Proceedings of Machine Learning Research vol XX:1–29, 2023

Provably Safe Model-Free Control with Low Latency

Ritabrata Ray RITABRAR@ANDREW.CMU.EDU
B42, Porter Hall, Carnegie Mellon University
Pittsburgh, PA 15213

Yorie Nakahira YNAKAHIR@ANDREW.CMU.EDU
B20, Porter Hall, Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

In this paper, we propose a low-latency safe control algorithm that guarantees system safety at all times. This algorithm applies to any non-linear control-affine system and is robust against uncertainties in the system dynamics model. It can also be adapted with reinforcement learning algorithms like REINFORCE to guarantee safe exploration at all times for learning in model-free settings. Further, we extend our methods to develop another low-latency algorithm that applies to the same settings and enjoys a provable guarantee for quick recovery from an initial unsafe state to safe behaviour.

Keywords: Safe control, Forward invariance/convergence, Model-free reinforcement learning.

1. Introduction

Safety is of critical importance for many intelligent control systems. Although many safe control algorithms are developed for known systems, it is challenging to ensure safety for systems with unknown dynamics. The system dynamics may have large uncertainty when learning the system model and control actions in real time, or the dynamics could be too complex to accurately model. In this paper we present an algorithm that assumes very little knowlede of the system dynamics and can be merged with existing model-based and model-free reinforcement learning algorithms to provably guarantee safe behavior at all times.

Related Work. Several methods exist for safe control of non-linear systems with unknown dynamics, which can be broadly classified into model-free and model-driven approaches. Model-based techniques are usually preferred when the system dynamics can be modelled properly and the uncertainties stem from unknown parameters in the system dynamics model. Common model-based techniques include safe adaptive control with Lyapunov methods: Taylor and Ames (2020); Lopez et al. (2021); Fan et al. (2020); Choi et al. (2021); Nguyen and Sreenath (2020); Dean et al. (2020); Cosner et al. (2021); system identification for safe control techniques: Ho et al. (2021); Grover et al. (2021); Wigren et al. (2022); Jagtap et al. (2020); Nelles (2001), regret-based online-learning algorithms: Luo et al. (2022); Kakade et al. (2020); Gaussian process methods for model-based safe reinforcement learning: Berkenkamp et al. (2017); Ma et al. (2022); Cowen-Rivers et al. (2020); Fan and Li (2019); Polymenakos et al. (2017); Sui et al. (2015); Turchetta et al. (2016). The model-based techniques usually require restricted classes of functions for system dynamics, and small uncertainties (bounded or Gaussian) in the unknown parameters. Such techniques lead to safe behavior only after the algorithm has learned the unknown parameters to a sufficient degree of accuracy and suffer from high latency issues.

On the other hand, model-free reinforcement learning algorithms are used when the system dynamics cannot be accurately modelled, and the source of uncertainties are unknown. Such techniques directly learn the safe control action from the observed roll-out trajectories and gradually learn a policy which commits lesser mistakes (unsafe behavior) as the training progresses. Such techniques either penalize the reward functions of the MDP: Zhang et al. (2022); Dong et al. (2020), or use trust region methods for constrained reinforcement learning: Achiam et al. (2017); Bharadhwaj et al. (2021); Bisi et al. (2020); Han et al. (2020); Vuong et al. (2019); Yang et al. (2020), or use Lyapunov techniques along with reinforcement learning: Qin et al. (2022); Zhao et al. (2021); Choi et al. (2020); Chow et al. (2018, 2019); Dong et al. (2020); Huh and Yang (2020); Jeddi et al. (2021); Kumar and Sharma (2017); Perkins and Barto (2003). Gu et al. (2022); García et al. (2015); Dawson et al. (2022) survey some of the existing techniques for safe control.

Challenges. The existing methods for safe control have the following limitations:

1. Model-based adaptive control techniques assume a certain structure to the dynamics model and assume bounded uncertainties in the unknown parameters. Such methods can lead to unsafe behavior near the starting time until the algorithm learns the parameters to a sufficient degree of accuracy.
2. Model-based Reinforcement learning algorithms using Gaussian processes also assume Gaussian uncertainties and similar to adaptive control methods, may lead to unsafe behavior during the initial phases of exploration.
3. Model-free reinforcement learning algorithms suffer from large sample complexity and it takes longer for the agent to learn the safe control. Trust region based constrained optimization methods only limit the safety violations up to a constant upper bound and hence cannot avoid safety violations at all times.
4. Lyapunov based techniques (could be combined with Reinforcement Learning algorithms) also require restrictions on the initial state to guarantee forward invariance for safety. These techniques usually incur high latency.

Our contibutions. We present a safe control algorithm which provably guarantees safe behavior at all times (zero violations). Our *Sign-Flipping* algorithm is designed for systems with dynamics which are affine in the control variable. It only requires the knowledge of the signs (with a non-zero confidence margin) along which the control variables affect the system state dynamics. It has the following merits:

1. The proposed method assumes no knowledge of the system dynamics other than the signs along which the control variables affect the system dynamics. This addresses the challenges:1,2 of bounded/Gaussian uncertainty as well as restricted system dynamics model. Figures 1, 2 in 8 and theorem 2 in 7 demonstrate this.
2. The proposed method does not require to perform any computationally heavy task to compute the safe control action. This addresses the latency challenge 4 and makes it suitable for real-time applications.
3. The proposed method ensures zero-safety violations at all times, thus addressing challenge 3 associated with standard constrained policy optimization based deep reinforcement learning algorithms. Figures 2, 4 and theorem 2 illustrate this merit.

4. The above merits allow our method to be used as a low-latency tool that can be used on top of model-free reinforcement learning algorithms and allow for provable zero-violation guarantees during the exploration and lead to learning of safe control policies. Figures 3 and 4 in section 8 demonstrate this merit.
5. Similarly, if a nominal controller (satisfying performance objectives) is known for model-based adaptive control/reinforcement learning settings then our method can be used to create a low-latency tool that sits on top of the control algorithm and modifies the control action to guarantee zero-violations when the system state comes close to the unsafe region. Figures 1, 2 and theorem 2 support this merit.
6. Our technique can be easily extended to get a low-latency recovery tool which works in the same settings as above and can quickly recover the system from an initial unsafe state. Figures 5, 6 and theorem 3 show this merit.

Notation. Throughout the paper we adopt the following notations:

1. For any function $f : \mathbb{R}^d \to \mathbb{R}$, ∇f denotes its gradient with respect to x, i.e. $\nabla_x f(x)$. Let $x_t : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ denote any dynamical system as a function of time t, then $\dot{f}$ denotes the time derivative of the function $f(x_t)$, $\dot{f}^+$ denotes its right hand time derivative, and $\dot{f}^-$ denotes its left hand time derivative: i.e.,
$$\dot{f}^+ = \lim_{\delta t \to 0^+} \frac{f(x_{t+\delta t}) - f(x_t)}{\delta t} \tag{1}$$
$$\dot{f}^- = \lim_{\delta t \to 0^+} \frac{f(x_t) - f(x_{t-\delta t})}{\delta t}. \tag{2}$$
2. For any two vectors u, v of the same dimensions, $u \cdot v$ denotes their standard inner product in Euclidean space.

2. Preliminaries

We consider a continuous time system whose state x_t is continuous in time t and it's dynamics is assumed to take the following form:

$$\dot{x}_t^+ = f(x_t) + g(x_t)u_t, \tag{3}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$, and $g : \mathbb{R}^d \to \mathbb{R}^{d\times p}$ are continuous functions, $u \in \mathbb{R}^p$ is the control action. We assume that the state history $\{x_\tau : \tau \leq t\}$ is fully observable by the controller at time t. We assume that $g(x_t)$ has full row-rank for any x_t and consequently its right Moore-Penrose pseudo-inverse $g^+(x_t) \in \mathbb{R}^{p\times d}$ exists and satisfies:

$$g(x_t)g^+(x_t) = I, \tag{4}$$

where I is the $d \times d$ identity matrix. A system is considered *safe* when the state x_t lies within a certain region $\mathcal{S} \subset \mathbb{R}^d$ called the safe set at all times t.

Definition 1 (Forward Invariant Set) *A set $\mathcal{S}$ is said to be forward invariant with respect to a dynamical system x_t, if starting at a point $x_0 \in \mathcal{S}$, we have $x_t \in \mathcal{S}$, $\forall t$.*

Therefore, the system is safe if the safe set $\mathcal{S}$ is forward invariant with respect to the closed loop dynamics. We characterize the safe set $\mathcal{S}$ by the level set of a continuously differentiable *barrier function* $\phi : \mathbb{R}^d \to \mathbb{R}$ as follows:

$$\mathcal{S} = \{x : \phi(x) \geq 0,\ x \in \mathbb{R}^d\}. \tag{5}$$

Given a threshold parameter $\theta > 0$, we define the safety subset:

$$\mathcal{S}_\theta = \left\{x : \phi(x) \geq \theta\right\} \subset \mathcal{S}, \tag{6}$$

to be the super-level set of the barrier function ϕ, and its boundary to be

$$\partial\mathcal{S}_\theta = \left\{x : \phi(x) = \theta\right\}. \tag{7}$$

In certian cases, there may exist a *nominal controller*:

$$u_t = u_{nom}(x_t), \tag{8}$$

where $u_{nom} : \mathbb{R}^d \rightarrow \mathbb{R}^p$ is a continuous mapping from the state space to the control action space, which is designed to achieve the operational/performance objectives of the system but it does not necessarily guarantee safety.

3. Problem Statement

For a non-linear system with control dynamics given by (3), the objective is to design a controller that guarantees safety at all times with minimum operational interruptions. We assume that the user has no knowledge of the function $f(.)$, and very little knowledge of the function $g(.)$. We develop the algorithm in two stages, first when the user knows the function $g(.)$ (see Appendix of the full paper[1]) and then the general case with only partial structural information about $g(x_t)$ through an Oracle call. See the appendix of the full paper for some justifications behind the Oracle assumptions.

4. The function $g(.)$ is unknown with Oracle access

Here, we present our proposed algorithm that can work for systems with dynamics described by (3) with a completely unknown $f(x)$ and the user has access to only partial information about the function $g(x)$. This partial information comes in two forms: existing offline information about the structure of the matrix $g(x_t)$ and online information about some parameters of the matrix $g(x_t)$ which comes from an Oracle call.

Partial Information Setting

We assume the following regarding the partial information about $g(x_t)$ that is available to the user at any time instant t:

Assumption I: Offline Structural Assumption.
The rows of the matrix $g(x_t)$ (denoted by $v_{1,t}, v_{2,t}, \ldots, v_{d,t}$) are orthogonal and non-zero vectors, for every x_t. We have:

$$v_{i,t} \cdot v_{j,t} = 0,\ v_{i,t} \neq 0\ \forall i \neq j, i, j \in \{1, \ldots, d\}. \tag{9}$$

Consequences of this assumption: The singular value decomposition of $g(x_t)$ can be written as:

$$g(x_t) = I\Sigma_t V_t^\intercal. \tag{10}$$

[1] Full-Version

Here I is the $d \times d$ identity matrix, $\Sigma \in \mathbb{R}^{d\times p}$ with

$$\Sigma_{t,(i,j)} = \begin{cases} \lambda_{i,t}, \text{ if } i = j, i \in \{1, \dots, d\} \\ 0, \text{ otherwise} \end{cases} \tag{11}$$

and

$$V_t = \Big[u_1, u_2, \dots, u_d\Big] \tag{12}$$

is an orthonormal matrix satisfying:

$$V_t V_t^{\mathsf{T}} = I, \tag{13}$$

where the columns of V_t are the normalized row vectors of $g(x_t)$, i.e.

$$u_i = \frac{1}{\left\|v_{i,t}\right\|} v_{i,t} \;\; \forall i \in \{1, \dots, d\}.$$

Since the rows of V_t are non-zero, the singular values $\lambda_{i,t} \neq 0, \; \forall t, i \in \{1, \dots, d\}$.

Assumption II: Online Oracle Assumption.
There exists an oracle $\mathcal{O} : \mathbb{R}^d \to \mathbb{R}^{p\times d} \times \mathbb{R}^d$, which takes the state variable x_t as input and immediately returns the output:

$$\mathcal{O}(x_t) = \Big(V_t, (\hat{\lambda}_{1,t}, \dots, \hat{\lambda}_{d,t})\Big). \tag{14}$$

Here, the matrix V_t is the orthonormal matrix corresponding to the rows of $g(x_t)$ given by (12), and the numbers $\hat{\lambda}_{i,t}$ are coarse bounds representing the unknown singular values $\lambda_{i,t}$ with the correct sign, i.e. $\forall t, i \in \{1, \dots, d\}$, we have:

$$\hat{\lambda}_{i,t} \in \begin{cases} (0, \lambda_{i,t}] \text{ if } \lambda_{i,t} > 0, \\ [\lambda_{i,t}, 0) \text{ if } \lambda i, t < 0. \end{cases} \tag{15}$$

Consequences of this assumption: The Oracle's output given by (14), allows the algorithm to construct the following representative singular value matrix $\hat{\Sigma}_t \in \mathbb{R}^{d\times p}$, given by:

$$\hat{\Sigma}_{t,(i,j)} = \begin{cases} \hat{\lambda}_{i,t}, & \text{if } i = j, i \in \{1, \dots, d\} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

and computes its right pseudo-inverse as the matrix $\hat{\Sigma}_t^+$ ($\in \mathbb{R}^{p\times d}$) given by:

$$\hat{\Sigma}^+_{t,(i,j)} = \begin{cases} \frac{1}{\hat{\lambda}_{i,t}}, & \text{if } i = j, i \in \{1, \dots, d\} \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

The algorithm uses these to computes the representative matrix $\hat{g}(x_t)$ of the original matrix $g(x_t)$ as:

$$\hat{g}(x_t) = I\hat{\Sigma}_t V_t^{\mathsf{T}}. \tag{18}$$

Algorithm Description. The proposed algorithm is summarized in algorithm 1. This algorithm performs exactly in the same way as algorithm 4, except for the case when the state variable is at the boundary of the safety sub-region $\mathcal{S}_\theta$ and is pointing outwards of the safe set i.e., $\phi(x_t) = \theta$, and $\dot{\phi}^-(x_t) < 0$. Here it calls the oracle with the state variable x_t as input and from its output computes the representative matrix $\hat{g}(x_t)$ of the matrix $g(x_t)$ as described by (18). The algorithm uses this representative matrix $\hat{g}(x_t)$ given by (18), to play a role similar to that of the known $g(x_t)$ in algorithm 4. It then computes the right pseudo-inverse $\hat{g}^+(x_t)$ of this representative matrix as:

$$\hat{g}^+(x_t) = V_t \hat{\Sigma}_t^+ I, \tag{19}$$

where I is the $d \times d$ identity matrix, $\hat{\Sigma}_t^+$ is the $p \times d$ matrix given by (17), and V_t is the $p \times p$ matrix given by (12). The algorithm then computes the correction strength multiplier matrix $\gamma_t(x_t) \in \mathbb{R}^{d \times d}$ as follows:

$$\gamma_{t,(i,j)}(x_t) = \begin{cases} 1, \text{ if } i = j, \text{ and } \nabla\phi(x_t)_i \dot{x}_{t,i}^- < 0, i \in \{1, \ldots, d\} \\ 0, \text{ otherwise.} \end{cases} \tag{20}$$

Using the above correction strength multiplier matrix, the algorithm computes its correction control u_{corr} as:

$$u_{corr} = u_{last} - 2\hat{g}^+(x_t)\gamma_t(x_t)\dot{x}_t^-. \tag{21}$$

Algorithm 1: Sign-Flipping Algorithm with Oracle

Input: Control Barrier Function: $\phi(.)$, Threshold: $\theta > 0$, Initial value of the state variable: $x_0 \in \mathcal{S}$ such that $\phi(x_0) > \theta$, the Oracle $\mathcal{O}$ as defined in (14), and the nominal controller: $u_{nom}(.)$.
Hyperparameters: Discretization time step: T_s.
Initialize: $x_{t=0} \leftarrow x_0,\ u_{last} \leftarrow u_{nom}(x_0),\ x_{t=0}^- \leftarrow x_0$.

```
for every $t > 0$, do
    Receive the current state $x_t$ from observation.
    Compute $\phi(x_t)$.
    Compute the time derivative of state variable: $\dot{x}_t^- \leftarrow \frac{1}{T_s}(x_t - x_t^-)$.
    Compute: $\dot{\phi}^-(x_t) \leftarrow \nabla\phi(x_t) \cdot \dot{x}_t^-$.
    switch ()
    case $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) \geq 0$:
        $u \leftarrow u_{last}$.
    case $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) < 0$:
        Call the Oracle with input $x_t$, and receive: $\mathcal{O}(x_t) = \left(V_t, (\hat{\lambda}_{1,t}, \ldots, \hat{\lambda}_{d,t})\right)$.
        Compute the representative matrix: $\hat{g}(x_t)$ using (18).
        Compute the right Pseudo-inverse: $\hat{g}^+(x_t)$ of the matrix $\hat{g}(x_t)$ using (19).
        Compute: $u_{corr}$, using (21) as $u \leftarrow u_{corr}$.
    default:
        $u \leftarrow u_{nom}(x_t)$.
    end switch
    Play $u$.
    $u_{last} \leftarrow u$.
    Store last value of $x_t$ as $x_t^- \leftarrow x_t$.
end for
```

5. Safe exploration for model-free reinforcement learning algorithms

In this section, we illustrate merit 4 by presenting our technique as a low-latency tool which can be merged on top of the model-free reinforcement learning algorithm REINFORCE Williams (1992). We discretize the state and action spaces to finite spaces $\mathcal{S}$ and $\mathcal{A}$ respectively. At each time step, the agent receives a reward $r(s,a)$ for the current state and action pair (s,a) where the reward function $r:\mathcal{S}\times\mathcal{A}\to\mathbb{R}$ is stationary with respect to time and is designed according to the environment and the desired task. The goal is to maximize the expected discounted aggregate sum of rewards, i.e., $J=\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]$, where $\gamma\in[0,1)$ is the discounting factor. Let $C(x,u,u_{last})$ denote the correction controller given by algorithm 1 where the safety threshold θ is chosen at $\theta=0$. That is,

$$C(x,\dot{x}^-,u,u_{last})=\begin{cases}u & \text{if } \phi(x)>0\\ u_{last} & \text{if } \phi(x)\le 0 \text{ and } \dot{\phi}^-(x)\ge 0\\ u_{corr}\text{ given by (21)} & otherwise\end{cases} \tag{22}$$

We show in the Appendix of the full paper that the policy gradient estimated by the algorithm 2 is indeed an unbiased estimate of the true policy gradient. During the entire process, safety is guaranteed by theorem 2, where the action sampled from the stochastic policy could be seen as the nominal controller action. This allows for safe exploration while learning a safe controller in the model-free setting.

6. Recovery from unsafe initial state

In this section, we illustrate merit 6 of our technique by extending algorithms 4, 1 with added guarantee of recovery to the safe region, when started at an unsafe point $x_0\notin\mathcal{S}$. Here, we use the recovery control as:

$$u_{rec}=u_{last}-\hat{g}^+(x_t)\Big(\gamma_t(x_t)\dot{x}_t^-+n_d(x_t)\nabla\phi(x_t)\Big), \tag{23}$$

where $n_d(x)$ is given by

$$n_d(x)=-\frac{\eta}{\left\|\nabla\phi(x)\right\|^2}. \tag{24}$$

7. Performance Guarantees

In this section, we state the main results of this paper: theorems stating the performance guarantees of the algorithms in the above sections. The proofs of the following results can be found in the Appendix of the full version of the paper. The first result guarantees safety at all times for our proposed method 1.

Theorem 2 (Forward Invariance for algorithm 1) *If $x_0\in\mathcal{S}_\theta$, then the set $\mathcal{S}_\theta$ is forward invariant with respect to the controller given by algorithm 1.*

Next we have the guarantee for quick recovery from an unsafe initial state for algorithm 3.

Theorem 3 (Forward Persistence for algorithm 3) *If $x_0\notin\mathcal{S}_\theta$, then the set $\mathcal{S}_\theta$ is forward convergent (i.e., $\exists\tau>0$ such that $x_\tau\in\mathcal{S}_\theta$) when the controller given by algorithm 3 is used and the rate of convergence is lower bounded by the hyperparameter η. Further, once the system state enters $\mathcal{S}_\theta$, the set $\mathcal{S}_\theta$ is forward invariant (i.e., $x_t\in\mathcal{S}_\theta\ \forall t\ge\tau$.) for the system when algorithm 5 is used.*

Algorithm 2: REINFORCE with Safety

Input: Finite State and Action spaces $\mathcal{S}$, $\mathcal{A}$ respectively, Terminal state $T \in \mathcal{S}$, Control Barrier Function $\phi(.)$, the correction controller $C(x, u, u_{last})$ as defined in (22), Initial value of the state variable: $s_0 \in \mathcal{S}$ such that $\phi(s_0) > 0$, the Oracle $\mathcal{O}$ as defined in (14), a deep neural network with parameters $\theta \in \Theta$.

Hyperparameters: Discounting Factor γ, discretization time step: T_s.

Initialize: $x_{t=0} \leftarrow x_0,\ u_{last} \leftarrow u_{nom}(x_0),\ x^-_{t=0} \leftarrow x_0, \theta \leftarrow \theta_0$.

```
for every episode $\tau = 0, 1, 2, \ldots$ do
    Roll-out Episode $\tau$.
    Set $t \leftarrow 0$, $G_{-1} \leftarrow 0$.
    while $x_t \neq T$ do
        Receive the current state $x_t$ from observation, obtain the corresponding discrete state $s_t$.
        Compute the time derivative of state variable: $\dot{x}_t^- \leftarrow \frac{1}{T_s}(x_t - x_t^-)$.
        Sample $a_t \sim \pi_{\theta_\tau}(.|s_t)$.
        Overwrite the sampled control action $a_t$ as: $u_t = C(s_t, \dot{x}_t^-, a_t, u_{last})$.
        Play $u_t$ and collect reward $r_t \leftarrow r(s_t, u_t)$.
        $u_{last} \leftarrow u_t$, $x_t^- \leftarrow x_t$, $t \leftarrow t + 1$.
    end while
    For the above episode $\tau$: Let $(s_0, a_0, u_0, s_1, a_1, u_1, s_2, \ldots)$ be the roll-out trajectory collected.
    Compute the variables $G_t$ as: $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$.
    Estimate the policy gradient sample as:
        $$\nabla J(\theta_\tau) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta=\theta_\tau} G_t.$$
    Update the policy network parameters as: $\theta_{\tau+1} \leftarrow \theta_\tau + \nabla J(\theta_\tau)$.
end for
```

8. Numerical Study

In this section, we experimentally demonstrate the merits 3, 4, 6, 1 of our method through numerical simulations [2]. First, we present a system in which our algorithm (denoted here as sign flip) guarantees safety at all times and compare it against the performance of the following adaptive safe control algorithms on the same system and with the same initialization: aCBF from Taylor and Ames (2020), RaCBF and RaCBF+SMID (here denoted as RaCBFS) from Lopez et al. (2021), $\mathcal{A}_\pi-$SEL algorithm from Ho et al. (2021) (denoted here as cbc), and the BALSA algorithm from Fan et al. (2020) (denoted here as balsa). The implementation details of each algorithm could be found in the appendix. We obtain the plots 1, 2.

We observe that all other algorithms except Sign flip take a non-zero time to ultimately converge to the safe region.

Next, we demonstrate algorithm 2 and compare it against some model-free reinforcement learning algorithms like REINFORCE Williams (1992), CPO Achiam et al. (2017). We obtain the following

[2] Simulation Codes

```
  Algorithm 3: Safety and Recovery with Oracle
Input: Control Barrier Function: $\phi(.)$, threshold: $\theta > 0$, initial value of the state variable: $x_0$, the Oracle $\mathcal{O}$ as defined in (14), and the nominal controller: $u_{nom}(.)$.
Hyperparameters: Discretization time step: $T_s$, recovery hyperparameter $\eta > 0$.
Initialize: $x_{t=0} \leftarrow x_0,\ u_{last} \leftarrow u_{nom}(x_0),\ x^-_{t=0} \leftarrow x_0$.
  for every $t > 0$, do
    Receive the current state $x_t$ from observation and compute $\phi(x_t)$.
    Compute the time derivative of state variable: $\dot{x}^-_t \leftarrow \frac{1}{T_s}(x_t - x^-_t)$.
    Compute: $\dot{\phi}^-(x_t) \leftarrow \nabla\phi(x_t) \cdot \dot{x}^-_t$.
    switch ()
    case 1. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) \geq 0$:
      Assign $u \leftarrow u_{last}$.
    case 2. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) < 0$:
      Call the Oracle with input $x_t$, and receive: $\mathcal{O}(x_t) = \Big(V_t, (\hat{\lambda}_{1,t}, \dots, \hat{\lambda}_{d,t})\Big)$.
      Compute the representative matrix: $\hat{g}(x_t)$ using (18).
      Compute the right Pseudo-inverse: $\hat{g}^+(x_t)$ of the matrix $\hat{g}(x_t)$ using (19).
      Compute: $u_{corr}$, using (21) as $u \leftarrow u_{corr}$.
    case 3. $\phi(x_t) < \theta$:
      Call the Oracle with input $x_t$, and receive: $\mathcal{O}(x_t) = \Big(V_t, (\hat{\lambda}_{1,t}, \dots, \hat{\lambda}_{d,t})\Big)$.
      Compute the representative matrix: $\hat{g}(x_t)$ using (18).
      Compute the right Pseudo-inverse: $\hat{g}^+(x_t)$ of the matrix $\hat{g}(x_t)$ using (19).
      Compute the matrix $\gamma_t(x_t)$ using (20).
      Compute: $n_d(x)$ using (24) and $u_{rec}$ using (23), assign $u \leftarrow u_{rec}$.
    default:
      $u \leftarrow u_{nom}(x_t)$.
    end switch
    Play $u$, $u_{last} \leftarrow u$, $x^-_t \leftarrow x_t$.
  end for
```

plots: 3, and 4. It shows that our algorithm stays safe at all times and also converges to learn a safe route from source to destination (longer) without much loss in the convergence rate.

Lastly, we demonstrate the quick recovery ability (forward convergence and persistence) of algorithm 3 by starting the system at five distinct initial positions, one of which is safe and the other four unsafe. From the plots: 5, and 6, we observe that the system quickly recovers to the safe region (forward convergence; recovery time depends on initial state), and the system remains safe afterwords (forward persistence). The details of the experimental setup can be found in the Appendix of the full version of the paper.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*,

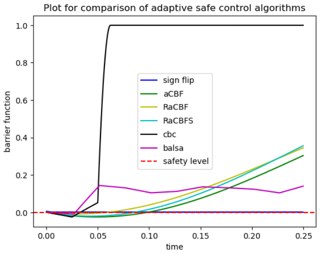

Figure 1: The barrier function $\phi(x)$ with respect to time for several safe adaptive control algorithms vs our algorithm Sign flip

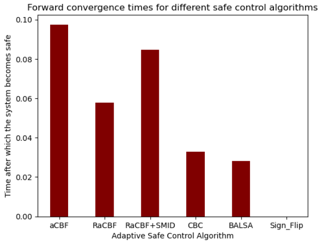

Figure 2: Comparison of Forward Convergence times of several adaptive safe control algorithms vs our algorithm Sign flip.

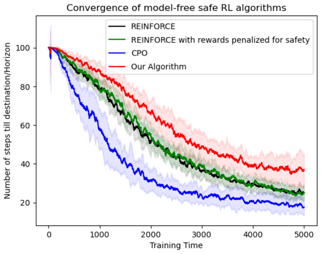

Figure 3: The average number of steps needed to reach the destination with respect to the training epoch.

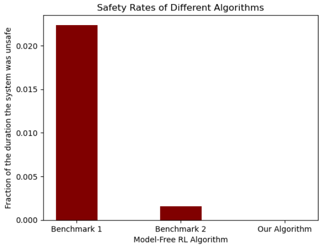

Figure 4: Fraction of time the system was in unsafe states for different algorithms. Benchmark 1 is REINFORCE with rewards adjusted for safety, Benchmark 2 is CPO.

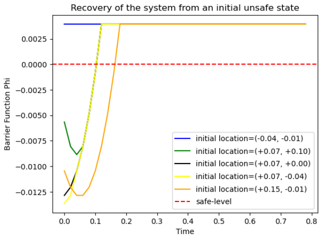

Figure 5: The average number of steps needed to reach the destination with respect to the training epoch.

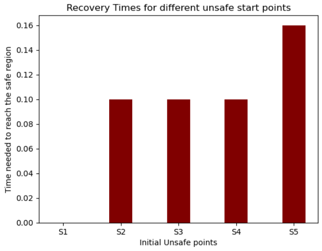

Figure 6: Fraction of time the system was in unsafe states for different algorithms. Benchmark 1 is REINFORCE with rewards adjusted for safety, Benchmark 2 is CPO.

pages 22–31. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees, 2017. URL https://arxiv.org/abs/1705.08551.

Homanga Bharadhwaj, Aviral Kumar, Nicholas Rhinehart, Sergey Levine, Florian Shkurti, and Animesh Garg. Conservative safety critics for exploration. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=iaO86DUuKi.

Lorenzo Bisi, Luca Sabbioni, Edoardo Vittori, Matteo Papini, and Marcello Restelli. Risk-averse trust region optimization for reward-volatility reduction. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4583–4589. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/632. URL https://doi.org/10.24963/ijcai.2020/632. Special Track on AI in FinTech.

Yuen Ren Chao. A note on "Continuous mathematical induction.". *Bulletin of the American Mathematical Society*, 26(1):17 – 18, 1919. doi: bams/1183425067. URL https://doi.org/.

Jason Choi, Fernando Castañeda, Claire J. Tomlin, and Koushil Sreenath. Reinforcement learning for safety-critical control under model uncertainty, using control lyapunov functions and control barrier functions, 2020. URL https://arxiv.org/abs/2004.07584.

Jason J. Choi, Donggun Lee, Koushil Sreenath, Claire J. Tomlin, and Sylvia L. Herbert. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 6814–6821, 2021. doi: 10.1109/CDC45484.2021.9683085.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/4fe5149039b52765bde64beb9f674940-Paper.pdf.

Yinlam Chow, Ofir Nachum, Aleksandra Faust, Mohammad Ghavamzadeh, and Edgar A. Duéñez-Guzmán. Lyapunov-based safe policy optimization for continuous control. *CoRR*, abs/1901.10031, 2019. URL http://arxiv.org/abs/1901.10031.

Pete L. Clark. The instructor's guide to real induction, 2012. URL https://arxiv.org/abs/1208.0973.

Ryan K. Cosner, Andrew W. Singletary, Andrew J. Taylor, Tamas G. Molnar, Katherine L. Bouman, and Aaron D. Ames. Measurement-robust control barrier functions: Certainty in safety with uncertainty in state, 2021. URL https://arxiv.org/abs/2104.14030.

Alexander Imani Cowen-Rivers, Daniel Palenicek, Vincent Moens, Mohammed Amin Abdullah, Aivar Sootla, Jun Wang, and Haitham Ammar. SAMBA: safe model-based & active reinforcement learning. *CoRR*, abs/2006.09436, 2020. URL https://arxiv.org/abs/2006.09436.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods. *arXiv preprint arXiv:2202.11762*, 2022.

Sarah Dean, Andrew J. Taylor, Ryan K. Cosner, Benjamin Recht, and Aaron D. Ames. Guaranteeing safety of learned perception modules via measurement-robust control barrier functions, 2020. URL https://arxiv.org/abs/2010.16001.

Yunlong Dong, Xiuchuan Tang, and Ye Yuan. Principled reward shaping for reinforcement learning via lyapunov stability theory. *Neurocomputing*, 393:83–90, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2020.02.008. URL https://www.sciencedirect.com/science/article/pii/S0925231220301831.

David D. Fan, Jennifer Nguyen, Rohan Thakker, Nikhilesh Alatur, Ali-akbar Agha-mohammadi, and Evangelos A. Theodorou. Bayesian learning-based adaptive control for safety critical systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4093–4099, 2020. doi: 10.1109/ICRA40945.2020.9196709.

Jiameng Fan and Wenchao Li. Safety-guided deep reinforcement learning via online gaussian process estimation. *CoRR*, abs/1903.02526, 2019. URL http://arxiv.org/abs/1903.02526.

Javier García, Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL http://jmlr.org/papers/v16/garcia15a.html.

Jaskaran Grover, Changliu Liu, and Katia Sycara. System identification for safe controllers using inverse optimization. *IFAC-PapersOnLine*, 54(20):346–353, 2021. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2021.11.198. URL https://www.sciencedirect.com/science/article/pii/S2405896321022424. Modeling, Estimation and Control Conference MECC 2021.

Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, Yaodong Yang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications, 2022. URL https://arxiv.org/abs/2205.10330.

Minghao Han, Yuan Tian, Lixian Zhang, Jun Wang, and Wei Pan. Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee, 2020. URL https://arxiv.org/abs/2011.06882.

Dimitar Ho, Hoang M. Le, John C. Doyle, and Yisong Yue. Online robust control of nonlinear systems with large uncertainty. 2021. doi: 10.48550/ARXIV.2103.11055. URL https://arxiv.org/abs/2103.11055.

Subin Huh and Insoon Yang. Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach. *CoRR*, abs/2002.10126, 2020. URL https://arxiv.org/abs/2002.10126.

Pushpak Jagtap, George J. Pappas, and Majid Zamani. Control Barrier Functions for Unknown Nonlinear Systems using Gaussian Processes. 2020. URL http://arxiv.org/abs/2010.05818.

Ashkan B. Jeddi, Nariman L. Dehghani, and Abdollah Shafieezadeh. Lyapunov-based uncertainty-aware safe reinforcement learning. *CoRR*, abs/2107.13944, 2021. URL https://arxiv.org/abs/2107.13944.

S. Kakade, A. Krishnamurthy, K. Lowrey, M. Ohnishi, and W. Sun. Information theoretic regret bounds for online nonlinear control. In *Advances in Neural Information Processing Systems*, 2020.

Abhishek Kumar and Rajneesh Sharma. Fuzzy lyapunov reinforcement learning for non linear systems. *ISA Transactions*, 67, 01 2017. doi: 10.1016/j.isatra.2017.01.026.

Brett T. Lopez, Jean-Jacques E. Slotine, and Jonathan P. How. Robust adaptive control barrier functions: An adaptive and data-driven approach to safety. *IEEE Control Systems Letters*, 5(3): 1031–1036, 2021. doi: 10.1109/LCSYS.2020.3005923.

Wenhao Luo, Wen Sun, and Ashish Kapoor. Sample-efficient safe learning for online nonlinear control with control barrier functions, 2022. URL https://arxiv.org/abs/2207.14419.

Yecheng Jason Ma, Andrew Shen, Osbert Bastani, and Jayaraman Dinesh. Conservative and adaptive penalty for model-based safe reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5404–5412, Jun. 2022. doi: 10.1609/aaai.v36i5.20478. URL https://ojs.aaai.org/index.php/AAAI/article/view/20478.

Mitio Nagumo. Über die lage der integralkurven gewöhnlicher differentialgleichungen. 1942.

Oliver Nelles. *Nonlinear system identification. From classical approaches to neural networks and fuzzy models*. 01 2001. ISBN 978-3-642-08674-8. doi: 10.1007/978-3-662-04323-3.

Quan Nguyen and Koushil Sreenath. Robust safety-critical control for dynamic robotics, 2020. URL https://arxiv.org/abs/2005.07284.

Theodore Perkins and Andrew Barto. Lyapunov design for safe reinforcement learning control. *J. Mach. Learn. Res.*, 3:803–, 05 2003. doi: 10.1162/jmlr.2003.3.4-5.803.

Kyriakos Polymenakos, Alessandro Abate, and Stephen Roberts. Safe policy search with gaussian process models, 2017. URL https://arxiv.org/abs/1712.05556.

Zengyi Qin, Dawei Sun, and Chuchu Fan. Sablas: Learning safe control for black-box dynamical systems, 2022. URL https://arxiv.org/abs/2201.01918.

Yanan Sui, Alkis Gotovos, Joel Burdick, and Andreas Krause. Safe exploration for optimization with gaussian processes. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 997–1005, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/sui15.html.

Andrew J. Taylor and Aaron D. Ames. Adaptive safety with control barrier functions. In *2020 American Control Conference (ACC)*, pages 1399–1405, 2020. doi: 10.23919/ACC45564.2020.9147463.

Matteo Turchetta, Felix Berkenkamp, and Andreas Krause. Safe exploration in finite markov decision processes with gaussian processes. 2016. doi: 10.48550/ARXIV.1606.04753. URL https://arxiv.org/abs/1606.04753.

Quan Vuong, Yiming Zhang, and Keith W. Ross. SUPERVISED POLICY UPDATE. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJxTroR9F7.

Anna Wigren, Johan Wågberg, Fredrik Lindsten, Adrian G. Wills, and Thomas B. Schön. Nonlinear system identification: Learning while respecting physical models using a sequential monte carlo method. *IEEE Control Systems Magazine*, 42(1):75–102, 2022. doi: 10.1109/MCS.2021.3122269.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. Projection-based constrained policy optimization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rke3TJrtPS.

Linrui Zhang, Li Shen, Long Yang, Shixiang Chen, Bo Yuan, Xueqian Wang, and Dacheng Tao. Penalized proximal policy optimization for safe reinforcement learning, 2022. URL https://arxiv.org/abs/2205.11814.

Weiye Zhao, Tairan He, and Changliu Liu. Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=UGp6FDaxB0f.

Appendix

Appendix A. Algorithm when the function $g(.)$ is known.

The proposed algorithm is summarized in algorithm 4 and detailed below.

Algorithm Description. When the state is at a sufficient distance from the unsafe region i.e., $x_t \in \mathcal{S}_\theta \setminus \partial\mathcal{S}_\theta$, we want our system to use the nominal controller u_{nom} given by (8). This is the default case of the algorithm 4. When the system is about to get outside of $\mathcal{S}_\theta$ i.e., $x_t \in \partial\mathcal{S}_\theta$ and $\dot{\phi}^-(x_t) < 0$, (case 2 of the algorithm 4) then the algorithm switches to a correction controller $u_{corr}(x)$ given by:

$$u_{corr} = u_{last} - 2\gamma g^+(x_t)\dot{x}_t^-, \tag{25}$$

where $\gamma > \frac{1}{2}$ is the correction-strength hyperparameter, $g^+(x_t) \in \mathbb{R}^{p\times d}$ is the right pseudoinverse given by (4). Here, the degree of closeness to the boundary when the algorithm switches to the correction controller is decided by the input threshold: $\theta > 0$ (see the definition of $\mathcal{S}_\theta$ from (6)). However when the state is moving inwards of $\mathcal{S}_\theta$ at the boundary i.e., $x_t \in \partial\mathcal{S}_\theta$ and $\dot{\phi}^+(x_t) \geq 0$, then the algorithm intends to continue the motion (case 1 of algorithm 4) by playing the previously used action u_{last}. Once the state comes sufficiently inside the safe set $\mathcal{S}$ i.e., $x_t \in int(\mathcal{S}_\theta) = \mathcal{S}_\theta \setminus \partial\mathcal{S}_\theta$, the algorithm switches back to using the nominal controller u_{nom}. The correction control u_{corr} from (25) pushes the state variable back inside the safety sub-region $\mathcal{S}_\theta$. The rate of this

```
Algorithm 4: Sign-Flipping Algorithm for Safety
Input: Control Barrier Function $\phi(.)$, Threshold: $\theta > 0$, Initial value of the state variable: $x_0 \in \mathcal{S}$
such that $\phi(x_0) > \theta$, function $g(.)$, and the nominal controller: $u_{nom}(.)$.
Hyperparameters: Correction strength $\gamma > \frac{1}{2}$, discretization time step: $T_s$.
Initialize: $x_{t=0} \leftarrow x_0,\ u_{last} \leftarrow u_{nom}(x_0),\ x^-_{t=0} \leftarrow x_0$.
 for every $t > 0$, do
   Receive the current state $x_t$ from observation.
   Compute $\phi(x_t)$.
   Compute the time derivative of state variable: $\dot{x}^-_t \leftarrow \frac{1}{T_s}(x_t - x^-_t)$.
   Compute: $\dot{\phi}^-(x_t) \leftarrow \nabla\phi(x_t) \cdot \dot{x}^-_t$.
   switch ()
   case 1. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) \geq 0$:
     $u \leftarrow u_{last}$.
   case 2. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) < 0$:
     Compute the Pesudoinverse $g^+(x_t)$ of $g(x_t)$ given by (4).
     Compute: $u_{corr}$, according to (25).
     $u \leftarrow u_{corr}$.
   default:
     $u \leftarrow u_{nom}(x_t)$.
   end switch
   Play $u$.
   $u_{last} \leftarrow u$.
   Store last value of $x_t$ as $x^-_t \leftarrow x_t$.
 end for
```

recovery is governed by the magnitude of the correction-strength hyperparameter γ, and if $\gamma \leq \frac{1}{2}$, then this push back may not be sufficient. However, there is a limitation to using arbitrarily large values of γ. The magnitude of γ measures the discontinuity in the control signal and a very large γ would indicate huge jumps in the control action and possibly unrealistically large magnitudes of the control action. The practical limitations of this system would lend an upper bound on this γ. Therefore, γ is a hyperparameter that needs to be properly tuned. Algorithm 4 illustrates merits 2, 5 when the adaptive control algorithm dictates the nominal controller. Further merit 1 is conpletely illustrated by the algorithm 1 which works in a stronger setting.

For algorithm 4 we have the following performance guarantee of safety at all times:

Theorem 4 (Forward Invariance for algorithm 4) *If $x_0 \in \mathcal{S}_\theta$, then the set $\mathcal{S}_\theta$ is forward invariant when the controller given by algorithm 4 with $\gamma > \frac{1}{2}$ is used.*

We have a few preliminary lemmas, before starting the proof of theorem 4. We begin with the following lemma inspired by the well-known Nagumo's theorem Nagumo (1942).

Lemma 5 *Let $h : \mathbb{R}^d \to \mathbb{R}$ be a differetiable function, and let $x_t : \mathbb{R}_{\geq 0} \to \mathbb{R}^d$ denote the state-variable of a dynamical system which is continuous with respect to time t. Let $\mathcal{C} = \{x : h(x) \geq 0\}$, and let $\partial\mathcal{C} = \{x : h(x) = 0\}$ be the boundary of the set $\mathcal{C}$. Define $\dot{h}^+(x_t) = \nabla h(x_t) \cdot \dot{x}^+_t$, and let $T > 0$ be any finite time-horizon. If $x_0 \in \mathcal{C}$, $\dot{h}^+(x_t)$ exists $\forall t \in [0, T]$, and*

$$\dot{h}^+(x_t) \geq 0 \text{ whenever } x_t \in \partial\mathcal{C},$$

then $x_t \in \mathcal{C}\ \forall t \in [0, T]$.

We prove this lemma using mathematical induction on real numbers Clark (2012), Chao (1919).

Theorem 6 ***(Theorem 2 in Clark (2012))***
Let $a < b$ be real numbers. Let $A \subset [a, b]$ be a set satisfying the following three conditions:

(RI-1): $a \in A$.

(RI-2): For any $x \in [a, b)$, we have:

$$x \in A \implies \exists y > x \text{ such that } [x, y] \subset A.$$

(RI-3): For any $x \in (a, b]$, we have:

$$[a, x) \subset A \implies x \in A.$$

Then $A = [a, b]$.

Proof [Proof of Lemma 5] Setting $A = \left\{t : \phi(x_t) \in \mathcal{C}\right\} \subseteq [0, T]$ in theorem 6, the lemma follows if the following three conditions hold:

C-1: $0 \in A$.

C-2: For any $t \in (0, T]$:

$$[0, t) \subset A \implies t \in A.$$

C-3: For any $t \in [0, t)$:

$$t \in A \implies \exists \delta t > 0 \text{ such that } [t, t + \delta t] \subset A.$$

So, it suffices to prove each of the above conditions.

Derivation of of C-1: $x_0 \in \mathcal{C}$ implies $0 \in A$.

Derivation of C-2: From the lemma assumptions x_t is continuous with respect to t and $h(.)$ is differentiable with respect to x. So, we must have $h(x_t)$ continuous in t. So, if $[0, t) \subset A$, then $h(x_\tau) \geq 0$ for every $\tau \in [0, t)$. Because $h(x_t)$ is continuous in time t, we have:

$$h(x_t) = \lim_{\tau \to t^-} h(x_\tau) \geq 0,$$

implying $t \in A$, proving *C-2*.

Derivation of C-3: We show that *C-3* holds in the following four cases below:

Case I: $h(x_t) > 0$:
In this case, $h(x_t) > \epsilon$ for some $\epsilon > 0$. Using continuity of $h(x_t)$ with respect to t, there exists a $\delta t > 0$, (δt depending upon ϵ) such that $|h(x_\tau) - h(x_t)| \leq \epsilon$, for every $\tau \in [t, t + \delta t]$. So, we have $\forall \tau \in [t, t + \delta t]$:

$$\begin{aligned} h(x_\tau) &\geq h(x_t) - \epsilon \\ &\geq 0, \end{aligned}$$

which implies $[t, t + \delta t] \subset A$.

Case II: $h(x_t) = 0$ and $\dot{h}^+(x_t) > 0$:
In this case, we can write $\dot{h}^+(x_t) \geq \epsilon$ for some $\epsilon > 0$. From the existence of the right hand derivative $\dot{h}^+(x_t)$, there exists a $\delta t > 0$ such that:

$$\begin{aligned} &\left| \frac{h(x_\tau) - h(x_t)}{\tau - t} - \dot{h}^+(x_t) \right| \leq \epsilon \\ \implies &\frac{h(x_\tau) - h(x_t)}{\tau - t} \geq \dot{h}^+(x_t) - \epsilon \geq 0 \\ &\forall \tau \in [t, t + \delta t]. \end{aligned}$$

Thus, we have:

$$h(x_\tau) \geq h(x_t) = 0, \; \forall \tau \in [t, t + \delta t].$$

This implies $[t, t + \delta t] \subset A$ in this case as well.

Case III: $h(x_t) = 0$, $\dot{h}^+(x_t) = 0$ and there exists $\delta t > 0$ such that $\dot{h}^+(x_\tau) = 0$, $\forall \tau \in [t, t + \delta t]$.
The continuity of $h(x_t)$ with respect to t, and $\dot{h}^+(x_\tau) = 0$ for every $\tau \in [t, t+\delta t]$ collectively imply

$$h(x_\tau) = h(x_t) = 0 \; \forall \tau \in [t, t + \delta t].$$

Thus, we have $[t, t + \delta t] \subset A$, for this case too.

Case IV: $h(x_t) = 0$, $\dot{h}^+(x_t) = 0$ and there exists a $\delta t > 0$ such that $\dot{h}^+(x_\tau) > 0$, $\forall \tau \in (t, t + \delta t]$
Similar to case 2 above, we can prove:

$$\begin{aligned} &\dot{h}^+(x_\tau) > 0 \;\; \forall \tau \in (t, t + \delta t] \\ \implies &h(x_\tau) \geq h(x_t) = 0, \; \forall \tau \in (t, t + \delta t]. \end{aligned}$$

This again implies $[t, t + \delta t] \subset A$ for this case as well.

Thus, *C-3* holds in all the four cases above. From the lemma condition:

$$\dot{h}^+(x_t) \geq 0, \; \text{whenever } x_t \in \partial \mathcal{C},$$

we conclude that the above cases are mutually exhaustive, which establishes *C-3*.

And since all the three conditions *C-1, C-2, C-3* hold, from 16 6 we have $A = [0, T]$. So, we have $h(x_t) \geq 0$, for any $t \in A = [0, T]$, proving the lemma. ■

Before starting the proof of 15 4, we need another lemma which is a straightforward extension of the chain rule for differentiation to right hand hand derivatives.

Lemma 7 *If $\phi : \mathbb{R}^d \to \mathbb{R}$ is differentiable with respect to x $\forall x \in \mathbb{R}^d$, x_t is continuous with respect to time t, and $\dot{x}_t^+$ exists at time t, then*

$$\dot{\phi}^+(x_t) = \nabla \phi(x_t) \cdot \dot{x}_t^+. \tag{26}$$

Proof [Proof of Lemma 7] Since ϕ is differentiable with respect to x, we define $Q(y, x_t)$ for any $y \in \mathbb{R}^d$ as:

$$Q(y, x_t) = \begin{cases} \begin{pmatrix} \frac{\phi(y)_1 - \phi(x_t)_1}{y_1 - x_{t1}} \\ \vdots \\ \frac{\phi(y)_i - \phi(x_t)_i}{y_i - x_{ti}} \\ \vdots \\ \frac{\phi(y)_d - \phi(x_t)_d}{y_d - x_{td}} \end{pmatrix} & \text{if } y \neq x_t, \\ \nabla\phi(x_t) \text{ otherwise.} \end{cases}$$

We observe that $Q(y, x_t)$ is continuous with respect to y due to the differentiability of ϕ at x_t. And since x_τ is continuous with respect to τ, so $Q(x_\tau, x_t)$ is also continuous with respect to τ. Now let $\tau = t + \delta t$, for some $\delta t > 0$. Then, we have:

$$\begin{aligned} \dot{\phi}^+(x_t) &= \lim_{\delta t \to 0^+} \frac{\phi(x_{t+\delta t}) - \phi(x_t)}{\delta t} \\ &= \lim_{\tau \to t^+} Q(x_\tau, x_t) \cdot \frac{1}{\tau - t}(x_\tau - x_t) \\ &= \lim_{\tau \to t^+} Q(x_\tau, x_t) \cdot \lim_{\tau \to t^+} \frac{1}{\tau - t}(x_\tau - x_t) \\ &= Q(x_t, x_t) \cdot \dot{x}_t^+ \\ &= \nabla\phi(x_t) \cdot \dot{x}_t^+, \end{aligned}$$

where the first line uses (1), the second line follows from the definition of $Q(x_\tau, x_t)$, the third and fourth line use the continuity of $Q(x_\tau, x_t)$ with respect to τ, the existence of $\dot{x}_t^+$, and the fact that the limit of a product of two functions is the product of the limits of the two functions when the limits exist, and the last line again follows from the definition of $Q(y, x_t)$. ■

Similarly, we can also prove the following:
When $\dot{x}_t^-$ exists, we have:

$$\dot{\phi}^-(x_t) = \nabla\phi(x_t) \cdot \dot{x}_t^-. \tag{27}$$

We are now ready to prove theorem 4, using lemma 5 and lemma 7.

Proof [Proof of Theorem 4] Since x_t is assumed to be continuous with respect to time t, and by construction the barrier function $\phi(.)$ is a continuous function of the state x, $\phi(x_t)$ is continuous in time t.
Next, we show the following:

$$\dot{\phi}^+(x_t) \text{ exists } \forall t, \text{ and} \tag{28.1}$$

$$\dot{\phi}^+(x_t) \geq 0 \text{ whenever } x_t \in \partial\mathcal{S}_\theta. \tag{28.2}$$

From lemma 7 and (3), at each instant t, $\dot{\phi}^+(x_t)$ exists and is given by:

$$\dot{\phi}^+(x_t) = \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)u_t),$$

where u_t is the control variable at time t. This establishes (28.1). Next we establish (28.2) by considering the following three mutually exclusive and exhaustive cases in the algorithm:

Case I: $\phi(x_t) > \theta$ (Default case of algorithm 4):
Since $\phi(x_t) > \theta$, we have for this case $x_t \notin \partial\mathcal{S}_\theta$. So, (28.2) vacuously holds here.

Case II: $\phi(x_t) \leq \theta$ and $\dot{\phi}^-(x_t) \geq 0$:
From line 8 of algorithm 4, we have $u = u_{last}$ in this case, so u is left continuous at t. Notice that if the control variable u_t is left-continuous at time t, then the continuity of f, g with respect to x_t and the continuity of x_t with respect to t implies:

$$\dot{x}_t^- = f(x_t) + g(x_t)u_{last}. \tag{29}$$

Thus, from (29), we have:

$$\dot{x}_t^- = f(x_t) + g(x_t)u_{last} = f(x_t) + g(x_t)u = \dot{x}_t^+. \tag{30}$$

Thus, using (26) and (3), we have for this case:

$$\begin{align}
\dot{\phi}^+(x_t) &= \nabla\phi(x_t) \cdot \dot{x}_t^+ \tag{31}\\
&= \nabla\phi(x_t) \cdot \dot{x}_t^- \tag{32}\\
&= \dot{\phi}^-(x_t) \tag{33}\\
&\geq 0, \tag{34}
\end{align}$$

where the (32) follows from (30), (33) uses (27), and (34) follows from the fact that $\dot{\phi}^-(x_t) \geq 0$ in this case.

Case III: $\phi(x_t) \leq \theta$ and $\dot{\phi}^-(x_t) < 0$:
Here, from line 12 of algorithm 4, the algorithm u given by (25). We observe that u is left continuous at this time t, so (29) holds in this case too. So, using (26) and (3), we have:

$$\begin{align}
&\dot{\phi}^+(x_t) = \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)u_{corr}) \tag{35}\\
&= \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)(u_{last} - 2\gamma g^+(x_t)\dot{x}_t^-)) \tag{36}\\
&= \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)u_{last} - 2\gamma g(x_t)g^+(x_t)\dot{x}_t^-) \tag{37}\\
&= \nabla\phi(x_t) \cdot (\dot{x}_t^- - 2\gamma x_t^-) \tag{38}\\
&= -(2\gamma - 1)\nabla\phi(x_t) \cdot x_t^- \tag{39}\\
&= -(2\gamma - 1)\dot{\phi}^-(x_t) \tag{40}\\
&> 0, \tag{41}
\end{align}$$

where (36) uses (25); (38) uses (4) and (29); (40) uses (27); and (41) uses $\gamma > \frac{1}{2}$, and $\dot{\phi}^-(x_t) < 0$.

Since (28.2) holds in all the three cases above, and since the cases are mutually exhaustive, this establishes (28.2).

The continuity of $\phi(x_t)$ with respect to t, (28.1), (28.2), and lemma 5 with $h(x) = \phi(x) - \theta$, $\mathcal{C} = \mathcal{S}_\theta$ jointly imply $x_t \in \mathcal{S}_\theta$. This establishes the desired forward invariance and completes the proof. ■

Appendix B. Remarks on the Assumptions in section 4

The above two assumptions are quite natural and apply to many realistic control systems. The matrix $g(x_t)$ decides how the control variable, u affects the system dynamics at time t. The structural assumption considers the special case where the effect of the control variable, u on each coordinate of the state variable is ”orthogonal” to the effect on other variables. From, (3), we have for the control variable u_t at time t:

$$\dot{x}_{i,t} = f(x_t)_i + v_{i,t}^\intercal u_t,\ \forall i \in \{1, \dots, d\}, \tag{42}$$

i.e. rate of change of each state coordinate has an affine dependence on projection of the control vector u along its corresponding direction, $v_{i,t}$, which is orthogonal to the corresponding directions for other coordinates. It is also natural to assume that the directions in which the control variable u_t will affect its state variables is known. This knowledge is equivalent to the oracle assumption, which is knowing the unit vectors corresponding to each state variable (columns of the matrix V_t) and some bounds the singular values $\lambda_{i,t}$ through the singular value representatives $\hat{\lambda}_{i,t}$, where the representation errors $|\hat{\lambda}_{i,t} - \lambda_{i,t}|$ could be arbitrarily large, as long as the signs are represented correctly. For instance, the assumptions are satisfied by a kinematic dynamic system where the momentum added along each axis is a projection of the control signal along an orthogonal direction corresponding to the axis.

Appendix C. Proof of Theorem 2

Proof [Proof of Theorem 2] Similar to the proof of theorem 4, it suffices to prove the conditions *C1, C2,* and *C3* hold at all times. The proofs for the conditions *C1, C2* and and the first two cases in the proof of *C3* are identical to the ones in the proof of 15 4. So, it suffices to prove that the condition *C3* holds in case 3 i.e., if $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) < 0$, then it suffices to show that $\dot{\phi}(x_t)^+ > 0$ when algorithm 1 is used. The rest of the proof is devoted to showing this. Note that for this case, u is left-continuous at time t. So, (29) holds here.

We have for this case:

$$\begin{aligned}
\dot{\phi}(x_t)^+ &= \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)u_{corr}) && (43)\\
&= \nabla\phi(x_t) \cdot (f(x_t) + g(x_t)(u_{last}(x_t) - 2\hat{g}^+(x_t)\gamma_t(x_t)\dot{x}_t^-)) && (44)\\
&= \nabla\phi(x_t) \cdot (\dot{x}_t^- - 2g(x_t)\hat{g}^+(x_t)\gamma_t(x_t)\dot{x}_t^-) && (45)\\
&= \nabla\phi(x_t) \cdot (I - 2\Sigma_t\hat{\Sigma}_t^+\gamma_t(x_t))\dot{x}_t^- && (46)\\
&= \sum_{i=1}^{d} \nabla\phi(x_t)_i\dot{x}_{t,i}^- \left(1 - 2\gamma_t(x_t)_{i,i}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right), && (47)
\end{aligned}$$

where (43) uses (26) and (3); (44) uses (21); (45) uses (29); (46) uses (18), (19), (13); and (47) uses (11), (17).

Now, since in this case $\dot{\phi}^-(x_t) < 0$, using (27), we have:

$$\dot{\phi}^-(x_t) = \nabla\phi(x_t) \cdot \dot{x}_t^- = \sum_{i=1}^{d} \nabla\phi(x_t)_i\dot{x}_{t,i}^- < 0. \tag{48}$$

From (15), we observe that $\forall t, i \in \{1, \dots, d\}$:

$$\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}} \geq 1. \tag{49}$$

Using (49) and the definition of $\gamma_t(x_t)$ from (20), we have:

$$\left(1 - 2\gamma_t(x_t)_{i,i}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right) \begin{cases} = 1, & \text{if } \nabla\phi(x_t)_i \dot{x}^-_{t,i} \geq 0 \\ \leq -1, & \text{if } \nabla\phi(x_t)_i \dot{x}^-_{t,i} < 0. \end{cases} \tag{50}$$

Thus, irrespective of the sign of $\nabla\phi(x_t)_i \dot{x}^-_{t,i}$, we have from (50):

$$\nabla\phi(x_t)_i \dot{x}^-_{t,i}\left(1 - 2\gamma_t(x_t)_{i,i}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right) \geq |\nabla\phi(x_t)_i \dot{x}^-_{t,i}|, \tag{51}$$

and the strict inequality in (48) implies that

$$\nabla\phi(x_t)\dot{x}^-_{t,i} < 0 \tag{52}$$

for some $i \in \{1, \ldots, d\}$. Substituting (51) back in (47), we have:

$$\begin{aligned} \dot{\phi}^+(x_t) &= \sum_{i=1}^{d} \nabla\phi(x_t)_i \dot{x}^-_{t,i}\left(1 - 2\gamma_t(x_t)_{i,i}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right) \\ &\geq \sum_{i=1}^{d} |\nabla\phi(x_t)_i \dot{x}^-_{t,i}| \\ &> 0, \end{aligned} \tag{53}$$

where the last inequality is strict due to (52). Thus, we have $\dot{\phi}^+(x_t) > 0$ in this case, as desired. ■

Appendix D. Proof of Theorem 3

Before presenting the proof of theorem 3 we present an algorithm for recovery in the simpler setting of known $g(.)$ function. We extend the technique of algorithm 4 for the case of recovery to the safe region, when started at an unsafe point $x_0 \notin \mathcal{S}$. This property allows for the concept of forward persistence in our algorithm, forward convergence to the safe region $\mathcal{S}$, followed by forward convergence with respect to the same set. The recovery rate (w.r.t. time) is lower-bounded by the hyperparameter $\eta > 0$ and the recovery control u_{rec} is given by:

$$u_{rec} = u_{last} - g^+(x_t)\left(\dot{x}^-_t + n_d(x_t)\nabla\phi(x_t)\right), \tag{54}$$

where $n_d(x)$ is given by (24) as before.

The algorithm for recovery and safe control for a system with known $g(x)$ is given below in algorithm 5.

We have the following *forward persistence* guarantee for the above algorithm 5.

```
 Algorithm 5: Safety and recovery for known $g(x)$
Input: Control Barrier Function $\phi(.)$, Threshold: $\theta > 0$, Initial value of the state variable: $x_0 \in \mathcal{S}$,
function $g(.)$, and the nominal controller: $u_{nom}(.)$.
Hyperparameters: Correction $\gamma > \frac{1}{2}$, discretization time $T_s$, recovery hyperparameter: $\eta > 0$.
Initialize: $x_{t=0} \leftarrow x_0,\ u_{last} \leftarrow u_{nom}(x_0),\ x^-_{t=0} \leftarrow x_0$.
  for every $t > 0$, do
    Receive the current state $x_t$ from observation, and compute $\phi(x_t)$.
    Compute the time derivative of state variable: $\dot{x}^-_t \leftarrow \frac{1}{T_s}(x_t - x^-_t)$.
    Compute: $\dot{\phi}^-(x_t) \leftarrow \nabla\phi(x_t) \cdot \dot{x}^-_t$.
    Compute the Pseudoinverse $g^+(x_t)$ of $g(x_t)$ given by (4).
    switch ()
    case 1. $\phi(x) < \theta$:
      Compute $n_d(x_t)$ using (24), $u_{rec}$ using (54). Assign $u \leftarrow u_{rec}$.
    case 2. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) \geq 0$:
      $u \leftarrow u_{last}$.
    case 3. $\phi(x_t) \leq \theta$, and $\dot{\phi}^-(x_t) < 0$:
      Compute: $u_{corr}$, according to (25), assign $u \leftarrow u_{corr}$.
    default:
      $u \leftarrow u_{nom}(x_t)$.
    end switch
    Play $u$, $u_{last} \leftarrow u$, $x^-_t \leftarrow x_t$.
  end for
```

Theorem 8 (Forward Persistence for algorithm 5) *If $x_0 \notin \mathcal{S}_\theta$, then the set $\mathcal{S}_\theta$ is forward convergent (i.e., $\exists \tau > 0$ such that $x_\tau \in \mathcal{S}_\theta$) when the controller given by algorithm 5 is used and the rate of convergence is given by the hyperparameter η. Further, once the system state enters $\mathcal{S}_\theta$, the set $\mathcal{S}_\theta$ is forward invariant (i.e., $x_t \in \mathcal{S}_\theta\ \forall t \geq \tau$.) for the system when algorithm 5 is used.*

Proof The proof for the forward invariance of $\mathcal{S}_\theta$ after $t \geq \tau$ holds exactly the same way as in the proof of theorem 4. So, we need to prove the forward convergence property of $\mathcal{S}_\theta$ for algorithm 5. When $x_t \notin \mathcal{S}_\theta$, by the definition of $\phi(.)$ and $\mathcal{S}_\theta$ (see (6)), we have $\phi(x_t) < \theta$. Since $x_0 \notin \mathcal{S}_\theta$, we have $\phi(x_0) < \theta$. Thus, at the beginning, case 1 of algorithm 5 applies. In this case (for any $x_t \notin \mathcal{S}_\theta$), we have from (3):

$$\dot{x}^+_t = f(x_t) + g(x_t)u_{rec}. \tag{55}$$

Next we will show that $\dot{\phi}^+(x_t) = \eta > 0$ in this case. This will imply the existence of τ (forward convergence to $\mathcal{S}_\theta$. In particular, let

$$d(x_t, \mathcal{S}_\theta) = \theta - \phi(x_t) > 0, \tag{56}$$

denote the distance of the point x_t from the safety subset $\mathcal{S}_\theta$ in the barrier function space. Since $x_0 \notin \mathcal{S}_\theta$, and since $\dot{\phi}^+(x_t) \geq \eta$ whenever $\phi(x_t) < \theta$, we have within time:

$$\tau \leq \frac{d(x_0, \mathcal{S}_\theta)}{\eta},\ x_\tau \in \mathcal{S}_\theta. \tag{57}$$

So, it only remains to prove $\dot{\phi}^{+}(x_t) \geq \eta$, for completing the proof of the forward convergence property as shown below.

$$\begin{aligned}
\dot{\phi}^{+}(x_t) &= \nabla\phi(x_t) \cdot \dot{x}_t^{+} && (58)\\
&= \nabla\phi(x_t) \cdot \big(f(x_t) + g(x_t)u_{rec}\big) && (59)\\
&= \nabla\phi(x_t) \cdot \left(f(x_t) + g(x_t)\left(u_{last} - g^{+}(x_t)\left(\dot{x}_t^{-} + n_d(x_t)\nabla\phi(x_t)\right)\right)\right) && (60)\\
&= \nabla\phi(x_t) \cdot \left(\dot{x}_t^{-} - \dot{x}_t^{-} - n_d(x_t)\nabla\phi(x_t)\right) && (61)\\
&= \nabla\phi(x_t) \cdot \left(\frac{\eta}{\left\|\nabla\phi(x_t)\right\|^{2}}\nabla\phi(x_t)\right) && (62)\\
&= \eta > 0. && (63)
\end{aligned}$$

Here, the (58) uses (26); (59) uses (55); (60) uses (54); (61) uses (4) and (29) [since u is left-continuous this holds]; (62) uses (24); and finally (63) uses $\left\|\nabla\phi(x_t)\right\|^{2} = \nabla\phi(x_t) \cdot \nabla\phi(x_t)$. This completes the proof for the forward persistence of algorithm 5. ■

Now we are ready to present a proof of theorem 3, which is similar to the above proof but extending it for the unknown $g(.)$ setting.

Proof [Proof of Theorem 3]

The proof for the forward invariance of $\mathcal{S}_\theta$ is identical to the proof of theorem 2. So, we only need to prove the forward convergence property. We follow in a fashion similar to the proof of theorem 8, i.e. we show that if $\phi(x_t) < \theta$, then $\dot{\phi}^{+}(x_t) \geq \eta > 0$. This will imply:

$$\exists \tau > 0, \;\; \tau \leq \frac{d(x_0, \mathcal{S}_\theta)}{\eta}, \tag{64}$$

where $d(x_0, \mathcal{S}_\theta)$ is the distance of the initial point x_0 from the safety sub-set $\mathcal{S}_\theta$ in the barrier function space as defined in (56). For case 3 of the algorithm 3, we have from (3),

$$\dot{x}_t^{+} = f(x_t) + g(x_t)u_{rec}. \tag{65}$$

So, it remains to prove that $\dot{\phi}^{+}(x_t) \geq \eta$ when case 3 of the algorithm 3 occurs, which is shown below:

$$
\begin{aligned}
\dot{\phi}^+(x_t) &= \nabla\phi(x_t)\cdot \dot{x}_t^+ && (66)\\
&= \nabla\phi(x_t)\cdot \big(f(x_t)+g(x_t)u_{rec}\big) && (67)\\
&= \nabla\phi(x_t)\cdot \left(f(x_t)+g(x_t)\left(u_{last}-\hat{g}^+(x_t)\left(\gamma_t(x_t)\dot{x}_t^- + n_d(x_t)\nabla\phi(x_t)\right)\right)\right) && (68)\\
&= \nabla\phi(x_t)\cdot \left(\dot{x}_t^- - \Sigma_t\hat{\Sigma}_t\gamma_t(x_t)\dot{x}_t^- - \Sigma_t\hat{\Sigma}_t n_d(x_t)\nabla\phi(x_t)\right) && (69)\\
&= \nabla\phi(x_t)\cdot \left(\left(I-\Sigma_t\hat{\Sigma}_t\gamma_t(x_t)\right)\dot{x}_t^-\right) + \frac{\eta}{\left\|\nabla\phi(x_t)\right\|^2}\nabla\phi(x_t)\cdot\left(\Sigma_t\hat{\Sigma}_t\nabla\phi(x_t)\right) && (70)\\
&= \sum_{i=1}^{d}\left(\nabla\phi(x_t)_i\dot{x}_{t,i}^-\left(1-\gamma_t(x_t)_{i,i}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right)\right) + \frac{\eta}{\left\|\nabla\phi(x_t)\right\|^2}\sum_{i=1}^{d}\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\nabla\phi(x_t)_i^2 && (71)\\
&\geq \eta > 0. && (72)
\end{aligned}
$$

Here, (66) uses (26); (67) uses (65); (68) uses (23); (69) uses (18), (19), (13) and (29) [since u is left-continuous this holds]; (70) uses (24); (71) uses (16), (17); the inequality in (72) uses (49): $\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}} \geq 1$ which follows from (15),

$$
\left\|\nabla\phi(x_t)\right\|^2 = \sum_{i=1}^{d}\nabla\phi(x_t)_i^2,
$$

and the inequality

$$
\nabla\phi(x_t)_i\dot{x}_{t,i}^-\left(1-\gamma_t(x_t)\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}}\right)\geq 0,
$$

which follows from the definition of $\gamma_t(x_t)$ in (20) and using $\frac{\lambda_{t,i}}{\hat{\lambda}_{t,i}} \geq 1$. So, the first term of (71) is non-negative and the second term is at least η, which is positive by choice of the algorithm.

Thus, from (64), we observe that the algorithm 3 achieves the desired recovery/forward convergence property along with the forward invariance of $\mathcal{S}_\theta$ once reached $(t \geq \tau)$ as shown in the proof of theorem 2. This proves the forward persistence property of the algorithm 3 as desired.

■

Appendix E. Derivation of Policy Gradient for algorithm 2

In the REINFORCE algorithm, we employ stochastic policies to perform the task, i.e., at each step the action is chosen randomly from the probability distribution $\sim \pi(.|s)$, where s is the current state variable and π is the stochastic policy. In particular, we have a policy class $\Pi = \{\pi_\theta : \theta \in \Theta\}$, parameterized by θ. For instance, for a Neural Network based policy, θ would denote the weights of the Neural Network. Thus, the objective function becomes:

$$
J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right], \tag{73}
$$

where $a_t \sim \pi_\theta(.|s_t),\;\; s_{t+1} \sim P(.|s_t, a_t),\;\; \forall t$, and $P(.|s,a)$ denotes the stationary transition probability matrix of the underlying Markov Decision Process (the unknown system dynamics model).

The goal is to find the optimal stochastic policy within the class, i.e.

$$\theta^* = \arg\max_{\theta \in \Theta} J(\theta), \tag{74}$$

which is iteratively approximated at each episode τ using Stochastic Gradient Ascent as follows:

$$\theta_{\tau+1} = \theta_\tau + \alpha \nabla_\theta J(\theta_\tau). \tag{75}$$

Derivation of Policy Gradient

Let τ denote an episode, and $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$ denote the discounted sum reward for the trajectory in that episode. If the entire trajectory is sampled using the MDP's transition probabilities and the current stochastic policy π_θ, we denote it as $\tau \sim \pi_\theta$. Also, let $\mathcal{T}$ denote the set of trajectories and with an abuse of notation, we index each trajectory using τ. Then, we have:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \, \mathbb{E}_{\tau \sim \pi_\theta} \, R(\tau) \\
&= \nabla_\theta \sum_{\tau \in \mathcal{T}} \pi_\theta(\tau) R(\tau) \\
&= \sum_{\tau \in \mathcal{T}} \nabla_\theta \pi_\theta(\tau) R(\tau) \\
&= \sum_{\tau \in \mathcal{T}} \pi_\theta \left(\tau\right) \left(\frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} R(\tau) \right) \\
&= \mathbb{E}_{\tau \sim \pi_\theta} \, \nabla_\theta \ln \pi_\theta(\tau) R(\tau) \\
&= \mathbb{E}_{\tau \sim \pi_\theta} \, \nabla_\theta \Big[\ln \mu(s_0) + \ln \pi_\theta(a_0|s_0) + \ln \Pr[a_0' = C(s_0, \dot{x}_{s_0}^-, a_0, u_{last})] + \ln P(s_1|s_0, a_0') \Big] \, R(\tau) \\
&+ \Big[\ln \pi_\theta(a_1|s_1) + \ln \Pr[a_1' = C(s_1, \dot{x}_{s_1}^-, a_1, a_0')] + \ln P(s_2|s_1, a_1') \ldots \Big] \, R(\tau) \\
&= \mathbb{E}_{\tau \sim \pi_\theta} \sum_{t=0}^{\infty} \left[\nabla_\theta \ln \pi_\theta(a_t|s_t) \right] R(\tau).
\end{aligned} \tag{76}$$

Here, μ is the initial distribution over states and the last line uses the fact that $\mu,\; P(.|s,a),\; C(s_t, \dot{x}_t^-, a_t, a_{t-1})$ are independent of θ. In particular, $C(s_t, \dot{x}_t^-, a_t, a_{t-1})$ is a deterministic function given by (22). So, the gradient of these with respect to θ is zero. We observe that (76) is indeed the policy gradient used by algorithm 2.

To sample the gradient according to above, at each episode τ we sample a trajectory (called ”rollout” step) using the current stochastic policy π_{θ_τ} and collect the rewards $r(s_0, a_0), r(s_1, a_1), \ldots$. The trajectory length L_τ is a geometric random variable with success probability $1 - \gamma$. Using the current policy, the log probabilities of each step in the trajectory is computed. Note that in the above equation (76), when we sample the gradient from a trajectory, the $R(\tau)$ coefficient of a particular log probability $\ln \pi_{\theta_\tau}(a_t|s_t)$ would only contain the reward terms starting at t, that is $r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \ldots$, which is $Q^{\pi_{\theta_\tau}}(s_t, a_t)$. Thus, the gradient is estimated as:

$$\nabla_{\theta_\tau} J(\theta_\tau) \sim \sum_{t=0}^{\infty} \nabla_{\theta_\tau} \left(\ln \pi_{\theta_\tau}(a_t|s_t) \right) Q^{\pi_{\theta_\tau}}(s_t, a_t) \tag{77}$$

The estimation of this $Q^{\pi_{\theta_\tau}}(s_t, a_t)$ for each log probability could be done by a ”roll-in” step by further sampling using policy π_{θ_τ} starting from (s_t, a_t) and we stop at each step with probability $1-\gamma$. Such a nested sampling is very hard to implement, and in practice $Q^{\pi}_{\theta_\tau}(s_t, a_t)$ is estimated from the same roll-out trajectory as follows:

$$Q^{\pi}_{\theta_\tau}(s_t, a_t) = \sum_{k=t}^{L_\tau} \gamma^{k-t} r(s_k, a_k) \tag{78}$$

Using (77) and (78), the gradient $\nabla_{\theta_\tau} J(\theta_\tau)$ is sampled for each episode, and the policy gradient algorithm then uses this estimate for the gradient ascent step (75). In practice, $Q^{\pi}_{\theta_\tau}(s_t, a_t)$ are rescaled and normalized to get zero mean, variance 1 variables, to reduce the variance in the gradient estimation.

Appendix F. Implementation Details of Numerical Simulations from section 8

Simulation of Adaptive Safe Control Algorithms

We consider the simple one dimensional system below:

$$\dot{x} = 1.5x + u, \tag{79}$$

where $f(x) = 1.5x$, $g(x) = 1$, with a nominal controller: $u_{nom}(x) = -x$.We define the safe set as $\mathcal{S} = [-0.2, 0.2]$, and the initial state is $x_0 = 0.1999$. The shorthand for denoting each algorithm here is the same as the ones listed at the Simulations subsection of section 8.

1. **[sign flip]** This is our algorithm 4 where we use the nominal controller $u_{nom} = -x$, which makes the closed loop dynamics:
$$\dot{x} = 0.5x,$$
which is exponentially increasing with time and is expected to quickly exit the safe set $\mathcal{S}$. We choose the barrier function $\phi(x) = 1 - 25x^2$, corresponding to the safe set $\mathcal{S}$ and we take $\theta = 0.001$, and $\gamma = 4$. If algorithm 1 were used then an oracle reply of $\hat{\lambda} \leq 0.25$, would give the same result according to (53).

2. **[aCBF]** For the Adaptive CBF algorithm Taylor and Ames (2020), we have:
$f(x) = x$, $g(x) = 1$, $u_{nom}(x) = -x$, $F(x) = x$, $\theta^* = 0.5$, $\Gamma = 5$, $\hat{\theta}_0 = 0$, $\phi_a(x, \hat{\theta}) = 1 - 25x^2$.
This gives us an adaptation rule of:
$$\dot{\hat{\theta}} = \Gamma \tau(x, \hat{\theta}), \tag{80}$$
$$\text{where } \tau(x, \hat{\theta}) = -F(x)\frac{\partial \phi_a}{\partial x} = 50x^2. \tag{81}$$
And for the λ_{cbf} function we have:
$$\lambda_{cbf}(x, \hat{\theta}) = \hat{\theta} - \Gamma \frac{\partial \phi_a}{\partial \hat{\theta}} = \hat{\theta}. \tag{82}$$

So, we have the $aCBF$ controller given by the following quadratic program:

$$\begin{aligned} \text{minimize}_{u\in\mathbb{R}} \quad & \|u - u_{nom}(x)\| \\ \text{subject to} \quad & \sup_{u\in\mathbb{R}}(-50x[x+\hat{\theta}x+u]) \geq 0. \end{aligned}$$

3. **[RaCBF]** For the RaCBF algorithm Lopez et al. (2021), we have the same setting as above on the same system (79) and with the same λ_{cbf} (82) and adaptation rule (80). But here, we assume that the error in parameter estimation: $\tilde{\theta} = \theta^* - \hat{\theta} \in [-\tilde{\nu}, +\tilde{\nu}]$ (i.e. parameter estimation error lies in a bounded convex region). Here we have taken $\tilde{\nu} = 2$. We also took $\alpha(x)$ as the identity function: $\alpha(x) = x$, so we have the following quadratic program for the RaCBF controller:

$$\begin{aligned} \text{minimize}_{u\in\mathbb{R}} \quad & \|u - u_{nom}(x)\| \\ \text{subject to} \quad & \sup_{u\in\mathbb{R}}(-50x[x+\hat{\theta}x+u]) \\ & \geq -\phi_a(x,\hat{\theta}) + \frac{2}{5}. \end{aligned}$$

4. **[RaCBFS]** This algorithm referred to as the RaCBF+SMID algorithm in Lopez et al. (2021) improves upon the previous algorithm (RaCBF) of the same paper, by updating the uncertainty bound $\tilde{\nu}$ periodically after every few instances. The algorithm, updates its uncertainty over θ^* so that it is consistent with the trajectory history so far. Depending on the hyperparameter D (we have taken $D = 0.07$), we set r_{min} and r_{max} such that at time t:

$$\begin{gathered} \forall\theta \in [r_{min}, r_{max}] : \\ |\dot{x}_\tau - f(x_\tau) - g(x_\tau)u_\tau - F(x_\tau)\theta| \leq D \\ \forall\tau \in [0, t). \end{gathered}$$

Now with this uncertainty quantification over θ^* given by r_{min}, r_{max}, the algorithm updates $\tilde{\nu}$ every 2.5ms (where the time horizon is 0.25s) as follows:

$$\tilde{\nu} = \max\left\{|r_{min} - \hat{\theta}|, |r_{max} - \hat{\theta}|\right\}.$$

5. **[cbc]** This is the convex body chasing algorithm from Ho et al. (2021). Here we search for consistent models corresponding to a pair (α_t, β_t) at each round t which serve as estimates for the actual parameters (α^*, β^*) governing the system dynamics as: $\dot{x} = \alpha^* x + \beta^* u$, with initial uncertainties taken as $|\alpha^*| \leq 5$, and $2.5 \times 10^{-6} \leq \beta^* \leq 0.05$. Discretizing the system dynamics for a small sampling time $T_s = 2.5 \times 10^{-4}s$, we look for all candidates (α, β) consistent with the current trajectory history as the polytope $P_t \subset P_{t-1}$ given by the linear program corresponding to the constraints: $|\alpha x_i + \beta u_i - x_{i+1}| \leq \eta$, for all discrete points i in history up to time t, where η is a suitably chosen hyperparameter. At every discrete step t, then the candidate $(\alpha_t, \beta_t) \in P_t$ is chosen as the Euclidean projection of the previous candidate $(\alpha_{t-1}, \beta_{t-1})$ on the current polytope: P_t. At every round t, once the candidate model parameters (α_t, β_t) are found the control action u_t is then chosen as: $u_t = -\frac{\alpha_t}{\beta_t} x_t$. The loss function $\mathcal{G}(x_t, u_t)$ is taken as the indicator function, $\mathcal{G}(x_t, u_t) = \mathbf{1}\{x_t \notin \mathcal{S}\}$.

6. **[balsa]** This is the Bayesian learning for adaptive safety algorithm from Fan et al. (2020). Here a proxy $\hat{f}(x)$ for the function $f(x)$ is available to the controller and it has full knowledge of $g(x)$. From the pseudo control μ, it computes the control action as:

$$u = g^{-1}(x)(\mu - \hat{f}(x)), \text{ where} \tag{83}$$

$$\mu = \mu_{rm} + \mu_{pd} + \mu_{qp} - \mu_{ad}. \tag{84}$$

Here, the modelling error $\Delta(x) = f(x) - \hat{f}(x)$ is estimated by a Bayesian Neural Network estimator that takes as input x and predicts m, s as the mean and standard deviation of the prediction $\Delta(x)$. The neural network is trained on the trajectory data history after every $0.025s$, where the total horizon is $0.25s$. From this we take $\mu_{ad} \sim \mathcal{N}(m, s^2)$ is sampled for the adaptive part of the pseudo control to cancel the modelling error. For this setting, we have taken $\mu_{rm} = 0$ (since there is no tracking here), $\mu_{pd}(x) = \dot{x} - g(x)u_{last} + g(x)u_{nom}(x) = \mu_{nominal}(x)$, which is the effective pseudo control for our nominal controller used in ***sign flip***. Finally the pseudo control μ_{qp} for safety is obtained by solving the quadratic program:

$$\begin{aligned} \text{minimize}_{\mu_{qp}, d \in \mathbb{R}} \quad & \mu_{qp}^2 + 5d^2 \\ \text{subject to:} \quad & \phi_0 + \phi_1 \mu_{qp} \geq d \\ & d \leq 0, \end{aligned}$$

where $\phi_0 = \nabla\phi(x) \cdot \mu_{pd}(x)$, and $\phi_1 = \nabla\phi(x)$.

Simulation of Model-Free RL Algorithms

We consider a 2-dimensional non-linear kinematic system whose state space dynamics is given by:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0.1(x_1 - x_2)^2 \\ 0.007|x_2| \end{bmatrix} + \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \tag{85}$$

with $\lambda_1 = 2$, and $\lambda_2 = -1$. Here the function $f(x) = \begin{bmatrix} 0.1(x_1 - x_2)^2 \\ 0.007|x_2| \end{bmatrix}$ is unknown to the user and the matrix $g(x) = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$ clearly satisfies the assumptions with the row vectors being the orthogonal unit vectors along the x- and y- axes and the singular value's signs are known to the user with some confidence in advance. (This is the Oracle side information). In this case, the user has estimates $\hat{\lambda}_1 = 0.2$ and $\hat{\lambda}_1 = -0.1$ with the correct signs.

The state space of $[-1, +1] \times [-1, +1]$ is discretized into 100×100 cells to get a finite state space $\mathcal{S}$ and the action space of $[-7, +7] \times [-7, +7]$ is discretized into 10×10 cells to obtain a finite action space $\mathcal{A}$. The time is discretized to steps of size $\tau = 0.02$ seconds. The system has a start/source point of $s_0 = (-1.0, +1.0)$ and a destination/sink d at $(0.5, -0.2)$. Facing the destination point from the source, there is a big circular obstacle in front of it with centre at: $(h, k) = (0.07, -0.01)$ and a radius of $r = 0.11$. The system is considered unsafe if the agent comes close to the circular obstacle. It must maintain a gap distance of 7% of r from the nearest point of the obstacle boundary to be considered safe. This corresponds to the barrier function:

$$\phi(x) = (x_1 - h)^2 + (x_2 - k)^2 - (1.07r)^2 \tag{86}$$

for the above system. Given the system dynamics of (85), the goal is to learn a safe path from the source s_0 to the destination d.
The REINFORCE algorithm chooses a reward function $r(s, a)$ given by:

$$r(s,a,s') = \begin{cases} 21,000 & \text{if not terminated yet and } dist(d,s') < 1.6 \times \text{unit cell dimension; terminate} \\ 0 & \text{if terminated already} \\ 1/(dist(d,s')+0.01) & \text{if not terminated yet and } dist(d,s') > 1.6\, times \text{ unit cell dimension,} \end{cases} \tag{87}$$

where $dist(x, y)$ denotes the Euclidean distance between any two points x and y.

The simple REINFORCE algorithm Williams (1992) does not care about safety and learns a source to destination path in about 4500-5000 training episodes.

To set benchmrks for model-free safe RL algorithms we set the following benchmarks:

1. Benchmark 1: REINFORCE with penalized rewards:

$$r'(s,a,s') = r(s,a,s') - \mathbb{1}\{\phi(s') < 0\}.$$

2. Benchmark 2: CPO algorithm Achiam et al. (2017) with a single constraint:

$$\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t, s_{t+1}) \le 100,$$

where the cost function is given by:

$$c(s_t,a_t,s_{t+1}) = \begin{cases} (1000\phi(s_{t+1}))^2 & \text{if } \phi(s_{t+1}) < 0 \\ 0 & \text{otherwise} \end{cases}$$

The training is done for 5000 epochs/episodes and the discounting factor γ was taken as 0.7. Each rollout trajectory/episode/epoch runs for at most 100 steps and immediately stops if ”terminate” is reached.

Recovery Simulations

These simulations are performed for the same kinematic obstacle avoidance system descried by (85), (86). The hyperparameter η of algorithm 3 is taken as $\eta = 0.1$ and the CBF threshold $\theta = 0.1$ is used. Same Oracle estimates $\hat{\lambda}_1 = 0.2$ and $\hat{\lambda}_2 = -0.1$ are used as in the model-free RL simulations. The coordinates of five initial points are listed in the legend of figure 5.