

# Energy-Efficient Control Adaptation with Safety Guarantees for Learning-Enabled Cyber-Physical Systems

Yixuan Wang, Chao Huang and Qi Zhu

yixuanwang2024@u.northwestern.edu, {chao.huang,qzhu}@northwestern.edu

Electrical and Computer Engineering, Northwestern University, USA

## ABSTRACT

Neural networks have been increasingly applied for control in learning-enabled cyber-physical systems (LE-CPSs) and demonstrated great promises in improving system performance and efficiency, as well as reducing the need for complex physical models. However, the lack of safety guarantees for such neural network based controllers has significantly impeded their adoption in safety-critical CPSs. In this work, we propose a controller adaptation approach that automatically switches among multiple controllers, including neural network controllers, to guarantee system safety and improve energy efficiency. Our approach includes two key components based on formal methods and machine learning. First, we approximate each controller with a Bernstein-polynomial based hybrid system model under bounded disturbance, and compute a safe invariant set for each controller based on its corresponding hybrid system. Intuitively, the invariant set of a controller defines the state space where the system can always remain safe under its control. The union of the controllers' invariants sets then define a safe adaptation space that is larger than (or equal to) that of each controller. Second, we develop a deep reinforcement learning method to learn a controller switching strategy for reducing the control/actuation energy cost, while with the help of a safety guard rule, ensuring that the system stays within the safe space. Experiments on a linear adaptive cruise control system and a non-linear Van der Pol's oscillator demonstrate the effectiveness of our approach on energy saving and safety enhancement.

## CCS CONCEPTS

• **Computer systems organization** → **Embedded and cyber-physical systems**; • **Software and its engineering** → **Formal methods**.

## KEYWORDS

safety guarantees, invariant, control adaptation, neural network, energy saving

## 1 INTRODUCTION

Learning-enabled cyber-physical systems (LE-CPSs) [7, 15, 32, 37] often leverage machine learning techniques in their perception of the environment, and increasingly also in the consequent decision making process for planning, navigation, control, etc. In particular, neural network based controllers have been applied to a variety of LE-CPSs, such as building HVAC control [35], autonomous vehicles [23], smart grid [24] and robotics [38], due to their improvement on control performance and efficiency, and the fact that they do not require building a complex physical model of system dynamics. However, the uncertainties from the system input and the neural network itself make it quite challenging to ensure the safety of

neural-network controlled systems, which has significantly hindered their adoption in safety-critical CPSs [36].

In this work, we present an approach to leverage multiple controllers (including but not limited to neural network controller) and design an intelligent adaptor for switching among them to enhance both system safety and efficiency. At each sampling instant, the adaptor will choose the appropriate controller based on the current system state, and then applies the control input computed by the chosen controller. Our approach is motivated by the intuition that for many CPSs, multiple controllers designed based on different methodologies may each have their advantages at different system states. Thanks to the rapid advancement in learning-based control, there are a variety of learning methodologies that can help build neural network controllers for a system [11, 14, 22]. In addition, well-established model-based controllers, such as PID [4], LQR [6] and MPC [27], have their own advantages and could be complementary to data-driven neural network controllers. Then, with effective adaptation/switching strategy, multiple such controllers can jointly provide a larger operation space the facilities the improvement of system safety and efficiency.

With this intuitive motivation, our approach addresses two key technical challenges for **achieving the guarantee of system safety and the improvement of system energy efficiency**:

- We develop an invariant-based formal method for analyzing the safe configuration space of each controller to guide the adaptor for making the safe choice. Computing an invariant for classical systems have been extensively explored [29, 40]. However, it still remains an open problem for neural-network controlled systems (NNCSs). To address this challenge, our method provides a general approach to compute the (robust) invariant set for a large variety of controllers, including linear, polynomial, and neural network based ones. First, we approximate each controller with Bernstein polynomials under bounded error, and if the approximation precision is not sufficient, further refine the approximation by partitioning the system state space. Then, using over-approximation, we convert the system with each controller to a hybrid polynomial system under bounded disturbance and compute its (robust) invariant set with semi-definite programming (SDP) [40]. After obtaining the invariant of each controller, the adaptor can ensure the system safety by only choosing from the controllers whose invariant set covers the current system state.
- Given the computed invariant sets for the controllers, the second challenge is to intelligently switch among the controllers for reducing the energy consumption while guaranteeing safety. An effective strategy should select the appropriate controller from all safe choices to reduce the *overall* energy. Given the complexity and heterogeneity of multiple controllers, traditional methods based on optimization techniques can hardly handle it. Thus, we

develop a deep reinforcement learning (DRL) algorithm that automatically learns the adaptation strategy among safe controllers. At each sampling instant, the adaptor make a choice among the safe controllers based on the current system state, and find the most efficient one for reducing overall energy consumption. This is achieved by a carefully designed reward function in learning and a safety guard rule to discard the rare unsafe choice.

**Related work:** Our work is related to a rich literature on the safety verification of controlled systems. General safety verification relies on the computation of the reachable set, which contains all possible system states after a finite time for a given initial state set. Existing techniques falls into two main categories: 1) explicitly evaluating the reachable set [2, 3, 18, 31], and 2) implicitly considering the reachable set such as barrier certificates [17, 26, 28, 41]. The main difference between the invariant set in our approach and the reachable set in the literature is that the invariant set enables infinite-time safety verification while the reachable set provides a finite-time horizon. In [10, 18], Bernstein polynomials are applied in reachable set computation to approximate neural network controllers, but only on a small part of the state space. In contrast, our method applies Bernstein polynomials on the entire space for invariant set computation.

As we develop a DRL-based method with safety guarantees, our approach is related to the research topic of safe reinforcement learning [1, 20, 21]. The action exploration in RL causes the unsafe state. Thus, one idea is to force the agent to explore within the action set that is a prior known to be safe at a given state [13]. Our approach falls into the same idea but with the formally verified safety results. Formal methods are also used in [12] for linear adaptive cruise control(ACC) with the tabular Q-learning method. In contrast, our approach mainly targets neural network controllers.

Our work is also related to [19], which also tries to reduce system energy consumption while guaranteeing its safety. In particular, it guarantees the safety by deriving three different levels of safety sets and reduces the energy consumption by skipping the control input. However, that approach cannot be applied to neural network controllers, which is the focus of this work.

In summary, our work makes the following contributions:

- We develop a novel framework for energy-efficient control with safety guarantees by intelligently switching among multiple controllers (including neural network controllers) for LE-CPSs.
- Our framework guarantees infinite-time system safety, as long as the initial state is within the joint safe configuration space computed through a novel Bernstein polynomial based controller approximation method.
- We develop a new DRL method to learn an adaptation strategy that reduces the overall control energy consumption, while ensuring the system stay within the safe space.
- We conduct extensive experiments on a linear ACC system and a non-linear Van der Pol’s oscillator system. The results indicate the effectiveness of our approach in enhancing system safety and energy efficiency, when compared with using a single controller.

The rest of the paper is organized as follows. Section 2 introduces an illustrating example and defines problem formulation. Section 3 presents our approach. Section 4 shows the experimental results, and Section 5 provides further discussion. Section 6 concludes the paper.

## 2 PROBLEM FORMULATION

We will start with an illustrating example that helps explain the problems we are trying to solve, and then formally formulate them.

**Illustrating Example [Van der Pol’s Oscillator]:** Van der Pol’s oscillator [5] is a 2-dimensional non-linear system whose discrete-time dynamics is given as

$$\begin{cases} x_1(t+1) = x_1(t) + x_2(t)\delta \\ x_2(t+1) = x_2(t) + \delta[(1-x_1^2(t))x_2(t) - x_1(t) + u(t)] + \omega(t) \end{cases} \quad (1)$$

where  $\delta = 0.05$  is the sampling period,  $u(t)$  is the control input, and  $\omega(t)$  is the external disturbance that is uniformly random distributed over  $[-0.05, 0.05]$ .  $(x_1, x_2)$  are the state variables. The safe state space is a box  $[-2, 2] * [-2, 2]$ .

Previous works [16, 25, 34, 39] have designed neural networks to control the oscillator to the origin point. In this paper, we use two neural network controllers  $\kappa_1$  and  $\kappa_2$  for the oscillator that are designed with the DDPG method [22], as detailed in Section 4.

Assume the oscillator is at an initial state  $(1, 1)$  within the safe space, we are interested in the following questions. Does the system always stay within the safe box by applying  $\kappa_1$ ? If not, from what other initial states, the system could be always safe by applying  $\kappa_1$ ? Similar questions could be asked for the oscillator with controller  $\kappa_2$ . Then, if we verify that system with the initial state  $(1, 1)$  can be safely controlled by either  $\kappa_1$  or  $\kappa_2$ , which controller should we pick for the overall energy reduction? Trying to answer these questions motivates our formal definition of the problems below and our proposed approach. The illustrating example will be used throughout the paper and its solution will be shown in the experiments in Section 4.

**Formulation:** We consider a discrete-time polynomial system:

$$x(t+1) = f(x(t), u(t), \omega(t)), \forall t \geq 0, \quad (2)$$

where  $x(t) \in \mathbb{R}^n$  is the state variable,  $u(t) \in \mathbb{R}^m$  is the feedback control input variable,  $\omega(t) \in \mathbb{R}^k$  is a bounded external disturbance, and  $f: \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \rightarrow \mathbb{R}^n$  is a polynomial function.

The safe state space, the constraints on control input, and the external disturbance are given by

$$x(t) \in X, \quad u(t) \in U, \quad \omega(t) \in \Omega, \quad (3)$$

where  $X = \{x \in \mathbb{R}^n \mid \bigwedge_{i=1}^{n_0} h_{\omega,i}(x) \leq 0\}$ ,  $U \in \mathbb{R}^m$  and  $\Omega = \{\omega \in \mathbb{R}^k \mid \bigwedge_{i=1}^{n_\omega} h_{\omega,i}(\omega) \leq 0\}$ .  $h$  denotes the linear box constraint function. Moreover, We use 1-norm  $\|u(t)\|_1$  to denote the control/actuation energy consumption over time step  $t$  in this paper.

The trajectory  $\varphi_{x(0)}$  to the system (2) starting from an initial state  $x(0) \in X$  follows the discrete dynamics denoted by

$$\varphi_{x(0)}(t+1) = f(\varphi_{x(0)}(t), u(t), w(t)),$$

where  $\varphi_{x(0)}(0) = x(0)$ . As stated in Section 1, we may obtain/design multiple continuous controllers  $\kappa_i (i = 1, 2, \dots, M)$  for such a system, including neural network controllers. Then, the first problem we want to address is the safety verification of the system with each controller  $\kappa_i$ , formulated as the Problem 1.

**PROBLEM 1.** *Given a dynamical system defined with Equation (2) and (3) and  $M$  continuous controllers  $\kappa_i (i = 1, 2, \dots, M)$  including neural network controllers, the safety verification problem for the system with each controller  $\kappa_i$  is to determine whether the controlled trajectory  $\varphi_{x(0)}(t) \in X, \forall t \geq 0, \forall \omega(t) \in \Omega, \forall x(0) \in X$ .*

With the verification results of the above problem, we then want to design an adaptation strategy  $g(x(t)) : \mathbb{R}^n \rightarrow \{1, \dots, M\}$  to reduce the overall energy consumption by switching among controllers based on the system state. Here  $g$  maps the system state at each time step  $t$  to a controller choice. The overall control energy consumption is defined as in Definition 2.1, and the adaptation optimization problem with safety guarantees is formulated as the Problem 2.

*Definition 2.1.* If with infinite-time safety guarantee, the overall control energy consumption of the system in Equation (2) as a function of the adaptation strategy  $g$  is defined as <sup>1</sup>

$$e(g) = \sum_{t=0}^{+\infty} \|\kappa_{g(x(t))}\|_1$$

**PROBLEM 2.** Given a system defined with Equation (2) and (3) and multiple continuous controllers  $\kappa_i$  ( $i = 1, 2, \dots, M$ ) including neural network controllers, and  $\forall x(0) \in X$ , the problem of optimizing the overall energy consumption with safety guarantee by adaptation strategy function  $g$  is formulated as

$$\begin{cases} \min_g e(g), \\ \text{s.t. } x(t+1) = f(x(t), \kappa_{g(x(t))}, w(t)), \forall t \geq 0 \\ \varphi_{x(0)}(t) \in X, \forall t \geq 0, \forall \omega \in \Omega \end{cases}$$

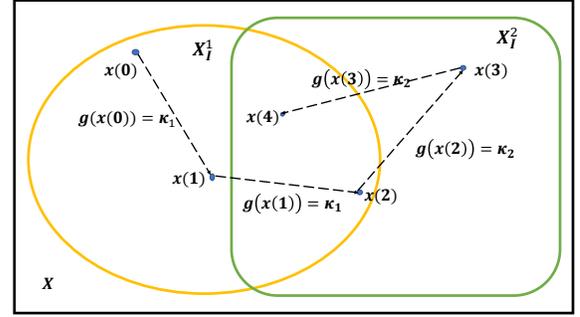
### 3 ENERGY-EFFICIENT CONTROLLER ADAPTATION WITH SAFETY GUARANTEE

As stated in Section 1, there are two key aspects of our approach: 1) computing the robust invariant set of each controller to build a joint safe configuration space, and 2) developing a DRL-based method to learn an efficient adaptation strategy within the joint safe configuration space.

For 1), informally, robust invariant set  $X_I^i \subseteq X$  of the controller  $\kappa_i$  is a set that any controlled trajectory starting from it will never leave it under any possible disturbance within  $\Omega$ . To compute the  $X_I^i$  ( $i = 1, 2, \dots, M$ ), we first apply Bernstein polynomials with bounded error to overly approximate each controller via state space partition. This approximation converts each original controlled system such as an NNCS into a hybrid polynomial system with bounded disturbance. We can then obtain the inner-approximation of the  $X_I^i$  with SDP by using existing techniques [40]. After that, we build the joint safe configuration space as the union of the computed inner-approximations of robust invariant sets, within which the infinite-time safety is guaranteed for the system.

For 2), we develop a DRL method to learn an efficient adaptation strategy within the joint safe configuration space, thus guaranteeing the system safety. More specifically, we set a reward function for punishing large control input and unsafe controller choice, so that the DRL agent can learn to reduce the energy consumption while maintaining safety. In the rare case that the DRL agent selects an unsafe controller choice, a safety guard rule will discard it and randomly choose a safe controller instead.

The schematic of our approach is illustrated in Figure 1. Its overall framework is described in Algorithm 1.



**Figure 1: Illustration of the schematic of our approach:** Consider the oscillator with two NN controllers  $\kappa_1$  and  $\kappa_2$ . Here  $X$  is the defined safe state space. Assume  $X_I^1, X_I^2$  are the robust invariant sets for each controller, respectively. The joint safe configuration space is  $X_I^1 \cup X_I^2$ . For safety guarantee, when system is at the state  $x(3) \in X_I^2$ , we should choose  $\kappa_2$ . For energy efficiency, when system is at the state  $x(2) \in X_I^1 \cap X_I^2$ , where it can be safely controlled by using either  $\kappa_1$  or  $\kappa_2$ , the adaptor decides to choose controller  $\kappa_2$  to reduce overall control energy cost.

#### Algorithm 1 Framework of Our Approach.

**Require:** Multiple controllers  $\kappa_i$  ( $i = 1, 2, \dots, M$ ) for the system

- 1: Compute robust invariant set  $X_I^i$  for each controller  $\kappa_i$ .
- 2: Build the joint safe configuration space as  $\cup_{i=1}^M X_I^i$ .
- 3: Learn the adaptation strategy  $g$  for reducing energy consumption and maintaining system state within  $\cup_{i=1}^M X_I^i$  (see Algorithm 2).
- 4: Initialization:  $t \leftarrow 0, x(0) \in \cup_{i=1}^M X_I^i$ .
- 5: **while true do**
- 6:   Read the system state  $x(t)$ .
- 7:   Adaptor  $g$  selects controller  $\kappa_{g(x(t))}$  based on  $x(t)$ , with safety guard rule applied if needed.
- 8:   Actuate the control input  $\kappa_{g(x(t))}$ .
- 9:    $t \leftarrow t + 1$
- 10: **end while**

### 3.1 Deriving Joint Safe Configuration Space for Safety Guarantee

In this section, we show how to compute  $X_I^i$  for the system with  $\kappa_i$ . We first formally define the concept of robust invariant set  $X_I^i$ .

*Definition 3.1.* Consider a system where the dynamics are defined as Equation (2) and the constraint is defined in Equation (3). For a controller  $\kappa_i$ ,  $X_I^i$  is called an invariant if

$$X_I^i = \{x(0) \mid \forall t \geq 0, \omega(t) \in \Omega, \cdot \varphi_{x(0)}(t) \in X_I^i\}.$$

Moreover, any set that is a subset of the invariant is called an *inner-approximate* invariant.

Let  $X_I^i$  be the invariant for the  $i$ -th controller. Then, the joint safe configuration space by multiple controllers can be built as  $\cup_{i=1}^M X_I^i$ , within which the infinite-time safety is guaranteed for the system.

**PROPOSITION 3.2. (Soundness).** For any initial state  $x(0) \in \cup_{i=1}^M X_I^i$ , the system where dynamics and constraints are defined in Equation (2) and (3) with controllers  $\kappa_i$  ( $i = 1, 2, \dots, M$ ) is ensured to have infinite-time safety guarantee.

<sup>1</sup> $\kappa_{g(x(t))}$  is short for  $\kappa_{g(x(t))}$  in this paper.

*Proof.* Given any initial state  $x(0) \in \cup_{i=1}^M X_J^i$ , we can at least find one feasible controller  $\kappa_j$  such that  $x(0) \in X_J^j$ . Then, the system safety is ensured if we always choose  $\kappa_j$  as the system controller, since as due to Definition 3.1, the controlled trajectory  $\varphi_{x(0)}(t) \in X_J^j \subseteq X, \forall t \geq 0, \forall \omega \in \Omega$ .

REMARK 1. *In general, it is intractable to compute the exact robust invariant set  $X_J^i$  for a nonlinear system [9], especially for neural-network controlled systems. Thus in this paper, we compute an inner-approximation of the robust invariant set for the system with each controller, as the inner-approximation maintains the safety guarantee and is more tractable [9]. For simplicity, we somewhat abuse the notation for  $X_J$ . **When we use  $X_J^i$  in the rest of this paper, we point to the inner-approximation of robust invariant set for  $\kappa_i$ .***

To compute  $X_J^i$ , we first want to approximate controller  $\kappa_i$  with polynomials under bounded error. This is because neural network controllers are complex and hard to tackle with, while the polynomials are more tractable. This approximation converts the original controlled system such as an NNCS into a polynomial system with bounded disturbance. Prior work [18] shows that Bernstein polynomials can be effectively applied to approximate any continuous controller. However, a single polynomial approximation may have to use a very high degree to achieve certain precision, while the computation complexity of  $X_J^i$  increases drastically as the degree increases. Also, the error reduction by this measure is often limited in practice, resulting in an inner-approximation that is too conservative. Thus, following the idea of interpolation, we propose a partition approach to achieve more precise approximation using polynomials with a much lower degree. With such partition approximation, the original controlled system is converted into a hybrid system with low degrees on each subsystem. We can then obtain the inner-approximation of the robust invariant set for such a hybrid system by using SDP. We detail each of these steps in the next.

**3.1.1 Single Bernstein Polynomial with Bounded Error for Controller Approximation.** We first introduce the concept of Bernstein polynomial. Let  $d = (d_1, \dots, d_n) \in \mathbb{R}^n$  and  $\kappa_i$  be a continuous controller of the system over state variables  $x = (x_1, \dots, x_n) \in X$ . The polynomials related to controller  $\kappa_i$

$$B_{\kappa_i, d}(x) = \sum_{\substack{0 \leq a_j \leq d_j \\ j=1,2,\dots,n}} \kappa_i \left( \frac{a_1}{d_1}, \dots, \frac{a_n}{d_n} \right) \prod_{j=1}^n \binom{d_j}{a_j} x_j^{a_j} (1-x_j)^{d_j-a_j}$$

are called Bernstein polynomials of  $\kappa_i$  under degree  $d$ .

To obtain the inner-approximation  $X_J^i$  for the system with controller  $\kappa_i$ , we first overly approximate  $\kappa_i$  by a single Bernstein polynomial with bounded error in Equation (4) on the safe state space  $X$ , similar as in [18],

$$\kappa_i(x) \in B_{\kappa_i, d}(x) + [-\hat{\epsilon}, \hat{\epsilon}], \forall x \in X, \quad (4)$$

where  $\hat{\epsilon}$  is the approximation error bound. Since the controllers in this paper are all considered as continuous functions, according to [8], we can always ensure that such approximation exists.

This approximation converts the system with  $\kappa_i$  into a polynomial system. The disturbance for the converted system is the Minkowski sum  $\oplus$  of external disturbance and approximation error. Now, the

**Table 1: Error bound by different approximation methods for the oscillator’s neural network controller  $\kappa_2$ . The control input space is normalized into interval [-1, 1]. Note that the partition approximation achieves the smallest bound. Simply increasing the degree will reduce the error bound but has limited effect.**

3-Partition (d=3)	Single (d=3)	Single (d=5)	Single (d=7)
<b>0.102</b>	0.27	0.169	0.163

system with controller  $\kappa_i$  is approximated as

$$x(t+1) = f(x(t), B_{\kappa_i, d}(x(t)), \hat{\omega}(t)), t \geq 0,$$

with  $\hat{\omega}(t) = \omega(t) \oplus \hat{\epsilon}$

However, this single Bernstein polynomial approximation is not sufficient for all encountered neural network controllers in our experiments. Recall the oscillator example with the neural network controller  $\kappa_2$  (details in Section 4), a single Bernstein polynomial with a low degree, e.g.,  $d = 3$ , for the approximation introduces a large error bound<sup>2</sup>, as shown in Table 1. With such large error bound, we just get an empty set for  $X_J^2$  by SDP. To reduce the error bound, a simple way is to increase the degree, e.g., set  $d = 5$  or  $7$  for Bernstein polynomial approximation. However, the reduction is limited in practice, as shown in Table 1. Moreover, increasing the approximation degree converts the system into a higher order polynomial system, resulting in drastically-increasing computation complexity for  $X_J^2$ . Thus, we propose a partition approximation method with low-degree polynomials to reduce the error bound.

**3.1.2 Partition Approximation.** We first partition  $X$  into  $P$  boxes with each box named as  $X^p$ , for  $p = (1, 2, \dots, P)$ :

$$X^{p_1} \cap X^{p_2} = \emptyset, \text{ if } p_1 \neq p_2 \text{ and } \cup_{p=1}^P X^p = X,$$

where  $p_1, p_2 \in \{1, 2, \dots, P\}$ . Now each box  $X^p$  has its own state constraints, defined as  $X^p = \{x \in \mathbb{R}^n \mid \bigwedge_{i=1}^{n_p} h_{p,i}(x) \leq 0\}$ , where  $h$  denotes the linear box constraint function.

Then, on each box  $X^p$ , a Bernstein polynomial  $B_{\kappa_i, d}^p$  is applied for approximation, reducing the overall approximation error bound  $\hat{\epsilon} = \max(\hat{\epsilon}^p)$ , where  $\hat{\epsilon}^p$  is the error bound on box  $X^p$  as

$$\kappa_i(x) \in B_{\kappa_i, d}^p(x) + [-\hat{\epsilon}^p, \hat{\epsilon}^p], \forall x \in X^p.$$

With such partition, the system with each controller can now be converted into a hybrid polynomial system. Each partition now acts as a subsystem with Bernstein polynomial control input on it. For this hybrid system, the new bounded disturbance is the Minkowski sum  $\oplus$  of external disturbance  $\omega$  and overall approximation error bound  $\max(\hat{\epsilon}^p)$ . Such a hybrid system can be expressed as

$$x(t+1) = f(x(t), \hat{u}(t), \hat{\omega}(t)), t \geq 0,$$

where  $\hat{u}(t)$  and  $\hat{\omega}(t)$  are

$$\hat{u}(t) = \sum_{p=1}^P \mathbf{1}_{X^p} \cdot B_{\kappa_i, d}^p(x(t)), \quad \hat{\omega}(t) = \omega(t) \oplus \max(\hat{\epsilon}^p), \quad (5)$$

where  $\mathbf{1}_{X^p}$  is an indicator function,  $p = (1, 2, \dots, P)$ .

When we use the partition approach to approximate the  $\kappa_2$  of the oscillator with  $d = 3$ , we achieve the smallest error bound,

<sup>2</sup> $d = 3$  actually means  $d = (3, 3)$ , representing that the highest polynomial degree for the oscillator state  $(x_1, x_2)$  is  $(3, 3)$ . The same applies to  $d = 5, 7$ .

when compared with  $d = 3, 5, 7$  under the non-partitioned single-polynomial approximation. This is shown in Table 1.

**REMARK 2.** For polynomial controller  $\kappa_i$  with degree  $d_0$ , if we choose Bernstein polynomial  $B_{\kappa_i, d_0}$  also with degree  $d_0$ , then the approximation error  $\hat{\epsilon} = 0$ . For the feed-forward neural network controller, the partition approximation greatly reduces  $\hat{\epsilon}$  in practice, compared to single-polynomial approximations.

Next, the inner-approximation of the robust invariant set of such a converted hybrid system is computed.

**3.1.3 Inner-approximation of Robust Invariant Set.** Each converted hybrid system has constraints defined as Definition 3.3.

**Definition 3.3.** Each converted hybrid polynomial system is subject to state constraints on each partition  $X^P$ , the entire safe space  $X$  and the disturbance  $\hat{\Omega}$  ( $\hat{\omega}$  defined in Equation (5)), which can be expressed as the following sets.

$$\begin{cases} X = \{x \in \mathbb{R}^n \mid \bigwedge_{i=1}^{n_0} h_{0,i}(x) \leq 0\} \\ X^P = \{x \in \mathbb{R}^n \mid \bigwedge_{i=1}^{n_p} h_{p,i}(x) \leq 0\} \\ \hat{\Omega} = \{\hat{\omega} \in \mathbb{R}^k \mid \bigwedge_{i=1}^{n_{\hat{\omega}}} h_{\hat{\omega},i}(\hat{\omega}) \leq 0\} \end{cases}$$

where  $p = (1, 2, \dots, P)$ , and  $h$  denotes the linear box constraint.

Then, following the method in [40], the inner-approximation of robust invariant set for such a hybrid system can be obtained by solving an SDP. First, we compute the one-step reachable set  $R(X)$  as the states reachable from the  $X$  within one-step computation, i.e.,

$$R(X) := \{x \mid x = f(x, \hat{u}, \hat{\omega}), x \in X, \hat{\omega} \in \hat{\Omega}\} \cup X.$$

Then, we define a continuous function  $v(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ . When  $v(x)$  is constrained to the polynomial type and the system state is constrained in a ball  $B$  with  $H$  as a constant

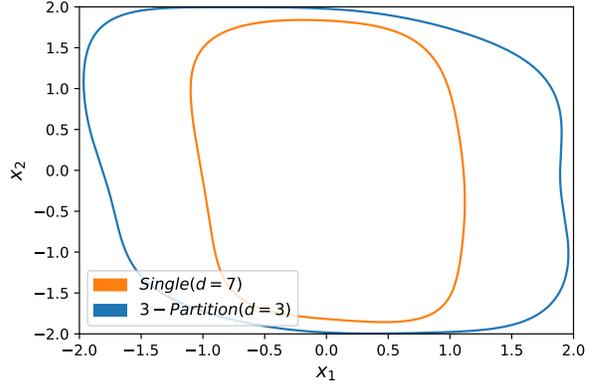
$$B = \{x \mid \|x\|_2 - H \leq 0\},$$

such that  $R(X) \subseteq B$ . Then, according to [40], the inner-approximation of the robust invariant set as  $\{x \in B \mid v(x) \leq 0\}$  can be obtained by solving an SDP optimization problem

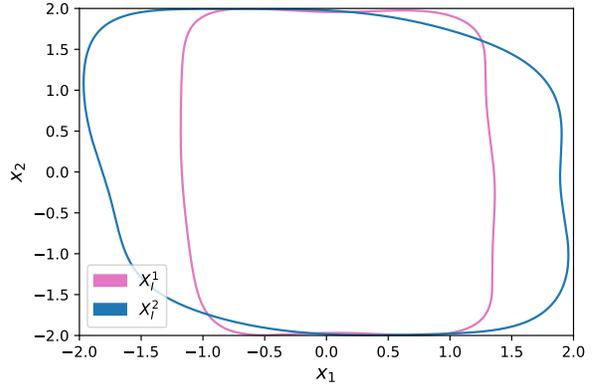
$$\begin{cases} \min & c \cdot w \\ v, s_{p,l_1}^{X^P}, s_{l_2}^{\hat{\Omega}}, s_p, s'_{1,j} & \\ v(x) - v(f(x, \hat{u}, \hat{\omega})) + \sum_{l_1=1}^{n_p} s_{p,l_1}^{X^P} h_{i,l_1}(x) + & \\ \sum_{l_2=1}^{n_{\hat{\omega}}} s_{l_2}^{\hat{\Omega}} h_{\hat{\omega},l_2}(\hat{\omega}) - s_p h(x) \in \text{SOS}(x, \hat{\omega}), & \\ (1 + h_{0,j}^2)v(x) - h_{0,j}(x) - s'_{1,j} h(x) \in \text{SOS}(x), & \end{cases}$$

where  $c \cdot w = \int_B v(x) dx$ ,  $c$  is the unknown coefficient vector in  $v(x)$ , and  $w$  is the vector of the integration for each monomial in  $v(x)$  over  $B$ .  $s_{p,l_1}^{X^P}$ ,  $s_{l_2}^{\hat{\Omega}}$ ,  $s_p$ ,  $s'_{1,j}$  are the sum-of-squares(SOS) polynomials, where  $p = (1, 2, \dots, P)$ ,  $l_1 = (1, 2, \dots, n_p)$ ,  $l_2 = (1, 2, \dots, n_{\hat{\omega}})$  and  $j = (1, 2, \dots, n_0)$ .  $s_{p,l_1}^{X^P}$ ,  $s_{l_2}^{\hat{\Omega}}$ ,  $s_p \in \text{SOS}(x, \hat{\omega})$  and  $s'_{1,j} \in \text{SOS}(x)$ .

**Safe Controller for the Illustrating Example:** Recall the illustrating example. By solving the above SDP problem, we obtain  $X_I^2$  for the controller  $\kappa_2$  in the oscillator example with different approximation methods. The proposed partition approximation achieves better result than the single-polynomial ones, as shown in Figure 2. Thus, we use it to obtain  $X_I^1$  and  $X_I^2$ , as shown in Figure 3. It is easy to check that state  $(1, 1)$  belongs to the invariant intersection in Figure 3, thus guaranteeing the safety by either  $\kappa_1$  or  $\kappa_2$ .



**Figure 2:**  $X_I^2$  of oscillator with  $\kappa_2$  by SDP for different approximation methods. For  $Single(d=3)$ , the SDP returns an empty set due to its large error bound. For  $Single(d=7)$ , we obtain a non-empty inner-approximation but it is much more conservative/inaccurate than the  $3-Partition$  method, where 3 polynomials are used with the partition approximation. Moreover, it took about 2 hours to compute  $X_I^2$  by  $3-Partition$  and 41 hours by  $Single(d=7)$  with Mosek 8.0 and Matlab 2015.



**Figure 3:** Inner-approximation of the robust invariant sets  $X_I^1, X_I^2$  for oscillator controlled by the DDPG controllers  $\kappa_1, \kappa_2$ . The joint safe configuration space is  $X_I^1 \cup X_I^2$ . The controller adaptation learned by DRL will try to reduce the overall energy consumption by intelligently switching  $\kappa_1$  and  $\kappa_2$  while maintaining the safety.

Once the joint safe configuration space is derived, we can develop a DRL method to learn an energy-saving adaptation strategy with the safety guarantees, as introduced next.

## 3.2 DRL-based Control Adaptation

Within the safe configuration space  $\mathcal{S} = \cup_{i=1}^M X_I^i$ , we develop a Double DQN algorithm [33] to learn an energy-efficient adaptation strategy with safety guarantees. The learning process can be formulated as a Markov decision process (MDP) with a tuple  $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ .  $\mathcal{S}$

represents the state space of MDP.  $\mathcal{A}$  is the action space.  $\mathcal{P}$  is the state transition probability, mapping the function  $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ .  $\gamma$  is the discounted factor, and  $\mathcal{R}$  is the reward function encoding the desired goal of the reinforcement learning agent. More specifically, they are formulated as follows.

**State:** To ensure that the adaptation guarantees safety, the state space  $\mathcal{S}$  here is defined as the joint safe configuration space. Moreover, the state of the Double DQN agent is the system state  $x(t)$ .

**Action:** We define the action space as the discrete space  $\mathcal{A} = \{1, \dots, M\}$ . At time  $t$ ,  $a(t) \in \mathcal{A}$  means that the Double DQN agent chooses controller  $\kappa_{a(t)}$  for controlling the system.

**Reward Function:** Reward design encodes the desired goals for the agent. First, we set a punishment for the energy cost as  $-||u(t)||_1$  for the time step  $t$ . In order to maximize the cumulative reward, the agent needs to learn to avoid large control input. Moreover, the agent needs to set a punishment for choosing any unsafe controller, i.e., choosing controller  $\kappa_{a(t)}$  while  $x(t) \notin X_I^{a(t)}$  (note that  $x(t) \in \cup_{i=1}^M X_I^i$ , which means a safe choice does exist), so that it can learn to avoid such choice. With these two considerations, we design the reward function as

$$r(x(t), a(t), x(t+1)) = \begin{cases} C - \lambda ||u(t)||_1 & \text{Otherwise,} \\ R_{pub} & \text{if } x(t) \notin X_I^{a(t)}, \end{cases} \quad (6)$$

where  $C$  is a positive constant,  $\lambda$  is the weight for the punishment of energy cost  $-||u(t)||_1$ ,  $R_{pub}$  is a negative constant that punishes the agent for choosing any potential unsafe controller. The reward is -100 when the state is controlled out of the safe space in training. Note that -100 is applied at most once during a training epoch, as the epoch would end after that.

We develop the Double DQN algorithm to learn an efficient and safe adaptation strategy based on the MDP specified above. The details of the learning process is shown in Algorithm 2.

---

**Algorithm 2** Double DQN for Learning Adaptation Strategy

---

**Require:** Joint safe configuration space  $\cup_{i=1}^M X_I^i$

- 1: Initialize replay memory  $D$ ,  $Q$  network with parameters  $\theta$ , target network  $\hat{Q}$  with parameters  $\hat{\theta}$ , and update period  $C_0$ .
- 2: **for**  $epoch = 0, \dots, N$  **do**
- 3:   Randomly initialize state  $x(0) \in \cup_{i=1}^M X_I^i$ .
- 4:   **for**  $t = 0, \dots, T$  **do**
- 5:      $a(t) = \epsilon - greedy(Q(x(t)), \epsilon)$ .
- 6:     **if**  $x(t) \notin X_I^{a(t)}$  **then**
- 7:       Update reward punishment  $R_{pub}$  and break.
- 8:     **end if**
- 9:     Switch to controller  $\kappa_{a(t)}$ ;  $x(t)$  evolves to  $x(t+1)$ ; receive reward  $r(t)$ ; store tuple  $(x(t), a(t), x(t+1), r(t))$  into  $D$ .
- 10:    Sample mini-batch from  $D$ ; compute TD error [30].
- 11:    Apply gradient descent to  $Q$ .
- 12:    Update  $\hat{\theta} = \theta$  every  $C_0$  steps.
- 13:   **end for**
- 14: **end for**
- 15: **return**  $Q$  forwarding function as the adaptation strategy  $g$ .

---

**Safety Guard Rule:** Although we have defined a punishment for any unsafe choice, the Double DQN agent may still occasionally

choose unsafe controllers due to the trial-and-error nature of reinforcement learning. In those rare cases, we set a safety guard rule for ensuring system safety. Specifically, if the agent chooses an unsafe controller, the safety guard will discard it and randomly choose a safe one. Note that as long as the system initial state belongs to the joint safe configuration space  $\mathcal{S}$ , such safe choice always exists.

**Energy-saving Controller for the Illustrating Example:** In this example, the learned Double DQN agent chooses controller  $\kappa_1$  for the system at the initial state  $(1, 1)$ , later switches between  $\kappa_1$  and  $\kappa_2$ , and keeps using  $\kappa_1$  after around 20 steps as the state is approaching the origin point.

## 4 EXPERIMENTAL RESULTS

Experiments on the illustrating Van der Pol’s oscillator example and an adaptive cruise control (ACC) system, a common safety-critical system, are conducted to evaluate the effectiveness of our approach.

### 4.1 Van der Pol’s Oscillator

The Van der Pol’s oscillator system is defined in Equation (1) in Section 2. As stated before, we train two controllers by the DDPG method with different reward designs, and name them  $\kappa_1$  and  $\kappa_2$ . The reward for the DDPG learning can be expressed as (note that this is for learning the underlying controllers  $\kappa_1$  and  $\kappa_2$ , and different from the Double DQN learning for controller adaptation in Equation (6)):

$$r = 10 - \lambda_1(|x_1| + |x_2|) - \lambda_2(|u| + |u - u'|),$$

where 10 is the reward for each safely-controlled step,  $\lambda_1, \lambda_2 \geq 0$  are weights for state and control input penalty, respectively, and  $u'$  is the control input of previous step. For controller  $\kappa_1$ , both  $\lambda_1$  and  $\lambda_2$  are set to 1. For  $\kappa_2$ ,  $\lambda_1$  and  $\lambda_2$  are set to 5 and 0.2, respectively.

To compute the robust invariant sets, both controllers need to be approximated by Bernstein polynomials with bounded errors via partitioning. Each inner-approximation of the robust invariant set is obtained, as shown in Figure 3. Then the Double DQN is applied to learn an adaptation strategy between  $\kappa_1$  and  $\kappa_2$ . The  $C$  in the reward Equation (6) is 2,  $\lambda$  is 1, and  $R_{pub}$  is -20. The hyper-parameters in Algorithm 2 is set as follows: the size of the replay buffer  $D$  is 5000,  $\gamma$  is 0.99,  $C_0$  is 100, and the learning rate is  $1e-4$ .

We set three baselines: using  $\kappa_1$  only, using  $\kappa_2$  only, and random adaptation. We conduct 500 test cases by randomly picking 500 initial states within  $X_I^1 \cup X_I^2$ , and run all the methods from the same initial state for 200 control steps for each case.

**Table 2: Comparison of results for the oscillator experiment.**

	Ours	$\kappa_1$ only	$\kappa_2$ only	Random
Safe control rate	<b>100 %</b>	86.4 %	95.6 %	92 %
Energy cost	<b>127.8</b>	130.1	164.1	383.8

**Comparison among Different Methods:** We compare the average system safety rate and energy cost among different methods, and show them in Table 2. Our approach formally guarantees 100% safety as the initial state is within  $X_I^1 \cup X_I^2$ , while the other methods all have significant number of unsafe cases. Note that the three baselines do not employ the safety guard rule, since they do not have the capability

to compute the safe invariant sets. However, for our approach, even without the safety guard rule, our system is safe for more than 99.6% of the cases, which shows the effectiveness of Double DQN for switching among controllers. Moreover, our approach also provides the lowest energy cost, which demonstrates that the reward function design in our Double DQN is effective for overall energy saving.

## 4.2 Adaptive Cruise Control

We also conducted experiments on an ACC system. We consider two vehicles in the system. The front vehicle is running with a velocity  $v_f$ , while the following/ego vehicle brakes or accelerates according to the control design. Overall, the system dynamics is

$$\begin{cases} s(t+1) = s(t) - (v(t) - v_f(t))\delta, \\ v(t+1) = v(t) - (kv(t) - u(t))\delta, \end{cases}$$

where  $s$  represents the distance between vehicles,  $v$  is the velocity of the ego vehicle,  $u$  is the control input,  $\delta = 0.1$  is the sampling period, and  $k = 0.2$  is the velocity resistance.  $v_f = 40 + w$ , where  $w$  is uniformly random distributed over  $[-4, 4]$ . The definition of the safe set  $X$  over state variable  $(s, v)$  is

$$X := \{(s, v) \mid s \in [120, 180], v \in [25, 55]\}.$$

Here we want this ACC system to be controlled stably to the equilibrium state  $(150, 40)$ . To this end, we design two different controllers – one is a Linear-Quadratic Regulator (LQR) controller  $\kappa_1$ , and the other is a neural network controller  $\kappa_2$  obtained by the DDPG method. The LQR’s parameters representing the weights for state and control input are set to 2 and 0.4, respectively. The DDPG controller has the reward function as

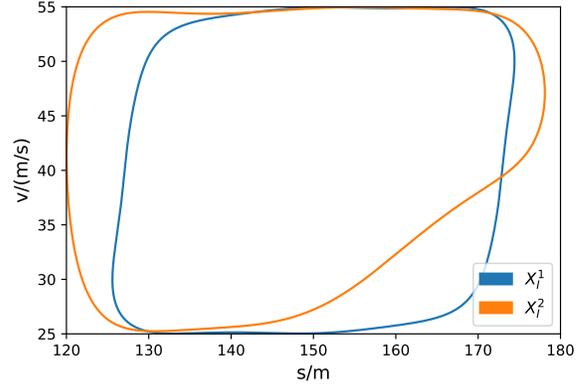
$$r = 25 - 0.5(|s - 150| + |v - 40| + |u| + |u - u'|),$$

where 25 is the reward for every successful control and  $u'$  is the previous control input (note that this reward function is for learning the underlying controller  $\kappa_2$ , not the Double DQN for adaptation).

For the LQR controller  $\kappa_1$ ,  $X_I^1$  can be directly obtained by SDP. For the DDPG controller  $\kappa_2$ , Bernstein polynomial approximation via partition is first applied, converting the NNCS into a hybrid polynomial system with bounded disturbance. Then,  $X_I^2$  is obtained for such a hybrid system.  $X_I^1$  and  $X_I^2$  for ACC are shown in Figure 4. Then, Double DQN is applied to learn the adaptation strategy.  $C$  in Equation (6) is 25,  $\lambda$  is 1, and  $R_{pub}$  is -50. The hyper-parameters in Algorithm 2 are set as follows: the size of the replay buffer  $D$  is 5000,  $\gamma$  is 0.99,  $C_0$  is 100, and the learning rate is  $1e-4$ .

We consider three baselines: using LQR  $\kappa_1$  only, using DDPG controller  $\kappa_2$  only, and random adaptation between the two. We conduct 500 test cases by randomly sampling 500 initial states within  $X_I^1 \cup X_I^2$ , and run all the methods from the same initial state for 100 control steps for each case.

**Comparison among Different Methods:** The comparison of our approach with three baselines are shown in Table 3. Consistent with the results for the Van der Pol’s oscillator, our approach achieves the least average energy cost and guarantees 100% safe control rate, outperforming the baselines. Note that in this example, even without the safety guard rule, our approach achieves 100% safe rate (although the safety guard is still needed in practice for guaranteeing safety).



**Figure 4: Inner-approximation of the robust invariant sets  $X_I^1, X_I^2$  for ACC by LQR controller  $\kappa_1$  and DDPG controller  $\kappa_2$ . Joint safe configuration space is  $X_I^1 \cup X_I^2$ . The controller adaptation learned by Double DQN reduces the overall control energy cost while maintaining the safety.**

**Table 3: Comparison of results for the ACC experiment.**

	Ours	$\kappa_1$ only	$\kappa_2$ only	Random
Safe control rate	<b>100 %</b>	97.4 %	99 %	99.6 %
Energy cost	<b>835.7</b>	854.8	997.5	1085.5

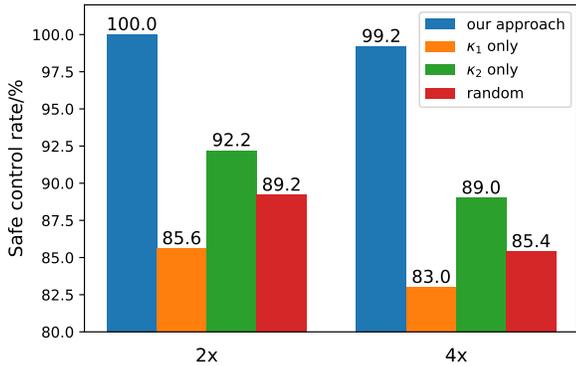
## 5 DISCUSSION

**Scale the External Disturbance:** In practice, the system may encounter stronger external disturbance that exceeds the original design expectation. The theoretical robust invariant set of the corresponding system would shrink by some extent in such scenario, and thus safety is no longer guaranteed with the computed invariant. Although, with the inner approximation, the system might still have some buffer to be able to handle such stronger external disturbance. We demonstrate this conjecture in both ACC and oscillator examples by scaling the disturbance to twice and four times of the design assumption.

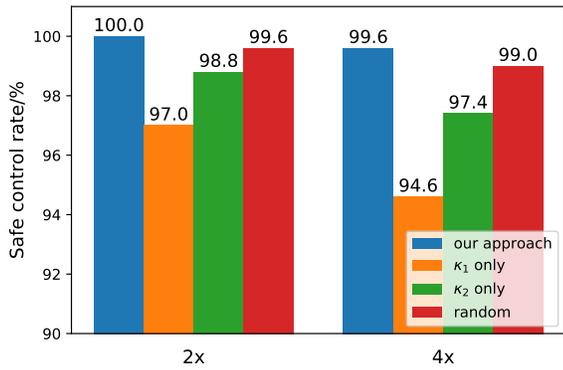
The results of this study are shown in Figure 5 and 6. As the disturbance scales, the safe control rates for all methods decrease. However, the safe rate of our approach decreases at a much slower pace than the baselines, showing its robustness to external disturbance (even when the disturbance unexpectedly exceeds the design assumption). Note that the safe rate of our approach is still 100% in the experiments when the disturbance doubles, although this is not always guaranteed.

**States Outside of the Joint Safe Configuration Space:** There might also be cases in practice where we cannot set the initial state to be within the joint safe configuration space and thus cannot guarantee the system safety. In this study, we conduct experiments to evaluate how our approach performs in such scenario, and how it compares with the baselines. Specifically, we train a Double DQN agent with the same reward design on the entire state space  $X$ , and we do not end a training epoch if the agent chooses an unsafe controller.

The results for the oscillator example (initial state  $x(0) = (-2, 2)$ ) and the ACC example (initial state  $x(0) = (177.74, 31.16)$ ) are shown



**Figure 5: Safe control rate for our approach and the baselines when scaling the external disturbance in the oscillator example by twice (left) and four times (right) of the design assumption.**



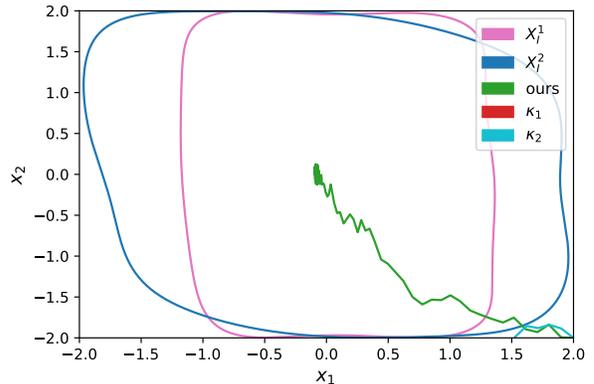
**Figure 6: Safe control rate for our approach and the baselines when scaling the external disturbance in the ACC example to twice (left) and fourth times (right) of the design assumption.**

in Figure 7 and 8, respectively. We can see that our approach can pull the system state into the joint safe configuration space and then always maintain its safety from that moment, while the baselines with a single controller cannot. This shows that even when the initial state is outside of the joint safe configuration space, our approach may still be able to adapt the system into such space for ensuring system safety.

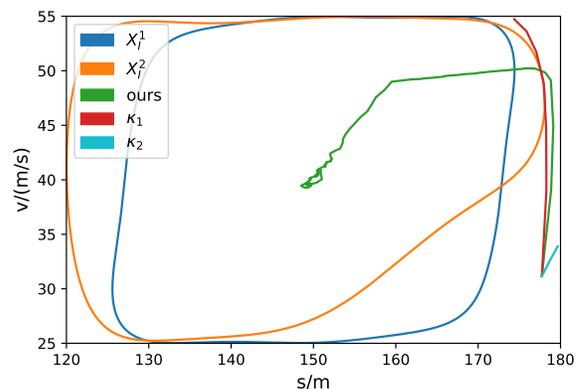
**Limitation:** It is difficult for our current approach to handle high-dimensional systems. First, it is challenging to accurately approximate neural network controllers with high-dimensional input by Bernstein polynomials. Second, the computation complexity of the robust invariant set increases drastically as the system state dimension increases. Our future work will focus on addressing these issues.

## 6 CONCLUSIONS

We present a controller adaptation approach based on formal methods and machine learning to guarantee system safety and improve energy efficiency for LE-CPSs. In particular, we first compute a joint safe configuration space of the multiple controllers, including neural



**Figure 7: Robust invariant sets and system trajectory under different methods when initial state  $[2, -2]$  is outside of the joint safe configuration space for the oscillator example. Our approach is able to pull the state into the joint safe configuration space and maintain system safety.  $\kappa_1$  fails after one step control (not visible),  $\kappa_2$  fails after a few steps. (Best viewed in color)**



**Figure 8: Robust invariant sets and system trajectory under different methods when initial state  $[177.74, 31.16]$  is outside of the joint safe configuration space for the ACC example. Our approach can pull the state into the joint safe configuration space and maintain system safety. LQR controller  $\kappa_1$  and DDPG controller  $\kappa_2$  both fail after a few steps. (Best viewed in color)**

network ones, with a novel method based on Bernstein polynomial approximation, state partitioning, conversion to hybrid systems, and robust invariant set computation. We then develop a DRL-based method to intelligently switch between controllers for reducing energy consumption while maintaining system safety by keeping its state within the safe space. Experimental results and analysis on two different case studies demonstrate that our approach significantly outperforms the baselines in both safety and energy efficiency.

## REFERENCES

- [1] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [2] Hirokazu Anai and Volker Weispfenning. 2001. Reach set computations using real quantifier elimination. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 63–76.
- [3] Eugene Asarin, Thao Dang, and Antoine Girard. 2003. Reachability analysis of nonlinear systems using conservative approximation. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 20–35.
- [4] Karl Johan Åström, Tore Hägglund, Chang C Hang, and Weng K Ho. 1993. Automatic tuning and adaptation for PID controllers—a survey. *Control Engineering Practice* 1, 4 (1993), 699–714.
- [5] Ramiro S Barbosa, JA Tenreiro Machado, BM Vinagre, and AJ Calderon. 2007. Analysis of the Van der Pol oscillator containing derivatives of fractional order. *Journal of Vibration and Control* 13, 9-10 (2007), 1291–1301.
- [6] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. 2002. The explicit linear quadratic regulator for constrained systems. *Automatica* 38, 1 (2002), 3–20.
- [7] Feiyang Cai and Xenofon Koutsoukos. 2020. Real-time Out-of-distribution Detection in Learning-Enabled Cyber-Physical Systems. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCP)*. IEEE, 174–183.
- [8] Louis De Branges. 1959. The stone-weierstrass theorem. *Proc. Amer. Math. Soc.* 10, 5 (1959), 822–824.
- [9] Elena De Santis, Maria Domenica Di Benedetto, and Luca Berardi. 2004. Computation of maximal safe sets for switching systems. *IEEE Trans. Automat. Control* 49, 2 (2004), 184–195.
- [10] Jiameng Fan, Chao Huang, Wenchao Li, Xin Chen, and Qi Zhu. 2020. ReachNN\*: A Tool for Reachability Analysis of Neural-Network Controlled Systems. In *International Symposium on Automated Technology for Verification and Analysis (ATVA)*.
- [11] Scott Fujimoto, Herke Van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477* (2018).
- [12] Nathan Fulton and André Platzer. 2018. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [13] Javier Garcia and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.
- [14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290* (2018).
- [15] Charles Hartsell, Nagabhushan Mahadevan, Shreyas Ramakrishna, Abhishek Dubey, Theodore Bapty, Taylor Johnson, Xenofon Koutsoukos, Janos Sztipanovits, and Gabor Karsai. 2019. Model-based design for cps with learning-enabled components. In *Proceedings of the Workshop on Design Automation for CPS and IoT*. 1–9.
- [16] Ali Heydari. 2014. Revisiting approximate dynamic programming and its convergence. *IEEE transactions on cybernetics* 44, 12 (2014), 2733–2743.
- [17] Chao Huang, Xin Chen, Wang Lin, Zhengfeng Yang, and Xuandong Li. 2017. Probabilistic safety verification of stochastic hybrid systems using barrier certificates. *ACM Transactions on Embedded Computing Systems (TECS)* 16, 5s (2017), 1–19.
- [18] Chao Huang, Jiameng Fan, Wenchao Li, Xin Chen, and Qi Zhu. 2019. ReachNN: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 1–22.
- [19] Chao Huang, Shichao Xu, Zhilu Wang, Shuyue Lan, Wenchao Li, and Qi Zhu. 2020. Opportunistic Intermittent Control with Safety Guarantees for Autonomous Systems. *arXiv preprint arXiv:2005.03726* (2020).
- [20] Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, and Joost-Pieter Katoen. 2016. Safety-constrained reinforcement learning for MDPs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 130–146.
- [21] D Kroening, A Abate, and M Hasanbeig. 2020. Towards verifiable and safe model-free reinforcement learning. *CEUR Workshop Proceedings*.
- [22] Timothy P Lillierap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).
- [23] Lei Lin, Siyuan Gong, Tao Li, and Srinivas Peeta. 2018. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles. In *The Autonomous Vehicles Symposium*, Vol. 2018.
- [24] Renzhi Lu, Seung Ho Hong, and Xiongfeng Zhang. 2018. A dynamic pricing demand response algorithm for smart grid: reinforcement learning approach. *Applied Energy* 220 (2018), 220–230.
- [25] Ali Mesbah. 2016. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine* 36, 6 (2016), 30–44.
- [26] Stephen Prajna. 2006. Barrier certificates for nonlinear model validation. *Automatica* 42, 1 (2006), 117–126.
- [27] S Joe Qin and Thomas A Badgwell. 2003. A survey of industrial model predictive control technology. *Control engineering practice* 11, 7 (2003), 733–764.
- [28] Muhammad Zakiyullah Romdloh and Bayu Jayawardhana. 2016. Stabilization with guaranteed safety using control Lyapunov–barrier function. *Automatica* 66 (2016), 39–47.
- [29] Matthias Rungger and Paulo Tabuada. 2017. Computing robust controlled invariant sets of linear systems. *IEEE Trans. Automat. Control* 62, 7 (2017), 3665–3670.
- [30] Richard S Sutton. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3, 1 (1988), 9–44.
- [31] Claire J Tomlin, Ian Mitchell, Alexandre M Bayen, and Meeke Oishi. 2003. Computational techniques for the verification of hybrid systems. *Proc. IEEE* 91, 7 (2003), 986–1001.
- [32] Cumhuri Erkan Tuncali, James Kapinski, Hisahiro Ito, and Jyotirmoy V Deshmukh. 2018. Reasoning about safety of learning-enabled components in autonomous cyber-physical systems. In *Proceedings of the 55th Annual Design Automation Conference*. 1–6.
- [33] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.
- [34] Shiyin Wei, Xiaowei Jin, and Hui Li. 2018. General solutions for nonlinear differential equations: a deep reinforcement learning approach. *arXiv preprint arXiv:1805.07297* (2018).
- [35] Tianshu Wei, Yanzhi Wang, and Qi Zhu. 2017. Deep reinforcement learning for building HVAC control. In *Proceedings of the 54th Annual Design Automation Conference 2017*. 1–6.
- [36] Weiming Xiang and Taylor T Johnson. 2018. Reachability analysis and safety verification for neural network control systems. *arXiv preprint arXiv:1805.09944* (2018).
- [37] Weiming Xiang, Patrick Musau, Ayana A Wild, Diego Manzanos Lopez, Nathaniel Hamilton, Xiaodong Yang, Joel Rosenfeld, and Taylor T Johnson. 2018. Verification for machine learning, autonomy, and neural networks survey. *arXiv preprint arXiv:1810.01989* (2018).
- [38] Jing Xu, Zhimin Hou, Wei Wang, Bohao Xu, Kuangen Zhang, and Ken Chen. 2018. Feedback deep deterministic policy gradient with fuzzy reward for robotic multiple peg-in-hole assembly tasks. *IEEE Transactions on Industrial Informatics* 15, 3 (2018), 1658–1667.
- [39] Xin Xu, Hong Chen, Chuanqiang Lian, and Dazi Li. 2018. Learning-based predictive control for discrete-time nonlinear systems with stochastic disturbances. *IEEE transactions on neural networks and learning systems* 29, 12 (2018), 6202–6213.
- [40] Bai Xue and Naijun Zhan. 2018. Robust invariant sets computation for switched discrete-time polynomial systems. *arXiv preprint arXiv:1811.11454* (2018).
- [41] Zhengfeng Yang, Chao Huang, Xin Chen, Wang Lin, and Zhiming Liu. 2016. A linear programming relaxation based approach for generating barrier certificates of hybrid systems. In *International Symposium on Formal Methods*. Springer, 721–738.