2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics

Shuo Li[1] and Osbert Bastani[2]

Abstract— **We propose a framework for safe reinforcement learning that can handle stochastic nonlinear dynamical systems. We focus on the setting where the nominal dynamics are known, and are subject to additive stochastic disturbances with known distribution. Our goal is to ensure the safety of a control policy trained using reinforcement learning, e.g., in a simulated environment. We build on the idea of model predictive shielding (MPS), where a backup controller is used to override the learned policy as needed to ensure safety. The key challenge is how to compute a backup policy in the context of stochastic dynamics. We propose to use a tube-based robust nonlinear model predictive controller (NMPC) as the backup controller. We estimate the tubes using sampled trajectories, leveraging ideas from statistical learning theory to obtain high-probability guarantees. We empirically demonstrate that our approach can ensure safety in stochastic systems, including cart-pole and a non-holonomic particle with random obstacles.**

I. Introduction

There has been much recent progress in reinforcement learning (RL) [1], [2], [3], [4], [5]. As a consequence, there has been interest in using RL to design control policies for solving complex robotics tasks [6], [7], [8], including grasping [9] and multi-agent planning [10], [11], [12]. In particular, learning-enabled controllers (LECs) have the potential to outperform optimization-based controllers [13]. In addition, optimization-based controllers can often only be used under strong assumptions about the system dynamics, system constraints, and objective functions [14], [15], which limits their applicability to complex robotics tasks.

However, safety concerns prevent LECs from being widely used in real-world tasks, which are often safety-critical in nature. In particular, unlike optimization-based controllers, it is typically infeasible to impose hard safety constraints on LECs. For example, even a simple safety property such as having a robot avoid static obstacles is nontrivial to impose. More complex safety properties, such as ensuring that a cart-pole or walking robot does not fall over, pose additional difficulty. Additionally, the real world typically has stochastic disturbances such as friction or wind, or the model used to simulate the dynamics may have errors; if an LEC is not robust to these perturbations, then it may fail catastrophically [16]. The issue is that the LEC is typically a black box machine learning model such as a deep neural network (DNN), and reasoning about the closed-loop behavior of the DNN poses computational challenges.

[1]Shuo Li is with the GRASP Lab, University of Pennsylvania, USA lishuo1@seas.upenn.edu

[2]Osbert Bastani is with the Department of Computer and Information Science, University of Pennsylvania, USA obastani@seas.upenn.edu

As a consequence, safe reinforcement learning has become an increasingly important area of research [17], [18], [19], [20], [21], [22], [23]. Many methods in this area leverage optimization tools to prove that a learned neural network policy satisfies a given safety constraint [24], [25], [26], [27], [19], [20], [23]. A related approach is *shielding*, which verifies a simpler *backup controller*, and then overrides the LEC using the backup controller when it can no longer ensure that using the LEC is safe [17], [18], [21], [28]. While these methods provide strong mathematical guarantees, they suffer from a number of shortcomings. For example, many of these methods scale exponentially in the state dimension, so they do not scale well to high-dimensional systems. Those that do typically rely on overapproximating reachable set of states, which can become very imprecise.

We build on a recently proposed idea called *model predictive shielding* (MPS), which has been used to ensure safety of learned control policies [29], [28], including extensions to the multi-agent setting [30]. The basic idea is that rather than checking which states are safe ahead-of-time, we can dynamically check whether we can maintain safety if we use the LEC, and only use the LEC if we can do so. However, existing approach are limited—there has been work considering nonlinear dynamics, but assuming they are deterministic [28], [30], and there has been work considering stochastic or nondeterministic dynamics, but assuming they are linear [29]. Nonlinearity is important because many tasks where LECs have the most promise are highly nonlinear. Stochasticity is important for a number of reasons. For instance, there are often small perturbations in real-world dynamical systems. Similarly, it can be used to model estimation error in the robot's state (e.g., uncertainty in its position). Finally, LECs are often learned in simulation using a model of the dynamics; there are often errors in the model that need to be robustly accounted for.

We propose an approach, called robust MPS (RMPS), that extends MPS to the setting of stochastic nonlinear dynamics. Our approach uses robust nonlinear model-predictive control (NMPC) as the backup controller. The reason for using NMPC is that the goals of the backup controller are qualitatively different from the goals of the LEC. For example, consider the problem of building a robot that can run. The LEC tries to make the robot run as fast as possible. It may be able to outperform the robust NMPC, since the robust NMPC treats the stochastic perturbations conservatively. However, because RL lacks theoretical guarantees, the LEC cannot guarantee safety. Thus, we want to use the LEC as often as possible, but override it using a backup controller if we

978-1-7281-7395-5/20/$31.00 ©2020 IEEE

are not sure whether it is safe to use the LEC. The NMPC is an effective choice for the backup controller, where the goal is to stop the system and bring it to an equilibrium point, after which a feedback controller can be used to stabilize it. Continuing our example, the NMPC might bring the robot to a halt (e.g., where it is standing).

To achieve our goals, we build on algorithms for robust NMPC [31], [32], which aims to control a dynamical system in the presence of stochastic or nondeterministic perturbations, and on sampling-based estimation of forward reachable sets [33], [34]; in particular, we build closely on tube-based robust NMPC where sampling is used to estimate a tube within which the NMPC is guaranteed to stay (i.e., the tube is the forward reachable set of the NMPC) [35], [36]. However, these approaches do not provide any finite sample probabilistic guarantees on their estimates of the tubes.

We propose to use results from statistical learning theory to obtain provable probabilistic guarantees on our estimates of the sizes of the tubes [37]. We develop a practical algorithm based on these theoretical results. There has been recent work that provides theoretical guarantees on the estimated forward reachable set [38]; however, they study the problem of obtaining *underapproximations* of the forward reachable set, as opposed to *overapproximations* that are needed for checking safety. Thus, their guarantees require qualitatively different techniques than the ones we use. To the best of our knowledge, we are the first to use these to construct estimates for forward reachable sets of nonlinear dynamical systems that come with probabilistic guarantees.

Contributions. Our key contributions are: (i) an extension of the MPS algorithm to stochastic nonlinear dynamical systems (Section IV), (ii) a novel algorithm for estimating tubes for RMPC with high-probability guarantees (Section IV), as well as heuristic modifications that enable us to scale this algorithm to real-world systems, and (iii) experiments demonstrating how our approach ensures safety of LECs for cart-pole and for a single particle with non-holonomic dynamics and random obstacles (Section V).

II. Preliminaries

Dynamics. We consider stochastic nonlinear dynamics

$$x(k+1) = f(x(k), u(k)) + w(k),$$

where k is the time step, $x(k) \in \mathcal{X} \subseteq \mathbb{R}^{n_X}$ is the state, $u(k) \in \mathcal{U} \subseteq \mathbb{R}^{n_U}$ is the control input, and $w(k) \in \mathcal{W} \subseteq \mathbb{R}^{n_X}$ is a zero-mean stochastic perturbation with distribution $\mathcal{P}_{\mathcal{W}}$.

Policy. A *policy* is a map $\pi : \mathcal{X} \to \mathcal{U}$. A trajectory generated using π from initial state $x \in \mathcal{X}$ is $\mathbf{x} = (x(0), x(1), ...)$, where $x(0) = x$ and $x(k+1) = f^{(\pi)}(x(k)) + w(k)$ and where $f^{(\pi)}(x) = f(x, \pi(x))$. Since $w(k)$ is random, $\mathbf{x}$ is a random sequence; we use $p(\mathbf{x} \mid \pi, x)$ to denote the probability of $\mathbf{x}$.

Objective. We consider a cost function $\ell : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ and a discount factor $\gamma \in (0, 1)$. Given a distribution $p(x)$ over initial states, the cost of a policy π is

$$\ell(\pi) = \mathbb{E}_{p(x), p(\mathbf{x}|\pi,x)} \left[\sum_{k=0}^{\infty} \gamma^k \ell(x(k), u(k)) \right].$$

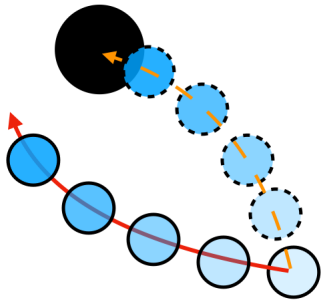

Fig. 1: An illustration of MPS. If using $\hat{\pi}$ (dashed orange line), the robot (blue) may unsafely crash into the obstacle (black). Using π_{backup} (solid red line) ensures safety.

Safety constraint. We consider a set of safe states $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$, with the goal of ensuring that the system stays in $\mathcal{X}_{\text{safe}}$. A trajectory $\mathbf{x}$ is safe if $x(k) \in \mathcal{X}_{\text{safe}}$ for all $k \geq 0$.

Shielding problem. Our goal is to construct a policy π that achieves low cost while ensuring safety. In general, since the dynamics are stochastic, it is impossible to guarantee safety. Instead, our goal is to ensure that safety holds with high probability. We establish a theoretical safety guarantee in Theorem 1; we interpret this theorem in Section IV-C.

Our approach is based on *shielding* [17], [18], [21], [28]. This approach takes as input a policy $\hat{\pi}$ that optimizes $\ell(\pi)$, but does not ensure safety (though a penalty for violating safety may be encoded in ℓ). We call $\hat{\pi}$ the *learned policy*, since our key motivation is when $\hat{\pi}$ is a neural network policy trained using RL—e.g., we use DDPG [39]. Nevertheless, our approach can be used with any controller $\hat{\pi}$.

Then, the *shielding problem* is to construct a policy π_{shield} that overrides $\hat{\pi}$ as needed to ensure safety. The key challenges are (i) determining how to override $\hat{\pi}$, and (ii) minimizing how often $\hat{\pi}$ is overridden.

Notation. For $k \in \mathbb{N}$, we let $[k] = \{1, ..., k\}$. We let $\mathbb{S}^n$ be the positive semidefinite matrices of dimension n. For $x \in \mathbb{R}^n$ and $M \in \mathbb{S}^n$, we let $\|x\|_M = x^\top M x$. For $\mathcal{A}, \mathcal{B} \subseteq \mathbb{R}^n$,

$$\mathcal{A} \oplus \mathcal{B} = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \mathcal{A},\ \mathbf{b} \in \mathcal{B}\}$$
$$\mathcal{A} \ominus \mathcal{B} = \{\mathbf{c} \mid \{\mathbf{c}\} \oplus \mathcal{B} \subseteq \mathcal{A}\}$$

denote their Minkowski sum and difference, respectively.

III. Background on Model Predictive Shielding

Model predictive shielding (MPS) is a shielding algorithm that dynamically checks when to override $\hat{\pi}$. The key idea is to maintain an invariant that it can always use a *recovery policy* π_{rec} to safely transition to an equilibrium point [29], [28], [30]; a state $x \in \mathcal{X}$ that satisfies this invariant is *recoverable* (denoted $x \in \mathcal{X}_{\text{rec}}$). Near the equilibrium point, we assume a linear feedback controller π_{stable} can be used to ensure safety for an infinite horizon. Thus, as long as the system remains in $\mathcal{X}_{\text{rec}}$, then MPS can guarantee safety. The combination of π_{rec} and π_{stable} is the *backup controller* π_{backup} used to override $\hat{\pi}$. This approach is shown in Fig. 1.

For example, consider a driving robot. In this setting, x is recoverable if the robot can safely apply the brakes to come

Algorithm 1 Compute the RMPS controller for state x.

procedure RMPS(x)
  if IsRecoverable($f^{(\hat{\pi})}(x)$) **then**
    $\pi_{\text{backup}} \leftarrow \varnothing$
    return $\hat{\pi}(x)$
  else if $\pi_{\text{backup}} \neq \varnothing \wedge$ IsRecoverable($f^{(\pi_{\text{backup}})}(x)$) **then**
    return $\pi_{\text{backup}}(x)$
  else
    $\pi_{\text{backup}} \leftarrow$ InitializeBackup(x)
    return $\pi_{\text{backup}}(x)$
  end if

Algorithm 2 Compute the backup controller for state x. It keeps internal state $(z_e, \mathcal{G}_e, \mathbf{x}^*, \mathbf{u}^*, t)$.

procedure Backup(x)
  if $t < T$ **then**
    Compute $\tilde{\mathbf{x}}^0, \tilde{\mathbf{u}}^0$ from $x(t) = x$ using (3)
    Update $t \leftarrow t + 1$
    return $\tilde{\mathbf{u}}^0(0)$
  else
    return $\pi_{\text{stable}}(x)$
  end if
procedure InitializeBackup(x)
  Compute target equilibrium point $z_e \leftarrow \rho(x)$
  Compute invariant set $\mathcal{G}_e$ for z_e
  Compute $\bar{\mathbf{x}}^*, \bar{\mathbf{u}}^*$ from $x(0) = x$ using (2)
  Initialize $t \leftarrow 0$
  return $\pi_{\text{backup}}(\cdot) = \pi_{\text{backup}}(\,\cdot\,; z_e, \mathcal{G}_e, \bar{\mathbf{x}}^*, \bar{\mathbf{u}}^*, t)$

to a stop. If x is recoverable, but $f^{(\hat{\pi})}(x)$ is not, then MPS uses π_{backup}. Since x is recoverable, using π_{rec} is guaranteed to keep the system safe. Thus, MPS ensures safety for an infinite horizon from any recoverable state.

IV. Robust Model Predictive Shielding

The key challenge of implementing MPS is how to check whether a state x is recoverable. When the dynamics are deterministic, this check can be performed by simulating the dynamics [28]. However, for stochastic dynamics, each simulation will result in different realizations of $w(k)$, so this approach no longer applies. When the dynamics are linear and the perturbations $w(k)$ are Gaussian, we can analytically check a high-probability variant of recoverability [29], but this approach does not generalize to nonlinear dynamics.

Our approach is based on robust control. First, we use tracking NMPC as the recovery policy [35]. This choice ensures that the system reaches its goal with high probability even with stochastic perturbations. Second, we check recoverability by estimating the reachable set of π_{backup}. In particular, we use the reachable set to ensure that the trajectory generated using π_{backup} (i) is safe, and (ii) reaches an invariant set around an equilibrium point. A key innovation in our approach is that we use tools from statistical learning theory to obtain provable guarantees on our check for recoverability.

A. The Backup Controller

We use a standard robust NMPC as the backup controller [35]. At a high level, this controller first computes a reference trajectory that transitions the system to an equilibrium point. Then, it uses NMPC to track this reference trajectory. Finally, once the trajectory has reached the invariant set $\mathcal{G}_e$ around equilibrium point $z_e \in \mathcal{X} \times \mathcal{U}$, it uses a feedback controller π_{stable} to stabilize the system within $\mathcal{G}_e$.

Stabilization near equilibrium points. We assume given a mapping $\rho : \mathcal{X} \to \mathcal{X} \times \mathcal{U}$, where $z_e = (x_e, u_e) = \rho(x)$ is an *equilibrium point*—i.e., $x_e = f(x_e, u_e)$. Intuitively, $\rho(x)$ returns the equilibrium point z_e "closest" to x. Then, π_{rec} tries to transition the system from x to z_e. Near z_e, we can use an LQR π_{stable} to stabilize the system and ensure safety.

We assume we can compute a safe invariant set $\mathcal{G}_e \subseteq \mathcal{X}_{\text{safe}}$ around z_e—i.e., for $x \in \mathcal{G}_e$, the trajectory $\mathbf{x}$ generated using π_{stable} from $x(0) = x$ (i) is safe, and (ii) remains in $\mathcal{G}_e$. Since the dynamics are stochastic, we cannot guarantee this property with probability 1 (unless $w(k)$ is bounded). Nevertheless, in our experiments, we find that π_{stable} is effective at ensuring safety and stability inside $\mathcal{G}_e$. We discuss how we compute $\mathcal{G}_e$ in Section IV-D.

Reference trajectory. We denote the nominal dynamics by $\bar{x}(k+1) = f(\bar{x}(k), \bar{u}(k))$, where $\bar{x}(k)$ is the nominal state and $\bar{u}(k)$ is the nominal control input. Given an initial state $x \in \mathcal{X}$ and a time horizon T, we compute a nominal trajectory to transition the system to an equilibrium point $z_e = \rho(x)$ by solving the following:

$$\begin{aligned} \underset{\bar{\mathbf{x}}, \bar{\mathbf{u}}}{\arg\min} \quad & \sum_{k=0}^{T-1} \ell(\bar{x}(k) - x_e, \bar{u}(k) - u_e) \qquad (1) \\ \text{subj. to} \quad & \bar{x}(0) = x,\ \bar{x}(T) = x_e,\ \bar{u}(T) = u_e, \\ & \bar{x}(k) \in \bar{\mathcal{X}}_{\text{safe}},\ \bar{u}(k) \in \mathcal{U}, \\ & \bar{x}(k+1) = f(\bar{x}(k), \bar{u}(k))\ (\forall k \in \{0, ..., T-1\}) \end{aligned}$$

where $\ell(x, u) = \|x\|_Q^2 + \|u\|_R^2$ for some $Q \in \mathbb{S}^{n_X}$ and $R \in \mathbb{S}^{n_U}$. Furthermore, $\bar{\mathcal{X}}_{\text{safe}} \subseteq \mathcal{X}_{\text{safe}}$ can be specified by the user to improve robustness; we describe heuristics for computing these sets in Section IV-D. We denote the solution to (1) by $(\bar{\mathbf{x}}^0, \bar{\mathbf{u}}^0) \in \mathcal{X}^{T+1} \times \mathcal{U}^{T+1}$. Since (x_e, u_e) is a nominal equilibrium, the infinite horizon trajectory

$$\bar{\mathbf{x}}^* = \bar{\mathbf{x}}^0 \circ (x_e, x_e, ...), \qquad \bar{\mathbf{u}}^* = \bar{\mathbf{u}}^0 \circ (u_e, u_e, ...), \qquad (2)$$

where $\circ$ is concatenation, is safe for the nominal dynamics.

Tracking NMPC. Once we have a reference trajectory $\bar{\mathbf{x}}^*, \bar{\mathbf{u}}^*$, we use NMPC to track this reference trajectory and try to reach the equilibrium z_e. In particular, if we are at state $x(t)$ after t steps, this controller solves the following:

$$\begin{aligned} \underset{\tilde{\mathbf{x}}, \tilde{\mathbf{u}}}{\arg\min} \quad & \sum_{k=0}^{T-1} \ell(\tilde{x}(k) - \bar{x}^*(t+k), \tilde{u}(k) - \bar{u}^*(t+k)) \qquad (3) \\ & \quad + V_f(\tilde{x}(T)) \\ \text{subj. to} \quad & \tilde{x}(0) = x(t),\ \tilde{x}(k) \in \mathcal{X}_{\text{safe}},\ \tilde{u}(k) \in \mathcal{U}, \\ & \tilde{x}(k+1) = f(\tilde{x}(k), \tilde{u}(k))\ (\forall k \in \{0, ..., T-1\}) \end{aligned}$$

Algorithm 3 Check if x is robustly recoverable.

procedure IsRecoverable(x)
  $\pi_{\text{backup}} \leftarrow$ InitializeBackup(x)
  Let $\bar{\mathbf{x}}^*$ be the reference trajectory of π_{backup}
  $\mathbf{B} \leftarrow$ EstimateReachableSets(x, π_{backup})
  for $t \in \{0, ..., T\}$ **do**
    if $\bar{x}^*(t) \notin \mathcal{X}_{\text{safe}} \ominus B(t)$ **then**
      return false
    end if
  end for
  if $\bar{x}^*(T) \notin \mathcal{G}_e \ominus B(T)$ **then**
    return false
  end if
  return true

where $V_f(x)$ is the cost-to-go function of the LQR for the linearization of the nominal dynamics f around z_e [35]. We let $\tilde{\mathbf{x}}^0, \tilde{\mathbf{u}}^0 \in \mathcal{X}^{T+1} \times \mathcal{U}^{T+1}$ be the solution of (3).

Backup controller. Given an state x, an equilibrium point $z_e \in \mathcal{X} \times \mathcal{U}$, and a time horizon T, our backup controller π_{backup} first computes the reference trajectory $\bar{\mathbf{x}}^0, \bar{\mathbf{u}}^0$ using (1), with corresponding infinite horizon reference trajectory $\bar{\mathbf{x}}^*, \bar{\mathbf{u}}^*$. Then, for each step $t \in \{0, ..., T-1\}$, π_{backup} solves (3) for the current state $x(t)$ to obtain $\tilde{\mathbf{x}}^0, \tilde{\mathbf{u}}^0$, and chooses control input $u(t) = \tilde{u}^0(0)$. Finally, for $t \geq T$, it chooses control input $u(t) = \pi_{\text{stable}}(x(t))$.

This procedure for computing the backup controller is summarized in Algorithm 2. Note that π_{backup} actually needs to keep internal state consisting of the target equilibrium point $z_e = \rho(x)$, its corresponding invariant set $\mathcal{G}_e$, the reference trajectory $\bar{\mathbf{x}}^*, \bar{\mathbf{u}}^*$ to the equilibrium point, and the number of steps t taken so far using the backup controller. This internal state is initialized in the context of a given state $x \in \mathcal{X}$ by the function call InitializeBackup(x).

B. Checking Robust Recoverability via Sampling

Building on ideas from tube NMPC [15], we use sampling to determine whether π_{rec} can safely reach the invariant set $\mathcal{G}_e$ from the current state x. In particular, we sample N trajectories according to the stochastic dynamics, and fit boxes $B(t)$ that cover all the states sampled on each given step $t \in \{0, ..., T\}$. Intuitively, if we take N to be sufficiently large, then the realized trajectory will lie in $B(t)$ at each step t with high probability. In contrast to prior work, we make this intuition precise using tools from statistical learning theory. Finally, to check if x is recoverable, we check that it is robustly safe according to the uncertainty in these boxes, and furthermore that it robustly enters the invariant set $\mathcal{G}_e$.

Robust recoverability. Our goal is to ensure that π_{rec} can always transition the system safely from the current state x to the invariant set $\mathcal{G}_e$ around $z_e = \rho(x)$. Due to the random perturbation $w(k)$ in the dynamics, we cannot make an absolute guarantee that this property holds. Following prior work [17], [18], [19], we instead aim to guarantee that this property holds with high probability.

Algorithm 4 Estimates the reachable sets $B(t)$ after t steps using Monte Carlo sampling.

procedure EstimateReachableSets(x, π_{backup})
  Compute N that satisfies (5)
  for $i \in [N]$ **do**
    Sample $w \sim \mathcal{P}_{\mathcal{W}}$, and let $x' = x + w$
    Sample $(x_i(0), ..., x_i(T))$ from $x_i(0) = x'$ using π_{backup}
  end for
  for $t \in \{0, 1, ..., T\}$ **do**
    Fit $B(t) \in \mathcal{B}$ to $\{x_i(t) \mid i \in [N]\}$
  end for
  return $\mathbf{B} = (B(0), ..., B(T))$

Definition 1: Let $\epsilon \in \mathbb{R}_{>0}$ and $x \in \mathcal{X}$ be given. Let $z_e = \rho(x)$, let $\mathcal{G}_e$ be an invariant set for π_{stable} around z_e, and let $\mathbf{x}^0 = (x(0), ..., x(T))$ be a trajectory generated using π_{backup} from $x(0) = x + w$, where $w \sim \mathcal{P}_{\mathcal{W}}$. We say x is ϵ *robustly recoverable* if with probability at least $1 - \epsilon$, (i) $x(t)$ is safe for every $t \in \{0, ..., T\}$, and (ii) $x(T) \in \mathcal{G}_e$.

In other words, $\mathbf{x}^0$ safely transitions the system from x to $\mathcal{G}_e$ with probability at least $1-\epsilon$. Algorithm 3 checks whether a state x is robustly recoverable using sampling. Prior work on safe reinforcement learning [17], [18], [19] has relied on thresholding the perturbation distribution and then using verification to obtain similar bounds. This approach can guarantee that x is robustly recoverable with probability 1. In contrast, using our approach, there is an additional chance δ (for any given $\delta \in \mathbb{R}_{>0}$) that our algorithm incorrectly concludes that x is robustly recoverable when it is not. The difference is that the ϵ error in robust recoverability is due to the noise in the dynamics, whereas our δ error is due to noise in the sampled trajectories taken by our algorithm.

We believe this added potential for error is reasonable for two reasons. First, there is already an ϵ chance of error, so the added error δ does not affect the nature of our guarantee. Second, the dependence of the running time of our algorithm $1/\delta$ is logarithmic, so we can choose δ to be very small.

Estimating reachable sets. Our approach is to compute sets $B(t)$ for $t \in \{0, ..., T\}$ such that the trajectory $\mathbf{x}^0$ satisfies $x(t) - \bar{x}^*(t) \in B(t)$ with probability at least $1 - \epsilon$, where $\bar{\mathbf{x}}^*$ is the reference trajectory used by π_{backup}—i.e.,

$$\Pr(x(t) - \bar{x}^*(t) \in B(t)) \geq 1 - \epsilon, \tag{4}$$

where the probability is taken over the randomness in the dynamics. To this end, a *box constraint* is a set

$$B = [a_1, b_1] \times ... \times [a_n, b_n] \subseteq \mathbb{R}^n,$$

where $[a, b] \subseteq \mathbb{R}$ denotes the closed interval from a to b. We use $\mathcal{B}$ to denote the set of all possible boxes. Now, we have the following theoretical guarantee.

Lemma 1: Let d be a distribution over $\mathbb{R}^n$ and $\epsilon, \delta \in \mathbb{R}_{>0}$ be given. Consider N i.i.d. samples $x_1, ..., x_N \sim d$, where

$$N \geq \frac{n \log \frac{2eN}{n} + \log \frac{4}{\delta}}{\epsilon^2}, \tag{5}$$

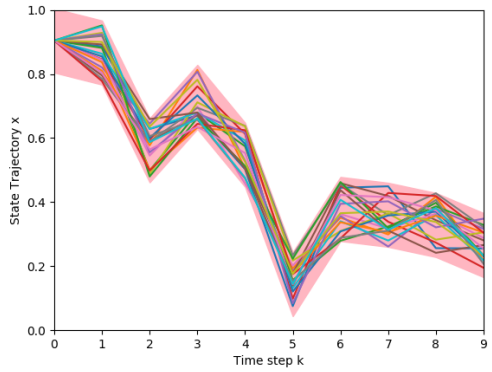

Fig. 2: An example of a tube (red region) estimated using Algorithm 4 for π_{backup}. We guarantee the realized trajectory lies in this tube with high probability.

and let $B \in \mathcal{B}$ be any box satisfying $x_i \in B$ for all $i \in \{1, ..., N\}$. Then, with probability at least $1-\delta$, we have

$$\Pr_{x\sim d}(x \in B) \geq 1 - \epsilon. \tag{6}$$

Intuitively, (6) says that at least a $1-\epsilon$ fraction of states (weighted by d) fall inside B, and this guarantee holds with probability at least $1-\delta$. The proof, based on tools from statistical learning theory, is given in Appendix A.

Algorithm 4 takes N samples of the trajectory $\mathbf{x}^0$ by simulating the dynamics, and fits a box $B(t)$ based on the sampled states on each step $t \in \{0, ..., T\}$. The following guarantee follows from Lemma 1 via a union bound:

Lemma 2: Let $\mathbf{B}$ be the sequence of boxes returned by Algorithm 4. With probability at least $1-(T+1)\delta$, we have

$$\Pr(\forall t \in \{0, ..., T\},\ x(t) - \bar{x}^*(t) \in B(t)) \geq 1 - (T+1)\epsilon.$$

As before, the $1-(T+1)\delta$ probability is according to the samples used by our algorithm, whereas the $1-(T+1)\epsilon$ probability is according to the randomness in the dynamics.

The sets $B(t)$ computed using Algorithm 4 can be thought of as an estimate of a tube in which the trajectories are guaranteed to stay [35]. In contrast to prior work, we have used results from statistical learning theory to obtain probabilistic guarantees on the correctness of these tubes [37]. An example of an estimated tube is shown in Fig. 2.

Checking recoverability. Given the boxes $B(t)$ for $t \in \{0, ..., T\}$, Algorithm 3 checks both properties required for robust recovery: (i) to check if $x(t) \in \mathcal{X}_{\text{safe}}$ with high probability, it checks if

$$\{\bar{x}^*(t) + x \mid x \in B(t)\} \subseteq \mathcal{X}_{\text{safe}},$$

which is equivalent to $\bar{x}^*(t) \in \mathcal{X}_{\text{safe}} \ominus B(t)$, and (ii) to check if $x(T) \in \mathcal{G}_e$ with high probability, it checks if

$$\{\bar{x}^*(t) + x \mid x \in B(T)\} \subseteq \mathcal{G}_e,$$

which is equivalent to $x(T) \in \mathcal{G}_e \ominus B(T)$. These checks ensure robust recoverability because Corollary 2 ensures that $x(t) \in B(t)$ with high probability for every $t \in \{0, ..., T\}$. Thus, we have the following guarantee:

Lemma 3: Given a state $x \in \mathcal{X}$, if Algorithm 3 returns true, then x is $(T+1)\epsilon$ robustly recoverable with probability $1-(T+1)\delta$ (according to the randomness in the algorithm).

C. Robust Model Predictive Shielding

Our robust model predictive shielding (RMPS) algorithm is shown in Algorithm 1. At state $x \in \mathcal{X}$, this algorithm computes a control input (denoted $\pi_{\text{shield}}(x)$) by checking whether next state $f^{(\bar{\pi})}(x)$ is robustly recoverable (with high probability) in simulation. Otherwise, it takes a step according to π_{backup}. One subtlety is that if π_{backup} has already been initialized, it actually needs to check if $f^{(\pi_{\text{backup}})}(x)$ is robustly recoverable. The issue is that robust recoverability is defined with respect to a freshly initialized backup policy, *not* the backup policy after it has taken some number of steps. We have the following guarantee:

Theorem 1: If $x \in \mathcal{X}$ is $1-(T+1)\epsilon$ robustly recoverable, then $f^{(\pi_{\text{shield}})}(x)+w$ (where $w \sim \mathcal{P}_{\mathcal{W}}$) is $1-(T+1)\epsilon$ robustly recoverable with probability at least $1-2(T+1)\delta$.

See Appendix VI-B for a proof. This guarantee is that it does not ensure safety of the infinite horizon trajectory. Given our assumptions, a stronger guarantee is impossible, since on every step there is a chance that the additive perturbation is large, causing the system to leave $\mathcal{X}_{\text{safe}}$. In practice, we find that the bounds can be tighter than the theory suggests, since the robust NMPC is actually conservatively overapproximating the reachable set. In other words, the robust NMPC ensures safety much more robustly than the probabilities in Theorem 1 would suggest.

We briefly discuss the running time of Algorithm 1. First, it is straightforward to see that the number of samples N is $\tilde{O}(1/\epsilon^2)$ (where $\tilde{O}$ indicates that we have suppressed log factors including $\log(1/\delta)$). For each sample, we need to simulate the closed-loop dynamics $f^{(\pi_{\text{rec}})}$ over a horizon of length T. Thus, assuming the cost of simulating $f^{(\pi_{\text{rec}})}$ for one step is τ, the running time of Algorithm 1 is $\tilde{O}(T\tau/\epsilon^2)$.

D. Practical Modifications

We describe several practical modifications to our algorithm designed to improve performance or computational tractability. These modifications may weaken our safety guarantees, but empirically, they do not affect safety.

Computing $\mathcal{G}_e$. We use a heuristic to compute $\mathcal{G}_e$ from [35]. We sample trajectories over a long horizon and estimate the reachable set B as in Algorithm 4. This approach does not provide any guarantees that the estimated set $\mathcal{G}_e$ is actually invariant, but it works well empirically.

Precomputing B. Computing the sets $B(t)$ (for $t \in \{0, ..., T\}$) on-the-fly can be prohibitively expensive, since we might need a large number of samples N for Lemma 1 to apply. Instead, we precompute these sets from a fixed initial state x_0. Then, we reuse the same states rather than recomputing them at each step. Intuitively, this approach works well in practice since the dynamics of the tracking NMPC are usually fairly similar for different initial states.

Using tighter constraints for NMPC. In the optimization problem (1) used to compute the reference trajectory, we

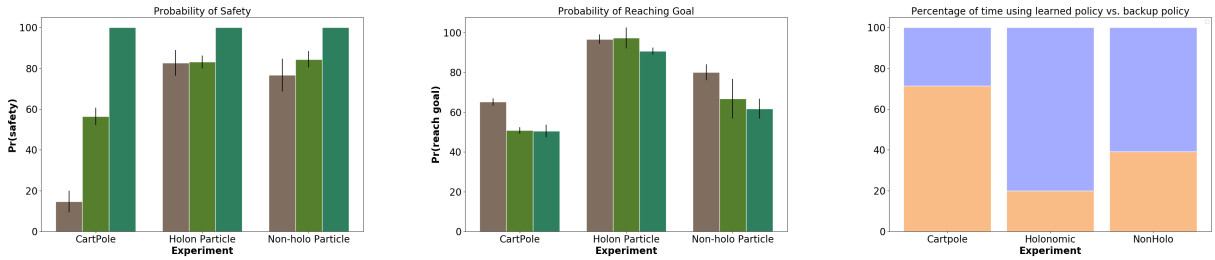

Fig. 3: Left: Safety probabilities for no shielding (light brown), non-robust MPS (dark brown), and RMPS (dark green). Middle: Probability of reaching goal for the same policies. Right: Percentage of time using the learned policy $\hat{\pi}$ (blue) compared to the backup policy π_{backup} (orange) by RMPS.

noted that we can use tighter state and input constraints $\bar{x}(k) \in \bar{\mathcal{X}}_{\text{safe}}$ than needed. In particular, by doing so, we can improve the robustness of the tracking NMPC—we use the "tightened set" $\bar{\mathcal{X}}_{\text{safe}} = \mathcal{X}_{\text{safe}} \ominus B(t)$. Unlike the previous two heuristics, this one does not affect our theoretical guarantees, since our guarantees hold for an arbitrary backup policy. While our approach would ensure safety even without the tighter state and input constraints, using tighter constraints improves the chances that the backup policy recovers the system from a given state x—i.e., it improves the size of $\mathcal{X}_{\text{rec}}$. In other words, it increases the probability that the robot reaches its goal without diminishing the probability of safety.

V. Experiments

Setting. We demonstrate how our algorithm can ensure safety of stochastic systems with nonlinear dynamics and nonconvex constraints. We consider three systems: a cart-pole, a holonomic particle, and a non-holonomic particle.

For the cart-pole, the states are $z = (x, v, \theta, \omega) \in \mathbb{R}^4$, where x is the cart position, v is the cart velocity, θ is the pole angle, and ω is the pole angular velocity, the control inputs is the cart acceleration $u \in \mathbb{R}$, the goal is to reach a target position x_{target}, the safety constraint is that the pole should not fall down while moving the cart, and the disturbances are uniform noise $w_i(k) \sim \text{Uniform}([-0.01, 0.01])$ for the velocity and angular velocity, and zero otherwise.

For the single particle with holonomic dynamics, the states are $z = (x, y, v_x, v_y) \in \mathbb{R}^4$, where (x, y) is position and (v_x, v_y) is velocity, and the control inputs are $(u_x, u_y) \in \mathbb{R}^2$, where (u_x, u_y) is the acceleration. The goal is to reach a state $g \in \mathbb{R}^2$ while avoiding obstacles o_i $(i \in [N])$ Disturbances are uniform noise $w_i(k) \sim \text{Uniform}([-0.01, 0.01])$ for the velocity and angular velocity, and zero otherwise.

For the single particle with non-holonomic dynamics, the states are $z = (x, y, v, h) \in \mathbb{R}^4$, where (x, y) is the position, v is the velocity, and h is the heading, and the control inputs are $(a, \omega) \in \mathbb{R}^2$, where a is acceleration and ω is angular acceleration. The system dynamics are

$$f(z, u) = z + (v \cdot \cos(h), v \cdot \sin(h), a, v \cdot \tan(\omega)/\ell)$$

where ℓ is the particle radius. The goal and disturbances are the same as for the holonomic particle.

We compare RMPS with: (i) using the learned policy $\hat{\pi}$ without any shielding, and (ii) using shielding without robust control [28], which we call non-robust MPS. For each experiment, we run 50 scenarios with 3 different random seeds and compute both the probability of safety and the probability of reaching the goal. Also, we use $N = 1500$ to estimate the tightened constraints in NMPC.

Results. The safety and performance of the three algorithms are shown in Fig. 3. First, as can be seen, RMPS achieves a safety probability of 1.0. In contrast, the learned policy is frequently unsafe, demonstrating the importance of shielding. Furthermore, RMPS only slightly diminishes performance compared to the learned policy. Similarly, the non-robust MPS performs slightly better than the learned policy alone in terms of safety, but still cannot guarantee that safety holds. In contrast, RMPS guarantees safety in each environment. Its performance is slightly diminished—for the particle with non-holonomic dynamics (the hardest environment), the probability of reaching the goal drops by about 20%. Thus, RMPS is much more suitable for safety-critical systems where safety must be guaranteed.

VI. Conclusion

We have proposed an algorithm for ensuring the safety of a learned policy for a stochastic nonlinear dynamical system. We use a sampling-based approach to estimate the reachable set of the backup policy, and use results from statistical learning theory to provide theoretical guarantees on our estimates. In our experiments, we show that our approach can ensure safety without sacrificing very much performance despite these modifications. Thus, our approach is a promising way to ensure safety in safety-critical systems.

A key direction for future work is reducing the running time of our algorithm. While our approach is already computationally tractable, it is still too slow for production systems. As we described, our current implementation uses two heuristics to improve performance, but these heuristics break our theoretical guarantees. Thus, improving the performance of our algorithm would enable our algorithm to be used with its theoretical guarantees intact. Additional directions for future work include enabling safe exploration and extending our results to partially observable systems, as well as demonstrating our approach on real robotics systems.

REFERENCES

[1] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: http://arxiv.org/abs/1509.06461

[2] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[4] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016. [Online]. Available: http://arxiv.org/abs/1602.01783

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

[6] R. Mahjourian, N. Jaitly, N. Lazic, S. Levine, and R. Miikkulainen, "Hierarchical policy design for sample-efficient learning of robot table tennis through self-play," *CoRR*, vol. abs/1811.12927, 2018. [Online]. Available: http://arxiv.org/abs/1811.12927

[7] N. O. Lambert, D. S. Drew, J. Yaconelli, R. Calandra, S. Levine, and K. S. J. Pister, "Low level control of a quadrotor with deep model-based reinforcement learning," *CoRR*, vol. abs/1901.03737, 2019. [Online]. Available: http://arxiv.org/abs/1901.03737

[8] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[9] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[10] A. Khan, C. Zhang, S. Li, J. Wu, B. Schlotfeldt, S. Y. Tang, A. Ribeiro, O. Bastani, and V. Kumar, "Learning safe unlabeled multi-robot planning with motion constraints," *CoRR*, vol. abs/1907.05300, 2019. [Online]. Available: http://arxiv.org/abs/1907.05300

[11] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar, "Graph policy gradients for large scale robot control," in *CoRL*, 2019.

[12] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *CoRL*, 2019.

[13] C. Gehring, S. Coros, M. Hutler, D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, and R. Siegwart, "Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot," *IEEE Robotics & Automation Magazine*, pp. 1–1, 02 2016.

[14] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice - a survey," *Automatica*, vol. 25, pp. 335–348, 1989.

[15] J. B Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*, 01 2009.

[16] J. Ho and S. Ermon, "Generative adversarial imitation learning," *CoRR*, vol. abs/1606.03476, 2016. [Online]. Available: http://arxiv.org/abs/1606.03476

[17] J. H. Gillula and C. J. Tomlin, "Guaranteed safe online learning via reachability: tracking a ground target using a quadrotor," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2723–2730.

[18] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1424–1431.

[19] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, 2017, pp. 908–918.

[20] O. Bastani, Y. Pu, and A. Solar-Lezama, "Verifiable reinforcement learning via policy extraction," in *Advances in Neural Information Processing Systems*, 2018, pp. 2494–2504.

[21] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] J. F. Fisac, N. F. Lugovoy, V. Rubies-Royo, S. Ghosh, and C. J. Tomlin, "Bridging hamilton-jacobi safety analysis and reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8550–8556.

[23] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. ACM, 2019, pp. 169–178.

[24] M. Fazlyab, M. Morari, and G. J. Pappas, "Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming," *CoRR*, vol. abs/1903.01287, 2019. [Online]. Available: https://arxiv.org/abs/1903.01287

[25] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas, "Efficient and accurate estimation of lipschitz constants for deep neural networks," *CoRR*, vol. abs/1906.04893, 2019. [Online]. Available: http://arxiv.org/abs/1906.04893

[26] D. Tran, "Safety verification of cyber-physical systems with reinforcement learning control, emsoft 2019," 07 2019.

[27] W. Xiang, D. M. Lopez, P. Musau, and T. T. Johnson, "Reachable set estimation and verification for neural network models of nonlinear dynamic systems," *CoRR*, vol. abs/1802.03557, 2018. [Online]. Available: http://arxiv.org/abs/1802.03557

[28] O. Bastani, "Safe planning via model predictive shielding," *CoRR*, vol. abs/1905.10691, 2019. [Online]. Available: http://arxiv.org/abs/1905.10691

[29] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," *CoRR*, vol. abs/1803.08552, 2018. [Online]. Available: http://arxiv.org/abs/1803.08552

[30] W. Zhang and O. Bastani, "Mamps: Safe multi-agent reinforcement learning via model predictive shielding." [Online]. Available: https://obastani.github.io/docs/mamps.pdf

[31] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014. [Online]. Available: https://doi.org/10.1080/00423114.2014.902537

[32] S. Yu, H. Chen, and F. Allgwer, "Tube mpc scheme based on robust control invariant set with application to lipschitz nonlinear systems," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, Dec 2011, pp. 2650–2655.

[33] R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone, "A machine learning approach for real-time reachability analysis," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2202–2208.

[34] P. Reist, P. Preiswerk, and R. Tedrake, "Feedback-motion-planning with simulation-based lqr-trees," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1393–1416, 2016.

[35] D. Q. Mayne, E. C. Kerrigan, E. J. van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rnc.1758

[36] D. Mayne, "Robust and stochastic model predictive control: Are we going in the right direction?" *Annual Reviews in Control*, vol. 41, pp. 184–192, 2016.

[37] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

[38] L. Liebenwein, C. Baykal, I. Gilitschenski, S. Karaman, and D. Rus, "Sampling-based approximation algorithms for reachability analysis with provable guarantees," in *RSS*, 2018.

[39] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. I–387–I–395. [Online]. Available: http://dl.acm.org/citation.cfm?id=3044805.3044850