pi, \vec{w}) \subseteq \mathcal{X}_{\text{safe}}) \geq 1 - \epsilon \qquad (\forall x_0 \in \mathcal{X}_0).$$

Given a *learned policy* $\hat{\pi}$, our goal is to construct a *shield policy* π_{shield} that is ϵ-safe. Because our algorithm relies on taking random samples to check safety, safety may fail to hold due to the randomness in these samples—i.e., given $\delta \in (0, 1]$, our algorithm ensures

$$\mathbb{P}_{\vec{\alpha} \sim \mathcal{P}^\infty_{\mathcal{A}}}(\pi_{\text{shield}} \text{ is } \epsilon\text{-safe}) \geq 1 - \delta.$$

where $\alpha \sim \mathcal{P}^t_{\mathcal{A}}$ is the randomness on step t of our algorithm. We make the following assumption:

Assumption III.2. We assume that $\mathcal{X} \subseteq \mathbb{R}^{n_X}$, $\mathcal{U} \subseteq \mathbb{R}^{n_U}$, and $\mathcal{W} \subseteq \mathbb{R}^{n_W}$; that $w \sim \mathcal{P}_{\mathcal{W}}$ is bounded $\|w\|_2 \leq w_{\max}$; and that f, $\nabla_x f$, and $\nabla_u f$ are L-Lipschitz continuous for $\|\cdot\|_2$ on $\mathcal{X}_{\text{safe}}$.

Remark III.3. Our approach extends to time-varying f, $\mathcal{P}_{\mathcal{W}}$, and $\hat{\pi}$, and to $\hat{\pi}$ with latent state. Our approach also extends to reinforcement learning where the dynamics have uncertain parameters with a prior over these parameters such as a Gaussian process [18, 2, 10]; see Appendix C. Our approach similarly extends to partially observed states with a prior over the initial state. Finally, our approach extends to disturbances that are correlated within a trajectory, as long as the sequence of disturbances $\vec{w}$ as a whole is i.i.d. across trajectories.

IV. Statistical MPS

Our SMPS algorithm for computing $\pi_{\text{shield}}(x, t)$, shown in Algorithm 1, combines $\hat{\pi}$ with a *backup policy* π_{backup} that is guaranteed to ensure safety starting from $x_0 \in \mathcal{X}_{\text{inv}}$ for some $\mathcal{X}_{\text{inv}} \subseteq \mathcal{X}_{\text{safe}}$. For now, we assume that such π_{backup} and $\mathcal{X}_{\text{inv}}$ are available; we describe how we construct them in Section V.

Assumption IV.1. $\mathcal{X}_{\text{inv}}$ is an invariant set for π_{backup}.

Then, SMPS checks whether π_{backup} can transition the system to $\mathcal{X}_{\text{inv}}$ with high probability. To this end, given $\vec{w} \sim \mathcal{P}^{N+1}_{\mathcal{W}}$ (where $N \in \mathbb{N}$ is a hyperparameter), it simulates a single step using $\hat{\pi}$ followed by N steps using π_{backup}. If the system reaches $\mathcal{X}_{\text{inv}}$, then the invariant holds for this sample. It uses K i.i.d. samples to check whether the invariant holds with high probability. If so, it uses $\hat{\pi}$; otherwise, it uses π_{backup}.

Definition IV.2. Given $\vec{w} \in \mathcal{W}^{N+1}$, a state $x \in \mathcal{X}_{\text{safe}}$ is *$\vec{w}$-recoverable* (denoted $x \in \mathcal{X}^{\vec{w}}_{\text{rec}}$) if for the trajectory

$$\begin{aligned} x_1 &= f(x, \hat{\pi}(x), w_0) \\ x_{t+1} &= f(x_t, \pi_{\text{backup}}(x_t), w_t) \quad (\forall t \in \{1, ..., N\}), \end{aligned}$$

there exists $\tau \in \{1, ..., N\}$ such that (i) $x_1, ..., x_\tau \in \mathcal{X}_{\text{safe}}$, and (ii) $x_\tau \in \mathcal{X}_{\text{inv}}$.

In other words, x is $\vec{w}$-recoverable if after taking one step using $\hat{\pi}$ and at most N subsequent steps using π_{backup}, the robot reaches $\mathcal{X}_{\text{inv}}$. In Algorithm 1, IsRecSingle returns whether x is $\vec{w}$-recoverable.

Algorithm 1 Statistical model predictive shielding (SMPS). Hyperparameters are $N \in \mathbb{N}$ and $\epsilon, \delta \in (0, 1]$; $K(\epsilon, \delta, t)$ is computed according to (1).

```
procedure SMPS($x, t$)
    if IsRec($x, t$) then
        return $\hat{\pi}(x)$
    else
        return $\pi_{\text{backup}}(x)$
    end if
end procedure
procedure IsRec($x, t$)
    for $i \in \{1, ..., K(\epsilon, \delta, t)\}$ do
        $\vec{w} \sim \mathcal{P}_{\mathcal{W}}^{N+1}$
        if $\neg$IsRecSingle($x, \vec{w}$) then
            return false
        end if
    end for
    return true
end procedure
procedure IsRecSingle($x, \vec{w}$)
    if $x \notin \mathcal{X}_{\text{safe}}$ then
        return false
    end if
    $x \leftarrow f(x, \hat{\pi}(x), w_0)$
    for $\tau \in \{1, ..., N\}$ do
        if $x \in \mathcal{X}_{\text{inv}}$ then
            return true
        else if $x \notin \mathcal{X}_{\text{safe}}$ then
            return false
        end if
        $x \leftarrow f(x, \pi_{\text{rec}}(x), w_\tau)$
    end for
    return false
end procedure
```

Definition IV.3. Given $\epsilon \in (0, 1]$, a state $x \in \mathcal{X}_{\text{safe}}$ is ϵ-*recoverable* (denoted $x \in \mathcal{X}_{\text{rec}}^{\epsilon}$) if

$$\mathbb{P}_{\vec{w} \sim \mathcal{P}_{\mathcal{W}}^{N+1}}(x \in \mathcal{X}_{\text{rec}}^{\vec{w}}) \geq 1 - \epsilon.$$

In other words, x is ϵ-recoverable if it is $\vec{w}$-recoverable with probability at least $1 - \epsilon$ with respect to $\vec{w} \sim \mathcal{P}_{\mathcal{W}}^{N+1}$. In Algorithm 1, IsRec checks whether x is ϵ_t-recoverable. Its result is not guaranteed to be correct, but its result is correct with probability at least $1 - \delta_t$.

Finally, π_{shield} uses $\hat{\pi}$ if it determines that x is ϵ_t-recoverable, and uses π_{backup} otherwise.

It remains to describe how many samples $K(\epsilon, \delta, t)$ are used to check recoverability. Due to the stochastic disturbances, we must increase the recovery probability over time to ensure safety. Let $\epsilon_t = 6\epsilon/((t+1)^2\pi^2)$ and $\delta_t = 6\delta/((t+1)^2\pi^2)$; the identity $\sum_{t=1}^{\infty}(1/t^2) = \pi^2/6$ ensures $\sum_{t=0}^{\infty}\epsilon_t = \epsilon$ and $\sum_{t=0}^{\infty}\delta_t = \delta$. Thus, by a union bound, we can ensure π_{shield} is ϵ-safe with probability at least $1 - \delta$. To achieve the desired safety level at step t, we choose

$$\begin{aligned} K(\epsilon, \delta, t) &= \operatorname*{arg\,min}_{K \in \mathbb{N}} K \quad \text{subj. to} \quad (1 - \epsilon_t)^K \leq \delta_t \\ &= \left\lceil \frac{\log(1/\delta_t)}{\log(1/(1 - \epsilon_t))} \right\rceil, \end{aligned} \tag{1}$$

where the second line follows by solving for K. Intuitively, each check IsRecSingle is a Bernoulli random variable Bernoulli(μ), and IsRec returns that x is ϵ_t-recoverable if none of the K random draws $b \sim$ Bernoulli(μ) are $b =$ false. Then, the left-hand side of the constraint in (1) bounds the probability that the algorithm makes a mistake—i.e., x is recoverable for all K samples but x is *not* ϵ-recoverable. Thus, the constraint in (1) bounds the probability of a mistake by δ. We have the following safety guarantee, which we prove in Appendix A:

Theorem IV.4. $\mathbb{P}_{\vec{\alpha} \sim \mathcal{P}_{\mathcal{A}}^{\infty}}(\pi_{\textit{shield}}$ *is* ϵ*-safe*$) \geq 1 - \delta$.

Remark IV.5. We expect that pratical implementations will use constant values $\epsilon_t = \epsilon$ and $\delta_t = \delta$. This approach does not ensure safety for an infinite horizon, but provides a per-step safety guarantee in line with previous work [51, 27]. Some prior work ensures safety for an infinite horizon [18, 2, 10], but they rely on an assumption that the true dynamics are deterministic.

Remark IV.6. The running time of our algorithm on step t is $O(N \cdot K(\epsilon, \delta, t))$ due to the call to IsRec (assuming $\hat{\pi}$ and π_{backup} run in constant time); furthermore, it is easy to see that $K(\epsilon, \delta, t) = O(\log(1/\delta_t)/\epsilon_t)$. If necessary, we can add a time out and have IsRec return false if it runs out of time.

V. Backup Policy

Our backup policy relies on *safe equilibrium points* of f where the robot remains safely at rest—i.e., points $z = (\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}} \subseteq \mathcal{X} \times \mathcal{U}$ such that (i) $\bar{x} = f(\bar{x}, \bar{u}, 0)$, and (ii) $\bar{x} \in \mathcal{X}_{\text{safe}}$. Then, π_{backup} consists of (i) a *recovery policy* π_{rec} that attempts to transition the robot to a safe equilibrium point z, and (ii) a family of *LQR policies* π_{LQR}^{z} for each $z \in \mathcal{Z}_{\text{eq}}$ that, if the robot is already near z, keeps it near z.

Recovery policy We begin by describing π_{rec}, which is designed to attempt to transition the robot to a safe equilibrium point. We note that our algorithm ensures safety regardless of π_{rec}; instead, a better choice of π_{rec} improves the performance of π_{shield}. We use RL to π_{rec}. First, we use initial state distribution d_{rec}—to sample $x \sim d_{\text{rec}}$, we (i) sample an initial state $x_0 \sim d_0$, where d_0 is a distribution over $\mathcal{X}_0$, (ii) sample a time horizon $t \sim$ Uniform($\{0, ..., T-1\}$), where $T \in \mathbb{N}$ is a hyperparameter, (iii) compute the trajectory $x_0, x_1, ..., x_t$ obtained using $\hat{\pi}$ from x_0, and (iv) reject if $x_t \notin \mathcal{X}_{\text{safe}}$ and take $x = x_t$ otherwise. Second, we use reward $r_{\text{rec}}(x, u) = -\|x - \bar{x}\|^2$, where $(\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}}$ is such that $\bar{x}$ is closest to x. Then, we use RL to learn

$$\pi_{\text{rec}} = \operatorname*{arg\,max}_{\pi} \mathbb{E}_{x_0 \sim d_{\text{rec}}, \vec{w} \sim \mathcal{P}_{\mathcal{W}}^{\infty}} \left(\sum_{t=0}^{\infty} \gamma^t \cdot r_{\text{rec}}(x_t, u_t) \right),$$

where $u_t = \pi_{\text{rec}}(x_t)$ and $x_{t+1} = f(x_t, u_t, w_t)$.

a) LQR policy: Next, we describe π^z_{LQR}. Intuitively, near $z = (\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}}$, f is closely approximated by its linearization, so we can use a *linear-quadratic regulator (LQR)* to keep the robot near z [11]. In particular, consider the approximation

$$\begin{aligned} f(x,u,w) &\approx f(\bar{x},\bar{u},0) + A(x-\bar{x}) + B(u-\bar{u}) \\ &= \bar{x} + A(x-\bar{x}) + B(u-\bar{u}) \end{aligned} \tag{2}$$

of f, where $A = \nabla_x f(\bar{x},\bar{u},0)$ and $B = \nabla_u f(\bar{x},\bar{u},0)$; note that we have both linearized f and ignored the disturbance w. Then, we compute the LQR K for the linear dynamics A, B (we choose the costs Q and R to be the identity). As long as the error in (2) is small, then we can keep the robot near z using the policy

$$\pi^z_{\text{LQR}}(x) = \bar{u} + K(x - \bar{x}),$$

in which case (2) becomes

$$f(x, \pi^z_{\text{LQR}}(x), w) \approx \bar{x} + (A + BK)(x - \bar{x}). \tag{3}$$

Next, recall that we want to use π^z_{LQR} to keep the robot safely near z. To determine when we can use π_{LQR}, we additionally need to compute the region where π_{LQR} can be used to ensure safety. In particular, we compute an invariant set $\mathcal{X}^z_{\text{inv}}$ for π^z_{LQR}—i.e., (i) $\pi^z_{\text{LQR}}(x) \in \mathcal{X}^z_{\text{inv}}$ for all $x \in \mathcal{X}^z_{\text{inv}}$, and (ii) $\mathcal{X}^z_{\text{inv}} \subseteq \mathcal{X}_{\text{safe}}$.

We consider $\mathcal{X}^z_{\text{inv}}$ of the form $\mathcal{X}^z_{\text{inv}} = B_2(\bar{x}, r^z)$—i.e., the L_2 ball around $\bar{x}$ of some radius $r^z \in \mathbb{R}_{>0}$ to be determined. Then, (i) is equivalent to

$$\|(A + BK)(x - \bar{x}) + \Delta_{x,w}\|_2 \le r^z \tag{4}$$

for all $x \in B_2(\bar{x}, r^z)$, where

$$\Delta_{x,w} = f(x, \pi^z_{\text{LQR}}(x), w) - \bar{x} - (A + BK)(x - \bar{x})$$

is the approximation error in (3). Now, note that if (i) $\|A + BK\|_2 \le \alpha$ for some $\alpha \in [0, 1)$, and (ii) $r^z \ge \|\Delta_{x,w}\|_2/(1 - \alpha)$, then (4) holds—in particular,

$$\|(A + BK)(x - \bar{x}) + \Delta_{x,w}\|_2 \le \alpha r^z + \|\Delta_{x,w}\|_2 \le r^z$$

for all $x \in B_2(\bar{x}, r^z)$. Thus, it suffices to choose the largest r^z such that (i) $r^z \ge \|\Delta_{x,w}\|_2/(1-\alpha)$ for all $x \in B_2(\bar{x}, r^z)$ and $w \in \mathcal{W}$, and (ii) $B_2(\bar{x}, r^z) \subseteq \mathcal{X}_{\text{safe}}$.

To compute such an r^z, we need to bound $\Delta_{x,w}$.

Lemma V.1. *For any $r \in \mathbb{R}_{>0}$, we have*

$$\|\Delta_{x,w}\|_2 \le \Delta_{max}(r) = Lw_{max} + \frac{L(1 + \|K\|_2)n_X^{3/2} r^2}{2}$$

for all $x \in B_2(\bar{x}, r)$ and $w \in \mathcal{W}$.

This result is follows from Taylor's theorem and Assumption III.2; see Appendix B for a proof. Note that the bound on $\Delta_{x,w}$ depends on a parameter $r \in \mathbb{R}_{>0}$—in particular, it holds for all $x \in B_2(\bar{x}, r)$ (larger r is better), but its magnitude depends r (smaller r is better). Based on this result, it suffices to choose

$$r^z = \max_{r \in \mathbb{R}_{>0}} \Delta_{\text{max}}(r)/(1 - \alpha) \text{ subj. to } B_2(\bar{x}, r) \subseteq \mathcal{X}_{\text{safe}}.$$

Then, we take $\mathcal{X}^z_{\text{inv}} = B_2(\bar{x}, r^z)$; if no solution exists, we take $\mathcal{X}^z_{\text{inv}} = \varnothing$. Based on our discussion, we have:

Lemma V.2. *$\mathcal{X}^z$ is an invariant set for π^z_{LQR}.*

We briefly comment on the condition $\|\tilde{A}\|_2 \le \alpha$, where $\tilde{A} = A + BK$. Note that this condition is basis dependent; if we change basis to the eigenbasis of $\tilde{A}$ (assuming it exists), then $\|\tilde{A}\|_2$ equals the maximum eigenvalue $\rho(\tilde{A})$ of $\tilde{A}$. In this case, the condition is equivalent to $\rho(\tilde{A}) \le \alpha$ for some $\alpha \in [0, 1)$. This condition is equivalent to stability of the discrete-time linear dynamical system $\tilde{x}_{t+1} = \tilde{A}\tilde{x}_t$ [11]. Intuitively, $z \in \mathcal{Z}_{\text{eq}}$ satisfies this condition as long as control actions can be taken to keep the system at $\bar{x}$. Practical examples include holding the brakes in an autonomous car or keeping a landed quadcopter at rest. In addition, for the inverted pendulum and the cart-pole, keeping the pole in the upright position satisfies this condition; similarly, a typical walking robot in the upright position also satisfies this condition.

b) Backup policy: Finally, our backup policy is

$$\pi_{\text{backup}}(x) = \begin{cases} \pi^z_{\text{LQR}} & \text{if } x \in \mathcal{X}^z_{\text{inv}} \\ \pi_{\text{rec}} & \text{otherwise.} \end{cases}$$

If multiple $z \in \mathcal{Z}_{\text{eq}}$ satisfy $x \in \mathcal{X}^z_{\text{inv}}$, it chooses $z = (\bar{x}, \bar{u})$ such that $\bar{x}$ is closest to x. Finally, define

$$\mathcal{X}_{\text{inv}} = \bigcup_{z \in \mathcal{Z}_{\text{eq}}} \mathcal{X}^z_{\text{inv}}.$$

Then, Lemma V.2 implies the following result, which says that π_{backup} and $\mathcal{X}_{\text{inv}}$ satisfy Assumption IV.1:

Theorem V.3. *$\mathcal{X}_{inv}$ is an invariant set for π_{backup}.*

VI. Experiments

We evaluate SMPS on several tasks, showing it ensures safety in a scalable way. All experiments are run on a 2.9 GHz Intel Core i9 CPU with 32GB memory.

a) Tasks: First, we consider the cart-pole [6] with continuous actions. Our goal is for the cart to have a target velocity of $v_0 = 0.1$ (i.e., move to the right). The safety constraint is that the pole angle does not exceed $\theta_{\text{max}} = 0.15$ rad from upright. The initial state distribution is Uniform($[-0.05, 0.05]^4$). Second, we consider a bicycle [45, 36], which has 4D states (x, y, v, θ), where (x, y) is the front of the car, v is the velocity, and θ is the heading, and a 2D actions (a, ψ), where a is the acceleration and ψ is the steering angle. We assume that $|a| \le a_{\text{max}} = 0.25$. The goal is to get from the initial state $(0, 0, 0, 0)$ to the target $x = 1$. The bicycle must avoid two obstacles, which have x positions 0.4 and 0.7, y positions sampled i.i.d. from Uniform($[-0.05, 0.05]$), and radius 0.05.

b) Reinforcement learning: We use backpropagation-through-time (BPTT)—i.e., backpropagate through both the policy and the dynamics—to learn $\hat{\pi}$ and π_{rec}. Each policy is a single-layer neural network with 200 hidden units and ReLU activations. For the bicycle, we include the y positions of the obstacles as inputs. For each task, we train using a time

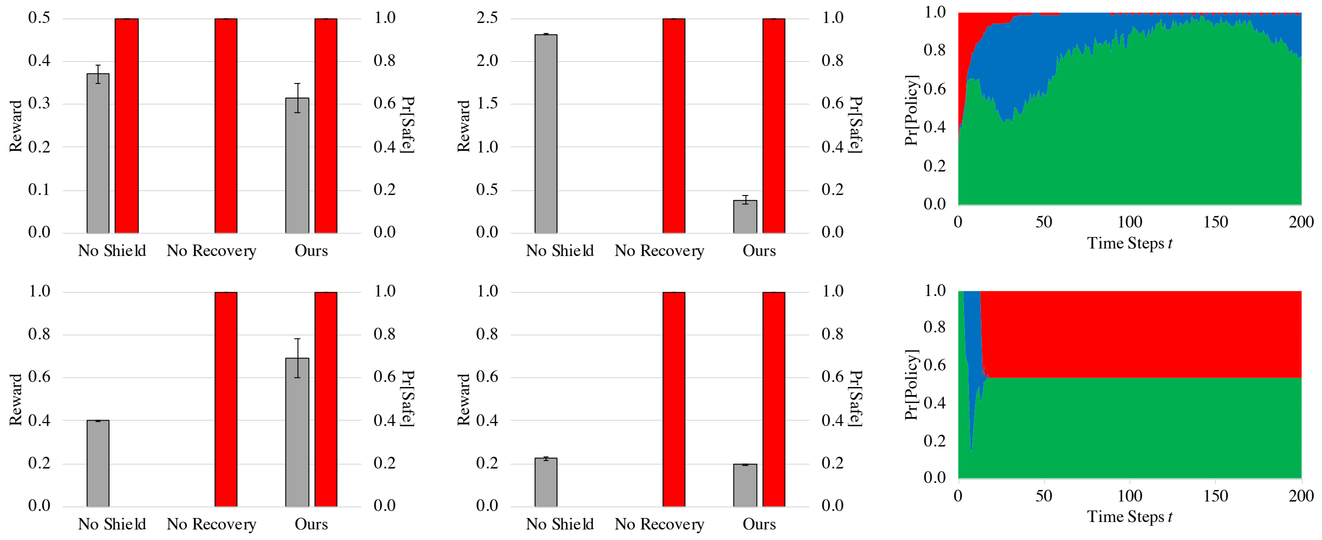


Fig. 2. For cart-pole (top) and bicycle (bottom): Reward (gray) and safety probability (red) for original (left) and modified (middle) environments, and probability of using of $\hat{\pi}$ (green), π_{rec} (blue), and π_{LQR} (red) as a function of t on the original cart-pole and bicycle environments (right). For modified cart-pole, we threshold reward at 2.0; "No Shield" achieves larger reward using unsafe behavior. We show means (all) and standard errors (left, middle) over 100 random rollouts.

horizon $T = 200$ steps and a discount factor $\gamma = 0.99$. We use ADAM with a learning rate tuned using cross validation.

c) Shielding: For the cart-pole, the equilibrium points are $((x, 0, 0, 0), 0)$—i.e., keep the cart still and the pole upright. For the bicycle, they are $((x, y, 0, \theta), (0, 0))$. We choose recovery horizon $N = 100$. We choose $\epsilon = \delta = 0.95$, so we need $K = 59$ samples per step.

d) Modified problems: Changes in the planning problem—e.g., different configurations of obstacles, longer time horizon, or increased noise—can cause $\hat{\pi}$ to become unsafe to use, since it is tailored to perform well in the original problem. To demonstrate how MPS can ensure safety in the face of such changes, we consider modifications to the cart-pole and bicycle. First, for the cart-pole, we perform two changes: (i) we increase the time horizon—though $\hat{\pi}$ and π_{rec} are trained with a time horizon of $T = 200$, we use them to control the robot for a time horizon of $T = 1000$, and (ii) we add i.i.d. Gaussian random noise to the state at each step. Second, for the bicycle, we also perform two changes: (i) we enlarge the obstacles to have radius 0.2, compared to 0.05 originally, and (ii) we add i.i.d. Gaussian random noise to the action at each step.

e) Results: In Figure 2 (left, middle), we show both the reward achieved for cart-pole, bicycle, and their modifications. We show results for (i) $\hat{\pi}$ ("No Shield"), (ii) π_{shield} without π_{rec}—i.e., using $N = 0$ ("No Recovery"), and (iii) π_{shield} ("Ours"). The reward shown is the actual performance—z for cart-pole (i.e., distance traveled by the cart), and x for the bicycle (i.e., distance traveled towards the target)—rather than the shaped reward used to learn $\hat{\pi}$. The rewards are the ones obtained either at the end of a rollout, or just before an unsafe state is reached. While our shielded policy (with $N = 100$) sometimes achieves reduced reward compared to $\hat{\pi}$, it always achieves perfect safety. We note that $\hat{\pi}$ can achieve higher reward than π_{shield} in cases where the reward function does not measure safety in any way.

We also show the safety probability—i.e., the probability that a random rollout is safe for the entire rollout. All of our shielded policies (i.e., both $N = 0$ and $N = 100$) achieve perfect safety. The learned policy $\hat{\pi}$ achieves good safety probability on the original cart-pole. Surprisingly, it does not perform well for the original bicycle environment; we observed that it is safe for the majority of each rollout, but invariably has an accident at some point. As expected, it performs very poorly on both modified environments, since training did not account for these modifications.

f) Policy usage: In Figure 2 (right), for π_{shield} with $N = 100$, we show the probability of using $\hat{\pi}$, π_{rec}, and π_{LQR} as a function of time t, on the original cart-pole and bicycle environments. For cart-pole, π_{shield} initially uses π_{LQR} to upright the pole, and then proceeds to use a combination of $\hat{\pi}$ and π_{rec}. For the modified environment (plot omitted), π_{shield} inevitably switches to using π_{LQR} only—$\hat{\pi}$ acts pathologically (and unsafely) for states with large z, since it was not trained on these states. For the bicycle, in about half the rollouts, π_{shield} switches to π_{LQR} and does not make further progress, likely because the obstacle was blocking the way. For the remaining rollouts, π_{shield} uses $\hat{\pi}$ for most of the rollout. For the modified environment (plot omitted), π_{shield} almost always uses π_{LQR}.

g) Running time: We study the running time of π_{shield}—i.e., how long it takes to compute a single action $u = \pi_{\text{shield}}(x)$. In Figure 3 (middle), we show the average running time varies as a function of the number of samples K, on cart-pole. As expected, the running time scales linearly with K.

In Figure 3 (right), we show how the average running time varies as a function of the recovery steps N, on cart-pole.

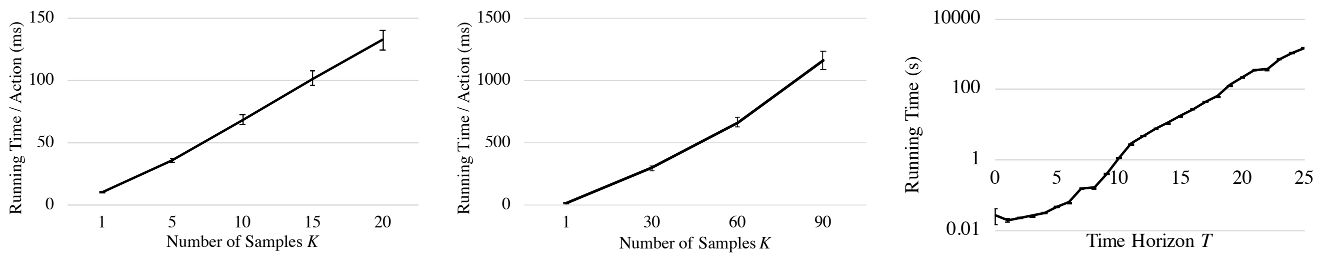


Fig. 3. Left: Time per action on modified cart-pole for π_{shield} as a function of $K \in \{1, 5, 10, 15, 20\}$. Middle: Time per action (black) and reward (green) on modified cart-pole for π_{shield} as a function of $N \in \{0, 25, 50, 75, 100\}$. Right: Time to verify the cart-pole policy from [8] as a function of T. We show means and standard errors over 10 random rollouts (left, middle) or 4 runs (right).

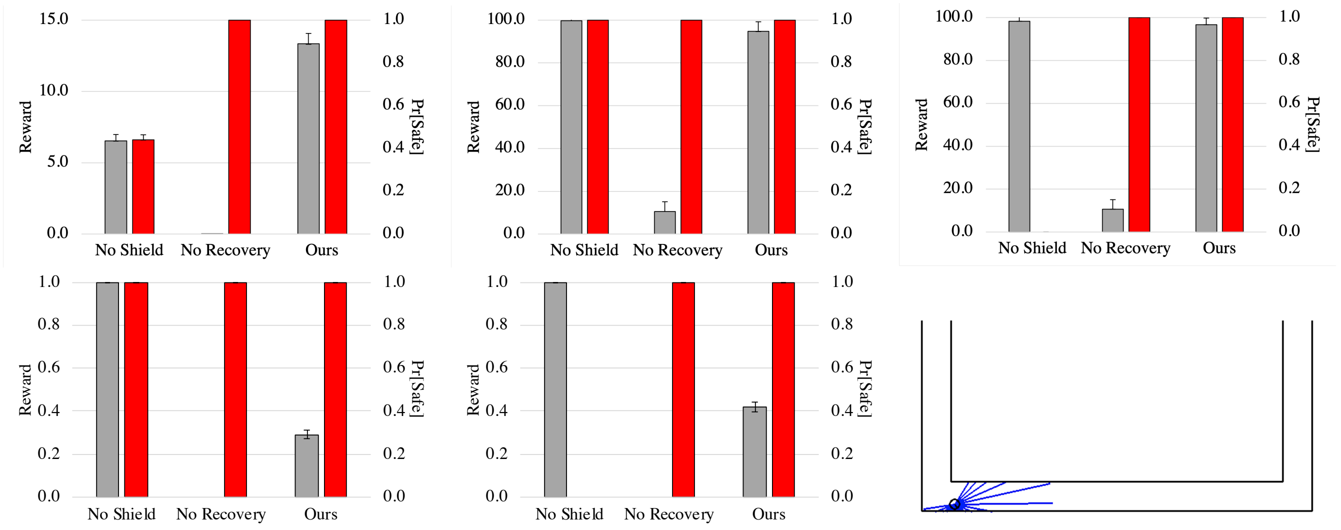


Fig. 4. Reward (gray) and safety probability (red) for the half-cheetah (top left), the original ant (top middle), the modified ant (top right), the original bicycle with LIDAR observations (bottom left), the fast bicycle with LIDAR observations (bottom middle). We also show a visualization of the LIDAR bicycle; the bicycle is the black circle, the black lines are walls, and the blue lines are LIDAR rays.

As expected, for $N = 100$, the running time is about $100\times$ the running time of $\hat{\pi}$ (36.0ms vs. 0.2ms), since it simulates the dynamics for 100 steps. We believe this overhead is an acceptable cost for guaranteeing safety. The worst case running time is linear in N, but can be sublinear if π_{rec} reaches an invariant state in fewer than N steps.

h) Need for a recovery policy: Our results demonstrate the importance of using π_{rec}—when $N = 0$, π_{shield} only uses π_{LQR}, and makes no progress. In Figure 3 (right), we additionally show how reward varies as a function of the recovery steps N for modified cart-pole. There is a large improvement even for $N = 25$; performance then levels off, with $N = 75$ and $N = 100$ achieving similar reward.

i) Comparison to RMPS: We compare our approach to RMPS [27], which uses robust optimization to compute π_{rec} at each step, focusing on the cart-pole. They provide similar guarantees to our per-step guarantees—i.e., given $\epsilon, \delta \in \mathbb{R}_{>0}$, it guarantees that the system is safe on the next step with probability at least $1 - \epsilon$ with respect to the stochastic disturbances and at least $1 - \delta$ with respect to the randomness of the algorithm. For $\epsilon = \delta = 0.1$, we estimate that their algorithm takes 11.88 seconds per step on average, whereas ours takes 0.15 seconds per step on average. For $\epsilon = \delta = 0.05$, theirs takes 56.92 seconds per step whereas ours takes 0.39 seconds per step. For $\epsilon = \delta = 0.01$, theirs takes 1958.37 seconds per step whereas ours takes 3.03 seconds per step.

Our technique outperforms theirs because we use samples directly to check safety. In contrast, they use samples to estimate reachable sets, and use bounds from statistical learning theory to obtain guarantees; then, they use these reachable sets to check safety. Formally, they require $\tilde{O}(n_X/\epsilon^2)$ samples per step, whereas we only require $\tilde{O}(1/\epsilon)$ samples per step.

j) Comparison to ahead-of-time verification: We compare our approach to ahead-of-time verification. One challenge is that these techniques typically only perform verification for a bounded state space or for a bounded time horizon T. In Figure 3, we show how the running time of a state-of-the art verification algorithm scales as a function of T for verifying a cart-pole policy [8]. The y-axis is log-scale—thus, verification is exponential in T. Even for $T = 25$, it takes about 24 minutes to perform verification. Though this computation is offline, it quickly becomes intractable for large T. As a rough estimate, the running time grows $10^2\times$ from $T = 10$ to $T = 20$; extrapolating this trend, the running time for $T = 200$ would be over 10^{30} years. In contrast, our approach not only ensures safety for an unbounded horizon, but is also substantially more computationally feasible.

k) LIDAR bicycle: We consider a variant of the bicycle based on [20], with LIDAR observations and driving in a hallway (visualized in Figure 4). The fact that the environment is partially observed complicates shielding. We use a strategy where we always ensure that the bicycle can come to a stop in the region visible based on the LIDAR readings. We consider both the original bicycle environment used for training, as well as an environment where the maximum speed of the bicycle is increased. We show results in Figure 4 (top); the trends are the same as for the cart-pole and bicycle.

l) MuJoCo: We consider the MuJoCo half-cheetah and ant environments [12]. For the half-cheetah, the safety property is that the head of the cheetah should not fall below a certain height, and the reward is to move as far as possible from the starting point. For the ant, the safety property is that the ant should not collide with an obstacle. We do not have the closed-form dynamics, so we cannot analytically bound the Lipschitz constant; instead, we use a sampling-based heuristic to estimate $\mathcal{X}_{\text{inv}}$ for the LQR around the origin [28, 27]. For the half-cheetah, the learned policy $\hat{\pi}$ was not always safe even for the original environment, so we did not consider a modified environment. For the ant, we consider a modified environment where the obstacle position has been changed. We show results in Figure 4.

VII. Conclusion

We have proposed a novel algorithm for safe reinforcement learning in the general setting of stochastic nonlinear dynamics. Our approach uses sampling to verify safety, enabling it to scale orders of magnitude better than prior approaches. We leave much room for future work—e.g., further improving the scalability of our approach, and demonstrating its ability to ensure safety on real robotics systems.

References

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *ICML*, 2017.

[2] Anayo K Akametalu, Shahab Kaynama, Jaime F Fisac, Melanie Nicole Zeilinger, Jeremy H Gillula, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *CDC*, pages 1424–1431. Citeseer, 2014.

[3] Mohammed Alshiekh, Roderick Bloem, Rudiger Ehlers, Bettina Konighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI*, 2018.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[5] Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *arXiv preprint arXiv:1808.00177*, 2018.

[6] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[7] Osbert Bastani. Safe planning via model predictive shielding. *arXiv preprint arXiv:1905.10691*, 2019.

[8] Osbert Bastani, Pu Yewen, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in neural information processing systems*, 2018.

[9] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Probabilistic verification of fairness properties via concentration. *Proceedings of the ACM on Programming Languages*, 3(OOPSLA):1–27, 2019.

[10] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems*, pages 908–918, 2017.

[11] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[13] Yinlam Chow, Ofir Nachum, and Edgar Duenez-Guzman. A lyapunov-based approach to safe reinforcement learning. In *NeurIPS*, 2018.

[14] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[15] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[16] Robin Deits, Twan Koolen, and Russ Tedrake. Lvis: Learning from value function intervals for contact-aware robot controllers. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7762–7768. IEEE, 2019.

[17] Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *JMLR*, 2015.

[18] Jeremy H. Gillula and Claire J. Tomlin. Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor. In *ICRA*, 2012.

[19] Jeevana Priya Inala, Jason Ma, Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. Safe human-interactive control modulo fault. 2021. URL https://jinala.github.io/assets/papers/safehumancontrol.pdf.

[20] Radoslav Ivanov, James Weimer, Rajeev Alur, George J. Pappas, and Insup Lee. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *HSCC*, 2019.

[21] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and

Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.

[22] Monroe Kennedy, Karl Schmeckpeper, Dinesh Thakur, Chenfanfu Jiang, Vijay Kumar, and Kostas Daniilidis. Autonomous precision pouring from unknown containers. *IEEE Robotics and Automation Letters*, 4(3):2317–2324, 2019.

[23] Arbaaz Khan, Ekaterina Tolstaya, Alejandro Ribeiro, and Vijay Kumar. Graph policy gradients for large scale robot control. In *CoRL*, 2019.

[24] Arbaaz Khan, Chi Zhang, Shuo Li, Jiayue Wu, Brent Schlotfeldt, Sarah Y Tang, Alejandro Ribeiro, Osbert Bastani, and Vijay Kumar. Learning safe unlabeled multi-robot planning with motion constraints. In *IROS*, 2019.

[25] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

[26] Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[27] Shuo Li and Osbert Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *ICRA*, 2020.

[28] D Limon, I Alvarado, T Alamo, and EF Camacho. Robust tube-based mpc for tracking of constrained linear systems with additive disturbances. *Journal of Process Control*, 20(3):248–260, 2010.

[29] David Q Mayne, James B Rawlings, Christopher V Rao, and Pierre OM Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[30] Hanna Michalska and David Q Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE transactions on automatic control*, 38(11):1623–1633, 1993.

[31] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[33] William H Montgomery and Sergey Levine. Guided policy search via approximate mirror descent. In *Advances in Neural Information Processing Systems*, pages 4008–4016, 2016.

[34] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019.

[35] Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. In *Advances in Neural Information Processing Systems*, pages 7553–7563, 2019.

[36] Romain Pepy, Alain Lambert, and Hugues Mounier. Path planning using a dynamic vehicle model. In *2006 2nd International Conference on Information & Communication Technologies*, volume 1, pages 781–786. IEEE, 2006.

[37] Theodore J. Perkins and Andrew G. Barto. Lyapunov design for safe reinforcement learning. *JMLR*, 2002.

[38] Dung Phan, Nicola Paoletti, Radu Grosu, Nils Jansen, Scott A Smolka, and Scott D Stoller. Neural simplex architecture. *arXiv preprint arXiv:1908.00528*, 2019.

[39] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[40] Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *International Conference on Computer Aided Verification*, pages 266–280. Springer, 2005.

[41] Danbing Seto, Bruce Krogh, Lui Sha, and Alongkrit Chutinan. The simplex architecture for safe online control system upgrades. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, volume 6, pages 3504–3508. IEEE, 1998.

[42] Lui Sha. Using simplicity to control complexity. *IEEE Software*, (4):20–28, 2001.

[43] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529 (7587):484, 2016.

[44] Elias M Stein and Rami Shakarchi. *Real analysis: measure theory, integration, and Hilbert spaces*. Princeton University Press, 2009.

[45] Saied Taheri. An investigation and design of slip control braking systems integrated with four-wheel steering. 1992.

[46] Russ Tedrake. *Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation*. 2018. URL http://underactuated.mit.edu/.

[47] Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *CoRL*, 2019.

[48] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. *arXiv preprint arXiv:1804.02477*, 2018.

[49] Kim P Wabersich and Melanie N Zeilinger. Linear model predictive safety certification for learning-based control. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 7130–7135. IEEE, 2018.

[50] Kim P Wabersich and Melanie N Zeilinger. Safe exploration of nonlinear dynamical systems: A predictive safety filter for reinforcement learning. *arXiv preprint arXiv:1812.05506*, 2018.

[51] Kim P Wabersich, Lukas Hewing, Andrea Carron, and

Melanie N Zeilinger. Probabilistic model predictive safety certification for learning-based control. *arXiv preprint arXiv:1906.10417*, 2019.

[52] Min Wen and Ufuk Topcu. Constrained cross-entropy method for safe reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7450–7460, 2018.

[53] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[54] Håkan LS Younes and Reid G Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *International Conference on Computer Aided Verification*, pages 223–235. Springer, 2002.

[55] Wenbo Zhang, Osbert Bastani, and Vijay Kumar. Mamps: Safe multi-agent reinforcement learning via model predictive shielding. *arXiv preprint arXiv:1910.12639*, 2019.

[56] He Zhu, Zikang Xiong, Stephen Magill, and Suresh Jagannathan. An inductive synthesis framework for verifiable reinforcement learning. In *PLDI*, 2019.

Appendix A
Proof of Theorem IV.4

First, we have the following:

Lemma A.1. *If IsRecoverable(x,t) returns true, then*

$$\mathbb{P}_{\alpha\sim\mathcal{P}_{\mathcal{A}}^{t}}(x\in\mathcal{X}_{rec}^{\epsilon_t})\geq 1-\delta_t,$$

where $\alpha\sim\mathcal{P}_{\mathcal{A}}^{t}$ is the randomness in the $K(\epsilon,\delta,t)$ samples $\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{N+1}$ taken by IsRecoverable.

Proof: In the execution of IsRecoverable(x,t), let $\vec{w}_i\sim\mathcal{P}_{\mathcal{W}}^{N+1}$ be the i.i.d. sample drawn on the ith iteration of the for-loop, where $i\in\{1,...,K\}$ and $K=K(\epsilon,\delta,t)$. Then, the value $b_i=\text{IsRecoverableSingle}(x,\vec{w})\in\{0,1\}$ is an i.i.d. Bernoulli random variable (where we represent true by 1 and false by 0). Note that

$$p=\mathbb{P}_{\vec{w}_i\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(b_i=0)=\mathbb{P}_{\vec{w}_i\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(x\notin\mathcal{X}_{\text{rec}}^{\vec{w}}),$$

and the goal of IsRecoverable(x,t) is to check whether $p<\epsilon_t$.

We need to show that the probability IsRecoverable(x,t) returns true given that $p\geq\epsilon_t$ is bounded by δ_t. To this end, assume that $p\geq\epsilon_t$. Note that IsRecoverable(x,t) returns true only if $b_i=1$ for every $i\in\{1,...,K\}$. We have

$$\begin{aligned}
&\mathbb{P}_{\vec{w}_1,...,\vec{w}_K\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(\text{IsRecoverable}(x,t)=\text{true})\\
&=\mathbb{P}_{\vec{w}_1,...,\vec{w}_K\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(\forall i\in\{1,...,K\}\ .\ b_i=1)\\
&=\prod_{i=1}^{K}\mathbb{P}_{\vec{w}_i\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(b_i=1)\\
&=(1-p)^K\\
&\leq(1-\epsilon_t)^K\\
&<\delta_t.
\end{aligned}$$

The claim follows. ■

Now, our proof of safety proceeds in two steps. First, we consider the following *ideal shield policy* π_{shield}^{*}, defined by

$$\pi_{\text{shield}}^{*}(x,t)=\begin{cases}\hat{\pi}(x) & \text{if } x\in\mathcal{X}_{\text{rec}}^{\epsilon_t}\\ \pi_{\text{backup}}(x) & \text{otherwise.}\end{cases}$$

This policy is the variant of π_{shield} where IsRecoverable never makes mistakes, so any unsafety is due to randomness in the transitions. First, we show that $\phi_{\text{safe}}^{\epsilon}(\pi_{\text{shield}}^{*})$ holds:

Lemma A.2. *Assume we are using policy π_{shield}^{*}. For any $t\in\mathbb{N}$ and any $x_t\in\tilde{\mathcal{X}}_{rec}^{\epsilon,t}=\mathcal{X}_{rec}^{\epsilon_t}\cup\mathcal{X}_{inv}$, we have*

$$\mathbb{P}_{\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{N+1}}(\phi_{rec}^{\vec{w},t})\geq 1-\epsilon_t,$$

where $\phi_{rec}^{\vec{w},t}$ is the event that there exists $\tau\in\{1,...,N\}$ such that (i) $x_{t+1},...,x_{t+\tau}\in\mathcal{X}_{safe}$, and (ii) $x_{t+\tau}\in\tilde{\mathcal{X}}_{rec}^{t+\tau}$, assuming the disturbances are $\vec{w}$.

Proof: First, consider the case $x_t\in\mathcal{X}_{\text{inv}}\setminus\mathcal{X}_{\text{rec}}^{\epsilon_t}$. In this case, π_{shield}^{*} chooses to use π_{backup}. Since $\mathcal{X}_{\text{inv}}$ is invariant for π_{backup}, we have $x_{t+1}\in\mathcal{X}_{\text{inv}}$, so $\phi_{\text{rec}}^{\epsilon,t}$ holds.

Second, consider the case $x_t\in\mathcal{X}_{\text{rec}}^{\epsilon_t}$. It suffices to show that $\phi_{\text{rec}}^{\vec{w},t}$ always holds on the event $x_t\in\mathcal{X}_{\text{rec}}^{\vec{w}}$; by the definition of $\mathcal{X}_{\text{rec}}^{\epsilon_t}$, this event holds with probability at least $1-\epsilon_t$, so the claim follows.

Consider the smallest $\tau\in\{1,...,N\}$ (if any) such that π_{shield}^{*} chooses to use $\hat{\pi}$ at state $x_{t+\tau}$. Then, it must be the case that $x_{t+\tau}\in\mathcal{X}_{\text{rec}}^{\epsilon_{t+\tau}}$. Furthermore, since π_{shield}^{*} used π_{backup} at each state $x_{t+\sigma}$ where $\sigma\in\{1,...,\tau-1\}$, and since we have assumed $x_t\in\mathcal{X}_{\text{rec}}^{\vec{w}}$, it follows that $x_{t+\sigma}\in\mathcal{X}_{\text{safe}}$ for each σ. Thus, $\phi_{\text{rec}}^{\vec{w},t}$ holds.

If no such τ exists, then π_{shield}^{*} chooses to use π_{backup} in every state $x_{t+\tau}$. Since we have assumed $x_t\in\mathcal{X}_{\text{rec}}^{\vec{w}}$, it follows that there exists $\tau\in\{1,...,N\}$ such that $x_{t+1},...,x_{t+\tau}\in\mathcal{X}_{\text{safe}}$ and $x_{t+\tau}\in\mathcal{X}_{\text{inv}}$. Thus, $\phi_{\text{rec}}^{\vec{w},t}$ holds. The claim follows. ■

In other words, Lemma A.2 says that if we are currently in $\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,t}$, then with probability at least $1-\epsilon_t$, we safely return to $x_t\in\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,t+\tau}$ for some $\tau\in\{1,...,N\}$. If $\epsilon_t=\epsilon$ were constant, then Lemma A.2 would say that $\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon}=\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,t}$ is a *probabilistically safe recurrent set* for π_{shield}^{*}—i.e., if $x_t\in\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon}$, then with probability at least $1-\epsilon$, we safely return to $\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon}$ in within $N+1$ steps. Lemma A.2 accounts for the fact that $\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon}$ shrinks over time.

Our next result enables us to deduce safety from existence of safely recurrent sets:

Lemma A.3. *Consider an arbitrary sequence of states $x_0,x_1,....$ Suppose that there exists sets $\mathcal{X}_{rec}^{t}\subseteq\mathcal{X}_{safe}$ indexed by $t\in\mathbb{N}$ such that $x_0\in\mathcal{X}_{rec}^{0}$, and for each $t\in\mathbb{N}$,*

$$(x_t\notin\mathcal{X}_{rec}^{t})\vee\phi_{rec}^{t}\qquad(5)$$

holds, where ϕ_{rec}^{t} is the event that there exists $\tau\in\{1,...,N\}$ such that (i) $x_{t+1},...,x_{t+\tau}\in\mathcal{X}_{safe}$, and (ii) $x_{t+\tau}\in\mathcal{X}_{rec}^{t+\tau}$. Then, $x_t\in\mathcal{X}_{safe}$ for all $t\in\mathbb{N}$.

Proof: Suppose to the contrary that $x_t\notin\mathcal{X}_{\text{safe}}$ for some $t\in\mathbb{N}$. Let $t'\in\{0,1,...,t-1\}$ be the largest time step such that $x_{t'} \in \mathcal{X}_{\text{rec}}^{t'}$. Such a t' must exist since we have assumed $x_0 \in \mathcal{X}_{\text{rec}}^{0}$. Now, by assumption, there exists $\tau \in \{1, ..., N\}$ such that $x_{t+1}, ..., x_{t+\tau} \in \mathcal{X}_{\text{safe}}$ and $x_{t+\tau} \in \mathcal{X}_{\text{rec}}^{t+\tau}$. Since t' is the largest time step less than t such that $x_{t'} \in \mathcal{X}_{\text{rec}}^{t'}$, we must have $t' < t < t + \tau$. But we also have $x_t \in \mathcal{X}_{\text{safe}}$ for every such t. Thus, we have reached a contradiction, and the claim follows. ∎

Next, we show that as a consequence, π^*_{shield} is safe:

Lemma A.4. *We have $\phi^{\epsilon}_{\text{safe}}(\pi^*_{\text{shield}})$.*

Proof: Fix an initial state $x_0 \in \mathcal{X}_0$, and consider the events

$$E_t(\vec{w}) = (x_t \notin \tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,t}) \vee \phi_{\text{rec}}^{\vec{w}_{t:t+N},t},$$

for $t \in \mathbb{N}$, where $\vec{w} \in \mathcal{W}^{\infty}$, $\vec{w}_{0:t} = (w_t, ..., w_{t+N})$, and $\phi_{\text{rec}}^{\vec{w}_{t:t+N},t}$ is as defined in Lemma A.2. It follows straightforwardly from Lemma A.2 that for all $t \in \mathbb{N}$, we have

$$\mathbb{P}_{\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{\infty}}(E_t(\vec{w})) \geq 1 - \epsilon_t.$$

Taking a union bound, and using the fact that $\sum_{t=0}^{\infty} \epsilon_t = \epsilon$, we have

$$\mathbb{P}_{\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{\infty}}(E(\vec{w})) \geq 1 - \epsilon,$$

where

$$E(\vec{w}) = \forall t \in \mathbb{N} \;.\; E_t(\vec{w}).$$

Now, note that on event $E(\vec{w})$, the assumptions in Lemma A.3 are satisfied for the trajectory $x_0, x_1, ...$ obtained using π^*_{shield} with disturbances $\vec{w}$, and for the sequence of sets $\tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,t}$—in particular,

$$x_0 \in \mathcal{X}_0 \subseteq \mathcal{X}_{\text{inv}} \subseteq \tilde{\mathcal{X}}_{\text{rec}}^{\epsilon,0},$$

and $E_t(\vec{w})$ is exactly the assumption (5). It follows that $x_0, x_1, ... \in \mathcal{X}_{\text{safe}}$ on event $E(\vec{w})$. The claim follows. ∎

Next, we show that π_{shield} acts exactly the same way as π^*_{shield} with probability $1 - \delta$:

Lemma A.5. *For a fixed initial state $x_0 \in \mathcal{X}_0$ and sequence of disturbances $\vec{w} \in \mathcal{W}^{\infty}$, let $x_0, x_1, x_2, ...$ be the trajectory obtained by using π_{shield}, and let $x'_0 = x_0, x'_1, x'_2, ...$ be the trajectory obtained by using π^*_{shield}. Then, we have*

$$\mathbb{P}_{\vec{\alpha}\sim\mathcal{P}_{\mathcal{A}}^{\infty}}(\forall t \in \mathbb{N} \;.\; x_t = x'_t) \geq 1 - \delta.$$

Proof: By Lemma A.1, for every $t \in \mathbb{N}$, we have

$$\mathbb{P}_{\alpha\sim\mathcal{P}_{\mathcal{A}}^{t}}(\text{IsRecoverable}(x_t, t) = x_t \in \mathcal{X}_{\text{rec}}^{\epsilon_t}) \geq 1 - \delta_t.$$

Taking a union bound, and using the fact that $\sum_{t=0}^{\infty} \delta_t = \delta$, we have

$$\mathbb{P}_{\vec{\alpha}\sim\mathcal{P}_{\mathcal{A}}^{\infty}}(E) \geq 1 - \delta,$$

where

$$E(\vec{\alpha}) = \forall t \in \mathbb{N} \;.\; \text{IsRecoverable}(x_t, t) = x_t \in \mathcal{X}_{\text{rec}}^{\epsilon_t}.$$

Now, to prove the claim, it suffices to prove that on event $E(\vec{\alpha})$, $x_t = x'_t$ for all $t \in \mathbb{N}$. We prove by induction. The base case $t = 0$ follows by assumption. The inductive case follows since on event $E(\vec{\alpha})$, we have

$$x_{t+1} = \pi_{\text{shield}}(x_t) = \pi^*_{\text{shield}}(x_t) = \pi^*_{\text{shield}}(x'_t) = x'_{t+1}.$$

The claim follows. ∎

Finally, we prove Theorem IV.4.

Proof: Fix an initial state $x_0 \in \mathcal{X}_0$. Let $E(\vec{\alpha})$ be the event in the statement of Lemma A.5—i.e., for a fixed initial state $x_0 \in \mathcal{X}_0$ and sequence of disturbances $\vec{w} \in \mathcal{W}^{\infty}$, the trajectory $x_0, x_1, x_2, ...$ obtained by using π_{shield} is identical to the trajectory $x'_0 = x_0, x'_1, x'_2, ...$ obtained by using π^*_{shield}. As a consequence, $x_0, x_1, x_2, ... \in \mathcal{X}_{\text{safe}}$ if and only if $x'_0, x'_1, x'_2, ... \in \mathcal{X}_{\text{safe}}$. In other words, by Lemma A.4, on event $E(\vec{\alpha})$, we have

$$\begin{aligned} 1 - \epsilon &\geq \mathbb{P}_{\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{\infty}}(\forall t \in \mathbb{N} \;.\; x'_t \in \mathcal{X}_{\text{safe}}) \\ &= \mathbb{P}_{\vec{w}\sim\mathcal{P}_{\mathcal{W}}^{\infty}}(\forall t \in \mathbb{N} \;.\; x_t \in \mathcal{X}_{\text{safe}}). \end{aligned}$$

By Lemma A.5, we have $\mathbb{P}_{\vec{\alpha}\sim\mathcal{P}_{\mathcal{A}}^{\infty}}(E(\vec{\alpha})) \geq 1 - \delta$. The claim follows. □ ∎

Appendix B
Proof of Lemma V.1

First, we have the following result bounding the Taylor series approximation error [39]:

Theorem B.1. *(Taylor's theorem) Let $g : \mathbb{R}^d \to \mathbb{R}$ be twice continuously differentiable and $\bar{x} \in \mathbb{R}^d$. For all $x \in \mathbb{R}^d$, there exists $\xi \in \mathbb{R}^d$ of the form $\xi = \bar{x} + c(x - \bar{x})$, where $c \in [0, 1]$, such that*

$$g(x) = g(\bar{x}) + \nabla_x g(\bar{x})(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^{\top} \nabla_x^2 g(\xi)(x - \bar{x}).$$

Now, consider the linear approximation $\bar{g} : \mathbb{R}^d \to \mathbb{R}$ of $g : \mathbb{R}^d \to \mathbb{R}$ near $\bar{x} \in \mathbb{R}^d$ defined by

$$g(x) \approx \bar{g}(x) = g(\bar{x}) + \nabla_x g(\bar{x})(x - \bar{x}).$$

Suppose that all derivatives of g are L-Lipschitz continuous for $\|\cdot\|_2$. Then, each entry in $\nabla_x^2 g(x)$ is uniformly bounded in absolute value by L. Thus, we have

$$\begin{aligned} |g(x) - \bar{g}(x)| &\leq \frac{\|\nabla_x^2 g(\xi)\|_2}{2} \|x - \bar{x}\|_2^2 \\ &\leq \frac{\|\nabla_x^2 g(\xi)\|_F}{2} \|x - \bar{x}\|_2^2 \\ &\leq \frac{Ld}{2} \|x - \bar{x}\|_2^2. \end{aligned}$$

As we describe next, we apply this result to each component of the dynamics to prove Lemma V.1.

In particular, let $z = (\bar{x}, \bar{u}) \in \mathcal{Z}_{\text{eq}}$ be a safe equilibrium point, let $K \in \mathbb{R}^{n_X \times n_U}$ be the LQR for z, and let $\tilde{f} : \mathcal{X} \times \mathcal{W} \to \mathcal{X}$ be the closed-loop dynamics $\tilde{f}(x, w) = f(x, Kx, w)$. Consider the approximation

$$\begin{aligned} \tilde{f}(x, w) &\approx \bar{f}(x) \\ &= \tilde{f}(\bar{x}, 0) + \nabla_x \tilde{f}(\bar{x}, 0)(x - \bar{x}) \\ &= \bar{x} + \nabla_x \tilde{f}(\bar{x}, 0)(x - \bar{x}), \end{aligned}$$

Algorithm 2 Generate a trajectory from x_0 for dynamics f with uncertain parameters $\theta \sim \mathcal{P}_\Theta$.

```
procedure RunUncertainSMPS($x_0$)
    $\hat{f} \leftarrow$ ConstructHistoryMDP($f$)
    $\hat{\zeta} \leftarrow$ RunSMPS($x_0$) using dynamics $\hat{f}$ and disturbance bound $w_{\max} + \theta_{\max}$
    $\zeta \leftarrow (x_0, x_1, ...)$ where $\hat{\zeta} = ((x_0, h_0), (x_1, h_1), ...)$
    return $\zeta$
end procedure
```

where the second inequality follows since we have assumed that $\bar{x}$ is an equilibrium point. Our goal is to bound the approximation error

$$\tilde{w} = \tilde{f}(x, w) - \bar{f}(x),$$

which includes both the disturbance w and the linearization error. To this end, we have

$$\begin{aligned}\|\tilde{w}\|_2 &= \|\tilde{f}(x, w) - \bar{f}(x)\|_2 \\ &\leq \|\tilde{f}(x, w) - \tilde{f}(x, 0)\|_2 + \|\tilde{f}(x, 0) - \bar{f}(x)\|_2 \\ &\leq L\|w\|_2 + \frac{\tilde{L}n_X^{3/2}}{2}\|x - \bar{x}\|_2^2 \\ &\leq Lw_{\max} + \frac{\tilde{L}n_X^{3/2}r^2}{2},\end{aligned}$$

where the first term is bounded since $\tilde{f}$ is L-Lipschitz continuous in w and since $\|w\|_2 \leq w_{\max}$, and the second term is bounded by the above discussion applied independently to each component of $\tilde{f}(x, 0) - \bar{f}(x)$. The claim follows. □

Appendix C
Extension to Gaussian Processes

Consider dynamics $f : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \times \Theta \to \mathcal{X}$, where $\theta \in \Theta \subseteq \mathbb{R}^{n_\Theta}$ are unknown parameters to be estimated from observations. To ensure safety, we must bound the initial uncertainty; we assume a prior $\theta \sim \mathcal{P}_\Theta$ such as a Gaussian process [18, 2, 10]. Given initial state $x_0 \in \mathcal{X}_0$, policy $\pi : \mathcal{X} \to \mathcal{U}$, disturbances $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$, and parameters $\theta \sim \mathcal{P}_\Theta$, the corresponding trajectory is $\zeta(x_0, \pi, \vec{w}, \theta) = (x_0, x_1, ...) \in \mathcal{X}^\infty$, where $x_{t+1} = f(x_t, \pi(x_t), w_t, \theta)$.

Assumption C.1. Given $\epsilon' \in \mathbb{R}_{>0}$, there exists $\theta_{\max} \in \mathbb{R}_{>0}$ such that $\mathbb{P}_{\theta \sim \mathcal{P}_\Theta}(\|\theta\|_2 \leq \theta_{\max}) \geq 1 - \epsilon'$.

To handle this setting, we apply SMPS to the *history MDP* that records the history of observations $h \in \mathcal{H} \subseteq \bigcup_{t=0}^\infty (\mathcal{X} \times \mathcal{U} \times \mathcal{X})^t$. Then, we use the posterior estimate $p(\theta \mid h) = \mathcal{P}_{\Theta,h}(\theta)$ of θ to sample the next state. For simplicity, we assume $\mathcal{P}_{\Theta,h}$ can be reparameterized—i.e., $\theta \sim \mathcal{P}_{\Theta,h}$ has distribution equal to that of $g(\psi, h)$, where $g : \Psi \times \mathcal{H} \to \Theta$, $\psi \in \Psi \subseteq \mathbb{R}^{n_\Psi}$, and $\psi \sim \mathcal{P}_\Psi$ is an i.i.d. random variable. Then, the history MDP has states $\hat{\mathcal{X}} = \mathcal{X} \times \mathcal{H}$, actions $\hat{\mathcal{U}} = \mathcal{U}$, disturbances $\hat{\mathcal{W}} = \mathcal{W} \times \Psi$, where $\mathcal{P}_{\hat{\mathcal{W}}}((w, \psi)) = \mathcal{P}_\mathcal{W}(w) \cdot \mathcal{P}_\Psi(\psi)$, and transitions $\hat{f} : \hat{\mathcal{X}} \times \hat{\mathcal{U}} \times \hat{\mathcal{W}} \to \hat{\mathcal{X}}$, where

$$\hat{f}((x, h), u, (w, \psi)) = (x', h \cup [(x, u, x')])$$

where

$$x' = f(x, u, w, g(\psi, h)).$$

We can use $\hat{f}$ in conjunction with SMPS; see Algorithm 2. It uses $w_{\max} + \theta_{\max}$ as the disturbance bound instead of $w_{\max}$. We have the following guarantee:

Theorem C.2. *We have*

$$\mathbb{P}_{\alpha \sim \mathcal{P}_\mathcal{A}^\infty}(A_{\epsilon,\epsilon'}) \geq 1 - \delta,$$

where $A_{\epsilon,\epsilon'}$ is the event

$$\mathbb{P}_{\theta \sim \mathcal{P}_\Theta, \vec{w} \sim \mathcal{P}_\mathcal{W}^\infty}(\zeta(x_0, \pi_{\mathit{shield}}, \vec{w}, \theta) \subseteq \mathcal{X}_{\mathit{safe}}) \geq 1 - \epsilon - \epsilon'.$$

For example, suppose $\mathcal{P}_\Theta$ is a Gaussian process—i.e., $\theta \sim \mathcal{P}_\Theta$ is a function $\theta : \mathcal{X} \times \mathcal{U} \times \mathcal{W} \to \mathcal{X}$, and

$$f(x, u, w, \theta) = f_0(x, u, w) + \theta(x, u, w),$$

where $f_0 : \mathcal{X} \times \mathcal{U} \times \mathcal{W}$; we assume $\theta(x, u, w)$ is uniformly bounded with probability at least $1 - \epsilon'$ [10]. Given $h \in \mathcal{H}$, we have

$$\mathcal{P}_{\Theta,h,x,u,w} = \mathcal{N}(\mu(h, x, u, w), \sigma(h, x, u, w)^2),$$

so letting $\psi \sim \mathcal{N}(0, 1)$ be an i.i.d. Gaussian random variable, we have

$$\begin{aligned}&\hat{f}((x, h), u, (w, \psi)) \\ &\quad = f_0(x, u, w) + \mu(h, x, u, w) + \sigma(h, x, u, w) \cdot \psi.\end{aligned}$$

We now prove Theorem C.2. First, consider a trajectory in the history MDP—i.e.,

$$\hat{\zeta}(\hat{x}_0, \hat{\pi}, \vec{\hat{w}}) = \hat{x}_0, \hat{x}_1, ... = (x_0, h_0), (x_1, h_1), ...$$

where

$$\hat{x}_{t+1} = \hat{f}(\hat{x}_t, \hat{\pi}(\hat{x}_t), \hat{w}_t).$$

Then, given $x_0 \in \mathcal{X}$, $\pi : \mathcal{X} \to \mathcal{U}$, $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$, and $\vec{\psi} \sim \mathcal{P}_\Psi$, let $\hat{x}_0 = (x_0, \varnothing) \in \hat{X}$, $\hat{\pi} : \hat{\mathcal{X}} \to \hat{\mathcal{U}}$ be defined by $\hat{\pi}((x, h)) = \pi(x)$, and

$$\vec{\hat{w}} = (w_0, \psi_0), (w_1, \psi_1), ...$$

Then, we can construct the *projected trajectory*

$$\bar{\zeta}(x_0, \pi, \vec{w}, \vec{\psi}) = x_0, x_1, ...$$

by constructing

$$\hat{\zeta}(\hat{x}_0, \hat{\pi}, \vec{\hat{w}}) = (x_0, h_0), (x_1, h_1), ...$$

and projecting out the histories h_t for all $t \in \mathbb{N}$.

Lemma C.3. *Given $x_0 \in \mathcal{X}$, policy $\pi : \mathcal{X} \to \mathcal{U}$, and $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$, the random variable $\zeta(x_0, \pi, \vec{w}, \theta) \in \mathcal{X}^\infty$, where $\theta \sim \mathcal{P}_\Theta$, is equal in distribution to the random variable $\bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi}) \in \mathcal{X}^\infty$, where $\vec{\psi} \sim \mathcal{P}_\Psi^\infty$—i.e.,*

$$\begin{aligned}&\mathbb{P}_{\theta \sim \mathcal{P}_\Theta}\Big(\zeta(x_0, \pi, \vec{w}, \theta) \in E\Big) \\ &\quad = \mathbb{P}_{\vec{\psi} \sim \mathcal{P}_\Psi^\infty}\Big(\bar{\zeta}(x_0, \pi, \vec{w}, \vec{\psi}) \in E\Big)\end{aligned}$$

for all events (i.e., measurable sets) E over $\mathcal{X}^\infty$.

Proof. First, we show that the distribution of $\zeta_T(x_0, \pi, \vec{w}, \theta) \in \mathcal{X}^{T+1}$, where $\theta \sim \mathcal{P}_\Theta$, equals the distribution of $\bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi}) \in \mathcal{X}^{T+1}$, where $\vec{\psi} \sim \mathcal{P}_\Psi^\infty$; here, ζ_T and $\bar{\zeta}_T$ denote partial trajectories up to time $T \in \mathbb{N}$. We prove by induction on T. The base case $T = 0$ follows since x_0 is constant. For the inductive case, let $\zeta_T(x_0, \pi, \vec{w}, \theta)$ have distribution $p(x_0, x_1, ..., x_T, x_{T+1})$, and $\bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi})$ have distribution $\bar{p}(x_0, x_1, ..., x_T, x_{T+1})$. Then, we have

$$\begin{aligned} &p(x_0, x_1, ..., x_T, x_{T+1}) \\ &\quad = p(x_{T+1} \mid x_0, x_1, ..., x_T) \cdot p(x_0, x_1, ..., x_T), \end{aligned}$$

and similarly for $\bar{p}$. By induction, we have $p(x_0, x_1, ..., x_T) = \bar{p}(x_0, x_1, ..., x_T)$; thus, it suffices to show that $p(x_{T+1} \mid x_0, x_1, ..., x_T) = \bar{p}(x_{T+1} \mid x_0, x_1, ..., x_T)$. For p, we have

$$x_{T+1} = f(x_T, u_T, w_T, \theta),$$

where $\theta \sim \mathcal{P}_{\Theta, h_T}$ is the posterior distribution of θ given history $h_T = (x_0, u_0, x_1), ..., (x_{T-1}, u_{T-1}, x_T) \in \mathcal{H}$ and $u_t = \pi(x_t)$ for $t \in \{0, ..., T\}$. By assumption, the distribution of θ equals that of $g(\psi, h_t)$, where $\psi \sim \mathcal{P}_\Psi$, so for p, we have

$$x_{T+1} = f(x_T, u_T, w_T, g(\psi, h_T)),$$

where $\psi \sim \mathcal{P}_\Psi$ i.i.d. For $\bar{p}$, we have

$$x_{T+1} = f(x_T, u_T, w_T, g(\psi_T, h_T)),$$

where $\psi_T \sim \mathcal{P}_\Psi$ i.i.d. Now, we note that the probability measure of $\zeta(x_0, \pi, \vec{w}, \theta)$ and $\bar{\zeta}(x_0, \pi, \vec{w}, \vec{\psi})$ on the infinite product space $\mathcal{X}^\infty$ is the product measure. Consider sets of the form $E = \prod_{t=0}^\infty E_t \subseteq \mathcal{X}^\infty$, where $E_t \subseteq \mathcal{X}$ is the tth component of E, where E is measurable and only has nontrivial components—i.e., $E_t \in \{\varnothing, \mathcal{X}\}$ for all but finitely many $t \in \mathbb{N}$; these sets are the *cylindrical sets*, and form a generating family for the measurable sets of $\mathcal{X}^\infty$ under the product measure [44]. Thus, to prove that $\zeta(x_0, \pi, \vec{w}, \theta)$ is equal in distribution to $\bar{\zeta}(x_0, \pi, \vec{w}, \vec{\psi})$, it suffices to prove that their probabilities are equal for all such events.

To this end, consider such a set E, let $\mathcal{T} = \{t_1, ..., t_k\}$ be the nontrivial components, and let $T = \max\{t_1, ..., t_k\}$. Note that we can assume $E_t = \mathcal{X}$ for all $t \notin \mathcal{T}$, since otherwise the probability of E is zero for both random variables. We have

$$\begin{aligned} &\mathbb{P}_{\theta \sim \mathcal{P}_\Theta}\Big(\zeta(x_0, \pi, \vec{w}, \theta) \in E\Big) \\ &= \mathbb{P}_{\theta \sim \mathcal{P}_\Theta}\Big(\forall t \in \mathbb{N} \ . \ \zeta(x_0, \pi, \vec{w}, \theta)_t \in E_t\Big) \\ &= \mathbb{P}_{\theta \sim \mathcal{P}_\Theta}\Big(\forall t \in \{0, 1, ..., T\} \ . \ \zeta(x_0, \pi, \vec{w}, \theta)_t \in E_t\Big) \\ &= \mathbb{P}_{\theta \sim \mathcal{P}_\Theta}\Big(\forall t \in \{0, 1, ..., T\} \ . \ \zeta_T(x_0, \pi, \vec{w}, \theta)_t \in E_t\Big) \\ &= \mathbb{P}_{\vec{\psi} \sim \mathcal{P}_\Psi^{T+1}}\Big(\forall t \in \{0, 1, ..., T\} \ . \ \bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi})_t \in E_t\Big) \\ &= \mathbb{P}_{\vec{\psi} \sim \mathcal{P}_\Psi^\infty}\Big(\forall t \in \{0, 1, ..., T\} \ . \ \bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi})_t \in E_t\Big) \\ &= \mathbb{P}_{\vec{\psi} \sim \mathcal{P}_\Psi^\infty}\Big(\forall t \in \mathbb{N} \ . \ \bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi})_t \in E_t\Big) \\ &= \mathbb{P}_{\vec{\psi} \sim \mathcal{P}_\Psi^\infty}\Big(\bar{\zeta}_T(x_0, \pi, \vec{w}, \vec{\psi}) \in E\Big), \end{aligned}$$

as claimed. □

Now, we prove Theorem C.2. The key challenge is that Assumption III.2 may not hold for $\hat{f}$. We note that Assumption III.2 is only used to establish Lemma V.1; thus, it suffices to prove that Lemma V.1 continues to hold. Furthermore, Lemma V.1 is only used to establish that

$$\zeta(x_0, \pi_{\text{LQR}}^z, \vec{w}) \subseteq \mathcal{X}_{\text{safe}}$$

for all $x_0 \in \mathcal{X}_0^z$ and $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$, where $z \in \mathcal{Z}_{\text{eq}}$ is a safe equilibrium point. Since π_{LQR}^z is not updated during the execution of π_{shield}, it suffices to prove Lemma V.1 with respect to the original dynamics f. We prove that Lemma V.1 holds on the event $\|\theta\|_2 \leq \theta_{\max}$, which by assumption, holds with probability at least $1 - \epsilon'$.

As in the proof of Lemma V.1, let the closed-loop dynamics $\tilde{f} : \mathcal{X} \times \mathcal{W} \times \Theta \to \mathcal{X}$ be

$$\tilde{f}(x, w, \theta) = f(x, Kx, w, \theta),$$

and let its linear approximation $\bar{f} : \mathcal{X} \to \mathcal{X}$ be

$$\begin{aligned} \tilde{f}(x, w, \theta) \approx \bar{f}(x) &= \tilde{f}(\bar{x}, 0, 0) + \nabla_x \tilde{f}(\bar{x}, 0, 0) \\ &= \bar{x} + \nabla_x \tilde{f}(\bar{x}, 0, 0). \end{aligned}$$

As before, our goal is to bound the approximation error $\tilde{w} = \tilde{f}(x, w, \theta) - \bar{f}(x)$, which includes both the disturbance w, the model uncertainty θ, and the linearization error. Thus, we have

$$\begin{aligned} \|\tilde{w}\|_2 &= \|\tilde{f}(x, w, \theta) - \bar{f}(x)\|_2 \\ &\leq \|\tilde{f}(x, w, \theta) - \tilde{f}(x, 0, 0)\|_2 + \|\tilde{f}(x, 0, 0) - \bar{f}(x)\|_2 \\ &\leq L\|w\|_2 + L\|\theta\|_2 + \frac{Ln_X^{3/2}}{2}\|x - \bar{x}\|_2^2 \\ &\leq L(w_{\max} + \theta_{\max}) + \frac{Ln_X^{3/2} r^2}{2}. \end{aligned}$$

Note that Algorithm 2 constructs π_{LQR}^z using disturbance bound $w_{\max} + \theta_{\max}$; with this modification, the claim of Lemma V.1 follows.

As a consequence, by Theorem IV.4, we have

$$\mathbb{P}_{\vec{\alpha} \sim \mathcal{P}_\mathcal{A}^\infty}(A_\epsilon) \geq 1 - \delta,$$

where A_ϵ is the event

$$\mathbb{P}_{\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty, \vec{\psi} \sim \mathcal{P}_\Psi^\infty}\Big(\bar{\zeta}(x_0, \pi_{\text{shield}}, \vec{w}, \vec{\psi}) \subseteq \mathcal{X}_{\text{safe}}\Big) \geq 1 - \epsilon.$$

By Lemma C.3, $\bar{\zeta}(x_0, \pi_{\text{shield}}, \vec{w})$, where $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$ and $\vec{\psi} \sim \mathcal{P}_\Psi^\infty$, has equal distribution as $\zeta(x_0, \pi_{\text{shield}}, \vec{w}, \theta)$, where $\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty$ and $\theta \sim \mathcal{P}_\Theta$. Thus, we have

$$\mathbb{P}_{\vec{\alpha} \sim \mathcal{P}_\mathcal{A}^\infty}(A_\epsilon) \geq 1 - \delta,$$

where A_ϵ is the event

$$\mathbb{P}_{\vec{w} \sim \mathcal{P}_\mathcal{W}^\infty, \theta \sim \mathcal{P}_\Theta}\Big(\zeta(x_0, \pi_{\text{shield}}, \vec{w}, \theta) \subseteq \mathcal{X}_{\text{safe}}\Big) \geq 1 - \epsilon$$

Finally, we have conditioned on the event $\|\theta\|_2 \leq \theta_{\max}$. The claim follows by a union bound. □