

# Stability Analysis for State Feedback Control Systems Established as Neural Networks with Input Constraints

Lukas Markolf<sup>a</sup> and Olaf Stursberg<sup>b</sup>

Control and System Theory, Dept. of Electrical Engineering and Computer Science, University of Kassel,  
Wilhelmshöher Allee 73, 34121 Kassel, Germany

**Keywords:** Constrained Control, Intelligent Control, Neural Networks, Reinforcement Learning, Stability.

**Abstract:** Considerable progress in deep learning has also led to an increasing interest in using deep neural networks (DNN) for state feedback in closed-loop control systems. In contrast to other purposes of DNN, it is insufficient to consider them only as black box models in control, in particular, when used for safety-critical applications. This paper provides an approach allowing to use the well-established indirect method of Lyapunov for time-invariant continuous time nonlinear systems with neural networks as state feedback controllers in the loop. A key element hereto is the derivation of a closed-form expression for the partial derivative of the neural network controller with respect to its input. By using activation functions of the type of sigmoid functions in the output layer, the consideration of box-constrained inputs is further ensured. The proposed approach does not only allow to verify the asymptotic stability, but also to find Lyapunov functions which can be used to search for positively invariant sets and estimates for the region of attraction.

## 1 INTRODUCTION

Synthesizing feedback controllers for solving regulation problems for general nonlinear dynamics  $\dot{x}(t) = f(x(t), u(t))$  is still challenging, in particular if input and state constraints have to be taken into consideration. In principle, dynamic programming (DP) (Bellman, 1957) formulates the solution to the named problem, but the computational complexity prevents its application for many real-world systems. In addition, the controller is obtained typically as look-up table rather than as functional representation, which may be undesirable for implementation and analysis.

With the recent renewed impetus on intelligent control, the “data-based learning” of such feedback controllers gets again into focus: The aim there is to establish the state feedback controller as a parametric structure, and to learn the parameters from data pairs of states and appropriate inputs. The state-input pairs may originate from DP or other off-line solutions of optimization problems for selected initial states, see e.g. (Markolf et al., 2020). For the case that complexity prevents the application of DP for data generation, “approximate dynamic programming” (ADP) (Lewis and Liu, 2013), also known as “adaptive dynamic


programming” (Liu et al., 2017) and “neuro-dynamic programming” (Bertsekas and Tsitsiklis, 1996), or “reinforcement learning” (RL) methods provide approximate DP solutions (Sutton and Barto, 2018). Feed-forward (deep) neural networks are widely used as parametric structure for nonlinear function approximation, motivated by universal approximation theorems (Cybenko, 1989), (Hornik et al., 1989) and recent success relying on “deep learning” (Goodfellow et al., 2016). However, neural networks are generally hard to analyze due to their nonlinear and large-scale nature, which explains why they are mostly used as black-box models without formal guarantees (Fazlyab et al., 2020). The applicability of neural networks for control, however, depends critically on the ability to provide guarantees, especially in safety-critical applications. In order to meet this requirement, it is insufficient to treat a neural network controller as a black-box.


### 1.1 Problem Statement

This paper considers time-invariant continuous-time nonlinear dynamic systems of the form:

$$\dot{x}(t) = f(x(t), u(t)), \quad (1)$$

with time  $t \in \mathbb{R}^{\geq 0}$ , state vector  $x(t) \in \mathbb{R}^n$ , and input vector  $u(t) \in \mathbb{R}^m$ .

<sup>a</sup>  <https://orcid.org/0000-0003-4910-8218>

<sup>b</sup>  <https://orcid.org/0000-0002-9600-457X>

**Assumption 1.** For the function vector  $f$ , let the following assumptions hold:

- $f(x, u)$  is continuous on the domain  $\mathbb{R}^n \times \mathbb{R}^m$ ;
- $[\partial f / \partial x](x, u)$  as well as  $[\partial f / \partial u](x, u)$  exist and are both continuous on  $\mathbb{R}^n \times \mathbb{R}^m$ ;
- and  $f(0, 0) = 0$ .

It is furthermore supposed that the states and inputs are constrained:

$$x(t) \in \mathcal{X} \subset \mathbb{R}^n, \quad u(t) \in \mathcal{U} \subset \mathbb{R}^m, \quad (2)$$

to account for, e.g., safety restrictions and actuation limits. For the constraints, let the following assumption hold:

**Assumption 2.** The sets  $\mathcal{X}$  and  $\mathcal{U}$  are compact (i.e. closed and bounded), nonempty and contain the origin in their interior. Further, it is assumed that the set  $\mathcal{U}$  is of the form  $\mathcal{U} = \{u \mid \underline{u} \in \mathbb{R}_{<0}^m, \bar{u} \in \mathbb{R}_{>0}^m : \underline{u} \leq u \leq \bar{u}\} \subset \mathbb{R}^m$ , i.e.  $\mathcal{U}$  encodes box constraints on the input vector.

For a given state feedback controller  $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , the closed-loop system is denoted by:

$$\dot{x}(t) = f(x, \phi(x)) =: f_{\text{cl}}(x). \quad (3)$$

The problem addressed in this paper is to design a state feedback controller  $\phi(x)$  to keep the system close to the origin, where  $\phi(x)$  is established as feed-forward neural network, named here *NN controller* for simplicity. The important aspect focused on in this paper is to ensure safety properties of the control loop, i.e. the design task includes:

- the step of determining the NN controller to map any admissible state into an admissible input:  $\phi_{\text{NN}} : \mathcal{X} \rightarrow \mathcal{U}$ ;
- the step of analyzing stability a-posteriori, i.e. to check whether any state  $x(t)$  reached along the solution of the initial-value problem:

$$\dot{x}(t) = f_{\text{cl}}(x(t)), \quad x(0) = x_0 \quad (4)$$

satisfies  $x(t) \in \mathcal{X}$  for any  $t \geq 0$ .

The second step includes to verify the existence and uniqueness of the solution of the initial-value problem. The generation of training data or the training procedure itself are, however, not in the focus of this work.

## 1.2 Overview of the Approach

The main contribution of this work is to show how Lyapunov's indirect method can be tailored to the NN controller in order to show that (3) is asymptotically stable with respect to  $x = 0$ . Whenever the method

succeeds, the true region of attraction of the origin can be estimated as detailed in the next section. This is not only beneficial for the verification of safety, but also in view of the objective to keep the state close to the origin.

To enable the application of Lyapunov's indirect method for closed-loop systems with NN controller, two fundamental steps are 1.) the derivation of the partial derivative of the NN controller with respect to its input in closed-form, and 2.) the manipulation of the bias vector in the output layer of the NN controller for ensuring that the origin is indeed an equilibrium point of the closed-loop system.

Continuous and continuously differentiable activation functions are considered in the units of the NN controller in order to allow the computation of the partial derivative in each state. Sigmoid functions with a characteristic "S"-shaped form are often used as activation functions to meet these properties. Their use in the units of the output layer makes it moreover straightforward to satisfy input constraints. The NN controller proposed in this work satisfies the input constraints even if the same type of sigmoid function is used in each unit of the output layer, making the implementation easier. On the basis of the proposed NN controller and its partial derivative with respect to its input, it will be shown that the closed-loop system is locally Lipschitz on the set of admissible states, which is beneficial for the verification of the existence and uniqueness of the solution.

For the case that the origin is asymptotically stable, Lyapunov's indirect method provides information for the construction of a Lyapunov function. In the numerical examples, interval arithmetics will be used to address the problem of determining the states for which the derivative of the Lyapunov function along the trajectories of the controlled system is negative. While this procedure may fail in a small neighborhood of the origin, the task of keeping the state close to the origin can still be solved, since a theorem similar to LaSalle's theorem can be used to ensure that a (desirably small) positively invariant set containing the origin is reached in finite time.

## 1.3 Related Work

First of all, the use of the properties of activation functions to ensure the satisfaction of input constraints is no novelty per se. In (Fazlyab et al., 2020), for example, rectified linear units are used for this purpose, which however may not be continuously differentiable. To the knowledge of the authors, work describing the constraint handling of NN controllers for general sigmoid functions in the output layer does not

exist so far, such that this step will be described in detail for the sake of completeness.

Pushed by the discovery that neural networks are vulnerable to adversarial attacks (Kurakin et al., 2016), the analysis of the output range of neural networks has been addressed recently, see e.g. (Dutta et al., 2018), (Fazlyab et al., 2019). Techniques originating from the analysis of the output range have found their way into the verification of closed-loop systems with NN controllers and are used for reachability analysis to verify safety by the over-approximation of reachable sets (Dutta et al., 2019), (Huang et al., 2019), (Ivanov et al., 2019), (Hu et al., 2020).

Lyapunov's stability theory is another established tool for safety verification. In (Perkins and Barto, 2002), for example, safety and reliability of reinforcement learning agents have been considered by switching among controllers constructed on the basis of Lyapunov design principles. The approach of learning Lyapunov functions has been investigated in several works, where the use of neural networks as Lyapunov functions can be found e.g. in (Petridis and Petridis, 2006), (Richards et al., 2018), (Chang et al., 2020), (Grüne, 2020). In (Chang et al., 2020), it is proposed to learn control functions and neural Lyapunov functions together. The focus there is on neural Lyapunov functions, and general parametric control functions are considered. In contrast, the present paper considers the explicit synthesis of NN controllers including safety analysis and the consideration of input constraints.

The paper is organized such that Sec. 2 recalls well-established results and concepts that are significant for the proposed approach. Section 3 introduces the general architecture of feed-forward neural networks and gives an overview about their training. Moreover, a closed-form expression for the partial derivative of such networks with respect to their inputs is provided. The main results are provided in Sec. 4, followed by application examples in Sec. 5 as well as conclusions in Sec. 6.

## 2 PRELIMINARIES

This section briefly recalls well-established results and concepts that are significant for this work, see e.g. (Khalil, 2002), (Khalil, 2015) for more details. The following facts will be used for verifying the existence and uniqueness of the solution for initial value problems.

**Lemma 1** ((Khalil, 2002)). *Let  $f_{cl}(x)$  and  $[\partial f_{cl}/\partial x](x)$  be continuous on an open and con-*

*nected domain  $D \subset \mathbb{R}^n$ , then  $f_{cl}$  is locally Lipschitz for any  $x \in D$ .*

**Lemma 2** ((Khalil, 2002)). *Let  $f_{cl}(x)$  be locally Lipschitz in  $x \in D \subset \mathbb{R}^n$ . Furthermore, let  $\mathcal{W}$  be a compact subset of  $D$ ,  $x_0 \in \mathcal{W}$ , and suppose it is known that every solution of the initial-value problem (4) lies entirely in  $\mathcal{W}$ . Then, a unique solution exists for the problem and is defined for all  $t \geq 0$ .*

Suppose that  $f_{cl}(x)$  in (3) is locally Lipschitz in  $x \in \mathcal{D} \subset \mathbb{R}^n$ , and that  $f_{cl}(0) = 0$ . Then,  $x = 0$  is an equilibrium point of the closed-loop system, which is called *stable* if for each  $\varepsilon > 0$  there is a  $\delta$  (dependent on  $\varepsilon$ ) such that

$$\|x(0)\| < \delta \Rightarrow \|x(t)\| < \varepsilon, \quad \text{for all } t \geq 0. \quad (5)$$

If the origin is stable and  $\delta$  can be chosen such that

$$\|x(0)\| < \delta \Rightarrow \lim_{t \rightarrow \infty} \|x(t)\| = 0, \quad (6)$$

then the origin is called *asymptotically stable*. Lyapunov's indirect method is used to verify asymptotic stability of the origin:

**Lemma 3** ((Khalil, 2002)). *Let  $x = 0$  be an equilibrium point of (3) and suppose that  $f_{cl}(x)$  is continuously differentiable in  $x = 0$ . Let  $\lambda_1$  to  $\lambda_n$  denote the eigenvalues of:*

$$A_{cl} := \left. \frac{\partial f_{cl}(x)}{\partial x} \right|_{x=0} = f_x(x) + f_u(x) \left. \frac{\partial \phi(x)}{\partial x} \right|_{x=0}, \quad (7)$$

where:

$$f_x(x) := \left. \frac{\partial f(x,u)}{\partial x} \right|_{u=\phi(x)}, \quad f_u(x) := \left. \frac{\partial f(x,u)}{\partial u} \right|_{u=\phi(x)}. \quad (8)$$

Then,

1. *the closed-loop system is exponentially stable (and by that also asymptotically stable) with respect to the origin if and only if  $\text{Re}[\lambda_i] < 0$  for all eigenvalues,  $i \in \{1, \dots, n\}$ ;*
2. *the closed-loop system is unstable in the origin, if  $\text{Re}[\lambda_i] > 0$  for one or more of the eigenvalues.*

A continuously differentiable function defined over a domain  $\mathcal{N} \subset \mathbb{R}^n$  is called *Lyapunov function* if it satisfies:

$$V(0) = 0 \text{ and } V(x) > 0 \text{ for all } x \in \mathcal{N} \text{ with } x \neq 0, \quad (9)$$

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} f_{cl}(x) \leq 0 \text{ for all } x \in \mathcal{N}. \quad (10)$$

For the case that  $A_{cl}$  is Hurwitz ( $\text{Re}[\lambda_i] < 0$  for all  $i$ ), a quadratic Lyapunov function can be found by solving

the Lyapunov equation for a positive definite matrix  $Q$ :

$$0 = PA_{cl} + A_{cl}^T P + Q, \quad (11)$$

$$V(x) = x^T P x. \quad (12)$$

If  $\Omega_c = \{x \in \mathbb{R}^n \mid V(x) \leq c\}$  is a subset of  $\mathcal{N}$  for a positive constant  $c$  and if  $\dot{V}(x) < 0$  for all  $x \in \Omega_c$ , then  $\Omega_c$  is *positively invariant* (i.e.,  $x(0) \in \Omega_c \Rightarrow x(t) \in \Omega_c$  for all  $t \geq 0$ ) and determines an inner approximation of the true *region of attraction* (i.e., the set of all points  $x(0) \in D$  for which the solution  $x(t)$  exists for all  $t \geq 0$  and converges to the origin as  $t \rightarrow \infty$ ). In fact, each positively invariant set  $I \subset \mathcal{N}$ , for which  $\dot{V}(x) < 0$  for all  $x \in I$ , is an inner approximation estimate of the true region of attraction. Let:

$$I = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, i \in \{0, 1, \dots, r\}\} \quad (13)$$

be a convex set, then positive invariance can be verified by:

$$f_{cl}(x) \in \mathcal{T}_I(x), \quad \text{for all } x \in \partial I, \quad (14)$$

where  $\partial I$  is the boundary of  $I$ , and  $\mathcal{T}_I(x)$  the tangent cone given by:

$$\mathcal{T}_I(x) = \{z \in \mathbb{R}^n \mid \nabla g_i(x)^T z \leq 0, \text{ for all } i \in \text{Act}(x)\}, \quad (15)$$

with

$$\text{Act}(x) = \{i \mid g_i(x) = 0\}. \quad (16)$$

(See (Blanchini and Miani, 2008) for more details.)

## 3 NEURAL NETWORKS

### 3.1 Architecture

While feed-forward neural networks are used in various types, this work focuses on networks with overall mapping defined by a chain structure of the form (Goodfellow et al., 2016):

$$h(x) := \left( h^{(L)} \circ \dots \circ h^{(2)} \circ h^{(1)} \right)(x), \quad (17)$$

with layers  $h^{(\ell)}$ ,  $\ell \in \{1, \dots, L\}$ . The final layer  $h^{(L)}$  is usually denoted as output layer, while the others are referred to as hidden layers. Let  $\eta^{(\ell)}$  denote the output of layer  $\ell$ , and  $\eta^{(0)}$  the input of the overall network:

$$\eta^{(0)}(x) = x, \quad (18)$$

$$\eta^{(\ell)}(x) = \left( h^{(\ell)} \circ \dots \circ h^{(1)} \right)(x). \quad (19)$$

The layers are functions of the form

$$h^{(\ell)} \left( \eta^{(\ell-1)} \right) := \left( g^{(\ell)} \circ \mu^{(\ell)} \right) \left( \eta^{(\ell-1)} \right), \quad (20)$$

where the  $\mu^{(\ell)}$  and  $g^{(\ell)}$  constitute affine and nonlinear transformations, respectively. Note that the hidden layers are typically vector-to-vector functions. The affine transformation  $\mu^{(\ell)}$  is defined to:

$$\mu^{(\ell)} \left( \eta^{(\ell-1)} \right) := W^{(\ell)} \eta^{(\ell-1)} + b^{(\ell)}, \quad (21)$$

and is affected by the choice of the weight matrix  $W^{(\ell)}$  and the bias vector  $b^{(\ell)}$ . Each layer can be understood to consist of parallel acting units, where each unit defines a vector-to-scalar function: Let  $S^{(\ell)}$  be an integer describing the number of units in layer  $\ell$ . The vector-to-scalar function of unit  $i$  in layer  $\ell$  is then the  $i$ -th component of  $h^{(\ell)}$ :

$$h_i^{(\ell)} \left( \eta^{(\ell-1)} \right) = g_i^{(\ell)} \left( \mu_i^{(\ell)} \right), \quad (22)$$

where:

$$\mu_i^{(\ell)} = \left( \sum_{j=1}^{S^{(\ell-1)}} W_{i,j}^{(\ell)} \eta_j^{(\ell-1)} \right) + b_i^{(\ell)}. \quad (23)$$

The function  $g_i^{(\ell)}$  is typically denoted as activation function. Rectified linear units or sigmoid functions are often chosen as activation functions in the units of the hidden layers. On the other hand, it is common to use the identity function as activation function in the output layer.

This work focuses on activation functions which are continuous and continuously differentiable everywhere, such as activation functions of the form:

$$g_i^{(\ell)} \left( \mu_i^{(\ell)} \right) = \sigma \left( \mu_i^{(\ell)}, \alpha_i^{(\ell)}, \beta_i^{(\ell)} \right), \quad (24)$$

with  $\alpha_i^{(\ell)} \in (0, \infty)$ ,  $\beta_i^{(\ell)} \in (-\infty, \infty)$ , and:

$$\sigma(\mu, \alpha, \beta) = \frac{\alpha}{1 + e^{-\alpha\mu}} - \beta. \quad (25)$$

Such functions belong to the family of sigmoid functions with a characteristic ‘‘S’’-shaped form bounded by:

$$\inf_{\mu \in \mathbb{R}} \sigma(\mu, \alpha, \beta) = \lim_{\mu \rightarrow -\infty} \sigma(\mu, \alpha, \beta) = -\beta, \quad (26)$$

$$\sup_{\mu \in \mathbb{R}} \sigma(\mu, \alpha, \beta) = \lim_{\mu \rightarrow \infty} \sigma(\mu, \alpha, \beta) = -\beta + \alpha. \quad (27)$$

Common choices are the logistic function  $\Gamma(\mu) = \sigma(\mu, 1, 0)$ , or the hyperbolic tangent function  $\tanh(\mu) = \sigma(\mu, 2, 1)$ . The use of sigmoid functions in the output layer is motivated by the fact that input constraints as considered here can be encoded straightforwardly if the feedback controller is established as neural network of the form (17) with  $g_i^{(L)}(\mu) = \sigma(\mu, \bar{u}_i - \underline{u}_i, -\underline{u}_i)$ . In Thm. 1 (Sec. 4), however, a more general approach will be established, allowing to use the same sigmoid function as activation function in each unit.

### 3.2 Derivative

The use of continuously differentiable activation functions enables one to compute  $[\partial g^{(\ell)} / \partial \mu^{(\ell)}](\mu^{(\ell)})$ . According to (22),  $[\partial g^{(\ell)} / \partial \mu^{(\ell)}](\mu^{(\ell)})$  is a diagonal matrix:

$$\frac{\partial g^{(\ell)}(\mu^{(\ell)})}{\partial \mu^{(\ell)}} = \text{diag} \left[ \frac{\partial g_i^{(\ell)}(\mu_i^{(\ell)})}{\partial \mu_i^{(\ell)}} \right]_{i=1}^{s^{(\ell)}}, \quad (28)$$

since  $[\partial g_i^{(\ell)} / \partial \mu_j^{(\ell)}](\mu_i^{(\ell)}) = 0$  for  $i \neq j$ . For sigmoid functions, it follows from (25) that:

$$\frac{\partial \sigma(\mu, \alpha, \beta)}{\partial \mu} = (\sigma(\mu, \alpha, \beta) + \beta) (\alpha - (\sigma(\mu, \alpha, \beta) + \beta)). \quad (29)$$

If  $g_i^{(\ell)}(\mu_i^{(\ell)}) = \tanh(\mu_i^{(\ell)}) = \sigma(\mu_i^{(\ell)}, 2, 1)$ , for instance, then  $[\partial g_i^{(\ell)} / \partial \mu_i^{(\ell)}](\mu_i^{(\ell)}) = 1 - \tanh^2(\mu_i^{(\ell)})$ . Of course, if the identity function is used as activation function, i.e.  $g_i^{(\ell)}(\mu_i^{(\ell)}) = \mu_i^{(\ell)}$ , then  $[\partial g_i^{(\ell)} / \partial \mu_i^{(\ell)}](\mu_i^{(\ell)}) = 1$  holds.

By use of (21), one obtains:

$$\frac{\partial \mu^{(\ell)}(\eta^{(\ell-1)})}{\partial \eta^{(\ell-1)}} = W^{(\ell)}, \quad (30)$$

such that the partial derivative of the mapping  $h^{(\ell)}$  of layer  $\ell$  with respect to its input vector  $\eta^{(\ell-1)}$  follows in closed-form by applying the chain rule:

$$\frac{\partial h^{(\ell)}(\eta^{(\ell-1)})}{\partial \eta^{(\ell-1)}} = \frac{\partial h^{(\ell)}(\mu^{(\ell)}(\eta^{(\ell-1)}))}{\partial \mu^{(\ell)}} \frac{\partial \mu^{(\ell)}(\eta^{(\ell-1)})}{\partial \eta^{(\ell-1)}}. \quad (31)$$

Finally, the chain rule can be used again to derive a closed-form expression of the overall mapping  $h$  of the neural network with respect to its input vector:

$$\frac{\partial h(x)}{\partial x} = \prod_{i=0}^{L-1} \frac{\partial h^{(L-i)}(\eta^{(L-(i+1))}(x))}{\partial \eta^{(L-(i+1))}}. \quad (32)$$

### 3.3 Training

The neural network (17) is a parametric architecture  $h(x; r)$  with parameter vector  $r$ , which contains the components of the weight matrices and the bias vectors:

$$r = \left[ W_{1,1}^{(1)} \quad \dots \quad W_{S^{(L)}, S^{(L-1)}}^{(L)} \quad b_1^{(1)} \quad \dots \quad b_{S^{(L)}}^{(L)} \right]^T. \quad (33)$$

Hence, the shape of the overall mapping  $h$  of the neural network can be affected by the choice of the parameter vector  $r$ .

In this work, the neural networks are used for regression tasks, i.e. the function  $h$  is used to predict  $y$  given some input  $x$ . Suppose that a data set  $(x^s, y^s)$ ,  $s \in \{1, \dots, q\}$  is available, where each  $y^s$  is a regression target providing an approximated value  $y$  for the corresponding example input  $x^s$ . The training procedure aims at adapting the parameter vector in order to improve the approximation performance by learning from the data set. Here, the mean squared error is considered as performance measure. The challenge is to perform well also for new, previously unseen inputs  $x$ . Hence, the training procedure is not only a pure optimization task searching for a parameter vector by minimizing the mean squared error for the known data set. It involves also the determination of a function which interpolates (or even extrapolates) to new data. A detailed discussion is out of the scope of this paper, but can be found in (Goodfellow et al., 2016).

## 4 MAIN RESULTS

### 4.1 Neural Network Controller

The following definition introduces the type of controller considered in this work and establishes it as neural network.

**Definition 1.** Given vectors  $\underline{u}, \bar{u}$  of lower and upper bounds of the  $m$ -dimensional input vector  $u$  according to Asm. 2, the NN controller is a state feedback controller:

$$\Phi_{NN}(x; r) = \text{diag} \left[ \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \right]_{i=1}^m \left( h(x; r) + \beta^{(L)} \right) + \underline{u}, \quad (34)$$

with a feed-forward neural network  $h$  as defined by (17) with  $L$  layers and the parameter vector  $r$  as in (33) obtained from training. The activation functions in the layers of the neural network are assumed to be continuous and continuously differentiable. The activation functions in the output layer  $L$  are chosen as sigmoid functions according to (25) with  $\alpha_i^{(\ell)} \in (0, \infty)$  and  $\beta_i^{(\ell)} \in (-\infty, \infty)$ .

Training of the NN controller with a data set  $(x^s, u^s)$ ,  $s \in \{1, \dots, q\}$  is here understood as adapting the parameters of  $h$  in (17) with the transformed data set  $(x^s, \eta^s)$ ,  $s \in \{1, \dots, q\}$ , where:

$$\eta^s(u^s) = \text{diag} \left[ \frac{\alpha_i^{(L)}}{\bar{u}_i - \underline{u}_i} \right]_{i=1}^m (u^s - \underline{u}) - \beta^{(L)}. \quad (35)$$

The following theorem shows that the NN controller meets the input constraints defined in Def. 1.

**Theorem 1.** *The NN controller  $\phi_{NN}(x)$  according to Def. 1 maps each state  $x \in \mathcal{X}$  into the input set  $\mathcal{U}$ , independently of the choice of the parameter vector  $r$  of the neural network  $h$ .*

*Proof.* The  $i$ -th output of the overall neural network  $h$  is equal to the  $i$ -th output of the final layer  $h^{(L)}$ , which in turn is the output of the activation function  $g_i^{(L)}$  in unit  $i$  of the output layer  $L$ , see (17) and (22). Since this activation function is required to be of the form (25), the properties (26) - (27) hold. Hence:

$$\inf_{x \in \mathbb{R}^n} h_i(x; r) \geq \inf_{\mu \in \mathbb{R}} g_i^{(L)}(\mu) = -\beta_i^{(L)}, \quad (36)$$

$$\sup_{x \in \mathbb{R}^n} h_i(x; r) \leq \sup_{\mu \in \mathbb{R}} g_i^{(L)}(\mu) = -\beta_i^{(L)} + \alpha_i^{(L)}. \quad (37)$$

Due to the linear dependency of :

$$\phi_{NN,i}(x) = \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} h_i(x; r) + \frac{\beta_i^{(L)}}{\alpha_i^{(L)}} (\bar{u}_i - \underline{u}_i) + \underline{u}_i \quad (38)$$

on  $h_i(x; r)$ , and since the term  $(\bar{u}_i - \underline{u}_i)/\alpha_i^{(L)}$  is greater than zero according to the definitions of its parameters, it follows that:

$$\begin{aligned} \inf_{x \in \mathbb{R}^n} \phi_{NN,i}(x) &= \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \inf_{x \in \mathbb{R}^n} h_i(x; r) + \frac{\beta_i^{(L)}}{\alpha_i^{(L)}} (\bar{u}_i - \underline{u}_i) + \underline{u}_i \\ &\geq \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \inf_{\mu \in \mathbb{R}} g_i^{(L)}(\mu) + \frac{\beta_i^{(L)}}{\alpha_i^{(L)}} (\bar{u}_i - \underline{u}_i) + \underline{u}_i \\ &= \underline{u}_i, \end{aligned} \quad (39)$$

and:

$$\begin{aligned} \sup_{x \in \mathbb{R}^n} \phi_{NN,i}(x) &= \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \sup_{x \in \mathbb{R}^n} h_i(x; r) + \frac{\beta_i^{(L)}}{\alpha_i^{(L)}} (\bar{u}_i - \underline{u}_i) + \underline{u}_i \\ &\leq \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \sup_{\mu \in \mathbb{R}} g_i^{(L)}(\mu) + \frac{\beta_i^{(L)}}{\alpha_i^{(L)}} (\bar{u}_i - \underline{u}_i) + \underline{u}_i \\ &= \bar{u}_i. \end{aligned} \quad (40)$$

Obviously, there is no  $x \in \mathcal{X} \subset \mathbb{R}^n$  for which  $\phi_{NN}(x) > \bar{u}_i$  or  $\phi_{NN}(x) < \underline{u}_i$ . This result is obviously independent of the choice of the parameter vector  $r$  of the neural network.  $\square$

On the basis of (32), the properties of the NN controller allow to derive a closed-form expression for the partial derivative of the NN controller  $\phi_{NN}$  with respect to its input  $x$ :

$$\frac{\partial \phi_{NN}(x)}{\partial x} = \text{diag} \left[ \frac{\bar{u}_i - \underline{u}_i}{\alpha_i^{(L)}} \right]_{i=1}^m \frac{\partial h(x)}{\partial x}. \quad (41)$$

## 4.2 Existence and Uniqueness

In Sec. 2, it has been shown that the Lipschitz continuity of the closed-loop system is beneficial not only for verifying the existence and uniqueness of the solution of the initial-value problem, but also for the verification of stability aspects.

**Lemma 4.** *For the NN controller  $u = \phi_{NN}(x)$  of type (34) with the properties defined in Def. 1, let the Asm. 1 and Asm. 2 hold for the closed-loop system  $\dot{x}(t) = f(x, \phi_{NN}(x)) =: f_{cl,NN}(x)$  formulated for (1). Then, the closed-loop dynamics  $f_{cl,NN}$  is locally Lipschitz in any  $x \in \mathcal{X}$ .*

*Proof.* According to Lemma 1, the function  $f_{cl,NN}(x)$  is locally Lipschitz in any  $x \in \mathcal{X}$ , if  $f_{cl,NN}(x)$  and the partial derivatives  $[\partial f_{cl,NN,i}/\partial x_j](x)$  for any pair  $i, j \in \{1, \dots, n\}$  are continuous on  $\mathcal{X}$ .

(a): According to Asm. 1,  $f(x, u)$  is continuous on the domain  $\mathcal{X} \times \mathcal{U}$ . Hence,  $f_{cl,NN}(x) = f(x, \phi_{NN}(x))$  is continuous on  $\mathcal{X}$ , if  $\phi_{NN}(x)$  is continuous on  $\mathcal{X}$ , where the latter follows from the choice of continuous activation functions according to Def. 1.

(b): The partial derivative of  $f_{cl,NN}$  with respect to  $x$  is:

$$\frac{\partial f_{cl,NN}(x)}{\partial x} = f_x(x) + f_u(x) \frac{\partial \phi_{NN}(x)}{\partial x}, \quad (42)$$

with  $[\partial \phi_{NN}/\partial x](x)$  denoting the partial derivative of the NN controller with respect to the states as defined in (41), while  $f_x(x)$  and  $f_u(x)$  are the functions defined in (8). According to Asm. 2,  $[\partial f/\partial x](x, u)$  as well as  $[\partial f/\partial u](x, u)$  exist and are both continuous on  $\mathcal{X} \times \mathcal{U}$ . Hence, the derivatives  $[\partial f_{cl,NN,i}/\partial x_j](x)$  are continuous on  $\mathcal{X}$  if  $\phi_{NN}(x)$  and  $[\partial \phi_{NN}/\partial x](x)$  are continuous on  $\mathcal{X}$ . It remains to check that  $[\partial \phi_{NN}/\partial x](x)$  is continuous – this holds, since continuously differentiable activation functions are to be chosen according to Def. 1.  $\square$

The following theorem states the existence and uniqueness of a solution for the initial-value problem formulated for the closed-loop system:

**Theorem 2.** *Consider the NN controller  $\phi_{NN}$  of type (1) with the properties specified in Def. 1, and let Asm. 1 and Asm. 2 hold for the closed-loop system  $\dot{x}(t) = f_{cl,NN}(x) := f(x, \phi_{NN}(x))$  formulated for (1). Let  $I \subseteq \mathcal{X}$  be a compact and positively invariant set. Then, a unique solution of  $\dot{x}(t) = f_{cl,NN}(x)$  exists for any  $x(0) \in I$ , and this solution stays in  $I$  for all  $t \geq 0$ .*

*Proof.* Due to the assumption that  $I$  is positively invariant, each solution  $x(t)$  of the closed-loop system with  $x(0) \in I$  lies entirely in  $I$  for all  $t \geq 0$ . Since the closed-loop system is locally Lipschitz in any  $x \in \mathcal{X}$

(Lemma 4) and because  $I \subseteq \mathcal{X}$  is a compact subset of  $\mathcal{X}$ , Lemma 2 implies that the solution is unique.  $\square$

### 4.3 Stability Analysis

The following fact provides insight into how to select  $b^{(L)}$  in order to obtain the origin as an equilibrium point of the closed-loop system.

**Lemma 5.** *Given the system (1), let Asm. 3 hold, and let  $\phi_{NN}(x)$  again be an NN controller (34) with the properties as in Def. 1, thus forming the closed-loop system  $\dot{x}(t) = f_{cl,NN}(x)$ . If each element of the bias vector  $b^{(L)}$  satisfies:*

$$b_i^{(L)} = - \left( \left( \sum_{j=1}^{s^{(L-1)}} w_{i,j}^{(L)} \eta_j^{(L-1)}(0) \right) + \frac{1}{\alpha_i^{(L)}} \ln \left( -\frac{\bar{u}_i}{u_i} \right) \right), \quad (43)$$

then  $x = 0$  is an equilibrium point of the closed-loop system, i.e.  $f_{cl,NN}(0) = 0$ .

*Proof.* According to Asm. 3,  $f(0,0) = 0$  and hence  $x = 0$  is an equilibrium point of the closed-loop system if  $\phi_{NN,i}(0) = 0$  for any  $i \in \{1, \dots, m\}$ . Considering the NN controller (34), the requirement  $\phi_{NN,i}(0) \stackrel{!}{=} 0$  is obviously equivalent to:

$$h_i(0) \stackrel{!}{=} \frac{\alpha_i^{(L)} u_i}{u_i - \bar{u}_i} - \beta_i^{(L)}. \quad (44)$$

It is shown next that this requirement is fulfilled, if any element of  $b^{(L)}$  satisfies (43): recall that the  $i$ -th output of the neural network  $h$  is equal to the  $i$ -th output of the final layer  $h^{(L)}$ , being equal to the output of the activation function  $g_i^{(L)}$  in unit  $i$  of the output layer  $L$ , see (17) and (22). Due to Def. 1,  $g_i^{(L)}$  is a sigmoid function (25), implying that:

$$h_i(0) = \sigma \left( \mu_i^{(L)} \left( \eta^{(L-1)}(0) \right), \alpha_i^{(L)}, \beta_i^{(L)} \right). \quad (45)$$

If an element of the bias vector satisfies (43), then it follows from (23) that:

$$\mu_i^{(L)} \left( \eta^{(L-1)}(0) \right) = -\frac{1}{\alpha_i^{(L)}} \ln \left( -\frac{\bar{u}_i}{u_i} \right). \quad (46)$$

Recall that  $u_i < 0$  and  $\bar{u}_i > 0$  per definition, as well as  $\alpha_i^{(L)} \in (0, \infty)$ . By inserting (46) into (45), it follows that (44) is fulfilled.  $\square$

So far, it has been shown that  $f_{cl,NN}(x)$  is locally Lipschitz on  $\mathcal{X}$ , that a closed-form expression for  $[\partial\phi_{NN}/\partial x](x)$  exists, and that the bias vector of the output layer can be modified to ensure

that  $f_{cl,NN}(0) = 0$ , such that the problem of verifying asymptotic stability of the origin can be addressed with Lyapunov's indirect method.

**Theorem 3.** *Given that the system (1) satisfies the Asm.s 1-3, let again  $\phi_{NN}(x)$  be an NN controller (34) as in Def. 1, forming the closed-loop system  $\dot{x}(t) = f_{cl,NN}(x)$ . If then any component of the bias vector satisfies (43) and if:*

$$A_{cl,NN} := \frac{\partial f_{cl,NN}(x)}{\partial x} \Big|_{x=0} = \frac{\partial f(x,u)}{\partial x} \Big|_{x=0,u=0} + \frac{\partial f(x,u)}{\partial u} \frac{\partial \phi_{NN}(x)}{\partial x} \Big|_{x=0,u=0} \quad (47)$$

has only eigenvalues with negative real parts, then (3) is stabilized exponentially on a neighborhood  $\mathcal{N}$  of the origin  $0 \in \mathcal{N}$ .

*Proof.* Since the controller satisfies (43), the origin is an equilibrium point of the closed-loop system, see Lemma 5. It follows from the proof of Lemma 4 that the closed-loop system is continuously differentiable in a neighborhood of the origin. Consequently, Lemma 3 can be applied to the closed-loop system. Then, for the case that  $A_{cl}$  has only eigenvalues with negative real parts, Lemma 3 implies exponential stability of (3) with respect to the origin.  $\square$

As mentioned in Sec. 1, interval arithmetics can be used to determine the regions in  $\mathcal{X}$  for which the derivative of the Lyapunov function  $V(x) = x^T P x$  along the trajectories of the closed-loop system is negative. In here,  $P$  is obtained by solving the Lyapunov equation  $0 = PA_{cl,NN} + A_{cl,NN}^T P + Q$  for a positive definite matrix  $Q$ . However, it was found in numeric studies that the analysis by interval arithmetics may fail in a very small neighborhood of the origin for numeric reasons. The next theorem shows that it is still possible to verify that a (desirably small) invariant set containing the origin is reached in finite time. This theorem and its proof bears similarities to LaSalle's invariance theorem, see e.g. (Khalil, 2002).

**Theorem 4.** *Let  $\dot{x}(t) = f_{cl,NN}(x)$  be the closed-loop dynamics of system (1) with the NN controller (34) following Def. 1. Suppose that  $I_1 \subset \mathcal{X}$  and  $I_2 \subseteq \mathcal{X}$  are compact, nonempty, and positively invariant sets satisfying  $I_1 \subset I_2$ . Let  $\Lambda$  be the relative complement of the interior  $I_1^\circ$  of  $I_1$  to  $I_2$ , i.e.,  $\Lambda := I_2 \setminus I_1^\circ$ . Furthermore, let  $V(x)$  be a continuous and continuously differentiable function defined over  $I_2$  for which the derivative  $\dot{V}(x)$  along the trajectories of  $\dot{x} = f_{cl,NN}(x)$  satisfies:*

$$\dot{V}(x) = \frac{\partial V(x)}{\partial x} f_{cl,NN}(x) < 0 \text{ for all } x \in \Lambda. \quad (48)$$

Then every solution starting in  $I_2$  enters  $I_1$  in finite time and stays therein for all future times.

*Proof.* Let  $x(t)$  be the solution of (3) for an initial state  $x(0) \in I_2$ . First, suppose that  $x(0) \in I_1$ . Since  $I_1$  is positively invariant, the solution  $x(t)$  must stay within  $I_1$  for all  $t \geq 0$ . Now suppose that  $x(0)$  is not an element of  $I_1$ , which implies that  $x(0) \in \Lambda$ . Next, it is shown by contradiction that  $x(t)$  must leave  $\Lambda$  in finite time: assume that  $x(t)$  stays within  $\Lambda$  for an infinite time. Since  $\dot{V}(x) < 0$  for all  $x \in \Lambda$ ,  $V(x(t))$  is a monotonically decreasing function over  $t$  as long as  $x(t) \in \Lambda$ , such that  $V(x(t))$  is unbounded from below. Note that  $\Lambda$  is also compact. Since  $V(x)$  is continuous on the compact set  $\Lambda$ , it must be bounded from below, which is a contradiction. Hence  $x(t)$  cannot stay within  $\Lambda$  for an infinite time and there exists a finite time at which  $x(t)$  leaves this set. Since  $I_2$  is positively invariant,  $x(t)$  can only enter the positively invariant set  $I_1$  when it leaves  $\Lambda$ , where it must stay for all future times.  $\square$

## 5 EXAMPLES

This section demonstrates the proposed approach for three benchmark problems. Benchmark 1 and Benchmark 2 originate from (Huang et al., 2019), where reachability analysis based on over-approximation has been used to verify if a goal set is reached starting from a specified initial set. The results there have also been compared with those obtained from the approaches proposed in (Dutta et al., 2019) and (Ivanov et al., 2019). On the other hand, Benchmark 3 is taken from (Liu et al., 2017), containing simulation studies for the demonstration of a proposed ADP technique. The right-hand sides of the three nonlinear dynamics and the associated input constraints to be considered for regulation problems are as follows:

**Benchmark 1:**  $\mathcal{U} = \{u \in \mathbb{R} \mid -1.5 \leq u \leq 1.5\}$ ,

$$f(x, u) = \begin{bmatrix} x_2 - x_1^3 \\ u \end{bmatrix}.$$

**Benchmark 2:**  $\mathcal{U} = \{u \in \mathbb{R} \mid -2 \leq u \leq 2\}$ ,

$$f(x, u) = \begin{bmatrix} -x_1 \left(0.1 + (x_1 + x_2)^2\right) \\ (u + x_1) \left(0.1 + (x_1 + x_2)^2\right) \end{bmatrix}.$$

**Benchmark 3:**  $\mathcal{U} = \{u \in \mathbb{R} \mid -2 \leq u \leq 2\}$ ,

$$f(x, u) = \begin{bmatrix} -x_1 + x_2 \\ -0.5x_1 - 0.5x_2 + 0.5x_2(\cos(2x_1) + 2)^2 \end{bmatrix} + \begin{bmatrix} 0 \\ \cos(2x_1) + 2 \end{bmatrix} u.$$

For each of the problems, a number of  $q = 10^4$  equally spaced grid points  $x^s$  in  $\mathcal{X} = \{x \in \mathbb{R}^2 \mid -1 \leq x_i \leq 1\}$  were chosen to obtain a training data set  $(x^s, u^s)$ ,  $s \in \{0, \dots, q\}$ . This was accomplished by solving a finite-horizon optimization problem:

$$\min_{u(t)} 10(x_1^2 + x_2^2) + \int_0^{10} (x_1^2 + x_2^2) \cdot dt \quad (49)$$

$$\text{subject to: } \dot{x}(t) = f(x(t), u(t)), x(0) = x^s, \quad (50)$$

$$x(t) \in \mathcal{X}, u(t) \in \mathcal{U} \quad (51)$$

for any  $x^s$ , leading to the optimal input  $u^s$  to be applied in  $x^s$ . The optimization problems were solved with the software tool `GPOPS-III` (Patterson and Rao, 2014).

For each NN controller, a structure with one hidden layer consisting of 30 units has been found to be appropriate, and the hyperbolic tangent  $\tanh(\mu) = \sigma(\mu, 2, 1)$  was used as activation function in all units. The neural networks were trained using Matlab's deep learning toolbox with the Levenberg-Marquardt training algorithm (Hagan and Menhaj, 1994).

For Benchmark 1, the NN controller had originally a bias  $b^{(L)} = 1.5865$  after training. The adaption of  $b^{(L)}$  according to (43) led to  $b^{(L)} = 1.5860$ . After adaption, the NN controller led to the closed-loop system with matrix:

$$A_{\text{cl,NN}} = \begin{bmatrix} 0 & 1 \\ -1.7419 & -2.2852 \end{bmatrix}$$

according to (47). Since  $\text{Re}[\lambda_1] = \text{Re}[\lambda_2] \approx -1.1426$  are the real parts of the eigenvalues of  $A_{\text{cl,NN}}$ , Thm. 3 implies that the origin is an exponentially stable equilibrium point. The matrix  $A_{\text{cl,NN}}$  has been used to obtain a quadratic Lyapunov function  $V(x)$  by solving the Lyapunov equation for  $Q$  chosen as identity matrix. Subsequently, INTLAB (Rump, 1999) (a Matlab toolbox for reliable computing) has been used to search for intervals in which  $\dot{V}(x)$  is definitely negative. Intervals with states for which  $\dot{V}(x)$  is or could be greater than zero are shown in Fig. 1 as filled boxes. The subsets  $I_2$  and  $I_1$  of  $\mathcal{X}$  in Fig. 1 are positively invariant, such that Thm. 4 guarantees that each solution starting in  $I_2$  stays therein (hence does not violate the state constraints) and gets into  $I_1$  after a finite time, in which it remains for all future time. The set  $I_1$  is a level set  $\Omega_c = \{x \in \mathbb{R}^n \mid V(x) \leq c\}$ . On the other hand,  $I_2$  is a polytope, for which INTLAB has verified the condition (14).

The same procedure was applied analogously to Benchmark 2 and 3, leading also to the result that the respective synthesized controllers stabilize the systems to the origin. While the results for Benchmark 1 are illustrated in the left column of Fig. 1, the corresponding ones for Benchmark 2 and 3 are illustrated in the middle and right column, respectively.



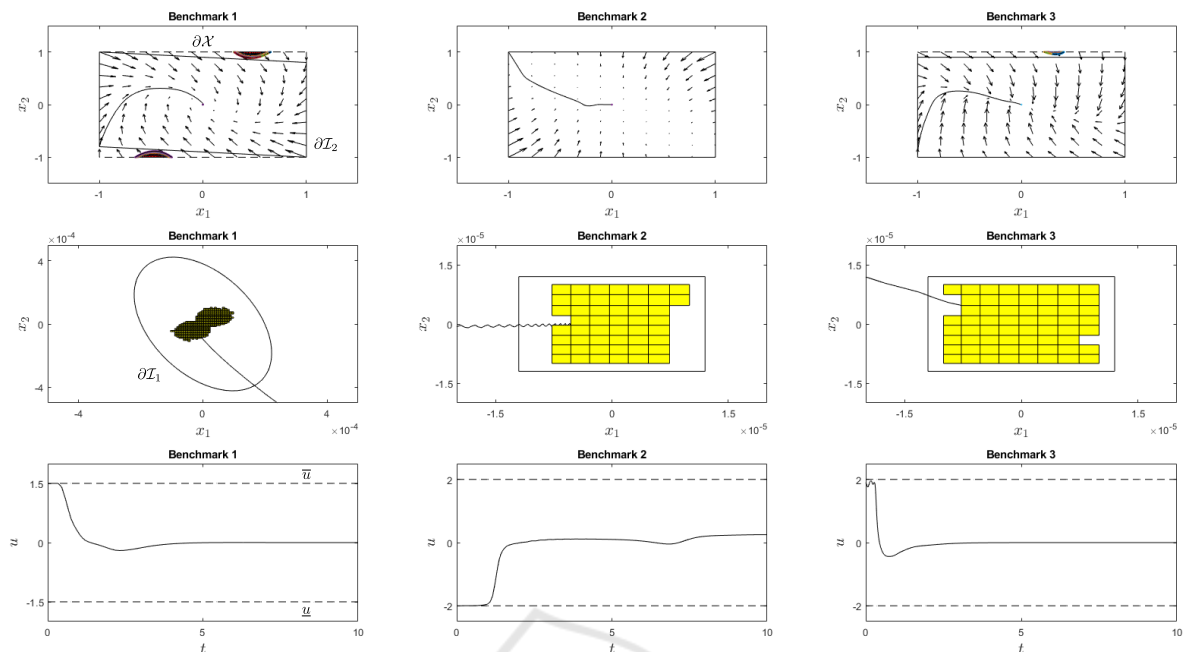


Figure 1: The figure shows the results obtained for the three benchmark problems, where the  $i$ -th column refers to Benchmark  $i$ . The first row illustrates the phase portraits of the closed-loop systems with the NN controllers. For each benchmark problem, the boundary of  $\mathcal{X}$  is illustrated by a dashed line, while a solid line is used to illustrate the boundary of  $I_2$ , the set verified to be positively invariant. Filled boxes show very small areas in which the derivative of the Lyapunov functions (determined by use of Thm. 3) along the trajectories of the closed-loop systems could not be verified to be negative by interval arithmetics. An enlarged version of the neighborhood of the origin is shown in the second row of figures: the boundary of the positively invariant set  $I_1 \subset I_2$  is illustrated for each benchmark problem by a solid line. Since the Lyapunov function determined for Benchmark  $i$  is strictly decreasing along the trajectories within  $\Lambda = I_2 \setminus I_1^o$ , each state trajectory starting in  $I_2$  reaches  $I_1$  in finite time, where it remains forever. This is shown for exemplary state trajectories, as well as for the corresponding input trajectories in the third row of figures, showing that the input constraints (dashed lines) are satisfied.

## 6 CONCLUSION

This work has proposed a scheme for synthesizing nonlinear controllers for nonlinear plants, such that stabilization to an equilibrium point as well as the satisfaction of input constraints is guaranteed. The controller is established as NN controller with continuous and continuously differentiable activation functions in the units of its layers. By using sigmoid functions as activation functions in the output layer, the satisfaction of input constraints is ensured, even if the same sigmoid functions are used in each unit. A closed-form expression for the partial derivative of the NN controller with respect to its input was derived. Furthermore, insight was provided in how to modify the bias vector in the output layer in order to ensure that the origin is indeed an equilibrium point of the closed-loop system. These aspects allow for the use of Lyapunov's indirect method in order to verify that the controlled system is asymptotically stable with respect to the equilibrium point. For the case that the verification of asymptotic stability is successful, a

quadratic Lyapunov function can be found by solving the Lyapunov equation for a selected positive definite matrix. This has been used in the numerical examples in order to verify safety and performance properties with a software for reliable computations on the basis of interval arithmetics. Future work addresses necessary modifications for higher-dimensional spaces as well as the investigation of other candidate Lyapunov functions.

## REFERENCES

- Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.
- Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.
- Blanchini, F. and Miani, S. (2008). *Set-Theoretic Methods in Control*. Birkhäuser Boston.
- Chang, Y.-C., Roohi, N., and Gao, S. (2020). Neural Lyapunov Control. *arXiv e-print:2005.00611*.
- Cybenko, G. (1989). Approximation by superpositions of a

- sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.
- Dutta, S., Chen, X., and Sankaranarayanan, S. (2019). 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*, pages 157–168.
- Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. (2018). Output range analysis for deep feedforward neural networks. In *NASA Formal Methods*, volume 10811 of *Lecture Notes in Computer Science*, pages 121–138. Springer International Publishing.
- Fazlyab, M., Morari, M., and Pappas, G. J. (2020). Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control (early access)*. doi: 10.1109/TAC.2020.3046193.
- Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. J. (2019). Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information processing Systems 32*, pages 11423–11434.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. The MIT Press.
- Grüne, L. (2020). Computing Lyapunov functions using deep neural networks. *arXiv e-print:2005.08965*.
- Hagan, M. T. and Menhaj, M. B. (1994). Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6):989–993.
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.
- Hu, H., Fazlyab, M., Morari, M., and Pappas, G. J. (2020). Reach-sdp: Reachability analysis of closed-loop systems with neural network controllers via semidefinite programming. In *59th IEEE Conference on Decision and Control*, pages 5929–5934.
- Huang, C., Fan, J., Li, W., Chen, X., and Zhu, Q. (2019). Reachnn: Reachability analysis of neural-network controlled systems. *ACM Transactions on Embedded Computing Systems*, 18(5s):1–22.
- Ivanov, R., Weimer, J., Alur, R., Pappas, G. J., and Lee, I. (2019). Verisig: Verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178.
- Khalil, H. K. (2002). *Nonlinear systems*. Prentice Hall.
- Khalil, H. K. (2015). *Nonlinear control*. Pearson.
- Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world. *arXiv e-print:1607.02533*.
- Lewis, F. L. and Liu, D., editors (2013). *Reinforcement learning and approximate dynamic programming for feedback control*. IEEE Press and Wiley.
- Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H. (2017). *Adaptive Dynamic Programming with Applications in Optimal Control*. Springer International Publishing.
- Markolf, L., Eilbrecht, J., and Stursberg, O. (2020). Trajectory planning for autonomous vehicles combining nonlinear optimal control and supervised learning. *IFAC-PapersOnLine*, 53(2):15608–15614.
- Patterson, M. A. and Rao, A. V. (2014). Gpops-ii. *ACM Transactions on Mathematical Software*, 41(1):1–37.
- Perkins, T. J. and Barto, A. G. (2002). Lyapunov design for safe reinforcement learning. *Journal of Machine Learning Research*, 3:803–832.
- Petridis, V. and Petridis, S. (2006). Construction of neural network based lyapunov functions. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 5059–5065.
- Richards, S. M., Berkenkamp, F., and Krause, A. (2018). The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of The 2nd Conference on Robot Learning*, volume 87, pages 466–476.
- Rump, S. (1999). INTLAB - INTerval LABoratory. In Csendes, T., editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers.
- Sutton, R. S. and Barto, A. (2018). *Reinforcement Learning: An Introduction*. The MIT Press.