A Review of Safe RL with Hard Safety Constraints

Ferlez *et al*. [21] propose a ShieldNN leveraging Barrier Function (BF) to design a safety filter neural network with safety guarantees. However, ShieldNN is specially designed for an environment with the kinematic bicycle model (KBM) [10] as the dynamical model, which is hard to generalize to other problem scopes. Fisac *et al*. [9] also propose a general safety framework based on Hamilton-Jacobi reachability methods that can work in conjunction with an arbitrary learning algorithm. However, these methods [9, 22] still exploit approximate knowledge of the explicit form of system dynamics to guarantee constraint satisfaction. Berkenkamp *et al*. [23] combine RL with Lyapunov analysis to ensure safe operation in discretized systems. Though provable safe control can be guaranteed under some Lipschitz continuity conditions, this method still requires the explicit knowledge of the system dynamics (analytical form). To relieve the requirements of explicit system dynamics, Grover *et al*. [24, 25] learn parameters of nominal control that is wrapped around with safety constraints using adaptive filtering approach. Dalal *et al*. [19] propose to learn the system dynamics directly from offline collected data, then add a safety layer that analytically solves a control correction formulation per each state to ensure every state is safe. However, the closed-form solution relies on the assumption of the linearized dynamics model, which is not true for most dynamics systems. They also assume the set of safe control can be represented by a linearized half-space for all states, which does not hold for most of the discrete-time system. Yinlam *et al*. [26] propose to project either the policy parameters or the action to the set induced by linearized Lyapunov constraints, which still suffers from the same linear approximation error and non control-affine systems as in [19] and is not able to guarantee zero-violation. In contrast, our proposed method does not require the explicit form of system dynamics, and our method can be guaranteed to generate safe control as long as the safe set is not a collection of singleton points, and the safe set itself can be disjoint and unconnected.

B Algorithms

The main body of AdamBA algorithm is summarized in Algorithm 1, and the main body of ISSA algorithm is summarized in Algorithm 2. The inputs for Adaptive Momentum Boundary Approximation are the approximation error bound (ϵ), learning rate (β), reference control (u^r), gradient vector covariance (Σ), gradient vector number (n), reference gradient vector ($\vec{v}^r$), and safety status of reference control (S). The inputs for ISSA are the approximation error bound (ϵ), the learning rate (β), gradient vector covariance (Σ), gradient vector number (n) and reference gradient vector ($\vec{v}^r$).

Algorithm 1 Adaptive Momentum Boundary Approximation

procedure ADAMBA($\epsilon, \beta, \Sigma, n, u^r, \vec{v}^r, S$)
  Initialize:
  if $\vec{v}^r$ is empty **then**
    Generate n Gaussian distributed unit gradient vectors $\vec{v}_i \sim \mathcal{N}(0, \Sigma)$, $i = 1, 2, \ldots, n$
  else
    Initialize one unit gradient vector $\vec{v}_1 = \frac{\vec{v}^r}{\|\vec{v}^r\|}$
  Approximation:
  for $i = 1, 2, \cdots, n$ **do**
    Initialize the approximated boundary point $P_i = u^r$, and *stage* = *exponential outreach*.
    while *stage* = *exponential outreach* **do**
      $P_i = P_i + \vec{v}_i\beta$
      if P_i is out of control set **then**
        Discard P_i
        break
      if P_i safety status $\neq S$ **then**
        stage = *exponential decay*
        break
      $\beta = 2\beta$
    while $\beta \geq \epsilon$ and *stage* = *exponential decay* **do**
      $\beta = 0.5\beta$
      $P_i = P_i + \vec{v}_i\beta$, if P_i safety status $= S$
      $P_i = P_i - \vec{v}_i\beta$, if P_i safety status $\neq S$
  Return Approximated Boundary Set P

Algorithm 2 Implicit Safe Set Algorithm (ISSA)

procedure ISSA($\epsilon, \beta, \Sigma, n, u^r$)
    Phase 1: ▷ Phase 1
    Use AdamBA($\epsilon, \beta, \Sigma, n, u^r, \emptyset$, *UNSAFE*) to sample a collection $\mathbb{S}$ of safe control on the boundary of $\mathcal{U}_S^D$.
    if $\mathbb{S} = \emptyset$ **then**
        Enter **Phase 2**
    else
        For each primitive action $u_i \in \mathbb{S}$, compute the deviation $d_i = \|u_i - u^r\|^2$
        return $\arg\min_{u_i} d_i$

    Phase 2: ▷ Phase 2
    Use grid sampling by iteratively increasing sampling resolution to find an anchor safe control u^a, s.t. safety status of u^a is *SAFE*.
    Use AdamBA(ϵ, $\frac{\|u^r - u^a\|}{4}, \Sigma, 1, u^r$, $\frac{u^a - u^r}{\|u^a - u^r\|}$, *UNSAFE*) to search for boundary point u^*
    if u^* is not found **then**
        Use AdamBA(ϵ, $\frac{\|u^r - u^a\|}{4}, \Sigma, 1, u^a$, $\frac{u^r - u^a}{\|u^r - u^a\|}$, *SAFE*) to search for boundary point u^{a*}
        Return u^{a*}
    else
        Return u^*

C Proof of Proposition 1

The motivation for synthesizing the safety index is to ensure that 1) there exists a nontrivial subset of the user-defined safe set that is forward invariant, and 2) at every state, there always exists a safe control such that $\phi(x_{t+1}) \leq \max\{\phi(x_t) - \eta, 0\}$. We consider the safety index as $\phi = \max_i \phi_i$, where ϕ_i has the following form:

$$\phi_i = \sigma + d_{min}^n - d_i^n - k\dot{d}_i, \tag{5}$$

where d_i denotes the relative distance between the robot and the i-th obstacle, and $\sigma, n, k, \eta \in \mathbb{R}^+$ are tunable parameters. This form works for collision avoidance for general second-order systems.

In the following discussions, we will prove proposition 1 specifically in collision avoidance in a 2D plane, where the dynamics are in the general form $x_{t+1} = f(x_t, u_t)$, not necessarily control affine.

C.1 Preliminary Results

Suppose d is the relative distance between the robot and the obstacle in 2D plane, where the obstacle is not necessarily static. We have $\ddot{d} = -a\cos(\alpha) + v\sin(\alpha)w$ and $\dot{d} = -v\cos(\alpha)$, where a is the relative acceleration of the robot in the obstacle frame. $\vec{v}$ is the relative velocity vector of the robot in the obstacle frame with magnitude v, α is the angle between $\vec{v}$ and vector from robot to obstacle, and $w = \dot{\alpha}$ is the relative angular velocity of the robot in the obstacle frame. According to Assumption 1, we have 1) $w \in [w_{min}, w_{max}]$, where $w_{min} \leq 0$ and $w_{max} \geq 0$; and 2) $a \in [a_{min}, a_{max}]$, where $a_{min} \leq 0$ and $a_{max} \geq 0$.

Before proving Proposition 1, we present preliminary results that are useful towards proving the Proposition. Note that the set of safe control $\mathcal{U}_S^D(x) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ is non-empty if it is non-empty in the following two cases: $\phi(x) - \eta < 0$ or $\phi(x) - \eta \geq 0$. Lemma 1 shows that the safety index design rule guarantees non-empty set of safe control when $\phi(x) - \eta \geq 0$. Lemma 2 shows the set of safe control is non-empty when $\phi(x) - \eta < 0$.

Lemma 1. *If the dynamic system satisfies the assumptions in Assumption 1 and there's only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures that the robot system in 2D plane has nonempty set of safe control at state where $\phi(x) - \eta \geq 0$.*

Proof. When $\phi(x) - \eta \geq 0$, the condition $\phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}$ becomes the condition:

$$\phi(f(x,u)) \leq \phi(x) - \eta \tag{6}$$

According to the last assumption in Assumption 1, we have $dt \to 0$. Therefore, the discrete time approximation error approaches zero, and $\phi(f(x,u)) = \phi(x) + dt \cdot \dot{\phi}(x,u) + \Delta$, where $\Delta \to 0$. Then we can rewrite (6) as:

$$\dot{\phi} \leq -\eta/dt \tag{7}$$

According to (5) and the condition that environment has only one obstacle, the condition on $\dot{\phi}$ is equivalent to $-nd^{n-1}\dot{d} - k\ddot{d} \leq -\eta/dt$. Therefore, in order to get a feasible control satisfying (7), we need to make sure that the following statement holds:

$$\forall x, \exists u, \text{s.t. } \ddot{d} \geq \frac{\eta/dt - nd^{n-1}\dot{d}}{k}. \tag{8}$$

According to Assumption 1, we have that for all possible values of a and w, there always exists a control u to realize such a and w. Based on state variables α, and v, the control variables a and w, and use the relationships between $\ddot{d}, \dot{d}$ and these state and control variables, we rewrite (8) as the following:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t. } -a\cos(\alpha) + v\sin(\alpha)w \geq \frac{\eta/dt + nd^{n-1}v\cos(\alpha)}{k} \tag{9}$$

According to the safety index design rule, we have $\eta = 0$. Note that velocity v is always a non-negative value. If $v = 0$, we have RHS of (9) is 0, and the LHS of (9) is $-a\cos(\alpha)$. Since

$a \in [a_{min}, a_{max}]$, where $a_{min} \leq 0$ and $a_{max} \geq 0$, there always exists a control such that LHS of (9) is non-negative, which satisfies (9).

Thus, we only need to consider $v > 0$, and prove that the safety index design rule can ensure nonempty set of safe control by satisfying the following condition :

$$\forall(\alpha, v), \exists(a, w), \text{ s.t. } -a\frac{\cos(\alpha)}{v} + \sin(\alpha)w \geq \frac{nd^{n-1}\cos(\alpha)}{k} \tag{10}$$

There are only three scenarios, $\cos(\alpha) > 0$, $\cos(\alpha) = 0$, or $\cos(\alpha) < 0$. First, we check the simplest case, which is $\cos(\alpha) = 0$. When $\cos(\alpha) = 0$, the RHS of the (10) is zero, and $-a\frac{\cos(\alpha)}{v}$ is also zero. Note that since w is the angular velocity action which can be either positive or negative, thus no matter $\sin(\alpha)$ is 1 or -1, we can always find a w to make sure $\sin(\alpha)w > 0$.

Next, we consider the second case where $\cos(\alpha) > 0$, which means that the vehicle is moving closer towards the obstacles ($\dot{d} < 0$). Now, we rewrite (10) as the following:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t. } -\frac{a}{v} + \tan(\alpha)w \geq \frac{nd^{n-1}}{k} \tag{11}$$

and (11) can be verified by showing:

$$\min_{v,\tan(\alpha)} \max_{a,w}(-\frac{a}{v} + \tan(\alpha)w) \geq \max_{d} \frac{nd^{n-1}}{k} \tag{12}$$

To prove (12), we will find the lower bound of LHS of (12), and upper bound of RHS of (12), and establish the constraint such that lower bound of LHS is greater than upper bound of RHS.

Now we find the lower bound of LHS of (12). Firstly, we have $\tan(\alpha) \in (-\inf, \inf)$ and $w \in [w_{min}, w_{max}]$, we have $\min_{\tan(\alpha)} \max_{w} \tan(\alpha)w \geq 0$. Secondly, since $a \in [a_{min}, a_{max}]$ and $v \in [0, v_{max}]$, we have:

$$\min_{v,\tan(\alpha)} \max_{a,w}(-\frac{a}{v} + \tan(\alpha)w) \geq \frac{-a_{min}}{v_{max}} \tag{13}$$

Since we are considering $\phi(x) - \eta \geq 0$, which is equivalent to $d \leq (\sigma + d_{min}^n - k\dot{d})^{1/n}$, thus the following condition holds:

$$\max_{d} \frac{nd^{n-1}}{k} \leq \max_{\dot{d}} \frac{n(\sigma + d_{min}^n - k\dot{d})^{\frac{n-1}{n}}}{k} = \frac{n(\sigma + d_{min}^n + kv_{max})^{\frac{n-1}{n}}}{k} \tag{14}$$

According to the safety index design rule, where $\frac{n(\sigma+d_{min}^n+kv_{max})^{\frac{n-1}{n}}}{k} \leq \frac{-a_{min}}{v_{max}}$. The following condition holds:

$$\min_{v,\tan(\alpha)} \max_{a,w}(-\frac{a}{v} + \tan(\alpha)w) \geq \frac{-a_{min}}{v_{max}} \geq \frac{n(\sigma + d_{min}^n + kv_{max})^{\frac{n-1}{n}}}{k} \geq \max_{d} \frac{nd^{n-1}}{k} \tag{15}$$

Finally, we consider the third case where $\cos(\alpha) < 0$, which means that the vehicle is moving away from the obstacles ($\dot{d} > 0$). We can rewrite (10) as following:

$$\forall(\alpha, v), \exists(a, w), \text{ s.t. } -\frac{a}{v} + \tan(\alpha)w \leq \frac{nd^{n-1}}{k} \tag{16}$$

which can be verified by showing:

$$\max_{v,\tan(\alpha)} \min_{a,w} -\frac{a}{v} + \tan(\alpha)w \leq \min_{d} \frac{nd^{n-1}}{k} \tag{17}$$

Similarly, we can show that:

$$\max_{v,\tan(\alpha)} \min_{a,w}(-\frac{a}{v} + \tan(\eta)w) \leq -\frac{a_{max}}{v_{max}} \tag{18}$$

and:

$$\min_d \frac{nd^{n-1}}{k} \geq 0 \tag{19}$$

Therefore, regardless of k, n, and σ value, (16) is always satisfied.

In summary, we have proven that if the control system satisfies the assumptions in Assumption 1 and there's only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures that the robot system in 2D plane has nonempty set of safe control at state where $\phi(x) - \eta \geq 0$. □

Lemma 2. *If the dynamic system satisfies the assumptions in Assumption 1 and there's only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures any control is a safe control for robot system in 2D plane at state where $\phi(x) - \eta < 0$.*

Proof. When $\phi(x) - \eta < 0$, the condition $\phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}$ becomes the condition:

$$\phi(f(x,u)) \leq 0 \tag{20}$$

According to the last assumption in Assumption 1, we have $dt \to 0$. Therefore, the discrete time approximation error approaches zero, and $\phi(f(x,u)) = \phi(x) + dt \cdot \dot{\phi}(x,u) + \Delta$, where $\Delta \to 0$. Then we can rewrite (20) as:

$$\dot{\phi} \leq -\phi/dt \tag{21}$$

According to (5) and the condition that environment has only one obstacle, the condition on $\dot{\phi}$ is equivalent to $-nd^{n-1}\dot{d} - k\ddot{d} \leq -\phi/dt$. Therefore, in order to get a feasible control satisfying (21), we need to make sure that the following statement holds:

$$\forall x, \exists u, \text{s.t. } \ddot{d} \geq \frac{\phi(x)/dt - nd^{n-1}\dot{d}}{k} \tag{22}$$

Since $\eta = 0$ according to the safety index design rule, $\phi(x) - \eta < 0$ indicates $\phi(x) < 0$. Since $dt \to 0$ according to Assumption 1, we further have $\phi(x)/dt \to -\infty$. According to Assumption 1, we have the state space is bounded, thus both d and $\dot{d}$ are bounded, which indicates $nd^{n-1}\dot{d}$ is also bounded. Therefore, the following condition holds:

$$\frac{\phi(x)/dt - nd^{n-1}\dot{d}}{k} \to -\infty \tag{23}$$

We have for all possible values of a and w, there always exists a control u to realize such a and w according to Assumption 1, which indicates the mapping from u to a, w is a non-injective surjective function. Since a and w are bounded and both can achieve zeros according to Assumption 1, we have $\forall u$, the corresponding a, w are bounded. Since $\ddot{d} = -a\cos(\alpha) + v\sin(\alpha)w$, the following condition holds:

$$\forall x, \forall u, \text{s.t. } \ddot{d} \text{ is bounded} \tag{24}$$

which indicates (22) holds. Therefore, we have proven that if the dynamic system satisfies the assumptions in Assumption 1 and there's only one obstacle in the environment, then the safety index design rule in Section 4.1 ensures that any control is a safe control for robot system in 2D plane at state where $\phi(x) - \eta < 0$. □

C.2 Proof of Proposition 1

Proof. Note that the set of safe control $\mathcal{U}_S^D(x_t) := \{u \in \mathcal{U} \mid \phi(f(x,u)) \leq \max\{\phi(x) - \eta, 0\}\}$ is non-empty if it is non-empty in the following two cases: $\phi(x) - \eta < 0$ or $\phi(x) - \eta \geq 0$.

Firstly, we consider the case where $\phi(x) - \eta \geq 0$. We have $\phi(x) - \eta = \max_i \phi_i(x) - \eta \geq 0$, where ϕ_i is the safety index with respect to the i-th obstacle. According to assumption 1, we have that at any given time, there can at most be one obstacle becoming safety critical, such that $\phi - \eta \geq 0$

(Sparse Obstacle Environment). Therefore, $\max_i \phi_i(x) - \eta \geq 0$ indicates there's only one obstacle (j-th obstacle) in the environment that $\phi_j(x) - \eta \geq 0$. Whereas for the rest of the obstacles, we have $\phi_k(x) - \eta < 0, k \neq j$.

Denote $\mathcal{U}_{S\ j}^{D}(x) := \{u \in \mathcal{U} \mid \phi_j(f(x,u)) \leq \phi(x)_j - \eta\}$ where $\phi_j(x) - \eta \geq 0$. According to Lemma 1, we have if the dynamic system satisfies the assumptions in Assumption 1, then the safety index design rule in Section 4.1 ensures $\mathcal{U}_{S\ j}^{D}(x)$ is nonempty.

Denote $\mathcal{U}_{S\ k}^{D}(x) := \{u \in \mathcal{U} \mid \phi_k(f(x,u)) \leq 0\}$ where $\phi_k(x) - \eta < 0$ and $k \neq j$. According to Lemma 2, we have if the dynamic system satisfies the assumptions in Assumption 1, then the safety index design rule in Section 4.1 ensures $\mathcal{U}_{S\ k}^{D}(x) = \mathcal{U}$. Since $\mathcal{U}_{S\ j}^{D}(x) \subset \mathcal{U}$, we have $\mathcal{U}_{S\ j}^{D}(x) \subset \mathcal{U}_{S\ k}^{D}(x)$. Thus, we have that $\phi_k(f(x,u)) \leq 0 \leq \phi_j(x) - \eta$, for $u \in \mathcal{U}_{S\ j}^{D}(x)$.

Therefore, we further have that if $\phi(x) - \eta \geq 0$, by applying $u \in \mathcal{U}_{S\ j}^{D}(x)$, the following condition holds:

$$\phi(f(x,u)) = \max_i \phi_i(f(x,u)) \leq \phi_j(x) - \eta = \max\{\phi(x) - \eta, 0\} \tag{25}$$

Secondly, we consider the case where $\phi(x) - \eta < 0$. We have $\phi(x) - \eta = \max_i \phi_i(x) - \eta < 0$, where ϕ_i is the safety index with respect to the i-th obstacle. Therefore, we have $\forall i, \phi_i(x) - \eta < 0$.

According to Lemma 2, we have that if the dynamic system satisfies the assumptions in Assumption 1, then the safety index design rule in Section 4.1 ensures that $\forall u \in \mathcal{U}$, we have $\forall i, \phi_i(f(x,u)) < 0$. Therefore, if $\phi(x) - \eta < 0$, by applying $u \in \mathcal{U}$, the following condition holds:

$$\phi(f(x,u)) = \max_i \phi_i(f(x,u)) \leq 0 = \max\{\phi(x) - \eta, 0\} \tag{26}$$

In summary, if the dynamic system satisfies the assumptions in Assumption 1, then the safety index design rule in Section 4.1 ensures that the robot system in 2D plane has nonempty set of safe control at any state, i.e., $\mathcal{U}_S^D(x) \neq \emptyset, \forall x$. □

D Proof of Proposition 2

D.1 Preliminary Results.

Before proving Proposition 2, we present preliminary results that are useful towards proving the proposition. Lemma 3 shows that if AdamBA is able to find two consecutive control inputs with different status (safe or unsafe), then AdamBA will converge to a control input on the boundary of the set of safe control. Lemma 4 shows that if the synthesized safety index can guarantee a non-trivial set of safe control, then we can find an anchor point in phase 2 of Algorithm 2 with finite iterations. Lemma 5 shows that if we enters phase 2 of Algorithm 2, we can always find a local optima solution of (4).

Lemma 3 (Convergence)**.** *If AdamBA enters the exponential decay phase (line 16 in algorithm 1), then it can always return an approximated boundary point with approximation error upper bounded by ϵ.*

Proof. Denote P_1 as the approximated boundary point when AdamBA first enters the *exponential decay* phase and P_{-1} as the approximated boundary point at second to last iteration during *exponential outreach* phase. Note that P_1 is also the approximated boundary point at the last iteration during *exponential outreach* phase.

Denote P_t as the approximated boundary point at t-th iteration of *exponential decay* phase, we also denote β_t as the learning rate at t-th iteration of *exponential decay* phase. To proof the lemma, we define an auxiliary procedure during *exponential decay* phase as following: we maintain two points P_N and P_S, which denote one unsafe control point, and one safe control point along $\vec{v}_i$. It is obvious that P_t is either P_N or P_S. Denoting the boundary point as P_b, we have $\exists \lambda \in (0,1), P_b = \lambda P_S + (1-\lambda) P_N$. Next, we choose the $P_{t+1} = \frac{1}{2}(P_N + P_S)$, and we update P_N and P_S, such that 1) $P_N = P_{t+1}, P_S = P_S$ if P_{t+1} safety status is unsafe, 2) $P_N = P_N, P_S = P_{t+1}$ if P_{t+1} safety status is safe. Then we have that P_b is still between P_N, P_S, i.e., $\exists \lambda_1 \in (0,1), P_b = \lambda_1 P_S + (1-\lambda_1) P_N$.

Next, we will show that *exponential decay* implicitly maintains P_N and P_S as described above. Suppose at time step t, there are points P_N and P_S along unit vector $\vec{v}_i$, where P_t is either P_N or P_S, and $\|P_N - P_S\| = \beta_t$. In fact, there are four different situations: 1) $\{P_t = P_N\}$-$\{S$ is safe$\}$, 2) $\{P_t = P_S\}$-$\{S$ is safe$\}$, 3) $\{P_t = P_N\}$-$\{S$ is unsafe$\}$, and 4) $\{P_t = P_S\}$-$\{S$ is unsafe$\}$, where S is the reference control safety status.

We begin by considering the first situation ($\{P_t = P_N\}$-$\{S$ is safe$\}$). According to *exponential decay*, we have the following holds for boundary point P_{t+1}:

$$\begin{aligned} P_{t+1} &= P_t - \frac{1}{2}\vec{v}_i \beta_t \\ &= P_N - \frac{1}{2}\frac{P_N - P_S}{\|P_N - P_S\|}\|P_N - P_S\| \\ &= \frac{1}{2}(P_N + P_S) \end{aligned} \tag{27}$$

where the safety status of P_{t+1} can either be safe or unsafe. If P_{t+1} safety status is safe, then there exist $P_S = P_{t+1}$, $P_N = P_N$ along $\vec{v}_i$, where $\|P_S - P_N\| = \frac{1}{2}\beta_t$. If P_{t+1} safety status is unsafe, then there exist $P_N = P_{t+1}$, $P_S = P_S$ along $\vec{v}_i$, where $\|P_S - P_N\| = \frac{1}{2}\beta_t$. Since $\beta_{t+1} = \frac{1}{2}\beta_t$, then we have shown there are points P_N and P_S along unit vector $\vec{v}_i$ at iteration $t+1$, where P_{t+1} is either P_N or P_S, and $\|P_N - P_S\| = \beta_{t+1}$.

Similarly, we can derive the same conclusions for the other three situations ($\{P_t = P_S\}$-$\{S$ is safe$\}$, $\{P_t = P_N\}$-$\{S$ is unsafe$\}$, and $\{P_t = P_S\}$-$\{S$ is unsafe$\}$).

Note that when we first enter the *exponential decay* prodedure, we have either $\{P_S = P_1\}$-$\{P_N = P_{-1}\}$ or $\{P_N = P_1\}$-$\{P_S = P_{-1}\}$, such that $\|P_N - P_S\| = \beta_1$.

By induction, at any iteration t during *exponential decay*, there exist P_N and P_S along $\vec{v}_i$, such that P_t is either P_N or P_S, and $\|P_N - P_S\| = \beta_t$, where $\beta_t = \frac{1}{2}^{(t-1)}\beta_1$. With the existence of P_N and P_S, we further have $\exists \lambda^t \in (0,1), P_b = \lambda^t P_S + (1-\lambda^t) P_N$. Therefore, we have the following

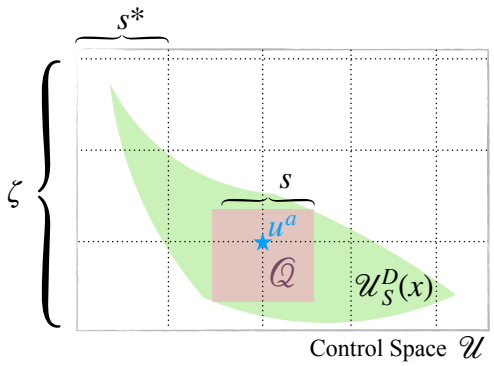


Figure 10: Illustration of the grid sampling to find anchor control point.

inequality holds:

$$\|P_t - P_b\| \leq \|P_N - P_S\| = \beta_t = \frac{1}{2}^{(t-1)} \beta_1$$

Note that the exit condition for *exponential decay* is $\frac{1}{2}^{(t-1)}\beta_1 < \epsilon$, where $\frac{1}{2}^{(t-1)} \to 0$ when $t \to \infty$. Therefore, we can always find a approximated boundary point P_t after we enter *exponential decay* phase, and the approximation error $\|P_t - P_b\|$ is upper bounded bounded by ϵ.

□

Lemma 4 (Existence)**.** *If the synthesized safety index can guarantee a non-empty set of safe control, then we can find an anchor point in phase 2 of Algorithm 2 with finite many iterations (line 11 in algorithm 2).*

Proof. By increasing k, we can always have $\frac{n(\sigma+d_{min}^n+kv_{max})^{\frac{n-1}{n}}}{k} < -\frac{a_{min}}{v_{max}}$ satisfying the safety index design rule. Therefore, we have the existence of a non-trivial set of safe control $\mathcal{U}_s^D$ such that:

$$\exists \mathcal{Q} \subset \mathcal{U}_S^D, \exists \kappa > 0, \text{ s.t. } s > \kappa \,, \tag{28}$$

where $\mathcal{Q}$ is a n_u-dimensional hypercube with the same length of s. Denote $\zeta_{[i]} = \max_{j,k} \|u_{[i]}^j - u_{[i]}^k\|$, where $u_{[i]}$ denotes the i-th dimension of control u, and $u^j \in \mathcal{U}_S^D, u^k \in \mathcal{U}_S^D$.

By directly applying grid sampling in $\mathcal{U}_s^D$ with sample interval s^* at each control dimension, such that $s^* < s$. The maximum sampling time T^a for finding an anchor point in phase 2 satisfies the following condition:

$$T^a < \prod_{i=1}^{n_u} \lceil \frac{\zeta_{[i]}}{s^*} \rceil \,, \tag{29}$$

where T^a is a finite number due to infimum condition of s in (28). Then we have proved that we can find an anchor point in phase 2 of Algorithm 2 with finite iteration (i.e. finite sampling time). The grid sampling to find anchor control point is illustrated in Figure 10.

□

Lemma 5 (Feasibility)**.** *If we enters the phase 2 of Algorithm 2 with an anchor safe control being sampled, we can always find a local optimal solution of* (4)*.*

Proof. Entering the phase 2 of Algorithm 2, we first apply grid sampling to find a safe control u^a as an anchor point. We then employ AdamBA to outreach from u^r following the vector direction from u^r to u^a (line 12). If the boundary point u^* is not found (u^* is empty), we then employ AdamBA again (line 14) to outreach from u^a following the vector direction from u^a to u^r. We will prove phase 2 of Algorithm 2 is guaranteed to find a safe control by proving at least one of these two AdamBA processes is able to return a valid boundary point (i.e, if line 12 fails to find u^*, then line

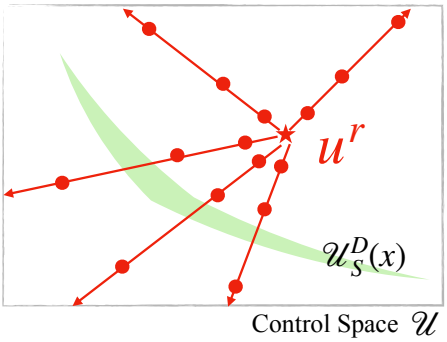


Figure 11: Illustration of the case where there is no return from phase 1. The red dots represent the sampled control points.

14 is guaranteed to find a boundary point). Intuitively, as long as we can proof that by choosing learning rate as $\frac{\|u^r-u^a\|}{4}$, at least one of these two AdamBA processes can enter *exponential decay* phase, then we can guarantee a boundary solution will be returned by lemma 3.

Specifically, when the system has a non-trivial set of safe control, by Lemma 4, we can always have one anchor safe control u^a satisfying the constraint of (4) after entering phase 2. Note that the condition we enter phase 2 is that there's no return from phase 1, which is illustrated in fig. 11.

Denote U^a as the connected set of safe control u^a belongs to, i.e., $u^a \in U^a \subseteq \mathcal{U}_S^D$. Denote direction of gradient vector from u^r to u^a as $\vec{v}_a = \frac{u^a-u^r}{||u^a-u^r||}$. And denote $U^a_{\vec{v}_a}$ to be the set of safe control points that are both within set of safe control U^a, and is along the $\vec{v}_a$ direction, i.e. $U^a_{\vec{v}_a} = \{u^a_{\vec{v}_a} | u^a_{\vec{v}_a} \in U^a, \frac{u^a_{\vec{v}_a}-u^r}{\|u^a_{\vec{v}_a}-u^r\|} = \vec{v}_a\}$. We further denote $u^n \in U^a_{\vec{v}_a}$ and $u^f \in U^a_{\vec{v}_a}$ as the Euclidean nearest and farthest control point with respect to u^r, i.e., $u^n = \arg\min_{u \in U^a_{\vec{v}_a}} \|u-u^r\|$, and $u^f = \arg\max_{u \in U^a_{\vec{v}_a}} \|u-u^r\|$. The illustrations of u^n and u^f are demonstrated in fig. 12.

Now consider the phase 2 of Algorithm 2. We employ AdamBA to outreach from u^r following the vector direction from u^r to u^a. Denote the boundary point returned by AdamBA in line 12 of Algorithm 2 as u^* (The return of AdamBA is a set, whereas the set here has at most one element). If we can find u^*, then u^* is on the boundary of the set of safe control, which indicates that phase 2 of Algorithm 2 can find a local optima of (4).

If we cannot find u^* (u^* is empty) in line 12 of Algorithm 2, as illustrated in Figure 12, then all the searched control points (red dots in Figure 12) along $\vec{v}_a$ (red arrow direction in Figure 12) is not within $U^a_{\vec{v}_a}$. Therefore, we have:

$$\begin{aligned} &\exists n \in \mathbb{N}, (2^n-1)||\vec{v}_a|| \in (0, ||\vec{\rho}||) \\ &\text{s.t. } \forall m \in \mathbb{N}, m > n, (2^m-1)||\vec{v}_a|| \notin (||\vec{\rho}||, ||\vec{\rho}+\vec{\psi}||) \,, \end{aligned} \tag{30}$$

where $\vec{\rho} = u^n - u^r$ and $\vec{\psi} = u^f - u^n$.

Consider the contrapositive of (30):

$$\begin{aligned} &\forall n \in \mathbb{N}, (2^n-1)||\vec{v}_a|| \in (0, ||\vec{\rho}||) \\ &\text{s.t. } \exists m \in \mathbb{N}, m > n, (2^m-1)||\vec{v}_a|| \in (||\vec{\rho}||, ||\vec{\rho}+\vec{\psi}||) \,, \end{aligned} \tag{31}$$

where (31) holds if $||\vec{\rho}|| \leq \frac{1}{2}||\vec{\psi}||$ ($\|\vec{\rho}\|$ and $\|\vec{\psi}\|$ are also illustrated in fig. 12). The proof is as follows:

$$\begin{aligned} &\forall n \in \mathbb{N}, (2^n-1)||\vec{v}_a|| \in (0, ||\vec{\rho}||), \max((2^{n+1}-1)||\vec{v}_a||) < 3||\vec{\rho}|| < ||\vec{\rho}+\vec{\psi}|| \\ &\text{i.e. } \exists m \in \mathbb{N}, m > n, (2^m-1)||\vec{v}_a|| \in (||\vec{\rho}||, ||\vec{\rho}+\vec{\psi}||) \,. \end{aligned} \tag{32}$$

Therefore, we can derive from (30) that:

$$||\vec{\rho}|| > \frac{1}{2}||\vec{\psi}|| \,, \tag{33}$$

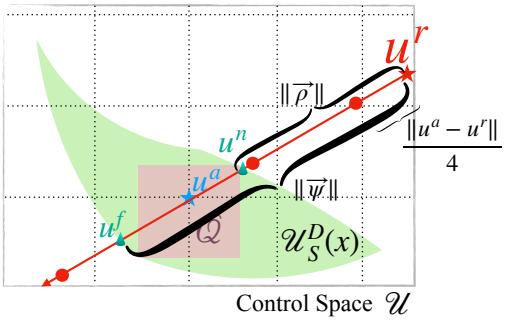


Figure 12: Illustration of the case when it is unable to find u^*.

where (33) holds if (30) holds.

Following the remaining process of the phase 2 of Algorithm 2, we start outreach process from u^a along the gradient vector $\frac{u^r-u^a}{\|u^r-u^a\|}$. Denote the n-th search control point as u_n^s. The Euclidean distance between u_n^s and u^a is equal to $\beta(2^n-1)$, where β is the initial learning rate (step size) of AdamBA.

Therefore, we have:

$$\exists n \in \mathbb{N}, \beta \in (0, ||(u^r - u^a)||), (2^n - 1)\beta \in (\frac{2}{3}||(u^r - u^a)||, ||(u^r - u^a)||) \,, \tag{34}$$

which is equivalent to:

$$\exists n \in \mathbb{N}, \beta \in (0, ||(u^r - u^a)||), [1 - (2^n - 1)]\beta \in (0, \frac{1}{3}||\vec{u^r} - \vec{u^a}||) \,. \tag{35}$$

Since $||u^r - u^a|| < ||\vec{\psi_i} + \vec{\rho_i}||$, we have:

$$\exists n \in \mathbb{N}, \beta \in (0, ||(u^r - u^a)||), [1 - (2^n - 1)]\beta \in (0, \frac{1}{3}||\vec{\psi} + \vec{\rho}||) \,. \tag{36}$$

Then we take (33) into (36):

$$\exists n \in \mathbb{N}, \beta \in (0, ||(u^r - u^a)||), [1 - (2^n - 1)]\beta \in (0, ||\vec{\rho}||) \,. \tag{37}$$

which means:

$$\exists n \in \mathbb{N}, ||u_n^s - u^r|| < ||\vec{\rho_i}|| \,. \tag{38}$$

Note that $||u_n^s - u^r|| < ||\vec{\rho_i}||$ means $u_n^s \notin U_{\vec{v}_a}^a$ (i.e., u_n^s is an unsafe control) and AdamBA will turn to decay phase to find boundary points.

In Algorithm 2, by take $\beta = \frac{1}{4}||(u^r - u^a)||$, we have:

$$u_1^s = u_a + \frac{1}{4}(u^r - u^a) \tag{39}$$

Note that u_1^s is always in the set of unsafe control. This is because we did not find a safe action in line 12 of Algorithm 2, where u_1^s has been sampled during the exponential outreach phase of AdamBA called by line 12 of Algorithm 2.

Therefore, we have proved that in the phase 2 of Algorithm 2, we can either directly find a boundary point u^*, or we can start from u^a and are able to find a control whose safety status is different from the safety status of u^a. By Lemma 3, an approximated safe control on the set of safe control boundary can always be found, which is also a local optima of (4). □

D.2 Proof of Proposition 2

Proof. We have proved Algorithm 2 is able to find local optimal solution of (4) by Lemma 3 and Lemma 5. We now prove Algorithm 2 can be finished within finite iterations.

If phase 1 of Algorithm 2 returns the safe control, then we find a local optima solution of (4) with finite iterations. The upper bound of the iterations T^1 of simulating black-box dynamics of phase 1 is the addition of maximum iterations from *exponential outreach* and *exponential decay*. Denote $\xi = \max_{j,k} \|u^j - u^k\|$, where $u^j \in \mathcal{U}_S^D, u^k \in \mathcal{U}_S^D$. The maximum iteration $T^1_{outreach}$ at *exponential outreach* satisfies:

$$T^1_{outreach} \leq n \log_2(\frac{\xi}{\beta} + 1) \tag{40}$$

where n is the number of gradient vectors. According to lemma 3, the exit condition for *exponential decay* should satisfy that the learning rate $< \epsilon$. Therefore, the maximum iteration T^1_{decay} at *exponential decay* satisfies:

$$T^1_{outreach} \leq n \log_2(\frac{\xi}{\beta} + 1) - n \log_2(\frac{\epsilon}{\beta}) \tag{41}$$

Therefore the maximum iteration T^1 for phase 1 satisfies:

$$T^1 \leq n * (\underbrace{\lceil n \log_2(\frac{\xi}{\beta} + 1) \rceil}_{\text{exponential outreach of phase 1}} + \underbrace{\lceil n \log_2(\frac{\xi}{\beta} + 1) - n \log_2(\frac{\epsilon}{\beta}) \rceil}_{\text{exponential decay of phase 1}}) , \tag{42}$$

If phase 1 of Algorithm 2 has no return, then we enter the phase 2 of Algorithm 2. If the safety index design can guarantee a non-trivial set of safe control, by Lemma 4 we can always find an anchor safe control with finite iterations T^a, which is bounded by:

$$T^a \leq \prod_{i=1}^{n_u} \lceil \frac{\xi_{[i]}}{s^*} \rceil , \tag{43}$$

If we can find an anchor safe control, by Lemma 5, we can always find a local optima solution of (4) by running Algorithm 1 at most twice (line 12 and line 14), where the iteration times T^2 of the phase 2 is bounded by:

$$T^2 \leq T^a + 2 * (n * (\underbrace{\lceil n \log_2(\frac{\xi^*}{\beta^*} + 1) \rceil}_{\text{exponential outreach of phase 1}} + \underbrace{\lceil n \log_2(\frac{\xi^*}{\beta^*} + 1) - n \log_2(\frac{\epsilon}{\beta^*}) \rceil}_{\text{exponential decay of phase 1}})) . \tag{44}$$

where $\xi^* = ||u^r - u^a||$ and $\beta^* = \frac{1}{4}||u^r - u^a||$. Then, we can give the upper bound of iteration times T^{ISSA} of Algorithm 2 as $T^{ISSA} = T^1 + T^2$.

In summary, Algorithm 2 can always find a local optima solution of (4) with finite iterations if the safety index design can guarantee a non-empty set of safe control, which proves Proposition 2. □

E Proof of Theorem 1

Here we define set $\mathcal{X}_S^D := \{x|\phi(x) \leq 0\}$. Before we proof the main theorem, we first present a preliminary result regarding $\mathcal{X}_S^D$ that is useful towards proving the main theorem:

Lemma 6 (Forward Invariance of $\mathcal{X}_S^D$)**.** *If the control system satisfies Assumption 1 and the safety index design follows the rule described Section 4.1, the implicit safe set algorithm guarantees the forward invariance to the set $\mathcal{X}_S^D$.*

Proof. If the control system satisfies the assumptions in Assumption 1 and the safety index design follows the rule described Section 4.1, then we can ensure the system has nonempty set of safe control at any state by Proposition 1. By Proposition 2, implicit safe set algorithm can always find local optima solution of (4). The local optima solution always satisfies the constraint $\phi(f(x_t, u_t)) \leq \max\{\phi(x_t) - \eta, 0\}$, which indicates that 1) if $\phi(x_{t_0}) \leq 0$, then $\phi(x_t) \leq 0, \forall t \geq t_0$. Note that $\phi(x) \leq 0$ demonstrates that $x \in \mathcal{X}_S^D$. □

E.1 Proof the Theorem 1

Proof. Leveraging Lemma 6, we then proceed to prove that the *forward invariance* to the set $\mathcal{X}_S^D$ guarantees the *forward invariance* to the set $S \subseteq \mathcal{X}_S$. Recall that $\mathcal{S} = \mathcal{X}_S \cap \mathcal{X}_S^D$. Depending on the relationship between $\mathcal{X}_S^D$ and $\mathcal{X}_S$, there are two cases in the proof which we will discuss below.

Case 1: $\mathcal{X}_S^D = \{x|\phi(x) \leq 0\}$ is a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

In this case, $\mathcal{S} = \mathcal{X}_S^D$. According to Lemma 6, If the control system satisfies the assumptions in Assumption 1 and the safety index design follows the rule described Section 4.1, the implicit safe set algorithm guarantees the forward invariance to the set $\mathcal{X}_S^D$ and hence $\mathcal{S}$.

Case 2: $\mathcal{X}_S^D = \{x|\phi(x) \leq 0\}$ is NOT a subset of $\mathcal{X}_S = \{x|\phi_0(x) \leq 0\}$.

In this case, if $x_t \in \mathcal{S}$, we have $\phi_0(x_t) = \max_i \phi_{0_i}(x_t) \leq 0$, which indicates $\forall i, \phi_{0_i} \leq 0$.

Firstly, we consider the case where $\phi_{0_i}(x_t) < 0$. Note that $\phi_{0_i}(x_{t+1}) = \phi_{0_i}(x_t) + \dot{\phi}_{0_i}(x_t)dt + \frac{\ddot{\phi}_{0_i}(x_t)dt^2}{2!} + \cdots$, since the state space and control space are both bounded, and $dt \to 0$ according to Assumption 1, we have $\phi_{0_i}(x_{t+1}) \to \phi_{0_i}(x_t) \leq 0$.

Secondly, we consider the case where $\phi_{0_i}(x_t) = 0$. Since $x_t \in \mathcal{S}$, we have $\max_i \phi_i(x_t) \leq 0$, which indicates $\forall i, \sigma + d_{min}^n - d_i^n - k\dot{d}_i \leq 0$. Since $\phi_{0_i}(x_t) = 0$, we also have $d_i = d_{min}$. Therefore, the following condition holds:

$$\begin{aligned} \sigma - k\dot{d}_i &\leq 0 \\ \dot{d}_i &\geq \frac{\sigma}{k} \end{aligned} \tag{45}$$

According to the safety index design rule, we have $k, \sigma \in \mathbb{R}^+$, which indicates $\dot{d}_i > 0$. Therefore, we have $\phi_{0_i}(x_{t+1}) < 0$.

Summarizing the above two cases, we have shown that if $\phi_{0_i}(x_t) \leq 0$ then $\phi_{0_i}(x_{t+1}) \leq 0$, which indicates if $\forall i, \phi_{0_i}(x_t) \leq 0$ then $\forall i, \phi_{0_i}(x_{t+1}) \leq 0$. Note that $\forall i, \phi_{0_i}(x_{t+1}) \leq 0$ indicates that $\phi_0(x_{t+1}) = \max_i \phi_{0_i}(x_{t+1}) \leq 0$. Therefore, we have that if $x_t \in \mathcal{S}$ then $x_{t+1} \in \mathcal{X}_S$. Thus, we also have $x_{t+1} \in \mathcal{S}$ by Lemma 6. By induction, we have if $x_{t_0} \in \mathcal{S}$, $x_t \in \mathcal{S}, \forall t > t_0$.

In summary, by discussing the two cases of whether $\mathcal{X}_S^D$ is the subset of $\mathcal{X}_S$, we have proven that if the control system satisfies the assumptions in Assumption 1 and the safety index design follows the rule described in Section 4.1, the implicit safe set algorithm guarantees the forward invariance to the set $\mathcal{S} \subseteq \mathcal{X}_S$.

□

F Expeiment Details

F.1 Safety Gym Experiment Details

F.1.1 Environment Settings

Goal Task In the Goal task environments, the reward function is:

$$r(x_t) = d^g_{t-1} - d^g_t + \mathbb{1}[d^g_t < R^g] \,,$$

where d^g_t is the distance from the robot to its closest goal and R^g is the size (radius) of the goal. When a goal is achieved, the goal location is randomly reset to someplace new while keeping the rest of the layout the same.

Push Task In the Push task environments, the reward function is:

$$r(x_t) = d^r_{t-1} - d^r_t + d^b_{t-1} - d^b_t + \mathbb{1}[d^b_t < R^g] \,,$$

where d^r and d^b are the distance from the robot to its closest goal and the distance from the box to its closest goal, and R^g is the size (radius) of the goal. The box size is 0.2 for all the Push task environments. Like the goal task, a new goal location is drawn each time a goal is achieved.

Hazard Constraint In the Hazard constraint environments, the cost function is:

$$c(x_t) = \max(0, R^h - d^h_t) \,,$$

where d^h_t is the distance to the closest hazard and R^h is the size (radius) of the hazard.

Pillar Constraint In the Pillar constraint environments, the cost $c_t = 1$ if the robot contacts with the pillar otherwise $c_t = 0$.

Black-box Dynamics The underlying dynamics of Safety Gym is directly handled by MuJoCo physics simulator [20]. This indicates the dynamics is not explicitly accessible but rather can be implicitly evaluated, which is suitable for our proposed implicit safe set algorithm. The implementation of block-box dynamics for ISSA is through simulation in the MuJoCo physics simulator and recovering to the pre-simulated state.

State Space The state space is composed of various physical quantities from standard robot sensors (accelerometer, gyroscope, magnetometer, and velocimeter) and lidar (where each lidar sensor perceives objects of a single kind). The state spaces of all the test suites are summarized in Table 2. Note that Vase is another type of constraint in Safety Gym [14] and all the returns of vase lidar are zero vectors (i.e., $[0, 0, \cdots, 0] \in \mathbb{R}^{16}$) in our experiments since none of our eight test suites environments have vases.

Control Space For all the experiments, the control space $\mathcal{U} \subset \mathbb{R}^2$. The first dimension $u_1 \in [-10, 10]$ is the control space of moving actuator, and second dimension $u_2 \in [-10, 10]$ is the control space of turning actuator.

F.1.2 Policy Settings

Detailed parameter settings are shown in Table 3. All the policies in our experiments use the default hyper-parameter settings hand-tuned by Safety Gym [14] except the cost limit $= 0$ for PPO-Lagrangian and CPO.

F.1.3 Safety Index Settings

The parameters of safety index design are summarized in Table 4, where we adopt $k = 0.375$ for test suites with constraint size of 0.05 and $k = 0.5$ for test suites with constraint size of 0.15.

Table 2: The state space components of different test suites environments.

State Space Option	Goal-Hazard	Goal-Pillar	Push-Hazard
Accelerometer ($\mathbb{R}^3$)	✓	✓	✓
Gyroscope ($\mathbb{R}^3$)	✓	✓	✓
Magnetometer ($\mathbb{R}^3$)	✓	✓	✓
Velocimeter ($\mathbb{R}^3$)	✓	✓	✓
Goal Lidar ($\mathbb{R}^{16}$)	✓	✓	✓
Hazard Lidar ($\mathbb{R}^{16}$)	✓	✗	✓
Pillar Lidar ($\mathbb{R}^{16}$)	✗	✓	✗
Vase Lidar ($\mathbb{R}^{16}$)	✓	✓	✓
Box Lidar ($\mathbb{R}^{16}$)	✗	✗	✓

Table 3: Important hyper-parameters of PPO, PPO-Lagrangian, CPO, PPO-SL and PPO-ISSA

Policy Parameter	PPO	PPO-Lagrangian	CPO	PPO-SL & PPO-ISSA
Timesteps per iteration	30000	30000	30000	30000
Policy network hidden layers	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Value network hidden layers	(256, 256)	(256, 256)	(256, 256)	(256, 256)
Policy learning rate	0.0004	0.0004	(N/A)	0.0004
Value learning rate	0.001	0.001	0.001	0.001
Target KL	0.01	0.01	0.01	0.01
Discounted factor γ	0.99	0.99	0.99	0.99
Advantage discounted factor λ	0.97	0.97	0.97	0.97
PPO Clipping ϵ	0.2	0.2	(N/A)	0.2
TRPO Conjugate gradient damping	(N/A)	(N/A)	0.1	(N/A)
TRPO Backtracking steps	(N/A)	(N/A)	10	(N/A)
Cost limit	(N/A)	0	0	(N/A)

F.1.4 Metrics Comparison

In this section, we report all the results of eight test suites by three metrics defined in Safety Gym [14]:

- The average episode return J_r.
- The average episodic sum of costs M_c.
- The average cost over the entirety of training ρ_c.

The average episode return J_r and the average episodic sum of costs M_c were obtained by averaging over the last five epochs of training to reduce noise. Cost rate ρ_c was just taken from the final epoch. We report the results of these three metrics in Table 5 normalized by PPO results. We calculate the converged reward $\bar{J}_r$ percentage of PPO-ISSA compared to other three safe RL baseline methods (PPO-Lagrangian, CPO and PPO-SL) over eight control suites. The computed mean reward percentage is 95%, and the standard deviation is 9%. Therefore we conclude that PPO-ISSA is able to gain $95\% \pm 9\%$ cumulative reward compared to state-of-the-art safe DRL methods.

Note that in safety critical environments, there is always a tradeoff between reward performance and safety, where safety guarantees prevent aggressive strategies for seeking high reward. On the other hand, the provable safety is prominently weighted in the tradeoff, since any safety violation may lead to property loss, life danger in the real robotics applications. Therefore, PPO-ISSA does a better job in terms of balancing the tradeoff between provable safety and task performance.

Table 4: Experiment-specific parameters of safety index design for PPO-ISSA.

Safety Index Parameter	Constraint size = 0.05	Constraint size = 0.15
n	1	1
k	0.375	0.5
η	0	0

Table 5: Normalized metrics obtained from the policies at the end of the training process, which is averaged over eight test suits environments and five random seeds.

(a) Goal-Hazard1-0.05

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.00	1.00	1.00
PPO-Lagrangian	1.003	1.587	0.859
CPO	1.012	1.052	0.944
PPO-SL [18' Dalal]	1.038	1.031	1.110
PPO-ISSA (Ours)	**1.077**	**0.000**	**0.000**

(b) Goal-Hazard4-0.05

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.983	0.702	0.797
CPO	**1.022**	0.549	0.676
PPO-SL [18' Dalal]	1.014	0.923	0.963
PPO-ISSA (Ours)	0.961	**0.000**	**0.000**

(c) Goal-Hazard1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	**1.086**	0.338	0.760
CPO	1.011	0.553	0.398
PPO-SL [18' Dalal]	1.018	0.898	1.048
PPO-ISSA (Ours)	1.008	**0.000**	**0.000**

(d) Goal-Hazard4-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.948	0.581	0.645
CPO	0.932	0.328	0.303
PPO-SL [18' Dalal]	**1.038**	0.948	1.063
PPO-ISSA (Ours)	0.895	**0.000**	**0.000**

(e) Goal-Pillar1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	0.968	0.196	0.239
CPO	0.976	0.328	0.494
PPO-SL [18' Dalal]	1.017	0.948	1.063
PPO-ISSA (Ours)	**1.056**	**0.000**	**0.000**

(f) Goal-Pillar4-0.15

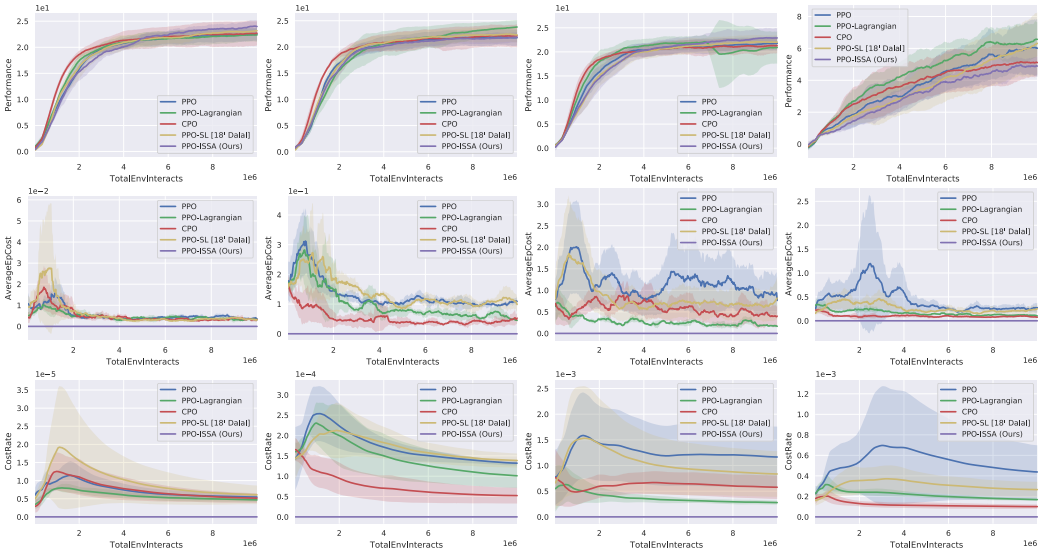
Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	1.035	0.105	0.159
CPO	1.060	0.304	0.221
PPO-SL [18' Dalal]	**1.094**	1.055	0.780
PPO-ISSA (Ours)	0.965	**0.000**	**0.000**

(g) Push-Hazard1-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	1.000	1.000	1.000
PPO-Lagrangian	**1.124**	0.356	0.384
CPO	0.872	0.231	0.228
PPO-SL [18' Dalal]	1.107	0.685	0.610
PPO-ISSA (Ours)	0.841	**0.000**	**0.000**

(h) Push-Hazard4-0.15

Algorithm	$\bar{J}_r$	$\bar{M}_c$	$\bar{\rho}_c$
PPO	**1.000**	1.000	1.000
PPO-Lagrangian	0.72	0.631	0.748
CPO	0.758	0.328	0.385
PPO-SL [18' Dalal]	0.914	1.084	1.212
PPO-ISSA (Ours)	0.727	**0.000**	**0.000**



(a) Goal-Hazard1-0.05 (b) Goal-Hazard1-0.15 (c) Goal-Pillar1-0.15 (d) Push-Hazard1-0.15

Figure 13: Average performance of PPO-ISSA and baseline methods on 1-constraint environments over five random seeds. The three rows represent average episodic return, average episodic cost and overall cost rate of constraints. The safety advantage of safe RL baseline methods over unconstrained RL method (PPO) becomes trivial as the constraint number and constraint size decrease, where the cost rate and average episode cost of PPO-Lagrangian, CPO and PPO are nearly the same when there is only one constraint with size 0.05.

G Scalability Analysis

The safe control algorithm ISSA is based on the sampling method algorithm 1 AdamBA. In this section, we demonstrate the scalability of AdamBA and ISSA in systems with higher dimensions of state and control.

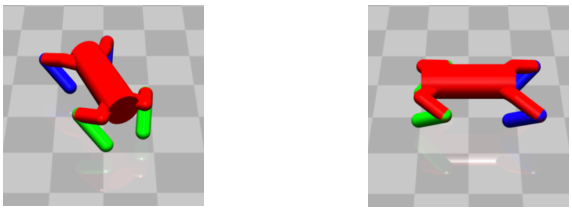


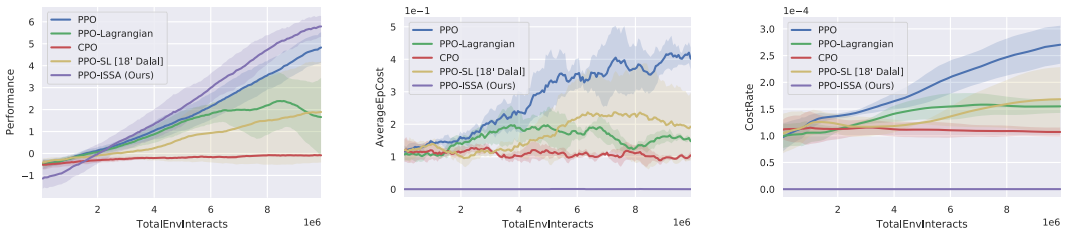
Figure 14: doggo robot: a quadrupedal robot with bilateral symmetry with 12-dimensional control space.

G.1 ISSA in higher dimensional constrol systems

We test ISSA with a doggo robot as shown in Figure 14 with 12 dimensional control space and 80 dimensional state space. We evaluate ISSA with the doggo robot in the Goal-Hazard1-0.15 suite. As shown in Figure 15, ISSA is able to guarantee zero-violation in higher dimensional control systems. We notice that in system with higher control dimensions, safe RL methods like PPO-Lagrangian and CPO perform poorly compared to PPO, which demonstrates the constrained RL algorithms struggle to learn good reward performance for complex locomotion behavior. Similar comparison results are also reported in Safety Gym benchmarks [14]. In contrast, PPO-ISSA is able to achieve the best reward performance while guaranteeing zero safety violation, showing the scalability of ISSA to achieve satisfying reward performance in systems with higher control dimensions.

To illustrate the computation cost of searching for a safe control using ISSA in systems with higher control dimensions, we apply ISSA to find safe controls for both Point robot and doggo robot in the Goal-Hazard1-0.15 suite with different number of unit gradient vectors generated by guassian distribution. Note that we desire to encourage ISSA phase 1 to find safe control, while the functionality of ISSA phase 2 is the fail-safe strategy for ISSA phase 1 to ensure feasibility of ISSA since 1) ISSA phase 2 is relatively more computational expensive than ISSA phase 1 due to random sampling, and 2) ISSA phase 2 can only return one safe control candidate. Thus, we are especially interested in analyzing the performance of ISSA phase 1.

We report the average computation time, ISSA phase 1 success rate and number of safe control candidates found by ISSA phase 1 under different robot types and number of unit gradient vectorss in Table 6, where the success of ISSA phase 1 is defined as it returns at least 1 safe control which may lead to a large deviation from the original nominal control. As demonstrated in Table 6, we only need 20 unit gradient vectors for 12 dimensional control space doggo robot to achieve satisfying success rate in ISSA phase 1 and number of safe control candidates. On the other hand, we need 10 unit gradient vectors for 2 dimensional control space Point robot. Therefore, higher control dimensionality will not necessarily increase the computation cost exponentially for ISSA to find safe control. We also notice that, even with the same number of vectors, the computaion time of doggo



(a) Average episode reward (b) Average episode cost (c) Average episode costrate

Figure 15: Average performance of PPO-ISSA and baseline methods on Goal-hazard1-0.15 environment of doggo robot over five seeds. The three columns represent average episode return, average episode cost and overallcost rate of constraints.

Table 6: Average Computation time, succuss ratio of ISSA phase 1 and number of safe control candidates of 200 ISSA runs on Goal-Hazard1-0.15.

	Number of vectors	Overall (non-parallel) ISSA Time T_{all} (s)	ISSA phase 1 success rate	Number of safe control
Point robot Control dimension = 2	$n = 3$	0.023	0.962	1.193
	$n = 5$	0.038	1.000	2.258
	$n = 10$ **(used)**	0.076	1.000	4.576
	$n = 20$	0.160	1.000	9.838
	$n = 40$	0.302	1.000	18.055
	$n = 100$	0.737	1.000	25.740
Doggo robot Control dimension = 12	$n = 3$	0.068	0.802	2.041
	$n = 5$	0.116	0.903	3.051
	$n = 10$	0.228	0.925	5.127
	$n = 20$ **(used)**	0.461	0.981	10.673
	$n = 40$	0.910	0.990	19.373
	$n = 100$	2.190	1.000	33.910

robot is 3 times longer than that of Point robot due to the doggo robot simulation takes longer than point robot simulaton per step in MuJoCo simulator. Here we also highlight a fact that the process of AdamBA outreach/decay for each unit gradient vector is independent from each other, thus we can always accelerate ISSA by parallel computation, which is then discussed in Appendix G.2.

G.2 AdamBA in higher dimensional space

As reported in Appendix G.1, in MuJoCo environment, the sampling cost of AdamBA is not exponentially increasing as the control dimensions increase, where we only need 10 vectors for point robot (2-dimensional control space) and only need 20 vectors for doggo robot (12-dimensional control space). However, let's consider a worse case for AdamBA, where the relative size of safe control space in the entire control space is exponentially decreasing with dimension linearly increases, which could result in the computation cost of AdamBA to find the boundary exponentially increasing. We synthesis a simple control problem, where only half of the control space is safe for each dimension, which means the portion of safe control space to the entire control space is $\frac{1}{2^n}$ where n is the number of control space dimensions. Note that for every unit gradient vector, the process of AdamBA outraech/decay is independent of the other unit gradient vectors, which means we could utilize parallel computation in practice when applying real-time robots. We report the average computation time of AdamBA on the prescribed toy control problem in Figure 16, where we can see that the computation cost of parallel AdamBA remains nearly the same as the number of vectors exponentially increases, which shows the potential capability of AdamBA scaling to higher dimensional real-time control systems.

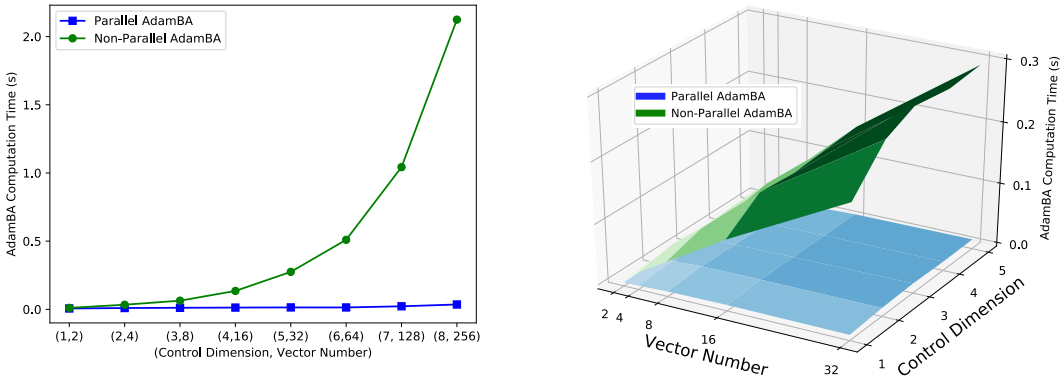


Figure 16: The average computation time of non-parallel AdamBA and parallel AdamBA on toy problem with different control dimensions and number of vectors.