Neural Barrier Certificates Synthesis of NN-Controlled Continuous Systems via Counterexample-Guided Learning

Hanrui Zhao[1], Niuniu Qi[1], Mengxin Ren[1], Xia Zeng[2], Zhenbing Zeng[3], Zhengfeng Yang[1,*]
[1]*Shanghai Key Lab of Trustworthy Computing, East China Normal University, Shanghai, China*
[2]*School of Computer and Information Science, Southwest University, Chongqing, China*
[3]*Department of Mathematics, Shanghai University, Shanghai, China*

ABSTRACT

There is a pressing need to ensure the safety of closed-loop systems with neural network controllers, as they are often incorporated into safety-critical applications. To address this issue, we propose a novel approach for generating barrier certificates, which combines counterexample-guided learning with efficient Sum-Of-Squares (SOS) based verification. By leveraging barrier certificate candidates obtained from the learning phase, our proposed method offers an efficient verification procedure that solves three Linear Matrix Inequality (LMI) constraint feasibility testing problems, instead of relying on an SMT solver to verify the barrier certificate conditions. We conduct comparison experiments on a set of benchmarks, demonstrating the advantages of our method in terms of efficiency and scalability, which enable effective verification of high-dimensional systems.

KEYWORDS

Safety verification, Neural barrier certificate, Controlled continuous dynamical system, Counterexample guidance

1 INTRODUCTION

Safety-critical systems incorporating AI components arise in wide-ranging applications. With the boom of deep learning, considerable research activities have focused on the use of neural networks (NNs) for control of cyber-physical systems such as unmanned aerial vehicles, self-driving cars, etc. To ensure that the controlled system satisfies certain safety requirements, it is necessary to be concerned with safety verification which aims to decide starting from an initial set, whether a system can evolve to some unsafe region in the state space.

Due to their ability to handle nonlinear systems for an infinite time horizon, barrier certificate synthesis is a promising method for safety verification. The notion of barrier certificate (BC) is a real function that maps all reachable states to non-negative real numbers and all the states in the unsafe set to negative reals. One conventional and typical routine is to synthesize polynomial barrier certificates through a polynomial optimization by SOS relaxation [11, 14]. However, the SOS-based methods are limited by the cost of prohibitive computational complexity, making it challenging to handle larger-scale system verification problems.

With the rapid development of artificial intelligence, the learning-based BCs generation method is gradually becoming a new paradigm. As an effective iterative refinement approach, counterexample-guided barrier certificate synthesis is gaining increasing attention. Sha et al. [14] present an iterative barrier certificates synthesis approach that relies on polynomial optimization to yield BC candidates and verify their conditions by using Satisfiability Modulo Theories (SMT) [2]. Peruffo et al. [1] leverage a Counterexample-Guided Inductive Synthesis (CEGIS) [15] procedure, comprising an NN *Learner* for training neural BCs and an SMT-based *Verifier* for formal verification. An effective and efficient SMT solver is a prerequisite for success in exploiting the counterexample-guided method. The SMT solver (e.g., *dReal*) [7] is limited at the cost of high computational complexity, wherein the search process often requires a large number of partitions and iterations, especially for high-dimensional systems.

This paper focus on quickly verifying the BC's conditions to identify the BC candidates yielded from the learning phase. We explore a cross-product neural barrier certificate structure, and present a novel counterexample-guided approach for neural barrier certificate learning of NN-controlled continuous systems. In the verifying phase, we exploit SOS-relaxation to check the BC's conditions. The direct computational method applies SOS relaxation to yield BCs, which results in a bilinear matrix inequality (BMI) solving problem belonging to the class of NP-hard problems. In this paper, benefiting from the BC candidates yielded from the learning phase, we can convert the non-convex BMI problem into a convex LMI feasibility testing problem and divide the large SOS programming into three sub-programming ones. Hence, our proposed method spends much less time in computation and has the potential to find solutions beyond the reach of existing methods.

The main contributions of this paper are summarized as follows:

- We provide a new abstract method using Chebyshev approximation to obtain polynomial inclusions for NN-controllers, and explore counterexample-guided learning approach to yield a neural barrier certificate for the closed-loop system with the abstract controllers.
- The proposed novel verification method that converts the non-convex BMI solving problems, derived from BC generation, into three convex LMI feasibility testing sub-problems.

The authors acknowledge the support of the National Key R&D Program of China (2023YFA1009402), the National Natural Science Foundation of China under Grants (No. 12171159, 62272397), and "Digital Silk Road" Shanghai International Joint Lab of Trustworthy Intelligent Software under Grant (No. 22510750100).
*Corresponding author: Zhengfeng Yang. Email: zfyang@sei.ecnu.edu.cn.

DAC'24, June 23-27, San Francisco, CA

ACM ISBN 979-8-4007-0601-1/24/06
https://doi.org/10.1145/3649329.3658256

The advantage significantly reduces the computation complexity, enabling more efficient handling of higher dimensional examples.
- Our proposed method is compared with three prevalent approaches over a set of benchmarks gathered from the literature, which shows that our method is more effective and provides much better efficiency than existing methods.

2 PRELIMINARIES

Consider a continuous dynamical system of the finite first-order ordinary differential equations:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{1}$$

where $\dot{\mathbf{x}}$ denotes the derivative of $\mathbf{x}$ with respect to the time variable t, and $\mathbf{f} = [f_1, \ldots, f_n]^T$ is the vector field on the state space $D \subset \mathbb{R}^n$. Assume that $\mathbf{f}$ satisfies the local Lipschitz condition, which ensures that given the initial state point $\mathbf{x}_0 \in D$, there exists a time $T > 0$ and a unique function $\tau : [0, T) \mapsto \mathbb{R}^n$ such that $\tau(0) = \mathbf{x}_0$. And $\mathbf{x}(t)$ is called a trajectory of (1) from $\mathbf{x}_0$.

A dynamical system consists of a domain $\Psi \subset D$ and an initial set $\Theta \subset \Psi$, which can be represented as a triple $C : \langle \mathbf{f}, \Theta, \Psi \rangle$. We say the system C is $safe$, if every trajectory of C that starts from the set Θ does not enter the given unsafe region Ξ, namely, for all $\mathbf{x}_0 \in \Theta$, there does not exist $t_1 > 0$ such that

$$\forall t \in [0, t_1].\mathbf{x}(t, \mathbf{x}_0) \in \Psi \quad \text{and } \mathbf{x}(t_1, \mathbf{x}_0) \in \Xi.$$

We consider a controlled continuous dynamical system (CCDS) $C : \langle \mathbf{f}, \Theta, \Psi \rangle$ with continuous dynamics defined by

$$\dot{\mathbf{f}} = \mathbf{f}(\mathbf{x}, u) \ \text{ with } \ u = k(\mathbf{x}), \tag{2}$$

where $u \in U \subset \mathbb{R}^n$ are the control inputs, and $k : \Psi \leftarrow U$ is the feedback controller function.

Several methods based on barrier certificate generation have proven successful in solving safety verification problems, primarily due to their ability to attack the infinite time horizon. A barrier certificate is a real function that assigns non-negative real values to all reachable states and negative real values to states within the unsafe region. The presence of barrier certificates is sufficient to ensure the safety of the given system concerning the unsafe region. In this context, we introduce a relaxed verification condition by incorporating auxiliary polynomials in the generation of barrier certificates, an approach adapted from [12].

Theorem 1. *Given a controlled CCDS $C : \langle \mathbf{f}, \Theta, \Psi \rangle$ with $\mathbf{f}$ defined by (2) and a feedback control law $u = k(\mathbf{x})$, a barrier certificate of C for safety with respect to the unsafe region Ξ is a real-valued function $B : \Psi \to \mathbb{R}$, such that the following conditions hold:*

(i) $B(\mathbf{x}) \geq 0 \quad \forall \mathbf{x} \in \Theta$,
(ii) $B(\mathbf{x}) < 0 \quad \forall \mathbf{x} \in \Xi$,
(iii) $\mathcal{L}_f B(\mathbf{x}) - \lambda(\mathbf{x}) B(\mathbf{x}) > 0 \ \forall \mathbf{x} \in \Psi$,

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

In this paper, we assume that the initial set Θ, the domain Ψ, and the unsafe set Ξ are compact semi-algebraic sets, which are defined by polynomial equations and inequalities. Specifically, the semi-algebraic sets Θ, Ψ, and Ξ are represented in the following manner:

$$\begin{aligned} \Theta : &= \{\mathbf{x} \in \mathbb{R}^n \mid \theta_1(\mathbf{x}) \geq 0, \ldots, \theta_q(\mathbf{x}) \geq 0\}, \\ \Psi : &= \{\mathbf{x} \in \mathbb{R}^n \mid \psi_1(\mathbf{x}) \geq 0, \ldots, \psi_r(\mathbf{x}) \geq 0\}, \\ \Xi : &= \{\mathbf{x} \in \mathbb{R}^n \mid \xi_1(\mathbf{x}) \geq 0, \ldots, \xi_p(\mathbf{x}) \geq 0\}, \end{aligned}$$

where $\theta_i, \psi_j, \xi_k \in \mathbb{R}[\mathbf{x}]$.

3 POLYNOMIAL INCLUSION OF NEURAL NETWORK CONTROLLERS

In this paper, we consider the NN controller with a single output, and the multiple-output cases can be handled in a similar manner. Formally, given a NN controller $k(\mathbf{x})$, we seek to compute an approximate polynomial $h(\mathbf{x}, \mathbf{h}) \in \mathbb{R}[\mathbf{x}]$ with a preassigned degree d. Let $[\mathbf{x}]_d$ be the vector consisting of monomials whose degrees are at most d, ordered in the graded lexicographic ordering, i.e., $[\mathbf{x}]_d = [1, x_1, x_2, \ldots, x_n, x_1^2, x_1 x_2, \ldots, x_{n-1} x_n^{d-1}, x_n^d]^T$. Denote by ν the dimension of $[\mathbf{x}]_d$, that is, $\nu = \dim([\mathbf{x}]_d) = \binom{n+d}{n}$. Denote by $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, and the monomial $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$. Thus, $h(\mathbf{x}, \mathbf{h})$ can be written as $h(\mathbf{x}, \mathbf{h}) = \mathbf{h}^T [\mathbf{x}]_d = \sum_{i=\alpha} h_\alpha \mathbf{x}^\alpha$. At a point $\mathbf{a} \in \mathbb{R}^n$, the approximate error between the controller function $k(\mathbf{x})$ and the polynomial $h(\mathbf{x}, \mathbf{h})$ is $e(\mathbf{a}, \mathbf{h}) = |\, k(\mathbf{a}) - h(\mathbf{a}, \mathbf{h})\,|$. Define the uniform approximate error over the set Ψ by $\rho(\mathbf{h}) = \sup_{\mathbf{x} \in \Psi} |k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})|$. For the purpose of polynomial inclusion, one may try to compute the tightest one. Therefore, our goal is to solve the following optimization problem:

$$\sigma = \inf_{\mathbf{h} \in \mathbb{R}^\nu} \sup_{\mathbf{x} \in \Psi} |k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})|. \tag{3}$$

Obviously, the error function $|k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})|$ can be written as $\max\{k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h}), h(\mathbf{x}, \mathbf{h}) - k(\mathbf{x})\}$, and for each $\mathbf{x} \in \Psi$, $k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})$ and $h(\mathbf{x}, \mathbf{h}) - k(\mathbf{x})$ are affine with respect to $\mathbf{h}$. Therefore, for each $\mathbf{x} \in \Psi$, $|k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})|$ is convex with respect to $\mathbf{h}$. Accordingly, the pointwise supremum function $\rho(\mathbf{h})$ is also convex in $\mathbf{h}$. To compute the polynomial inclusion for the controller $k(\mathbf{x})$, we need to find the verified optimum σ once the minimizer $\mathbf{h}$ of (3) is obtained. That is to say, we actually need to calculate the pair of $\mathbf{h}$ and σ that satisfy $|k(\mathbf{x}) - h(\mathbf{x}, \mathbf{h})| \leq \sigma, \forall \mathbf{x} \in \Psi$.

To solve (3), one needs to deal with an optimization problem involving with the nonlinear activation functions in $k(\mathbf{x})$ and the universal quantifier, which is a hard computation problem with exponential complexity in general. Hence, we propose a relaxation method to compute a suboptimal solution to (3) in the following. Precisely, we relax the problem (3) into a Chebyshev approximation problem, then solve it by linear programming tools, and finally yield the verified optimum by estimating the gap between the optima of (3) and the relaxed linear programming problem.

Let us construct a rectangular mesh M in Ψ with a mesh spacing $s \in \mathbb{R}_{>0}$ (say $s = 0.05$) and mesh points $\mathbf{y}_i, 1 \leq i \leq m$. Then (3) can be relaxed as the following Chebyshev approximation problem:

$$\tilde{\sigma} = \min_{\mathbf{h} \in \mathbb{R}^\nu} \max_{i=1,\ldots m} |h(\mathbf{y}_i, \mathbf{h}) - k(\mathbf{y}_i)|, \tag{4}$$

where $\mathbf{h}$ is variable, $h(\mathbf{y}_i, \mathbf{h})$ are affine in $\mathbf{h}$, $k(\mathbf{y}_1), \ldots, k(\mathbf{y}_m) \in \mathbb{R}$.

By introducing an auxiliary variable t, the above optimization problem (4) can be transformed to an equivalent linear programming problem,

$$\left.\begin{aligned}\tilde{\sigma} = \min\ & t \\ \text{s. t. } & h(\mathbf{y}_i,\mathbf{h}) - t \le k(\mathbf{y}_i), \quad i = 1,\cdots,m, \\ & -h(\mathbf{y}_i,\mathbf{h}) - t \le -k(\mathbf{y}_i), \ i = 1,\cdots,m,\end{aligned}\right\} \tag{5}$$

where the variables are $\mathbf{h}$ and t.

One is able to obtain the optimum $\tilde{\sigma}$ and the minimizer $\tilde{\mathbf{h}}$ of the above linear programming (5) by calling the LP solver. Thus, $\tilde{\sigma}$ and $\tilde{\mathbf{h}}$ satisfy the following inequality.

$$|\, h(\mathbf{y}_i,\tilde{\mathbf{h}}) - k(\mathbf{y}_i)\,| \le \tilde{\sigma},\ i = 1,\ldots,m. \tag{6}$$

Following (6), we know that the error between $k(\mathbf{x})$ and the polynomial $h(\mathbf{x},\tilde{\mathbf{h}})$ is not greater than $\tilde{\sigma}$ for all sampling points.

The following theorem shows the gap between the optima of the original optimization problem (3) and the relaxed linear programming (5).

Theorem 2. *Under the same assumptions as above, and let σ be the optimum of the optimization problem (3), and let $\tilde{\sigma}$ and $\tilde{\mathbf{h}}$ be the optimum and the minimizer of the linear programming (5), respectively. Assume L is a Lipschitz constant of the controller $k(\mathbf{x})$. Then we have $\tilde{\sigma} \le \sigma \le \tilde{\sigma} + \frac{1}{2}sL$.*

Proof. Consider $\mathbf{y}_i \in \Psi, 1 \le i \le m$, it is easy to show that $\tilde{\sigma} \le \sigma$. Following the construction of mesh M in Ψ, for each $\mathbf{x} \in \Psi$ there exists a point $\mathbf{y}_i, 1 \le i \le m$ such that $\|\mathbf{x} - \mathbf{y}_i\| \le \frac{1}{2}s$. And L is the Lipschitz constant of $k(\mathbf{x})$, which yields that

$$|k(\mathbf{x}) - k(\mathbf{y}_i)| \le \frac{1}{2}sL. \tag{7}$$

By combining (6) and (7), we can easily verify that

$$|k(\mathbf{x}) - h(\mathbf{x},\mathbf{h})| \le \tilde{\sigma} + \frac{1}{2}sL,\ \forall \mathbf{x} \in \Psi, \tag{8}$$

which concludes the claim. □

Remark 1. *Suppose M is a rectangular mesh in Ψ with the mesh spacing s. Theorem 2 yields that $\lim_{s\to 0} \tilde{\sigma} = \sigma$, that is, the optimum $\tilde{\sigma}$ of the relaxed linear program (5) will tend to the global optimum σ of the optimization problem (3) when s tends to 0. As a consequence, to obtain a tight polynomial inclusion of $k(\mathbf{x})$, one may construct a rectangular mesh M with a smaller spacing s.*

Now, for the NN controller $k(\mathbf{x})$ over a domain Ψ, we can apply the above method to obtain a polynomial inclusion. Concretely, we first construct a rectangular mesh M over Ψ and choose the sampling points $\mathbf{y}_i$ from M to build the corresponding linear programming problem (5). Denote by $\sigma^* = \tilde{\sigma} + \frac{1}{2}sL$, and we can obtain a polynomial inclusion controller $h(\mathbf{x},\tilde{\mathbf{h}}) + w$ with $w \in [-\sigma^*, \sigma^*]$, where $h(\mathbf{x},\tilde{\mathbf{h}})$ is a polynomial and σ^* is its valid approximate error bound yielded from (8). The estimation of Lipschitz constants L for deep neural networks can be obtained as described in [6].

4 THE FRAMEWORK OF NEURAL BARRIER CERTIFICATE SYNTHESIS

We propose a counterexample-guided iterative framework called *SNBC* for the neural barrier certificate synthesis of NN-controlled system approximated from Section 3. The framework is built upon the inductive loop between a *Learner* and a *Verifier* (cf. Fig. 1). The *Learner* jointly trains a polynomial $B(\mathbf{x})$ with a cross-product function activated NN and an auxiliary multiplier $\lambda(\mathbf{x})$ with a linear NN to fit the requirements for BCs (see 4.1). Then the *Verifier* identifies a real BC among the trained candidates by solving a convex LMI feasibility problem based on the SOS relaxation method (see 4.2). In cases where candidates fail to pass the verification, polynomial optimization techniques will be employed to generate counterexamples (Cexs), which indicate where the barrier conditions are violated and then fed back to the *Learner* for further BC updating (see 4.3).

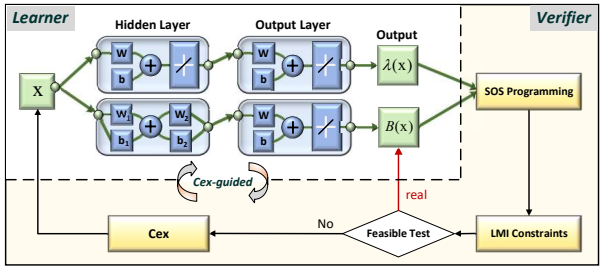

Figure 1: The framework of *SNBC*

4.1 Training of Neural BC Candidates

The *Learner* initializes the neural BC $B(\mathbf{x})$ along with the multiplier network $\lambda(\mathbf{x})$ using hyper-parameters l and n, which denote the NN depth and nodes in each layer. The training sets S_I, S_U and S_D are instantiated by randomly sampled from Θ, Ξ and Ψ, then the two NNs are jointly trained over on the samples and refined whenever counterexamples are added. The choice of activation functions is made with a focus on generating interpretable outputs from the neural network. This means the $B(\mathbf{x})$ NN should produce a scalar polynomial-form expression that can be easily processed by the *Verifier*. While the Square activation function is commonly employed in training polynomial neural networks, its fitting capability is limited due to the restricted output range. To overcome this problem, we introduce a neural network with cross-product activation function as follows:

Quadratic Network Let us consider a feedforward neural network consisting of an input layer $\mathbf{x}^{(0)}$, l hidden layers, and an activation-free output layer $\mathbf{x}^{(l+1)}$. Each hidden layer is characterized by its weight matrix $W^{(l)} \in \mathbb{R}^{n^{(l-1)}\times n^{(l)}}$ and bias vector $b^{(l)} \in \mathbb{R}^{n^{(l)}}$, where n is a positive integer denoting the node number. The structural components are depicted in the following diagram.

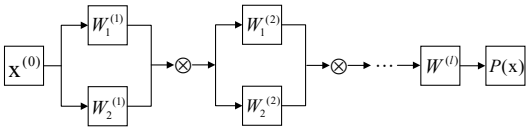

Figure 2: The structure of Quadratic Network

As illustrated in Fig. 2, we introduce two distinct weight matrices, denoted as $W_1^{(l)}$ and $W_2^{(l)}$. Similarly, the bias b also follows this convention. The quadratic network directly produces a scalar polynomial $P(\mathbf{x})$ as the expression for the neural BC, and the degree of $P(\mathbf{x})$ is determined by the number of hidden layers that follow the recursive transformations:

$$\mathbf{x}^{(l)} = (W_1^{(l)}\mathbf{x}^{(l-1)} + b_1^{(l)}) \otimes (W_2^{(l)}\mathbf{x}^{(l-1)} + b_2^{(l)})$$

where $\otimes$ represents the Hadamard (element-wise) product. With each recursion, a quadratic network with l hidden layers will eventually generate a polynomial with degree 2^l. Utilizing the chain rule, the gradient of each layer denotes as $\frac{\partial \mathbf{x}^{(i)}}{\partial \mathbf{x}^{(i-1)}} = \text{diag}[(W_1^{(i)}\mathbf{x}^{(i-1)} + b_1^{(i)})] \cdot W_2^{(i)} + \text{diag}[(W_2^{(i)}\mathbf{x}^{(i-1)} + b_2^{(i)})] \cdot W_1^{(i)}$. Ultimately, we can express the gradient of the output for the quadratic network:

$$\begin{aligned}\nabla P(\mathbf{x}) =&(\prod_{i=1}^{l-1} \text{diag}[(W_1^{(i)}\mathbf{x}^{(i-1)} + b_1^{(i)})] \cdot W_2^{(i)} + \\ &\prod_{i=1}^{l-1} \text{diag}[(W_2^{(i)}\mathbf{x}^{(i-1)} + b_2^{(i)})] \cdot W_1^{(i)}) \cdot W^{(l)}\end{aligned} \tag{9}$$

Compared with the square network with the activation function $\sigma_i = (W^{(i)}\mathbf{x}^{(i-1)} + b^{(i)})^2$, our quadratic network requires twice as many parameters. However, the output polynomial degree remains unchanged. Instead, it enhances the NN fitting capability.

We construct an empirical loss L by measuring the extent of violation for the three existence conditions of the barrier certificate. Through iterative convergence, we gradually reduce this loss to improve the satisfiability of the conditions and update the NN parameters. The training process minimizes the sum of three terms L_D, L_I and L_U, namely

$$\begin{aligned}L =&L_D + L_I + L_U = \frac{\eta_1}{S_D}\sum_{s_d\in\Psi} \max\{\epsilon, -(\mathcal{L}_f B(s_d) - \lambda(s_d))\} \\ &+\frac{\eta_2}{S_I}\sum_{s_i\in\Theta} \max\{\epsilon, -B(s_i)\} + \frac{\eta_3}{S_U}\sum_{s_u\in\Xi} \max\{\epsilon, B(s_u)\}\end{aligned} \tag{10}$$

where s_d, s_i and s_u are samples extracted from the datasets S_D, S_I, and S_U, respectively. In this context, we initially sample from Θ, Ξ, and Ψ with the same size of batch, constructing three equally-sized initial training sets, where $S_I = S_U = S_D$. The constant ϵ serves as a small positive offset to encourage strict inequality satisfaction and enhance the robustness of NN training. Additionally, we employ a continuously differentiable activation function, *Leaky ReLU* [10], to describe $max\{\epsilon, \cdot\}$ in (10). We introduce three positive tuning parameters η_1, η_2 and η_3 to balance the influence of the three sub-loss and compute the loss function in a parallel fashion. Likewise, the generated counterexamples are added to the relevant dataset for re-training the neural networks.

4.2 Identifying Real Barrier Certificate

In the previous section, the *Learner* obtained a candidate that partially satisfies the barrier certificate conditions on a limited dataset. Here, we employ the SOS relaxation method to verify the BC candidates in the entire state space and identify a real BC. To reduce computational complexity, we demonstrate the validity of candidates by formulating the LMI feasibility problem.

According to Theorem 1, in order to confirm $B(\mathbf{x})$ as a real BC, it is necessary to ensure its satisfaction of three valid conditions, which are outlined as follows:

$$\left.\begin{array}{l} B(\mathbf{x}) \geq 0, \ \ \forall \mathbf{x} \in \Theta, \\ B(\mathbf{x}) < 0, \ \ \forall \mathbf{x} \in \Xi, \\ \mathcal{L}_f B(\mathbf{x}) - \lambda(\mathbf{x})B(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in \Psi. \end{array}\right\} \tag{11}$$

All constraints can be expressed as non-negativity constraints for polynomials defined over the corresponding semi-algebraic sets. Consequently, problem (11) can be reformulated as LMI feasibility problem in the following form:

$$\left.\begin{array}{ll} \text{find} & \sigma_{\text{i}}(\mathbf{x}), \delta_{\text{i}}(\mathbf{x}), \phi_{\text{i}}(\mathbf{x}), \lambda(\mathbf{x}) \\ \text{s.t.} & B(\mathbf{x}) - \sum_{i=1}^{q} \sigma_i(\mathbf{x})\theta_i(\mathbf{x}) \in \sum[\mathbf{x}], \\ & -B(\mathbf{x}) - \sum_{i=1}^{p} \delta_i(\mathbf{x})\xi_i(\mathbf{x}) - \epsilon_1 \in \sum[\mathbf{x}], \\ & \mathcal{L}_f B(\mathbf{x}) - \lambda(\mathbf{x})B(\mathbf{x}) - \sum_{i=1}^{r} \phi(\mathbf{x})\psi_i(\mathbf{x}) - \epsilon_2 \in \sum[\mathbf{x}]. \end{array}\right\} \tag{12}$$

where $\epsilon_1, \epsilon_2 \in \mathbb{R}_{>0}$ are pre-specified small positive numbers, $\sigma_i(\mathbf{x})$, $\delta_i(\mathbf{x})$ and $\phi_i(\mathbf{x})$ are SOS polynomials in $\mathbb{R}[\mathbf{x}]$, and $\lambda(\mathbf{x})$ is a polynomial. The decision variables in the SOS programming (12) are the coefficients of all the unknown polynomials, i.e., $\sigma_i(\mathbf{x})$, $\delta_i(\mathbf{x})$, $\phi_i(\mathbf{x})$ and $\lambda(\mathbf{x})$. To make the computation tractable, we simply establish a truncated SOS programming by fixing a priori degree bound $2d$. As a result, the feasibility of (12) confirms that $B(\mathbf{x})$ is a real BC.

As described above, $B(\mathbf{x})$ is a known polynomial yielded from the *Learner*. Then we can equivalently divide (12) into the following three sub-programming problems, and solve them separately:

$$\left\{\begin{array}{ll} \text{find} & \sigma_{\text{i}}(\mathbf{x}) \\ \text{s.t.} & B(\mathbf{x}) - \sum_{i=1}^{q} \sigma_i(\mathbf{x})\theta_i(\mathbf{x}) \in \sum[\mathbf{x}], \end{array}\right. \tag{13}$$

$$\left\{\begin{array}{ll} \text{find} & \delta_{\text{i}}(\mathbf{x}) \\ \text{s.t.} & -B(\mathbf{x}) - \sum_{i=1}^{p} \delta_i(\mathbf{x})\xi_i(\mathbf{x}) - \epsilon_1 \in \sum[\mathbf{x}], \end{array}\right. \tag{14}$$

$$\left\{\begin{array}{ll} \text{find} & \phi_{\text{i}}(\mathbf{x}), \lambda(\mathbf{x}) \\ \text{s.t.} & \mathcal{L}_f B(\mathbf{x}) - \lambda(\mathbf{x})B(\mathbf{x}) - \sum_{i=1}^{r} \phi(\mathbf{x})\psi_i(\mathbf{x}) - \epsilon_2 \in \sum[\mathbf{x}]. \end{array}\right. \tag{15}$$

If all sub-problems (13-15) are feasible, then $B(\mathbf{x})$ is a real BC.

4.3 Generating Counterexamples

If the BC candidates do not pass the verification, then can use polynomial optimization solving techniques to obtain the points that violate the constraints. These points can then be used to create a dataset for a new round of training to update the BC. To achieve this, it is necessary to negate the conditions in equation (11) to capture the counterexample points. Given the continuity and differentiability of the barrier certificate $B(\mathbf{x})$, once one counterexample is identified, there will be multiple counterexamples within a range of γ from it. As each point exhibits different degrees of violation, we can isolate the point $\mathbf{x}^*$ with the most severe violation and generate a set of points randomly within the region defined by $\|\mathbf{x}-\mathbf{x}^*\|_2 \leq \gamma$ provided to the *Learner* for the correction of $B(\mathbf{x})$. This approach effectively reduces the number of guided iterations and accelerates the generation of candidate BC.

Concretely, the construction process of the counterexample set can be divided into two steps: solving for the worst counterexample $\mathbf{x}^*$ and computing the maximum radius γ around $\mathbf{x}^*$. Taking the first negative condition from (11) as an example, our objective is to find the worst-case point $\mathbf{x}^*_\Theta$ that satisfies $B(\mathbf{x}) < 0$ by formulating the following optimization problem:

$$\left\{\begin{array}{ll} \underset{\mathbf{x}}{\text{maximize}} & -B(\mathbf{x}) \\ \text{s.t.} & -B(\mathbf{x}) > 0, \ \ \forall \mathbf{x} \in \Theta \end{array}\right. \tag{16}$$

Next, we can formulate the following optimization problem to construct a maximal counterexample set $\{\mathbf{x} | \|\mathbf{x} - \mathbf{x}^*_\Theta\|_2 \leq \gamma\}$,

$$\left\{\begin{array}{ll} \text{maximize} & \gamma \\ \text{s.t.} & -B(\mathbf{x}) > 0, \ \ \forall \mathbf{x} \in \Theta \\ & \|\mathbf{x} - \mathbf{x}^*_\Theta\|_2 \leq \gamma. \end{array}\right. \tag{17}$$

To solve the problem of finding the worst-case counterexample point for the problem (16), we begin by applying the method of Lagrange multipliers to transform the aforementioned problem into an unconstrained optimization problem. Subsequently, we employ the gradient descent method to iteratively optimize this unconstrained problem, leading to the determination of the local optimum $\mathbf{x}^*$. Additionally, we employ the same methodology to obtain γ, thereby establishing the maximum counterexample set.

A procedure called *SNBC* is summarized in Algorithm 1, and a specification of Algorithm 1 is deferred to Section 4.

Algorithm 1 SNBC: Synthesizing neural barrier certificates for NN-controlled continuous systems

Input: A CCDS $C : \langle \mathbf{f}, \Theta, \Psi \rangle$; an given unsafe set Ξ; a pre-trained NN controller $k(\mathbf{x})$; training set S_I, S_U and S_D
Output: A real barrier certificate $B(\mathbf{x})$

```
Compute the approximation polynomial h(x, h) of k(x)
Initialize the neural networks of B(x) and λ(x)
Get a polynomial BC candidate B̃ and multiplier λ̃ by training B(x) and λ(x) with S_I, S_U, S_D
while Iter do
    Check the validity of candidate B̃ by (13-15)→ bFeasible
    if not bFeasible then
        Calculate the counterexamples Cex according to (11)
        Add Cex to training set S_I, S_U, or S_D
        Update B̃ and λ̃ according to (10)
    else
        return B(x)
    end if
end while
```

5 EXPERIMENTS

In this section, we first demonstrate our tool *SNBC* by applying it to a three-dimensional nonlinear continuous system. The *SNBC* code is available at https://github.com/blliu6/SNBC.

Example 1.[Academic 3D Model] Consider the following controlled continuous dynamical system in the plant:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} z + 8y \\ -y + z \\ -z - x^2 + u \end{bmatrix}. \tag{18}$$

The system domain is defined as Ψ=$\{\mathbf{x}$=$(x, y, z)^T \in \mathbb{R}^3 | -2.2 \leq x, y, z \leq 2.2\}$. Our goal is to confirm that all trajectories of the closed-loop system under the control u=$k(x, y, z)$, starting from the initial set Θ=$\{\mathbf{x}$=$(x, y, z)^T \in \mathbb{R}^3 | -0.4 \leq x, y, z \leq 0.4\}$, will never enter the unsafe region Ξ=$\{\mathbf{x}$=$(x, y, z)^T \in \mathbb{R}^3 | 2 \leq x, y, z \leq 2.2\}$.

We attempt to utilize the deep deterministic policy gradient (DDPG) reinforcement learning algorithm to train a controller u, and approximate it as a polynomial with degree 2. As shown in Fig. 3, after two iterations, we successfully obtain a real barrier certificate as follows:

$$\begin{aligned} B(\mathbf{x}) = {} & 0.159x^2 - 2.267xy + 1.083xz + 2.703x - 0.366y^2 + \\ & 0.126yz + 2.825y + 0.375z^2 + 5.469z - 10.541. \end{aligned} \tag{19}$$

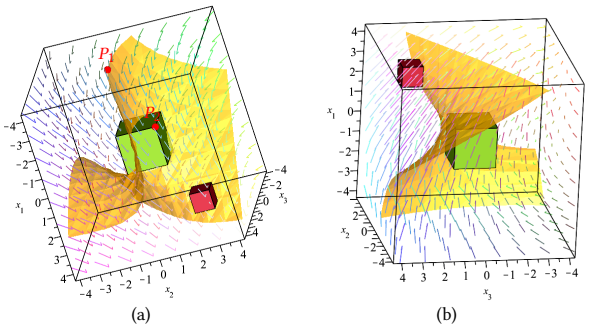

Figure 3: Phase portrait of the system in Example 1. Subfigure 3(a) describes the false BC candidate with the presence of two worst counterexample $\mathbf{x}_D^*$ and $\mathbf{x}_I^*$ (the red points $P_1 = (-4, -5.79e-01, -1.60e-10)$ and $P_2 = (-1, 1, 1)$). Subfigure 3(b) describes the zero level set of real barrier certificate $B(\mathbf{x})$ (the yellow surface) separates Ξ (the red cube) from all trajectories starting from Θ (the green cube).

We compare the proposed approach *SNBC* against two categories of verification tools. The first category includes the tools for combining BC learning with SMT verification, namely *FOSSIL* [1] and *NNCChecker* [14], the second category tool generates barrier certificate based on SDP relaxation, i.e. *SOSTOOLS* [11]. The performance evaluation of above methods is recorded in Table 1. Remark that the tool *NNCChecker* synthesizes polynomial BCs by combining the SOS numerical computation method with *dReal* for formal verification. The *FOSSIL* employs a typical CEGIS procedure, where the SMT-based *Verifier* is used to certify the neural BC or identify counterexamples for falsification.

In Table 1, $n_{\mathbf{x}}$ denotes the number of system variables; $d_{\mathbf{f}}$ denotes the maximal degree of the polynomials appearing in the vector fields, and $NN_{B(\mathbf{x})}$ denotes the network structure of the neural barrier certificate $B(\mathbf{x})$ in *SNBC* and *FOSSIL*. $NN_{\lambda(\mathbf{x})}$ in the column *SNBC* denotes the neural network of multiplier $\lambda(\mathbf{x})$, where c indicates that the multiplier is a constant. For example, the synthesized neural BC for C_9 has one hidden layer with 10 neurons and the multiplier NN has two hidden layers, each with 5 neurons. Here, all NNs have one output layer with 1 neuron; all d_B denotes the degree of the generated barrier certificate; I_s, I_f and I_n are the number of iterations; T_c denotes the computation time for counterexamples generation. $T_l(S)$, $T_l(F)$, and $T_l(N)$ represent the time required for learning BCs; $T_v(S)$, $T_v(F)$, and $T_v(N)$ are the time cost for verification. And T_e denotes the total amount of time spent. *OT* means failing to yield a barrier certificate after 7200 seconds of computation. The symbol $\times$ means that unable to return a feasible solution with the degree bounds $\deg(B) \leq 6$.

Table 1 shows that for all 14 examples, our *SNBC* successfully handle all of them, while *FOSSIL* finds 8 barrier certificates and *NNCChecker* finds 9 ones. *SNBC* seems to provide the best verifying capability among the verification tools with SMT solver. We compare the efficiency of the three methods with the time spent in generating barrier certificates. For the 8 examples that are treated by *SNBC*, *FOSSIL* and *NNCChecker*, on average, *FOSSIL* takes 872.22 seconds to find one barrier certificate and *NNCChecker* takes 24.24 seconds. While *SNBC* only needs 0.946 seconds, which is 922.01

Table 1: Performance Evaluation (time in seconds)

Ex.	$n_{\mathbf{x}}$	$d_{\mathbf{f}}$	$NN_{B(\mathbf{x})}$	SNBC							FOSSIL					NNCChecker					SOSTOOLS	
				$NN_{\lambda(\mathbf{x})}$	d_B	I_s	$T_l(S)$	T_c	$T_v(S)$	$T_e(S)$	d_B	I_f	$T_l(F)$	$T_v(F)$	$T_e(F)$	d_B	I_n	$T_l(N)$	$T_v(N)$	$T_e(N)$	d_B	$T_e(L)$
C_1 [4]	2	3	2-10-1	2-5-1	2	1	0.166	0	0.278	0.444	2	3	3.882	0.017	3.899	2	1	4.960	0.603	5.563	4	**0.133**
C_2 [3]	2	3	2-10-1	2-5-1	2	1	0.388	0	0.295	0.683	2	3	4.047	0.005	4.052	2	1	4.892	0.401	5.293	4	**0.115**
C_3 [4]	2	2	2-5-1	2-5-1	2	1	0.295	0	0.279	0.574	2	2	3.223	0.006	3.229	2	1	3.633	0.422	4.055	4	**0.125**
C_4 [16]	2	2	2-20-1	2-5-1	2	1	0.490	0	0.335	0.825	2	6	63.125	0.052	63.177	2	2	3.218	0.804	4.022	4	**0.149**
C_5 [16]	2	3	2-5-1	2-5-1	2	1	0.032	0	0.297	**0.329**	2	3	0.340	0.004	0.344	2	1	4.077	0.505	4.582	×	×
C_6 [3]	3	3	3-5-1	3-5-1	2	1	0.379	0	0.556	0.935	2	7	1.646	0.009	1.655	2	1	4.476	0.902	5.378	4	**0.248**
C_7 [5]	3	2	3-5-1	3-5-1	2	2	1.286	0.084	0.948	2.318	2	2	2.651	0.008	2.659	2	1	4.692	1.028	5.720	4	**0.478**
C_8 [4]	4	3	4-5-1	4-5-1	2	1	0.207	0	1.256	**1.463**	2	7	6.795	6892.012	6898.807	2	1	14.013	145.303	159.316	6	3.039
C_9 [13]	5	2	5-10-1	5-5(2)-1	2	4	2.731	3.232	7.814	**13.777**	-	-	-	-	OT	2	3	66.551	461.730	528.281	6	18.247
C_{10} [16]	6	2	6-15-1	c	2	4	11.346	8.933	13.625	**33.904**	-	-	-	-	OT	×	×	×	×	×	×	×
C_{11} [3]	6	3	6-20-1	c	2	8	18.341	6.405	25.221	**49.967**	-	-	-	-	OT	×	×	×	×	×	×	×
C_{12} [9]	7	1	7-20-1	7-5-1	2	12	294.269	23.428	50.955	**368.652**	-	-	-	-	OT	×	×	×	×	×	6	2037.865
C_{13} [9]	9	1	9-15-1	c	2	8	72.795	452.513	95.074	**620.382**	-	-	-	-	OT	×	×	×	×	×	×	×
C_{14} [8]	12	1	12-20-1	c	2	25	28.089	7.123	967.559	**1002.771**	-	-	-	-	OT	×	×	×	×	×	2	1210.985

times as fast as *FOSSIL*, and 25.62 times faster than *NNCChecker*. The empirical results show that our *SNBC* is more efficient.

Comparing *SNBC* with the *SOSTOOLS* based on the SOS relaxation method, our approach successfully deals with all 14 examples, whereas *SOSTOOLS* finds 10 barrier certificates. In addition, to evaluate the best performance of *SOSTOOLS*, we have tried some polynomial multipliers with random coefficients and the degree bound ≤ 2. There are 4 examples that can be treated by our *SNBC* tool but are unable to be handled by the *SOSTOOLS* after 7200 seconds of computation. The above analysis can be used to support that our tool does contribute to raising the effectiveness of barrier certificate generation. Furthermore, we compared the efficiency of the two methods for the cost time of barrier certificate generation. In cases where successful verification is possible, *SOSTOOLS* exhibits better efficiency than *SNBC* when $n_{\mathbf{x}} \leq 3$. However, for $n_{\mathbf{x}} \geq 4$, *SNBC* requires less time compared to *SOSTOOLS*, especially in C_{12}, where *SNBC* is 5.53 times faster than *SOSTOOLS*. The average computation time for safety verification is also 2.35 times faster in *SNBC* compared to the *SOSTOOLS* tool.

These experimental results demonstrate that our approach exhibits stronger barrier certificate learning capability compared to other neural barrier certificate syntheses. Furthermore, compared to methods that utilize SOS and semi-definite programming relaxations to numerically solve non-convex BMI or LMI optimization problems, our approach only involves testing the feasibility of convex LMIs. This advantage enhances its verification efficiency and expands its applicability to systems with high dimensions.

6 CONCLUSION

In this paper, we introduced a BC learning method with efficient SOS-based verification. We first abstracted the NN controller as a polynomial with an approximation error. Subsequently, we proposed a counterexample-guided framework for neural BC synthesis. Considering the generated BC candidates from learning phase, we converted the verification problem into three LMI feasibility testing programmings, instead of relying on an SMT solver to verify the BC conditions. The experimental results on a set of benchmarks demonstrate that our proposed approach is more scalable and effective compared to traditional SOS-based barrier certificate generation methods and state-of-the-art neural barrier certificate synthesis methods with SMT-based verification programs.

REFERENCES

[1] Alessandro Abate, Daniele Ahmed, Alec Edwards, Mirco Giacobbe, and Andrea Peruffo. 2021. FOSSIL: a software tool for the formal synthesis of lyapunov functions and barrier certificates using neural networks. In *Proceedings of the 24th HSCC*. 1–11.

[2] Clark Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli. 2021. Satisfiability Modulo Theories. In *Handbook of Satisfiability* (second ed.). IOS Press.

[3] Xin Chen, Chao Peng, Wang Lin, Zhengfeng Yang, Yifang Zhang, and Xuandong Li. 2020. A novel approach for solving the BMI problem in barrier certificates generation. In *Proceedings of CAV*. Springer, 582–603.

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

[5] Jyotirmoy V. Deshmukh, James P. Kapinski, Tomoya Yamaguchi, and Danil Prokhorov. 2019. Learning Deep Neural Network Controllers for Dynamical Systems with Safety Guarantees: Invited Paper. In *2019 IEEE/ACM ICCAD*. 1–7.

[6] Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George Pappas. 2019. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. In *Advances in NIPS*, Vol. 32.

[7] Sicun Gao, Soonho Kong, and Edmund M. Clarke. 2013. dReal: An SMT Solver for Nonlinear Theories over the Reals. In *Proceedings of the 24th CADE*. 208–214.

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

[9] Edda Klipp, Ralf Herwig, Axel Kowald, Christoph Wierling, and Hans Lehrach. 2005. *Systems biology in practice: concepts, implementation and application.*

[10] Andrea Peruffo, Daniele Ahmed, and Alessandro Abate. 2021. Automated and Formal Synthesis of Neural Barrier Certificates for Dynamical Models. In *Tools and Algorithms for the Construction and Analysis of Systems*. 370–388.

[11] Stephen Prajna. 2004. SOSTOOLS: Sum of squares optimization toolbox for MATLAB. *http://www. mit. edu/˜ parrilo/sostools/index. html* (2004).

[12] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. 2007. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE TAC* 52, 8 (2007), 1415–1429.

[13] Mohamed Amin Ben Sassi and Sriram Sankaranarayanan. 2015. Stabilization of polynomial dynamical systems using linear programming based on Bernstein polynomials. arXiv:1501.04578 [math.OC]

[14] Meng Sha, Xin Chen, Yuzhe Ji, Qingye Zhao, Zhengfeng Yang, Wang Lin, Enyi Tang, Qiguang Chen, and Xuandong Li. 2021. Synthesizing Barrier Certificates of Neural Network Controlled Continuous Systems via Approximations. In *Proceedings of 58th ACM/IEEE DAC*. 631–636.

[15] Armando Solar-Lezama, Liviu Tancau, Rastislav Bodik, Sanjit Seshia, and Vijay Saraswat. 2006. Combinatorial sketching for finite programs. In *Proceedings of the 12th ASPLOS*. 404–415.

[16] Xia Zeng, Wang Lin, Zhengfeng Yang, Xin Chen, and Lilei Wang. 2016. Darboux-type barrier certificates for safety verification of nonlinear hybrid systems. In *Proceedings of the 13th EMSOFT*. 1–10.