# Approximation of Separable Control Lyapunov Functions with Neural Networks [⋆]

Mario Sperl [a], Jonas Mysliwitz [a], Lars Grüne [a]

[a] *Mathematical Institute, University of Bayreuth, Bayreuth, Germany*

**Abstract**

In this paper, we investigate the ability of deep neural networks to provide curse-of-dimensionality-free approximations of control Lyapunov functions. To achieve this, we first prove an error bound for the approximation of separable functions with neural networks. Subsequently, we discuss conditions on the existence of separable control Lyapunov functions, drawing upon tools from nonlinear control theory. This enables us to bridge the gap between neural networks and the approximation of control Lyapunov functions as we identify conditions that allow neural networks to effectively mitigate the curse of dimensionality when approximating control Lyapunov functions. Moreover, we present a suitable network architecture and a corresponding training algorithm. The training process is illustrated using two 10-dimensional control systems.

*Key words:* control Lyapunov functions; neural networks; curse of dimensionality.

## 1 Introduction

Control Lyapunov functions (clfs) are a well-established tool in nonlinear control theory. They serve as a certificate of asymptotic null-controllability and can also be used to examine robustness against uncertainties and disturbances or to study performance criteria. However, their most common application lies in designing stabilizing feedback laws using the clf as guidance towards the equilibrium [4]. Given some asymptotically controllable system, we are thus interested in finding a corresponding clf. Since, in general, it is quite hard to compute clfs analytically, we rely on numerical methods. However, traditional numerical methods, which rely on a grid-based approach for the computation of the derivative of the clf, suffer from the curse of dimensionality. This means that, to achieve a certain accuracy, the number of required grid points and, thus, the numerical effort grows exponentially in the dimension of the state space. Consequently, such approaches become impractical in high dimensions.

This paper concerns the use of (deep) neural networks (NNs) to circumvent the curse of dimensionality for approximating clfs. Our approach is related to the work [39], which investigates structural properties on control systems that allow for an exact representation of a (possibly discontinuous) stabilizing feedback by NNs. Further, there exist several papers that present algorithms for the computation of clfs by NNs, see, e.g. [24,26,34]. However, while the algorithms therein have similarities with our numerical approach, none of them provides a complexity analysis regarding the curse of dimensionality. Establishing conditions for a curse-of-dimensionality-free approximation of clfs is the main contribution of this work. Addressing this challenge requires the identification of a suitable class of functions that can be approximated by NNs without suffering from the curse of dimensionality.

There exist various recent papers that discuss results regarding a curse-of-dimensionality-free approximation of solutions of particular kinds of partial differential equations, see, e.g., [3,10,14,18,19,33]. In particular, some of these references exploit the smoothness of solutions of 2nd order Hamilton-Jacobi-Bellman equations for a curse-of-dimensionality-free approximation to solve optimal control problems. However, when it comes to computing a clf for a deterministic system, which can be characterized as a solution of a particular first-order Hamilton-Jacobi-Bellman equation, we cannot expect such a level of smoothness. Thus, we rely on a different structural assumption that allows NNs to overcome

[⋆] A preliminary version of this paper was presented at the 12th IFAC Symposium on Nonlinear Control Systems NOLCOS 2022, see [15]. Corresponding author M. Sperl.

*Email addresses:* mario.sperl@uni-bayreuth.de (Mario Sperl), jonas.mysliwitz@uni-bayreuth.de (Jonas Mysliwitz), lars.gruene@uni-bayreuth.de (Lars Grüne).

the curse of dimensionality. To this end, we consider so-called separable functions. Informally speaking, a mapping is called separable if it can be written as a sum of functions that are each defined on some lower-dimensional domain. Separable functions fall into the class of compositional functions. The ability of NNs to overcome the curse of dimensionality for compositional functions has been discussed in [32]. A very detailed presentation of compositional functions, the corresponding network architecture, and a complexity analysis can be found in [21]. Furthermore, curse-of-dimensionality-free approximations of compositional functions have also recently been studied in the works [9] and [35]. Compared to general compositional functions, separable functions have a simpler structure that allows for more precise estimates, while the classes of control systems admitting separable clfs are still non-trivial.

*Contribution*

In this paper, we bridge the gap between NN approximation theory and the computation of clfs via NNs. Based on [16], we provide complexity results regarding the approximation of separable functions. Next, we extend the results for Lyapunov functions in [16] to clfs. Specifically, we use methods from nonlinear control theory to identify conditions on the control system such that a separable clf exists. Additionally, we expand upon the discussions in [15] to explore achieving separability through a state space transformation. Overall, we identify scenarios where NNs can provably overcome the curse of dimensionality in the computation of clfs. Finally, we propose the corresponding network architecture and training algorithm including the used loss function. In this context, we would also like to mention those topics that are not part of this paper. While this paper provides an expressivity result and proposes a training algorithm, it does not delve into the analysis of the convergence of the training algorithm or the generalization properties of the NN. Also, we only consider the case in which smooth clfs exist, which allows us to better focus on the main results of this paper. Nonsmooth clfs will be addressed in future research.

*Outline*

The remainder of this paper is organized as follows: The problem formulation is introduced in the next section. Afterwards, we provide a complexity analysis regarding the approximation of separable functions with NNs. In Section 4 we focus on the existence of separable clfs. To this end, we first discuss the use of techniques from nonlinear control theory that lead to separability and then consider the existence of separable clfs after suitable state space transformations. In Section 5 we introduce a numerical algorithm for training our NN to represent a clf and apply it to two test cases. Finally, Section 6 concludes the paper.

*Notation*

For $n \in \mathbb{N}$ we set $[n] := \{1, \dots, n\}$ and define $I_n \in \mathbb{R}^{n \times n}$ to be the identity matrix. Let $K \subset \mathbb{R}^n$ be some compact set. Then we denote the infinity norm for continuous functions $f$ on $K$ via $\|f\|_{\infty,K} := \sup_{x \in K} \|f(x)\|$. The symbol $D$ is used to denote the classic differential operator. Moreover, for some multi-index $\alpha \in \mathbb{N}^n$ we use $D_\alpha$ to denote the higher-order partial derivative with respect to $\alpha$. We make use of the comparison functions $\mathcal{K}$ and $\mathcal{K}_\infty$, where $\mathcal{K}$ denotes all continuous and strictly increasing functions $\gamma\colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with $\gamma(0) = 0$ and $\mathcal{K}_\infty$ comprises all $\mathcal{K}$-functions that satisfy $\lim_{r \to \infty} \gamma(r) = \infty$.

## 2 Problem Formulation

We consider a control system of the form

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

where the right-hand side $f\colon \mathbb{R}^n \times U \to \mathbb{R}^n$ is continuous, locally Lipschitz in $x$, and has an equilibrium at 0, i.e., $f(0, 0) = 0$. The input set is denoted as $U \subset \mathbb{R}^m$ and the admissible control functions are given as the set of measurable and locally essentially bounded functions $u\colon \mathbb{R}_{\geq 0} \to U$. In order to avoid technicalities, we assume our system (1) to be defined on the whole domain $\mathbb{R}^n$. We are interested in stabilizing the system towards the origin. To this end, we assume the control system (1) to be asymptotically controllable. In [38, Theorem 2.5] it has been shown that asymptotic controllability is equivalent to the existence of a clf in the sense of Dini. However, in the scope of this paper, we will only consider the case where our control system (1) admits a continuously differentiable clf, where the Dini derivate equals the gradient. This allows us to ensure compatibility with some theorems from the literature cited in the subsequent sections and avoids distracting technical difficulties. The important case that no smooth clf exists will be investigated in future research, cf. Section 6.

**Definition 1** *A continuously differentiable function $V\colon \mathbb{R}^n \to \mathbb{R}$ is called (smooth) control Lyapunov function (clf) for (1) if there exist $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ and $\alpha_3 \in \mathcal{K}$ such that for $x \in \mathbb{R}^n$*

$$\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|), \tag{2a}$$
$$\inf_{u \in U} DV(x) f(x, u) \leq -\alpha_3(\|x\|). \tag{2b}$$

Note that the existence of $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$ such that the two inequalities in (2a) hold is equivalent to the fact that $V$ is positive-definite and radially unbounded, cf. Lemma 4.3 in [23]. The reason for our preference of using the $\mathcal{K}_\infty$ functions $\alpha_1$ and $\alpha_2$ is due to our numerical algorithm, see Section 5. Given the existence of a smooth clf, it is our objective to numerically compute an approximation thereof on a compact set $K \subset \mathbb{R}^n$ via NNs. To this end, the approximation of functions via NNs is discussed in the following section.

## 3 Neural Networks Approximating Separable Functions

In the first part of this section, we briefly outline the general functionality of NNs, clarify our notation, and recall the universal approximation theorem. Subsequently, we define the notion of separable functions and state an approximation result for such kind of functions that extends Theorem 5.1 in [17].

### *3.1 Preliminaries on Neural Networks*

From a mathematical point of view, a (deep) neural network (NN) is a mapping that takes some input vector $x \in \mathbb{R}^n$ and processes it according to the structure and parameters that define the network in order to return some output. A NN consists of a certain number $P \in \mathbb{N}$ of layers that contain $N_l \in \mathbb{N}$, $0 \leq l \leq P-1$, neurons each. We denote the value of the neuron at position $k$ in layer $l$ with $y_k^l \in \mathbb{R}$. The first layer is the input layer, whence it consists of $N_0 = n$ neurons. The last layer is called output layer and all other layers are referred to as hidden layers. In case of a feedforward network, the value of a neuron is determined by the values of all neurons in the previous layer, that is, for $1 \leq l \leq P-1$ and $1 \leq k \leq N_l$ it holds that

$$y_k^l = \sigma^l \left( \sum_{i=1}^{N_{l-1}} w_{k,i}^l y_i^{l-1} + b_k^l \right), \tag{3}$$

where $w_{k,i}^l \in \mathbb{R}$, $1 \leq i \leq N_{l-1}$, are weight parameters, $b_k^l \in \mathbb{R}$ constitutes a bias term and $\sigma^l \colon \mathbb{R} \to \mathbb{R}$ is the activation function of layer $l$. Applying (3) to the last layer and substituting for the neurons in the previous layers, one obtains the mapping represented by the network. We denote this mapping as $x \mapsto W(x;\theta)$, where $\theta \in \mathbb{R}^q$ comprises all weights and biases of the network with $q = \sum_{l=1}^{P-1}(N_{l-1}+1)N_l$. Training of a NN means adapting $\theta$ to achieve the desired behavior of the resulting function $W(x;\theta)$. Parameters that are set before the training process and remain constant, such as the number and size of the layers, are referred to as hyperparameters. Throughout this work we solely consider feedforward networks with a one-dimensional output $W(x;\theta) \in \mathbb{R}$ and the identity as activation function in the last layer, i.e., $\sigma^{P-1} = I_1$.

It is now natural to ask which kind of functions can be approximated by NNs as in (3). It has been shown in [8] that the set of functions given by a NN with one hidden layer and a continuous sigmoidal activation function is dense in $C([0,1]^n)$. An overview of some extensions regarding single-layer NNs as universal approximators can be found in [31]. Since we are interested in the numerical effort needed for approximating a clf, we need a quantitative version of an approximation theorem. To this end, we characterize the complexity of a NN by the number of neurons in its hidden layers. It is worth noting that in the literature, complexity is also often quantified by the number of parameters. However, since the relation between the number of neurons and the number of parameters is polynomial, there is no difference regarding an investigation of the curse of dimensionality. We conclude this subsection by citing an approximation result that quantifies the number of neurons needed to achieve an approximation up to a certain accuracy. To this end, for $m \in \mathbb{N}, r \in \mathbb{R}_{>0}$, and a compact set $K \subset \mathbb{R}^n$ we introduce the Sobolev-like space

$$W_{m,r}(K) := \left\{ F \in C^m(K,\mathbb{R}) \;\middle|\; \|F\|_{W_m(K)} \leq r \right\},$$

where $\|F\|_{W_m(K)} := \sum_{0\leq|\alpha|\leq m} \|D_\alpha F\|_{\infty,K}$.

**Theorem 2** *Let $R \in \mathbb{R}_{>0}$ and $\sigma \in \mathcal{C}^\infty(\mathbb{R},\mathbb{R})$ be not a polynomial. Then for every $n \in \mathbb{N}$ there exists a constant $\mu_n > 0$ such that for all $M \in \mathbb{N}$ a NN $W(x;\theta)$ with one hidden layer consisting of $M$ neurons and activation function $\sigma^1 = \sigma$ satisfies for all $F \in C^m(K,\mathbb{R})$*

$$\inf_\theta \|W(\cdot;\theta) - F(\cdot)\|_{\infty,K} \leq \mu_n M^{-\frac{m}{n}} \widetilde{R} \|F\|_{W_m(K)},$$

*where $K := [-R,R]^n$, $\widetilde{R} := \max\{R,1\}$.*

Note that the constant $\mu_n$ depends on $n$ but is independent of $F$ and $M$. Theorem 2 has been derived in [27, Theorem 2.1] for the case $R = 1$ and its extension to $R \in \mathbb{R}_{>0}$ is proven in [20, Corollary 1]. The theorem has originally been stated for $\sigma \in \mathcal{C}^\infty(\mathbb{R},\mathbb{R})$ such that there exists $b \in \mathbb{R}$ with $\sigma^{(i)}(b) \neq 0$ for all $i \in \mathbb{N}$. It has been discussed in [32] that this condition is equivalent to $\sigma$ not being a polynomial. We can conclude from Theorem 2 that the number of neurons needed to provide an approximation up to some accuracy $\varepsilon > 0$ is given by $M = \mathcal{O}(\varepsilon^{-\frac{n}{m}})$, i.e., the number depends on the degree of smoothness $m$ and the dimension of the domain $n$. Moreover, it has also been shown in [27] that the order of neurons $\mathcal{O}(\varepsilon^{-\frac{n}{m}})$ is best possible. Thus, in general, NNs also suffer from the curse of dimensionality since the number of needed neurons to achieve a certain accuracy grows exponentially in $n$. Similar results exist for other kinds of activation functions, e.g., for ReLU functions $\sigma(x) = \max(0,x)$ in [43].

### *3.2 Overcoming the Curse of Dimensionality with Neural Networks*

The central part of this section is a result showing that NNs are capable of overcoming the curse of dimensionality for so-called separable functions.

**Definition 3** *Let $F \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R})$ and $d \in [n]$. Then $F$ is called (strictly) $d$-separable if there exist $s \in [n]$, $d_j \in [d]$, $j \in [s]$, and functions $F_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, \mathbb{R})$ such that for all $x \in \mathbb{R}^n$ it holds*

$$F(x) = \sum_{j=1}^{s} F_j(z_j), \tag{4}$$

*where $z_j = (x_{k_{j-1}}, \dots, x_{k_j - 1})$ with $k_0 := 1$ and $k_j := k_{j-1} + d_j$, $j \in [s]$.*

In other words, if $F$ is a $d$-separable function, its domain can be split into $s$ subspaces, having the form $\mathbb{R}^n = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_s}$, such that $F$ can be written as a sum of $s$ functions, which are defined on the respective subspaces. For the purpose of this paper it is sufficient to consider *strictly* separable functions, which means that the intersection of two such subspaces is always the origin, i.e., the domains are not overlapping. Constructions with overlapping regions have been pursued, e.g., in [28,42]. For simplicity, we will omit the term "strictly" in what follows. The benefit of a separable structure can be exploited by a NN as shown in Figure 1. It consists of two hidden layers, where the first one has a linear activation $\sigma^1 = I_1$, and the second one uses some nonlinear activation function $\sigma^2$. However, the first and second hidden layers are not fully connected but divide the NN into $n$ sublayers. These sublayers can be used to learn the functions $F_j$ in (4). The following theorem quantifies the ability of this network architecture to efficiently represent separable functions.

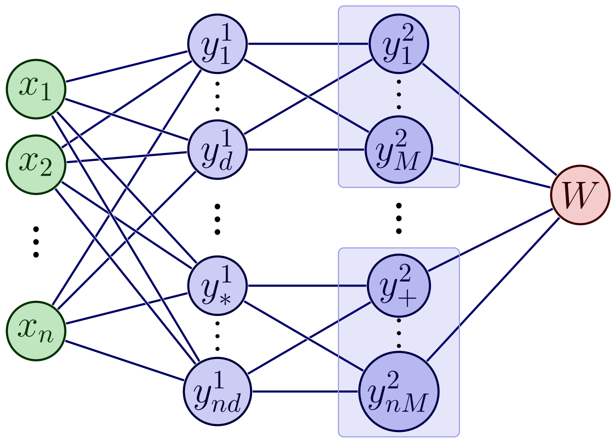


Fig. 1. Architecture of the NN with $* = (n-1)d + 1$, $+ = (n-1)M + 1$, and $W = W(x;\theta)$.

**Theorem 4** *Let $d \in \mathbb{N}$, $r, R \in \mathbb{R}_{>0}$, and $\sigma \in \mathcal{C}^\infty(\mathbb{R}, \mathbb{R})$ be not a polynomial. Define for $n \in \mathbb{N}$ the sets $K_n := [-R, R]^n$ and*

$$\mathcal{F}_{r,d}^{(n)} := \left\{ F \in W_{1,r}(K_n) \,\middle|\, F \text{ is } d\text{-separable}, F(0) = 0 \right\}.$$

*Then there exists a constant $\mu_d > 0$ such that for all $n \in \mathbb{N}$ and $M \in \mathbb{N}$ the NN $W(x;\theta)$ depicted in Figure 1 with $n(d+M)$ neurons and activation functions $\sigma^1 = I$ and $\sigma^2 = \sigma$ satisfies for all $F \in \mathcal{F}_{r,d}^{(n)}$*

$$\inf_\theta \|F(\cdot) - W(\cdot;\theta)\|_{\infty, K_n} \le n r \mu_d \max\{R, 1\} M^{-\frac{1}{d}}.$$

**PROOF.** By virtue of Theorem 2 there exists a constant $\mu_d > 0$ such that for all $M \in \mathbb{N}$ and $\widetilde{F} \in W_{1,r}(K_d)$ it holds that

$$\inf_{\tilde\theta} \|\widetilde{F}(\cdot) - \widetilde{W}(\cdot;\tilde\theta)\|_{\infty, K_d} \le r \mu_d M^{-\frac{1}{d}} \max\{R, 1\}, \tag{5}$$

where $\widetilde{W}(x;\theta)$ is a single-layer NN with activation function $\sigma^1 = \sigma$ and $M$ neurons in its hidden layer. Now fix $n, M \in \mathbb{N}$ and some $F \in \mathcal{F}_{r,d}^{(n)}$. Since $F$ is $d$-separable (see Definition 3), we can write $F(x) = \sum_{j=1}^{s} F_j(z_j)$ for some $s \in [n]$ and $F_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, \mathbb{R})$. As $F(0) = 0$, we have $\sum_{j=1}^{s} F_j(0) = 0$. Thus, by redefining $F_j(z_j) := F_j(z_j) - F_j(0)$, we can assume that $F_j(0) = 0$ for $j \in [s]$. This yields for $z_j \in \mathbb{R}^{d_j}$

$$F_j(z_j) = F_j(z_j) + \sum_{i \ne j} F_i(0) = F(0, \dots, 0, z_j, 0, \dots, 0). \tag{6}$$

Further, observe that for $x \in \mathbb{R}^n$

$$DF(x) = \Big[ DF_1(z_1) \; DF_2(z_2) \; \cdots \; DF_s(z_s) \Big]. \tag{7}$$

Consequently, we obtain from (6) and (7) for $j \in [s]$

$$\begin{aligned} \|F_j\|_{W_1(K_d)} &= \sum_{0 \le |\alpha| \le 1} \|D_\alpha F_j\|_{\infty, K_d} \\ &\le \sum_{0 \le |\alpha| \le 1} \|D_\alpha F\|_{\infty, K_n} \le r, \end{aligned}$$

whence $F_j \in W_{1,r}(K_d)$. Now we want to set the weights and biases corresponding to the first hidden layer of the network depicted in Figure 1 such that its first $s$ sublayers contain the vectors $z_j$, $j \in [s]$, respectively. To this end, we set $b_k^1 = 0$ for $k \in [nd]$ and define

$$w_{1,1}^1 = w_{2,2}^1 = \cdots = w_{d_1,d_1}^1 = 1,$$

in order to obtain the vector $z_1$ in the first sublayer, cf. (3). Next, we set $w_{n+i,d_1+i}^1 = 1$ for $i \in [d_2]$, as well as

$$w_{2n+i,d_1+d_2+i}^1 = 1,$$

for $i \in [d_3]$, to get $z_2$ and $z_3$ in the second and third sublayer, respectively. We continue this procedure until the $s$-th sublayer. All remaining weights in the first hidden layer, i.e., the weights that belong to some sublayer $j > s$

or to some sublayer $j \in [s]$, but exceed the number $d_j$, are set to 0. Furthermore, for the output layer we choose $w^3_{1,i} = 1$ for $i \in [dM]$, $w^3_{1,i} = 0$ for $i > dM$, and $b^3_1 = 0$. Observe that the output of the NN is now given as

$$W(x;\theta) = \sum_{j=1}^{n}\sum_{i=1}^{M} y^2_{(j-1)M+i} = \sum_{j=1}^{s}\sum_{i=1}^{M} y^2_{(j-1)M+i}, \quad (8)$$

where for each $j \in [s]$ the output $\sum_{i=1}^{M} y^2_{(j-1)M+i}$ of the $j$-th sublayer can be interpreted as the output of a NN with input $z_j$ and one hidden layer consisting of $M$ neurons, cf. Figure 1. Let us denote the respective subnetwork by $W_j(z_j;\theta_j)$. By applying (5) we obtain for $j \in [s]$

$$\inf_{\theta_j} \|F_j(\cdot) - W_j(\cdot;\theta_j)\|_{\infty,K_d} \le r\mu_d M^{-\frac{1}{d}} \max\{R,1\} := \rho$$

Finally, invoking (8) gives us for $x \in K_n$

$$\inf_{\theta} \|F(x) - W(x;\theta)\| = \inf_{\theta} \Big\| \sum_{j=1}^{s} F_j(z_j) - W_j(z_j;\theta_j) \Big\|$$
$$\le \sum_{j=1}^{s} \inf_{\theta_j} \|F_j(z_j) - W_j(z_j;\theta_j)\| \le s\rho.$$

Since $s \le n$, this shows the claim. $\square$

In the proof of Theorem 4, we have used the first (linear) layer of the network displayed in Figure 1 to compute the decomposition of the state $x$ into vectors $z_j$, $1 \le j \le s$, according to Definition 3. We can thus identify the first layer with the mapping $x \mapsto W^1 x$, where $W^1 \in \mathbb{R}^{nd \times n}$ denotes the matrix that represents the corresponding decomposition of $x$. However, since the weights $w^1_{k,i}$, $k \in [nd]$, $i \in [n]$, can take on any real value, the first layer of the NN depicted in Figure 1 can in fact express any matrix $W^1 \in \mathbb{R}^{nd\times n}$. This observation motivates the following definition.

**Definition 5** *Let $d \in [n]$, $F \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R})$, and $T \in \mathbb{R}^{n\times n}$ be invertible. Then $F$ is called linearly $d$-separable with respect to $T$ if the mapping*

$$x \mapsto F(Tx)$$

*is (strictly) $d$-separable.*
*Further, a function $G \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R}^l)$ is called linearly $d$-separable if each of its $l$ component functions is linearly $d$-separable.*

Definition 5 extends the class of separable functions to all functions that are separable after a suitable linear transformation of the state space. The following corollary generalizes Theorem 4 to the case of linearly $d$-separable functions. To this end, for $c \in \mathbb{R}_{>0}$ we define $GL^c_n$ as the space of invertible matrices $T \in \mathbb{R}^{n\times n}$ such that $\|T\|_\infty \le c$ and $\|T^{-1}\|_\infty \le c$. Note that after rescaling with $c/\|T\|_\infty$, every $T \in \mathbb{R}^{n\times n}$ with condition number $\le c^2$ lies in $GL^c_n$.

**Corollary 6** *Let $d \in \mathbb{N}$, $c, r, R \in \mathbb{R}_{>0}$, and $\sigma \in \mathcal{C}^\infty(\mathbb{R}, \mathbb{R})$ be not a polynomial. Define for $n \in \mathbb{N}$ the sets $K_n := [-R,R]^n$ and*

$$\mathcal{F}^{(n)}_{r,d,c} := \Big\{ F \in W_{1,r}(K_n) \;\Big|\; F \text{ is linearly } d\text{-separable w.r.t. some } T \in GL^c_n, F(0) = 0 \Big\}.$$

*Then there exists a constant $\mu_d > 0$ such that for all $n \in \mathbb{N}$ and $M \in \mathbb{N}$ the NN $W(x;\theta)$ depicted in Figure 1 with $n(d+M)$ neurons and activation functions $\sigma^1 = I_1$ and $\sigma^2 = \sigma$ satisfies for all $F \in \mathcal{F}^{(n)}_{r,d,c}$*

$$\inf_{\theta} \|F(\cdot) - W(\cdot;\theta)\|_{\infty,K} \le cnr\mu_d \max\{cR,1\} M^{-\frac{1}{d}}.$$

**PROOF.** Let $F \in \mathcal{F}^{(n)}_{r,d,c}$. Consider the mapping $G\colon T^{-1}K_n \to \mathbb{R}, x \mapsto F(Tx)$. By assumption, $G$ is a $d$-separable function. Further, note that $G(0) = 0$ and $T^{-1}K_n \subset cK_n = [-cR, cR]^n$. Moreover, it holds that

$$\|G\|_{W_1(T^{-1}K_n)} = \sum_{0\le|\alpha|\le 1} \|D_\alpha F(T\cdot)\|_{\infty,T^{-1}K_n}$$
$$= \|F(T\cdot)\|_{\infty,T^{-1}K_n} + \sum_{j=1}^{n} \|\frac{\partial}{\partial x_j} F(T\cdot)\|_{\infty,T^{-1}K_n}$$
$$\le \|F\|_{\infty,K_n} + c\sum_{j=1}^{n} \|\frac{\partial}{\partial x_j} F\|_{\infty,K_n} \le c\|F\|_{W_1(K_n)}.$$

Hence, applying Theorem 4 yields for $M \in \mathbb{N}$

$$\inf_{\theta} \|G(\cdot) - W(\cdot;\theta)\|_{\infty,T^{-1}K_n} \le ncr\mu_d \max\{cR,1\} M^{-\frac{1}{d}},$$

where $W(x;\theta)$ is the NN constructed in the proof of Theorem 4. Recall that its first hidden layer used the identity as activation function and all biases of the first hidden layer were set to 0. Thus, the neurons of the first hidden layer are obtained as $W^1 x$ for some matrix $W^1 \in \mathbb{R}^{nd\times n}$. Hence, having $T^{-1}K_n$ as input space and the matrix $W^1$ representing the first hidden layer can equivalently be replaced by using $K$ as input space and redefining $W^1 := W^1 T^{-1}$. With this, we obtain an approximation of $F$ on $K$. Since the linear transformation of the weights in the first layer is already included in the infimum over all parameters $\theta$, we obtain the claim. $\square$

As an immediate consequence of Corollary 6 we obtain that the number of neurons needed to approximate linearly $d$-separable functions grows only polynomially in the state dimension $n$.

**Corollary 7** *Let $\varepsilon > 0$ and consider the setting from Corollary 6. Then for $n \in \mathbb{N}$ the number of neurons $N \in \mathbb{N}$ needed to ensure*

$$\sup_{F \in \mathcal{F}^{(n)}_{r,d,c}} \inf_\theta \|F(\cdot) - W(\cdot;\theta)\|_{\infty,[-R,R]^n} \leq \varepsilon$$

*is given by*

$$N = nd + \frac{n^{d+1}}{\varepsilon^d}(cr\mu_d \max\{cR,1\})^d = \mathcal{O}\Big(nd + \frac{n^{d+1}}{\varepsilon^d}\Big).$$

**PROOF.** Applying Corollary 6 yields

$$M \geq (n\, c\, r\mu_d \max\{cR,1\})^d \varepsilon^{-d}.$$

The claim is obtained by counting the total number of neurons in the hidden layers in Figure 1. □

**Remark 8** *In the setting of Theorem 4 as well as in the corollaries 6 and 7 we have worked with $\mathcal{C}^1$ target functions and thus applied Theorem 2 for the case $m = 1$. If one is in a position to apply Theorem 2 for higher values of $m$, one still obtains a number of neurons that is polynomially increasing in $n$, albeit with the lower exponent $d/m + 1$ in place of $d + 1$.*

## 4 Existence of Separable Control Lyapunov Functions

In this section, we use of methods from nonlinear systems theory for providing conditions for the existence of (linearly) separable clfs. Thus, by invoking the results from Section 3 we can identify classes of systems that allow for a curse-of-dimensionality-free approximation of clfs by NNs.

### *4.1 Separability via small-gain theory and active nodes*

This subsection proves the existence of separable clfs based on small gain theory, leveraging the notion of active nodes from [7]. We consider a control system (1) and assume that it can be decomposed into $s \in \mathbb{N}$ subsystems denoted by

$$\Sigma_j: \quad \dot{z}_j = f_j(x, \tilde{u}_j) = f_j(z_j, z_{-j}, \tilde{u}_j), \quad j \in [s], \quad (9)$$

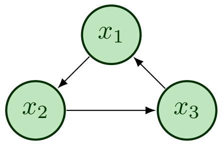
where

$$x = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_s \end{pmatrix}, \quad u = \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_s \end{pmatrix}, \quad f(x,u) = \begin{pmatrix} f_1(x,\tilde{u}_1) \\ f_2(x,\tilde{u}_2) \\ \vdots \\ f_s(x,\tilde{u}_s) \end{pmatrix}$$

with $z_j \in \mathbb{R}^{d_j}$, $U = U_1 \times U_2 \times \cdots \times U_s$, $\tilde{u}_j \in U_j$, $f_j : \mathbb{R}^n \times U_j \to \mathbb{R}^{d_j}$, and

$$z_{-j} := \Big(z_1, \ldots, z_{j-1}, z_{j+1}, \ldots, z_s\Big)^T \in \mathbb{R}^{n-d_j}.$$

We explicitly allow for the case that some subsystems $\Sigma_j$, are independent of the control $u$, which corresponds to the case $U_j = \{0\}$. In the following, we investigate whether there exist functions $V_j$ defined on the respective subspaces $\mathbb{R}^{d_j}$ such that their sum constitutes a clf for the whole system. To this end, in addition to a stability property for each subsystem, one needs to impose a condition on the coupling of the subsystems. For this purpose, we represent the decomposition as a directed graph that consists of $s$ nodes. Each node belongs to one subsystem and there exists an edge from node $i$ to node $j$, $j \neq i$, if the subsystem $i$ influences the subsystem $j$, i.e., if the function $f_j$ depends on the vector $z_i$. Figure 2 illustrates the graph corresponding to a decomposition into 1-dimensional subsystems of the control system (10) from Section 4 in [7].



$$\begin{aligned} \dot{x}_1 &= x_3 + u, \\ \dot{x}_2 &= x_1 - x_2 + x_1^2, \\ \dot{x}_3 &= x_2 - x_3. \end{aligned} \quad (10)$$

Fig. 2. A control system and its corresponding graph.

The following assumption imposes a stability condition on each subsystem:

**Assumption 9** *For each $j \in [s]$ there exists a feedback function $F_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, U_j)$, constants $\alpha_j \in \mathbb{R}_{>0}$, $\gamma_{i,j} \in \mathbb{R}_{>0}$, $i \neq j$, as well as a positive-definite and radially unbounded function $V_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, \mathbb{R})$ such that*

$$DV_j(z_j) f_j(z_j, z_{-j}, F_j(z_j)) \leq -\alpha_j \|z_j\|^2 + \sum_{i \neq j} \gamma_{i,j} \|z_i\|^2. \quad (11)$$

Note that for a subsystem $\Sigma_j$ that is not influenced by the control, the left-hand side in (11) does not depend on any feedback function $F_j$. In particular, Assumption 9 states that for all $j \in [s]$, the function $V_j$ is an ISS-Lyapunov function (see [41]) for the system

$$\dot{z}_j = f_j(z_j, z_{-j}, F_j(z_j)),$$

where $z_{-j}$ is seen as the external input. Given such a stability assumption on each of the subsystems, small-gain theory can be used to obtain a stability property of the overall system, see, for instance, [11,25,36]. In the following, we focus on the theory developed in [7] that allows us to formulate a graph-based criterion regarding the existence of a separable clf.

**Definition 10 (cf. Definition 3 in [7])** *Let $j \in [s]$ and consider a subsystem $\Sigma_j$ as in (9). The subsystem is called active if there exist $\gamma_{i,j} \in \mathbb{R}_{>0}$, $i \neq j$, and a function $V_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, \mathbb{R})$ such that for all $\alpha_j > 0$ there exists $F_j \in \mathcal{C}^1(\mathbb{R}^{d_j}, U_j)$ such that (11) holds.*

Using this notion of active subsystems (or active nodes) in the graph, the results of [7] yield the following proposition, where we call a tuple of nodes $(v_1, v_2, \ldots, v_l)$, $l \geq 3$, cycle if there exists an edge from $v_i$ to $v_{i+1}$ for all $i \in [l-1]$, and $v_1 = v_l$, that is, the starting node equals the last node.

**Proposition 11** *Consider a control system of the form (1) given through subsystems of the form (9) and let Assumption 9 hold. Moreover, assume that in each cycle of the directed graph corresponding to the decomposition (9) there is at least one active subsystem. Then there exists a d-separable clf for the system (1).*

**PROOF.** Let $V_j$, $j \in [s]$, denote the ISS-Lyapunov functions obtained from Assumption 9. Applying Proposition 1 and Proposition 2 in [7], respectively, yields the existence of coefficients $r_j \in \mathbb{R}_{>0}$, $j \in [s]$, such that $V := \sum_{j=0}^{s} r_j V_j$ is a Lyapunov function for

$$\dot{z_j} = f_j(z_j, z_{-j}, F_j(z_j)), \quad j \in [s]$$

This implies that $V$ satisfies condition (2b), whence $V$ is a clf for (1). Finally, redefining $V_j := r_j V_j$ gives us the decomposition of $V$ as $d$-separable function, cf. Definition 3. □

Revisiting the control system in (10), we can check that $V_j(x_j) = x_j^2$ is an ISS-Lyapunov function for each subsystem and that the first subsystem is active. Thus, Proposition 11 yields the existence of a 1-separable clf for Example 10. Overall, by invoking Corollary 7 we can conclude that Proposition 11 identifies a class of control systems, where a clf can be approximated by a NN without the curse of dimensionality. For an extension of the above theory towards systems with non-quadratic $\mathcal{K}_\infty$ functions in (11), we refer to [7, Section III].

### *4.2 Linear separability via linearization*

In this subsection, the discussion of Subsection 4.1 is extended to the existence of *linearly* $d$-separable clfs according to Definition 5. We motivate this extension by considering a variation of (10), namely the control system

$$\begin{aligned} \dot{x}_1 &= x_3 + u, \\ \dot{x}_2 &= x_1 - x_2 + x_2^2, \\ \dot{x}_3 &= x_2 + x_3. \end{aligned} \tag{12}$$

Applying the backstepping procedure (cf. Section 6.1 in [37]) yields the existence of a clf for the system (12). However, the obtained clf is not 1-separable. In fact, it follows from Lemma 5 in [15] that there does not exist a 1-separable clf for (12). However, one can show that the control system falls into a class of systems that possess a *linearly* 1-separable clf, at least on some neighborhood of the origin. In order to formulate the corresponding assertion, we first show that linear stabilizable systems always admit a linearly 1-separable clf.

**Proposition 12** *Consider a linear control system of the form*

$$\dot{x} = Ax + Bu, \tag{13}$$

*where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times m}$. Assume that $(A, B)$ is stabilizable. Then there exists a linearly 1-separable clf $V$ for the system (13). The function $V$ is quadratic, i.e., $V(x) = x^T P x$ for some $P \in \mathbb{R}^{n \times n}$ and satisfies*

$$\inf_{u \in \mathbb{R}^m} DV(x)(Ax + Bu) \leq DV(x)(A + BF)x \leq c\|x\|_2^2 \tag{14}$$

*for a suitable feedback matrix $F \in \mathbb{R}^{n \times m}$ and a constant $c > 0$.*

**PROOF.** It is known that for a linear and stabilizable system, there always exists a clf of the form $V(x) = x^T P x$ for a suitable symmetric and positive definite matrix $P \in \mathbb{R}^{n \times n}$. The inequality (14) then follows from the usual matrix Lyapunov inequality. As $P$ is symmetric and positive-definite, there exists an orthogonal matrix $T \in \mathbb{R}^{n \times n}$ such that

$$\widetilde{P} := T^{-1} P T = T^T P T = \operatorname{diag}(p_1, \ldots, p_n)$$

is a diagonal matrix. Thus,

$$V(Tx) = (Tx)^T P (Tx) = x^T \widetilde{P} x = \sum_{i=1}^{n} p_i x_i^2$$

is a 1-separable function. □

Proposition 12 implies that linearizable control systems locally possess linearly 1-separable clfs.

**Corollary 13** *Consider a control system (1) with a $\mathcal{C}^1$-function $f$ and assume that its linearization at the origin is stabilizable. Then the control system (1) possesses a linearly 1-separable clf on some neighborhood of the origin.*

**PROOF.** Write $f(x, u) = Ax + Bu + g(x, u)$ with

$$\lim_{\|(x,u)\| \to 0} \frac{\|g(x, u)\|}{\|(x, u)\|} = 0.$$

Since $(A, B)$ is stabilizable, Proposition 12 yields the existence of $c \in \mathbb{R}_{\geq 0}$, $F \in \mathbb{R}^{n\times m}$, and a linearly 1-separable function $V(x) = x^T Px$ such that (14) holds. Following the proof of [40, Theorem 19], we obtain

$$\inf_u DV(x)f(x,u) \leq -c\|x\|_2^2 + 2xPg(x, F(x)) < 0$$

for $x$ sufficiently small, since

$$\frac{\|2xPg(x, F(x))\|}{\|x\|^2} \to 0$$

for $x \to 0$. Hence, $V$ is a clf for the nonlinear system (1) in a suitable neighborhood of the origin. □

### 4.3 *Linear separability via feedback linearization*

Next we explore a class of systems for which Proposition 12 can be employed to achieve linear separability through a potential nonlinear transformation. To this end, we extend the definition of feedback linearizability from [40, Section 5.3] to multi-input systems.

**Definition 14** *An affine control system*

$$\dot{x} = f(x) + \sum_{j=1}^m g_j(x)u_j$$

*with control input $u = (u_1, \dots, u_m)^T \in \mathbb{R}^m$ is called feedback linearizable, if there exists a diffeomorphism $S \in \mathcal{C}^1(\mathbb{R}^n, \mathbb{R}^n)$ as well as maps $a_j, b_j\colon \mathbb{R}^n \to \mathbb{R}$, $j \in [m]$, such that the transformed control system*

$$\dot{\tilde{x}} = \tilde{f}(\tilde{x}) + \sum_{j=1}^m \tilde{g}_j(\tilde{x})v_j$$

*with transformed state $\tilde{x} = S(x)$, new control input $v = (v_1, \dots, v_m)^T \in \mathbb{R}^m$ and*

$$\tilde{f}(\tilde{x}) = DS(x)\Big(f(x) + \sum_{j=1}^m a_j(x)g_j(x)\Big),$$
$$\tilde{g}_j(\tilde{x}) = b_j(x)DS(x)g_j(x),$$

*is a linear control system, i.e., if there exist matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ auch that $\tilde{f}(\tilde{x}) = A\tilde{x}$ and $(\tilde{g}_1(\tilde{x}), \dots, g_m(\tilde{x})) = B$ holds for all $\tilde{x} \in \mathbb{R}^n$.*

**Theorem 15** *Consider a feedback linearizable affine control system with transformation map $S$ satisfying $S(0) = 0$, for which the pair $(A, B)$ is stabilizable. Then the control system has a clf $V$ of the form $V(x) = \widetilde{V}(S(x))$ with a linearly 1-separable function $\widetilde{V} : \mathbb{R}^n \to \mathbb{R}$.*

**PROOF.** According to Proposition 12, we have

$$\inf_{v\in\mathbb{R}^m} D\widetilde{V}(\tilde{x})(A\tilde{x}+Bv) \leq D\widetilde{V}(\tilde{x})(A\tilde{x}+BF\tilde{x}) \leq -c\|\tilde{x}\|_2^2$$

for suitable $c \in \mathbb{R}_{\geq 0}$, $F \in \mathbb{R}^{n\times m}$, and some linearly 1-separable mapping $\widetilde{V}$. For $V(x) = \widetilde{V}(S(x))$ and $u_j = a_j(x) + b_j(x)v_j$ we then obtain

$$\begin{aligned} &DV(x)\big(f(x) + \sum_{j=1}^m g_j(x)u_j\big) \\ &= D\widetilde{V}(S(x))DS(x)\big(f(x) + \sum_{j=1}^m g_j(x)(a_j(x) + b_j(x)v_j)\big) \\ &= D\widetilde{V}(\tilde{x})\big(\tilde{f}(\tilde{x}) + \sum_{j=1}^m \tilde{g}_j(\tilde{x})v_j\big) \;=\; D\widetilde{V}(\tilde{x})(A\tilde{x} + Bv). \end{aligned}$$

This implies

$$\inf_{u\in\mathbb{R}^m} DV(x)\Big(f(x) + \sum_{j=1}^m g_j(x)u_j\Big) \leq -c\|S(x)\|_2^2.$$

Since $S$ is a diffeomorphism with $S(0) = 0$, there exist $\alpha_1$, $\alpha_2 \in \mathcal{K}_\infty$ with

$$\alpha_1(\|x\|_2) \leq (\|S(x)\|_2) \leq \alpha_2(\|x\|_2),$$

see Lemma 1 in [22]. Thus, $V$ satisfies all inequalities in (2), whence it is a clf. □

The following corollary now follows immediately.

**Corollary 16** *Consider the setting of Theorem 15 and assume that the transformation map $S$ is linearly $k$-separable for some $k \in [n]$. Then the control system has a clf $V$ that is a composition of a linearly 1-separable function with a linearly $k$-separable function.*

Note that Corollary 16 in particular applies to linear mappings $S$, as linear mappings are always 1-separable. We can conclude that for control systems that satisfy the requirements in Corollary 16 there exists a curse-of-dimensionality-free approximation with a NN that is built as in Figure 1, but has one additional hidden layer at the beginning, which is used to represent the $k$-separable transformation $S$.

## 5 Network architecture and training algorithm

The present section discusses the NN structure and a corresponding training algorithm for the computation of a separable clf, whose performance is illustrated by numerical results for two test cases.

### 5.1 Network Structure

The structure of the NN that we use for the computation of a linearly separable clf is exactly the one depicted in Figure 1 with the modification of introducing a hyperparameter $s$ for the number of sublayers, i.e., replacing the $n$ sublayers in Figure 1 by $s$ sublayers. It is important to note that our NN architecture entails the 3 hyperparameters $d$, $s$, and $M$. The maximal input-dimension of each sublayer $d$ has to be chosen large enough such that the considered control system possesses a $d$-separable clf. Note that if some function is $d$-separable, then it is also $d+1$ separable with the same decomposition of its domain. However, since the numerical effort increases exponentially in $d$ (cf. Theorem 4), a good estimate for $d$ significantly improves the efficiency of the approach. Regarding the choice of $s$, according to Definition 3 it is always sufficient to choose $s = n$. However, although the numerical effort does not increase exponentially in $s$, a smaller choice for $s$ still enhances the numerical performance. Lastly, the number of neurons in each sublayer $M$ determines the accuracy to which the functions $V_j$ can be represented and thus determines the accuracy of our computed clf.

An important feature of this network architecture is the fact that the decomposition of the state vector $x$ into the vectors $z_j$, $1 \leq j \leq s$, is determined by the first hidden layer. Thus, the detection of a suitable splitting of the state space (see Definition 3) is part of the training process. This means that the numerical algorithm presented in this section does not need to know the splitting or coordinate transformation discussed in Section 4. Rather, this structure will be "learned" by the network in the training process. It is possible to incorporate the linear transformation computed by the first hidden layer in Figure 1 into the second hidden layer, that is, to merge the two hidden layers into one hidden layer. However, in our numerical test cases, the NN with two hidden layers as depicted in Figure 1 frequently demonstrated an improved numerical performance.

### 5.2 Basic Training Algorithm

Our goal is that the NN $x \mapsto W(x;\theta)$ approximately is a clf for the system (1). In order to achieve that, we need to train the parameters $\theta$ of our NN accordingly. To this end, we define a loss-function $L$ that penalizes the violation of any of the three inequalities defining a smooth clf in Definition 1. For any point $x \in K$ we set

$$\begin{aligned} &L(x, W(x;\theta), DW(x;\theta)) := \\ &\quad \left([W(x;\theta) - \alpha_1(\|x\|)]_-\right)^2 + \left([W(x;\theta) - \alpha_2(\|x\|)]_+\right)^2 \\ &\quad + \eta\Big(\Big[\alpha_3(\|x\|) + \inf_{u\in U} DW(x;\theta)f(x,u)\Big]_+\Big)^2, \end{aligned} \tag{15}$$

where $\alpha_1, \alpha_2 \in \mathcal{K}_\infty$, $\alpha_3 \in \mathcal{K}$, $[\cdot]_+ := \max(\cdot, 0)$, $[\cdot]_- := \min(\cdot, 0)$, and $\eta > 0$ is a weighting factor. Note that the functions $\alpha_1, \alpha_2$, and $\alpha_3$, as well as the parameter $\eta$ are hyperparameters of the algorithm. In particular, the choice of the lower bound $\alpha_1$ and the upper bound $\alpha_2$ play a crucial role. Suppose we know that some separable clf $V$ exists and fix some $\alpha_1 \in \mathcal{K}_\infty$. Then (possibly after rescaling $V$) there always exists $\alpha_2 \in \mathcal{K}_\infty$ such that $\alpha_1(\|x\|) \leq V(x) \leq \alpha_2(\|x\|)$ for all $x \in K$. Thus, the hyperparameter $\alpha_2$ only needs to be chosen large enough. However, a smaller choice of $\alpha_2$ leads to more significance of the second term in (15), thus enhancing the training process. In conclusion, finding a good choice of $\alpha_1$ and $\alpha_2$ may involve some example-dependent trial and error.

Note that $L$ depends on the point $x$, the evaluation $W(x;\theta)$ and the orbital derivative $DW(x;\theta)f(x,u)$. We calculate this orbital derivative alongside the evaluation of $W(x;\theta)$ via automatic differentiation. Moreover, we need to evaluate the expression $\inf_{u\in U} DW(x;\theta)f(x,u)$. This expression can be simplified for systems with $U = [-C, C]^m$ for some $C > 0$ and an affine linear control input of the form

$$\dot{x} = f(x,u) = h(x) + g(x)u,$$

since then we have

$$\begin{aligned} &\inf_{u\in U} DW(x;\theta)f(x,u) \\ =&DW(x;\theta)h(x) - C\|DW(x;\theta)g(x)\|_1, \end{aligned} \tag{16}$$

cf. [15, Lemma 6]. The training process of the NN is then performed by minimizing the value of the loss function (15) over some finite set of training data $\mathcal{D}_T \subset K$ using a stochastic gradient descent algorithm. The training process is stopped if

$$\|L(\cdot, W(\cdot,\theta), DW(\cdot,\theta))\|_{\infty,\mathcal{D}_V} \leq \varepsilon \tag{17}$$

for a fixed $\varepsilon > 0$ and a finite set of validation data $\mathcal{D}_V \subset K$ that is chosen independently of the training data $\mathcal{D}_T$.

**Remark 17** *Clfs can be characterized as solutions of Zubov's equation [5], i.e., as optimal value functions for suitable optimal control problems. For such problems, supervised learning approaches have been proposed in the literature, see e.g. [2,30]. However, they are difficult to apply in our setting, because while we assume that a separable clf exists, we do not know its precise form and thus also not the corresponding optimal control problem.*

### 5.3 Numerical Improvements

We now sketch two ideas to improve the training process of the NN. At first, we consider an adaptive sampling procedure for the training data $\mathcal{D}_T \subset K$. Initially, we choose $\mathcal{D}_T$ randomly and uniformly distributed in $K$. However, due to the characteristics of the control system or for numerical reasons, it might be more difficult to converge towards a clf on some subset of $K$ than on others. In order to react to such issues, we use another set $\widetilde{\mathcal{D}}_V \subset K$ that is chosen independently of $\mathcal{D}_V$ and $\mathcal{D}_T$ and locate the $\kappa_1 \in \mathbb{N}$ points in $\widetilde{\mathcal{D}}_V$ that have the largest loss value after training for a certain amount of epochs. Afterwards, for every such point $\tilde{x} \in \widetilde{\mathcal{D}}_V$, we add a certain amount $\kappa_2 \in \mathbb{N}$ of points that are uniformly chosen in some neighborhood of $\tilde{x}$ to our set of training data $\mathcal{D}_T$. Thus, we increase the focus of our training towards regions in $K$, where the largest errors occur. This idea is based on the works [29,44].

Furthermore, in our numerical tests it has turned out that the most significant error usually lies around the origin. We tackle this by adding the term $W(0;\theta)^2 + \|DW(0;\theta)\|^2$ to the loss-function used for the training of the network, cf. [6]. Note that $W(0;\theta) = 0$ is also enforced by (2a). However, this is only crucial for training points near the origin. Adding $W(0;\theta)^2$ to the loss function forces the used optimization routine to stay at $W(0;\theta) = 0$ during all stages of the training. Analogously, $DW(0;\theta) = 0$ is also implicitly enforced since (2a) requires 0 to be a global minimum of $W(\cdot,\theta)$. However, we could observe a numerical improvement by explicitly adding $W(0;\theta)^2 + \|DW(0;\theta)\|^2$. A different approach to address issues at the origin has been pursued in [12], where a small neighborhood around the origin is excluded from the domain $K$ of the computed clf.

## *5.4 Numerical Examples*

Finally, we illustrate the presented algorithm with two numerical experiments, where the corresponding control systems have a linear-affine control input, respectively. Our computations are conducted with Python 3.10.6 and Tensorflow 2.11.0 (see [1]) on an NVIDIA GeForce RTX 3070 GPU. The optimization has been performed with the ADAM stochastic gradient descent method.
At first, we consider an extension of the control system (10) to a 10-dimensional state space. We set

$$\begin{aligned}\dot{x}_1 &= x_{10} + u,\\ \dot{x}_2 &= x_1 - x_2 + x_1^2,\\ \dot{x}_j &= x_{j-1} - x_j, \qquad 3 \le j \le 10,\end{aligned} \tag{18}$$

and $U = [-5,5]$. We use the NN depicted in Figure 1 with $d = 1$, $M = 128$, $s = 10$ sublayers, and the softplus-function $\sigma(x) = \ln(1+\exp(x))$ as activation to compute a clf for (18) on $K = [-1,1]^{10}$. The training has been pursued choosing $\alpha_1(r) = 0.2r^2$, $\alpha_2(r) = 10r^2$, and $\alpha_3(r) = 0.01r^2$, as well as $|\mathcal{D}_T| = |\mathcal{D}_V| = 5 \cdot 10^5$. It has stopped after 21 epochs (using a batch size of 32) satisfying (17) with $\varepsilon = 10^{-3}$. Figure 3 shows the computed NN output $W(x;\theta)$ as surface plot. Further, the directional derivative $DW(x;\theta)f(x,u^*)$ with

$$u^*(x) = \arg\min DW(x;\theta)f(x,u) \tag{19}$$

is calculated according to (16) and depicted as wireframe plot. Note that the directional derivative is strongly decreasing with increasing values of $|x_1|$ due to the direct influence of $u$ on $x_1$, see (18). This is in contrast to the plot in Figure 4, where the clf and its derivate in direction of the optimally controlled vector field are shown on the $(x_2,x_3)$-plane. Figure 5 depicts the evaluation of $W(x;\theta)$ alongside the trajectory starting at $(1,\cdots,1)^T \in \mathbb{R}^{10}$ and being controlled via $u^*(x)$ as in (19).

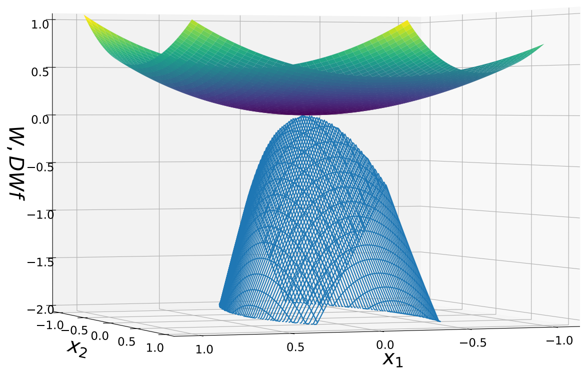


Fig. 3. Approximate clf (solid) and its corresponding orbital derivative (mesh) for the system (18) on the $(x_1,x_2)$-plane.

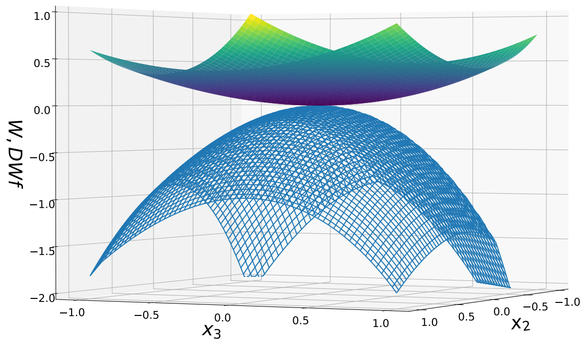


Fig. 4. Approximate clf (solid) and its corresponding orbital derivative (mesh) for the system (18) on the $(x_2,x_3)$-plane.

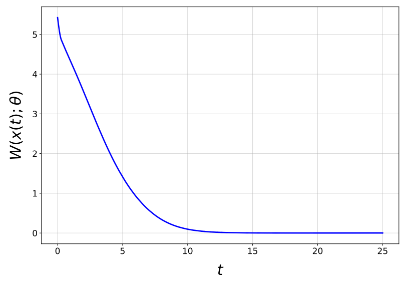


Fig. 5. Evaluation of $W(x;\theta)$ along a trajectory.

Secondly, we consider the 10-dimensional control system

$$\dot{x} = f(x,u) = \begin{pmatrix} -x_1 + 0.5x_2 - 0.1x_9^2 \\ -0.5x_1 - x_2 \\ -x_3 + 0.5x_4 - 0.1x_1^2 \\ -0.5x_3 - x_4 \\ 0.5x_5 + 0.1x_7^2 + u_1 \\ -0.5x_5 - x_6 \\ 0.5x_8 + u_2 \\ -0.5x_7 - x_8 \\ 0.5x_{10} - u_3 \\ 0.1x_2^2 - 0.5u_3 + u_4 \end{pmatrix} \tag{20}$$

with $U = [-10, 10]^4$, based on example (13) provided in [17]. Note that the system is asymptotically controllable, but is unstable for $u \equiv 0$. By defining $z_j = (x_{2j-1}, x_{2j})$, $j \in [5]$, we can decompose (20) into five two-dimensional subsystems that each consist of a linear stabilizable control system, possibly interconnected with other subsystems through non-linearities. To illustrate the ability of our approach to determine the subspaces that lead to separability, we apply a randomly generated invertible matrix $T \in \mathbb{R}^{10\times 10}$ and consider the resulting transformed system $\dot{x} = T^{-1}f(Tx, u)$. We employed the hyperparameters $\alpha_1(r) = 0.05r^2, \alpha_2(r) = 10r^2$, $\alpha_3(r) = 0.01r^2$, as well as $d = 2$, $s = 5$, and $M = 128$ in a training process with $\mathcal{D}_T = 2 \cdot 10^5$ and a batch size of 128. This training process was conducted to compute a clf on the domain $[-1, 1]^{10}$ and reached the tolerance $\varepsilon = 10^{-3}$ after 52 epochs. Figure 6 shows the computed clf and its derivative in direction of the optimally controlled vector field on the $(x_5, x_6)$-plane. Additionally, the lower bound $\alpha_1$ and the upper bound $\alpha_2$ are plotted in orange, respectively. As desired, all four functions touch at the origin. While $W(x; \theta)$ is strictly increasing within the space defined by its lower and upper bounds, the corresponding orbital derivate is strictly negative.

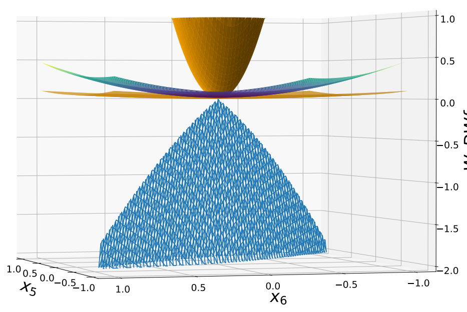


Fig. 6. Plot of the computed clf $W(x; \theta)$, its derivate $DW(x; \theta)f(x, u^*)$, and the bounds $\alpha_1$, $\alpha_2$ on the $(x_5, x_6)$-plane for example (20).

## 6 Conclusion

In this paper, we have discussed the capability of NNs to approximate clfs in high space dimensions. To this end, we have shown that NNs can overcome the curse of dimensionality for approximating (linearly) separable functions and provided conditions for the existence of (linearly) separable clfs. Thus, we have identified control systems that allow for a curse-of-dimensionality-free approximation of clfs with NNs. Moreover, a numerical algorithm was presented and illustrated on two ten-dimensional control systems.

For future research, we aim to investigate the approximation of non-smooth clfs with NNs. A possible basis for this may be recent results on curse-of-dimensionality-free approximations of non-smooth compositional functions with ReLU NNs, see [13], which rely on the approximation result in [43]. Moreover, we aim to explore the usage of noninteracting control via feedback decoupling and other methods from nonlinear control theory to identify situations where a clf (or more general an optimal value function) can be represented by a NN without the curse of dimensionality.

## Acknowledgements

The authors acknowledge funding by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) in the frame of the priority programme SPP 2298 “Theoretical Foundations of Deep Learning” - project number 463912816.

## References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] G. Albi, S. Bicego, and D. Kalise. Gradient-augmented supervised learning of optimal feedback laws using state-dependent Riccati equations. *IEEE Control Syst. Lett.*, 6:836–841, 2022.

[3] C. Beck, A. Jentzen, K. Kleinberg, and T. Kruse. Nonlinear monte carlo methods with polynomial runtime for Bellman equations of discrete time high-dimensional stochastic optimal control problems. *preprint arXiv:2303.03390*, 2023.

[4] P. Braun, L. Grüne, and C. M. Kellett. *(In-)Stability of Differential Inclusions–Notions, Equivalences, and Lyapunov-like Characterizations.* SpringerBriefs in Mathematics. Springer, Cham, 2021.

[5] F. Camilli, L. Grüne, and F. Wirth. Control Lyapunov functions and Zubov's method. *SIAM J. Control Optim.*, 47(1):301–326, 2008.

[6] Y. Chang, N. Roohi, and S. Gao. Neural Lyapunov control. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[7] K. Chen and A. Astolfi. On the active nodes of network systems. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5561–5566. IEEE, 2020.

[8] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2(4):303–314, 1989.

[9] W. Dahmen. Compositional sparsity, approximation classes, and parametric transport equations. *preprint arXiv: 2207.06128*, 2023.

[10] J. Darbon, G. Langlois, and T. Meng. Overcoming the curse of dimensionality for some Hamilton-Jacobi partial differential equations via neural network architectures. *Res. Math. Sci.*, 7, 2020.

[11] S. Dashkovskiy, B. Rüffer, and F. Wirth. Small gain theorems for large scale systems and construction of ISS Lyapunov functions. *SIAM J. Control Optim.*, 48(6):4089–4118, 2010.

[12] N. Gaby, F. Zhang, and X. Ye. Lyapunov-net: A deep neural network architecture for Lyapunov function approximation. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2091–2096. IEEE, 2022.

[13] Q. Gong, W. Kang, and F. Fahroo. Approximation of compositional functions with ReLU neural networks. *Systems Control Lett.*, 175:105508, 2023.

[14] L. Gonon and C. Schwab. Deep ReLU neural networks overcome the curse of dimensionality for partial integrodifferential equations. *Anal. Appl.*, 21(01):1–47, 2023.

[15] L. Grüne and M. Sperl. Examples for separable control Lyapunov functions and their neural network approximation. *IFAC-PapersOnLine*, 56(1):19–24, 2023.

[16] L. Grüne and F. Wirth. Computing control Lyapunov functions via a Zubov type algorithm. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 2129–2134. IEEE, 2000.

[17] L. Grüne. Computing Lyapunov functions using deep neural networks. *J. Comput. Dyn.*, 8(2):131–152, 2021.

[18] J. Han, A. Jentzen, and W. E. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci.*, 115(34):8505–8510, 2018.

[19] M. Hutzenthaler, A. Jentzen, T. Kruse, T. Anh Nguyen, and P. von Wurstemberger. Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. *Proceedings of the Royal Society A*, 476(2244):20190630, 2020.

[20] W. Kang and Q. Gong. Neural network approximations of compositional functions with applications to dynamical systems. *preprint arXiv:2012.01698*, 2020.

[21] W. Kang and Q. Gong. Feedforward neural networks and compositional functions with applications to dynamical systems. *SIAM J. Control Optim.*, 60(2):786–813, 2022.

[22] C. M. Kellett and P. M. Dower. Input-to-state stability, integral input-to-state stability, and $\mathcal{L}_2$-gain properties: Qualitative equivalences and interconnected systems. *IEEE Trans. Automat. Control*, 61(1):3–17, 2015.

[23] H. Khalil. *Nonlinear systems. Third Edition.* Prentice-Hall, Upper Saddle River, NJ, 2002.

[24] S. Khansari-Zadeh and A. Billard. Learning control Lyapunov function to ensure stability of dynamical system-based robot reaching motions. *Robotics and Autonomous Systems*, 62(6):752–765, 2014.

[25] T. Liu, D. Hill, and Z.-P. Jiang. Lyapunov formulation of ISS cyclic-small-gain in continuous-time dynamical networks. *Automatica J. IFAC*, 47(9):2088–2093, 2011.

[26] Y. Long and M. Bayoumi. Feedback stabilization: Control Lyapunov functions modelled by neural networks. In *Proceedings of 32nd IEEE Conference on Decision and Control*, pages 2812–2814. IEEE, 1993.

[27] H. Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural Comput.*, 8(1):164–177, 1996.

[28] S. Na, S. Shin, M. Anitescu, and V. M. Zavala. On the convergence of overlapping schwarz decomposition for nonlinear optimal control. *IEEE Trans. Autom. Control*, 67(11):5996–6011, 2022.

[29] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Adaptive deep learning for high-dimensional Hamilton–Jacobi–Bellman equations. *SIAM J. Sci. Comput.*, 43(2):A1221–A1247, 2021.

[30] T. Nakamura-Zimmerer, Q. Gong, and W. Kang. Neural network optimal feedback control with guaranteed local stability. *IEEE Open Journal of Control Systems*, 1:210–222, 2022.

[31] A. Pinkus. Approximation theory of the mlp model in neural networks. *Acta Numer.*, 8:143–195, 1999.

[32] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *International Journal of Automation and Computing*, 14(5):503–519, 2017.

[33] C. Reisinger and Y. Zhang. Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. *Anal. Appl.*, 18(06):951–999, 2020.

[34] S. Richards, F. Berkenkamp, and A. Krause. The Lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. In *Proceedings of The 2nd Conference on Robot Learning*, Proceedings of Machine Learning Research, pages 466–476. PMLR, 2018.

[35] A. Riekert. Deep neural network approximation of composite functions without the curse of dimensionality. *preprint arXiv:2304.05790*, 2023.

[36] B. S. Rüffer. *Monotone dynamical systems, graphs, and stability of large-scale interconnected systems.* PhD thesis, Universität Bremen, 2007.

[37] R. Sepulchre, M. Janković, and P. Kokotović. *Constructive Nonlinear Control.* Springer, London, 1997.

[38] E. D. Sontag. A Lyapunov-like characterization of asymptotic controllability. *SIAM J. Control Optim.*, 21(3):462–471, 1983.

[39] E. D. Sontag. Feedback stabilization using two-hidden-layer nets. In *1991 American control conference*, pages 815–820. IEEE, 1991.

[40] E. D. Sontag. *Mathematical Control Theory.* Springer New York, 1998.

[41] E. D. Sontag and Y. Wang. On characterizations of the input-to-state stability property. *Systems Control Lett.*, 24(5):351–359, 1995.

[42] M. Sperl, L. Saluzzi, L. Grüne, and D. Kalise. Separable approximations of optimal value functions under a decaying sensitivity assumption. *preprint arXiv:2304.06379*, 2023.

[43] D. Yarotsky. Optimal approximation of continuous functions by very deep ReLU networks. In *Conference on Learning Theory*, pages 639–649. PMLR, 2018.

[44] X. Zhang, J. Long, W. Hu, J. Han, et al. Initial value problem enhanced sampling for closed-loop optimal control design with deep neural networks. *preprint arXiv:2209.04078*, 2022.