Safety Guarantees for Neural Network Dynamic Systems via Stochastic Barrier Functions

Rayan Mazouz1, Karan Muvvala1, Akash Ratheesh1, Luca Laurenti2, Morteza Lahijanian1
1 Department of Aerospace Engineering Sciences, University of Colorado Boulder, USA
2 Delft Center for Systems and Control, Delft University of Technology, The Netherlands
{rayan.mazouz,karan.muvvala,akash.ratheesh,morteza.lahijanian}@colorado.edu
{l.laurenti}@tudelft.nl
* Equal Contribution

Abstract

Neural Networks (NNs) have been successfully employed to represent the state evolution of complex dynamical systems. Such models, referred to as NN dynamic models (NNDMs), use iterative noisy predictions of NN to estimate a distribution of system trajectories over time. Despite their accuracy, safety analysis of NNDMs is known to be a challenging problem and remains largely unexplored. To address this issue, in this paper, we introduce a method of providing safety guarantees for NNDMs. Our approach is based on stochastic barrier functions, whose role for safety is analogous to that of Lyapunov functions for stability. We first show a method of synthesizing stochastic barrier functions for NNDMs via a convex optimization problem, which in turn provides a lower bound on the system’s safety probability. A key step in our method is the employment of the recent convex approximation results for NNs to find piece-wise linear bounds, which allow the formulation of the barrier function synthesis problem as a sum-of-squares optimization program. If the obtained safety probability is above the desired threshold, the system is certified. Otherwise, we introduce a method of generating controls for the system that robustly minimize the unsafety probability in a minimally-invasive manner. We exploit the convexity property of the barrier function to formulate the optimal control synthesis problem as a linear program. Experimental results illustrate the efficacy of the method. Namely, they show that the method can scale to multi-dimensional NNDMs with multiple layers and hundreds of neurons per layer, and that the controller can significantly improve the safety probability.

1 Introduction

Safety is a major concern for autonomous dynamical systems, especially in safety-critical applications. Examples include UAVs, autonomous cars, and surgical robots. To this end, various methods are developed to provide safety guarantees for such systems [1,3]. Most of these methods, however, assume (simple) analytical models, which are often unavailable or too expensive to obtain due to complexity (nonlinearities) in real-world systems [4]. This has given rise to recent efforts to harvest the representational power of neural networks (NNs) to predict how the state of a system evolves over time, referred to as NN dynamic models (NNDMs) [5,6]. Despite their efficient and accurate representation of realistic dynamics, it has been difficult to make formal claims about the behavior of NNDMs over time. This poses a major challenge in safety-critical domains: how to provide safety guarantees for NNDMs?

In this paper, we propose a framework to provide safety certificates for NNDMs. The approach is based on stochastic barrier functions [7,8], whose role for safety is analogous to that of Lyapunov functions for stability, and which can guarantee forward invariance of a safe set. Specifically, we
show a method of synthesizing barrier functions for NNDMs via an efficient convex optimization method, which in turn provides a lower bound on the safety probability of the system trajectories. The benefit of our approach is that it does not require unfolding of the NNDM to compute this probability; instead, we base our reasoning on non-negative super-martingale inequalities to bound the probability of leaving the safe set. The key to this is to utilize the recent convex approximation results for NNs to obtain piece-wise linear bounds, and a formulation of the optimization problem that enables the use of efficient solvers. If the obtained safety probability is above a desired threshold, the system can be certified. For the cases that such safety certificates cannot be provided, we introduce a method of generating controls for the system, which robustly maximizes the safety probability in a minimally-invasive manner. We exploit the convexity property of the barrier function to formulate the control synthesis problem as a linear program. Experimental results on various NNDMs trained using state-of-the-art RL techniques [9–11] illustrate the efficacy of the method. Namely, they show that the method can scale to multi-dimensional models with various layers and hundreds of neurons per layer, and that the controller can significantly improve the safety probability.

In summary, this paper makes the following main contributions: (i) a method of computing a lower bound for the safety probability of NNDMs via stochastic barrier functions and efficient nonlinear optimization, thereby providing safety guarantees, (ii) a correct-by-construction control synthesis framework for NNDMs to ensure safety while being minimally-invasive, i.e., intervening only when needed, (iii) demonstration of the efficacy and scalability of the framework in benchmarks on various NNDMs with multiple architectures, i.e., multiple hidden layers and hundreds of neurons per layer.

Related Work

In recent years, NNs have been successfully employed to model complex dynamical systems [5, 6]. Such models use iterative noisy predictions of an NN to estimate the distribution of the system over time. They provide means of predicting state evolution of the systems that have no analytical form, e.g., soft robotics [12]. In addition, because of their ability to quickly unfold trajectories, NNDMs have been used to enhance controller training in Reinforcement Learning (RL) frameworks [9, 11]. In these works, specifically, a NNDM of the system is trained in closed-loop with an NN controller, which can be concatenated in a single NN representing the dynamics of the closed-loop system. Despite their popularity and increased usage, little work has gone to safety analysis of NNDMs. This paper aims to fill this gap.

Barrier functions provide a formal methodology to prove safety of dynamical systems [13, 14]. The state-of-the-art to find stochastic barrier functions is arguably to rely on Sum-of-Squares (SOS) optimization [8]. Barrier certificates obtained via SOS optimization have been studied for deterministic, uncontrolled nonlinear systems [15], and control-affine stochastic systems [16, 8], as well as for hybrid models [13, 14]. In addition, the control synthesis formulation in this paper relaxes the limitations of the existing SOS methods, which generally leads to a non-convex optimization problem [8], by reducing it to a simple linear program.

Many recent studies have focused on the verification of NNs. However, most of them only aim to provide certification guarantees against adversarial examples [17], with approaches ranging from SMT [18], and convex relaxations [19] to linear programming [20]. Such methods, because of their local nature, cannot be directly employed to perform safety analysis on the entire state space and over the trajectories as required for NNDMs. Another line of work focuses on infinite time horizon safety of Bayesian neural networks [21], or the verification of neural ODEs with stochastic guarantees [22]. They consider neural network controllers with known dynamics [23]. Other methods have been employed for trajectory-level properties of NNs based on solution of the Bellman equations [24], formal (finite) abstractions [25], and mixed integer programming [26], which do not support noisy dynamics. In contrast, our approach supports additive noise and only requires to check static properties on the NN by showing that there exists a function whose composition with the NN forms a stochastic barrier function.

2 Problem Formulation

In this work, we are interested in providing safety guarantees for dynamical systems that are described by NNs. Specifically, we consider the following discrete-time stochastic process whose time evolution is given by iterative predictions of an NN

$$x_{k+1} = f^w(x_k) + v_k, \quad k \in \mathbb{N}, \quad x_k, v_k \in \mathbb{R}^n, \quad (1)$$
Then, given threshold $\delta$, which describes the probability density of $f^w(x)$ with an NN controller. NNDMs such as the one in System (1) are increasingly employed in robotics for both model representation and NN controller training. These models are often obtained by state-of-the-art model-based RL techniques (e.g., [28]) or perturbations (step response) of the actual system (e.g., [12]).

The evolution of System (1) can be characterized by the predictive posterior distribution $p_k(X|\bar{x})$, which follows that, for a given initial condition $x_0 = x_0 \in \mathbb{R}^n$, $x_k$ is a Markov process with a well-defined probability measure $\Pr$ uniquely generated by the stochastic kernel $T$. From the definition of $T$ it follows that, for a given initial condition $x_0 = x_0 \in \mathbb{R}^n$, $x_k$ is a Markov process with a well-defined probability measure $\Pr$ uniquely generated by the stochastic kernel $T$ [27] Proposition 7.45]. For $X_0, X_k \subseteq \mathbb{R}^n$, this probability measure is inductively defined as

\[
\Pr[x_0 \in X_0] = 1_{X_0}(x_0), \quad \Pr[x_k \in X_k | x_{k-1} = x] = T(X_k | x),
\]

where indicator function $1_{X_0}(x_0) = 1$ if $x_0 \in X_0$, otherwise 0. The definition of $\Pr$ allows one to make probabilistic statements over the trajectories of System (1). In this work, we are particularly interested in two main problems: safety certification and safe controller synthesis for System (1).

**Problem 1** (Safety Certificate). Let $X_s \subseteq \mathbb{R}^n$ and $X_0 \subseteq X_s$ be basic closed semi-algebraic sets representing respectively the safe set and the set of initial states. Denote the probability that the system initialized in $X_0$ remains in the safe set in the next $N \in \mathbb{N}$ steps by

\[
P_s(X_s, X_0, N) = \Pr[\forall x_0 \in X_0, \forall k \leq N, x_k \in X_s].
\]

(2)

Then, given threshold $\delta_s \in [0, 1]$, provide a safety certificate for System (1) by proving that $P_s(X_s, X_0, N) \geq \delta_s$.

**Problem 2** (Safe Control Synthesis). Let $X_s \subseteq \mathbb{R}^n$ and $X_0 \subseteq X_s$ be basic closed semi-algebraic sets representing respectively the safe set and the set of initial states, and $N \in \mathbb{N}_{>0}$ a time horizon. Then, given threshold $\delta_s$, synthesize a feedback controller $\pi : \mathbb{R}^n \rightarrow \mathbb{R}^c$ such that $u_k = \pi(x_k)$ guarantees safety of System (1).

**Remark 1.** There may exist many possible controllers that are solutions to Problem 2. Since System (1) represents a closed-loop system with a given controller, one may want to specifically synthesize a safety controller that is “minimally-invasive,” i.e., it minimally intervenes to guarantee safety. In Section 5 we elaborate on this notion and introduce methods of generating such controllers.

**Overall Approach** Our approaches to Problems 1 and 2 are based on (stochastic) barrier functions. Specifically, we aim to synthesize barrier functions that can serve as a safety certificate by providing a lower bound for $P_s$. To achieve this, we rely on recent results for local convex relaxation of an NN to find piece-wise linear over- and under-approximations of $f^w$. This allows us to formulate the problem of finding a stochastic barrier function for System (1) as an SOS optimization problem.
Our (minimally-invasive) control synthesis method (Problem 2) focuses on injecting controls to System (3) only in the regions of the state space that potentially contribute to unsafety of the system. To this end, we exploit two main properties of the barrier function: the required (super-) martingale property and the convexity property of SOS polynomials. The former allows us to design a method of identifying the “unsafe” regions, and the latter enables us to formulate a Linear Programming (LP) optimization problem to find controllers that robustly minimize the probability of unsafety.

**Remark 3.** For ease of presentation, $g$ in (3) is taken to be a constant, however, we note that the proposed approach can also handle $g$ being a continuous function of $x$, or even a NN. For the latter, we introduce an interval bound propagation approach in Section 4.7 that allows the computation of upper and lower bounds on $g(x)$, which in turn can be incorporated in the LP optimization formulation in Section 5 for control synthesis (Problem 2) in a straightforward manner.

### 3 Preliminaries

In this section, we provide a brief background on stochastic barrier functions and non-negative polynomials, which are central components of our framework.

#### 3.1 Stochastic Barrier Functions

Consider stochastic process $x_{k+1} = F(x_k, v_k)$, where $x_k \in X \subseteq \mathbb{R}^n$, $v_k \in V \subseteq \mathbb{R}^m$ is a well-defined stochastic process, and $F: X \times V \rightarrow X$ is a locally Lipschitz continuous function. Then, given initial set $X_0 \subseteq X$, safe set $X_s \subseteq X$, and unsafe set $X_u \subseteq X$, a twice differentiable function $B : X \rightarrow \mathbb{R}$ is called a stochastic barrier function or simply barrier function if the following conditions hold for $\alpha \geq 1$ and $\beta, \eta \in [0, 1]$:

\[
\begin{align*}
B(x) &\geq 0 & \forall x \in X \\
B(x) &\leq \eta & \forall x \in X_0 \\
B(x) &\geq 1 & \forall x \in X_u \\
E[B(F(x, v)) \mid x] &\leq B(x)/\alpha + \beta & \forall x \in X_s
\end{align*}
\]

If such a $B$ exists, then for any $N \in \mathbb{N}_{\geq 0}$, it follows, assuming $\alpha = 1$ without loss of generality, that

\[
\Pr[\forall k \leq N, \ x_k \in X_s \mid x_0 \in X_0] \geq 1 - (\eta + \beta N).
\]

Intuitively, Conditions (4a)-(4d) allow one to use non-negative super-martingale inequalities to bound the probability that the system reaches the unsafe region. This results in the probability bound in (5) (see [29] Chapter 3, Theorem 3). A major benefit of using barrier functions for safety analysis is that while Conditions (4a)-(4d) are static properties, they allow one to make probabilistic statements on the time evolution of the system without the need to evolve the system over time.

#### 3.2 Non-negative Polynomials

Synthesis of barrier functions can be formulated as a nonlinear optimization problem. To solve it efficiently via convex programming, we formulate the problem using SOS polynomials, which is a sufficient condition for polynomial non-negativity [30]. This formulation allows the use of efficient semidefinite programming solvers, which have polynomial worst-case complexity [31].

**Definition 1** (SOS Polynomial). A multivariate polynomial $\lambda(x)$ is a Sum-Of-Squares (SOS) for $x \in \mathbb{R}^n$ if there exists some polynomials $\lambda_i, i = 1, \ldots, r$, for some $r \in \mathbb{N}$, such that $\lambda(x) = \sum_{i=1}^{r} \lambda_i^2(x)$. If $\lambda(x)$ is an SOS, then $\lambda(x) \geq 0$ for all $x \in \mathbb{R}^n$. The set of all SOS polynomials is denoted by $\Lambda$.

Now, consider a basic closed semi-algebraic set $X = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0 \ \forall i \in \{1, \ldots, l\}\}$, which is defined as the intersection of $l$ polynomial ($h_i(x)$) inequalities. Then, Proposition 1 allows one to enforce non-negativity on these semi-algebraic sets.

**Proposition 1** (Putinar’s Certificate, [32]). Let $X \subseteq \mathbb{R}^n$ be a compact basic semi-algebraic set defined as $X = \{x \in \mathbb{R}^n \mid h_i(x) \geq 0 \ \forall i \in \{1, \ldots, l\}\}$, where $h_i(x)$ is a polynomial. Then, polynomial $\gamma(x)$ is non-negative on $X$ if, for some $\lambda_0(x), \lambda_i(x) \in \Lambda$, $\gamma(x) = \lambda_0(x) + \sum_{i=1}^{l} \lambda_i(x)h_i(x)$. 
As commonly practiced in the literature, e.g., [15, 8, 37], we restrict our search for a \((4b)-(4d)\) iff \(\eta < s\) satisfying Conditions \((4b)-(4d)\) for \(f\) to be a valid candidate barrier function for System \((1)\). That is, there does not exist a linear function \(B\) that is Linear Relaxation based Perturbation Analysis (LiRPA) algorithms, which perform a symbolic back propagation followed by a forward evaluation of the bounds [35, 36, 19]. Such LiRPA-based algorithms produce tighter bounds but can be slow. In our implementation, we use the off-the-shelf toolbox [36] to compute linear bounds for \(B\). Recall that a valid barrier function \(B(x)\) must satisfy Conditions \((4b)-(4d)\). With the above local linear bounds for \(f^w\), Condition \((4d)\) can be redefined over the partitioned regions in \(Q\). That is, Condition \((4d)\) can be replaced with the following constraints: for each \(q \in Q\),

\[
E[B(f^w_q(x) + v) \mid x] \leq B(x)/\alpha + \beta \quad \text{and} \quad f^w_q(x) \leq f^w_q(x) \leq f^w_q(x) \quad \forall x \in q.
\]

In this formulation, \(f^w_q(x)\) is a (free) variable that is bounded by two linear functions, avoiding the non-convexity of \(f^w_q(x)\). Consequently, an efficient evaluation of the constraint becomes feasible.

**Remark 4.** Note that there are in fact three constraints in (7) (one super-martingale condition on \(B\) and two bounds on \(f^w_q(x)\)). If they are treated separately, it leads to a conservative barrier function because each constraint needs to hold for all \(x \in q\) separately. A less conservative approach is to pose all constraints for the same \(x \in q\) at the same time. In Theorem 7, we show how this can be achieved.

As commonly practiced in the literature, e.g., [15, 8, 37], we restrict our search for a \(B(x)\) to the class of polynomial functions. Then, the degree of the polynomial becomes a design parameter. In the following lemma, we show that the degree of the polynomial must be at least 2 for the polynomial to be a valid candidate barrier function for System \((1)\). That is, there does not exist a linear function satisfying Conditions \((4b)-(4d)\) for \(\eta < 1\), which implies that the lower bound in (5) is trivially 0.

**Lemma 1.** For \(k \in \mathbb{R}^n\) and \(c \in \mathbb{R}\), consider linear function \(B(x) = k^T x + c\). Then, \(B(x)\) satisfies Conditions \((4b)-(4d)\) iff \(k = 0\) and \(c = 1\).

**Proof.** There are two cases to consider: (i) \(k = 0\): in this case \(B(x) = c\), which satisfies Conditions \((4b)-(4d)\) iff \(c = \eta = 1\). (ii) \(k \neq 0\): in this case, due to the linearity of \(B(x)\), there always exists \(x' \in \mathbb{R}^n\) such that \(B(x') < 0\), violating Condition \((4a)\).
Having established that the polynomial degree must be at least 2, we further restrict $B(x)$ to SOS polynomials. Theorem 1 shows a formulation of the barrier function synthesis problem as an SOS program, which can be solved in polynomial time. In particular, the theorem shows how to formulate Conditions (4b)–(4c) and (7) as SOS constraints.

**Theorem 1 (NNDM Barrier Certificate).** Consider SOS polynomial function $B(x)$, and safe set $X_{s} = \{ x \in \mathbb{R}^{n} \mid h_{s}(x) \geq 0 \}$, initial set $X_{0} = \{ x \in \mathbb{R}^{n} \mid h_{0}(x) \geq 0 \}$, unsafe set $X_{u} = \mathbb{R}^{n} \\setminus X_{s} = \{ x \in \mathbb{R}^{n} \mid h_{u}(x) \geq 0 \}$, and partition region $Q = \{ x \in \mathbb{R}^{n} \mid h_{q}(x) \geq 0 \}$ for all $q \in Q$. Let $L_{q}(x)$, $L_{0}(x)$ and $L_{\alpha}(x)$ be vectors of SOS polynomials with the same dimension as $h_{s}$, $h_{0}$, and $h_{u}$, respectively. Likewise, let $L_{q,x}(x)$ and $L_{q,y}(x)$ be vectors of SOS polynomials with the same dimension as $h_{q}$. Then, a stochastic barrier certificate $B(x)$ for System (1) with time horizon $N \in \mathbb{N}_{\geq 0}$ can be obtained by solving the following SOS optimization problem for $\eta, \beta \in [0, 1]$:

$$\min_{\beta, \eta} \eta + \beta N \quad \text{subject to:}$$

$$B(x) \in \Lambda,$$  

$$- B(x) - L_{q}^{T}(x)h_{q}(x) + \eta \in \Lambda,$$  

$$B(x) - L_{q}^{T}(x)h_{q}(x) - 1 \in \Lambda,$$  

$$- E[B(y + v) \mid x] + B(x)/\alpha + \beta - L_{q}^{T}(x)h_{q}(x) - L_{q,y}(x)(\bar{f}_{q}(x) - y)(y - f_{q}(x)) \in \Lambda \quad \forall q \in Q$$  

which guarantees safety probability $P_{s}(X_{s}, X_{0}, N) \geq 1 - (\eta + \beta N)$.

**Proof.** It suffices to show that if $B$ satisfies Constraints (8a)–(8d), then Conditions (4a)–(4d) are satisfied. As Constraint (8a) guarantees that $B$ is a SOS polynomial, hence non-negative by definition, Condition (4a) holds. By Corollary 1 it holds that if Constraints (8b)–(8c) hold, then $B$ is respectively smaller that $\eta$ in $X_{0}$ and greater than 1 in $X_{u}$. What is left to show is that Constraint (8d) guarantees the satisfaction of Condition (4d). This can be done as follows. For every region $q \in Q$, the expectation term in Condition (4d) can be expressed as $E[B(y + v) \mid x]$, where $y$ is bounded by under-approximation $\underline{f}_{q}(x)$ and over-approximation $\bar{f}_{q}(x)$ of $f^{w}(x)$ in System (1) for all $x \in q$.

Then by Corollary 1, Constraint (8d) is obtained for each $q$. Note that $B(x)$ is a polynomial, so if $E[B(y + v) \mid x]$ is also a polynomial, the condition is a valid SOS constraint. Given that $y + v$ is linear, $E[B(y + v) \mid x]$ is a sum of monomials in terms of $y$ and expectation moments $E[v^{d}]$, where $d \geq 0$. Since, random variable $v$ has a normal distribution, $E[v^{d}]$ is constant. Therefore, $E[B(y + v) \mid x]$ is a polynomial only in terms of (components of) $y$.

**Remark 5.** We note that Condition (8d) uses linear equations in (6) to bound $f^{w}(x)$ for all $x \in q$. Removing the dependence on $x$, by replacing the equations for $\bar{f}_{q}(x)$ and $\underline{f}_{q}(x)$ with their extreme values in region $q$, relaxes the optimization problem and results in a speed-up. It is however an additional over-approximation, and hence the obtained safety probability bound is strictly smaller than the one generated by using the actual linear equations for $\bar{f}_{q}(x)$ and $\underline{f}_{q}(x)$. This is the classical “efficiency versus accuracy” trade-off. In our implementation, we compare the two approaches.

## 5 Minimally-invasive Barrier Controller Synthesis

Here, we focus on Problem 2. The assumption is that a barrier function $B(x)$ is synthesized for System (1) via Theorem 1 which has a safety probability that is lower than threshold $d_{s}$. Hence, our goal is to design a feedback controller for System (3) that guarantees safety. Ideally, this controller is minimal in the interruptions it causes to the system evolution. Below, we first provide a definition for a minimally-invasive controller, and then show how it can be synthesized.

Recall that a feedback controller $\pi : \mathbb{R}^{n} \rightarrow U$ is a function that assigns a control value to each state. Let $X_{\pi} \subseteq X_{s}$ be the set of states, to which $\pi$ assigns non-zero control values, i.e., $\pi(x) \neq 0$ for all $x \in X_{\pi}$. Further, we denote by $V(X)$ the volume of set $X \subset \mathbb{R}^{n}$. Then, we say a feedback controller $\pi$ is minimally-invasive if it minimizes $V(X_{\pi})$. Moreover, we say $\pi$ is $\epsilon$-minimally-invasive if $V(X_{\pi}) - \min_{\pi'} V(X_{\pi'}) \leq \epsilon$. In this work, we seek $\epsilon$-minimally-invasive controller with an arbitrary small $\epsilon$. To this end, we make the observation that if the discretization of $X_{s}$ discussed in Section 4.1...
is uniform, \(V(X_\pi)\) for a given \(\pi\) is directly proportional to the number of discrete regions in \(Q\) to which \(\pi\) applies a non-zero control. We denote by \(Q_\pi\) the set of such regions. Then, a feedback controller \(\bar{\pi}\) that minimizes \(|Q_\pi|\) is \(\varepsilon\)-minimally-invasive, and as the discretization becomes finer, \(\epsilon\) monotonically approaches zero. Therefore, given barrier function \(B(x)\), our first goal is to identify the minimal number of regions in \(Q\) that require a non-zero controller to guarantee safety. Recall that, given \(B(x)\), we require the safety probability to be \(1 - (\eta + \beta N)\geq \delta_s\). While \(\eta\) is related to the value of \(B(x)\) in the initial set, \(\beta\) is a compensation needed to turn a sub-martingale inequality to a super-martingale inequality (Condition (3d) or (4d)). We can compute this compensation for each region \(q\in Q\) by evaluating Condition (3d) or (4d). Let \(\beta_q\) denote the computed compensation term for region \(q\). Then, region \(q\) requires a non-zero controller if \(\beta_q > (1 - \delta_s - \eta)/\delta_s\).

We now turn our attention to designing a controller that reduces \(\beta_q\). In our approach, we tap directly into the convex nature of the SOS polynomial \(B(x)\) (SOS-convex). Let \(\bar{\pi}\) denote a controller that drives System (3) to a point closest to the minimum point of \(B(x)\), also robustly minimizes \(\beta_q\).

**Lemma 2.** Given \(B(x)\), let \(x^* = \arg \min_x B(x), x' = f^u(x) + gu + v\) as in System (3), and \(\beta_q \geq E[B(x') | x] - B(x)\) for all \(x\) in region \(q\subseteq X_s\). Then, the controller that minimizes \(\|x' - x^*\|_1\), where \(x' = f^u(x) + gu\), also robustly minimizes \(\beta_q\), i.e., minimizes an upper bound of \(\beta_q\).

**Proof.** Given that \(B\) is a polynomial and \(x'\) is affine in both \(u\) and \(v\), we can write \(E[B(x') | x] = B(E[x' | x]) + \xi\), where \(\xi\), called Jensen’s gap [38], is a non-negative polynomial of \(x, u, v\), and moments \(E[v^d]\), where \(2 \leq d \leq m\) is an even integer. This polynomial can always be upper bounded by a constant \(\xi \geq 0\) for bounded \(x, u\) and \(v\). Then, with \(x' = E[x']\), the min \(\beta_q\) is upper bounded by

\[
\min_u \beta_q \leq \min_{x \in q} \max_u (B(x') - B(x') + \xi) \leq \min_{x \in q} \max_u B(x') - \min_{x \in q} \xi.
\]

The second inequality holds because \(B\) is a non-negative function resulting from a SOS optimization, and is thus SOS-convex [39]. Since \(B(x')\) monotonically decreases as \(x' \to x^*\), \(\max_{x \in q} B(x') \leq x\) is minimized by \(\arg \min_u \|f^u(x) + gu - x^*\|_1\) \(\forall x \in q\).

We note that, in the proof of Lemma 2, the moments \(E[v^d]\) appear in every monomial of polynomial \(\xi\) (Jensen’s gap). Hence, for \(v\) with variance less than 1, Jensen’s gap is very small and vanishes as the variance becomes smaller, making the controller in the lemma optimal. Furthermore, even though Lemma 2 uses \(L_1\) norm for \(\|x' - x^*\|_1\), the statement also holds for \(L_2\) and \(L_\infty\) norms. We specifically use \(L_1\) norm to be able to compute the controller via an LP as stated in Theorem 2.

**Theorem 2 (Controller Synthesis).** Consider convex barrier function \(B(x)\) and its minimum argument \(x^*\) as specified in Lemma 2 and linear bounds \(f_q(x)\) and \(\bar{f}_q(x)\) for \(f^u(x)\) in region \(q\subseteq R^n\) as in (6). A controller that robustly minimizes \(\beta_q\) for System (3) is the solution to the following LP optimization problem, where \(\theta = (\theta_1, \theta_2, \ldots, \theta_n) \in R^n\), and the vector inequality relations are element-wise:

\[
\min_{\theta_i, z', y, u} \sum_{i=1}^n \theta_i \quad \text{subject to:}
\]

\[
y - x^* \leq \theta \quad (10a) \quad y \geq f_q(z) + gu \quad (10d)
\]

\[
x^* - y \leq \theta \quad (10b) \quad y \leq \bar{f}_q(z) + gu \quad (10e)
\]

\[
z = z' - z'' \quad (10c) \quad z', z'' \geq 0, \quad z \in q, \quad u \in U \quad (10f)
\]

**Proof.** By Lemma 2 it holds that to robustly minimizes \(\beta_q\), it suffices to minimize \(\|x' - x^*\|_1\). This norm is the sum of \(n\) absolute values, each of which can be set to \(\theta_i\) with linear Constraints (10a) and (10b). Hence, the minimization of \(\|x' - x^*\|_1\) is equivalent to minimization of \(\sum_{i=1}^n \theta_i\) subject to (10a) and (10b). The bounds on the dynamics of System (3) can be expressed as linear Conditions (10d) and (10e), each of which has to hold for all \(z \in q\), i.e., \(z\) is a free variable. To turn the optimization into an LP, \(z\) is expressed as a difference of two non-negative decision variables \(z'\) and \(z''\) in Constraint (10c).

**Remark 6.** Note that the controller obtained via the LP is a vector of scalars for each region \(q\). This leads to a feedback controller \(\pi\) over the regions \(q \subseteq Q\). It is possible to express \(u\) as SOS polynomials, which would be a feedback controller over \(x\). That could potentially provide a tighter bound for \(\beta_q\) at the cost of extra computation since SOS optimization is more expensive than LP.
5.1 Control Synthesis Algorithm

Our framework for computing a barrier certificate as well as minimally-invasive controllers for NNDDMs is outlined in Algorithm 1. First, the safe set \( X_s \) is uniformly partitioned into a finite set of regions \( Q \), and bounds of \( f^w \) are computed as piece-wise linear functions per Section 4.1. Then, for a given polynomial degree \( m \geq 2 \), a barrier certificate along with its corresponding safety probability bound are computed in accordance with Theorem 1. If this probability is greater than or equal to \( \delta_s \), the NNDM is certified, and the algorithm terminates. Otherwise, it iteratively searches for \( \epsilon \)-minimally-invasive controllers. In each iteration, upper bound \( \eta \) is set on \( \eta \) (initially with its largest possible value \( 1 - \delta_s \) and then reduced by \( \Delta \eta > 0 \)), and a new barrier function is computed with this bound, where the objective function is minimization of \( \beta \) to ensure \( \epsilon \)-minimally-invasive controllers. Then, for each \( q \in Q \), \( \beta_q \) is checked against the threshold. If exceeding, a controller is computed for \( q \) in accordance with Theorem 2. This process repeats until a satisfactory \( P_s \) is obtained or \( \eta < 0 \).

We highlight two important properties of this algorithm: it clearly terminates in finite time, and if \( P_s \geq \delta_s \), the synthesized controller is correct-by-construction.

**Algorithm 1:** NNDDM Controller Synthesis

**Input:** NN \( f^w \), initial set \( X_0 \), safe set \( X_s \), noise covariance \( R \), polynomial degree \( m \), threshold \( \delta_s \), and step size \( \Delta \eta \).

**Output:** feedback controller \( \pi \), and safety probability bound \( P_s \).

1. \( Q, f^w, T_q \leftarrow \text{PartitionAndComputeBounds}(X_0, f^w), \quad k \leftarrow 0 \)
2. \( \eta, \beta, B(x) \leftarrow \text{ComputeBarrierCertificate}(Q, f^w, T_q, R, m) \) \quad // Theorem 1
3. while \( P_s < \delta_s \) and \( \eta > 0 \) do
4. \( \eta \leftarrow 1 - (\delta_s - k \Delta \eta), \quad k \leftarrow k + 1 \) \quad // set upper bound on \( \eta \)
5. \( \eta, \beta, B(x) \leftarrow \text{ComputeBarrierCertificate}(Q, f^w, T_q, R, m, \eta) \) \quad // Theorem 1
6. for \( q \in Q \) do
7. \( \pi \leftarrow (q, 0) \)
8. if \( \beta_q > (1 - \delta_s - \eta)/N \) then
9. \( \pi \leftarrow (q, u_q) \) \quad \text{computeControl}(\arg \min B(x), q, f^w, T_q, U) \quad // Theorem 2
10. \( \beta_q \leftarrow \text{UpdateBeta}(B(x), u_q) \) \quad // evaluate Condition 8d
11. \( P_s \leftarrow 1 - (\eta + N \max_{q \in Q} \beta_q) \)
12. return \( \pi, P_s \)

6 Case Studies

We demonstrate the efficacy of our framework on several case studies. We first show that our verification method is able to produce non-trivial safety guarantees for NNDDMs trained using imitation learning techniques [40] from data gathered by rolling out an RL agent trained with state-of-the-art RL techniques [41]. We then show that our control approach significantly increases the safety probability.

**Models** We trained the Pendulum (2D), Cartpole (4D), and Acrobot (6D) agents from the OpenAI gym environment, as well as 4D and 5D Husky robot models. For the OpenAI models, we collected input-output states of the system under an expert controller, either as a look-up table or an NN controller. We picked the best controller available for the agents from the OpenAI Leaderboard [41].

The Husky models were trained to move from one point to another while staying within a lane. We initially trained a NNDDM along with an NN controller on the trajectory data generated by PyBullet (physics simulator [42]) as outlined in [9]. We then used these NNs to generate input-output data and trained closed-loop NNDDMs as specified above. In total, we trained 10 fully-connected multi-layer perceptron NNDDM architectures with up to 5 hidden layers and 512 neurons per layer, each with ReLU activation function. The dimensionality and architecture information is provided in Table 1.

The state space of the Pendulum consists of the pole angle \( \theta \) and angular velocity \( \dot{\theta} \), with the safe set defined as \( [-\pi/15, \pi/15] \times [-1, 1] \), and the initial set as \( \theta_0 \in [-\pi/36, \pi/36] \). For the Cartpole, the state space consists of the cart position \( x \), velocity \( \dot{x} \), pole angle \( \theta \), and angular velocity \( \dot{\theta} \). The safe sets for cart position are defined as \( x \in [-1, 1] \) and \( \theta \in [-\pi/15, \pi/15] \), with the initial set \( \theta_0 \in [-\pi/36, \pi/36] \). For the 4D Husky, the states are position \( x \) and \( y \), orientation \( \theta \), and linear velocity \( v \). The task of the controller is to keep the robot within a lane, and hence the safe set is \( y \in [-1, 1] \), and the initial set is defined as any position \( (x, y) \) that is within a radius of 0.1 from the
Table 1: Benchmark results for barrier certificate and control synthesis on various NNDMs. Threshold $\delta_s = 0.95$ was used for the minimally-invasive control synthesis. Architecture $h \times [r]$ denotes a NNDM with $h$ hidden layers, each with $r$ neurons. $n$ is the dimensionality of the system.

| Model       | $n \times [r]$ | $|Q|$ | Verification | Control |
|-------------|----------------|-----|--------------|---------|
|             | $\delta$ | $P_s$ | Time (min) | $\beta P_s$ | Time (min) | $\delta$ | $P_s$ | Time (min) | $\beta P_s$ | Time (min) |
| Pendulum    | 1 x [64]  | 0.531 | 0.06 | 0.005 | 0.09 | $10^{-6}$ | 0.999 | 0.04 | - | - |
|             | 2 x [64]  | 0.555 | 0.05 | 0.196 | 0.08 | $10^{-6}$ | 0.999 | 0.04 | $10^{-6}$ | 0.999 | 0.05 |
|             | 3 x [64]  | 0.674 | 0.05 | 0.388 | 0.06 | $10^{-6}$ | 0.994 | 0.06 | $10^{-6}$ | 0.985 | 0.05 |
|             | 5 x [64]  | 0.866 | 0.40 | 0.821 | 0.44 | $10^{-6}$ | 0.907 | 0.37 | $10^{-6}$ | 0.951 | 0.34 |
| Cartpole    | 1 x [128] 4 | 0.493 | 0.10 | 0.825 | 0.17 | $10^{-6}$ | 0.971 | 12.83 | $10^{-6}$ | 0.998 | 15.53 |
|             | 2 x [128] 4 | 0.796 | 0.19 | 0.385 | 0.22 | $10^{-6}$ | 0.979 | 158.51 | $10^{-6}$ | 0.985 | 164.41 |
|             | 1 x [256] 4 | 0.878 | 0.10 | 0.666 | 0.13 | $10^{-6}$ | 0.980 | 148.49 | $10^{-6}$ | 0.997 | 160.46 |
| Husky       | 1 x [256] 4 | 0.672 | 0.20 | 0.691 | 0.25 | $10^{-6}$ | 0.812 | 11.27 | $10^{-6}$ | 0.951 | 14.02 |
|             | 2 x [256] 4 | 0.667 | 0.21 | 0.673 | 0.28 | $10^{-6}$ | 0.878 | 33.27 | $10^{-6}$ | 0.991 | 35.10 |
|             | 5 x [512] 5 | 1.000 | 0.00 | 0.502 | 0.22 | $10^{-6}$ | 0.847 | 49.01 | $10^{-6}$ | 0.951 | 52.02 |
| Acrrobat    | 1 x [512] 6 | 1.000 | 0.00 | 0.502 | 0.22 | $10^{-6}$ | 0.847 | 49.01 | $10^{-6}$ | 0.951 | 52.02 |
|             | 2 x [512] 6 | 1.000 | 0.00 | 0.502 | 0.22 | $10^{-6}$ | 0.847 | 49.01 | $10^{-6}$ | 0.951 | 52.02 |
|             | 3 x [512] 6 | 1.000 | 0.00 | 0.502 | 0.22 | $10^{-6}$ | 0.847 | 49.01 | $10^{-6}$ | 0.951 | 52.02 |
|             | 5 x [512] 6 | 1.000 | 0.00 | 0.502 | 0.22 | $10^{-6}$ | 0.847 | 49.01 | $10^{-6}$ | 0.951 | 52.02 |

Implementation and Experimental Setup: We implemented our algorithms in Python and Julia (code available in [43]). For collecting data and training, we used the TensorFlow framework [44]. To compute the linear approximations of the NNDMs, we utilized α-CROWN [36]. For the optimization, we used Julia’s SumOfSquares.jl package [45][46]. The optimizations are all performed single-threaded on a computer with 3.9 GHz 8-Core CPU and 64 GB of memory.

Benchmarks: To the best of our knowledge, there is no other work on safety verification of NNDMs, against which we can compare. Hence as a baseline, we compare our approach based on linear under and over approximation of the NNDM against interval bounding approaches cf. Remark 5. The interval bounding approach results in a relaxed optimization problem at the cost of more conservative safety probabilities. The results are shown in Table 1. The computation times indicate the time to synthesize the barrier and the controller using Theorems 1 and 2 respectively. The barrier degree is 4 for all the case studies. Below, we provide a brief discussion on the results. For more in-depth discussions and details on the discretization and hyperparameters, see Appendix A.

Verification: As expected, the safety probability always increases with $|Q|$ (finer partitions), since the linear and interval bounds for $f^{i\infty}$ become more accurate. This behavior is encountered in all the systems and architectures, regardless of the number of hidden layers or neurons per layer. However, it is also observed that this trend of finer discretizations and increased safety probability consistently comes at the cost of an increased computation time. This is due to the fact that the number of partitions directly dictate the number of constraints in the optimization problems. Note that the probability of safety using interval bounds is strictly worse than linear approximations for all the experiments. Using either linear or interval bounds, we are able to compute non-trivial probability of safety for all the case-studies and observe the discretization vs accuracy trend. Take for instance, the 1-layer Cartpole model, where an increase in $|Q|$ from 960 to 9840 causes an increase in $P_s$ from

origin. For the 5D Husky, the additional state over the 4D model is the angular velocity $\omega$; the task, the safe set, and the initial set remain the same. Finally, for the Acrobot, the state space consists of the cosines and sines of $\theta_1$ and $\theta_2$, and angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$. Here $\theta_1$ and $\theta_2$ are the pole angle of the first link and the angle of the second link relative to the first link, respectively. The task is for the tip of the second link to reach a height of at least $y = 1$, with a safety constraint to never reach beyond $y = 1.2$. Thus, the safe set is defined as $\sin(\theta_1) \in [-0.6, 0.6]$ and $\sin(\theta_2) \in [-0.6, 0.6]$, and the initial set is any point within a radius of 0.1 around the origin in the first 4 dimensions.
0.375 to 0.599, at the expense of an increased computation time of more than 12 times. This behavior is also true for the 4D Husky model, where increasing $|Q|$ from 1800 to 4800 for the 2-layer model, increases $P_s$ from 0.010 to 0.384 at the price of a more than 5-fold increase in the computation time.

**Control Synthesis:** When the certification probability is below $\delta_s = 0.95$, the control strategy is utilized to increase the probability of safety. Using interval bounds, controllers had to be synthesized for all case-studies, which illustrates the looseness of the bounds. In fact, in several case studies, the controller was not able to meet the desired threshold. For the linear bounds, observe that for all except two NNDMs, whose $P_s < \delta_s$, the controller synthesis algorithm was able to produce $P_s \geq \delta_s$, showing the effectiveness of our control method. The first exception is for the 4D Husky model with 2-hidden layers and $|Q| = 1800$, where the controller increases $P_s$ from 0.010 to 0.800 (73-fold increase) yet cannot meet the 0.95 threshold. Similarly, for the 1-layer 5D Husky model with $|Q| = 432$, the controller increases $P_s$ from 0.00 to 0.949, but the 0.95 threshold cannot be reached. Nonetheless, for all the linear bound cases, the controller reduces $\beta_q$ to $10^{-6}$.

Figure [1] illustrates the minimally-invasive aspect of the controllers. It shows the $\beta_q$ values for the Pendulum NNDM with 2 layers before and after the application of the controllers with three different sizes of $|Q|$. For $|Q| = 120$, all the regions require controllers, but for $|Q| = 240$ and 480, respectively, only 69.2% and 23.75% of the regions require controllers; hence, the minimally-invasive controller focuses specifically on those regions and does not affect those with already-small $\beta_q$.

## 7 Conclusion

In this work, we introduced a methodology for providing safety guarantees for NNDMs using stochastic barrier functions. We showed that the problem of finding a stochastic barrier function for a NNDM can be relaxed to an SOS optimization problem. Furthermore, we derived a novel framework for synthesis of an affine controller with the goal of maximizing the safety probability in a minimally-invasive manner. Experiments showcase that, along with a minimally-invasive controller, we can guarantee safety for various standard reinforcement learning problems.

A bottleneck to our approach is the discretization step, which is required to obtain piece-wise linear under- and over-approximations of the neural network, and can be expensive for higher dimensional spaces. Future directions should focus on mitigating this curse of dimensionality.

## Acknowledgments and Disclosure of Funding

This work was supported in part by the NSF grant 2039062 and NASA COLDTech Program under grant #80NSSC21K1031.
References


Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? [Yes] See the paragraph above Related Work in Section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is motivated by providing safety guarantees for NNDMs in safety-critical applications, especially when deploying these NNDMs in real-world scenarios. This line of work is meant to contribute to society in a positive way. Potential negative societal impacts have hence not been a part of the discussion in this paper.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] All main theorems, lemmas and assumptions have been outlined in the paper. Theorems 1-2 as well as Lemmas 1-2 form the core of this paper. The discussion regarding all of these has been detailed to describe our approach as clear as possible to the reader.
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs for Lemmas 1-2 as well as Theorems 1-2 are included in the main body of the paper.

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code is included in the supplementary material as a docker file, including detailed instructions on how to reproduce the experimental results. This code can also be found on Github, see [43]. Note, a Mosek license is required to run the optimizations, which can be obtained from [https://www.mosek.com](https://www.mosek.com).
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 6 and Appendix A.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 6.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] State-of-the-art linear approximates of NNs have been used as existing assets. The main takeaways of these assets used in this research work have been captured in Section 4.1 (Auto-LiRPA Toolbox). Further, we use the model-based model-free RL approach mentioned in Section 6 to train a NNDM model in PyBullet (Physics Simulator). Naturally, the necessary citations have been provided to credit the creators. At last, the Mosek toolbox has been utilized for optimization. While not directly cited in the text, this toolbox has been given credit in our online available code.
   (b) Did you mention the license of the assets? [Yes] See the supplementary material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] A new framework has been provided related to the work of the paper. This code is included as supplementary material, and on Github (see [43]).
   (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] This discussion is not part of this paper. The
included material does not contain personally identifiable information or offensive content, to the best of our knowledge.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]