Solving Stabilize-Avoid Optimal Control via Epigraph Form and Deep Reinforcement Learning

Oswin So, Chuchu Fan
Massachusetts Institute of Technology
{oswinso, chuchu}@mit.edu

Abstract—Tasks for autonomous robotic systems commonly require stabilization to a desired region while maintaining safety specifications. However, solving this multi-objective problem is challenging when the dynamics are nonlinear and high-dimensional, as traditional methods do not scale well and are often limited to specific problem structures. To address this issue, we propose a novel approach to solve the stabilize-avoid problem via the solution of an infinite-horizon constrained optimal control problem (OCP). We transform the constrained OCP into epigraph form and obtain a two-stage optimization problem that optimizes over the policy in the inner problem and over an auxiliary variable in the outer problem. We then propose a new method for this formulation that combines an on-policy deep reinforcement learning algorithm with neural network regression. Our method yields better stability during training, avoids instabilities caused by saddle-point finding, and is not restricted to specific requirements on the problem structure compared to more traditional methods.

We validate our approach on different benchmark tasks, ranging from low-dimensional toy examples to an F16 fighter jet with a 17-dimensional state space. Simulation results show that our approach consistently yields controllers that match or exceed the safety of existing methods while providing ten-fold increases in stability performance from larger regions of attraction.

I. INTRODUCTION

Autonomous systems are becoming increasingly prevalent in our lives in recent years; however, designing controllers for complex dynamics systems is difficult. For example, an unmanned aerial vehicle may be required to to observe a target location while maintaining line of sight to a base station \[5\]. Another example is the problem of satellite docking that requires approaching a target satellite from a specific direction \[23\]. Robot control tasks often involve both stability and safety requirements, where the controller must both drive the system towards and remain stable within some goal region while avoiding unsafe regions. We denote this as the stabilize-avoid problem. However, synthesizing a policy that achieves both tasks in the presence of input constraints is challenging as these objectives can often be contradictory \[22\].

Reach-Avoid. Reachability analysis and the reach-avoid problem \[50\] are very closely related to the stabilize-avoid problem that we tackle in this paper. Given a dynamical system, the reach-avoid problem aims to solve for the set of initial conditions and the appropriate control policy to drive a system to a desired goal set while avoiding undesirable states. Hamilton-Jacobi analysis \[27\] provides a methodology for computing the solution to reach-avoid problems, and is conventionally solved via numerical partial differential equation (PDE) techniques that use state space discretization. These methods are limited in practice to systems with up to 5 continuous state variables \[30\]. Recent works have applied Deep Reinforcement Learning (DeepRL) to solve reach-avoid problems for higher dimensional systems (e.g., 6-dimensional system in \[21\]). While reaching a goal set is related to stabilization to a goal set, the two objectives have important differences. The goal of the stabilizing controller is to induce stability of the system within a subset of the goal set. This need not be true for the reach controller, where the goal set may not contain any equilibrium point at all. In the worst case, unwanted oscillations could be introduced into the system. We discuss the relationship between the two formulations in Section II-A.

Constrained Reinforcement Learning. Works that address the problem of task completion with safety constraints from the reinforcement learning community usually do so from the constrained Markov Decision Process (CMDP) \[5\] framework. Many of these works adopt techniques from constrained optimization to handle the additional safety constraints, of which the use of Lagrangian duality is popular due to its simplicity \[11\, 25\, 26\, 48\]. However, the CMDP formulation considers the discounted sum over constraints instead of enforcing the constraint at each state. This allows the additional constraint terms to be treated in the same way as the objective
function at the expense of allowing constraint violations. While we can modify the CMDP formulation to disallow all constraint violations, this can lead to an ill-conditioned problems. The relationship between our work and the CMDP setting is discussed in Section IV-D.

**Lyapunov Methods.** Lyapunov theory provides an attractive option for synthesizing safe, stabilizing controllers by using control Lyapunov functions (CLFs) [7, 33] and control barrier functions (CBFs) [6, 52]. CLFs and CBFs provide conditions for synthesizing controllers that are certified to be stable and safe respectively. However, they are difficult to construct analytically for general nonlinear systems [16]. CLFs / CBFs can be synthesized via convex optimization (e.g., sum-of-squares programming [2, 12, 47]). However, such approaches are limited to systems with polynomial dynamics and rely on the use of solvers for semidefinite programs which can face numerical difficulties [52]. Alternatively, neural networks can be used to synthesize these certificate functions [11, 13, 34]. However, one problem that remains is that CLFs and CBFs cannot be easily combined to yield combined safety and stability guarantees when the set of feasible controls induced by the CLF and CBF do not intersect, forcing the controller to pick one and sacrifice either safety or stability. This can lead to the presence of unwanted local minima [36]. Although this can be resolved by learning a joint Control Lyapunov Barrier Function (CLBF) [14, 37], the training process requires the ability to sample from the control-invariant set, which is difficult in the case of complex nonlinear dynamics where the control-invariant set is not known. Our experiments show that this method is difficult to apply in practice. We provide discussions on this in Section VII.

**Model Predictive Control.** Online optimization-based control methods such as model predictive control (MPC) have become increasingly popular for general-purpose control synthesis with the increase in computational power available for robotic systems. Moreover, they can be viewed as a finite-horizon approximation to an infinite horizon optimal control problem (OCP), which is closely linked to Lyapunov stability [18]. However, the constrained OCPs that need to be solved online are computationally expensive, making it difficult to achieve high frequency control updates in practice [35]. Moreover, accurate gradient information is typically necessary for solving nonlinear OCPs quickly, making it further difficult to use with dynamics that have expensive gradients or are non-differentiable. Finally, guaranteeing the recursive feasibility of MPC for general nonlinear systems is challenging [28], and in many cases, requires the solution (or approximation) of a control invariant set which can be difficult to find.

**Continuous-Time Constrained OCP.** In addition to the discrete-time formulation used MPC, there exists a number of methods that investigate the problem of constrained OCP in continuous time. Works within this area mainly focus on investigating theoretical properties of the value function for the finite and infinite horizon problems [4, 10, 24, 29, 41, 42]. We note similarities of our problem formulation to the one discussed in [4], where the continuous-time constrained optimal control problem is solved via transformation into epigraph form. An associated Hamilton-Jacobi PDE and its properties are investigated and used as the basis of a numerical PDE solver for a two-dimensional finite-horizon problem. To the best of our knowledge, the transformation of the infinite-horizon constrained OCP problem into epigraph form for discrete-time problems has not been proposed before in literature.

In this work, we solve the stabilize-avoid problem by formulating an infinite-horizon constrained OCP, inspired by the global asymptotic stability guarantees in the unconstrained case [33]. Our method for solving the constrained OCP departs from traditional Lagrangian duality based methods and uses an epigraph form which we denote as the Epigraph Form Constrained Optimal Control Problem (EF-COCP). We then solve the EF-COCP using DeepRL by deriving a corresponding policy gradient theorem and applying the proximal policy optimization (PPO) algorithm [39]. This allows us to tackle a wider range of systems compared to non-RL-based methods and handle general nonlinear non-differentiable black box dynamics with minimal computational cost online.

**Contributions.** We summarize our contributions below.

- We propose a new formulation of the safety-constrained OCP via an epigraphic reformulation (EF-COCP) which is easier to interpret and avoids the optimization instability of existing Lagrangian duality methods.
- We derive a policy-gradient theorem for the inner problem of EF-COCP and propose the EF-PPO algorithm for solving the stabilize-avoid problem using DeepRL.
- The proposed EF-PPO method is validated on a range of challenging systems, yielding promising empirical results on complex systems such as a 17-dimensional F16 fighter jet, visualized in Figure 1.

II. THE STABILIZE-AVOID PROBLEM

We consider arbitrary nonlinear discrete-time dynamical systems of the form

\[ x_{k+1} = f(x_k, u_k) \]  

where \( x \in \mathcal{X} \subseteq \mathbb{R}^n \), \( u \in \mathcal{U} \subseteq \mathbb{R}^m \) and \( f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X} \). In this paper, we consider the following control synthesis problem.

**Problem 1 (Stabilize-Avoid Problem).** Given a nonlinear system with a goal set \( \mathcal{G} \subseteq \mathcal{X} \) and avoid set \( \mathcal{A} \subseteq \mathcal{X} \), find a control policy \( u = \pi(x) \) that maximizes the size of the set \( \mathcal{R} \subseteq \mathcal{X} \) defined as the set of initial states \( x_0 \) such that all trajectories started from \( x_0 \) evolving under the dynamics

\[ x_{k+1} = f(x_k, \pi(x_k)), \quad x_0 \in \mathcal{R}, \]  

also satisfy the following two properties.

Stabilize: \( \limsup_{k \to \infty} \min_{y \in \mathcal{G}} \|x_k - y\| = 0 \).

Avoid: \( x_k \notin \mathcal{A} \) for all \( k \geq 0 \).
D. The reach controller can reach the goal region at \( p \) will satisfy the reach-avoid problem but not the stabilize-avoid \( G \) (red line) faster than stabilize (compare magenta triangles), but the controller never stabilizes to \( G \) and induces a periodic orbit. The stable controller reaches \( G \) slower but remains in the set.

In short, the objective is to reach and (asymptotically) stabilize to a goal set \( G \) while avoiding the set of unsafe states \( A \).

A. Relationship with the Reach-Avoid Problem

Note that we use \( \lim \sup \) when defining stabilize instead of minimizing over time as in the reach formulation

\[
\min_k \min_{y \in \mathcal{U}} \| x_k - y \| = 0. \tag{3}
\]

A system that enters but then subsequently exits the goal set \( G \) will satisfy the reach-avoid problem but not the stabilize-avoid problem, as we illustrate in the following example.

Example 1 (Stabilize vs Reach). Consider a double-integrator with states \( x = [p, v] \in \mathbb{R}^2 \), control \( u \in \mathbb{R}^1 \), and the following task specification:

\[
\begin{align*}
\text{Constraints} & : \quad |u| \leq 1, \quad A := \emptyset \\
\text{Goal} & : \quad G := \{ x \mid p = 1.0 \}
\end{align*}
\]

One solution to the reach problem (minimizing the first hitting time) yields a controller with periodic orbits, while minimizing the stabilize objective yields a globally stabilizing controller on \( G \) (Figure 2).

We note that both formulations yield similar results when \( G \) consists only of equilibrium points. However, the specifications for \( G \) may include non-equilibrium points. Applying reach to this problem to obtain stability would require the set of equilibrium points for arbitrary nonlinear dynamics, which is a challenging task in itself to find and may not even exist. In this work, we consider dynamics for which such a set is not known \( a \ priori \). Hence, applying reach-avoid methods to the stabilize-avoid problem may not give desirable results.

III. STABILIZE-AVOID AS INFINITE-HORIZON CONSTRAINTED OPTIMAL CONTROL PROBLEM

We tackle Problem 1 by solving an infinite-horizon constrained OCP. To motivate this problem formulation, let \( l : \mathcal{X} \to \mathbb{R}_{\geq 0} \) denote a \textit{non-negative} cost function that takes zero value on \( G \) and is positive outside \( G \), and define the infinite-horizon \textit{undiscounted} policy value function \( V^{l,\pi} \) for an arbitrary policy \( \pi \) as

\[
V^{l,\pi}(x_0) := \sum_{k=0}^{\infty} l(x_k), \quad x_{k+1} = f(x_k, \pi(x_k)). \tag{4}
\]

Since \( l \) is non-negative, \( V^{l,\pi} \) is also non-negative. Using dynamic programming principles, we obtain

\[
V^{l,\pi}(x_k) = l(x_k) + V^{l,\pi}(f(x_k, \pi(x_k))). \tag{5}
\]

The above equations are very close to satisfying the conditions for a discrete-time Lyapunov function [13]. By imposing additional assumptions on \( l \), we can show that \( V^{l,\pi} \) is a Lyapunov function. We leave the proof in the Appendix A1 for conciseness. Note that, for a given \( \pi \), the set over which stability holds may be very small or even empty. This motivates us to ask whether this set can be maximized. The answer is affirmative here: solving the undiscounted infinite-horizon OCP

\[
\min_\pi \sum_{k=0}^{\infty} l(x_k) \tag{6a}
\]

\[
s.t. \quad x_{k+1} = f(x_k, \pi(x_k)), \tag{6b}
\]

gives a globally asymptotically stabilizing controller under some mild assumptions on the cost function and the controllability of the dynamics [33]. This provides an answer to Problem 1 when safety constraints are not considered.

However, solving an \textit{unconstrained} infinite-horizon OCP does not guarantee satisfaction of the safety constraints \( x_k \notin A \). Hence, we consider solving a \textit{constrained} infinite-horizon OCP to obtain a policy that is safe by construction. Let the superlevel set of \( h : \mathcal{X} \to \mathbb{R} \) describes the avoid set

\[
A := \{ x : h(x) > 0 \}. \tag{7}
\]

We then solve the following \textit{constrained} infinite-horizon OCP

\[
\min_\pi \sum_{k=0}^{\infty} l(x_k) \tag{8a}
\]

\[
s.t. \quad x_{k+1} = f(x_k, \pi(x_k)), \tag{8b}
\]

\[
h(x_k) \leq 0, \quad k \geq 0. \tag{8c}
\]

In the constrained setting, the proof from [33] that the optimal policy is globally asymptotically stabilizing is not applicable here. However, the proof can be extended to handle this case under certain conditions, which we leave as future work.

IV. SAFETY CONSTRAINED OPTIMAL CONTROL VIA EPIGRAPHIC REFORMULATION

The previous section describes how solving the stabilize-avoid problem (Problem 1) can be cast as solving an infinite-horizon constrained OCP (6). In this section, we propose solving the constrained OCP by reformulating the problem into its epigraph form and then solving the resulting two-stage optimization problem.

---

Fig. 2: Trajectory and value functions comparing a solution to the stabilize and reach objectives of the double integrator in 1D. The reach controller can reach the goal region at \( p = 1 \) (red line) faster than stabilize (compare magenta triangles), but the controller never stabilizes to \( G \) (green line) and induces a periodic orbit. The stable controller reaches \( G \) slower but remains in the set.
A. Epigraph Form

For any optimization problem of the form

\[ \begin{align*}
\min_x & \quad J(x) \\
\text{s.t.} & \quad h(x) \leq 0,
\end{align*} \tag{9a} \]

the epigraph form [8, pp 134] of the above (9) is the optimization problem

\[ \begin{align*}
\min_x z & \quad \text{s.t.} \quad h(x) \leq 0, \\
& \quad J(x) \leq z.
\end{align*} \tag{10a} \]

where \( z \in \mathbb{R} \) is an auxiliary optimization variable. It is a standard result in optimization [8, pp 134] that (9) and (10) are equivalent. Now, observe that the constraints (10b) and (10c) can be combined to yield the following:

\[ \begin{align*}
\min_{x, z} & \quad z \\
\text{s.t.} & \quad \max\{h(x), J(x) - z\} \leq 0.
\end{align*} \tag{11a} \]

We can further move the minimization of the \( x \) variable into the constraint (11b) (see Appendix A2 for proof) to yield

\[ \begin{align*}
\min_{z} & \quad z \\
\text{s.t.} & \quad \min_{x} \max\{h(x), J(x) - z\} \leq 0.
\end{align*} \tag{12a} \]

This form allows us to convert the original constrained problem (9) into an unconstrained inner problem over \( x \) (12b) and a constrained outer problem (12c) over the scalar decision variable \( z \). At the optimal point \( (x^*, z^*) \), optimality conditions imply that \( z^* = J(x^*) \). Solving for \( z^* \) can thus be thought of as solving for the cost \( J \) at the optimal solution. Consequently, if we can bound the value of \( J(x^*) \), then \( z^* \) will lie within the same bound. This facilitates treating \( z \) as a “cost budget” (with units of \( J \)) for satisfying \( h \). As \( z \to \infty \) (i.e., the “cost budget” for \( J(x) \) increases), \( h(x) \) will dominate the \( \max \), and \( x^* \) will focus on minimizing \( h \) more. On the other hand, as \( z \to -\infty \) (i.e., the “cost budget” for \( J(x) \) decreases), \( J(x) - z \) will dominate the \( \max \), and \( x^* \) will focus on minimizing \( J \).

B. Epigraph Form Constrained OCP

We now apply this to the constrained OCP (8). First, we express the safety constraint (8c) equivalently as

\[ \max_{k \geq 0} h(x_k) \leq 0. \tag{13} \]

Using this, the epigraph form of the constrained OCP (8) reads

\[ \begin{align*}
\min_{z} & \quad z \\
\text{s.t.} & \quad \tilde{V}(x_0, z) \leq 0,
\end{align*} \tag{14a} \]

where the auxiliary value function \( \tilde{V} \) is the OCP analogue of the LHS of (12b)

\[ \tilde{V}(x_0, z) := \min_{\pi} \tilde{J}^\pi(x_0, z) \]

\[ \text{s.t.} \quad x_{k+1} = f(x_k, \pi(x_k)), \tag{15} \]

with \( \tilde{J}^\pi \) defined as

\[ \tilde{J}^\pi(x_0, z) := \max_{k \geq 0} \left\{ \max_{l=0}^{\infty} l(x_k) - z \right\}. \tag{16} \]

We denote this the epigraph form constrained OCP (EF-COCP).

C. Dynamic Programming for EF-COCP

Note that (16) has both a maximization and a sum and hence has a different structure compared to the single sum in the objective function of the typical OCP. Consequently, the Bellman equation cannot be used in this case. We derive the corresponding dynamic programming equations below as

\[ \tilde{V}(x_k, z_k) = \min_{u_k} \max_{\pi} \left\{ h(x_k), V(x_{k+1}, z_{k+1}) \right\}, \tag{17} \]

where \( z_{k+1} \) has the following “dynamics”

\[ z_{k+1} = z_k - l(x_k). \tag{18} \]

This can again be understood from the intuition of \( z \) as a “cost budget” in \( J \) for satisfying the constraints \( h \). Moving from timestep \( k \) to \( k+1 \) incurs the cost \( l(x_k) \), which is subtracted from the current “budget” \( z_k \) to yield the next budget \( z_{k+1} \). If the “budget” \( z_k \) falls low enough, the cost term in (16) will dominate the \( \max \), and we will have “run out of budget” to focus on constraint satisfaction.

D. Relationship with Lagrangian Duality

The epigraph form (11a) shares some similarities with the Lagrangian duality formulation commonly used in (undiscounted) constrained MDPs

\[ \max_{\lambda \geq 0} \min_{\pi} \sum_{k=0}^{\infty} l(x_k) + \lambda \sum_{k=0}^{\infty} [h(x_k)]^+, \tag{19} \]

where \([\cdot]^+ = \max(0, \cdot)\), is used to disallow constraints violation. Both formulations (19), (14) are two-stage optimization problems, where the outer problem consists of an extra scalar variable \( z, \lambda \) respectively while the inner problem optimizes with respect to the policy. The inner problem for both formulations is shown in Figure 3. We note the following two differences.

Optimization Stability. Due to the \([\cdot]^+ \) in (19), gradients of the inner problem with respect to \( \lambda \) will always be non-negative. Consequently, as long as the constraints are not satisfied, \( \lambda \) will continue to increase. However, large values of \( \lambda \) are problematic when constraints are not satisfied, since the gradients \( \nabla_x \mathcal{L} \) with respect to \( x \) scale linearly in \( \lambda \). In [45], a solution to this problem is proposed by rescaling \( \mathcal{L} \) by \( 1/(1 + \lambda) \). However, when \( \lambda \) is large, the gradients for the \( l \) terms will instead vanish. Moreover, since \( \lambda \) is a non-decreasing function of the number of optimization iterations, this problem will only become worse as optimization proceeds.

In contrast, since \( z \) is additive within the \( \max \), the scale of gradients is not altered. Consequently, EF-COCP does not suffer from this issue of optimization instability.
Intuition. The auxiliary variable \( z \) in the epigraph form is in units of cost and represents a cost budget, as shown earlier. On the other hand, the Lagrange multiplier \( \lambda \) is a ratio representing the cost per unit constraint, but this is harder to interpret when cost and constraints cannot be easily compared. Consequently, it is much easier to estimate upper bounds for \( z \). We take advantage of this to bound the range of \( z \) used for solving the inner problem, which we discuss in the next section.

V. SOLVING EF-COCP WITH DEEP REINFORCEMENT LEARNING

The previous section introduces a new epigraph form of the constrained optimal control problem, but does not provide a method of solving this formulation. In this section, we tackle this problem via reinforcement learning and introduce a framework for learning controllers for complex, nonlinear, potentially non-smooth dynamics.

Given that the inner optimization problem of the epigraph form \([15]\) still retains many similarities with the original problem, we choose to solve for the policy and the value function using reinforcement learning. Specifically, we use Proximal Policy Optimization (PPO) \([39]\) but with modified definitions of the value function, advantage functions, returns, and generalized advantage estimator (GAE) \([38]\).

While we have treated the policy \( \pi \) as a deterministic function, we will use a stochastic \( \pi \) for the purpose of improved exploration while performing Deep RL, where \( \pi(a|x) \) now defines a distribution over controls. However, while \( E[a + b] = E[a] + E[b] \), the same does not hold over the max operator used to define \( J^\pi \) in \([16]\). Consequently, we need to be careful when defining \( \hat{V}^\pi \) for a stochastic policy such that an analogous dynamic programming equation to the deterministic case \([17]\) can be applied. Consider the following nested expectation form of an OCP policy value function:

\[
\begin{align*}
\hat{V}^\pi(x_0) & := \lim_{K \to \infty} E_0 \left[ \sum_{k=0}^{K} I_k \right], \\
& = \lim_{K \to \infty} E_0 \left[ I_0 + \cdots + I_{K-1} + I_K \right], \\
& = E_0 \left[ I_0 + \hat{V}^\pi(x_1) \right],
\end{align*}
\]

where, for conciseness, we denote

\[
E_k := E_{u_k|x_k}, \quad E_{k,t} := E_{u_{k+1},\ldots,u_T|x_k}, \quad l_k := l(x_k).
\]

Note how the nested expectations of \((20b)\) lends itself to the dynamic programming equations of \((20c)\). In the case of EF-COCP, we define \( \hat{V}^\pi \) analogously to obtain

\[
\hat{V}^\pi(x_0, z_0) = \lim_{K \to \infty} E_0 \left[ h_{K-1} \vee E_K \left[ E_{K} \left[ K \vee \sum_{k=0}^{K} I_k - z_0 \right] \right] \right],
\]

\[
= E_0 \left[ h_0 \vee \hat{V}^\pi(x_1, z_1) \right],
\]

where we have used \( a \vee b := \max(a, b) \) for conciseness and where \( x_k, z_k \) follow the dynamics \((1)\) and \((15)\). We also define the action-value function \( \hat{Q}^\pi(x, z, u) = \max \left( h(x), \hat{V}^\pi(x_{k+1}, z_{k+1}) \right) \).

Theorem 1 (Policy Gradient Theorem). The gradient of the policy value function \( \hat{V}^\pi(x_0, z_0) \) for the inner subproblem satisfies

\[
\nabla_\theta \hat{V}^\pi(x_0, z_0) = \nabla_\theta \left( h(x) + \hat{V}^\pi(x_{k+1}) \right) = \nabla_\theta \hat{V}^\pi(x_{k+1}),
\]

where the binary random variable \( \xi_k \) is defined to be equal to 1 when \( h(x_t) \leq V^\pi(x_{t+1}, z_{t+1}) \) is true for all \( t = 0, \ldots, k \).

Proof: The proof follows from the proof of the normal policy gradient theorem \([46]\), differing only in the expression for \( \nabla_\theta \hat{Q}^\pi \). In the normal setting \([46], (20c)\) gives

\[
\nabla_\theta \hat{Q}^\pi(x_k, u) = \nabla_\theta \left( h(x_k) + \hat{V}^\pi(x_{k+1}) \right) = \nabla_\theta \hat{V}^\pi(x_{k+1}).
\]

In the case of EF-COCP with \((22c)\),

\[
\nabla_\theta \hat{Q}^\pi(x_k, u) = \nabla_\theta \max \left( h(x_k), \hat{V}^\pi(x_{k+1}, z_{k+1}) \right) = \nabla_\theta \hat{V}^\pi(x_{k+1}, z_{k+1}).
\]
Following the rest of the normal proof [46] then yields [24].

From Theorem 1, we can construct a basic on-policy DeepRL algorithm to solve the inner problem (15) over all states \( x \) parametrized by a range of \( z \), yielding a learned (stochastic) controller \( \pi_\theta(x, z) \). However, we can do better by performing variance reduction via subtracting the baseline \( \hat{V}^\pi(x, z) \) to get the advantage \( \hat{A}^\pi(x, z, u) := Q^\pi(x, z, u) - \hat{V}(x, z) \), as \( \mathbb{E}_{\pi}(\nabla_\theta \ln \pi_\theta) = 0 \). Following PPO, we also apply the GAE estimate [38], perform clipped importance sampling and add in an entropy bonus to arrive at an algorithm that is very similar to PPO [39] but with \( \hat{Q}^\pi \) and \( \hat{V}^\pi \) defined as above.

**Stochastic policy considerations:** While we have performed the above developments using a stochastic policy and derived a stochastic policy gradient theorem, our desired solution to the inner optimization problem is a deterministic controller. Hence, we only take the mode of the learned policy, and treat the stochasticity purely as a means of performing exploration, discouraging premature convergence to local minima and smoothing the optimization landscape [8] [19] [31]. We also fine-tune the learned value function at the end of the DeepRL training by freezing the obtained deterministic policy and performing policy evaluation.

It can be appealing to consider using a deterministic policy and apply a deterministic policy gradient theorem [40]. However, the proof of this theorem requires the transition distribution \( p(x_{t+1}|x_t, u_t) \) to be continuous [40]. This does not hold in our problem since the transition distribution is degenerate (and hence discontinuous) due to the use of deterministic dynamics, i.e.,

\[
p(x_{t+1}|x_t, u_t) = \delta(x_{t+1} - f(x_t, u_t)),
\]

preventing the use of deterministic policy gradient in this case. After obtaining \( \hat{V}^\pi \) and \( \pi \) for the deterministic policy, we now turn to the outer problem (14). Since the outer optimization problem is only 1 dimensional, we can solve for \( z^* \) easily via classical scalar optimization methods such as the bisection method which run quickly. Instead of running bisection online, however, we choose to learn the optimal \( z^*: \mathcal{X} \to \mathbb{R} \) offline by using the result of bisection as the label for a regression problem. This gives us \( z^*(x) \), which in turn provides the optimal policy for the original constrained optimal control problem.

**Algorithm 1:** EF-PPO Inner Problem

**input:** Estimate of maximum cost upper bound \( z_{\text{max}} \)

repeat

if reset environments then

Sample random \( x_0 \in \mathcal{X} \) and \( z_0 \in [0, z_{\text{max}}] \)

end if

Run policy \( \pi \) in environments for \( T \) timesteps

Compute advantage estimates \( A^\pi \) with GAE

Update policy \( \pi \) and baseline \( \hat{V}^\pi \) with PPO clipping and entropy

until converged

Fine-tune \( \hat{V}^\pi \) using the mode of \( \pi \) via policy evaluation.

**Algorithm 2:** EF-PPO Outer Problem

**input:** Estimate of maximum cost upper bound \( z_{\text{max}} \)

Sample dataset of \( x \) randomly from the state space

Bisect \( \hat{V}^\pi(x, \cdot) \) over \([0, z_{\text{max}}]\) to obtain labels \( z^* \) for each \( x \)

repeat

Train network \( \hat{z}^*(x) \) to predict \( z^* \) given \( x \) via regression

until converged

The proposed EF-PPO algorithm is summarized in Algorithms 1 and 2 and illustrated in Figure 4. EF-PPO solves the two-stage optimization problem of EF-COCP (14a) sequentially. The inner problem Equation (15) is solved via policy gradient using Theorem 1 for a range of \( z \) values and uses the improvements in PPO such as GAE estimates, clipped importance sampling, and an entropy bonus. Next, we extract the mode of the stochastic \( \pi \) and fine-tune \( \hat{V}^\pi \). Then, we fix the value function \( \hat{V}^\pi \) and the deterministic policy \( \pi(x, z) \) and learn \( z^*(x) \) by randomly sampling states in the state space and minimizing the residual to the analytical solution of (14) found by applying the bisection method. Since this is a 1D optimization problem, the bisection method converges to almost machine precision within tens of iterations. The final policy is then obtained as \( \pi(x, z^*(x)) \). When solving the inner problem, we randomly sample states from the state space and random sample \( z \) within the range \([0, z_{\text{max}}]\), where \( z_{\text{max}} \) is an upper-bound estimate of the total cost \( \sum_{k=0}^\infty f(x_k) \). To prevent using \( z_{\text{max}} = \infty \) since this term may be unbounded, we introduce a small discount factor only in the total costs. More details on the effect of discounting are available in the Appendix C.
VI. EXPERIMENTS

To evaluate the performance of the proposed EFPPO algorithm, we compare EFPPO against related algorithms on simulated stabilize-avoid problems with increasing complexity. The last problem involves a non-differentiable, nonconvex, non-control-affine system and demonstrates the ability of our approach to both maintain safety and successfully stabilize the system within the goal region even in nontrivial high-dimensional environments. We compare EFPPO against the following baselines methods.

- **PPO** [39], a popular on-policy unconstrained DeepRL method. Despite its unconstrained nature, it is common to apply “soft constraints” in the form of penalties incurred when an undesirable state is reached [9, 51]. Unlike constrained methods such as EFPPO, the scale of the penalties is a hyperparameter that must be tuned to trade-off between constraint violation and training stability. We denote by PPO(λ) the family of methods where λ denotes the penalty scale of the modified cost function \( \hat{l} \), i.e.,

\[
\hat{l}(x) = l(x) + \lambda [h(x)]^+, \tag{29}
\]

where PPO(0) solves the unconstrained problem.

- **CPPO** [45], a representative algorithm among the family of constrained DeepRL methods which use Lagrangian duality, and is an improvement on PPO-Lagrangian from [1]. As noted in Section IV-D the constrained MDP formulation allows constraint violations up to a cost threshold. To learn policies that strictly satisfy the constraints, we set the cost threshold to 0. We perform a manual hyperparameter search to select the PID parameters.

- **PPO-SIS** [26], a constrained RL method that applies the safety constraint at each state instead of in expectation as in CMDP. Moreover, a safety certificate is learned jointly to improve the safety of the learned policy.

- **CLBF [14]**, which learns a Lyapunov function using a neural network via losses which penalize violations of the Lyapunov conditions. Safety can be guaranteed by enforcing that a sublevel set of the learned neural Lyapunov function lies outside the avoid set. Given a learned CLBF, we consider two methods for synthesizing the controller: (1) CLBF(QP) solves the CLBF-QP using the CLBF as a constraint; (2) CLBF(Opt) applies the (bang-bang) control that minimizes \( \tilde{V} \).

As noted previously, while the reach-avoid problem is closely related to the stabilize-avoid problem considered in this work, the policies obtained from reach-avoid do not induce stability when the goal set is not an equilibrium point (see Section II-A).

For a fair comparison, a feedforward neural network with tanh activations is used for both the policy (if used) and value functions. Additional networks are also defined using the same architecture above but with modified final activation function (e.g., softplus) when used in the original implementation.

---

**TABLE I: Comparison of controller performance on the double integrator example. Metrics are computed from states randomly sampled from the control-invariant set.**

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Safety Rate ( \uparrow )</th>
<th>Cost ( \downarrow )</th>
<th>Stabilize Rate ( \uparrow )</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPO(0)</td>
<td>0.55</td>
<td>1.251</td>
<td>1.0</td>
</tr>
<tr>
<td>PPO(1)</td>
<td>0.61</td>
<td>1.283</td>
<td>1.0</td>
</tr>
<tr>
<td>PPO(10)</td>
<td>1.0</td>
<td>1.299</td>
<td>1.0</td>
</tr>
<tr>
<td>CPPO [45]</td>
<td>1.0</td>
<td>1.314</td>
<td>1.0</td>
</tr>
<tr>
<td>PPO-SIS [26]</td>
<td>1.0</td>
<td>2.227</td>
<td>0.182</td>
</tr>
<tr>
<td>CLBF(QP) [14]</td>
<td>0.96</td>
<td>3.042</td>
<td>0.180</td>
</tr>
<tr>
<td>CLBF(Opt) [14]</td>
<td>0.99</td>
<td>3.180</td>
<td>0.035</td>
</tr>
<tr>
<td>EFPPO (Ours)</td>
<td>1.0</td>
<td>1.285</td>
<td>1.0</td>
</tr>
</tbody>
</table>

To compare each method, we use the safety rate, cost and stabilize rate metrics computed by rolling out the learned policy on randomly sampled initial states. When the true control-invariant set that can guarantee safety is analytically known, the sampled states are sampled from the control-invariant set. However, for systems with complex dynamics where such a set cannot be found analytically, we sample from a crude box estimate. The safety rate and stabilize rate are defined as the fraction of trajectories that satisfy the safety constraints and can reach and stay within the goal set for the last 50 timesteps respectively. The top method for each metric is shown in **bold**. For the cost, we only highlight the best performing method among the methods that handle constraints (i.e., unconstrained PPO is not taken into account).

More details regarding the specific dynamics and constraints for each task are provided in the Appendix D for brevity.

A. 1D Double-Integrator

We first consider a simple double-integrator dynamics in 1D, where the optimal policy and corresponding optimal control-invariant region for the problem can be computed. Given states \( \{p, v\} \in \mathbb{R}^2 \), controls \( a = u \in \mathbb{R} \) the task here is to stabilize to the region \( p_{\text{goal}} := [0.65, 0.85] \) while satisfying box constraints on both states and controls

\[
|p| \leq 1, \quad |v| \leq 1, \quad |u| \leq 1. \tag{30}
\]

While all constrained methods are able to maintain safety for all states within the control-invariant region, only EFPPO is able to also stabilize all trajectories to the goal set \( \mathcal{G} \) with a cost similar to the unconstrained solution from PPO(0) (see Table I). PPO-SIS has regions of the state space outside \( \mathcal{G} \) which are equilibrium points (see Figure 5), while CPPO has a much larger cost compared to the unconstrained solution.

The unconstrained PPO(λ) methods require the penalty weight \( \lambda \) to be large enough for a safe policy, but this trade-off comes at the cost of the policy’s performance and requires careful tuning of the penalty weight \( \lambda \). In contrast, EFPPO is able to synthesize a performant safe controller without requiring any cost function tuning.
is smaller than the complement of the avoid set.

We suspect that this is insufficient for tasks where the avoid set $A$ is relatively complex. Finally, we see that PPO(10) learns a particularly poor performing controller due to the large penalty weights. In particular, the difference in scales between the large costs in the unsafe regions and the small costs near the goal set poses a challenge for learning the value function accurately, resulting in the learned policy putting more weight on constraint satisfaction.

### B. 2D Single-Integrator with Sector Obstacle

Next, we consider a single-integrator in 2D where the avoid set is defined as a sector of the circle and the goal set defined in the center. The state is defined as the positions $[p_x, p_y] \in \mathbb{R}^2$ with controls denoting the velocities $[v_n, v_t] \in [-1, 1]^2 \subset \mathbb{R}^2$, where $v_n$ and $v_t$ denote the normal and tangent components of the velocity vector to the center of the circle.

Table II summarizes the results in this task. Due to the control parametrization and the shape of the avoid set, the optimal control at all states is the constant vector $[1, 0]$. Of the constrained methods, only EFPPO is able to learn this, with other methods learning suboptimal versions of the optimal policy. Note that the rollouts for CLBF (14) actually move away from the center when starting near the sector obstacle on the right (see Figure 6). This is because the CLBF learns a distorted metric near the obstacles (as seen from the value function) to ensure the level-set is contained within the control-invariant set. Moreover, we see that SIS also fails to reliably stabilize on this simple example. The safety index used in SIS (26) is taken from (54) and consists of only three parameters.

We suspect that this is insufficient for tasks where the avoid set $A$ is relatively complex. Finally, we see that PPO(10) learns a particularly poor performing controller due to the large penalty weights. In particular, the difference in scales between the large costs in the unsafe regions and the small costs near the goal set poses a challenge for learning the value function accurately, resulting in the learned policy putting more weight on constraint satisfaction.

### C. Hopper Stabilization

The preceding benchmarks are low-dimensional toy-examples that provide intuition on how the different methods behave for simple linear systems. Our next example demonstrates the ability of EFPPO to stabilize to goal sets while maintaining safety for more complex dynamics. We consider the hopper in the Brax simulator, a nonlinear non-differentiable system with a 12 dimensional state space and a 3 dimensional control space. Note that the original dynamics only consider a 2 dimensional state space as they discard the x-coordinate, which we keep. The goal set $\mathcal{G}$ in this task is defined as

$$
\mathcal{G} := \{ \mathbf{x} \mid p_x \in [2.8, 3.0] \},
$$

while the constraints limit the height and rotation of the hopper’s torso

$$
p_x \geq 0.7, \quad |\theta| \leq 0.2.
$$

Unlike the normal setup for Hopper, stabilization to $\mathcal{G}$ requires keeping track of the x-position. Consequently, the optimal policy is no longer a limit-cycle and even requires the hopper to move backwards for some initial states. We report our results in Table III. On this problem, we see that both CPPO
and EFPPO achieve high safety rates, while PPO-SIS fails to stabilize. We see the same trend of PPO(λ) presenting a trade-off between stability and safety, except for PPO(10) which is more unsafe than PPO(1). We suspect this is due to larger costs destabilizing training.

D. F16 Ground Collision Avoidance In a Low Altitude Flight Corridor

Finally, we showcase the scalability of our method on a ground collision avoidance example involving the F16 fighter jet [20]. This system is non control-affine, non-smooth and involve lookup tables, making this a challenging system to solve stabilize-avoid on in addition to the high 17-dimensional state space and 4-dimensional control space. The task here is to stabilize the F16 to a target altitude defined by the set $[50, 150]$ ft under box control constraints while staying within a flight corridor heading North (box constraints on the East-Up plane). For positions $(p_E, p_N, p_U)$, this corresponds to

$$-200 \text{ ft} \leq p_E \leq 200 \text{ ft}, \quad 0 \text{ ft} \leq p_U \leq 1000 \text{ ft.}$$

The results are summarized in Table IV and Figure 8. Following the trend from before, we see that EFPPO is able to achieve high safety rates at the expense of being unable to stabilize to the goal set. In contrast, EFPPO using the optimal $z^*$ achieves similar safety rates as CPPO but has a 70-fold increase in the stabilize rate. Moreover, note that EFPPO(0) and EFPPO(1.8) denote using a constant value of $z$ for the policies from the inner EFCOCP problem [15]. As expected, using a larger value of $z$ results in improved constraint satisfaction at the cost of a lower stabilize rate. By using the optimal value of $z^*$, EFPPO($z^*$) is able to achieve the high stabilize rate of EFPPO(0) while maintaining the high safety rates of EFPPO(1.8). However, note that the safety rate of EFPPO($z^*$) is actually slightly lower than that of EFPPO(1.8). This is because $z^*$ is approximated via a neural network which may have small approximation errors. We also see the same trends for PPO(λ) from Hopper carry over to this task.

VII. DISCUSSION

Many Lagrangian duality-based methods suffer from focusing too much on satisfying the safety constraints and consequently suffer in terms of their stability. As noted in Section IV-D, the non-negative gradients of the Lagrange multipliers $\lambda$ means that they will continue to increase as long as constraints are not being satisfied. However, a large $\lambda$ causes the optimization problem to become badly conditioned, hindering the performance of these algorithms. Moreover, in systems with complex dynamics where the control-invariant set is not known, it may be impossible for constraints satisfaction to occur in all states. In this case, $\lambda$ will grow unbounded until either the training algorithm becomes unstable or it hits a user-defined maximum limit. This is a failure that the PID mechanism in CPPO does not fix, as $\lambda$ does not oscillate. We found all three coefficients of the PID played a similar role in adjusting the rate at which $\lambda$ (monotonically) increases and eventually destabilizes training. Although the CMDP problem formulation can be adapted to solve stabilize-avoid problems, these methods are a poor fit.

Additionally, we observe that the CLBF performs poorly even on both toy examples. We believe there are two reasons for this.
Fig. 8: Ground collision avoidance and stabilization to a low altitude flight corridor for the F16 fighter jet system.

1) Learning of the CLBF assumes that the control-invariant set is known (or can be sampled from) a-priori. However, in the experiments considered in this work, we do not assume knowledge of these sets. While we do give CLBF a small region around the equilibrium subset of the goal region, information that was not available to other algorithms, this was not enough for CLBFs to perform well.

2) The learning problem formulated in [14] is under-parametrized. More specifically, given any CLBF $V$ which satisfies the CLBF conditions, $\alpha V$ will induce the same set of feasible constraints and will perform identically to $V$. However, if $V$ does not satisfy the CLBF conditions, the violation error can be reduced arbitrarily by taking $\alpha \to 0$. This yields a value function where regions close to the goal set have near-zero gradients which may violate the CLBF conditions and compromise the controller’s performance, which we have observed empirically in our experiments.

In contrast, the proposed EF-PPO algorithm can perform well even when large areas of the state space are not control-invariant as in our experiments. This is due to two reasons: (a) $z$ does not affect the magnitude of the gradients directly, unlike $\lambda$ which scales the gradients of the constraint (and also the cost, if the reweighing scheme from [45] is used). Consequently, while $z \to \infty$ will make no difference to EF-PPO (if the $z$ feature variable is normalized correctly), taking $\lambda \to \infty$ will cause the Lagrangian to diverge and cause training instabilities. (b) Since $z$ can be interpreted as a “cost budget” and it is easier to estimate an upper-bound on costs, we can afford to solve the two-stage optimization problem sequentially. In contrast, it is much harder to bound the optimal $\lambda$ since the units of “cost to constraint ratio” is more difficult to reason about.

VIII. CONCLUSION

We present a new method for synthesizing nonlinear feedback controllers for performing stabilization while maintaining safety under control constraints. By formulating the stabilize-avoid problem as an infinite-horizon epigraph-form constrained optimal control problem and applying deep reinforcement learning, our approach is able to sidestep numerical challenges that other methods face and achieve vastly larger regions of attraction while still maintaining safety for high dimensional complex systems.

Limitations and future work: The EF-PPO algorithm currently splits the task of learning $V, \pi$ and $z^*$ into separate stages of optimization, and relies on random sampling of $z$ in the first stage to cover the state-space. However, given the structure of the epigraph form constrained OCP, it should be possible to perform both optimizations simultaneously such that only a single stage of optimization is required. Moreover, the current method does not account for model errors and parametric uncertainties which may compromise the safety and stability of the learned controllers. An extension of the current method to consider this would allow for more robustness when deploying such a controller to the hardware systems in the real world. Additionally, PPO is an on-policy online reinforcement learning algorithm. Consequently, a simulator of the dynamics is necessary to solve the EF-COCP problem. Extending this work to off-policy offline reinforcement learning setting will allow our method to be applied to settings when a dynamics simulator is not available. Finally, it is difficult to provide useful statements about the convergence of practical DeepRL algorithms. Nevertheless, it is important to understand properties of the learned policy, especially as violations of safety constraints are serious and undesirable in any system. We leave this as future work.

ACKNOWLEDGMENTS

This work is partially supported by the MIT Lincoln Lab under the Safety in Aerobic Flight Regimes (SAFR) program. However, this article solely reflects the opinions and conclusions of its authors and not the MIT Lincoln Lab.


