Safe Reinforcement Learning using Finite-Horizon Gradient-based Estimation

Juntao Dai [1 2] **Yaodong Yang** [3] **Qian Zheng** [1 2] **Gang Pan** [1 2]

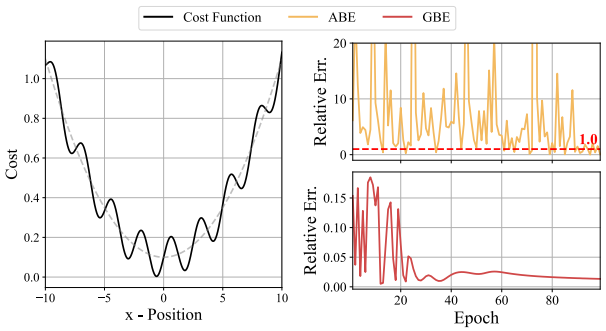
Abstract

A key aspect of Safe Reinforcement Learning (Safe RL) involves estimating the constraint condition for the next policy, which is crucial for guiding the optimization of safe policy updates. However, the existing *Advantage-based Estimation* (ABE) method relies on the infinite-horizon discounted advantage function. This dependence leads to catastrophic errors in finite-horizon scenarios with non-discounted constraints, resulting in safety-violation updates. In response, we propose the first estimation method for finite-horizon non-discounted constraints in deep Safe RL, termed *Gradient-based Estimation* (GBE), which relies on the analytic gradient derived along trajectories. Our theoretical and empirical analyses demonstrate that GBE can effectively estimate constraint changes over a finite horizon. Constructing a surrogate optimization problem with GBE, we developed a novel Safe RL algorithm called *Constrained Gradient-based Policy Optimization* (CGPO). CGPO identifies feasible optimal policies by iteratively resolving sub-problems within trust regions. Our empirical results reveal that CGPO, unlike baseline algorithms, successfully estimates the constraint functions of subsequent policies, thereby ensuring the efficiency and feasibility of each update.

1. Introduction

Reinforcement Learning (RL) (Sutton & Barto, 2018) stands as a powerful paradigm in artificial intelligence. Over the past few years, RL has achieved notable success across various challenging tasks, such as video games (ElDahshan et al., 2022), robotic control (Okamura et al., 2000; Singh et al., 2022), Go (Silver et al., 2016; 2017), and the training of large language models (Ouyang et al., 2022; Rafailov et al., 2023). Recently, there has been a growing emphasis on prioritizing the safety of policy learning. This shift is driven by the critical need for safety in real-world applications, such as autonomous driving (Muhammad et al., 2020) and service robots (Bogue, 2017). In response, Safe RL (Garcıa & Fernández, 2015) has emerged as a related paradigm, aiming to provide reliable and robust policy learning in the face of complex and dynamic environments. The Constrained Markov Decision Process (CMDP) (Altman, 1999) stands as a foundational framework of Safe RL, augmenting the traditional Markov Decision Process (MDP) (Puterman, 1990) by incorporating constraints.

In most Safe RL benchmarks (Ray et al., 2019; Gronauer, 2022; Ji et al., 2023b), the common practical constraints are represented as non-discounted sums over finite trajectories, subject to scalar thresholds. However, existing deep Safe RL algorithms uniformly apply an infinite-horizon approach, the *Advantage-based Estimation* (ABE) (Achiam et al., 2017), across all types of constraints. This fundamental discrepancy hinders these algorithms' ability to accurately predict the constraint values for subsequent policies,



(a) Cost Function (b) Relative Estimation Errors

Figure 1. ***Advantage-based Estimation*** **fails even in simple environments under finite-horizon constraints.** (a) The cost obtained by the agent while traversing along the x-axis, namely, $c_t = c(x_t)$. (b) Relative errors in the estimation of changes in the finite-horizon cumulative constraint (i.e., $\sum_{t=1}^{\top} c_t \leq b$). The ABE method generates relative errors even greater than 1.0, showing completely incorrect estimations. Refer to Appendix C for more details.

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou, China [2]The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou, China [3]Center for AI Safety and Governance, Peking University, Beijing, China. Correspondence to: Qian Zheng <qianzheng@zju.edu.cn>, Gang Pan <gpan@zju.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

thereby misleading the optimization direction in constrained problems. Our straightforward experiment illustrates this issue in Figure 1. Despite the simplicity of the task, the ABE method generates relative errors exceeding 1.0.

To bridge the gap in estimating finite-horizon constraints, we introduce a new estimation methodology, *Gradient-based Estimation* (GBE). Unlike previous methods, GBE avoids reliance on the infinite-horizon assumption, instead leveraging first-order gradients derived along finite trajectories (Mohamed et al., 2020). This approach facilitates precise estimation of constraints, particularly where a non-discounted cumulative sum over a finite horizon is compared against a scalar threshold. Utilizing the GBE, we construct the constrained surrogate problem and develop a novel Safe RL algorithm, called *Constrained Gradient-based Policy Optimization* (CGPO). Additionally, informed by an error analysis of GBE, we implement a trust region approach in the parameter space to mitigate errors and further ensure the feasibility of policy updates.

Our contributions are threefold: **(1)** We introduce the GBE method, designed to estimate finite-horizon non-discounted constraints in Safe RL tasks, along with relevant theoretical analysis. **(2)** We propose the CGPO algorithm. To our best knowledge, CGPO is the first deep Safe RL algorithm that can effectively address tasks with finite-horizon constraints. Based on precise estimations of the GBE method, CGPO ensures the efficiency and feasibility of each update. Our analysis includes worst-case scenarios and introduces an adaptive trust region radius strategy for improved performance. **(3)** We develop a series of Safe RL tasks with differentiable dynamics to test our algorithm. Comparative evaluations demonstrate our algorithm's superior update efficiency and constraint satisfaction against baselines.

2. Related Works

Differentiable RL. In many RL applications, the knowledge of the underlying transition function facilitates policy optimization through analytical gradients, a method known as Differentiable RL (Mohamed et al., 2020; Jaisson, 2022). The development of differentiable simulators (Degrave et al., 2019; Werling et al., 2021; Xian et al., 2023), which represent systems as differentiable computational graphs, significantly advance this research area. Some well-known differentiable RL algorithms include BPTT (Mozer, 1995), POSD (Mora et al., 2021), and SHAC (Jie Xu et al., 2022). Moreover, differentiable RL techniques extend to model-based algorithms that access analytical gradients from World Models (Clavera et al., 2020; As et al., 2022; Parmas et al., 2023b). Our algorithm, consistent with other Differentiable RL methods, depends on either a differentiable simulator or the World Model method for modeling task environments. Considering the swift advancements in related techniques, we view this requirement optimistically. For further discussion, see Section 7.

Safe RL. The work closely related to ours focuses on on-policy deep Safe RL algorithms, which are categorized into Lagrangian and Convex Optimization methods (Xu et al., 2022). Lagrangian methods, such as PDO (Chow et al., 2018) and CPPO-PID (Stooke et al., 2020), alternate between addressing the primal and the dual problems. The latest APPO (Dai et al., 2023) mitigates the oscillatory issue by augmenting a simple quadratic term. Convex Optimization methods, such as CPO (Achiam et al., 2017) and CVPO (Liu et al., 2022), optimize based on the solution to the primal problem. The latest method, CUP (Yang et al., 2022), divides updates into two steps: updating in the steepest direction first, then mapping into the feasible domain if constraints are violated. Above all methods use estimates of objective and constraint functions to formulate surrogate problems (Achiam et al., 2017), which we refer to as *Advantage-based Estimation*. Thus, the accuracy of estimates impacts the efficiency and feasibility of updates.

However, research on finite-horizon constraints has been limited to non-deep Safe RL contexts (Kalagarla et al., 2021; Guin & Bhatnagar, 2023). Despite their prevalence in the Safe RL Benchmark, aforementioned deep Safe RL algorithms treat these constraints as if they were infinite-horizon (such as Safety-Gym (Ray et al., 2019), Bullet-Safety-Gym (Gronauer, 2022), Safety-Gymnasium (Ji et al., 2023b), and OmniSafe (Ji et al., 2023c)). This leads to poor constraint satisfaction within these benchmarks.

3. Preliminaries

3.1. Constrained Markov Decision Process

RL is typically framed as a Markov Decision Process (MDP) (Puterman, 2014), denoted $\mathcal{M} \triangleq \langle \mathcal{S}, \mathcal{A}, r, P, \mu_0 \rangle$, which encompasses the state space $\mathcal{S}$, the action space $\mathcal{A}$, a reward function r, the transition probability P, and the initial state distribution μ_0. A stationary policy π represents a probability distribution defining the likelihood of taking action a in state s. The set Π symbolizes the collection of all such stationary policies. The primary objective of RL is to optimize the performance metric typically formulated as the total expected return over a finite horizon T, namely, $\mathcal{J}_R(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} r(s_t, a_t)\right]$.

Generally, Safe RL is formulated as a Constrained MDP (CMDP) (Altman, 1999), $\mathcal{M} \cup \mathcal{C}$, augmenting the standard MDP $\mathcal{M}$ with an additional set of constraints $\mathcal{C}$. This constraint set $\mathcal{C} = \{(c_i, b_i)\}_{i=1}^m$ consists of pairs of cost functions c_i and corresponding thresholds b_i. The constraint function is defined as the cumulative cost over the horizon T, $\mathcal{J}_{C_i}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{T-1} c_i(s_t, a_t)\right]$, and the feasible

policy set is $\Pi_{\mathcal{C}} = \cap_{i=1}^{m}\{\pi \in \Pi \mid \mathcal{J}_{C_i}(\pi) \leq b_i\}$. The objective of Safe RL is to find the optimal feasible policy, $\pi^\star = \arg\max_{\pi\in\Pi_{\mathcal{C}}} \mathcal{J}_R(\pi)$.

3.2. Advantage-based Estimation Method

By introducing the discount factor γ, we define the objective function for an infinite-horizon scenario as $\mathcal{J}_R^\gamma(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)]$ and the constraint function as $\mathcal{J}_{C_i}^\gamma(\pi) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t c_i(s_t,a_t)]$. We express the infinite-horizon value function as $V_\pi^\gamma(s) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s]$ and the state-action value function as $Q_\pi^\gamma(s,a) = \mathbb{E}_{\tau\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s, a_0 = a]$. The infinite-horizon advantage function is $A_\pi^\gamma(s,a) = Q_\pi^\gamma(s,a) - V_\pi^\gamma(s)$. The discounted future state distribution d_π^γ is denoted as $d_\pi^\gamma(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t P(s_t = s \mid \pi)$. Then, the difference in some metrics between two policies π, π' can be derived as (Kakade & Langford, 2002):

$$\mathcal{J}_f^\gamma(\pi') - \mathcal{J}_f^\gamma(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_{\pi'}^\gamma \\ a\sim\pi'}}\Big[A_\pi^\gamma(s,a)\Big], \quad (1)$$

where $\mathcal{J}_f^\gamma$ represents infinite-horizon $\mathcal{J}_R^\gamma$, $\mathcal{J}_C^\gamma$. Equation (1) is difficult to estimate since $s \sim d_{\pi'}^\gamma$ is unknown. Achiam et al. (2017) approximates $s \sim d_{\pi'}^\gamma$ with $s \sim d_\pi^\gamma$ and employs importance sampling techniques to derive the following estimation:

$$\bar{\mathcal{J}}_f^\gamma(\pi') \triangleq \mathcal{J}_f^\gamma(\pi) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d_\pi^\gamma \\ a\sim\pi}}\Big[\frac{\pi'(s,a)}{\pi(s,a)}A^\pi(s,a)\Big], \quad (2)$$

where d^π and $A^\pi(s,a)$ are both defined in the infinite-horizon format and it requires that $\gamma \neq 1$. Thus, algorithms based on Equation (2) have to treat all constraints as if they were infinite-horizon.

3.3. Policy Optimization with Differentiable Dynamics

Differentiable simulators (Freeman et al., 2021) represent physical rules using a differentiable computational graph, $\boldsymbol{s}_{t+1} = \mathcal{F}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, allowing them to participate in gradient back-propagation process (Mohamed et al., 2020). The fundamental approach for policy optimization with differentiable physical dynamics is Back-propagation Through Time (BPTT) (Mozer, 2013). Multiple trajectories $\{\tau_i\}_{i=1}^N$ are collected from and then derivation is performed along the trajectory. Then, the loss function is

$$\mathcal{L}^{\text{BPTT}} = -\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=0}^{T-1} r(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i). \quad (3)$$

The Short-Horizon Actor-Critic (SHAC) approach (Jie Xu et al., 2022) modifies BPTT by ultilizing the value function V to segment trajectories into sub-windows of length h:

$$\mathcal{L}^{\text{SHAC}} = -\frac{1}{Nh}\sum_{i=1}^{N}\Big[\sum_{t=t_0}^{t_0+h-1}\gamma^{t-t_0} r(\boldsymbol{s}_t^i, \boldsymbol{a}_t^i) + \gamma^h V(\mathbf{s}_{t_0+h}^i)\Big]. \quad (4)$$

This change limits the maximum length for gradient back-propagation to h, thereby mitigating the issues of gradient vanishing/exploding and making the training more stable.

4. Constrained Surrogate Problem using Gradient-based Estimation

Differentiable environments represent physical dynamics as differentiable computational graphs, allowing first-order gradients of objective and constraint functions to be derived via gradient back-propagation along trajectories. In this section, we propose a new estimation method based on this feature, called *Gradient-based Estimation* (GBE). It facilitates more accurate approximations for the next police's objective and constraint functions, which leads to a new constrained surrogate problem for solving Safe RL.

4.1. Gradient-based Estimation for Objective and Constraint Functions

Consider a parameterized policy π_θ within a parameter space Θ, e.g., represented by a neural network. In the context of differentiable environments, both the objective function $\mathcal{J}_R(\boldsymbol{\theta})$ and the constraint function $\mathcal{J}_C(\boldsymbol{\theta})$ can be regarded as differentiable over Θ. Consequently, we can compute the first-order gradients of them along the trajectories, denoted as $\nabla_{\boldsymbol{\theta}}\mathcal{J}_R(\boldsymbol{\theta})$ and $\nabla_{\boldsymbol{\theta}}\mathcal{J}_C(\boldsymbol{\theta})$, through back-propagation. For simplicity and generality, we will focus on a single constraint scenario, though the method applies to multiple constraints via corresponding matrix operations.

For a minor update $\boldsymbol{\delta}$ in the policy parameter space, transitioning from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_0 + \boldsymbol{\delta}$, consider the function $\mathcal{J}_f$, which encapsulates both the objective function $\mathcal{J}_R$ and the constraint function $\mathcal{J}_C$. We perform a first-order Taylor expansion of $\mathcal{J}_f$ at $\boldsymbol{\theta}_0$ to obtain:

$$\begin{aligned}\mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_f(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top\nabla_{\boldsymbol{\theta}}\mathcal{J}_f(\boldsymbol{\theta}_0) \\ + \frac{1}{2}\boldsymbol{\delta}^\top\nabla_{\boldsymbol{\theta}}^2\mathcal{J}_f(\boldsymbol{\theta}_0 + t\boldsymbol{\delta})\boldsymbol{\delta}.\end{aligned} \quad (5)$$

Wherein, $t \in (0,1)$ and $\frac{1}{2}\boldsymbol{\delta}^\top\nabla_{\boldsymbol{\theta}}^2\mathcal{J}_f(\boldsymbol{\theta}_0 + t\boldsymbol{\delta})\boldsymbol{\delta}$ represents the Peano remainder term, which is $o(\|\boldsymbol{\delta}\|)$. When $\boldsymbol{\delta}$ is sufficiently small, the remainder term becomes negligible. Therefore, we propose the *Gradient-based Estimation* method to estimate the values of the objective and constraint functions after a minor update $\boldsymbol{\delta}$ from $\boldsymbol{\theta}_0$ as follows:

$$\hat{\mathcal{J}}_R(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) \triangleq \mathcal{J}_R(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top\nabla_{\boldsymbol{\theta}}\mathcal{J}_R(\boldsymbol{\theta}_0), \quad (6)$$

$$\hat{\mathcal{J}}_C(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) \triangleq \mathcal{J}_C(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top\nabla_{\boldsymbol{\theta}}\mathcal{J}_C(\boldsymbol{\theta}_0). \quad (7)$$

Regarding the error analysis for these estimates, the following lemma states:

Lemma 4.1. *Assume $\boldsymbol{\theta}_0 \in \Theta$ and $\mathcal{J}_f(\boldsymbol{\theta})$ is twice differentiable in a neighborhood surrounding $\boldsymbol{\theta}_0$. Let $\boldsymbol{\delta}$ be a small update from $\boldsymbol{\theta}_0$. If we estimate $\mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})$ as $\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_f(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0)$ and given that $\epsilon = \max_{t\in(0,1)} \left|\nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_0 + t\boldsymbol{\delta})\right|$, the estimation error can be bounded as:*

$$\left|\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})\right| \leq \frac{1}{2}\epsilon\|\boldsymbol{\delta}\|_2^2. \tag{8}$$

Proof. The primary source of estimation error is the neglect of higher-order infinitesimal remainders. Thus, the error is $\left|\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})\right| = \frac{1}{2}\left|\boldsymbol{\delta}^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0 + t\boldsymbol{\delta})\boldsymbol{\delta}\right| \leq \frac{1}{2}\epsilon\|\boldsymbol{\delta}\|_2^2$. See Appendix A.1 for more details. □

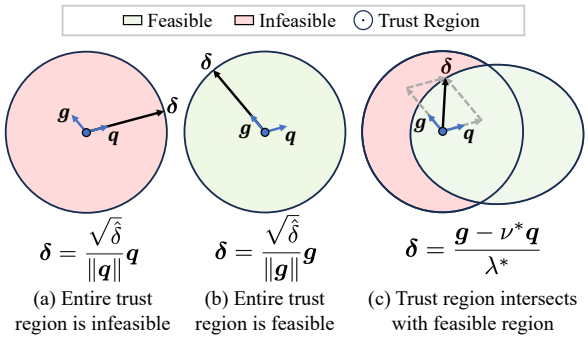
This finding indicates that as policy parameters are updated from $\boldsymbol{\theta}_0$ to $\boldsymbol{\theta}_0 + \boldsymbol{\delta}$, the upper bounds of the estimation errors for both the objective and constraint functions are positively correlated with the square of the L_2 norm of the update vector $\boldsymbol{\delta}$. Consequently, when employing $\hat{\mathcal{J}}_R$ and $\hat{\mathcal{J}}_C$ as surrogate objective and constraint functions, meticulous control over the magnitude of these updates becomes essential. By carefully managing $\|\boldsymbol{\delta}\|_2^2$, we could ensure that the estimation error remains within an acceptable range, thus facilitating both effective performance improvement and precise adherence to constraints.

4.2. Constrained Surrogate Problem within Trust Region

Based on the analysis of Theorem 4.1, the idea of controlling the error by managing $\|\boldsymbol{\delta}\|_2^2$ naturally aligns with the concept of trust regions (Schulman et al., 2015; Meng et al., 2022). Given a trust region radius $\hat{\delta}$, we suppose that updated parameter $\boldsymbol{\theta_{k+1}}$ of the k^{th} iteration within the trust region $\Theta_k = \left\{\boldsymbol{\theta} \in \Theta \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2 \leq \hat{\delta}\right\}$ are credible.

By employing the approximations of the objective function in Equation (6) and the constraint function in Equation (7), We transform the solution of the primal Safe RL problem into an iterative process of solving a series of sub-problems within predefined trust regions. Given the initial policy parameter $\boldsymbol{\theta}_k$ of the k^{th} iteration, our sub-problem targets finding the next optimal and credible parameter $\boldsymbol{\theta}_{k+1} \in \Theta_k$, which not only maximize the surrogate objective function $\hat{\mathcal{J}}_R(\boldsymbol{\theta}_{k+1})$ but also conform to the surrogate constraint $\hat{\mathcal{J}}_C(\boldsymbol{\theta}_{k+1}) \leq b$. Thus, the surrogate sub-problem within a given trust region at k^{th} iteration can be represented as:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}\in\Theta} \quad & (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) \\ \text{s.t.} \quad & \mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) \leq b \\ & (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \leq \hat{\delta}. \end{aligned} \tag{9}$$



(a) Entire trust region is infeasible (b) Entire trust region is feasible (c) Trust region intersects with feasible region

Figure 2. The computational relationship between the policy update $\boldsymbol{\delta}$, the gradient of the objective function $\boldsymbol{g}$, and the gradient of the constraint function $\boldsymbol{g}$ varies in three scenarios.

5. Constrained Gradient-based Policy Optimization

Based on the constrained surrogate sub-problem in Equation (9), we develop a novel Safe RL algorithm named *Constrained Gradient-based Policy Optimization* (CGPO).

5.1. Solution to Surrogate Sub-problem

Notations. Considering the k^{th} iteration within the trust region Θ_k, we introduce additional notations to make the discussion more concise: $\boldsymbol{g}_k \triangleq \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k)$, $\boldsymbol{g}_k \triangleq \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k)$, $c_k \triangleq \mathcal{J}_C(\boldsymbol{\theta}_k) - b$, and $\boldsymbol{\delta} \triangleq \boldsymbol{\theta} - \boldsymbol{\theta}_k$. With these definitions, we rewrite the sub-problem (9) within the trust region:

$$\max_{\boldsymbol{\delta}} \ \boldsymbol{g}_k^\top \boldsymbol{\delta} \quad \text{s.t. } c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} \leq 0, \quad \boldsymbol{\delta}^\top \boldsymbol{\delta} \leq \hat{\delta} \tag{10}$$

Since the sub-problem (10) may have no solution, we first discuss the conditions under which this problem is unsolvable. The following theorem holds:

Theorem 5.1 (Solvability Conditions for the Sub-problem)**.** *The sub-problem (10) is unsolvable if and only if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ and $c_k > 0$.*

Proof. Consider whether there is an intersection between the trust region and the feasible half-space. For a detailed proof, refer to Appendix A.2.1. □

Through a similar proof, we arrive at the following corollary:

Corollary 5.2. *$\boldsymbol{\theta}$ is deemed feasible for every $\boldsymbol{\theta}$ within Θ_k if and only if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ and $c_k \leq 0$ are both satisfied.*

Based on Theorem 5.1 and Corollary 5.2, we solve for $\boldsymbol{\theta}_{k+1}$ of sub-problem (9) in three different scenarios, as illustrated in Figure 2.

(a) If $c^2/\boldsymbol{q}^\top\boldsymbol{q} - \hat{\delta} > 0$ and $c > 0$, the entire trust region is infeasible. We update the policy along the direction of the steepest descent of the constraint function, namely, $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\sqrt{\hat{\delta}}}{\|\boldsymbol{q}_k\|}\boldsymbol{q}_k$.

(b) If $c^2/\boldsymbol{q}^\top\boldsymbol{q} - \hat{\delta} > 0$ and $c \leq 0$, the trust region lies entirely within the constraint-satisfying half-space. Similarly, we update along the direction of the steepest ascent of the objective function, namely, $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\sqrt{\hat{\delta}}}{\|\boldsymbol{g}_k\|}\boldsymbol{g}_k$.

(c) In other cases, the trust region partially intersects with the feasible region, so we have to solve the constrained quadratic sub-problem (10).

The sub-problem (10) is characterized as convex. When the trust region is partially feasible, the existence of at least one strictly feasible point is guaranteed. Consequently, strong duality holds as the conditions of Slater's theorem are fulfilled. By introducing two dual variables λ and ν, we construct the dual function of the primal problem as follows:

$$L(\boldsymbol{\delta}, \lambda, \nu) = -\boldsymbol{g}_k^\top\boldsymbol{\delta} + \nu\left(c_k + \boldsymbol{q}_k^\top\boldsymbol{\delta}\right) + \frac{\lambda}{2}\left(\boldsymbol{\delta}^\top\boldsymbol{\delta} - \hat{\delta}\right) \tag{11}$$

Since the sub-problem (10) satisfies the strong duality condition, any pair of primal and dual optimal points $(\boldsymbol{\delta}_k^*, \lambda_k^*, \nu_k^*)$ must satisfy the KKT conditions, namely,

$$\nabla_{\boldsymbol{\delta}} L(\boldsymbol{\delta}, \lambda, \nu) = -\boldsymbol{g}_k + \nu\boldsymbol{q}_k + \lambda\boldsymbol{\delta} = 0, \tag{12}$$

$$\nu\left(c_k + \boldsymbol{q}_k^\top\boldsymbol{\delta}\right) = 0, \tag{13}$$

$$\lambda\left(\boldsymbol{\delta}^\top\boldsymbol{\delta} - \hat{\delta}\right) = 0, \tag{14}$$

$$c_k + \boldsymbol{q}_k^\top\boldsymbol{\delta} \leq 0, \quad \boldsymbol{\delta}^\top\boldsymbol{\delta} - \hat{\delta} \leq 0, \quad \nu \geq 0, \quad \lambda \geq 0. \tag{15}$$

Note that the optimal dual variables (λ_k^*, ν_k^*) for Equation (12)-(15) can be directly expressed as a function of $\boldsymbol{g}_k$ and $\boldsymbol{q}_k$. We provide their pseudo-code in Algorithm 2. For a detailed solution process of Equation (12)-(15), refer to Appendix A.2.2. Then, following the derivation from Equation (12), we update the policy by $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\boldsymbol{g}_k - \nu_k^*\boldsymbol{q}_k}{\lambda_k^*}$.

So far, the three scenarios for updating policies by solving the surrogate sub-problem (9) have been fully presented. This forms the core component of our CGPO algorithm.

5.2. Worst-Case Analysis

In the following theorem, we detail the bounds for performance update and constraint violations under the worst-case scenarios after the policy update which results from solving the surrogate sub-problem (9).

Theorem 5.3 (Worst-Case Performance Update and Constraint Violation)**.** *Suppose $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_{k+1} \in \Theta$ are related by Equation (9). If $\boldsymbol{\theta}_k$ is feasible, a lower bound on the policy performance difference between $\boldsymbol{\theta}_{k+1}$ and $\boldsymbol{\theta}_k$ is*

$$\mathcal{J}_R(\boldsymbol{\theta}_{k+1}) - \mathcal{J}_R(\boldsymbol{\theta}_k) \geq -\frac{1}{2}\epsilon_k^R\hat{\delta} \tag{16}$$

where $\epsilon_k^R = \max_{t\in(0,1)}\left|\nabla^2_{\boldsymbol{\theta}}\mathcal{J}_R(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k))\right|$.

An upper bound on the constraint objective function of $\boldsymbol{\theta}_{k+1}$ is

$$\mathcal{J}_C(\boldsymbol{\theta}_{k+1}) \leq b + \frac{1}{2}\epsilon_k^C\hat{\delta}, \tag{17}$$

where $\epsilon_k^C = \max_{t\in(0,1)}\left|\nabla^2_{\boldsymbol{\theta}}\mathcal{J}_C(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k))\right|$.

Proof. See Appendix A.3. □

Theorem 5.3 ensures that the application of the update method purposed in Section 5.1, even in the worst-case scenario, will not drastically reduce performance or severely violate constraints with each iteration. This provides theoretical support for the stability and reliability of the algorithm.

Moreover, according to Theorem 5.3, the bounds for performance update and constraint violation are related to ϵ_k^R and ϵ_k^C, which are challenging to obtain in practice. Therefore, we indirectly mitigate their impact through an adaptive radius $\hat{\delta}$. Based on Equation (16) and (17), we define two new metrics, the reduction ratio ρ_k for the objective $\mathcal{J}_R$ and the toleration ratio ζ_k for the constraint, namely, $\rho_k \triangleq \frac{\mathcal{J}_R(\boldsymbol{\theta}_k) - \mathcal{J}_R(\boldsymbol{\theta}_{k+1})}{\hat{\mathcal{J}}_R(\boldsymbol{\theta}_k) - \hat{\mathcal{J}}_R(\boldsymbol{\theta}_{k+1})}$, $\zeta_k \triangleq \frac{|b - \mathcal{J}_C(\boldsymbol{\theta}_{k+1})|}{\left|\hat{\mathcal{J}}_C(\boldsymbol{\theta}_{k+1}) - \hat{\mathcal{J}}_C(\boldsymbol{\theta}_{k+1})\right|}$.

The reduction ratio ρ_k evaluates the consistency between estimated and actual changes in the objective function. The toleration ratio ζ_k assesses error tolerance in the constraint satisfaction, suggesting more cautious updates as it approaches the threshold. If ρ_k and ζ_k fall short of expectations, we adjust the update radius $\hat{\delta}$ accordingly. Given thresholds $0 < \eta_1 < \eta_2 < 1$ and update rates $0 < \beta_1 < 1 < \beta_2$ within the $\hat{\delta}$ range of $[\underline{\delta}, \overline{\delta}]$, we derive the following $\hat{\delta}$ update rule:

$$\hat{\delta}_{k+1} = \begin{cases} \max(\beta_1\hat{\delta}_k,\ \underline{\delta}) & \text{if } \rho_k < \eta_1 \mid \zeta_k < \eta_1, \\ \hat{\delta}_k & \text{otherwise}, \\ \min(\beta_2\hat{\delta}_k,\ \overline{\delta}) & \text{if } \rho_k \geq \eta_2\ \&\ \zeta_k \geq \eta_2. \end{cases} \tag{18}$$

The above adaptive approach can be considered a plugin for CGPO, transforming a challenging-to-tune parameter into several more manageable parameters, thereby enhancing the algorithm's stability and convergence.

5.3. Practical Implementation

Given our focus on employing gradients to solve Safe RL problems, our algorithm can incorporate any differentiable method to compute gradients $\boldsymbol{g_k}$ and $\boldsymbol{q_k}$. Various methods are available, broadly classified into two categories: the

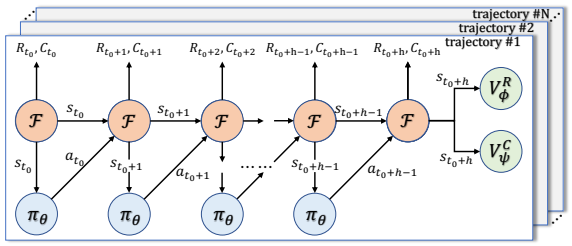


Figure 3. Gradient computation graph for the short-horizon approach. Here, $\mathcal{F}(\boldsymbol{s})$ represents the differentiable dynamics of the environment, R is the reward signal, C is the cost signal, π_θ is the parameterized policy, and V_ϕ^R and V_ϕ^C are the value functions for the return and constraint.

Zero-order Batch Gradient (ZoBG) method and the First-order Batch Gradient (FoBG) method (Suh et al., 2022). In this section, we present an example implementation of CGPO based on a variant of SHAC (Jie Xu et al., 2022), which is a FoBG method. Due to space limitations, we leave the discussion comparing ZoBG and FoBG in the Appendix B. Our empirical results show that the advantages of FoBG are more suited for the Safe RL field.

The short-horizon approach from SHAC segments the entire trajectory into sub-windows using the value function, improving differentiability and smoothing the optimization space. In constrained tasks, we need to compute the objective gradients $\boldsymbol{g_k}$ and the constraint gradients $\boldsymbol{q_k}$ within the same computational graph. Consequently, the entire computational graph, as Figure 3, is associated with the corresponding loss of objective and constraint functions:

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{t=t_0}^{t_0+h-1} r(\mathbf{s}_t^i, \mathbf{a}_t^i) + V_\phi^R(\mathbf{s}_{t_0+h}^i)\Big], \quad (19)$$

$$\mathcal{L}_C(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{t=t_0}^{t_0+h-1} c(\mathbf{s}_t^i, \mathbf{a}_t^i) + V_\psi^C(\mathbf{s}_{t_0+h}^i)\Big]. \quad (20)$$

Please see Appendix A.4 for detailed calculations.

After updating the policy $\pi_{\boldsymbol{\theta}}$, we use these trajectory $\{\tau_i\}_{i=1}^N$ to update critic networks V_π^R and V_ψ^C. Considering that the constraint function is an accumulative sum over a finite trajectory, it is necessary for the time step t to be accepted by the critic network to construct value functions $V(s_t^i, t)$ for finite-horizon return.

$$L_V^\lambda = \frac{1}{Nh}\sum_{i=1}^{N}\sum_{t=0}^{h-1}\Big(G_t^{\lambda,i} - V(s_t^i, t)\Big)^2 \quad (21)$$

where $G_t^{\lambda,i}$ is the estimated value computed through the TD(λ) formulation (Sutton & Barto, 2018). Refer to Appendix A.5 for more details.

The pseudo-code of CGPO is provided in Algorithm 1.

Algorithm 1 Constrained Gradient-based Policy Optimization (CGPO)

Input: Initialize policy $\pi_{\boldsymbol{\theta}_0}$, critic $V_{\boldsymbol{\phi}_0}$ and $V_{\boldsymbol{\psi}_0}^c$, radius $\hat{\delta}_0$, and number of iterations K.
for $k = 1, 2, \ldots, K$ **do**
  Sample a set of trajectories $\mathcal{D} = \{\tau\} \sim \pi_{\boldsymbol{\theta}_k}$.
  Compute the $\boldsymbol{g}_k$, $\boldsymbol{q}_k$ using (19) and (20).
  if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta}_k \geq 0$ and $c_k > 0$ **then**
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\sqrt{\hat{\delta}_k}}{\|\boldsymbol{q}_k\|}\boldsymbol{q}_k$.
  else if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta}_k \geq 0$ and $c_k < 0$ **then**
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\sqrt{\hat{\delta}_k}}{\|\boldsymbol{q}_k\|}\boldsymbol{q}_k$.
  else
    Compute dual variables λ_k^*, ν_k^* using Algorithm 2.
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\boldsymbol{g}_k - \nu_k^* \boldsymbol{q}_k}{\lambda_k^*}$.
  end if
  Update $V_{\boldsymbol{\phi}_k}$, $V_{\boldsymbol{\psi}_k}^c$ using (21), and $\hat{\delta}_{k+1}$ using (18).
end for
Output: Policy $\pi_{\boldsymbol{\theta}_K}$.

6. Experiments

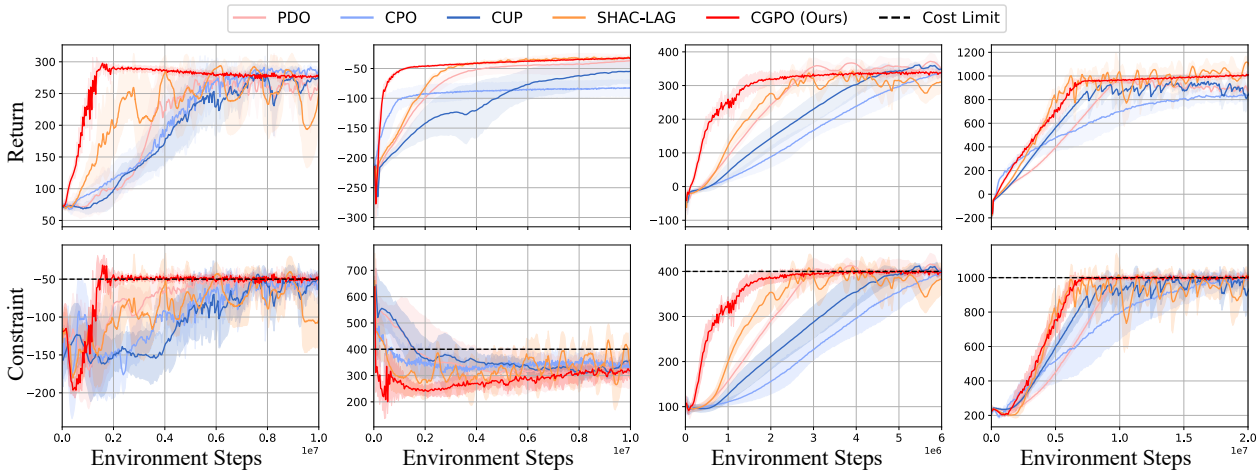
We conduct experiments to validate the effectiveness of our proposed CGPO. Our focus is primarily on four aspects:

- CGPO outperforms the baseline algorithms in Safe RL, demonstrating more efficient performance improvement and more precise constraint satisfaction (Section 6.2).
- CGPO employs the GBE method to obtain accurate estimations, unlike the ABE method fails (Section 6.3).
- CGPO can overcome the differentiability requirements through Model-based approaches (Section 6.4).
- CGPO can achieve more stable constraint convergence through an adaptive trust region radius (Section 6.5).

6.1. Experimental Details

Please refer to Appendix E for the detailed implementation of experimental tasks and Appendix F for baseline algorithms.

Differentiable Safe RL environments. Due to the lack of differentiable Safe RL environments, we develop a series of constrained differentiable tasks on an open-source differentiable physics engine Brax (Freeman et al., 2021). These tasks are based on four differentiable robotic control tasks in Brax (CartPole, Reacher, Half Cheetah, and Ant), with the addition of two common constraints: limiting position(Achiam et al., 2017; Ji et al., 2023b) and limiting velocity(Zhang et al., 2020). It is important to note that while adding these constraints, we maintained the differentiability of the physical dynamics.



(a) CartPole (b) Reacher (c) Half Cheetah (d) Ant

Figure 4. Training curves of certain algorithms on different tasks, showing episodic return and constraint for 5 random seeds. Solid lines represent the mean, while the shaded areas indicate variance, without any smoothing to the curves. CGPO demonstrates superior efficiency in improvement and constraint satisfaction. The rest of the training curves can be found in Appendix D.1.

Table 1. The number of environmental steps to converge to a feasible optimal solution (Conv. Steps), and the proportion of constraint violations during updates in safety-critical areas (Vio. Ratio) for various algorithms. CGPO significantly outperforms others in both metrics. The calculation methods of metrics are explained in Appendix F.2.

	CartPole		Reacher		HalfCheetah		Ant	
Algorithms	Conv. Steps ↓	Vio. Ratio (%) ↓	Conv. Steps ↓	Vio. Ratio (%) ↓	Conv. Steps ↓	Vio. Ratio (%) ↓	Conv. Steps ↓	Vio. Ratio (%) ↓
PDO	4.99e+06	12.01	3.99e+06	64.42	3.43e+06	74.58	1.70e+07	51.84
APPO	5.95e+06	6.59	4.22e+06	46.04	3.58e+06	18.33	1.08e+07	9.54
CPO	6.72e+06	5.96	1.98e+06	30.68	5.81e+06	-†	1.69e+07	24.56
CUP	6.84e+06	19.21	7.76e+06	56.12	5.12e+06	57.40	9.29e+06	29.79
BPTT-LAG	3.84e+06	23.99	2.73e+06	67.88	2.32e+06	61.56	2.13e+07	-†
SHAC-LAG	2.32e+06	46.31	2.96e+06	50.59	3.15e+06	65.90	1.67e+07	57.53
CGPO	**1.77e+06**	**3.96**	**1.15e+06**	**14.29**	**2.02e+06**	**7.31**	**8.52e+06**	**6.44**

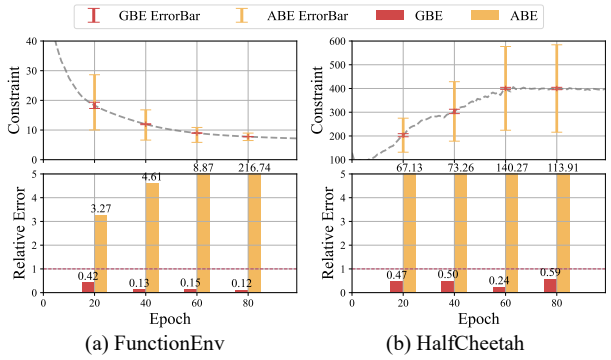
†: The empty entries result from the corresponding algorithms failing to find the optimal feasible policy, leading to a lack of updates in the safety-critical region.

Baselines. We compared CGPO with two categories of algorithms. First, we compare it with traditional Safe RL methods, including both classic and latest Primal-Dual methods: PDO (Chow et al., 2018) and APPO (Dai et al., 2023), as well as classic and latest Primal methods: CPO (Achiam et al., 2017) and CUP (Yang et al., 2020). Secondly, we combined the current SOTA differentiable algorithms, BPTT (Mozer, 2013) and SHAC (Jie Xu et al., 2022), with the Lagrangian method to satisfy constraints. Specifically, we used the Lagrange multiplier λ to trade off the objective $\mathcal{J}_R$ and the constraint $\mathcal{J}_C$. The refined algorithms are later termed as BPTT-Lag and SHAC-Lag.

6.2. Overall Performance

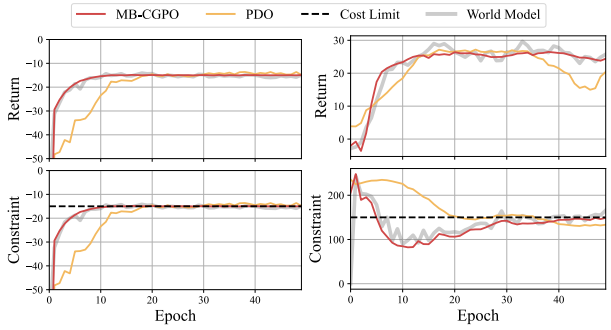
Figure 4 shows the learning curves of different algorithms across various tasks. We observe that CGPO not only converges to a constraint-satisfying policy more quickly and stably than baseline algorithms but also demonstrates enhanced efficiency in improving performance.

Compared to conventional Safe RL algorithms, CGPO demonstrates superiority in both constraint satisfaction and sample efficiency. Firstly, as illustrated in Figure 4, CGPO exhibits more precise control over constraints. In scenarios where constraint satisfaction conflicts with performance improvement (i.e., CartPole), CGPO can strictly adhere to constraint thresholds to achieve higher performances. Additionally, as shown in Table 1, CGPO significantly reduces the proportion of constraint violations in critical safety zones compared to other baseline algorithms. The advantage of CGPO in constraint satisfaction largely sources from the accuracy of the GBE method, which will be further discussed in Section 6.3. Secondly, CGPO demonstrates higher sample efficiency by directly employing analytical gradients, as opposed to Monte Carlo sampling estimation. As shown in Table 1, CGPO requires less than 28.8%-91.7% of samples at convergence compared to baselines. This finding aligns with previous works (Mora et al., 2021; Jie Xu et al., 2022).



(a) FunctionEnv (b) HalfCheetah

Figure 5. GBE and ABE errors across four training stages for equal step length updates and averaging over 100 repetitions, where Relative Error $= \frac{\text{Estimation Error}}{\text{Constraint Change}}$. GBE effectively predicts the constraint function of the next policy without failing like ABE.



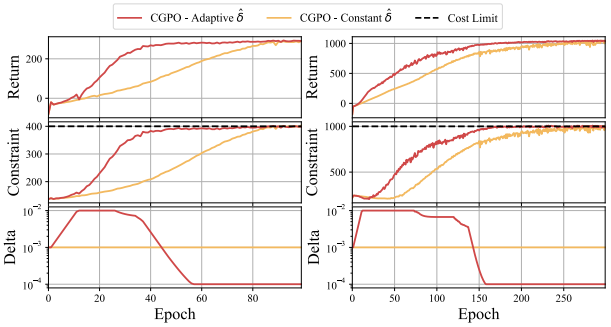
(a) Non-Diff. FunctionEnv (b) Non-Diff. Swimmer

Figure 6. CGPO uses gradients from the World Models in non-differentiable tasks. The grey shading represents predictions from the World Model.

Compared to SHAC-Lag and BPTT-Lag, CGPO exhibits more stable convergence. As Figure 4 illustrates, while SHAC-Lag and BPTT-Lag achieve efficient sample utilization, they oscillate near the threshold. This behavior sources from an inherent issue of Lagrangian methods: the delayed response of dual variables to constraint violations (Platt & Barr, 1987; Wah et al., 2000). Such delays result in inherent oscillations and cost overshoots.

6.3. Ablation on Estimation Errors

The enhanced performance of CGPO in our experiments primarily results from the GBE method's accurate estimation of finite-horizon constraints for future policies, a task at which the ABE method fails. Our ablation results to compare GBE and ABE errors are illustrated in Fig. 5. We observe that the estimation error of GBE is much smaller than that of ABE; the relative error of ABE even exceeds 1.0, highlighting its almost ineffectiveness. Thus, solutions from the ABE-based surrogate optimization problem are not as feasible for the primal Safe RL problem, preventing traditional algorithms from precisely meeting safety constraints.



(a) HalfCheetah (b) Ant

Figure 7. Adaptive trust region radius $\hat{\delta}$ enables CGPO to achieve better performance improvements and constraint satisfaction.

In the HalfCheetah task, errors are higher than in the FunctionEnv task, primarily due to the reduced system differentiability. This exposes a limitation of CGPO, its potential failure in poorly differentiable systems. CGPO has the potential to overcome it by training a World Model for gradient provision, as discussed in the next section.

6.4. Ablation on World Model Augmentation

In systems with limited differentiability, we can access analytic gradients by training a World Model. The fundamental implementation of the model-based CGPO algorithm is presented by Algorithm 3. We evaluate its efficacy on two non-differentiable tasks, with specific settings described in Appendix F.4. As shown in Figure 6, the implementation of a World Model enables CGPO to achieve improved performance while adhering to safety constraints in environments with limited differentiability. This demonstrates that the World Model has the potential to expand CGPO to a broader range of applications, overcoming the limitations of environmental differentiability. Nonetheless, incorporating the World Model incurs additional time and computational overhead, necessitating further optimization.

6.5. Ablation on Adaptive Trust Region Radius

Our method dynamically adjusts the trust region radius $\hat{\delta}$ to maintain estimation error within a tolerable limit. Adaptive $\hat{\delta}$, as Figure 7 demonstrates, enhances performance and ensures more accurate adherence to constraints than a static radius. The learning curve of the adaptive radius has two phases: initially, it prioritizes improvements in efficient performance, ensuring consistency between actual and predicted performance changes, while achieving more larger updates. Later, as policy updates approach constraint limits, the focus shifts to reducing estimation errors within the constraint budget, thereby preventing violations. More ablation

studies on the hyper-parameters of radius adaptation can be found in Appendix D.2

7. Limitation and Future Works

Although our algorithm demonstrates theoretical and empirical superiority over baseline algorithms, it has certain limitations. Like other differentiable reinforcement learning methods, CGPO relies on environmental differentiability. As discussed in Section 6.3, systems with poor differentiability are susceptible to environmental noise, resulting in increased system errors and reduced precision in GBE estimation. While CGPO mitigates the upper bound of worst-case estimation errors through an adaptive trust region radius method, an excessively small radius can reduce update efficiency.

We identify three promising solutions to address this limitation. First, advancing differentiable physics engines to enhance environmental differentiability (Degrave et al., 2019; Freeman et al., 2021; Howell et al., 2022). Second, developing algorithms to minimize system errors at the software level (Metz et al., 2021), such as the reparameterization technique (Lee et al., 2018). Third, providing analytic gradients for systems with poor differentiability through more accurate world model training (Parmas et al., 2023a). In Section 6.4, we discuss this approach and its basic implementation using a two-layer Multi-Layer Perceptron (MLP) (Popescu et al., 2009), resulting in low-precision gradients from the world model.

For future work, it is crucial to develop a more universal framework based on our algorithm that incorporates the three aforementioned approaches to mitigate existing limitations. Additionally, we plan to extend the application of our algorithm to fields beyond robotic control (He et al., 2024; Wang et al., 2024), such as the safety alignment (Ji et al., 2023a) of large language models (Ji et al., 2024b; Dai et al., 2024; Ji et al., 2024a) and autonomous driving (Das et al., 2016; Muhammad et al., 2020).

8. Conclusion

Since current deep Safe RL algorithms uniformly apply an infinite-horizon approach to all constraints, they fail to guarantee the feasibility of each update. In this paper, we introduce the *Gradient-based Estimation* (GBE) method to bridge the gap of finite-horizon constraint estimation in deep Safe RL. Based on GBE, we formulate a surrogate optimization problem and propose the first deep Safe RL algorithm, named *Constrained Gradient-based Policy Optimization* (CGPO), able to handle tasks with finite-horizon constraints. CGPO leverages precise GBE estimations to ensure the efficiency and feasibility of each update. We also present a differentiable Safe RL environment for algorithm testing. Comparative evaluations reveal that our algorithm outperforms baselines in terms of update efficiency and constraint satisfaction.

Acknowledgements

This research is supported by the Natural Science Foundation of China (No. 61925603), STI 2030 Major Projects (2021ZD0200400)

Impact Statement

This paper presents work whose goal is to advance the field of machine learning, specifically focusing on RL within differentiable simulation environments or world models. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization, 2017.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC press, 1999.

As, Y., Usmanova, I., Curi, S., and Krause, A. Constrained policy optimization via bayesian world models. *arXiv preprint arXiv:2201.09802*, 2022.

Bogue, R. Robots that interact with humans: a review of safety technologies and standards. *Industrial Robot: An International Journal*, 44(4):395–400, 2017.

Chow, Y., Ghavamzadeh, M., Janson, L., and Pavone, M. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18 (167):1–51, 2018.

Clavera, I., Fu, V., and Abbeel, P. Model-augmented actor-critic: Backpropagating through paths. *arXiv preprint arXiv:2005.08068*, 2020.

Dai, J., Ji, J., Yang, L., Zheng, Q., and Pan, G. Augmented proximal policy optimization for safe reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 7288–7295, 2023.

Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., and Yang, Y. Safe RLHF: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=TyFrPOKYXw.

Das, B., Subudhi, B., and Pati, B. B. Cooperative formation control of autonomous underwater vehicles: An overview.

International Journal of Automation and computing, 13: 199–225, 2016.

Degrave, J., Hermans, M., Dambre, J., et al. A differentiable physics engine for deep learning in robotics. *Frontiers in neurorobotics*, pp. 6, 2019.

ElDahshan, K. A., Farouk, H., and Mofreh, E. Deep reinforcement learning based video games: A review. In *2022 2nd International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pp. 302–309. IEEE, 2022.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. Brax - a differentiable physics engine for large scale rigid body simulation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Garcıa, J. and Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

Gronauer, S. Bullet-safety-gym: A framework for constrained reinforcement learning. 2022.

Guin, S. and Bhatnagar, S. A policy gradient approach for finite horizon constrained markov decision processes. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 3353–3359. IEEE, 2023.

He, X., Hu, Z., Yang, H., and Lv, C. Personalized robotic control via constrained multi-objective reinforcement learning. *Neurocomputing*, 565:126986, 2024.

Howell, T. A., Cleac'h, S. L., Brüdigam, J., Kolter, J. Z., Schwager, M., and Manchester, Z. Dojo: A differentiable physics engine for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

Jaisson, T. Deep differentiable reinforcement learning and optimal trading. *Quantitative Finance*, 22(8):1429–1443, 2022.

Ji, J., Qiu, T., Chen, B., Zhang, B., Lou, H., Wang, K., Duan, Y., He, Z., Zhou, J., Zhang, Z., et al. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*, 2023a.

Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., and Yang, Y. Safety-gymnasium: A unified safe reinforcement learning benchmark. *arXiv preprint arXiv:2310.12567*, 2023b.

Ji, J., Zhou, J., Zhang, B., Dai, J., Pan, X., Sun, R., Huang, W., Geng, Y., Liu, M., and Yang, Y. Omnisafe: An infrastructure for accelerating safe reinforcement learning research. *arXiv preprint arXiv:2305.09304*, 2023c.

Ji, J., Chen, B., Lou, H., Hong, D., Zhang, B., Pan, X., Dai, J., and Yang, Y. Aligner: Achieving efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*, 2024a.

Ji, J., Liu, M., Dai, J., Pan, X., Zhang, C., Bian, C., Chen, B., Sun, R., Wang, Y., and Yang, Y. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36, 2024b.

Jie Xu, V., Makoviychuk, Y., and Narang, F. R. Accelerated policy learning with parallel differentiable simulation. In *ICLR*, 2022.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.

Kalagarla, K. C., Jain, R., and Nuzzo, P. A sample-efficient algorithm for episodic finite-horizon mdp with constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 8030–8037, 2021.

Lee, W., Yu, H., and Yang, H. Reparameterization gradient for non-differentiable models. *Advances in Neural Information Processing Systems*, 31, 2018.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., and Zhao, D. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pp. 13644–13668. PMLR, 2022.

Meng, W., Zheng, Q., Shi, Y., and Pan, G. An off-policy trust region policy optimization method with monotonic improvement guarantee for deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(5):2223–2235, 2022. doi: 10.1109/TNNLS.2020.3044196.

Metz, L., Freeman, C. D., Schoenholz, S. S., and Kachman, T. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Mohamed, S., Rosca, M., Figurnov, M., and Mnih, A. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

Mora, M. A. Z., Peychev, M., Ha, S., Vechev, M., and Coros, S. Pods: Policy optimization via differentiable simulation. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 7805–7817. PMLR, 18–24 Jul 2021.

Mozer, M. A focused backpropagation algorithm for temporal pattern recognition. *Complex Systems*, 3, 01 1995.

Mozer, M. C. A focused backpropagation algorithm for temporal pattern recognition. In *Backpropagation*, pp. 137–169. Psychology Press, 2013.

Muhammad, K., Ullah, A., Lloret, J., Del Ser, J., and de Albuquerque, V. H. C. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22 (7):4316–4336, 2020.

Nilsson, J., Fredriksson, J., and Coelingh, E. Trajectory planning with miscellaneous safety critical zones**this work was supported by ffi - strategic vehicle research and innovation. *IFAC-PapersOnLine*, 50(1):9083–9088, 2017. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2017.08.1649. URL https://www.sciencedirect.com/science/article/pii/S2405896317322541. 20th IFAC World Congress.

Okamura, A. M., Smaby, N., and Cutkosky, M. R. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pp. 255–262. IEEE, 2000.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

Parmas, P., Seno, T., and Aoki, Y. Model-based reinforcement learning with scalable composite policy gradient estimators. In *International Conference on Machine Learning*, pp. 27346–27377. PMLR, 2023a.

Parmas, P., Seno, T., and Aoki, Y. Model-based reinforcement learning with scalable composite policy gradient estimators. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27346–27377. PMLR, 23–29 Jul 2023b. URL https://proceedings.mlr.press/v202/parmas23a.html.

Platt, J. and Barr, A. Constrained differential optimization. In *Neural Information Processing Systems*, 1987.

Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., and Mastorakis, N. Multilayer perceptron and neural networks. *WSEAS Transactions on Circuits and Systems*, 8(7):579–588, 2009.

Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

Ray, A., Achiam, J., and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

Shi, L., Li, S., Cao, L., Yang, L., and Pan, G. Tbq (σ): Improving efficiency of trace utilization for off-policy reinforcement learning. *arXiv preprint arXiv:1905.07237*, 2019.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Singh, B., Kumar, R., and Singh, V. P. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, pp. 1–46, 2022.

Stooke, A., Achiam, J., and Abbeel, P. Responsive safety in reinforcement learning by pid lagrangian methods. In *International Conference on Machine Learning*, pp. 9133–9143. PMLR, 2020.

Suh, H. J., Simchowitz, M., Zhang, K., and Tedrake, R. Do differentiable simulators give better policy gradients? In *International Conference on Machine Learning*, pp. 20668–20696. PMLR, 2022.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Wah, B. W., Wang, T., Shang, Y., and Wu, Z. Improving the performance of weighted lagrange-multiplier methods for nonlinear constrained optimization. *Information Sciences*, 124(1-4):241–272, 2000.

Wang, Y., Zhang, M., Li, M., Cui, H., and Chen, X. Development of a humanoid robot control system based on ar-bci and slam navigation. *Cognitive Neurodynamics*, pp. 1–14, 2024.

Werling, K., Omens, D., Lee, J., Exarchos, I., and Liu, C. K. Fast and feature-complete differentiable physics engine for articulated rigid bodies with contact constraints. In *Robotics: Science and Systems*, 2021.

Xian, Z., Zhu, B., Xu, Z., Tung, H.-Y., Torralba, A., Fragkiadaki, K., and Gan, C. Fluidlab: A differentiable environment for benchmarking complex fluid manipulation. *arXiv preprint arXiv:2303.02346*, 2023.

Xu, M., Liu, Z., Huang, P., Ding, W., Cen, Z., Li, B., and Zhao, D. Trustworthy reinforcement learning against intrinsic vulnerabilities: Robustness, safety, and generalizability. *arXiv preprint arXiv:2209.08025*, 2022.

Yang, L., Shi, M., Zheng, Q., Meng, W., and Pan, G. A unified approach for multi-step temporal-difference learning with eligibility traces in reinforcement learning. *arXiv preprint arXiv:1802.03171*, 2018.

Yang, L., Ji, J., Dai, J., Zhang, L., Zhou, B., Li, P., Yang, Y., and Pan, G. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35:9111–9124, 2022.

Yang, T.-Y., Rosca, J., Narasimhan, K., and Ramadge, P. J. Projection-based constrained policy optimization. *arXiv preprint arXiv:2010.03152*, 2020.

Zhang, Y., Vuong, Q., and Ross, K. First order constrained optimization in policy space. *Advances in Neural Information Processing Systems*, 33:15338–15349, 2020.

Appendix

Table of Contents

A. Supplementing Theoretical Aspects

A.1. The First-Order Approximation

We employ the following first-order estimation to approximate the objective and constraint functions:

$$\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_f(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0), \tag{22}$$

where $\hat{\mathcal{J}}_f$ represents the both objective function $\hat{\mathcal{J}}_R$ and constraint function $\hat{\mathcal{J}}_C$:

$$\hat{\mathcal{J}}_R(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_R(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_0), \tag{23}$$

$$\hat{\mathcal{J}}_C(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_C(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_0). \tag{24}$$

This approximation introduces the following approximation error:

Lemma 4.1. *Assume $\boldsymbol{\theta}_0 \in \Theta$ and $\mathcal{J}_f(\boldsymbol{\theta})$ is twice differentiable in a neighborhood surrounding $\boldsymbol{\theta}_0$. Let $\boldsymbol{\delta}$ be a small update from $\boldsymbol{\theta}_0$. If we estimate $\mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})$ as $\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_f(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0)$ and given that $\epsilon = \max_{t\in(0,1)} \left|\nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_0 + t\boldsymbol{\delta})\right|$, the estimation error can be bounded as:*

$$\left|\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})\right| \leq \frac{1}{2}\epsilon \|\boldsymbol{\delta}\|_2^2. \tag{8}$$

Proof. The function $\mathcal{J}_f(\boldsymbol{\theta})$ is expanded using a Taylor series at the point $\boldsymbol{\theta}_0 + \boldsymbol{\delta}$, resulting in:

$$\exists t_0 \in (0,1), \quad \mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) = \mathcal{J}_f(\boldsymbol{\theta}_0) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0) + \frac{1}{2}\boldsymbol{\delta}^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0 + t_0\boldsymbol{\delta})\boldsymbol{\delta} \tag{25}$$

where $\frac{1}{2}\boldsymbol{\delta}^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0 + t_0\boldsymbol{\delta})\boldsymbol{\delta}$ represents the Peano remainder term. Substituting this expanded result yields:

$$\left|\hat{\mathcal{J}}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta}) - \mathcal{J}_f(\boldsymbol{\theta}_0 + \boldsymbol{\delta})\right| = \frac{1}{2}\left|\boldsymbol{\delta}^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_f(\boldsymbol{\theta}_0 + t_0\boldsymbol{\delta})\boldsymbol{\delta}\right| \leq \frac{1}{2}\epsilon \|\boldsymbol{\delta}\|_2^2. \tag{26}$$

□

A.2. The Solution to Constrained Surrogate Sub-Problem

Based on Theorem 4.1, we find that the estimation error is positively correlated with the update step size. Therefore, by controlling the update step size, we can keep the estimation error within an acceptable range. Given the trust region radius, we consider updated parameter $\boldsymbol{\theta_{k+1}}$ of the k^{th} iteration within the trust region $\Theta_k = \left\{\boldsymbol{\theta} \in \Theta \mid \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2 \leq \hat{\delta}\right\}$ to be credible. By employing the approximations of the objective function in Equation (23) and the constraint function in Equation (24), We transform the solution of the primal Safe RL problem into an iterative process of solving a series of sub-problems within predefined trust regions Θ_k:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}\in\Theta} \quad & (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) \\ \text{s.t.} \quad & \mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) \leq b \\ & (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \leq \hat{\delta}, \end{aligned} \tag{27}$$

To simplify the expression, we introduce some new notations. For the k^{th} iteration, we denote the gradient of the objective as $\boldsymbol{g}_k \triangleq \mathcal{J}_R(\boldsymbol{\theta}_k)$, the gradient of the constraint as $\boldsymbol{q}_k \triangleq \mathcal{J}_C(\boldsymbol{\theta}_k)$, and define $c_k \triangleq \mathcal{J}_C(\boldsymbol{\theta}_k) - b$, $\boldsymbol{\delta}_k \triangleq \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$. Thus, we rewrite the sub-problem as

$$\boldsymbol{\delta}_k = \arg\max_{\boldsymbol{\delta}} \boldsymbol{g}_k^\top \boldsymbol{\delta}, \quad \text{s.t. } c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} \leq 0, \quad \boldsymbol{\delta}^\top \boldsymbol{\delta} \leq \hat{\delta} \tag{28}$$

A.2.1. SOLVABILITY CONDITIONS FOR THE SUB-PROBLEM

Since the sub-problem (28) may have no solution, we first discuss the conditions under which this problem is solvable. The following theorem holds:

Theorem 5.1 (Solvability Conditions for the Sub-problem)**.** *The sub-problem* (10) *is unsolvable if and only if* $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ *and* $c_k > 0$.

Proof. Denoting the boundary of the feasible region as $\mathring{\Delta}_k^C = \left\{\boldsymbol{\delta} \mid c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} = 0\right\}$, let $\mathring{\boldsymbol{\delta}}_k$ be the smallest update that places the next policy $\boldsymbol{\theta}_{k+1}$ on the boundary of the feasible region, namely

$$\mathring{\boldsymbol{\delta}}_k = \arg\min_{\boldsymbol{\delta}} \boldsymbol{\delta}^\top \boldsymbol{\delta}, \quad \text{s.t. } c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} = 0. \tag{29}$$

To derive the analytical form of $\mathring{\boldsymbol{\delta}}_k$, we introduce the dual variable $\lambda \geq 0$ to construct the Lagrangian dual function of the problem (29):

$$L(\boldsymbol{\delta}, \lambda) = \boldsymbol{\delta}^\top \boldsymbol{\delta} + \lambda \left(c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta}\right). \tag{30}$$

Since both the objective function and the constraint function in the problem (29) are convex, the optimal solution $(\mathring{\boldsymbol{\delta}}_k, \lambda^*)$ should satisfy the Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} &\nabla_{\boldsymbol{\delta}} L(\boldsymbol{\delta}, \lambda) = \boldsymbol{\delta} + \lambda \boldsymbol{q}_k = 0 \\ &c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} = 0 \end{aligned} \tag{31}$$

This can be solved to obtain $\lambda^* = c_k/\boldsymbol{q}_k^\top \boldsymbol{q}$ and $\mathring{\boldsymbol{\delta}}_k = -c_k \boldsymbol{q}_k / \boldsymbol{q}_k^\top \boldsymbol{q}_k$.

$\Longrightarrow$: if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q} - \hat{\delta} > 0$ and $c_k > 0$,

$$\mathring{\boldsymbol{\delta}}_k^\top \mathring{\boldsymbol{\delta}}_k = \frac{c_k^2}{\boldsymbol{q}_k^\top \boldsymbol{q}_k} > \hat{\delta} \tag{32}$$

Denoting the trust region of the k^{th} iteration as $\Delta_k = \left\{\boldsymbol{\delta} \in \Theta \mid \|\boldsymbol{\delta}\|^2 \leq \hat{\delta}\right\}$. For all $\boldsymbol{\delta} \in \mathring{\Delta}_k^C$ we have $\boldsymbol{\delta}^\top \boldsymbol{\delta} \geq \mathring{\boldsymbol{\delta}}_k^\top \mathring{\boldsymbol{\delta}}_k > \hat{\delta}$. Thus, $\forall \boldsymbol{\delta} \in \mathring{\Delta}_k^C, \delta \notin \Delta_k$, namely, the trust region Δ_k does not intersect with the boundary plane $\mathring{\Delta}_k^C$.

Let the feasible region for the k^{th} iteration be defined as $\Delta_k^C = \left\{\boldsymbol{\delta} \in \Theta \mid c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} \leq 0\right\}$, which is a half-space determined by the plane $\mathring{\Delta}_k^C$. Since the trust region Δ_k does not intersect with the plane $\mathring{\Delta}_k^C$, only two scenarios are possible, namely, either $\Delta_k \not\subset \Delta_k^C$ or $\Delta_k \subset \Delta_k^C$. When $c_k > 0$, $\mathbf{0} \in \Delta_k$ and $\mathbf{0} \notin \Delta_k^C$. Assuming $\Delta_k \subset \Delta_k^C$, then for all $\boldsymbol{\delta} \in \Delta_k$, it follows that $\boldsymbol{\delta} \in \Delta^C$. This leads to a contradiction, hence $\Delta_k \not\subset \Delta_k^C$. In other words, the sub-problem expressed in Equation (28) is unsolvable.

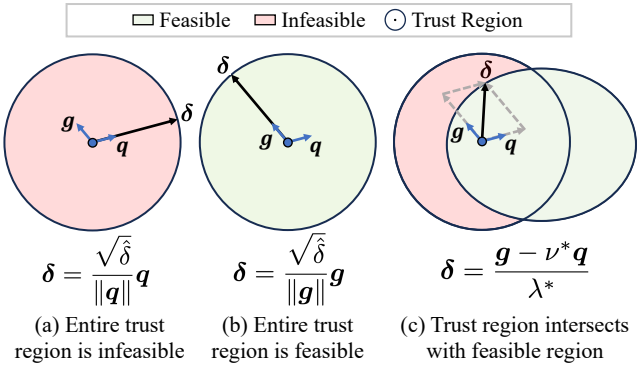
$\Longleftarrow$: If the sub-problem expressed in Equation (28) is unsolvable, it follows that $\Delta_k \not\subset \Delta_k^C$ and $\Delta_k \cap \mathring{\Delta}_k^C = \emptyset$.

Given that $\Delta_k \not\subset \Delta_k^C$ and considering $\mathbf{0} \in \Delta_k$ but $\mathbf{0} \notin \Delta_k^C$, it is deduced that $c_k + \boldsymbol{q}_k^\top \mathbf{0} = c_k > 0$. Furthermore, the condition $\Delta_k \cap \mathring{\Delta}_k^C = \emptyset$ implies that for any $\boldsymbol{\delta} \in \mathring{\Delta}_k^C$, the relation $\boldsymbol{\delta}^\top \boldsymbol{\delta} > \hat{\delta}$ holds true. Consequently, for $\mathring{\boldsymbol{\delta}}_k \in \mathring{\Delta}_k^C$, it is observed that $\mathring{\boldsymbol{\delta}}_k^\top \mathring{\boldsymbol{\delta}}_k = c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k > \hat{\delta}$. Thus, it is concluded that $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ and $c_k > 0$. □

In addition, the following corollary holds:

Corollary 5.2. *$\boldsymbol{\theta}$ is deemed feasible for every $\boldsymbol{\theta}$ within Θ_k if and only if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ and $c_k \leq 0$ are both satisfied.*

Proof. The proof is analogous to that of Theorem 5.1. The condition $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta} > 0$ is necessary and sufficient for the trust region Δ_k not to intersect with the boundary of the half-space $\mathring{\Delta}_k^C$. Since the trust region Δ_k does not intersect with the plane $\mathring{\Delta}_k^C$, only two scenarios are possible: either $\Delta_k \not\subset \Delta_k^C$ or $\Delta_k \subset \Delta_k^C$. Moreover, $c_k \leq 0$ is the necessary and sufficient condition for $\Delta_k \subset \Delta_k^C$ under these circumstances. □



(a) Entire trust region is infeasible

(b) Entire trust region is feasible

(c) Trust region intersects with feasible region

Figure 8. The computational relationship between the policy update $\boldsymbol{\delta}$, the gradient of the objective function $\boldsymbol{g}$, and the gradient of the constraint function $\boldsymbol{g}$ varies in three scenarios.

A.2.2. THE SOLUTION TO THE SUB-PROBLEM

Based on Theorem 5.1 and Corollary 5.2, solving the sub-problem within the trust region, as defined in Equation 28, should be discussed in three scenarios: (a) when the entire trust region is infeasible, (b) when the entire trust region is feasible, and (c) when the trust region is partially feasible, as shown in Figure 8.

(a) If $c^2/\boldsymbol{q}^\top \boldsymbol{q} - \hat{\delta} > 0$ and $c > 0$, the entire trust region is infeasible. We update the policy along the direction of the steepest descent of the constraint function, namely,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\sqrt{\hat{\delta}}}{\|\boldsymbol{q}_k\|} \boldsymbol{q}_k. \tag{33}$$

(b) If $c^2/\boldsymbol{q}^\top \boldsymbol{q} - \hat{\delta} > 0$ and $c \leq 0$, the trust region lies entirely within the constraint-satisfying half-space. We directly update the step length to $\sqrt{\hat{\delta}}$ along the direction of the steepest ascent of the objective function, represented by the gradient vector $\boldsymbol{g}$, namely,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\sqrt{\hat{\delta}}}{\|\boldsymbol{g}_k\|} \boldsymbol{g}_k. \tag{34}$$

(c) In other cases, the trust region partially intersects with the feasible region, so we have to solve the constrained quadratic sub-problem (28).

The sub-problem (28) is characterized as convex. When the trust region is partially feasible, the existence of at least one strictly feasible point is guaranteed. Consequently, strong duality holds as the conditions of Slater's theorem are fulfilled. By introducing two dual variables λ and ν, we construct the dual function of the primal problem as follows:

$$L(\boldsymbol{\delta}, \lambda, \nu) = -\boldsymbol{g}_k^\top \boldsymbol{\delta} + \nu \left(c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta}\right) + \frac{\lambda}{2} \left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - \hat{\delta}\right) \tag{35}$$

Since the sub-problem (28) satisfies the strong duality condition, any pair of primal and dual optimal points $(\boldsymbol{\delta}^*, \lambda^*, \nu^*)$ must satisfy the KKT conditions, namely,

$$\nabla_{\boldsymbol{\delta}} L(\boldsymbol{\delta}, \lambda, \nu) = -\boldsymbol{g}_k + \nu \boldsymbol{q}_k + \lambda \boldsymbol{\delta} = 0, \tag{36}$$

$$\nu \left(c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta}\right) = 0, \tag{37}$$

$$\lambda \left(\boldsymbol{\delta}^\top \boldsymbol{\delta} - \hat{\delta}\right) = 0, \tag{38}$$

$$c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} \leq 0, \quad \boldsymbol{\delta}^\top \boldsymbol{\delta} - \hat{\delta} \leq 0, \quad \nu \geq 0, \quad \lambda \geq 0 \tag{39}$$

Algorithm 2 Dual variable solver in the case of solvability.

Input: the objective gradient $\boldsymbol{g}_k$, the constraint gradient $\boldsymbol{q}_k$, the constraint satisfaction c_k, and the trust region radius $\hat{\delta}$.
Compute the $r_k = \boldsymbol{g}_k^\top \boldsymbol{g}_k$, $s_k = \boldsymbol{g}_k^\top \boldsymbol{q}_k$, and $t_k = \boldsymbol{q}_k^\top \boldsymbol{q}_k$.
Compute $\Lambda_a = \{\lambda | \lambda c_k + s_k > 0, \lambda \geq 0\}$ and $\Lambda_b = \{\lambda | \lambda c_k + s_k \leq 0, \lambda \geq 0\}$.
Compute $\lambda_a = \text{Proj}\left(\sqrt{\frac{r_k - s_k^2/t_k}{\hat{\delta} - c_k^2/t_k}}, \Lambda_a\right)$.
Compute $\lambda_b = \text{Proj}\left(\sqrt{\frac{r_k}{\hat{\delta}}}, \Lambda_b\right)$.
Compute $L_a(\lambda_a) = \frac{1}{2\lambda_a}\left(\frac{s_k^2}{t_k} - r_k\right) + \frac{\lambda_a}{2}\left(\frac{c_k^2}{t_k} - \hat{\delta}\right) + \frac{s_k c_k}{t_k}$.
Compute $L_b(\lambda_b) = -\frac{1}{2}\left(\frac{r_k}{\lambda_b} + \lambda_b \hat{\delta}\right)$.
if $L_a(\lambda_a) > L_b(\lambda_b)$ **then**
  $\lambda_k^* = \lambda_a$;
else
  $\lambda_k^* = \lambda_b$.
end if
Compute $\nu_k^* = \max\left(\frac{\lambda_k^* c_k + s_k}{t_k},\ 0\right)$.
Output: λ_k^*, ν_k^*.

Based on Equation (36), we derive that

$$\boldsymbol{\delta}^* = \frac{1}{\lambda}\left(\boldsymbol{g}_k - \nu \boldsymbol{q}_k\right). \tag{40}$$

Substituting this into Equations (37), we obtain

$$\nu\left(\lambda c_k + \boldsymbol{q}_k^\top \boldsymbol{g} - \nu \boldsymbol{q}^\top \boldsymbol{q}\right) = 0 \tag{41}$$

To simplify the expression, we denote that $r_k \triangleq \boldsymbol{g}_k^\top \boldsymbol{g}_k$, $s_k \triangleq \boldsymbol{g}_k^\top \boldsymbol{q}_k$, and $t_k \triangleq \boldsymbol{q}_k^\top \boldsymbol{q}_k$. Thus, it follows that

$$\nu = \begin{cases} \frac{\lambda c_k + \boldsymbol{q}_k^\top \boldsymbol{g}_k}{\boldsymbol{q}_k^\top \boldsymbol{q}_k}, & \text{if } \lambda c_k + \boldsymbol{q}_k^\top \boldsymbol{g}_k > 0; \\ 0, & \text{if } \lambda c_k + \boldsymbol{q}_k^\top \boldsymbol{g}_k \leq 0. \end{cases} = \begin{cases} \frac{\lambda c_k + s_k}{t_k}, & \text{if } \lambda c_k + s_k > 0; \\ 0, & \text{if } \lambda c_k + s_k \leq 0. \end{cases} \tag{42}$$

Substituting Equation (40) and Equation (42) into Equation (38), we obtain

$$\begin{cases} r_k - 2 s_k \left(\frac{\lambda c_k + s_k}{t_k}\right) + t_k \left(\frac{\lambda c_k + s_k}{t_k}\right)^2 = \lambda^2 \hat{\delta}, & \lambda \in \Lambda_a; \\ r_k = \lambda^2 \hat{\delta}, & \lambda \in \Lambda_b. \end{cases} \tag{43}$$

where $\Lambda_a \doteq \{\lambda | \lambda c_k + s_k > 0, \lambda \geq 0\}$, $\Lambda_b \doteq \{\lambda | \lambda c_k + s_k \leq 0, \lambda \geq 0\}$. It is noted that when $c_k < 0$, $\Lambda_a = [0, -s_k/c_k)$ and $\Lambda_b = [-s_k/c_k, \infty)$; when $c > 0$, $\Lambda_a = [-s_k/c_k, \infty)$ and $\Lambda_b = [0, -s_k/c_k)$. Additionally, the inequality $r_k - s_k^2/t_k > 0$ is satisfied, which is a consequence of the Cauchy-Schwarz inequality, as it implies $\|\boldsymbol{g}_k\|_2^2 \|\boldsymbol{q}_k\|_2^2 \geq (\boldsymbol{g}_k^\top \boldsymbol{q}_k)^2$. Therefore, the following solution holds:

$$\lambda^* = \begin{cases} \lambda_a \triangleq \text{Proj}\left(\sqrt{\frac{r_k - s_k^2/t_k}{\hat{\delta} - c_k^2/t_k}}, \Lambda_a\right), & L_a(\lambda_a) > L_b(\lambda_b); \\ \lambda_b \triangleq \text{Proj}\left(\sqrt{\frac{r_k}{\hat{\delta}}}, \Lambda_b\right), & \text{otherwise.} \end{cases} \tag{44}$$

In above context, L_a and L_b are the results obtained by substituting equations (40) and (42) into dual function (35), namely,

$$\begin{cases} L_a(\lambda) \triangleq \frac{1}{2\lambda}\left(\frac{s_k^2}{t_k} - r_k\right) + \frac{\lambda}{2}\left(\frac{c_k^2}{t_k} - \hat{\delta}\right) + \frac{s_k c_k}{t_k}, & \lambda \in \Lambda_a; \\ L_b(\lambda) \triangleq -\frac{1}{2}\left(\frac{r_k}{\lambda} + \lambda \hat{\delta}\right), & \lambda \in \Lambda_b. \end{cases} \tag{45}$$

Since the dual variable λ is used to maximize the dual function (35), we can determine which one of λ_a and λ_b is the optimal solution by comparing $L_a(\lambda_a)$ and $L_b(\lambda_b)$.

Consequently, the optimal ν can be expressed in terms of λ^* as follows:

$$\nu^* = \left(\frac{\lambda^* c_k + s_k}{t_k} \right)_+. \tag{46}$$

The pseudo-code for computing the optimal (λ_k^*, ν_k^*) is provided in Algorithm 2.

Then, we update the policy as

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\boldsymbol{g}_k - \nu^* \boldsymbol{q}_k}{\lambda^*}. \tag{47}$$

A.3. Worst-Case Analysis

In the subsequent corollary, we detail the bounds for both performance improvement and constraint violations under the worst-case conditions following the policy update from resolving the sub-problem (27).

Theorem 5.3 (Worst-Case Performance Update and Constraint Violation)**.** *Suppose $\boldsymbol{\theta}_k$, $\boldsymbol{\theta}_{k+1} \in \Theta$ are related by Equation (9). If $\boldsymbol{\theta}_k$ is feasible, a lower bound on the policy performance difference between $\boldsymbol{\theta}_{k+1}$ and $\boldsymbol{\theta}_k$ is*

$$\mathcal{J}_R(\boldsymbol{\theta}_{k+1}) - \mathcal{J}_R(\boldsymbol{\theta}_k) \geq -\frac{1}{2} \epsilon_k^R \hat{\delta} \tag{16}$$

where $\epsilon_k^R = \max_{t \in (0,1)} \left| \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)) \right|$.

An upper bound on the constraint objective function of $\boldsymbol{\theta}_{k+1}$ is

$$\mathcal{J}_C(\boldsymbol{\theta}_{k+1}) \leq b + \frac{1}{2} \epsilon_k^C \hat{\delta}, \tag{17}$$

where $\epsilon_k^C = \max_{t \in (0,1)} \left| \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)) \right|$.

Proof. Let the feasible region of the sub-problem be denoted as

$$\Theta_k^C = \left\{ \boldsymbol{\theta} \in \Theta \mid \mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) \leq b, \quad \|\boldsymbol{\theta} - \boldsymbol{\theta}_k\|^2 \leq \hat{\delta} \right\}. \tag{48}$$

Furthermore, because

$$\boldsymbol{\theta}_{k+1} = \arg \max_{\boldsymbol{\theta} \in \Theta_k^C} (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k), \tag{49}$$

it follows that

$$\forall \boldsymbol{\theta} \in \Theta_k^C, \quad (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) \geq (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k). \tag{50}$$

Since $\boldsymbol{\theta}_k$ is feasible, it follows that $\boldsymbol{\theta}_k \in \Theta_k^C$. Consequently, we have

$$(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) \geq (\boldsymbol{\theta}_k - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) = 0. \tag{51}$$

Further, we have $\exists t_k \in (0,1)$ such that

$$\begin{aligned}
\mathcal{J}_R(\boldsymbol{\theta}_{k+1}) - \mathcal{J}_R(\boldsymbol{\theta}_k) &= (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) + \frac{1}{2} (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k + t_k(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k))(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) \\
&\geq -\frac{1}{2} \left| (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k + t_k(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k))(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k) \right| \\
&\geq -\frac{1}{2} \epsilon_k^R \|\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k\|_2^2 \\
&\geq -\frac{1}{2} \epsilon_k^R \hat{\delta}
\end{aligned} \tag{52}$$

where $\epsilon_k^R = \max_{t \in (0,1)} \left| \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)) \right|$.

On the other hand, since $\boldsymbol{\theta}_{k+1}$ is feasible, it follows that

$$\mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) \leq b \tag{53}$$

Furthermore, we have

$$\begin{aligned}\mathcal{J}_C(\boldsymbol{\theta}_{k+1}) &= \mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k)^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k + t_k(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k))(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k) \\ &\leq b + \frac{1}{2}(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k)^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k + t_k(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k))(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k) \\ &\leq b + \frac{1}{2}\left|(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k)^\top \nabla^2_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k + t_k(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k))(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k)\right| \\ &\leq b + \frac{1}{2}\epsilon_k^C \left\|\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k\right\|_2^2 \\ &\leq b + \frac{1}{2}\epsilon_k^C \hat{\delta}\end{aligned} \tag{54}$$

where $\epsilon_k^C = \max_{t\in(0,1)} \left|\nabla^2_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k + t(\boldsymbol{\theta}_{k+1}-\boldsymbol{\theta}_k))\right|$.

By combining Equation (52) and Equation (54), Corollary 5.3 is thus proven. □

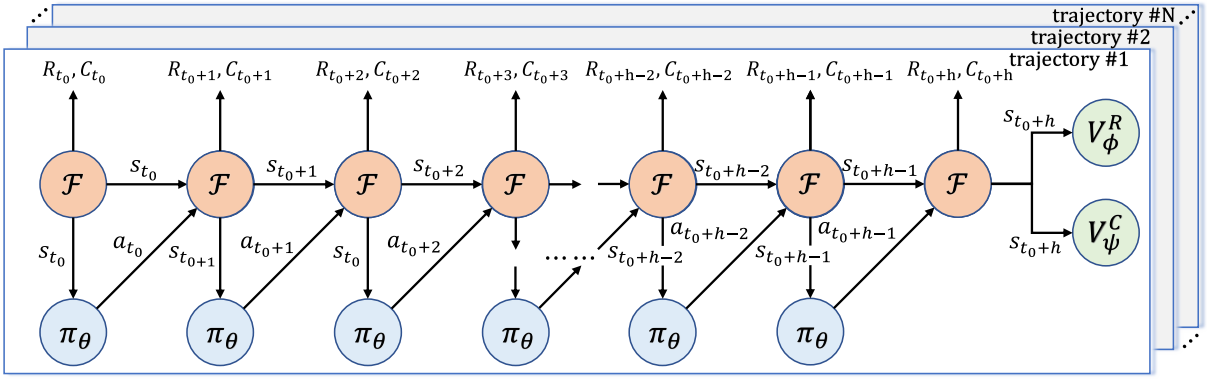


Figure 9. Computation graph of CGPO.

A.4. Gradient Calculation

Given our focus on employing gradients to solve Safe RL problems, our algorithm can incorporate any differentiable method for computing gradients $\boldsymbol{g_k}$ and $\boldsymbol{q_k}$, such as BPTT (Mozer, 2013) and SHAC (Jie Xu et al., 2022). Here, we give an example of implementation using SHAC approach.

We compute the gradients of the objective function $\mathcal{J}_R$ and constraint functions $\mathcal{J}_C$ concerning policy θ using a methodology similar to SHAC (Jie Xu et al., 2022). While training the policy network, we concurrently train a reward critic network V_ϕ^R and a cost one V_ψ^C. We divide the entire trajectory into short-horizon sub-windows, each with a length of h. Subsequently, the gradients of the objective function (constraint function) are calculated by multi-step rewards (costs) within sub-windows plus a final value estimation from the corresponding learned critics. Specifically, considering the N-trajectory collection $\{\tau_i\}_{i=1}^N$ obtained from a given policy $\boldsymbol{a}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$ and differentiable simulator $\boldsymbol{s}_{t+1} = \mathcal{F}(\boldsymbol{s}_t, \boldsymbol{a}_t)$, we employ the following two loss functions to compute gradients on a short-horizon sub-window:

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=t_0}^{t_0+h-1} r(\mathbf{s}_t^i, \mathbf{a}_t^i)\right) + V_\phi^R(\mathbf{s}_{t_0+h}^i)\right], \tag{55}$$

$$\mathcal{L}_C(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\sum_{t=t_0}^{t_0+h-1} c(\mathbf{s}_t^i, \mathbf{a}_t^i)\right) + V_\psi^C(\mathbf{s}_{t_0+h}^i)\right], \tag{56}$$

where t_0 is the starting time step of a sub-window. To compute the gradients $\frac{\partial \mathcal{L}_R(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathcal{L}_C(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$, we treat the simulator $\mathcal{F}$ as a differentiable layer and incorporate it into the overall computational graph, as illustrated in Figure 9. The gradients are

then computed by back-propagation through time (BPTT) (Mozer, 1995), with the replacement of the final time step by values $V_\psi^R(\mathbf{s}_{t_0+h}^i)$, $V_\psi^C(\mathbf{s}_{t_0+h}^i)$, and with truncation at the beginning of each sub-window.

When performing specific gradient calculations, we need to start from the last time step t_0+h of the sub-window for each trajectory τ_i:

$$\frac{\partial \mathcal{L}_R(\theta)}{\partial s_{t_0+h}^i} = \frac{1}{N}\frac{\partial V_\phi^R(s_{t_0+h}^i)}{\partial s_{t_0+h}^i}, \tag{57}$$

$$\frac{\partial \mathcal{L}_C(\theta)}{\partial s_{t_0+h}^i} = \frac{1}{N}\frac{\partial V_\psi^C(s_{t_0+h}^i)}{\partial s_{t_0+h}^i}. \tag{58}$$

Then, we can compute the adjoints in the previous steps $t_0 \le t < t_0+h$ in reverse order:

$$\frac{\partial \mathcal{L}_R(\theta)}{\partial s_t^i} = \frac{1}{N}\frac{\partial R(s_t^i,a_t^i)}{\partial s_t^i} + \left(\frac{\partial \mathcal{L}_R(\theta)}{\partial s_{t+1}^i}\right)\left(\left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial s_t^i}\right) + \left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial a_t^i}\right)\left(\frac{\pi_\theta(s_t^i)}{\partial s_t^i}\right)\right) \tag{59}$$

$$\frac{\partial \mathcal{L}_C(\theta)}{\partial s_t^i} = \frac{1}{N}\frac{\partial C(s_t^i,a_t^i)}{\partial s_t^i} + \left(\frac{\partial \mathcal{L}_C(\theta)}{\partial s_{t+1}^i}\right)\left(\left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial s_t^i}\right) + \left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial a_t^i}\right)\left(\frac{\pi_\theta(s_t^i)}{\partial s_t^i}\right)\right) \tag{60}$$

$$\frac{\partial \mathcal{L}_R(\theta)}{\partial a_t^i} = \frac{1}{N}\frac{\partial R(s_t^i,a_t^i)}{\partial a_t^i} + \left(\frac{\partial \mathcal{L}_R(\theta)}{\partial s_{t+1}^i}\right)\left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial a_t^i}\right) \tag{61}$$

$$\frac{\partial \mathcal{L}_C(\theta)}{\partial a_t^i} = \frac{1}{N}\frac{\partial C(s_t^i,a_t^i)}{\partial a_t^i} + \left(\frac{\partial \mathcal{L}_C(\theta)}{\partial s_{t+1}^i}\right)\left(\frac{\partial \mathcal{F}(s_t^i,a_t^i)}{\partial a_t^i}\right) \tag{62}$$

From all the computed adjoints, we can compute the objective loss and the constraint loss by

$$\frac{\partial \mathcal{L}_R(\theta)}{\partial \theta} = \sum_{i=1}^{N}\sum_{t=t_0}^{t_0+h-1}\left(\frac{\partial \mathcal{L}_R(\theta)}{\partial a_t^i}\right)\left(\frac{\partial \pi_\theta(s_t^i)}{\partial \theta}\right) \tag{63}$$

$$\frac{\partial \mathcal{L}_C(\theta)}{\partial \theta} = \sum_{i=1}^{N}\sum_{t=t_0}^{t_0+h-1}\left(\frac{\partial \mathcal{L}_C(\theta)}{\partial a_t^i}\right)\left(\frac{\partial \pi_\theta(s_t^i)}{\partial \theta}\right) \tag{64}$$

A.5. The Update Method of Critic Network

As mentioned in Section 5.3, line 309, the time step t is incorporated into the state as an input to the critic network, namely, $V(s_t,t)$. This value function forecasts the expected return for a future finite horizon of $T-t$ at time step t and state s:

$$V_\pi(s,t) \doteq \mathbb{E}_\pi\left[\sum_{k=1}^{T-t} R_{t+k} \mid S_t = s\right] \tag{65}$$

We define a series of n-step estimates as follows:

$$G_{t:t+n} \doteq R_{t+1} + R_{t+2} + \cdots + R_{t+n} + V_\pi(S_{t+n}, t+n) \tag{66}$$

Specifically, in our implementation, the data $\{\tau_{t_0:t_h}^i\}_{i=1}^N$ obtained from each short horizon h rollout is used to calculate $G_{t:t+n}, n = 1,2,\cdots,h-t$. Given λ, an unbiased finite-horizon TD(λ) estimate of $V(s_t^i,t)$, $s_t^i \in \tau_{t_0:t_h}^i$ can be obtained by the following formula:

$$G_t^{\lambda,i} = (1-\lambda)\sum_{n=1}^{h-t-1}\lambda^{n-1}G_{t:t+n}^i + \lambda^{h-t-1}G_{t:t+h}^i \tag{67}$$

Note that the weight decays as n increases and the total summation is 1. Thus, the updated Loss function can be defined as:

$$L^\lambda(\phi) = \frac{1}{Nh}\sum_{i=1}^{N}\sum_{t=0}^{h-1}\left(G_t^{\lambda,i} - V_\phi(s_t^i,t)\right)^2 \tag{68}$$

B. Zero-order Batched Gradient method vs First-order Batched Gradient method

Given our focus on employing gradients to solve Safe RL problems, our algorithm can incorporate any differentiable method to compute gradients $\boldsymbol{g_k}$ and $\boldsymbol{q_k}$. Various methods are available, broadly classified into two categories: the Zero-order Batch Gradient (ZoBG) method and the First-order Batch Gradient (FoBG) method (Suh et al., 2022).

compared to zeroth-order batched gradient (ZoBG) estimation on $\{\tau_i\}_{i=1}^N$,

$$\nabla_\theta \mathcal{J}(\theta) \simeq g_{\text{ZoBG}} \doteq \frac{1}{N}\sum_{i=1}^{N}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta(a_t^i \mid s_t^i) R(\tau_i)\right], \tag{69}$$

we find that FoBG is more suitable for the Safe RL field for the following two reasons:

First, using ZoBG to estimate the constraint at $\boldsymbol{\theta}_k + \boldsymbol{\delta}$ results in significant estimation errors. In Safe RL, the estimation of constraint condition $\mathcal{J}_C(\boldsymbol{\theta}_k + \boldsymbol{\delta}) \leq b$ serves as an important basis for solving the optimal feasible update $\boldsymbol{\delta}^*$. Thus, inaccurate estimations can lead to suboptimal constraint satisfaction. Intuitively, since ZoBG does not utilize the first-order gradient of environmental dynamics, it fails to capture how policy changes affect state visitation changes. Particularly in long trajectories, this effect is amplified, resulting in greater changes to states visited later due to earlier state modifications. However, first-order batched gradient (FoBG) estimation mitigates this amplification effect by incorporating the first-order gradient of environmental dynamics, allowing the gradient to propagate along the trajectory.

We empirically confirmed this through experiments comparing the relative errors of ZoBG and FoBG in estimating constraint functions. These experiments were conducted in simple FunctionEnv environments, as detailed in Appendix D.1, across various trajectory lengths and update magnitudes $\hat{\delta}$. Each experiment ran for 100 epochs using 5 random seeds, with the results displayed in the tables below:

Table 2. Relative error of ZoBG in constraint function estimation in the simple FunctionEnv which is detailed in Appendix C.

Traj. Len. →	1	5	10	50	100	200
$\hat{\delta}$ = 0.0001	0.3387 ± 0.2420	0.8862 ± 0.3499	3.093 ± 1.394	15.40 ± 15.93	35.90 ± 29.11	65.19 ± 138.6
$\hat{\delta}$ = 0.0001	0.6291 ± 1.4107	0.4401 ± 1.722	15.575 ± 6.813	32.09 ± 132.9	150.6 ± 68.59	39.17 ± 134.4
$\hat{\delta}$ = 0.0001	0.3171 ± 2.039	8.709 ± 33.90	31.38 ± 139.1	68.71 ± 292.6	197.6 ± 488.8	66.40 ± 348.7
$\hat{\delta}$ = 0.0001	1.630 ± 11.76	4.393 ± 17.18	22.56 ± 132.8	276.8 ± 372.7	202.4 ± 291.1	35.99 ± 181.0
$\hat{\delta}$ = 0.0001	12.00 ± 26.20	11.51 ± 66.89	65.47 ± 118.9	94.81 ± 93.27	174.9 ± 515.3	71.00 ± 288.0

Table 3. Relative error of FoBG in constraint function estimation in the simple FunctionEnv which is detailed in Appendix C.

Traj. Len. →	1	5	10	50	100	200
$\hat{\delta}$ = 0.0001	0.0081 ± 0.0070	0.0033 ± 0.0029	0.0018 ± 0.0015	0.0007 ± 0.0006	0.0011 ± 0.0014	0.0015 ± 0.0024
$\hat{\delta}$ = 0.0001	0.0029 ± 0.0022	0.0027 ± 0.0021	0.0027 ± 0.0022	0.0030 ± 0.0028	0.0059 ± 0.0163	0.0129 ± 0.0606
$\hat{\delta}$ = 0.0001	0.0050 ± 0.0038	0.0046 ± 0.0038	0.0045 ± 0.0039	0.0056 ± 0.0064	0.0084 ± 0.0154	0.0165 ± 0.0661
$\hat{\delta}$ = 0.0001	0.0283 ± 0.0295	0.0221 ± 0.0226	0.0194 ± 0.0170	0.0189 ± 0.0227	0.0224 ± 0.0327	0.0494 ± 0.2030
$\hat{\delta}$ = 0.0001	0.1433 ± 1.1200	0.0371 ± 0.0502	0.0308 ± 0.0330	0.0290 ± 0.0406	0.0399 ± 0.0577	0.1045 ± 0.5258

Table 2 shows that ZoBG effectively estimates the constraint function only at short trajectory lengths (≤ 5) and small update magnitudes (≤ 0.001). However, with longer trajectory lengths and larger update magnitudes, ZoBG quickly loses its ability to estimate constraints (Relative Error ≥ 1). In contrast, Table 3 demonstrates that FoBG consistently estimates the constraint function accurately across all trajectory lengths and update magnitudes, indicating superior stability. These results highlight FoBG's significant advantages over ZoBG in constraint prediction within the Safe RL domain.

Second, ZoBG, which employs the Monte Carlo method, exhibits significant estimation variance and poor sample efficiency. To validate this, we introduced a new baseline, CGPO-ZoBG, in our training, where ZoBG is utilized for gradient estimation in our algorithm. The training curve of CGPO-ZoBG is depicted in the figure below:

Figure 10 illustrates that CGPO-ZoBG has low and unstable update efficiency, complicating the search for the optimal feasible update. This evidence further underscores ZoBG's inadequacy for Safe RL.

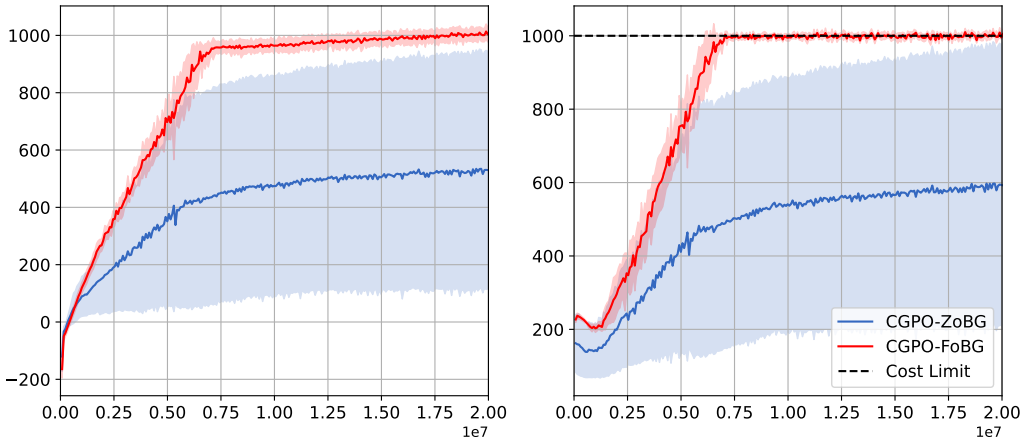


Figure 10. Training curves of CGPO using ZoBG method and FoBG method to calculate the gradients, showing episodic return and constraint for 5 random seeds. Solid lines represent the mean, while the shaded areas indicate variance, without any smoothing to the curves.

In summary, despite challenges in its application, FoBG's advantages in Safe RL are more pronounced than those of ZoBG. Furthermore, as the techniques of differentiable simulators and world models continue to evolve, FoBG's application potential will significantly expand. Therefore, we argue that integrating differentiable RL methods into our algorithm is essential.

C. Experiment Highlighting the Shortcomings of the ABE Method

C.1. Task Settings

To clearly demonstrate the shortcomings of the *Advantage-based Estimation* (ABE) method, we designed a simple task. The goal of this task is to train an agent to move along the x-axis, receiving rewards and incurring costs based on its position on the x-axis. The specific settings of the task are as follows:

Action Space: The direction and step length of the agent's movement along the x-axis constitute the action space, which is a single-element vector, i.e., $a \in (-1, 1)$.

Observation Space: The observation data for the agent is its position on the x-axis, which is also a single-element vector, i.e., $s \in \mathbb{R}$. The task employs a simple, artificially set state transition function: $s_{t+1} = s_t + 0.2 \times a_t$.

Reward Function and Cost Function: Since the purpose of this simple task is to verify the accuracy of the estimation method, we set the reward function and cost function in the same form, i.e.,

$$f(x) = \left(\frac{x}{10.0}\right)^2 + 0.1 + 0.1 \times \sin\left(\frac{8x}{\pi}\right) \tag{70}$$

This function is sourced from Metz et al. (2021), and its visualization is shown in Figure 11.

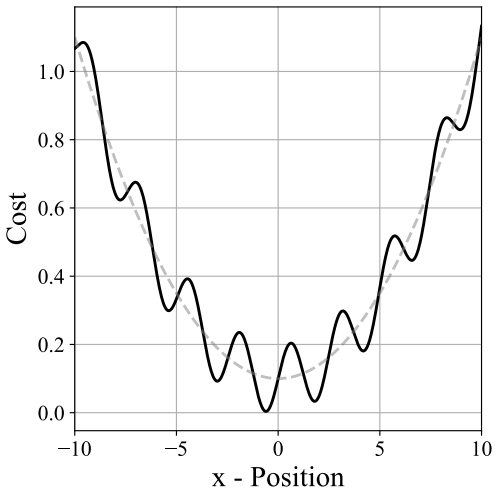


Figure 11. Cost values at different x-axis positions for the agent.

Constraint: Since the task is solely aimed at verifying the accuracy of the estimation method, the set constraint has no physical significance. Specifically, the cumulative cost on a trajectory of length 100 for the agent should be less than 8.0:

$$\mathcal{J}_C(\pi) \triangleq \sum_{t=0}^{99} f(x_t) \leq 8.0 \tag{71}$$

This constraint setting is consistent with many well-known Safe Benchmarks (such as Safety-Gym (Ray et al., 2019), Bullet-Safety-Gym (Gronauer, 2022), Safety-Gymnasium (Ji et al., 2023b), and OmniSafe (Ji et al., 2023c)), where the undiscounted cumulative cost on a finite-length trajectory must be less than a scalar threshold, namely, $\sum_{t=0}^{T-1} c_t \leq b$.

C.2. Experimental Details

In the training procedure, the parameterized initial strategy π_{θ_0} is updated iteratively, leveraging both the Advantage-based Estimation (ABE) and the Gradient-based Estimation (GBE) methods. The magnitude of each update, denoted by $\boldsymbol{\delta}$, remains fixed at $\|\boldsymbol{\delta}\| = 0.01$. For every update, the value of the updated constraint function is computed separately using the ABE and GBE methods:

ABE method:

$$\hat{\mathcal{J}}_C^{\text{ABE}}(\boldsymbol{\theta}+\boldsymbol{\delta}) \triangleq \mathcal{J}_C(\boldsymbol{\theta}) + \frac{1}{1-\gamma}\mathbb{E}_{\substack{s\sim d^{\pi_{\boldsymbol{\theta}}} \\ a\sim\pi_{\boldsymbol{\theta}}}}\left[\frac{\pi_{\boldsymbol{\theta}+\boldsymbol{\delta}}(s,a)}{\pi_{\boldsymbol{\theta}}(s,a)} A^{\pi_{\boldsymbol{\theta}}}(s,a)\right] \tag{72}$$

GBE method:

$$\hat{\mathcal{J}}_C^{\text{GBE}}(\boldsymbol{\theta}+\boldsymbol{\delta}) \triangleq \mathcal{J}_C(\boldsymbol{\theta}) + \boldsymbol{\delta}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}). \tag{73}$$

Metrics: Given that both methods aim to approximate the variation relative to $J_C(\boldsymbol{\theta})$, the relative error in the actual shift of the constraint function is employed as the evaluative metric:

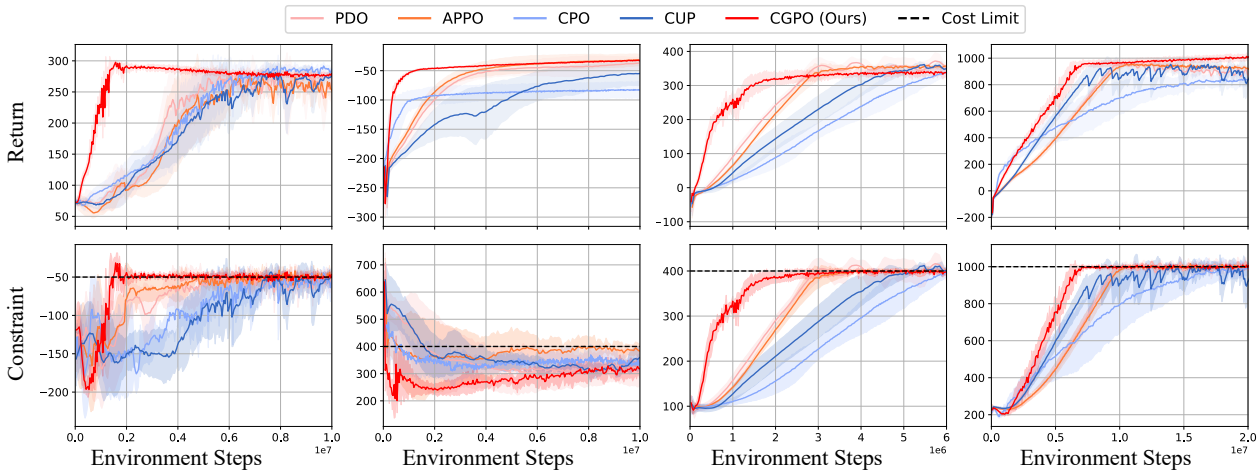
$$\Delta_{\text{relative}} = \frac{\left|\mathcal{J}_C(\boldsymbol{\theta}+\boldsymbol{\delta}) - \hat{\mathcal{J}}_C(\boldsymbol{\theta}+\boldsymbol{\delta})\right|}{\left|\mathcal{J}_C(\boldsymbol{\theta}+\boldsymbol{\delta}) - \mathcal{J}_C(\boldsymbol{\theta})\right|} \tag{74}$$

Notably, an estimation is considered valid when the relative error stays below 1.0. Conversely, an estimation loses its scalar referential value when the relative error exceeds 1.0.

D. More Experimental Results

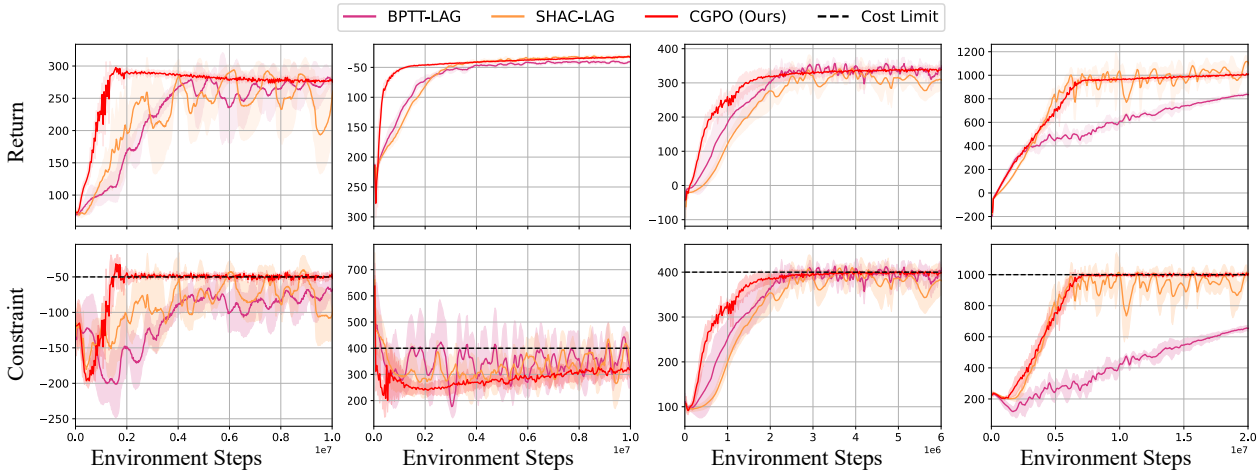
D.1. Main Results

To ensure clarity, we avoid placing numerous training curves on a single graph as this can lead to difficulties in differentiation. Given the space constraints in the main text, the comprehensive set of training curves for all algorithms is provided exclusively in the appendix. Additionally, the baseline algorithms have been categorized into two distinct groups: Traditional Safe RL Algorithms and Lagrangian-Revised Differentiable RL Algorithms.



(a) CartPole (b) Reacher (c) Half Cheetah (d) Ant

Figure 12. Training curves compared to the Traditional Safe RL Algorithms on different tasks, showing episodic return and constraint for 5 random seeds. Solid lines represent the mean, while the shaded areas indicate variance, without any smoothing to the curves.



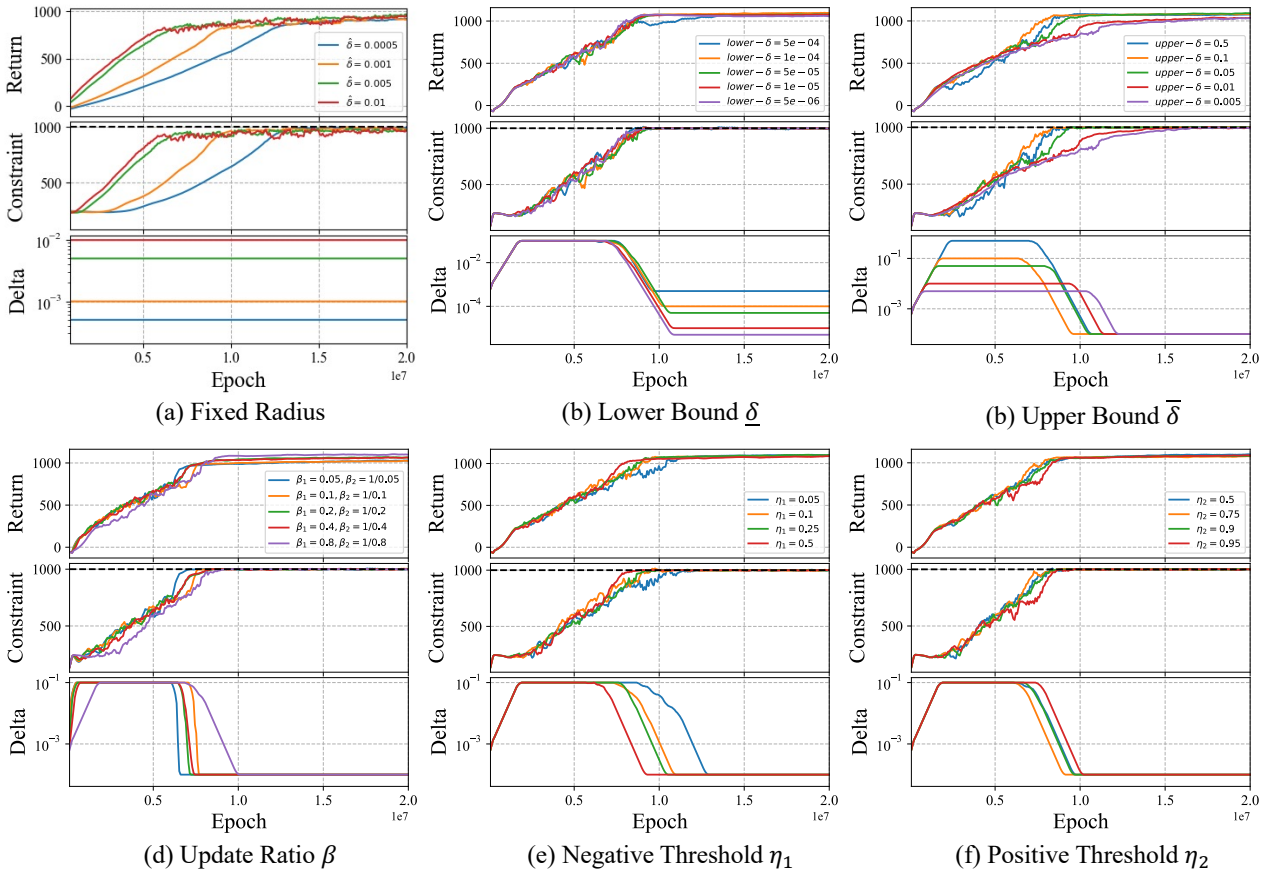
(a) CartPole (b) Reacher (c) Half Cheetah (d) Ant

Figure 13. Training curves compared to the Lagrangian-revised differentiable Safe RL algorithms on different tasks, showing episodic return and constraint for 5 random seeds. Solid lines represent the mean, while the shaded areas indicate variance, without any smoothing to the curves.

D.2. More Ablation Experiments on Radius Adaptation

Large changes in hyper-parameter $\hat{\delta}$ have some impact on the convergence speed and constraint satisfaction, as shown in Figure 14 (a). Therefore, we propose an adaptive delta setting method, which can automatically adjust the value of delta according to the current training status, further reducing the difficulty of setting this hyperparameter. To validate the

effectiveness of the adaptive method in managing hyperparameter settings, we conducted a series of ablation studies. The outcomes of these experiments are presented below:



(a) Fixed Radius (b) Lower Bound $\underline{\delta}$ (b) Upper Bound $\overline{\delta}$

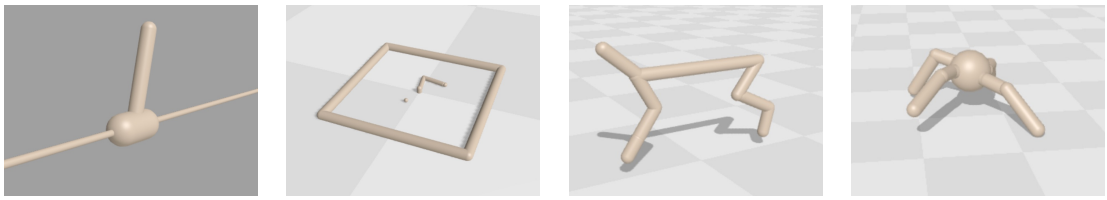
(d) Update Ratio β (e) Negative Threshold η_1 (f) Positive Threshold η_2

Figure 14. Ablation studies on the hyper-parameters of the adaptive method of trust-region radius.

Figure 14 (b) - (f) illustrate that the adaptive approach's hyper-parameters exhibit robustness, demonstrating its effectiveness in transforming a challenging-to-tune hyper-parameter into several more manageable hyper-parameters.

E. Implementation of Differentiable Safe RL Tasks

Our algorithm requires computing first-order gradients along trajectories, necessitating a guarantee of system differentiability. Given the rapid advancement of differentiable physics engines (Degrave et al., 2019; Werling et al., 2021; Xian et al., 2023), we selected robotic control tasks for our tests. We developed the first Differentiable Safe RL task series, based on four robotic control tasks in the fully differentiable Brax engine (Freeman et al., 2021), as shown in Figure 15 We introduced two common constraints to these four unconstrained tasks: one on position (Achiam et al., 2017; Yang et al., 2018; Ji et al., 2023b) and another on velocity (Zhang et al., 2020; Shi et al., 2019). The specific constraint settings may lack sufficient physical significance, yet they suffice for algorithm testing. This section details the settings of these tasks.



(a) CartPole (b) Reacher (c) HalfCheetah (d) Ant

Figure 15. Four distinct agents used in our experiments originate from Brax, a fully differentiable physics engine.

E.1. Position-constrained CartPole

This environment features a cart that moves linearly with one end of a pole fixed to it and the other end free. The goal is to push the cart left or right to balance the pole on its top by applying forces. The constraint requires maintaining the cart's position at a specific distance from the origin. Since the optimal policy involves the cart staying stationary at its initial position, the constraint introduces a contradiction between the constraints and the goal.

Action Space: The agent takes a 1-element vector for actions. The action space is a continuous range, $a \in [-3, 3]$. The action denotes the numerical force applied to the cart, with its magnitude indicating the force amount and its sign in the direction.

Observation Space: The state space includes positional values of the pendulum system's body parts, followed by the velocities of these parts, with positions listed before velocities. The observation is a vector with a shape of $(4,)$.

Reward Function: Receives a reward of 1.0 at each time step, i.e., $r_t = 1.0$.

Cost Function: Incurs a cost equal to the negative square of the cart's position at each time step, i.e., $c_t = -\|\boldsymbol{x}_{\text{pos}}\|^2$.

Constraint: The cumulative cost on a trajectory of length 300 for the agent should be less than -50.0.

E.2. Position-constrained Reacher

Reacher is a two-jointed robot arm. The goal is to guide the robot's end effector (fingertip), close to a target appearing at a random location. We set a constraint opposite to the objective function, requiring that the fingertip must not be too close to the target point.

Action Space: The action space is defined as a bounded box, designated as Box(-1, 1, (2,), float32). This notation represents a two-dimensional continuous space, with each action represented by a pair (a, b). These values represent the torques' magnitudes applied to the hinge joints, enabling precise control of the system's movements.

Observation Space: The observation data includes key aspects: cosine and sine values of the two arms' angles, target coordinates, arms' angular velocities, and a three-dimensional vector from the target to the fingertip. This information is compiled into a vector with a shape of $(11,)$.

Reward Function: At each time step, the reward function comprises two parts: a distance reward $r_{\text{dist}} = -\|\boldsymbol{x}_{\text{pos}}\|$ and a control reward $r_{\text{ctrl}} = -(a^2 + b^2)$, and $r_t = r_{\text{dist}} + \beta_{\text{ctrl}} \cdot r_{\text{ctrl}}$. Here, β_{ctrl} is a pre-configured coefficient.

Cost Function: Incurs a cost equal to the square of the position of the fingertip at each time step, i.e., $c_t = \|\boldsymbol{x}_{\text{pos}}\|^2$.

Constraint: The cumulative cost on a trajectory of length 300 for the agent should be less than 400.0.

E.3. Velocity-constrained HalfCheetah

The HalfCheetah, a 2D robot, features 9 linked segments and 8 joints, with two joints designed to mimic paws. The main goal is to strategically apply torque to these joints to make the cheetah sprint forward swiftly. The agent gets a positive reward for distance moved forward and a negative reward for moving backward. Notably, the cheetah's torso and head are static, with a torque applied only to six joints: the front and back upper legs (attached to the torso), lower legs (attached to the upper legs), and feet (attached to the lower leg). The constraint involves requiring the velocity to be less than a threshold, as referenced from Zhang et al. (2020).

Action Space: The action space is defined as a bounded box, designated as Box(-1, 1, (6,), float32). The agents operate using a set of six actions, each represented by a vector element. The values for these actions fall within the range of -1.0 to 1.0.

Observation Space: In the state space, the cheetah's body parts' positions are detailed first, followed by their velocities, representing their rate of change. This information is organized with all positional data listed before the velocity data. The observation is a vector shaped as $(18,)$.

Reward Function: At each time step, the reward function comprises two parts: an x-axis velocity reward $r_{\text{vel}} = v_x$ and a control reward $r_{\text{ctrl}} = \|\boldsymbol{a}\|^2$, and $r_t = r_{\text{dist}} + \beta_{\text{ctrl}} \cdot r_{\text{ctrl}}$. Here, β_{ctrl} is a pre-configured coefficient.

Cost Function: At each time step, the incurred cost is equal to the value of the velocity, denoted as $c_t = \|\boldsymbol{v}\|$.

Constraint: The cumulative cost on a trajectory of length 200 for the agent should be less than 400.0. The cost thresholds are calculated using 50% of the speed attained by an unconstrained PPO agent after training for sufficient samples (Zhang et al., 2020).

E.4. Velocity-constrained Ant

The Ant is a 3D robot with a torso-like, freely rotating body and four attached legs. Each leg comprises two segments joined by hinges. The goal is to maneuver these legs coordinately, propelling the robot forward and to the right. This is achieved by precisely applying forces to the eight hinges connecting the leg segments to the torso, managing a system of nine parts and eight hinges. The constraint involves requiring the velocity to be less than a threshold, as referenced from Zhang et al. (2020).

Action Space: The action space is defined as a bounded box, designated as Box(-1, 1, (8,), float32). The agent operates with an 8-element vector, where each element represents an action. The action space is continuous, spanning eight components in the range of -1.0 to 1.0. These components indicate the numerical torques at hinge joints, with each action representing torque intensity and direction.

Observation Space: The state space represents the ant's body, detailing the positions of its body parts followed by their corresponding velocities. This is represented by a 27-element vector, a numerical array that captures a comprehensive view of the ant's physical state. The observation is a vector shaped as $(27,)$.

Reward Function: At each time step, the reward function comprises two parts: an x-axis velocity reward $r_{\text{vel}} = v_x$ and a control reward $r_{\text{ctrl}} = \|\boldsymbol{a}\|^2$, and $r_t = r_{\text{dist}} + \beta_{\text{ctrl}} \cdot r_{\text{ctrl}}$. Here, β_{ctrl} is a pre-configured coefficient.

Cost Function: At each time step, the incurred cost is equal to the value of the velocity, denoted as $c_t = \|\boldsymbol{v}\|$.

Constraint: The cumulative cost on a trajectory of length 300 for the agent should be less than 1000.0. The cost thresholds are calculated using 50% of the speed attained by an unconstrained PPO agent after training for sufficient samples (Zhang et al., 2020).

F. Experimental Details and Hyper-parameters

This chapter outlines our experimental setup, covering baseline algorithm comparisons (Section F.1), evaluation metrics (Section F.2), specific hyper-parameters, training equipment details (Section F.3), and ablation study insights on utilizing gradients from trained world models (Section F.4).

F.1. Implementation of Baseline Algorithms

We implemented traditional safe reinforcement learning algorithms such as PDO (Chow et al., 2018), APPO (Dai et al., 2023), CPO (Achiam et al., 2017), and CUP (Yang et al., 2022), strictly following their original papers. For the training process, we refer to widely-recognized benchmarks like Safety-Gym (Ray et al., 2019) and OmniSafe (Ji et al., 2023c), achieving the optimal algorithm performance through multiple parameter adjustments.

Our discussion will focus on the BPTT and SHAC algorithms, which have been modified using the Lagrangian method to meet constraints and are named BPTT-Lag and SHAC-Lag, respectively. These methods are also implemented based on the *Gradient-based Estimation* of the objective and constraint functions, thereby solving the subsequent surrogate optimization problem:

$$\begin{aligned} \boldsymbol{\theta}_{k+1} = \arg\max_{\boldsymbol{\theta}\in\Theta} \quad & (\boldsymbol{\theta}-\boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k) \\ \text{s.t.} \quad & \mathcal{J}_C(\boldsymbol{\theta}_k) + (\boldsymbol{\theta}-\boldsymbol{\theta}_k)^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k) \leq b \end{aligned} \tag{75}$$

Notations. Considering the k^{th} iteration, we also introduce additional notations to make the discussion more concise: $\boldsymbol{g}_k \triangleq \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k)$, $\boldsymbol{g}_q \triangleq \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k)$, $c_k \triangleq \mathcal{J}_C(\boldsymbol{\theta}_k) - b$, and $\boldsymbol{\delta} \triangleq \boldsymbol{\theta} - \boldsymbol{\theta}_k$.

With these definitions, we rewrite the surrogate problem (75):

$$\max_{\boldsymbol{\delta}} \ \boldsymbol{g}_k^\top \boldsymbol{\delta} \quad \text{s.t. } c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta} \leq 0. \tag{76}$$

By introducing Lagrangian multipliers λ, we construct the dual function of the above constrained problem:

$$L(\boldsymbol{\delta}, \lambda) \triangleq -\boldsymbol{g}_k^\top \boldsymbol{\delta} + \lambda \cdot (c_k + \boldsymbol{q}_k^\top \boldsymbol{\delta}) \tag{77}$$

By solving the following primal-dual problem,

$$\max_{\lambda} \min_{\boldsymbol{\delta}} \ L(\boldsymbol{\delta}, \lambda) \tag{78}$$

iteratively updating the policy parameters $\boldsymbol{\theta}$ and the dual variable λ, we can approximate the local optimal point $(\boldsymbol{\theta}^*, \lambda^*)$.

In this process, $\boldsymbol{g}_k = \nabla_{\boldsymbol{\theta}} \mathcal{J}_R(\boldsymbol{\theta}_k)$, $\boldsymbol{g}_q = \nabla_{\boldsymbol{\theta}} \mathcal{J}_C(\boldsymbol{\theta}_k)$ can be provided through both the BPTT and SHAC methods. Thus, we obtain two similar algorithms as follows.

BPTT-Lag: We refer to the method of deriving $\boldsymbol{g}_k$ and $\boldsymbol{q}_k$ through the following loss functions

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T-1} r(\mathbf{s}_t^i, \mathbf{a}_t^i) \Big], \tag{79}$$

$$\mathcal{L}_C(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=0}^{T-1} c(\mathbf{s}_t^i, \mathbf{a}_t^i) \Big]. \tag{80}$$

and iteratively solving Equation (78) to update the policy parameters as BPTT-Lag.

SHAC-Lag: We refer to the method of deriving $\boldsymbol{g}_k$ and $\boldsymbol{q}_k$ through the following loss functions

$$\mathcal{L}_R(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=t_0}^{t_0+h-1} r(\mathbf{s}_t^i, \mathbf{a}_t^i) + V_\phi^R(\mathbf{s}_{t_0+h}^i) \Big], \tag{81}$$

$$\mathcal{L}_C(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \Big[\sum_{t=t_0}^{t_0+h-1} c(\mathbf{s}_t^i, \mathbf{a}_t^i) + V_\psi^C(\mathbf{s}_{t_0+h}^i) \Big]. \tag{82}$$

and iteratively solving Equation (78) to update the policy parameters as SHAC-Lag.

F.2. Evaluation Metrics

In Section 6, we apply two evaluation metrics to quantify the convergence efficiency and the satisfaction of constraints, respectively.

F.2.1. Number of Environment Steps at Convergence (Conv. Steps)

This evaluation metric measures the efficiency of algorithmic performance improvement by determining the number of environmental steps needed for convergence on the episodic return curve. The convergence point on the episodic return curve is defined as the point from which the range of variation within a given window is less than a set threshold. In Safe RL tasks, we hope the algorithm can converge to a feasible point, so it is additionally required that this convergence point meets the constraints.

The example code for calculating this evaluation metric is as follows:

```
def find_convergence_point(
    env_step, returns, constraints, constraint_threshold, window_size, threshold
):
    for i in range(len(returns)):
        for j in range(i + 1, min(i + window_size, len(returns))):
            if (
                (np.abs(returns[j] - returns[i]) > threshold)
                or (constraints[i] > constraint_threshold)
            ):
                break
        else:
            return env_step[i]

    return -1
```

F.2.2. Violation Ratio within Safety Critical Areas (Vio. Ratio)

Safety critical areas are defined as key regions where safety issues frequently occur (Nilsson et al., 2017; Xu et al., 2022). Thus, updates nearing the constraint threshold in the policy update process are considered to occur within the safety critical areas. Consequently, we use the proportion of constraint violations in safety critical areas during the policy update process as a measure of the capability of the update algorithm to meet constraints. Additionally, in deep Safe RL, where safety constraints are expressed as the expected value of sampling, a soft margin is introduced to determine if the policy violates constraints. The example code for calculating this evaluation metric is as follows:

```
def violation_ratio_in_critic_area(
    constraints, constraint_threshold, critic_area_radius, soft_margin
):
    num_in_critic_area = np.sum(constraints > (constraint_threshold - critic_area_radius))
    num_violation = np.sum(constraints > (constraint_threshold + soft_margin))
    if num_in_critic_area > 0:
        return num_violation / num_in_critic_area
    else:
        return -1
```

F.3. Hyper-parameters

In this section, we present the hyper-parameters for CGPO across four distinct tasks.

Table 4. Hyper-parameters for CGPO in different tasks.

Tasks	CartPole	Reacher	HalfCheetah	Ant
num_epochs	100	300	300	300
num_envs	128	128	128	256
episode_length	300	300	200	300
delta_init	1e-3	1e-2	1e-3	1e-3
delta_upper	1e-2	1e-1	1e-2	1e-2
delta_lower	1e-4	1e-4	1e-4	1e-4
beta_1	0.8	0.8	0.8	0.8
beta_2	1.25	1.25	1.25	1.25
critic_learning_rate	1e-3	1e-3	1e-3	1e-3
short_horizon	10	10	10	10
mini_batch_size	64	64	64	64
max_gradient_norm	1e9	1e9	1e9	1e9
normalize_observations	True	True	True	True
cost_limit	-50	400	400	1000

F.4. Ablation Experiments with World Model

We demonstrated through ablation experiments that CGPO has the potential to obtain the system's analytic gradient by training a world model, which is detailed in the description of this experiment.

F.4.1. Implementation of Non-Differentiable Tasks

Non-differentiable Function Environment. We design a function environment similar to the simple environment in Figure 1. The goal of this task is to train an agent to move along the x-axis, receiving rewards and incurring costs based on its position on the x-axis. The specific settings of the task are as follows:

Action Space: The direction and step length of the agent's movement along the x-axis constitute the action space, which is a single-element vector, i.e., $a \in (-1, 1)$.

Observation Space: The observation data for the agent is its position on the x-axis, which is also a single-element vector, i.e., $s \in \mathbb{R}$. The task employs a simple, artificially set state transition function: $s_{t+1} = s_t + 0.2 \times a_t$.

Reward Function and Cost Function: Since the purpose of this simple task is to verify the accuracy of the estimation method, we set the reward function and cost function in the same form, i.e.,

$$f(x) = \left(\frac{\text{floor}(x)}{10.0}\right)^2 + 0.1 + 0.1 \times \sin\left(\frac{8x}{\pi}\right) \tag{83}$$

where $\text{floor}(x)$ rounds x down to the nearest integer, and its visualization is shown in Figure 16.

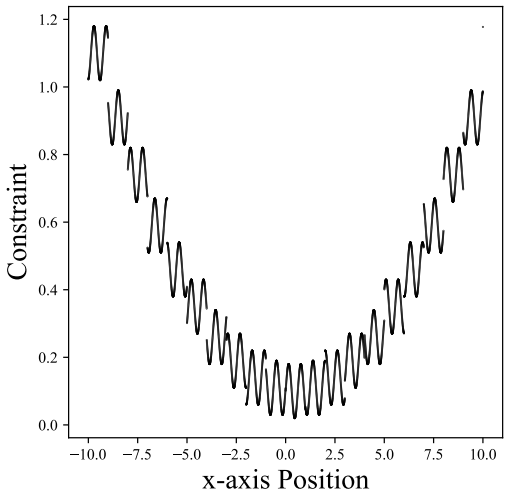


Figure 16. Cost values at different x-axis positions for the agent.

Constraint: Since the task is solely aimed at verifying the accuracy of the estimation method, the set constraint has no

physical significance. Specifically, the cumulative cost on a trajectory of length 100 for the agent should be less than 8.0:

$$\mathcal{J}_C(\pi) \triangleq \sum_{t=0}^{99} f(x_t) \leq 8.0 \tag{84}$$

This constraint setting is consistent with many well-known Safe Benchmarks, where the undiscounted cumulative cost on a finite-length trajectory must be less than a scalar threshold, namely, $\sum_{t=0}^{T-1} c_t \leq b$.

Non-differentiable Swimmer. The swimmers consist of three or more segments, known as 'links', connected by a slightly smaller number of articulation joints, termed 'rotors'. Each rotor joint smoothly connects two links, forming a linear sequence. These swimmers elegantly move through a two-dimensional aquatic environment. Their goal is to glide swiftly to the right, utilizing the torque applied to the rotors and effectively using the water's resistance. The constraint involves requiring the velocity to be less than a threshold, as referenced from Zhang et al. (2020). This task has poor differentiability because the differentiable kernel of Brax is not used.

Action Space: The action space is defined as a bounded box, designated as Box(-1, 1, (2,), float32). The agent takes a 2-element vector for actions. The values for these actions fall within the range of -1.0 to 1.0. It represents the torque applied on the rotor.

Observation Space: The state space is composed of a series of elements that define the position and movement dynamics within the environment. Specifically, these elements provide detailed information about the positions, angles, and their respective velocities and angular velocities of the segments in our system. The observation is a vector shaped as $(8,)$.

Reward Function: At each time step, the reward function comprises two parts: an x-axis velocity reward $r_{\text{vel}} = v_x$ and a control reward $r_{\text{ctrl}} = \|\boldsymbol{a}\|^2$, and $r_t = r_{\text{dist}} + \beta_{\text{ctrl}} \cdot r_{\text{ctrl}}$. Here, β_{ctrl} is a pre-configured coefficient.

Cost Function: At each time step, the incurred cost is equal to the value of the velocity, denoted as $c_t = \|\boldsymbol{v}\|$.

Constraint: The cumulative cost on a trajectory of length 300 for the agent should be less than 150.0. The cost thresholds are calculated using 50% of the speed attained by an unconstrained PPO agent after training for sufficient samples (Zhang et al., 2020).

Algorithm 3 Model-based Constrained Gradient-based Policy Optimization (MB-CGPO)

Input: Initialize World Model $\mathcal{F}_{\varphi_0}$, policy $\pi_{\boldsymbol{\theta}_0}$, critic $V_{\boldsymbol{\phi}_0}$ and $V^c_{\boldsymbol{\psi}_0}$, radius $\hat{\delta}_0$, and number of iterations K.
for $k = 1, 2, \ldots, K$ **do**
  Sample a set of trajectories $\mathcal{D} = \{\tau\} \sim \pi_{\boldsymbol{\theta}_k}$.
  Update World Model $\mathcal{F}_{\varphi_k}$ with data $\mathcal{D}$ using (85).
  Sample a set of trajectories $\mathcal{D}_M = \{\tau\} \sim \pi_{\boldsymbol{\theta}_k}$ in World Model $\mathcal{F}_{\varphi_{k+1}}$.
  Compute the $\boldsymbol{g}_k$, $\boldsymbol{q}_k$ using (19) and (20) with $\mathcal{D}_M$.
  if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta}_k \geq 0$ and $c_k > 0$ **then**
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \frac{\sqrt{\hat{\delta}_k}}{\|\boldsymbol{q}_k\|}\boldsymbol{q}_k$.
  else if $c_k^2/\boldsymbol{q}_k^\top \boldsymbol{q}_k - \hat{\delta}_k \geq 0$ and $c_k < 0$ **then**
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\sqrt{\hat{\delta}_k}}{\|\boldsymbol{g}_k\|}\boldsymbol{g}_k$.
  else
    Compute dual variables λ_k^*, ν_k^* using Algorithm 2.
    Update the policy as $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \frac{\boldsymbol{g}_k - \nu_k^* \boldsymbol{q}_k}{\lambda_k^*}$.
  end if
  Update $V_{\boldsymbol{\phi}_k}$, $V^c_{\boldsymbol{\psi}_k}$ using (21), and $\hat{\delta}_{k+1}$ using (18).
end for
Output: Policy $\pi_{\boldsymbol{\theta}_K}$.

F.4.2. Basic Model-based CGPO algorithm

Here, we present a basic implementation of using a World Model to provide analytic gradients for CGPO, which we refer to as MB-CGPO. The input to this world model is s_t and a_t, and the output is a combined vector of s_{t+1}, r_t, and c_t. By using

the MSE loss, we train the world model $\mathcal{F}$ with collected data, as shown in:

$$L(s_t, s_{t+1}, r_t, c_t) = \left(\mathcal{F}_s(s_t, a_t) - s_{t+1}\right)^2 + \left(\mathcal{F}_r(s_t, a_t) - r_t\right)^2 + \left(\mathcal{F}_c(s_t, a_t) - c_t\right)^2 \tag{85}$$

When computing analytic gradients, we collect trajectories through the World Model and then derive along the trajectory in the exact same manner as with a differentiable simulator. Thus, the MB-CGPO is given in Algorithm 3.