Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2020-TR-004

2020-TR-004

Synthesizing Barrier Certificates of Neural Network Controlled Continuous Systems via approximations

Meng Sha, Xin Chen, Qingye Zhao, Enyi Tang, Xuandong Li

Technical Report 2020-TR-004

Synthesizing Barrier Certificates of Neural Network Controlled Continuous Systems via approximations

***Abstract*—The paper presents a barrier certificate based approach to verifying safety properties of closed-loop systems using neural networks as controllers. It deals with the verification problem in the infinite time horizon and exploits the approximated system of the original one to synthesize the candidate barrier certificates, where the behavior of a neural network controller is approximated by a polynomial with a bounded error. Satisfiability Modulo Theories solvers are then utilized to identify real barrier certificates from those candidates. As a barrier certificate can separate the over-approximation of the reachable set from the unsafe region, once it is constructed, the safety property gets proved. We show the advantage of our approach in barrier certificates synthesis by comparing it with the state-of-art work on a set of benchmarks.**

***Index Terms*—Safety verification, Barrier certificates, Neural network controller, Satisfiability-Modulo-Theories**

I. Introduction

There has been a sudden upsurge in research focusing on Deep Neural Network (DNN) controlled Cyber-Physical Systems (CPS) such as self-driving cars, drones, and air traffic collision avoidance systems. However, their safety and reliability problems have not been adequately explored. Some incidents of safety-critical CPSs have highlighted the need of techniques providing formal guarantees for correctness of DNNs as well as overall safety of the systems being controlled.

Safety verification of a system aims at proving all its trajectories starting from the initial set never enter the unsafe region. It is directly performed by computing the reachable set or the over-approximation of the reachable set, and then proving the intersection of reachable set or its over-approximation and the unsafe region is empty.

Dutta et al. propose an approach to inferring the over-approximation of the reachable set for continuous-time dynamical systems with neural network controllers in [1]. The behavior of the network is abstracted by a local polynomial approximation along with exact error bounds. Then it is integrated with a Taylor model-based flowpipe construction scheme for continuous differential equations to derive the reachable set. To estimate the error between a neural network and its polynomial approximation, a piecewise linearization (PWL) of the polynomial is introduced as the intermediator. Here, branch and bound search with interval analysis are adopted to find the error between the polynomial and the PWL approximation. while mixed integer linear programming with local gradient-descent search are deployed to determine the error between the neural network model and the PWL approximation.

Sun et al. propose a reachability analysis based method to certify discrete-time linear dynamics governed robots, where a Rectified Linear Unit (ReLU) neural network produces control inputs from LiDAR images to drive the robot avoiding a set of polytopic obstacles marked as unsafe regions [2]. They construct a finite state abstraction of the robot and compute the set of safe initial states by Satisfiability Modulo Convex (SMC) encoding.

Ivanov et al. enable the reachable set computation of sigmoid neural network controlled hybrid systems by transforming a sigmoid-based neural network into an equivalent hybrid system and composing it with the plant [3]. Such a transformation utilizes the special characteristics of the sigmoid activation function, that is the sigmoid is the solution to a quadratic differential equation. It thus can't be extended to networks using other activation functions.

Reachable set computation based approaches suffer from two limitations. Firstly, the heavy computational complexity arising from the nonlinear ODEs, the nonlinear activation functions and a huge number of neurons make it difficult to scale. Besides, as it computes the reachable set step by step, it can only guarantee safety over a finite time horizon.

Verification approaches using *barrier certificates* (BC) can prove safety properties over an infinite time horizon. A barrier certificate is a continuous function of states that separates reachable set or its over-approximation from unsafe regions [4]. Thus, the existence of a barrier certificate forms a sufficient condition of retaining the safety property.

As far as we know, the only work devoted to synthesizing barrier certificates for neural network controlled systems was proposed by Tuncali et al. [5]. It first specifies a generator function $W(x)$ as a positive polynomial with undetermined coefficients of the monomial terms. Then, a set of simulation trajectories starting from randomly selected initial states are collected to generate a set of linear constraints about the coefficients that makes the $W(x)$ be positive and decrease along system trajectories. A feasible solution to the linear program (LP) problem corresponds to a candidate generator function which is further certified to be positive and decreasing over the whole state invariant except for the initial states by SMT solvers. The candidate barrier certificate $B(x)$ of the form $W(x) - l, l \in R^{>0}$ is derived from a certified generator function $W(x)$ by determining the unknown level set size l. The appropriate l value is expected to make the set $L = \{x|B(x) < 0\}$ contains the initial state set and does not intersect with the unsafe region. Again SMT solvers as well as a binary search are adopted to find a correct l.

Note that, the classical construction condition only requires barrier certificates to be decreasing on its zero level set [4]. However, when building the generator function $W(x)$, due to the difficulty in encoding linear constraints as both coefficients and state variables are unknown, the condition has to be relaxed to be decreasing over the whole state invariant except for the initial states. It is then unsurprising that their method rules out many real barrier certificates.

To overcome this limitation, we investigate an alternative approach, that finds candidate barrier certificates via the classical construction conditions. We first build a polynomial approximation system and synthesize its barrier certificate from the classical construction conditions using Sum-of-Squares tools. An approximated system consists of the polynomial approximation of the DNN controller and the plant, where the upper bound of the approximation error is estimated by simulation and treated as disturbance to control inputs in BC synthesis. The derived barrier certificate doesn't become a real one until its soundness to the system under verification is certified by SMT solvers. Furthermore, an iterative scheme is developed that allows the polynomial approximation system to be iteratively refined when counterexamples are found by SMT solvers.

We have developed a prototyping tool NNCChecker implementing the proposed approach and compared it with the state-of-art work on a set of benchmark examples where neural network controllers use three popular activation functions: ReLU, Sigmoid and Tanh. In the experiment, for the 12 examples, our NNCChecker can handle 10 of them while the opponent only solve 5 cases, which shows that our approach is much more effective in barrier certificate generation.

II. Preliminaries

The section formulates the safety verification problem considered in this paper. We consider a continuous system driven by a deep neural network controller (CSD), as shown in Figure 1, with states x, measurements y, and a network controller producing input u. The system is formalized first, followed by the DNN controller and the safety verification problem statement itself.

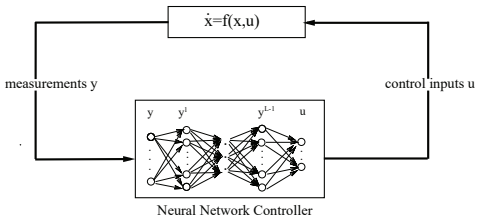

Fig. 1. A diagram of a continuous system with a neural feedback controller

A continuous dynamical system is modeled by an ODE $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, where $\mathbf{x} \in \mathbb{R}^n$ is the system state vector, $\mathbf{u} \in \mathbb{R}^m$ is the control input vector, and $\dot{\mathbf{x}}$ denotes the derivative of $\mathbf{x}$ with respect to the time variable t. It is assumed that $\mathbf{f}$ satisfies the Lipschitz continuous in $\mathbf{x}$ and $\mathbf{u}$, which ensures that there exists a unique solution to the ODE for the given input, given a time $T > 0$. The formal definition is provided below.

Definition 1: (Continuous Dynamical System). A continuous dynamical system with inputs $\mathbf{u}$ and outputs $\mathbf{y}$ consists of a quintuple $S : \langle V, \Theta, D, \Psi, \mathbf{g} \rangle$, where

- $V = \{x_1, ..., x_n\}$, a set of real-valued system *variables*. A *state* is an interpretation of V, assigning to each $x_i \in V$ a real value;
- Θ, the set of specifying the *initial* condition;
- $\mathcal{D}$, a map that assigns a vector field $\mathbf{f}$, i.e., $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$;
- Ψ, the state invariant;
- $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^k$, the observation model such that $\mathbf{y} = \mathbf{g}(\mathbf{x})$.

DNN Controller. A deep neural network controller $f_c : \mathbb{R}^k \to \mathbb{R}^m$ maps measurements $\mathbf{y}$ to control inputs $\mathbf{u}$. As depicted in Fig. 1, the DNN controller consists of an input layer, an output layer, and multiple hidden layers in between. Neurons in a DNN are distributed in disjoint layers, with each neuron in one layer connected to the next layer. Specifically, $\mathbf{y}, \mathbf{u}$ are the input and the output respectively, and $\mathbf{y}^1, \ldots, \mathbf{y}^{L-1}$ correspond to the values of the neurons in hidden layers L^i with $i = 1, \cdots, L-1$. In hidden layers, the output of each neuron is assigned by a non-linear activation function whose input is a linear combination of the neuron outputs, residing in the previous layer, i.e.,

$$\mathbf{y}^i = f^i(\mathbf{y}^{i-1}) = \phi(W^i \mathbf{y}^{i-1} + b^i).$$

Frequently used activation functions include ReLU, Sigmoid, and Tanh. For instance, the commonly used ReLU activation function is defined as $\mathrm{ReLU(x)} = \max(0, \mathrm{x})$. Based on the DNN architecture, the controller can be represented as a composition function:

$$\mathbf{u} = f_c(\mathbf{y}) = f^L(f^{L-1}(\cdots(f^1(\mathbf{y})))).$$

In the paper, we assume that the DNN controller is pre-trained, i.e. the parameters of DNN, including the weighted matrices and bias vectors, are determined in the training phase.

Given a dynamical system $\mathcal{S}$, its controller can be regarded as the composition of the observation $g(\mathbf{x})$ and the DNN function $\mathbf{u} = f_c(\mathbf{y})$, i.e., $\mathbf{u} = f_c(g(\mathbf{x}))$. Accordingly, the vector field of $\mathcal{S}$ can also be represented as follows

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \text{ with } \mathbf{u} = f_c(g(\mathbf{x})). \tag{1}$$

To ease presentation, this work focuses on the vector field with the following simple form:

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

where $f(\mathbf{x})$ is a polynomial vector. Moreover, the initial set Θ and the state invariant Ψ in $\mathcal{S}$ are semialgebraic, described by polynomial equations and inequalities.

Safety Verification. The problem of safety verification of a continuous system $\mathcal{S} : \langle V, \Theta, D, \Psi, g \rangle$ with respect to an unsafe assertion Ξ_u, is to verify that each trajectory $\mathbf{x}(t)$ of $\mathcal{S}$, starting from the initial state $\mathbf{x}_0$ chosen from Θ randomly, can not invade the unsafe region specified by Ξ_u, i.e.,

$$\mathrm{Safety}(\mathcal{S}, \Xi_\mathrm{u}) \triangleq \forall \mathbf{x}_0 \in \Theta \; \forall \mathrm{t} \geq 0 \Rightarrow \mathbf{x}(\mathrm{t}) \notin \Xi_\mathrm{u}. \tag{3}$$

Remark that $\mathbf{x}(t)$ is also called a reachable state if it appears in some trajectories of $\mathcal{S}$, and the collection including all reachable states is called the reachable set of $\mathcal{S}$ with respect to the initial set Θ. Therefore, $\mathcal{S}$ is safe if and only if the reachable set never intersect with the unsafe region Ξ_u.

Barrier certificates provide useful unbounded-time safety certificates of the system. In the following, we recall the formal definition of barrier certificate provided in [4].

Definition 2: Let $\mathcal{S} : \langle V,\ \Theta,\ D,\ \Psi, \mathbf{g}\rangle$ be a continuous system with a DNN controller, and Ξ_u be an unsafe region. If there exist real-valued functions $B(\mathbf{x}), \lambda(\mathbf{x})$ which satisfy the following conditions:

1) $B(\mathbf{x}) \geq 0,\ \forall x \in \Theta$.
2) $B(\mathbf{x}) = 0 \implies \dot{B}(\mathbf{x}) > 0. \forall \mathbf{x} \in \Psi$,
3) $B_\ell(\mathbf{x}) < 0,\ \forall \mathbf{x} \in \Xi_u$.

then $B(\mathbf{x})$ is called a barrier certificate of $\mathcal{S}$ w.r.t. Ξ_u.

Depending on Conditions 1) and 2) in Definition 2, it is seen that $B(\mathbf{x}) \geq 0$ for all reachable states $\mathbf{x}$, and Condition 3) yields that $B(\mathbf{x}) < 0$ for all $\mathbf{x} \in \Xi_u$. Therefore, $B(\mathbf{x})$ can separate all system trajectories starting from the initial state Θ from the unsafe set Ξ_u, namely, the existence of a barrier certificate suffices to prove the safety of the system.

III. Computational Method

We present an iterative scheme to synthesize barrier certificates for continuous systems with a DNN controller (CSD). The key idea in each iteration is to construct the polynomial approximation system of the CSD, and then establish a polynomial optimization problem helping to generate barrier certificates for the approximation system. As the polynomial approximation system is designed to enclose the CSD, barrier certificates for the former are candidate BCs for the latter. We identify real barrier certificates using an SMT solver. The SMT solver either returns UNSAT, which certifies that the candidate is sound, or produces a counterexample that results in a refined polynomial approximation system. There are certain nuances in each step that are described below.

A. Polynomial Approximation of CSDs

Let $\mathcal{S} : \langle V, \Theta, D, \Psi, \mathbf{g}\rangle$ be a continuous system driven by a neural network controller $\mathbf{u} = \mathbf{f}_c(\mathbf{x})$, where the state invariant Ψ is a compact convex set. For such a CSD $\mathcal{S}$, we try to build an polynomial approximation system $\widetilde{\mathcal{S}}$ through polynomial approximation on the controller, such that $\mathcal{S}$ is within $\widetilde{\mathcal{S}}$.

Consider the controller $\mathbf{u} = \mathbf{f}_c(\mathbf{x})$, whose function is defined as:

$$\mathbf{u}_i = \mathbf{f}_{c,i}(\mathbf{x}),\ i = 1, \ldots, n, \tag{4}$$

where $\mathbf{x}$ takes values in $\Psi \subseteq \mathbb{R}^n$. Here, the $i-th$ element of the controller $\mathbf{f}_{c,i}(\mathbf{x})$ is approximated by a polynomial $p_i(\mathbf{x}),\ for\ \ i = 1, \ldots, n$, and μ_i the bounds of $|\mathbf{f}_{c,i}(\mathbf{x}) - p_i(\mathbf{x})|$ for all $\mathbf{x} \in \Psi$, namely,

$$|\mathbf{f}_{c,i}(\mathbf{x}) - p_i(\mathbf{x})| < \mu_i, \forall \mathbf{x} \in \Psi. \tag{5}$$

In the paper, multivariate polynomial interpolation is applied to compute an approximate polynomial. For easy presentation, in the following, we use $\phi(\mathbf{x})$ to denote the i-th element of the given DNN function. Our goal is to obtain an approximate polynomial $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ with a error bound $\mu \in \mathbb{R}_+$, such that

$$|\phi(\mathbf{x}) - p(\mathbf{x})| < \mu, \forall \mathbf{x} \in \Psi,$$

and the bound μ is expected to be small enough.

The technique of oversampling is explored to get a better approximate polynomial, i.e., the number of the interpolation points is greater than that of the terms of the target polynomial $p(\mathbf{x})$. Given the interpolation points, the approximate polynomial $p(\mathbf{x})$ can be worked out by solving a least squares problem. Specifically, a real coefficient vector $\mathbf{c}$ is used to parameterize a polynomial $p(\mathbf{x}, \mathbf{c})$ with a given degree d, i.e, $p(\mathbf{x}, \mathbf{c}) = \sum_j c_j b_j(\mathbf{x})$, where $b_j(\mathbf{x})$ are monomials with total degree $\leq d$. A mesh M on Ψ with a small spacing $s \in \mathbb{R}_+$ is constructed to produce enough sampling points by computing $y_j = \phi(\mathbf{v}_j) \in \mathbb{R}$ for $1 \leq j \leq m$ at mesh points $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m\}$.

Let the coefficient vector $\mathbf{c}$ of $p(\mathbf{x})$ be undetermined. We construct a linear system

$$A \cdot \mathbf{c} = \mathbf{y}, \tag{6}$$

where A is of size $m \times \nu$ with $m > \nu$. By solving the above overdetermined system, we obtain $p(\mathbf{x}, \mathbf{c})$ as the approximation of $\phi(\mathbf{x})$ with $x \in \Psi$.

Having $p(\mathbf{x}, \mathbf{c})$, we proceed to consider how to evaluate an error bound μ, namely, $|\phi(\mathbf{x}) - p(\mathbf{x}, \mathbf{c})| < \mu, \forall \mathbf{x} \in \Psi$. To estimate the error bound μ. The property of Lipschitz continuity supports the following fact easily.

Fact 1. Suppose $r(\mathbf{x})$ is a continuous and differential function on a compact convex set $\Psi \subset \mathbb{R}^n$. Assume that a mesh is constructed in Ψ, wherein $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_m\}$ and $s \in \mathbb{R}_{>0}$ are the mesh points and mesh spacing of Ψ, respectively. Let $\mu_0 = \max\{|r(\mathbf{v}_1)|, |r(\mathbf{v}_2)|, \ldots, |r(\mathbf{v}_m)|\}$, and $\beta = \sup_{\mathbf{x}\in\Psi} \| \bigtriangledown r(\mathbf{x})\|$, then

$$|r(\mathbf{x})| \leq \beta s + \mu_0,\ \forall \mathbf{x} \in \Psi. \tag{7}$$

Consider that the DNN controller $\mathbf{u} = \mathbf{f}_c(\mathbf{x})$ whose activation function is ReLU. It is easy to see that the controller function is non-differential. In this situation, we extend Fact 1 to estimate the error between the non-differential function $\phi(\mathbf{x})$ and the polynomial $p(\mathbf{x})$.

Theorem 1: Under the same assumptions as above, and assume that $r(\mathbf{x})$ is continuous function on Ψ. Suppose $\beta = \sup_{\mathbf{x}\in\Psi} \| \bigtriangledown^+ r(\mathbf{x})\|$ with $\bigtriangledown^+ r(\mathbf{x}) = [\frac{\partial^+ r}{\partial^+ x_1}, \ldots, \frac{\partial^+ r}{\partial x_n}]^T$, where ∂^+ is the one-sided directional derivative. Then the error bound estimation (7) in Fact 1 still holds, i.e.,

$$|r(\mathbf{x})| \leq \beta s + \mu_0,\ \forall \mathbf{x} \in \Psi. \tag{8}$$

Notably, the bound of the error function $r(\mathbf{x})$ in Theorem 1 partially depends on the mesh spacing s. Thus, the error estimation (8) can yield a tight bound when s is small enough.

Sequentially, the Lipschitz constant β of the error function $r(\mathbf{x})$ needs to be computed. Following the idea presented in [6], an estimation of β can be computed by combining

extreme value theory with sampling evaluations $\|\bigtriangledown^{+} r(\mathbf{x}^{(i)})\|$. More details can be found in [6].

Given a CSD $\mathcal{S}$ with the controller $\mathbf{u}(\mathbf{x}) = f_c(\mathbf{x})$, the above method can yield the approximate polynomials with the error bounds, i.e., for $1 \leq i \leq n$ we have

$$\forall \mathbf{x} \in \Psi,\ |\mathbf{f}_{c,i}(\mathbf{x}) - p_i(\mathbf{x})| < \mu_i, 1 \leq i \leq n.$$

As a result, the controller can be represented as an interval polynomial vector, i.e.,

$$\mathbf{f}_{c,i}(\mathbf{x}) = p_i(\mathbf{x}) + r_i,\ \text{ with } r_i \in [-\mu_i, \mu_i].$$

A polynomial approximation continuous system $\widetilde{\mathcal{S}}$ with continuous dynamic $\dot{\mathbf{x}} = \tilde{\mathbf{f}}(\mathbf{x}, \mathbf{r})$ is represented as:

$$\dot{x}_i = \mathbf{f}_i(\mathbf{x}) + p_i(\mathbf{x}) + r_i,\ \text{ with } r_i \in [-\mu_i, \mu_i]. \tag{9}$$

It is worth noting that as the error bound μ_i is computed from not the real β, but an estimation of it, there is no formal guarantee ensuring that the approximation system $\widetilde{\mathcal{S}}$ encloses $\mathcal{S}$. However, by gradually enlarge β, the enclosure system $\widetilde{\mathcal{S}}$ can surely be obtained.

B. Barrier Certificate Generation for Polynomial Approximation Continuous Systems

The section explores SOS relaxation optimization method to produce the barrier certificate of the approximate continuous system $\widetilde{\mathcal{S}}$. For the verification problem concerning a CSD with the unsafe region Ξ_u, we introduce $g_{Init}(\mathbf{x})$,$g_I(\mathbf{x})$ and $g_{Unsafe}(\mathbf{x})$ to define the initial set Θ, the state invariant Ψ and the unsafe region as follows:

$$\begin{cases} \Theta = \{\mathbf{x} \in \mathbb{R}^n : g_{Init}(\mathbf{x}) < 0\}, \\ \Psi = \{\mathbf{x} \in \mathbb{R}^n : g_I(\mathbf{x}) \geq 0\}, \\ \Xi_u = \{\mathbf{x} \in \mathbb{R}^n : g_{Unsafe}(\mathbf{x}) \geq 0\}. \end{cases}$$

The approximation errors $\mathbf{r}$ are treated as disturbances and are taken into account when synthesizing barrier certificates. Let $\widetilde{\mathcal{S}}$ be an polynomial approximation system whose vector field is $\tilde{f}(\mathbf{x}, \mathbf{r})$, and the parameters $\mathbf{r}$ select values in the $n-$dimensional hypercube $R = [-\mu_1, \mu_1] \times \cdots \times [-\mu_n, \mu_n]$. We may define

$$g_{para}(\mathbf{r}) = \begin{bmatrix} (r_1 + \mu_1)(\mu_1 - r_1) \\ \vdots \\ (r_n + \mu_n)(\mu_n - r_n) \end{bmatrix}, \tag{10}$$

with variable vector $\mathbf{r} = [r_1, \ldots, r_n]^T$. Clearly, $\mathbf{r} \in R$ is expressed as the polynomial inequalities, $g_{para}(\mathbf{r}) \geq 0$.

Following Definition 2, the synthesis of a polynomial barrier certificate is to find $B(\mathbf{x})$ and $\lambda(\mathbf{x})$ satisfy the following constraints

$$B(\mathbf{x}) \geq 0 \text{ if } g_{Init}(\mathbf{x}) \geq 0, \tag{11}$$

$$\dot{B}(\mathbf{x}) - \lambda(\mathbf{x})B(\mathbf{x}) > 0,\ \text{if } g_I(\mathbf{x}) \geq 0,\ g_{para}(\mathbf{r}) \geq 0, \tag{12}$$

$$-B(\mathbf{x}) > 0,\ \text{if } g_{Unsafe}(\mathbf{x}) \geq 0, \tag{13}$$

where $\dot{B}(\mathbf{x}) = \frac{\partial B}{\partial \mathbf{x}}(\mathbf{x})\tilde{f}(\mathbf{x}, \mathbf{r})$. To compute a polynomial certificate, real coefficient vector $\mathbf{c}$ is deployed to parameterize $B(\mathbf{x})$, that is, $B(\mathbf{x}, \mathbf{c}) = \sum_j c_j b_j(\mathbf{x})$, where $b_j(\mathbf{x})$ are monomials of degree $\leq$ the prior bound.

Following the spirit in [4], a computational method based on SOS relaxation can be used to yield a barrier certificate. Consider the constraint (11), a sufficient condition for $B(\mathbf{x}) \geq 0$ on a semi-algebraic set $\{\mathbf{x} \in \mathbb{R}^n : g_{Init}(\mathbf{x}) \geq 0\}$ is to find Sum-of-Squares(SOS) polynomials $\sigma_0(\mathbf{x}), \sigma_1(\mathbf{x})$ such that $B(\mathbf{x})$ can be written as $B(\mathbf{x}) = \sigma_0(\mathbf{x}) + \sigma_1(\mathbf{x})g_{Init}(\mathbf{x})$. In the same manner, the constraints (12-13) can also be expressed as the associated SOS representations. Then the desired barrier certificate can be derived by solving the following SOS program,

$$\left.\begin{array}{ll} \text{find} & \mathbf{c} \\ \text{s.t.} & B(\mathbf{x}, \mathbf{c}) - \sigma(\mathbf{x})g_{Init}(\mathbf{x}) \text{ is SOS}, \\ & \dot{B}(\mathbf{x}, \mathbf{c}) - \lambda(\mathbf{x})B(\mathbf{x}, \mathbf{c}) - \phi(\mathbf{x})g_I(\mathbf{x}) \\ & \qquad -\gamma(\mathbf{x}, \mathbf{r})g_{para}(\mathbf{r}) - \epsilon_1 \text{ is SOS}, \\ & -B(\mathbf{x}, \mathbf{c}) - \mu(\mathbf{x})g_{Unsafe}(\mathbf{x}) - \epsilon_2 \text{ is SOS}, \\ & \sigma(\mathbf{x}),\ \phi(\mathbf{x}),\ \gamma(\mathbf{x}, \mathbf{r}),\ \mu(\mathbf{x}) \text{ are SOSes}, \end{array}\right\} \tag{14}$$

where $\epsilon_1, \epsilon_2 \in \mathbb{R}_{>0}$ are prespecified small positive numbers.

Remark 1: The product between $B(\mathbf{x})$ and its multiplier $\lambda(\mathbf{x})$ in (14) makes the construction condition non-convex. To make the problem tractable, the canonical approach is to let the multiplier $\lambda(\mathbf{x})$ be fixed polynomial [4]. Here this strategy can also be explored to simplify the non-convex constraint into the associated convex one.

C. Identify Real Barrier Certificates

Ideally, the reachable set of the polynomial approximation system $\widetilde{\mathcal{S}}$ is a superset of the system $\mathcal{S}$, as the vector field $\tilde{f}(\mathbf{x}, \mathbf{r})$ is designed to enclose $f(\mathbf{x})$. Consequently, it is reasonable to assume a barrier certificate of $\widetilde{\mathcal{S}}$ being a candidate BC of the original system.

Unfortunately, the inclusion of vector fields has no rigorous guarantees as the Lipschitz constant β used for calculating approximation errors is estimated on sample points. We thus perform an iterative schema to identify real barrier certificates from the candidates.

In each iteration, we first resort to SMT solvers to check the following property over the invariant Ψ:

$$\begin{array}{l} (\exists x \in \Theta, B(x) < 0) \\ \vee\, (\exists x \in \Xi_u, B(x) \geq 0) \\ \vee\, (\exists x \in \Psi, B(x) = 0 \wedge \dot{B}(x) < 0) \end{array} \tag{15}$$

The above query is UNSAT when $B(\mathbf{x})$ satisfies all construction conditions of barrier certificates, which means $B(\mathbf{x})$ certifies safety property of the system $\mathcal{S}$. If a counterexample is returned by the SMT solver, it indicates that the polynomial approximation continuous system $\tilde{\mathcal{S}}$ doesn't completely wrap up the system $\mathcal{S}$. In this situation, we enlarge β to build a new polynomial approximation continuous system, and pass it to the next iteration.

Following the above discussions, we present an algorithm to synthesize polynomial certificates for the given continuous system with a DNN controller.

Algorithm 1: Verification framework for a continuous system with a DNN controller

Input: A CSD $\mathcal{S} : \langle V, \Theta, \Delta, \Psi, \mathbf{g}, \Xi_u \rangle$.
d : degree of polynomial approximation.
γ : zoom factor.
Output: SAFE or UKN
Procedure `NNCChecker`**:**
  Get the polynomial approximation p of the DNN controller.
  Evaluate Lipschitz constant β.
  Calculate the error bound μ using Theorem 1.
  while *1* **do**
    Build the polynomial approximation continuous system $\widetilde{\mathcal{S}}(\mathcal{S}, p, \mu)$.
    Generate barrier certificate B of $\widetilde{\mathcal{S}}$ by solving the program (14).
    if *exist* B **then**
      SMT solver checks B according to (15).
      if *UNSAT* **then**
        return SAFE.
      else
        $\beta = \gamma * \beta$.
        Update the error bound μ using β.
    else
      return UKN.

IV. Experiments

We have implemented the proposed approach as a prototype tool NNCChecker in Matlab and Python. In the following, we demonstrate its effectiveness by synthesis barrier certificates of several examples, and then compare its performance with the state-of-art work in [5] over a set of benchmarks.

Example 1: Consider a mass-spring-damper system depicted in Fig. 2 [7], where x_1, v_1 and x_2, v_2 represent the displacements and velocities of m_1, m_2 respectively, k_1 and k_2 denote the springs, d is the anti-damper, u is the forcing input to m_1.

The dynamics of the system is defined by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{v}_1 \\ \dot{x}_2 \\ \dot{v}_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ x_2 - 2x_1 + \frac{1}{10}(-x_1^3 + (x_2 - x_1)^3 + v_1 - v_2) \\ v_2 \\ x_1 - x_2 + \frac{1}{10}(x_1 - x_2)^3 + \frac{1}{10}(v_2 - v_1) \end{bmatrix} + \begin{bmatrix} 0 \\ u \\ 0 \\ 0 \end{bmatrix}$$

with domain $\Psi = \{\mathbf{x} \in \mathbb{R}^4 \,|\, x_1^2 + x_2^2 + v_1^2 + v_2^2 \leq 10\}$.

Set the initial set Θ of the system as

$$\Theta = \{\mathbf{x} \in \mathbb{R}^4 \,|\, -0.2 \leq x_1, x_2, v_1, v_2 \leq 0.2\},$$

a DNN controller $u = f_c(x_1, x_2, v_1, v_2)$ is trained by a standard MPC control scheme, which is designed to retain the state variables within the range of $[-0.5, 0.5]$. It consists of 5 hidden layers, 200 neurons in each layer, and uses ReLU activation functions.

Our goal is to verify that all trajectories of the system starting from the initial set Θ never enter the unsafe region Ξ_u, defined as:

$$\{\mathbf{x} \in \mathbb{R}^4 \,|\, (x_1 - 1.5)^2 + (x_2 - 1.5)^2 + v_1^2 + v_2^2 \leq 1\}.$$

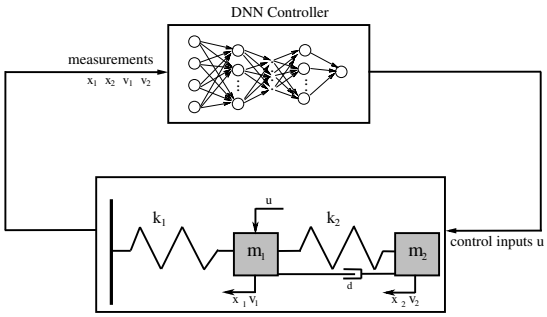

Fig. 2. A mass-spring-damper system

Following Algorithm 1, we first obtain a polynomial p with degree 2 as the approximation of the DNN controller u,

$$\begin{aligned} p =& 0.1392 - 35.5842x_1 - 7.6984v_1 + 22.3924x_2 - 14.5562v_2 \\ &+ 0.0059x_1^2 + 0.0003x_2^2 + 0.0013x_1v_1 - 0.0041x_1x_2 \\ &+ 0.0027x_1v_2 - 0.0001v_1x_2 - 0.0003x_2v_2, \end{aligned}$$

where the associated error bound between u and p is identified as: $\mu = 0.2155$.

By fixing its degree bound $d = 2$ and parameterizing it with an unknown coefficient vector $\mathbf{c}$, the barrier certificate $B(x_1, x_2, v_1, v_2)$ is derived by solving the SOS program of the form (14).

The SDP solver *SOSTOOLS* returns a solution, which corresponds to the following polynomial barrier certificate $B(x_1, x_2, v_1, v_2)$:

$$\begin{aligned} B =& 49134.8 - 59531.6\,x_1 - 7550.9\,v_1 \\ &+ 41159.5\,x_2 - 30158.5\,v_2 - 89054.8\,x_1^2 \\ &- 7959.1\,v_1^2 + 32777.6\,x_2^2 + 18433.7\,v_2^2 \\ &- 45609.1\,x_1\,v_1 + 27464.7\,x_1\,x_2 - 22506.1\,x_1\,v_2 \\ &+ 11032.5\,v_1\,x_2 - 13792.9\,v_1\,v_2 - 23070.8\,x_2\,v_2. \end{aligned}$$

The SMT solver returns UNSAT when certifying B according to (15), which gives a sound proof of safety. □

We further compare the performance of NNCChecker against the state-of-art work [5], which combines simulation and LP to find candidate barrier certificates. Those benchmark continuous dynamical systems come from the literate [4], [7]–[14], while the standard MPC control scheme is adopted to train DNN controllers. Here, the SOS problem is solved by the tool PENBMI while the LP problem is solved by Gurubi 9.0. For fairness, we use the same SMT solver dReal to identify real barrier certificates.

All the experiments were run on a Windows 10 desktop computer, with 3.4 GHz Intel Core i5 CPU and 16 GB RAM. The experimental results are listed in Table I.

TABLE I
PERFORMANCE ON BENCHMARKS

Examples	n_x	d_f	N_{Neu}	ϕ	NNCChecker						$Simulation + LP$ [5]		
					t_{Abs}	μ	Itr	t_{SOS}	t_{Query}	t_{Total}	t_{LP}	t_{Query}	t_{Total}
C1 [7]	4	3	1000	ReLU	13.27	0.2155	1	5.99	220.63	240.18	-	-	-
C2 [8]	2	3	100	Tanh	5.15	0.2095	2	4.51	0.42	14.83	1.01	0.36	1.55
C3 [8]	2	3	120	Sigmoid	5.54	0.1559	1	4.61	0.64	11.07	1.21	0.56	2.14
C4 [8]	4	3	200	Sigmoid	13.30	0.1971	1	5.46	5.73	24.77	-	-	-
C5 [9]	2	3	400	Sigmoid	5.24	0.5042	4	-	-	-	-	-	-
C6 [4]	2	1	500	Tanh	5.25	0.2637	1	4.56	1.03	11.11	-	-	-
C7 [10]	2	3	500	ReLU	5.49	0.5203	3	-	-	-	1.26	0.48	1.72
C8 [11]	3	3	600	ReLU	5.92	0.7449	1	5.35	59.76	71.31	1.64	21.36	23.94
C9 [12]	4	3	1500	ReLU	13.42	0.3798	2	12.05	234.95	507.42	-	-	-
C10 [11]	4	3	600	Tanh	11.93	0.3925	1	8.25	170.52	191.01	-	-	-
C11 [13]	2	2	200	ReLU	5.22	0.1363	1	4.61	0.15	10.26	1.14	0.31	2.56
C12 [14]	2	3	200	Tanh	4.99	0.1602	1	5.11	0.20	10.56	-	-	-

In Table I, n_x denotes the number of the state variables; $d_{\mathbf{f}}$ denotes the maximal degree of the polynomials in the vector fields; N_{Neu} denotes the number of neurons in the controller; ϕ denotes the type of activation functions; t_{Abs} denotes the time spent in building the approximation systems; μ is the bound of approximation errors; Itr records the number of iteration passed; t_{SOS} and t_{Query} records the average time taken by SOS solver and SMT solver in each iteration; t_{LP} records the time taken by LP solver and t_{Total} reports the total time spent in barrier certificates synthesis, respectively.

The experiments cover neural network controllers with three widely used activation functions: ReLU, Sigmoid and Tanh. The table shows that for the 12 examples, NNCChecker can successfully handle 10 of them while our opponent only solves 5. Our NNCChecker are more effective in synthesizing barrier certificates. It is also disclosed by the experiments that the relaxation that requires $B(x)$ to be decreasing over the whole domain has put too strong constraint on the generation process, and constraint refinement based on counterexamples and more trajectories does not take effects as expected, as such constraints are not always linearly independent to existing ones. Thus, simulation combined with LP seems not to be effective enough. Our NNCChecker failed in 2 examples, as it cannot find a polynomial approximation system good enough to enclose the original system while having barrier certificates separate the reachable set from unsafe region. From efficiency aspect, since SOS solver is much less efficient than LP solver, NNCChecker takes much more time in all the cases.

V. CONCLUSION

In this paper, we have presented an approach to iteratively generating barrier certificates for neural network controlled continuous systems. By abstracting a neural network with a polynomial with varied approximation errors, a polynomial approximation system is constructed whose barrier certificates act as the candidate BCs of the system under verification. A SMT solver is then used to identify real barrier certificates from them. We illustrate the proposed method by one case study and show its advantage in synthesizing barrier certificates against the state-of-art work by performing experiments on a set of benchmarks.

REFERENCES

[1] S. Dutta, X. Chen, and S. Sankaranarayanan, "Reachability analysis for neural feedback systems using regressive polynomial rule inference," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pp. 157–168.

[2] X. Sun, H. Khedr, and Y. Shoukry, "Formal verification of neural network controlled autonomous systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019*, pp. 147–156.

[3] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "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, HSCC 2019*, pp. 169–178.

[4] S. Prajna, A. Jadbabaie, and G. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1429, 2007.

[5] C. E. Tuncali, J. Kapinski, H. Ito, and J. V. Deshmukh, "Reasoning about safety of learning-enabled components in autonomous cyber-physical systems," in *Proceedings of the 55th Annual Design Automation Conference, DAC 2018, San Francisco, CA, USA, June 24-29, 2018*, 2018, pp. 30:1–30:6.

[6] T. Weng, H. Zhang, P. Chen, J. Yi, D. Su, Y. Gao, C. Hsieh, and L. Daniel, "Evaluating the robustness of neural networks: An extreme value theory approach," in *6th International Conference on Learning Representations(ICLR)*, 2018.

[7] Z. W. Jarvis-Wloszek, "Lyapunov based analysis and controller synthesis for polynomial systems using sum-of-squares optimization," Ph.D. dissertation, University of California, Berkeley, 2003.

[8] G. Chesi, "Computing output feedback controllers to enlarge the domain of attraction in polynomial systems," *IEEE Trans. Automat. Contr.*, vol. 49, no. 10, pp. 1846–1853, 2004.

[9] C. G., "Estimating the domain of attraction for non-polynomial systems via LMI optimizations," *Automatica*, vol. 45, no. 6, pp. 1536–1541, 2009.

[10] E. Aylward, P. Parrilo, and J. Slotine, "Stability and robustness analysis of nonlinear systems via contraction metrics and SOS programming," *Automatica*, vol. 44, no. 8, pp. 2163–2170, 2008.

[11] S. Ratschan and Z. She, "Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions," *SIAM Journal on Control and Optimization*, vol. 48, no. 7, pp. 4377–4394, 2010.

[12] M. A. B. Sassi and S. Sankaranarayanan, "Stabilization of polynomial dynamical systems using linear programming based on bernstein polynomials," *arXiv preprint arXiv:1501.04578*, 2015.

[13] A. Sogokon, K. Ghorbal, and T. T. Johnson, "Non-linear continuous systems for safety verification (benchmark proposal)," in *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, 2016.

[14] O. Bouissou, A. Chapoutot, A. Djaballah, and M. Kieffer, "Computation of parametric barrier functions for dynamical systems using interval analysis," in *Proceedings of the IEEE 53rd Annual Conference on Decision and Control (CDC)*. IEEE, 2014, pp. 753–758.