Safe Controller Synthesis for Nonlinear Systems via Reinforcement Learning and PAC Approximation

Xia Zeng[1], Banglong Liu[2], Zhenbing Zeng[3], Zhiming Liu[1], Zhengfeng Yang[2*]
[1]Southwest University, [2]East China Normal University, [3]Shanghai University

ABSTRACT

Controller synthesis for nonlinear systems is an important research issue. Deep Neural Network (DNN) control policies obtained through reinforcement learning (RL), though exhibiting good performance in simulations, cannot be applied to safety-critical systems for lack of formal guarantee. To address this, this paper considers fully utilizing the advantages of RL for complex control tasks to obtain a well-performing DNN controller. Then, using PAC (Probably Approximately Correct) techniques, a polynomial surrogate controller with probabilistically controllable approximation error is obtained. Finally, the safety of the control system under the designed polynomial controller is verified using barrier certificate generation. Experiments demonstrate the effectiveness of our method in generating controllers with safety guarantees for systems with high dimensions and degrees.

KEYWORDS

Formal verification, Controller synthesis, Reinforcement learning, Probably Approximately Correct, Barrier certificate

ACM Reference Format:
Xia Zeng[1], Banglong Liu[2], Zhenbing Zeng[3], Zhiming Liu[1], Zhengfeng Yang[2]. 2024. Safe Controller Synthesis for Nonlinear Systems via Reinforcement Learning and PAC Approximation. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3657332

1 INTRODUCTION

Control and decision-making software design for autonomous systems, encompassing unmanned aerial vehicles, ground vehicles, and robotics, is a pivotal area in industrial applications [8]. The synthesis of controllers for ensuring the safety of system behaviors is a significant research issue in applications involving safety-critical aspects. A very promising approach at present is the learning-based controller synthesis targeting the optimization of user-defined cost/reward functions. And a typical way is to use the framework of reinforcement learning (RL) which evaluates and improves the controller's performance by interacting with the environments [15, 16]. Currently, the method of generating control policies based on RL for specific needs has exhibited good performance in many control tasks, particularly in complex, high-dimensional control systems. However, formal reasoning of the required properties of such DNN-controlled dynamical systems is a big challenge which makes the practical use of RL in safety-critical systems still limited.

The challenges of performing safety verification of learned control systems are primarily evident in two aspects: firstly, no specific network structure has been found universally effective for different control tasks, and thus there is no unified systematic solution for conducting safety verification of control systems under neural network (NN) strategies, especially for infinite time settings, and existing works often devise distinct verification strategies for NNs with specific activation functions [9, 19]; secondly, for complex control systems, such as high-dimensional ones, achieving satisfactory control performance requires the design of controllers with complex network structures, which poses significant challenges for mainstream verification methods based on Satisfiability Modulo Theories (SMT) solvers [6, 9, 18]. In summary, existing mainstream methods for formal verification of NN-controlled systems are significantly limited in the range and scale of problems they can address.

The central aim of this paper is to leverage the advantages of learning to synthesize controllers for ease of safety verification. To this end, we have designed a framework that integrates RL-guided learning with PAC model approximation and Barrier Certificate (BC) synthesis, ultimately encoding the problem into a numerically solvable polynomial constraint problem. Firstly, we make the most of RL methods focusing on safety control needs, and the well-trained NNs from this step will be used to design futher controllers. Secondly, we develop a method guided by PAC (Probably Approximately Correct) principles [3] for generating polynomial controllers that are easy to verify. This method surpasses the commonly used Least Squares (LS) approaches by probabilistically characterizing approximation errors and more systematically guiding the setting of polynomial templates to achieve controllable error margins, resulting in controllers with the lowest possible degree, which is advantageous for subsequent verification. Finally, we employ the notion of BC to transform the safety verification of the control system into to a polynomial-constraint-solving problem which can be solved efficiently. Compared to methods based on SMT, our approach is more effective and capable of handling synthesis and verification for larger-scale systems up to dimension 12.

To summarize, the main contributions of this paper are:

- We establish an innovative method to construct a verifiable controller for nonlinear systems with RL-guidance and PAC-model based approximation.
- We develop a systematic PAC-based algorithm to approximate DNNs with controllable error, effectively balancing approximation precision and safety verification efficiency.
- We fulfill the formal verification for nonlinear systems with our designed controller over an unbounded time horizon.
- We conduct a thorough experimental evaluation to demonstrate the efficiency and practicality of our approach.

*Corresponding author

2 PRELIMINARIES

Notations. Let $\mathbb{R}$ be the field of real numbers. $\mathbb{R}[\mathbf{x}]$ denotes the ring of polynomials with coefficients in $\mathbb{R}$ over n-dimensional variables $\mathbf{x} = [x_1, x_1, \ldots, x_n]^T$, and $\mathbb{R}[\mathbf{x}]^n$ denotes the n-dimensional polynomial vector. $\Sigma[\mathbf{x}] \subset \mathbb{R}[\mathbf{x}]$ denotes the space of SOS polynomials.

2.1 Safety Control Problem

This section formulates the safe controller synthesis problem. A controlled continuous dynamical system is modeled by first-order ordinary differential equations

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{u} = \pi(\mathbf{x}) \end{cases}, \tag{1}$$

where $\mathbf{x} \in D \subseteq \mathbb{R}^n$ is the system state, $\mathbf{u} \in U \subseteq \mathbb{R}^m$ is the control input, and $\mathbf{f} \in \mathbb{R}[\mathbf{x}]^n$ is the vector field defined on the state space $D \subseteq \mathbb{R}^n$. Assume $\mathbf{f}$ satisfies the local Lipschitz condition, which ensures (1) has a unique solution $\mathbf{x}(t, \mathbf{x}_0)$ in D for every initial state $\mathbf{x}_0 \in D$ at $t = 0$.

In many contexts, a dynamical system is equipped with a domain $\Psi \subset D$ and an initial set $\Theta \subset \Psi$, represented as a triple $C \doteq (\mathbf{f}, \Psi, \Theta)$. Given a prespecified unsafe region $X_u \subset D$, we say that the system C is *safe* if all system trajectories starting from Θ can not evolve into any state specified by X_u, which has been widely investigated in safety critical applications.

Definition 1 (Safety). *For a controlled constrained continuous dynamical system (CCDS) $C = (\mathbf{f}, \Psi, \Theta)$ and a given unsafe region X_u, the system is safe if for all $\mathbf{x}_0 \in \Theta$, there does not exist $t_1 > 0$ s.t. $\forall t \in [0, t_1).\mathbf{x}(t, \mathbf{x}_0) \in \Psi$ and $\mathbf{x}(t_1, \mathbf{x}_0) \in X_u$.*

Definition 2 (Safe Controller Synthesis). *Given a controlled CCDS $C = (\mathbf{f}, \Psi, \Theta)$ with $\mathbf{f}$ defined by (1) with an unsafe set X_u, design a locally Lipschitz continuous feedback control law π such that the closed-loop system C with $\mathbf{f} = \mathbf{f}(\mathbf{x}, \pi(\mathbf{x}))$ is safe as per Definition 1.*

The concept of *barrier certificates* (BCs) plays an important role in safety verification of continuous systems. The essential idea is to use the zero level set of a function $B(\mathbf{x})$ as a barrier to separate all the reachable states from the unsafe region. The following result can be used to guarantee the safety of a given controlled CCDS.

Theorem 1. [17] *Given a controlled CCDS $C = (\mathbf{f}, \Psi, \Theta)$, with $\mathbf{f}$ defined by (1), a feedback control law $\mathbf{u} = \pi(\mathbf{x})$, and the unsafe region X_u, if there exists a real-valued function $B : \Psi \to \mathbb{R}$ satisfying:*

(i) $B(\mathbf{x}) \geq 0 \quad \forall \mathbf{x} \in \Theta$,

(ii) $B(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in X_u$,

(iii) $B(\mathbf{x}) = 0 \Rightarrow \mathcal{L}_\mathbf{f} B(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Psi$,

where $\mathcal{L}_f B(\mathbf{x})$ denotes the Lie-derivative of $B(\mathbf{x})$ along the vector field $\mathbf{f}(\mathbf{x})$, i.e., $\mathcal{L}_\mathbf{f} B(\mathbf{x}) = \sum_{i=1}^n \frac{\partial B(\mathbf{x})}{\partial x_i} \cdot f_i(\mathbf{x})$, then $B(\mathbf{x})$ is a barrier certificate (BC) for the closed-loop system C with the control law $\pi(\mathbf{x})$, and the safety of system C is guaranteed.

Throughout this paper, we assume that the initial set Θ, the domain Ψ and the unsafe set X_u are compact semi-algebraic sets, represented by polynomial equations and inequalities, i.e.,

$$\begin{cases} \Theta := \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \geq 0, 1 \leq i \leq m_1\}, \\ \Psi := \{\mathbf{x} \in \mathbb{R}^n \mid h_j(\mathbf{x}) \geq 0, 1 \leq j \leq m_2\}, \\ X_u := \{\mathbf{x} \in \mathbb{R}^n \mid q_k(\mathbf{x}) \geq 0, 1 \leq k \leq m_3\}, \end{cases}$$

for some polynomials $g_i, h_j, q_k \in \mathbb{R}[\mathbf{x}]$.

2.2 Scenario Optimization

Scenario optimization is well utilized in robust control design to address a category of optimization problems statistically, focusing on a finite, randomly sampled subset from infinitely many convex constraints [3, 4, 13].

We consider the following optimization problem,

$$\begin{aligned} \min_{\mathbf{y} \in \Gamma \subseteq \mathbb{R}^\kappa} \quad & \mathbf{b}^\top \mathbf{y} \\ s.t. \quad & f_\omega(\mathbf{y}) \leq 0, \ \ \forall \omega \in \Omega, \end{aligned} \tag{2}$$

where f_ω is a convex and continuous function of κ-dimensional optimization variable $\mathbf{y}$ for evey $\omega \in \Omega$, and both Ω and Γ are convex and closed. Theoretically, solving (2) is a challenge due to the presence of infinitely many constraints. Calafiore et al. introduced a scenario approach to address problem (2), providing a solution with a Probably Approximately Correct (PAC) guarantee in [3].

Definition 3. *Let $\mathbb{P}$ be a probability measure on Ω. The scenario approach to handle (2) is to solve the following problem:*

$$\begin{aligned} \min_{\mathbf{y} \in \Gamma \subseteq \mathbb{R}^\kappa} \quad & \mathbf{b}^\top \mathbf{y} \\ s.t. \quad & \bigwedge_{i=1}^{K} f_{\omega_i}(\mathbf{y}) \leq 0, \end{aligned} \tag{3}$$

where $(\omega_i)_{i=1}^K$ are K independent and identically distributed (i.i.d.) samples extracted from Ω according to $\mathbb{P}$.

The scenario approach relaxes the infinitely many constrains in (2) by only considering a finite subset of K constrains. Meanwhile, a PAC guarantee depending on K, between the scenario solution in (3) and its original solution in (2) is proved in [3], which is further improved by [4] in reducing the number of samples K. Specifically, the following theorem establishes a condition on K for (3) which assures that its solution satisfies the constrains in (2) statistically.

Theorem 2. *If (3) is feasible and has an optimal solution $\mathbf{y}_K^*$, and*

$$\epsilon \geq \frac{2}{K}(\ln \frac{1}{\eta} + \kappa),$$

where ϵ and η are the pre-defined error rate and the significance level, respectively, then with confidence at least $1 - \eta$, the optimal $\mathbf{y}_K^*$ satisfies all the constrains in Ω but only at most a fraction of probability measure ϵ, i.e., $\mathbb{P}(f_\omega(\mathbf{y}_K^*) > 0) \leq \epsilon$.

Assumption 1. *In this paper, we set $\mathbb{P}$ to be the uniform distribution on Ω in (3).*

3 PAC-BASED VERIFIABLE CONTROLLER SYNTHESIS WITH RL-GUIDANCE

In this section we design an easy-to-verify control policy with the aid of reinforcement learning (RL) and scenario optimization. Using RL to develop controllers is a promising approach, especially for complex, high-dimensional control systems. Typically, the NN controllers trained through RL for different control tasks tend to have complex structures, making the verification (in unbounded time) of the system under these control strategies a challenge. This issue is also a bottleneck preventing the use of learned controllers in safety-critical situations.

In response, we take advantage of the strengths of learning control strategies by using well-performing controllers as an aid. We design a method for generating polynomial-type controllers guided by Probably Approximately Correct (PAC) principles. This method has advantages over traditional Least Squares (LS) methods, as it can probabilistically characterize approximation errors and guide the setting of polynomial templates to achieve controlled approximation effects. Furthermore, utilizing the formal characteristics of the controllers obtained by this method, we develop a numerical method for polynomial constraint solving for subsequent safety verification of the control system. This approach is more effective than verification methods based on Satisfiability Modulo Theories (SMT) and can handle safety verification for larger-scale systems.

3.1 Training well-performing DNN controllers using reinforcement learning

In this subsection, we illustrate the training process of the DNN controller using RL. Here, we focus on how to train a well-performing controller by RL, which means that the controller could lead the system to avoid the obstacles within the simulation time bound.

Reinforcement learning is an unsupervised learning method, and training with RL requires the completion of four components: the state space, the action space, the environment, and the reward function, respectively. For a control task, we take the system state $\mathbf{x}$ which is a vector of n dimensions as the state, the output of the controller u which is of dimension m as the action, and the system dynamics as the environment. Furthermore, the design of the reward function for the safety requirement is explained as follows.

We construct the reward function through encoding the desired behaviours of the closed-loop system under the DNN controller, which assures unsafe region avoidance.

We hope that the RL helps to synthesize an ideal controller by the designed reward, and all the trajectories of the closed-loop system starting from the initial set Θ cannot evolve into the unsafe region X_u under the trained DNN controller. Thus, the reward function is preliminarily defined as

$$\hat{r}_t = \beta_1 \cdot \mathrm{dist}(X_u, \mathbf{x}_t)$$

where $\beta_1 > 0$ is the scale factor, and $\mathrm{dist}(X_u, \mathbf{x}_t)$ denotes the distance between the state $\mathbf{x}_t$ and the unsafe region X_u.

In addition, once the trajectory approaches and nearly hits the boundary of the unsafe set, the system behavior should be penalized. For this purpose, the reward function is updated as

$$r_t = \begin{cases} \hat{r}(\mathbf{x}_t) - \min(\beta_2 \cdot \frac{1}{\mathrm{dist}(X_u, \mathbf{x}_t)}, \Delta r_{\min}), & \mathbf{x} \in S_{belt} \\ \hat{r}(\mathbf{x}_t), & \text{otherwise} \end{cases} \tag{4}$$

where $S_{belt} = \{\mathbf{x} \mid \mathrm{dist}(X_u, \mathbf{x}) < \delta\}$, δ is a small positive value, $\beta_2 > 0$ is the scale factor and $\Delta r_{\min} > 0$ is the threshold avoiding too large fluctuations of reward value. In this work, we set $\beta_1 = 1.0$, $\beta_2 = 5.0$, $\delta = 0.1$. The setting r_t of (4) can be kept within a certain range, making the convergence effect better.

The remaining problem is to train the controller via RL. Here we use Deep Deterministic Policy Gradient (DDPG) [14] which is a popular RL approach suited for continuous control applications. The DDPG algorithm combines the value-based and policy-based methods, and is made up of two neural networks: the critic network and actor network. Concretely, the critic network $Q : \mathbb{R}^{n+m} \to \mathbb{R}$ takes the state $\mathbf{x}_t$ and the action $\mathbf{u}_t$ as input, and outputs a scalar Q-value $Q(\mathbf{x}_t, \mathbf{u}_t | \theta^Q)$, which evaluates the current action at the given state $\mathbf{x}_t$ and updates the parameter θ^Q by minimizing the L^Q as following

$$L^Q = \frac{1}{N} \sum_t (y_t - Q(\mathbf{x}_t, \mathbf{u}_t | \theta^Q))^2, \tag{5}$$

where $y_t = reward(\mathbf{x}_t) + \gamma Q'(\mathbf{x}_{t+1}, \mathbf{u}'_{RL}(\mathbf{x}_{t+1}))$. The actor network $\mathbf{u}_{RL}$: $\mathbb{R}^n \to \mathbb{R}^m$ receives a state $\mathbf{x}_t$ at time step t, and directly outputs a continuous action $\mathbf{u}_t = \mathbf{u}_{RL}(\mathbf{x}_t | \theta^A)$, and updates the parameter θ^A by minimizing the J as following

$$J = -\frac{1}{N} \sum_t Q(\mathbf{x}_t, \mathbf{u}_{RL}(\mathbf{x}_t | \theta^A)), \tag{6}$$

where the Q' and $\mathbf{u}'_{RL}$ are both target networks and have the same structure as Q and $\mathbf{u}_{RL}$ respectively. In DDPG, they do not update parameters but slowly synchronize with the original network. The $\gamma \in (0, 1)$ is the reward decay factor.

To train the desired controller, we first generate a set of initial states from Θ. For each sampled initial state $\mathbf{x}_0$, with the help of $\mathbf{u}_{RL}$, one may yield the associated trajectory as a discrete time state sequence $\{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_m\}$ which does not enter the unsafe area, and then collect the transition tuples $(\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{u}_t, r_t)$ to form a replay buffer. Every few time steps, a batch size of data is sampled from the replay buffer to update the parameters θ^Q and θ^A, and then the new controller is used to simulate the trajectory to collect new data until the controller behaves well.

3.2 Polynomial controller synthesis based on PAC-model of DNN

Assuming a sophisticated DNN architecture can be employed to obtain a well-performing controller through reinforcement learning. For systems where safety is paramount, it's essential to have formal assurances of such controllers. Nevertheless, ensuring the safety properties for the closed-loop systems controlled by the trained controllers is a formidable task due to the intricate nature of DNNs. As a promising solution, we derive a polynomial as a surrogate controller that closely approximates the trained DNN with remarkable precision. It could then be a suitable candidate controller for the control system to conduct subsequent safety verification through polynomial constraint solving, which is more effective comparing with SMT-based method.

Definition 4 (PAC model). *Let $\pi : \mathbb{R}^n \to \mathbb{R}^m$, $B \subseteq \mathbb{R}^n$ and $\mathbb{P}$ a probability measure on B. Let ϵ, $\eta \in (0, 1)$ be the given error rate and significance level, respectively. Let e be the margin. A function $\hat{\pi} : B \to \mathbb{R}^m$ is a PAC model of g on B w.r.t. η, ϵ and e, denoted by $\hat{\pi} \approx_{\eta,\epsilon,e} \pi$, if*

$$\mathbb{P}(||\hat{\pi} - \pi||_\infty \le e) \ge 1 - \epsilon,$$

with confidence $1 - \eta$.

Assumption 2. *In the rest of the paper, to ensure clarity of expression, we assume that π is a real-valued function, i.e., $m = 1$.*

Based on the trained DNN controller $\mathbf{u}_{RL}(\mathbf{x})$ through RL, we construct an easily verifiable controller with a polynomial form, which is actually a PAC model of $\mathbf{u}_{RL}(\mathbf{x})$. The error bound between the PAC-based polynomial controller and the original auxiliary controller $\mathbf{u}_{RL}(\mathbf{x})$ can be evaluated in a statistical way given the pre-defined error rate and significance level.

We first seek to compute an approximate polynomial $p(\mathbf{x}, \mathbf{c}) \in \mathbb{R}[\mathbf{x}]$ with a pre-assigned degree d. Let $[\mathbf{x}]_d$ be the vector consisting of monomials whose degree is at most d, ordered in the graded lexicographic ordering, i.e.,

$$[\mathbf{x}]_d = (1, x_1, x_2, \cdots, x_n, x_1^2, x_1x_2, \cdots, x_1x^{d-1}, x^d)^\top.$$

Let the dimension of $[\mathbf{x}]_d$ be ν, i.e., $\nu = \dim([\mathbf{x}]_d) = \binom{n+d}{d}$. Define $\mathbb{N}_d^n$ as $\{\alpha = (\alpha_1, \cdots, \alpha_n) \in \mathbb{N}^n \mid \sum_1^n \alpha_i \le d\}$ and the monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$. Thus, $p(\mathbf{x}, \mathbf{c})$ can be written as

$$p(\mathbf{x}, \mathbf{c}) = \mathbf{c}^\top [\mathbf{x}]_d = \sum_{\alpha \in \mathbb{N}_d^n} \mathbf{c}_\alpha \mathbf{x}^\alpha.$$

Essentially, we aim to find a polynomial $p(\mathbf{x})$ that approximates as closely as possible to $\mathbf{u}_{RL}(\mathbf{x})$, which amounts to solve

$$\begin{aligned} \min_{\mathbf{c}} \quad & e \\ s.t. \quad & |\mathbf{u}_{RL}(\mathbf{x}) - p(\mathbf{x}, \mathbf{c})| \le e, \quad \forall \mathbf{x} \in \Psi. \end{aligned} \tag{7}$$

However, it is impossible for a polynomial of a specified degree to be an arbitrarily close approximation to a DNN within a specified region. In practice, we seek the polynomial $p(\mathbf{x})$ that approximates $\mathbf{u}_{RL}(\mathbf{x})$ to achieve a high degree of accuracy, that is, with an approximation error less than a tolerable threshold τ. To this end, we synthesize a PAC-model of $\mathbf{u}_{RL}(\mathbf{x})$ in a polynomial form.

For pre-specified ϵ and η, we seek a polynomial $p(\mathbf{x}, \mathbf{c})$ with degree d which is a PAC-model of $\mathbf{u}_{RL}(\mathbf{x})$ by solving the transformed linear programming from (7) using scenario optimization approach, that is, by solving:

$$\begin{aligned} \min_{\mathbf{c}} \quad & e \\ s.t. \quad & |\mathbf{u}_{RL}(\mathbf{x}_i) - p(\mathbf{x}_i, \mathbf{c})| \le e, \quad \forall \mathbf{x}_i \in \widetilde{\Psi}, \end{aligned} \tag{8}$$

where $\widetilde{\Psi}$ is constructed by randomly sampling from Ψ. According to Theorem 2, there are requirements for the number of sampling points to achieve credible approximation error, as given by the following theorem.

Theorem 3. *Let K be the cardinality of $\widetilde{\Psi}$, i.e., $K = |\widetilde{\Psi}|$. For the pre-specified ϵ and η, if K satisfies*

$$\epsilon \ge \frac{2}{K}(\ln \frac{1}{\eta} + \kappa) \tag{9}$$

where $\kappa = \nu + 1$, i.e., $\binom{n+d}{d} + 1$. Then $p(\mathbf{x}, \mathbf{c}^*)$ is a PAC-model of $\mathbf{u}_{RL}(\mathbf{x})$ with error bound e^* as per Definition 4, where $\mathbf{c}^*$ and e^* are the optimal solution of (8).

Therefore, through solving the linear programming (8), we obtain a polynomial $p(\mathbf{x}, \mathbf{c}^*)$ which is an approximation of $\mathbf{u}_{RL}(\mathbf{x})$ with a statistical error bound e^* at a given confidence level. Furthermore, if e^* exceeds the specified empirical threshold τ, the error bound between the approximate polynomial and the auxiliary controller $\mathbf{u}_{RL}(\mathbf{x})$ can be further improved by increasing the degree of the polynomial template and re-solving the problem (8), which increases the dimensionality of the feasible domain for the polynomial coefficients. We propose an algorithm to compute the polynomial PAC-model of DNN controllers as follows.

Algorithm 1 PAC model based polynomial controller synthesis

```
Require: DNN, $\eta$, $max\_degree$, $\tau$
Ensure: $p(\mathbf{x}, \mathbf{c})$, $e$, $\epsilon$
 1: $\epsilon\_list$ = [0.1, 0.01, 0.001, 0.0001]
 2: for $d$ = 1 to $max\_degree$ do
 3:     $error\_list$ = []
 4:     for $\epsilon$ in $\epsilon\_list$ do
 5:         Compute a least $K$ in (9)
 6:         data ← randomly generate $K$ samples from $\Psi$
 7:         $p(\mathbf{x}, \mathbf{c})$, $e$ = fit(data, $d$)
 8:         $error\_list$.append($e$)
 9:         if check($error\_list$) and $e \le \tau$ then
10:             return $p(\mathbf{x}, \mathbf{c})$, $e$, $\epsilon$
11:         end if
12:     end for
13: end for
```

In Algorithm 1, fit(data, d) in Line 7 is to solve the linear programming (8) resulted from the K sampled points; check($error\_list$) in Line 9 is to check whether $|\Delta e|$, the absolute difference between the last two values in *error_list*, is sufficiently small which implies that the approximation error e has converged under the present d, no matter how large K (or how small ϵ) is. In our experiments, we uniformly set η to 10^{-6}, τ to 0.05, *max_degree* to 4, and the criterion bound of $|\Delta e|$ to 0.001.

Algorithm 1 offers a systematic method for approximating the DNN auxiliary controller, producing a verification-oriented controller candidate with the lowest possible degree. Commonly used least squares (LS) approximation, for lacking of guidance in polynomial template setting and quantification of approximation error, could lead to failure of verification due to excessive approximation error or the high degree of the polynomial controller, which increases problem-solving complexity and greatly limits scalability. In contrast, our proposed PAC-based polynomial approximation method effectively quantifies approximation error and, under tolerable error conditions, yields the lowest degree approximate polynomial, substantially increasing the likelihood of successful subsequent verification.

4 SAFETY VERIFICATION WITH BARRIER CERTIFICATES GENERATION

We have elaborated a polynomial controller synthesis method for a controlled CCDS subject to the safety requirement with the aid of reinforcement learning and PAC-model approximation. To ensure the safety property of the control system under the synthesized controller, denoted by $p(\mathbf{x})$, we propose to generate a BC as stated in Theorem 1. To make the computation tractable, the basic idea is to translate the BC generation problem into a polynomial optimization problem which can be solved numerically in an efficent way. Specifically, the Sum-of-Squares (SOS) relaxation technique is applied to encode the BC constraints as an SOS problem involving linear/bilinear matrices inequalities (LMI/BMI).

Assume that the barrier function $B(\mathbf{x})$ is a polynomial of degree d from the vector space of dimension $s = \binom{n+d}{d}$ with the canonical basis $(\mathbf{x}^\alpha)$ of monomials, and that the coefficients of $B(\mathbf{x})$ are unknown, denoted by $\mathbf{b} = (b_\alpha) \in \mathbb{R}^s$. Then write

$$B(\mathbf{x}, \mathbf{b}) = \sum_{\alpha \in \mathbb{N}_d^n} b_\alpha \mathbf{x}^\alpha = \sum_{\alpha \in \mathbb{N}_d^n} b_\alpha \, x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}.$$

As stated in Theorem 1, the controlled CCDS C is safe under the designed controller $p(\mathbf{x})$, if there exists such $\mathbf{b}$ solving the following feasibility problem:

$$\begin{cases} \text{find} \quad \mathbf{b} \\ \text{s.t.} \;\; B(\mathbf{x}, \mathbf{b}) \geq 0, \;\; \forall \mathbf{x} \in \Theta, \\ \qquad \mathcal{L}_\mathbf{f} B(\mathbf{x}, \mathbf{b}) > 0, \;\; \forall \mathbf{x} \in \Psi \text{ and } B(\mathbf{x}, \mathbf{b}) = 0, \\ \qquad B(\mathbf{x}, \mathbf{b}) < 0, \;\; \forall \mathbf{x} \in X_u. \end{cases} \tag{10}$$

Moreover, SOS-relaxation can be applied to encode (10) as an SOS program. Given a basic semi-algebraic set $\mathbb{K}$ defined by:

$$\mathbb{K} = \{\mathbf{x} \in \mathbb{R}^n \mid g_1(\mathbf{x}) \geq 0, \ldots, g_\ell(\mathbf{x}) \geq 0\},$$

where $g_j \in \mathbb{R}[\mathbf{x}]$, $1 \leq j \leq \ell$, a sufficient condition for the nonnegativity of a given polynomial $f(\mathbf{x})$ on the set $\mathbb{K}$ is:

$$f(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sum_{i=1}^{\ell} \sigma_i(\mathbf{x}) g_i, \tag{11}$$

where $\sigma_i \in \Sigma[\mathbf{x}]$, $1 \leq i \leq \ell$. It is obvious that (11) ensures the nonnegativity of $f(\mathbf{x})$ on $\mathbb{K}$.

Thus, the problem (10) can be transformed into the following optimization problem through SOS relaxation:

$$\begin{cases} \text{find} \quad \mathbf{b} \\ \text{s.t.} \;\; B(\mathbf{x}, \mathbf{b}) - \sum_i \sigma_i(\mathbf{x}) g_i(\mathbf{x}) \in \Sigma[\mathbf{x}], \\ \qquad \mathcal{L}_\mathbf{f} B(\mathbf{x}, \mathbf{b}) - \lambda(\mathbf{x}) B(\mathbf{x}, \mathbf{b}) - \sum_j \phi_j(\mathbf{x}) h_j(\mathbf{x}) - \rho \in \Sigma[\mathbf{x}], \\ \qquad -B(\mathbf{x}, \mathbf{b}) - \rho' - \sum_k \xi_k(\mathbf{x}) q_k(\mathbf{x}) \in \Sigma[\mathbf{x}], \end{cases} \tag{12}$$

where $\rho, \rho' > 0$, $\lambda(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$, and $\sigma_i(\mathbf{x}), \phi_j(\mathbf{x}), \xi_k(\mathbf{x}) \in \Sigma[\mathbf{x}]$. Note that ρ, ρ' are needed to ensure positivity or negativity as required in the second and third constraints in (10). Clearly, the feasibility of (12) is sufficient to imply the feasibility of (10).

Investigating (12), the product of undetermined coefficient parameters from $\lambda(\mathbf{x})$ and $B(\mathbf{x}, \mathbf{b})$ in the second constraint makes the problem a BMI problem, which can be carried out by calling a Matlab package PENBMI solver [12]. We also tried initially generating a random constant (or a random linear function) for $\lambda(\mathbf{x})$ when solving (12), which then becomes a more easily solvable LMI problem.

In summary, the safety verification problem is transformed into a BMI/LMI problem (12) for the parameters $\mathbf{b}$. After solving and obtaining the solution $\mathbf{b}^*$ to (12), a barrier certificate $B(\mathbf{x}, \mathbf{b}^*)$ is yielded, which means that the closed-loop system under the designed polynomial controller $p(\mathbf{x})$ is safe.

5 EXPERIMENTS

In this section, we first present a nonlinear system to illustrate our approach, and then conduct and report an experimental evaluation of our method over a set of benchmark examples. In addition, we also conduct a comparison with a mainstream method of safe controller synthesis. All experiments are conducted on a 2.5GHz AMD Ryzen 9 7945HX CPU under Windows 11 with 32GB RAM.

Example 1. *We consider the Pendulum system [10]:*

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -0.056x_1^5 + 1.56x_1^3 - 9.875x_1 - 0.1x_2 + u \end{bmatrix}.$$

Table 1: The running of Algorithm 1 on Example 1

d	η	ϵ	K	e	Δe	τ
1	1e-6	0.0001	356311	0.150963	6e-5	0.05
2	1e-6	0.001	41632	0.065265	6e-4	0.05
3	1e-6	0.001	49632	0.029328	7e-4	0.05

The system domain is $\Psi = \{\mathbf{x} = (x_1, x_2)^T \in \mathbb{R}^2 \mid -\pi \leq x_1 \leq \pi, -5 \leq x_2 \leq 5\}$. *We aim to design a control law* $u = p(\mathbf{x})$ *such that all trajectories of the closed-loop system under* u *starting from the initial set* $\Theta = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 2.2^2\}$ *will never enter the unsafe region* $X_u = \{\mathbf{x} \in \Psi \mid x_1^2 + x_2^2 \geq 2.5^2\}$.

We initially adopt the RL-training method proposed in this paper to obtain a DNN auxiliary controller with a structure of 5 layers, each containing 20 neurons, which exhibits good control performance during the testing phase. During the PAC approximation phase, we utilize Algorithm 1 with the confidence parameter η *set to* 10^{-6} *and* τ *set to* 0.05. *The running of Algorithm 1 on this example has been exhibited in Table 1. When the polynomial degree is elevated to* 3 *in the third iteration, the final error* e *converges to 0.0293 at* $\epsilon = 0.001$. *Thus, the eventually obtained polynomial* $p(\mathbf{x})$ *has a* 99.9% *probability of achieving an approximation error within* 0.0293 *compared to the learnt DNN control policy, with a confidence level of* $1 - 10^{-6}$. *This makes* $p(\mathbf{x})$ *a strongly potential safe controller candidate. Furthermore, by using the verification method based on BC generation introduced in Section 4, we have obtained a BC of degree* 4 *with a time cost of* $T_p(s) = 2.871s$ *(see the* C_1 *case in Table 2).*

We present a detailed experimental evaluation on a set of benchmarks in Table 2, where our method is named 'Poly.controller'. The origins of these 10 widely used cases are provided in the first column; $n_\mathbf{x}$ and $d_\mathbf{f}$ denote the number of state variables and the maximal degree of the polynomials (or the polynomial abstractions for non-polynomial systems) in the vector fields, and we can see that the examples are with dimension up to 12 and some have a high degree of 5; 'DNN *Struc.*' denotes the network structure of the DNN controller synthesized by RL. Here, all DNNs are with ReLU activation functions except for tanh on the output layer.

As shown in Example 1, we use the PAC approximation method to obtain a suitable polynomial controller, and the relevant parameters ϵ, η, K, e of the PAC model obtained from Algorithm 1 are listed in the corresponding columns of Table 2; d_p and d_B denote the degrees of the computed polynomial controllers $p(\mathbf{x})$ and the barrier certificates $B(\mathbf{x})$ for verification, respectively; T_p denotes the verification time cost.

Table 2 also shows the performance comparison of our method with the safe controller synthesis method 'nncontroller', which uses supervised learning and SMT verification [18]. We successfully handled all systems with dimensions up to 12 and degrees up to 5. The 'nncontroller' method, however, could only generate controllers and pass safety verification for three of these systems, i.e. C_1, C_2, and C_3, with the highest dimension being 3. For examples both methods succeed in, comparing $T_p(s)$ and $T_n(s)$, it is evident that our method required significantly less time for verification compared to the 'nncontroller' method.

The scalability and efficiency of our method can be explained as follows: 1) The verification process for the designed controllers can

Table 2: Performance evaluation of our method

Benchs.	$n_{\mathbf{x}}$	$d_{\mathbf{f}}$	DNN *Struc.*	*Poly.controller*							*nncontroller*	
				ϵ	η	K	e	d_p	d_B	$T_p(s)$	BC *Struc.*	$T_n(s)$
C_1[10]	2	5	2-20(4)-1	0.001	10^{-6}	49632	0.0293	3	4	2.871	2-30-1	32.507
C_2[18]	2	5	2-30(5)-1	0.0001	10^{-6}	416311	0.0388	2	4	2.304	2-30-1	20.492
C_3[6]	3	2	3-30(5)-1	0.01	10^{-6}	4964	0.0157	2	4	8.154	3-30-1	45.56
C_4[5]	4	3	4-30(5)-1	0.0001	10^{-6}	396311	0.0221	1	2	3.477	×	×
C_5[1]	5	2	5-30(5)-1	0.01	10^{-6}	4164	0.0132	1	2	7.011	×	×
C_6[2]	6	3	6-30(5)-1	0.01	10^{-6}	4364	3.57e-5	1	2	1.102	×	×
C_7[11]	7	2	7-30(5)-1	0.001	10^{-6}	101632	0.0294	2	2	0.476	×	×
C_8[11]	9	2	9-30(5)-1	0.001	10^{-6}	49632	0.0255	1	2	0.814	×	×
C_9[11]	9	2	9-30(5)-1	0.01	10^{-6}	13964	0.00273	2	4	37.90	×	×
C_{10}[7]	12	1	12-30(5)-1	0.01	10^{-6}	5564	0.0006	1	2	6.786	×	×

utilize numerical computation, thereby avoiding the complexities of SMT solvers; 2) Furthermore, our method based on PAC approximation captures the beneficial control performance derived from learning, while achieving a lower-degree polynomial controller and ensuring minimal approximation error, and thus offers advantages in computational efficiency and broader possibilities for BC selection in subsequent verification.

To summarize, Table 2 shows that on a set of commonly used benchmark examples, controllers can all be synthesized and verified effectively, which demonstrates that our RL-PAC-based safe controller synthesis method is quite promising.

6 CONCLUSION

This paper introduces a methodology for designing controllers aimed at safety verification for nonlinear systems, based on RL guidance and PAC polynomial approximation. The method leverages the advantages of RL in learning safe control strategies for large-scale problems, and fully retaining the flexible control capabilities brought about by the structural diversity and complexity of neural network controllers. Using the well-performing DNN controller as an auxiliary, we propose a systematic PAC polynomial controller synthesis algorithm using scenario optimization. This method effectively balances approximation accuracy with subsequent verification efficiency, producing a candidate polynomial controller that is effectively verifiable and performs well. Finally, an effective safety verification method is provided, tailored to the characteristics of our designed polynomial controller, by effectively transforming the verification problem into a polynomial contraint solving problem, thus breaking away from the traditional dependence on SMT solvers and their limitations in handling large-scale problems. The experiments demonstrate the effectiveness of our method for examples with high dimensions and degrees.

7 ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Project, China under Grant 2022YFA1005101, in part by the National Natural Science Foundation of China under Grants (No. 62272397, No. 12171159, No. 62032019), and “Digital Silk Road” Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant No. 22510750100.

REFERENCES

[1] M. Amin, B. Sassi, and S. Sankaranarayanan. 2015. Stabilization of polynomial dynamical systems using linear programming based on Bernstein polynomials. arXiv:1501.04578

[2] O. Bouissou, A. Chapoutot, A. Djaballah, and M. Kieffer. 2014. Computation of parametric barrier functions for dynamical systems using interval analysis. In *53rd IEEE Conference on Decision and Control*. 753–758.

[3] G. Calafiore and M.Campi. 2006. The scenario approach to robust control design. *IEEE Transactions on automatic control* 51, 5 (2006), 742–753.

[4] M. Campi, S. Garatti, and M. Prandini. 2009. The scenario approach for systems and control design. *Annual Reviews in Control* 33, 2 (2009), 149–157.

[5] G. Chesi. 2004. Computing output feedback controllers to enlarge the domain of attraction in polynomial systems. *IEEE Trans. Automat. Control* 49, 10 (2004), 1846–1853.

[6] J. V. Deshmukh, J. P. Kapinski, T. Yamaguchi, and D. Prokhorov. 2019. Learning Deep Neural Network Controllers for Dynamical Systems with Safety Guarantees: Invited Paper. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[7] Gao, Sicun. 2016. Quadcopter model. https://github.com/dreal/benchmarks.

[8] Z. Huang, Y. Wang, S. Mitra, G. Dullerud, and S. Chaudhuri. 2015. Controller synthesis with inductive proofs for piecewise linear systems: An smt-based algorithm. In *2015 54th IEEE conference on decision and control (CDC)*. IEEE, 7434–7439.

[9] R. Ivanov, J. Weimer, R. Alur, G. Pappas, and I. Lee. 2019. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 169–178.

[10] W. Jin, Z. Wang, Z. Yang, and S. Mou. 2020. Neural Certificates for Safe Control Policies. (2020).

[11] E. Klipp, R. Herwig, A. Kowald, C. Wierling, and H. Lehrach. 2005. *Systems biology in practice: concepts, implementation and application*. John Wiley & Sons.

[12] M. Kočvara and M. Stingl. 2005. PENBMI User's guide (Version 2.0). (2005).

[13] R. Li, P. Yang, C. Huang, Y. Sun, B. Xue, and L. Zhang. 2022. Towards Practical Robustness Analysis for DNNs Based on PAC-Model Learning. In *Proceedings of the 44th International Conference on Software Engineering (ICSE '22)*. Association for Computing Machinery, 2189–2201.

[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, and et al. 2015. Continuous control with deep reinforcement learning. *Computer science* (2015).

[15] V. Mnih, K. Kavukcuoglu, D. Silver, and et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[16] Junhyuk Oh, Valliappa Chockalingam, Honglak Lee, et al. 2016. Control of memory, active perception, and action in minecraft. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[17] S. Prajna, A. Jadbabaie, and G. J. Pappas. 2007. A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates. *IEEE Trans. Automat. Control* 52, 8 (2007), 1415–1428.

[18] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock. 2021. Learning safe neural network controllers with barrier certificates. *Formal Aspects of Computing* (2021), 1–19.

[19] Q. Zhao, X. Chen, Z. Zhao, Y. Zhang, E. Tang, and X. Li. 2022. Verifying Neural Network Controlled Systems Using Neural Networks. In *Proceedings of the 25th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '22)*. Article 3, 11 pages.