Safety Certification for Stochastic Systems via Neural Barrier Functions

Mathiesen, Frederik Baymler; Calvert, S.C.; Laurenti, L.

DOI
10.1109/LCSYS.2022.3229865

Publication date
2023

Document Version
Final published version

Published in
IEEE Control Systems Letters

Safety Certification for Stochastic Systems via Neural Barrier Functions

Frederik Baymler Mathiesen, Simeon C. Calvert, *Member, IEEE*, and Luca Laurenti

***Abstract*—Providing non-trivial certificates of safety for non-linear stochastic systems is an important open problem. One promising solution to address this problem is the use of barrier functions. Barrier functions are functions whose composition with the system forms a Martingale and enable the computation of the probability that the system stays within a safe set over a finite time horizon. However, existing approaches to find barrier functions generally restrict the search to a small class of functions, often leading to conservatism. To address this problem, in this letter, we parameterize barrier functions as neural networks and show that bound propagation techniques and linear programming can be successfully employed to find Neural Barrier Functions. Further, we develop a branch-and-bound scheme based on linear relaxations that improves the scalability of the proposed framework. On several case studies we show that our approach scales to neural networks of hundreds of neurons and multiple hidden layers and often produces certificates of safety that are tighter than state-of-the-art methods.**

***Index Terms*—Neural networks, system verification, stochastic systems, linear programming.**

I. Introduction

MODERN autonomous systems are inherently non-linear, incorporate complex feedback controllers, and are subject to uncertainty. Regardless of these complexities, autonomous systems are commonly employed for safety-critical applications, such as autonomous cars, where a failure can have catastrophic consequences. For these applications, formal quantification of probabilistic safety, defined as the probability that the system does not enter an unsafe region of the state space, is of paramount importance [1]. Despite the recent efforts [2], [3], guaranteeing probabilistic safety for non-linear stochastic systems still remains a particularly challenging problem. A promising approach is to use Stochastic Barrier Functions (SBFs), which allows one to verify temporal properties of a stochastic system without the need to explicitly evolve it over time [4]. Specifically, when composed with the system, a SBF forms a *c-martingale* [4]; consequently, martingale inequalities can be employed to compute (a lower bound on) probabilistic safety [5]. The main challenge with SBFs is to find a function that satisfies the conditions to be a SBF and that is expressive enough to return a non-vacuous lower bound on probabilistic safety. Existing approaches generally formulate the search for a barrier function as a convex optimization problem by restricting the barrier to a limited class of functions, generally exponential [6] or low-degree Sum-of-Squares (SoS) polynomials [7], which leads to conservative results. In this context, Neural Networks (NNs) hold great potential due to their universal approximation power and their training flexibility, yet no work has explored NNs to provide safety certificates for stochastic systems.

In this letter, we consider non-linear stochastic systems and propose a framework to train and certify NNs as SBFs; also called Neural Barrier Functions (NBFs). By relying on linear relaxation techniques for NNs [8], [9], we show that certifying that a NN is a barrier function can be reduced to the solution of a set of linear programs. Specifically, we partition the state space and leverage linear bound propagation [8] to find linear bounds to prove satisfaction of the barrier conditions. We propose a branch-and-bound method to reduce conservativity of our approach by adaptively refining the partition of the state space. Fig. 1(b) visualizes the relationship between the dynamics, a NBF, and the gap in linear relaxations to drive the branch-and-bound scheme. We experimentally investigate the efficacy of our framework on various examples including a non-linear vehicle dynamics model [10]. We find that our approach consistently outperforms state-of-the-art based on SoS optimization [7]. For instance, experiments on the vehicle dynamics model show that while the SoS approach returns a lower bound of safety of 0% or fails due to computational constraints, our method can certify a lower bound of 87%. In summary, this letter makes the following contributions: (i) introduce a novel framework to train NBFs for a given non-linear stochastic system, (ii) present a branch-and-bound scheme to compute a lower bound on the safety probability via NBFs based on the solution of linear programs, and (iii) on multiple benchmarks show that our framework can train and certify NBFs with multiple hidden layers of hundreds of neurons and substantially outperform state-of-the-art methods.

Manuscript received 16 September 2022; revised 6 November 2022; accepted 1 December 2022. Date of publication 16 December 2022; date of current version 29 December 2022. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Frederik Baymler Mathiesen.)*

Frederik Baymler Mathiesen and Luca Laurenti are with the Delft Center for Systems and Control, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: f.b.mathiesen@tudelft.nl; l.laurenti@tudelft.nl).

Simeon C. Calvert is with the Department for Transport and Planning, Delft University of Technology, 2628 CN Delft, The Netherlands (e-mail: s.c.calvert@tudelft.nl).

Digital Object Identifier 10.1109/LCSYS.2022.3229865

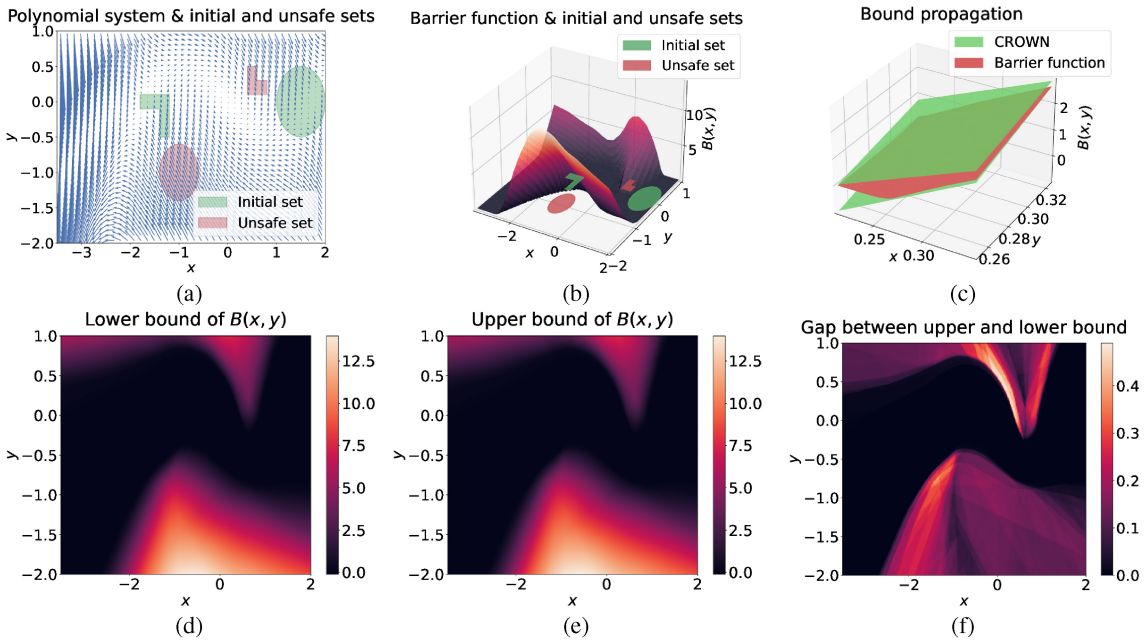


Fig. 1. Polynomial system with additive Gaussian noise adapted from [10] and corresponding NBF analyzed using bound propagation. **(a)** Plot of the nominal system, i.e., of the vector field without noise, and of initial set and unsafe region. **(b)** Neural Barrier Function (NBF) as synthesized by our framework relative to initial and unsafe sets. The composition of the NBF with the system forms a $c-$martingale, which allows us to use martingale inequalities to compute the probability that starting from the initial set, the system will avoid the unsafe set. **(c)** Linear relaxation of a NN for a single hyperrectangular region using CROWN [8]. We will rely on linear relaxations to prove that a NN is a NBF. **(d, e)** Bounds of the neural network computed partitioning the state space in a 320 $\times$ 320 grid and using CROWN in each partition. **(f)** The gap between the upper and lower bounds.

Related work: Safety guarantees for stochastic systems can generally be obtained with either abstraction-based methods based on model checking [2], [3], or barrier function-based methods based on energy-like functions, which avoid analyzing the flow of the system hence potentially improving scalability [4], [7]. SoS optimization is the state-of-the-art method for finding polynomial SBFs for discrete time [7], continuous time [7], and stochastic hybrid systems [4]. Furthermore, [6] has employed SoS to find exponential SBFs under the assumption that the safe set is an ellipsoid.

NNs for representing barrier or Lyapunov functions are collectively called neural certificates [11]. A major challenge is verifying that a neural network is a valid certificate. Existing methods are generally based on Satisfiability Modulo Theory (SMT) [10], Mixed-Integer Linear Programming (MILP) [11], and Lipschitz constants [12]. SMT and MILP-based certification are generally limited to very few neurons (< 20) in at most two hidden layers. On the other hand, the Lipschitz method is more scalable, but generally conservative [13]. To the best of our knowledge, [12], [14] are the only works that focus on neural certificates for stochastic systems. Different to this letter, they only consider asymptotic stability. Furthermore, to certify that a NN is a valid Lyapunov function, they either employ SMT, which leads to lack of scalability, or require the computation of the Lipschitz constant of both the underlying system and the NN, which leads to conservatism. In contrast, our approach does not require the computation of any Lipschitz constant and can scale to larger networks of hundreds of neurons and multiple hidden layers as we will show in Section IV. A novel approach to neural certification of systems is Neural System Level Synthesis where a closed loop system is synthesized to guarantee stability [15].

II. Problem Formulation

We consider a discrete-time stochastic system described by the following stochastic difference equation:

$$\mathbf{x}[k+1] = F(\mathbf{x}[k], \mathbf{v}[k]) \tag{1}$$

where $\mathbf{x}[k] \in \mathbb{R}^n$ is the state of the system at time k, and $\mathbf{v}[k]$ is an independent random variable distributed according to a distribution $p(v)$ over an uncertainty space $V \subseteq \mathbb{R}^n$. We use **bold** for random variables and *italic* for vectors, e.g., $\mathbf{x}$ vs x. The function $F : \mathbb{R}^n \times V \to \mathbb{R}^n$ is a continuous function representing the one-step dynamics of System (1). System (1) represents a general model of non-linear stochastic system, which also includes non-linear systems in closed loop with feedback controllers synthesized with standard control theory methods (e.g., LQR) as well as NNs.

Given an initial condition x_0, $\mathbf{x}[k]$ is a Markov process with a well-defined probability measure P [16, Proposition 7.45] generated by the noise distribution $p(v)$ such that for sets $X_0, X_{k+1} \subseteq X$ it holds that

$$\begin{aligned} P(\mathbf{x}[0] \in X_0) &= \mathbb{1}_{X_0}(x_0) \\ P(\mathbf{x}[k+1] \in X_{k+1} \mid \mathbf{x}[k] = x_k) &= T(X_{k+1} \mid x_k), \end{aligned} \tag{2}$$

where T is the stochastic kernel for the dynamical system, i.e., $T(X_{k+1} \mid x_k) \coloneqq \int_V \mathbb{1}_{X_{k+1}}(F(x_k, v))p(v)dv$ and $\mathbb{1}_{X_k}(x_k) = \begin{cases} 1 & \text{if } x_k \in X_k \\ 0 & \text{otherwise} \end{cases}$ is the indicator function for set X_k. In this

letter, we focus on verifying the safety of System (1) defined as the probability that for a given finite time horizon $H \in \mathbb{N}$, $\mathbf{x}[k]$ remains within a (measurable and bounded) safe set $X_s \subset \mathbb{R}^n$.

Problem 1 (Probabilistic Safety): Given a safe set $X_s \subset \mathbb{R}^n$, a finite time horizon $K = \{0, 1, \dots, H\}$, and an initial set of states $X_0 \subseteq X_s$, compute probabilistic safety defined as

$$P_s(X_s, X_0, H) = \inf_{x_0 \in X_0} P(\forall k \in K, \mathbf{x}[k] \in X_s \mid \mathbf{x}[0] = x_0).$$

Note that the assumption of a finite time horizon is often not limiting. In fact, if $\mathbf{v}[k]$ is additive with unbounded support, the probability of entering any unsafe region over an unbounded horizon is trivially 1. Furthermore, we stress that the distribution of $\mathbf{x}[k]$ is often analytically intractable, because $\mathbf{x}[k]$ is the result of iterative predictions over a non-linear function F, which is generally analytically intractable even in the case of Gaussian noise. Consequently, the computation of $P_s(X_s, X_0, H)$ generally requires approximations. Our approach is to rely on barrier functions parameterized as neural networks to compute a lower bound of P_s.

III. Probabilistic Safety via Neural Barrier Functions

Our framework to compute probabilistic safety for System (1) is based on Stochastic Barrier Functions (SBFs), which we parameterize as neural networks. Since this letter only focuses on SBFs, we sometimes refer to them as just barrier functions. In what follows, we first introduce SBFs (Section III-A) and then show in Section III-B how to verify that a neural network is a SBF for System (1) using the linear relaxation methods. Section III-B1 introduces a branch-and-bound scheme to refine the verification result, while controlling the required computational cost. Finally, in Section III-C we show how techniques commonly used for robust training of NNs can be used to train NBFs for System (1).

A. Stochastic Barrier Functions

SBFs rely on the theory of c-martingales[1] to show that a stochastic process does not exit a given safe set with high probability.

Definition 1 (Stochastic Barrier Function): Let $X_s \subset \mathbb{R}^n$, $X_0 \subseteq X_s$ and $X_u = \mathbb{R}^n \setminus X_s$ be respectively safe set, set of initial states, and unsafe set. Then, we say that a non-negative continuous almost everywhere function $B : \mathbb{R}^n \to [0, 1]$ is a stochastic barrier function for $\mathbf{x}[k]$ if there exists $\beta \geq 0$ and $\gamma \in [0, 1)$ such that

$$B(x) \geq 0 \qquad \forall x \in \mathbb{R}^n \tag{3a}$$
$$B(x) \geq 1 \qquad \forall x \in X_u \tag{3b}$$
$$B(x) \leq \gamma \qquad \forall x \in X_0 \tag{3c}$$
$$\mathbb{E}[B(F(x, \mathbf{v}))] \leq B(x) + \beta \qquad \forall x \in X_s. \tag{3d}$$

A barrier function is considered a valid certificate if Cond. (3a)-(3b) are satisfied. Fig. 2 illustrates Cond. (3a)-(3d).

[1]For the rest of the letter we will use β instead of c as is custom for stochastic barrier functions.

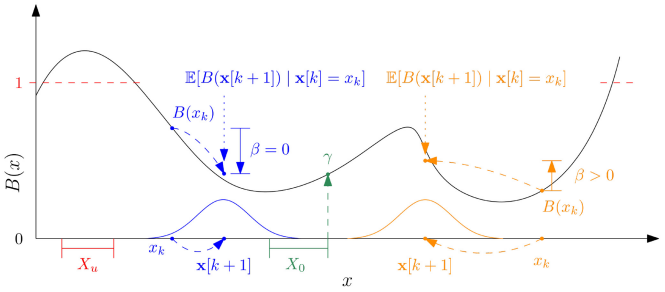


Fig. 2. A barrier function B is a non-negative function that is greater than 1 in the unsafe region X_u. $\beta \in \mathbb{R}_{\geq 0}$ is an upper bound on the expected increase in value after one time step of System (1) in the safe set $X_s = \mathbb{R}^n \setminus X_u$. γ is an upper bound of $B(x)$ for $x \in X_0$. Proposition 1 guarantees that for a finite horizon H it holds that $P_s(X_s, X_0, H) \geq 1 - (\gamma + \beta \cdot H)$.

Intuitively, as shown in Fig. 1(a), these conditions guarantee that the expectation of the composition of B with the dynamics of $\mathbf{x}$ does not grow by more than β in X_s. This allows us to bound $P_s(X_s, X_0, H)$ [5].

Proposition 1 [6]: Let B be a barrier function for $\mathbf{x}[k]$ and $H \in \mathbb{N}$ be a time horizon. Then, for $\varepsilon = \gamma + \beta \cdot H$ it holds that $P_s(X_s, X_0, H) \geq 1 - \varepsilon$.

Note that Cond. (3a)-(3d) allow for a SBF to return α and β such that $\varepsilon \geq 1$. However, in this case we obtain a trivial certificate $P_s(X_s, X_0, H) \geq 0$.

B. Neural Stochastic Barrier Functions

In this letter, we parametrize SBFs as Neural Networks (NNs), which we call Neural Barrier Functions (NBFs). How to train a NN to be a NBF will be the object of Section III-C. First, in what follows, we show how to check that a given NN B_θ with continuous activation functions, where θ represents the parameters (weights and biases), is a valid SBF for System (1), i.e., it satisfies Cond. (3a)-(3d). Our verification approach is based on building piece-wise linear functions that under- and over-approximate B_θ.

Definition 2 (Linear Relaxation): Linear relaxations of a continuous function $f : \mathbb{R}^n \to \mathbb{R}^1$ over a set $X \subseteq \mathbb{R}^n$ are two linear functions $\underline{A}x + \underline{b}$ and $\bar{A}x + \bar{b}$ with $\underline{A}, \bar{A} \in \mathbb{R}^{1 \times n}$ and $\underline{b}, \bar{b} \in \mathbb{R}^1$ such that $\underline{A}x + \underline{b} \leq f(x) \leq \bar{A}x + \bar{b}, \quad \forall x \in X$.

To find linear relaxations of a NN, given a region of the input, we employ CROWN [8]. In CROWN, non-linearities are sequentially relaxed by analytical expressions through input bounds to each non-linearity. Linear relaxations are propagated backwards through the NN, both for local input bounds to relax non-linearities and for the final relaxation. Together, this yields a complexity of $O(cw)$ where c is the number of non-linearities in the computation graph and w is the complexity of a forward pass. If a NN is composed with a continuous function, then CROWN-like techniques can still be applied on the composite computation graph to derive linear relaxations of the composed function. In particular, the continuous function can be treated as the first layer of neural network and perform backwards bound propagation [9].

Turning our attention back to the verification of B_θ, we start by assuming $B_\theta(x) \geq 0$ for all $x \in \mathbb{R}^n$, that is Cond. (3a),

which can be encoded by taking ReLU of or squaring the output. Similarly, Cond. (3b) can be trivially satisfied by assuming $B_\theta(x) = 1$ for all $x \in X_u$. Note that this is possible under the general assumption that the boundary of X_s has measure zero because of the requirement of B_θ to be continuous almost everywhere.

For verifying Cond. (3c), (3d), we start by partitioning X_s into a finite set of regions $Q = \{q_1, \ldots, q_{|Q|}\}$ and, using linear relaxation techniques, for each $q \in Q$ we can find matrices $\underline{A}_q, \bar{A}_q \in \mathbb{R}^{1\times n}$ and $\underline{b}_q, \bar{b}_q \in \mathbb{R}$ such that

$$\underline{A}_q x + \underline{b}_q \leq B_\theta(x) \leq \bar{A}_q x + \bar{b}_q, \quad \forall x \in q.$$

Then, the following lemma follows trivially

Lemma 1: Let $Q_{X_0} \subseteq Q$ be sets of regions such that $X_0 \subseteq \cup_{q\in Q_{X_0}} q$. Choose

$$\gamma = \max_{q\in Q_{X_0}} \max_{x\in q} \bar{A}_q x + \bar{b}_q.$$

Then, Cond. (3c) is satisfied.

Note that under the assumption that q is a convex polytope, which can always be enforced by the partition strategy, then Lemma 1 reduces to the solution of a set of linear programs. We now turn our attention to β, and consequently to the computation of the martingale condition, Cond. (3d). Unfortunately, due to the non-linearity of the functions involved, computing $\mathbb{E}[B_\theta(F(x, \mathbf{v}))]$ is analytically intractable. As a consequence, we again rely on computing local under- and over-approximations. In particular, consider finite partitions Q and Q_V respectively of the safe set X_s and of the uncertainty space V, and let $\tilde{Q} = Q \times Q_v$. Then, as discussed above for each $q \in Q$ and $\tilde{q} = (q, q_v) \in \tilde{Q}$ we can find row vectors $\underline{A}_q, \bar{A}_q, \bar{A}_{q_v} \in \mathbb{R}^{1\times n}$ and scalars $\underline{b}_q, \bar{b}_{\tilde{q}} \in \mathbb{R}$ such that

$$\forall x \in q, \quad \underline{A}_q x + \underline{b}_q \leq B_\theta(x) \tag{4}$$

$$\forall (x, v) \in \tilde{q}, \quad B_\theta(F(x, v)) \leq \bar{A}_q x + \bar{A}_{q_v} v + \bar{b}_{\tilde{q}}. \tag{5}$$

To explicitly encode Cond. 3b, we assume $\bar{A}_q = \bar{A}_{q_v} = 0$ and $\bar{b}_{\tilde{q}} = 1$ if $\{F(x, v) \mid (x, v) \in \tilde{q}\} \cap X_u \neq \emptyset$. The following theorem uses the above relaxations to bound $\mathbb{E}[B(F(x, \mathbf{v}))]$ and consequently find a lower bound on β.

Theorem 1: Let Q and Q_V respectively be partitions of X_s and V. For $\tilde{q} = (q_x, q_v) \in Q \times Q_V$ define

$$A_q = -\underline{A}_q + \sum_{q_v\in Q_V} \bar{A}_q \int_{q_v} p(v)dv,$$

$$b_q = -\underline{b}_q + \sum_{q_v\in Q_V} \bar{b}_{\tilde{q}} \int_{q_v} p(v)dv + \bar{A}_{q_v} \int_{q_v} vp(v)dv,$$

and choose $\beta \geq \max_{q\in Q} \max_{x\in q} (A_q x + b_q)$. Then, for any $x \in X_s$ it holds $\mathbb{E}[B_\theta(F(x, \mathbf{v}))] - B_\theta(x) \leq \beta$.

Proof: For $x \in q$ it holds that

$$\begin{aligned}\mathbb{E}[B_\theta(F(x, \mathbf{v})) \mid x] &= \sum_{q_v\in Q_V} \int_{q_v} B_\theta(F(x, v))p(v)dv \\ &\leq \sum_{q_v\in Q_V} \int_{q_v} \big(\bar{A}_q x + \bar{A}_{q_v} v + \bar{b}_{\tilde{q}}\big)p(v)\, dv.\end{aligned} \tag{6}$$

Rearranging the above bound and combining with $\underline{A}_q x + \underline{b}_q$, we get an upper bound $A_q x + b_q$ for $\mathbb{E}[B_\theta(F(x) + \mathbf{v})] - B_\theta(x)$ for all $x \in q$. It then follows that

$$\begin{aligned}&\max_{x\in X_s} (\mathbb{E}[B_\theta(F(x) + \mathbf{v})] - B_\theta(x)) \\ &\qquad \leq \max_{q\in Q_{X_s}} \max_{x\in q} \big(A_q x + b_q\big) \leq \beta.\end{aligned} \tag{7}$$

■

The computation of A_q and b_q in Theorem 1 requires the evaluation of integrals $\int_{q_v} p(v)\, dv$ and $\int_{q_v} vp(v)\, dv$. For various classes of distributions, such as Gaussian with diagonal covariance matrix or uniform distributions, these integrals can be computed in closed forms. Otherwise, numerical approximations may be required. Another challenge is if $p(v)$ has unbounded support then some $q_v \in Q_V$ are infinite in size and linear relaxations for these regions may not exist. However, we can use the fact that by construction $B_\theta(x) \leq 1$ for all $x \in \mathbb{R}^n$. Consequently, for any $x \in \mathbb{R}^n$ it holds that $\int_{q_v} B_\theta(F(x, v))p(v)\, dv \leq \int_{q_v} p(v)\, dv$ for any $x \in \mathbb{R}^n$.

1) A Branch and Bound Scheme for Verification: To improve the scalability of our verification framework, we adapt the branch-and-bound partitioning scheme of [17] to our setting. In particular, starting from a coarse partitioning of X, the method gradually refines it by splitting regions and pruning those that already satisfy the barrier conditions. For convenience, we assume that all regions q are hyperrectangles. We perform the branch-and-bound independently for Cond. (3c)-(3d). In what follows, we explain the partitioning scheme for Cond. (3c), Cond. (3d) follows analogously.

We start with a coarse initial partition Q_{X_0} of X_0. Then, as shown in Lemma 1, Cond. (3c) reduces to the computation $\max_{q\in Q_{X_0}} \min_{x\in q} \underline{A}_q x + \underline{b}_q$. As we start with a coarse partition initially our bounds may be very conservative. Consequently, we gradually refine Q_{X_0}. At each iteration we split all regions in Q_{X_0} and prune a region q if q does not influence the satisfaction of Cond. 3c and can be discarded. Specifically, we may prune q if either $q \cap X_0 = \emptyset$ or the maximum value of B in q is less than the smallest lower bound in another region $q' \in Q_{X_0}$. One iteration of splitting and pruning is shown in Fig. 3. Finally, we stop the partitioning when the gap between upper and lower bound for $\max_{x\in X_0} B_\theta(x)$ is less than a threshold $t_{gap} > 0$, that is if

$$\max_{q\in Q_{X_0}} \max_{x\in q} \bar{A}_q x + \bar{b}_q - \max_{q\in Q_{X_0}} \min_{x\in q} \underline{A}_q x + \underline{b}_q < t_{gap}.$$

Note that while the branch-and-bound scheme improves scalability in practice, its theoretical complexity is still exponential in the dimension of the state space.

C. Training Stochastic Neural Barrier Functions

We now describe the neural network training procedure, which is key to obtain a valid stochastic barrier function B_θ. As Cond. (3a)-(3d) need to hold over regions in the state space, the rationale behind our approach is to adapt certified adversarial training of NNs [18] to our setting. Our training procedure starts by sampling independently m training points from each

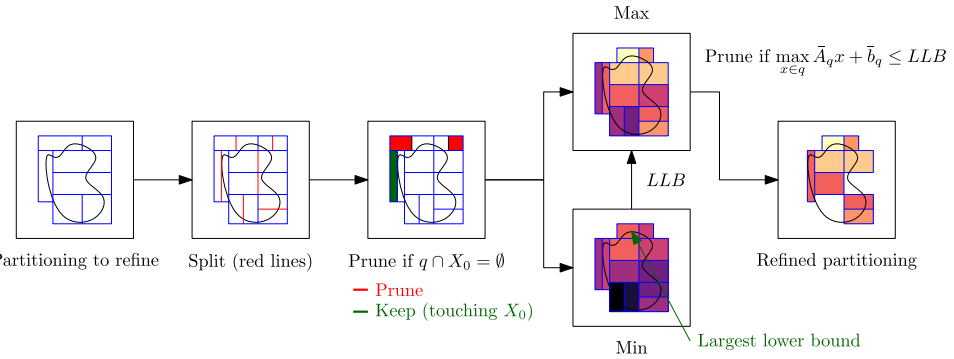


Fig. 3. One iteration of the automatic partitioning scheme with splitting and pruning for Cond. (3c). The set X_0 is shown as a black blob, and hyperrectangles $q \in Q_{X_0}$ are split and pruned. $LLB = \max_{q\in Q_{X_u}} \min_{x\in q} \underline{A}_q x + \underline{b}_q$ denotes the largest lower bound.

set X, X_u, X_0, and X_s. We denote the resulting data set respectively as $Q_X^{(m)}$, $Q_{X_u}^{(m)}$, $Q_{X_0}^{(m)}$, $Q_{X_s}^{(m)}$. Furthermore, l noise vectors $v_1, \ldots, v_l$ are independently sampled from $p(v)$. Then, the loss function $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = (1-\kappa)\mathcal{L}_v + \kappa(\gamma^{(m)} + \beta^{(m)} \cdot H)$$
$$\mathcal{L}_v = \sum_{x\in Q_X^{(m)}} \frac{\mathcal{R}(-B_\theta(x),\epsilon)}{m} + \sum_{x\in Q_{X_u}^{(m)}} \frac{\mathcal{R}(1-B_\theta(x),\epsilon)}{m}$$
$$\gamma^{(m)} = \max_{x\in Q_{X_0}^{(m)}} \mathcal{R}(B_\theta(x),\epsilon)$$
$$\beta^{(m)} = \max_{x\in Q_{X_s}^{(m)}} \mathcal{R}\left(\frac{1}{l}\sum_{j=1}^{l} B_\theta(F(x,v_j)) - B_\theta(x), \epsilon\right).$$

where $\kappa \in [0,1]$ weighs between a valid stochastic barrier function and tight probability bounds. For a function $f : \mathbb{R}^n \to \mathbb{R}$, $\mathcal{R}(f(x),\epsilon) = \max_{x':\|x-x'\|_\infty \leq \epsilon} \max(f(x'),0)$ is an upper bounds of $f(x)$ over an ϵ-hyperrectangle input set, which we employ to enforce the satisfaction of Cond. (3a)–(3d) on regions of the input space around the training points. To find a certified upper bound for $\mathcal{R}(f(x),\epsilon)$ we can employ CROWN [8]. Note that the analytical nature of CROWN allows employing standard auto-diff tools for taking the gradient of $\mathcal{L}$ wrt. θ. Intuitively, large values of ϵ will facilitate the task of finding a valid barrier function, while small ϵ may lead to tighter bounds due to the conservatism introduced by CROWN. With training progressing, we gradually reduce $\kappa \in [0,1]$ to shift from minimizing $\gamma^{(m)} + \beta^{(m)} \cdot H$ to minimizing $\mathcal{L}_v$ with the goal of first finding regions of the parameter space with tight probability bounds and then anneal the neural network to a valid barrier. Notice also that if B_θ is a stochastic barrier function, then $\mathcal{L}_v = 0$. However, $\mathcal{L}$ will in general not be zero even in this case. This is because of term $\kappa(\gamma^{(m)} + \beta^{(m)} \cdot H)$, which aims to minimize $\gamma^{(m)} + \beta^{(m)} \cdot H$, an upper bound on the probability of reaching the unsafe set.

IV. Experimental Evaluation

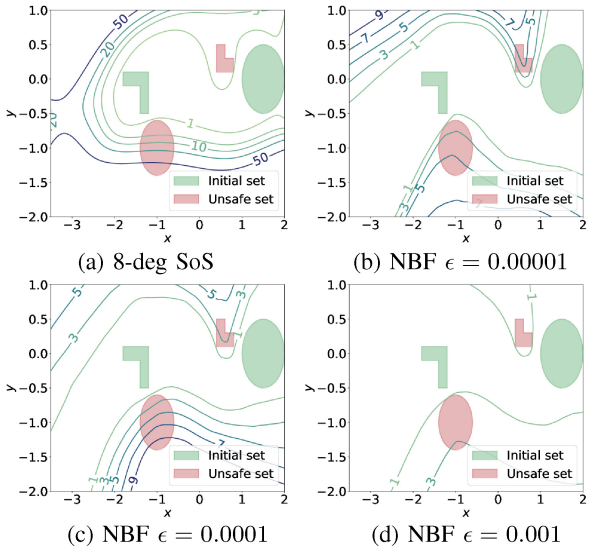
The framework is evaluated on three benchmarks: a 2-D linear system from [7], the 2-D polynomial system shown in Figure 1(b) from [4], and a 3-D, discrete-time, non-polynomial Dubin's car model [10]. To show the flexibility of our framework, for all systems we consider the same NBF architecture: a feed-forward neural network with 3 hidden layers with 128 neurons per layer and ReLU activation. Experiments are conducted on a computer with a Intel i7-6700k CPU, 16GB DDR4 RAM, Nvidia GTX 1060 GPU.[2] The optimization is done using ADAM with a learning rate of 1e−4, with training set batches of $m = 50$. We train the NBF for 60000 iterations.

TABLE I
Certified Lower Bound for P_s. Higher Is Better, and the Best Result for Each System Is Highlighted in **bold**. Cells With "–" Denotes That SoS Failed to Compute a Barrier. BaB Abbreviates Branch-and-Bound

Method	Linear	2-D polynomial	Dubin's car
SoS (4) [7]	0.690906	0.000000	-
SoS (8)	0.975079	0.232710	-
SoS (13)	0.998405	0.681383	-
SoS (15)	0.999761	-	-
Lipschitz [12]	0.824654	0.792392	0
NBF (grid)	0	0	0
NBF (BaB)	**0.999969**	**0.991664**	**0.870272**

To illustrate the efficacy of our framework, in Table I we compare the lower bound of P_s obtained with our model with a sum-of-squares optimization based approach [7], which arguably is the state-of-the-art for finding barrier functions for stochastic systems, a Lipschitz-method adapted from [12], and NBF with a grid-based partitioning (the grid considered for both the linear and polynomial system is 320x320, and for Dubin's car it is $40 \times 40 \times 60$). For all the benchmarks we consider SoS polynomials of order up to 15. For all our benchmarks it is possible to observe that our approach based on NBFs outperforms both SoS optimization and Lipschitz certification in terms of the tightness of the bounds. For instance, for the Dubin's car model, arguably the hardest example we consider due to its non-polynomial nature, SoS fails due to excessive memory requirements and Lipschitz only finds a trivial certificate of 0, while our framework obtains a lower bound of 0.87. In contrast, for the linear system, both SoS and

[2]Code for NBF, Lipschitz certification and SoS is available at https://github.com/DAI-Lab-HERALD/neural-barrier-functions.



(a) 8-deg SoS (b) NBF $\epsilon = 0.00001$

(c) NBF $\epsilon = 0.0001$ (d) NBF $\epsilon = 0.001$

Fig. 4. Levelset for **(a)** a SoS SBF and **(b, c, d)** NBFs with varying values of ϵ, all for the 2-D polynomial system. The NBFs are more flexible to capture complex shapes of the unsafe set and less sharply increasing in value compared to the SoS SBF. **(b, c, d)** show that larger ϵ results in a flatter surface, yielding a smaller β at the expense of a larger γ. (a) 8-deg SoS (b) NBF $\epsilon = 0.00001$ (c) NBF $\epsilon = 0.0001$ (d) NBF $\epsilon = 0.001$

our approach obtain a similar certified level of safety, but SoS is substantially faster (orders of minutes for the linear system), as our framework still requires to first train a neural network and then certify it (orders of few hours for all benchmarks as we used the same neural network architecture).

To understand the difference in certified safety, in Fig. 4 we study contour plots of the barrier functions for the 2-D polynomial example obtained with SoS and with NBF for different values of ϵ, where ϵ is the training parameter introduced in Section III-C. Note that the certified lower bound for P_s via Proposition 1 can be non-zero only in regions where $B(x) < 1$. Interestingly, we observe this region is significantly smaller for SoS compared to NBF for all ϵ, which is attributed to the reduced expressivity of the small-degree SoS polynomial. The differences in region sizes and the distances to the initial sets explain the result in Table I. Furthermore, we should stress that $\epsilon = 0.00001$ is the smaller value of ϵ for which our approach could find a valid barrier, illustrating the importance of this parameters in balancing between finding a valid barrier and obtaining high lower bounds of safety.

V. Conclusion

We propose an approach based on Neural Barrier Function (NBF) to compute probabilistic safety for stochastic systems. Novel algorithms to train NBFs are presented and it is shown that the problem of certifying a NBF for a given stochastic system reduces to the solution of a set of linear programs. The scalability of our framework is improved by a branch-and-bound approach. The method is evaluated on linear, polynomial, and non-polynomial systems, beating state-of-the-art methods on all systems, certifying previously intractable non-linear systems. Hence, this letter takes an important step towards the adoption of autonomous systems in safety-critical settings.

References

[1] W. Glover and J. Lygeros, "A stochastic hybrid model for air traffic control simulation," in *Proc. HSCC*, 2004, pp. 372–386.
[2] L. Laurenti, M. Lahijanian, A. Abate, L. Cardelli, and M. Kwiatkowska, "Formal and efficient synthesis for continuous-time linear stochastic hybrid processes," *IEEE Trans. Autom. Control*, vol. 66, no. 1, pp. 17–32, Jan. 2021.
[3] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 2, pp. 115–140, May 2019.
[4] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Trans. Autom. Control*, vol. 52, no. 8, pp. 1415–1428, Aug. 2007.
[5] H. J. Kushner, *Stochastic Stability and Control*. New York, NY, USA: Acad. Press, 1967.
[6] J. Steinhardt and R. Tedrake, "Finite-time regional verification of stochastic non-linear systems," *Int. J. Robot. Res.*, vol. 31, no. 7, pp. 901–923, 2012.
[7] C. Santoyo, M. Dutreix, and S. Coogan, "A barrier function approach to finite-time stochastic system verification and control," *Automatica*, vol. 125, Mar. 2021, Art. no. 109439.
[8] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Proc. NeurIPS*, 2018, pp. 4944–4953.
[9] K. Xu et al., "Automatic perturbation analysis for scalable certified robustness and beyond," in *Proc. NeurIPS*, 2020, pp. 1129–1141.
[10] A. Abate, D. Ahmed, A. Edwards, M. Giacobbe, and A. Peruffo, "FOSSIL: A software tool for the formal synthesis of Lyapunov functions and barrier certificates using neural networks," in *Proc. HSCC*, 2021, pp. 1–11.
[11] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural Lyapunov, barrier, and contraction methods," 2022, *arXiv:2202.11762*.
[12] M. Lechner, Đ. Žikelić, K. Chatterjee, and T. A. Henzinger, "Stability verification in stochastic control systems via neural network supermartingales," in *Proc. AAAI*, 2022, pp. 7326–7336.
[13] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 11427–11438.
[14] A. Abate, M. Giacobbe, and D. Roy, "Learning probabilistic termination proofs," in *Proc. CAV*, 2021, pp. 3–26.
[15] L. Furieri, C. L. Galimberti, and G. Ferrari-Trecate, "Neural system level synthesis: Learning over all stabilizing policies for nonlinear systems," 2022, *arXiv:2203.11812*.
[16] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete-Time Case*. Belmont, MA, USA: Athena Sci., 2004.
[17] R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, "A unified view of piecewise linear neural network verification," in *Proc. NeurIPS*, 2018, pp. 4795–4804.
[18] H. Zhang et al., "Towards stable and efficient training of verifiably robust neural networks," in *Proc. ICLR*, 2020, pp. 1–25.