Learning Geometrically-Informed Lyapunov Functions with Deep Diffeomorphic RBF Networks

Samuel Tesfazgi
TU Munich

Leonhard Sprandl
TU Munich

Sandra Hirche
TU Munich

Abstract

The practical deployment of learning-based autonomous systems would greatly benefit from tools that flexibly obtain safety guarantees in the form of certificate functions from data. While the geometrical properties of such certificate functions are well understood, synthesizing them using machine learning techniques still remains a challenge. To mitigate this issue, we propose a diffeomorphic function learning framework where prior structural knowledge of the desired output is encoded in the geometry of a simple surrogate function, which is subsequently augmented through an expressive, topology-preserving state-space transformation. Thereby, we achieve an indirect function approximation framework that is guaranteed to remain in the desired hypothesis space. To this end, we introduce a novel approach to construct diffeomorphic maps based on RBF networks, which facilitate precise, local transformations around data. Finally, we demonstrate our approach by learning diffeomorphic Lyapunov functions from real-world data and apply our method to different attractor systems.

1 INTRODUCTION

With recent advances in robotics and machine learning, data-driven autonomous systems are increasingly deployed in safety-critical application scenarios such as autonomous driving (Liu et al., 2024) or robotic rehabilitation (Ai et al., 2023). While learning-based systems are particularly well-suited for such complex and uncertain environments, a limitation that inhibits their deployment is the lack of formal safety and stability guarantees. A practical method to ascertain the desired safety and stability properties of a dynamical system is through the construction of certificate functions, e.g., a Lyapunov function to show convergence to an equilibrium point (Khalil, 2002). A major strength of these approaches is the existence of converse theorems, i.e., if the desired property holds, a certificate function is guaranteed to exist (Teel et al., 2014; Liu, 2022). Besides certification, Lyapunov functions may also be utilized for control synthesis (Sontag, 1989; Tesfazgi et al., 2024) and have additionally been deployed as value function approximators in the context of reinforcement learning (Chow et al., 2018; Chang et al., 2019; Tesfazgi et al., 2021).

In general, certificate functions express the long-term behavior of a system's trajectory through invariant set constraints. Thereby, the set of states to which the system is bounded or converges to, is geometrically encoded in the level sets of the certificate function. A candidate Lyapunov function, for instance, has to be positive-definite with a strictly decreasing time-derivative. While these conditions can be resolved in simple settings, e.g., when the dynamics are known, and the hypothesis space is limited to sum-of-squares polynomials (Parrilo, 2000), no constructive approach is known for general, nonlinear systems. Therefore, the need for expressive learning techniques that construct certificate functions directly from data arises.

Recently, the deployment of neural networks (NNs) has been proposed to learn Lyapunov functions from observations (Richards et al., 2018; Ravanbakhsh and Sankaranarayanan, 2019; Chang et al., 2019; Manek and Kolter, 2019; Dawson et al., 2021). However, even though NNs have the advantage of strong representational capabilities, imposing the necessary constraints is an open issue. Existing methods either induce the Lyapunov conditions via soft-constraints (Chang et al., 2019), only admitting empirical statements, or strictly by extensively searching for counter-examples (Ravanbakhsh and Sankaranarayanan, 2019), which is

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

computationally demanding. A promising perspective is to geometrically constrain the output by using a suitable architecture (Raissi et al., 2019). However, the imposed output constraints are either not specific enough, only guaranteeing positive definiteness (Richards et al., 2018; Dawson et al., 2021), or overly conservative, e.g., using input convex NNs (Amos et al., 2017; Manek and Kolter, 2019).

Contribution. In this work, we follow an alternative approach of encoding structural knowledge and imposing desired geometric properties on the inferred function by deploying smooth and bijective maps, so-called diffeomorphisms (Boumal, 2023). In particular, instead of constraining the output of a function approximator directly, we specify a simple base function with desired geometric properties and subsequently learn a topology-preserving, state-space transformation under which the augmented base function adheres to the data, thereby indirectly obtaining a Lyapunov function. To facilitate the synthesis, we propose a novel diffeomorphism model based on RBF networks in this paper. Our kernel-based architecture enables the generation of precise, local transformations, which help induce constraint satisfaction at the data points. Beyond point attractors, we additionally demonstrate the applicability of our approach for more general system classes, including multiple equilibria and limit cycles.

While the regularity-preserving properties of diffeomorphisms have been used in the context of imitation learning (Rana et al., 2020) and control (Sun et al., 2023), their utilization for learning certificate functions remains understudied. In particular, since the existence of a diffeomorphic map between a linear base system and a nonlinear, data-generating system is generally not guaranteed (Bevanda et al., 2022), the problem of learning diffeomorphic certificate functions directly from data remains highly relevant and warrants separate consideration.

2 PRELIMINARIES

Lyapunov stability theory. Consider an autonomous system[1]

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

with continuous state $\boldsymbol{x} \in \mathbb{R}^n$ and system dynamics $\boldsymbol{f}\colon \mathbb{R}^n \to \mathbb{R}^n$. The problem of certifying stability is concerned with analyzing the behavior of $\boldsymbol{x}(\tau)$ for time $\tau \to \infty$, given some initial state $\boldsymbol{x}(\tau_0) = \boldsymbol{x}_0$. In order to formalize this property, we introduce the following concept of stability.

Definition 2.1 (Khalil (2002))**.** A system (1) has an asymptotically stable equilibrium $\boldsymbol{x}^*$ on the set $\mathcal{X}$ if

1. for all $d > 0$, there exist $\delta > 0$, $\tau_0 \geq 0$ such that $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| < \delta$ implies $\|\boldsymbol{x}(\tau) - \boldsymbol{x}^*\| < d$, $\forall \tau \geq \tau_0$.
2. $\lim_{\tau\to\infty} \|\boldsymbol{x}(\tau) - \boldsymbol{x}^*\| = 0$ for all $\boldsymbol{x}_0 \in \mathcal{X}$.

If the conditions hold for all states, i.e., $\boldsymbol{x}_0 \in \mathbb{R}^n$, the equilibrium $\boldsymbol{x}^*$ is globally asymptotically stable. Without loss of generality, we assume $\boldsymbol{x}^* = \mathbf{0}$ from now on. A practical method to ascertain the convergence property of a system, without solving the underlying dynamics equations, is by means of Lyapunov stability theory.

Theorem 2.2 (Lyapunov Stability Theorem, Khalil (2002))**.** *Let $\boldsymbol{x}^* = \mathbf{0}$ be an equilibrium point for* (1) *and $\mathcal{X} \subset \mathbb{R}^n$ be the domain of $f : \mathcal{X} \mapsto \mathbb{R}^n$ with $\boldsymbol{x}^* \in \mathcal{X}$. Let $V : \mathcal{X} \mapsto \mathbb{R}$ be a continuously differentiable function such that:*

$$V(\mathbf{0}) = 0 \tag{2a}$$

$$V(\boldsymbol{x}) > 0 \quad \forall \boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\} \tag{2b}$$

$$\dot{V}(\boldsymbol{x}) = \nabla_x^\intercal V(\boldsymbol{x}) f(\boldsymbol{x}) < 0 \quad \forall \boldsymbol{x} \in \mathcal{X} \setminus \{\mathbf{0}\} \tag{2c}$$

Then, $\boldsymbol{x}^*$ is locally asymptotically stable in the sense of Definition 2.1.

Thus, finding a function $V(\cdot)$ that satisfies (2a)-(2c) is sufficient to certify stability of $f(\cdot)$.

Multiple Equilibria and Limit Cycles. Beyond asymptotic stability to a single equilibrium, a dynamical system may also exhibit other types of attractor landscapes, such as multiple equilibria, where system trajectories converge to different states out of a set

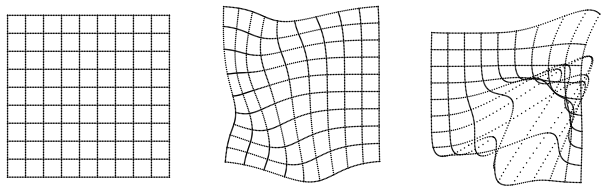
$$\mathcal{X}^* := \{\boldsymbol{x} \in \mathbb{R}^n \mid f(\boldsymbol{x}) = \mathbf{0}\}$$

depending on the initial state $\boldsymbol{x}_0$. Another common attractor are limit cycles, which describe invariant sets $\mathcal{X}^\circ$ under the dynamics $f(\cdot)$ for some orbital period T

$$\mathcal{X}^\circ := \{\boldsymbol{x} | \boldsymbol{x}(\tau) = \boldsymbol{x}(\tau + T),\ f(\boldsymbol{x}) \neq \mathbf{0},\ \forall \tau \geq 0,\ \exists T > 0\}.$$

In order to extend the notion of Lyapunov stability analysis to such systems, it is common to introduce a *Lyapunov-like* function (Patrão, 2011; Björnsson et al., 2015) that satisfies the conditions (2a)-(2c) for the respective sets $\mathcal{X}^*$ or $\mathcal{X}^\circ$, instead of only $\{\mathbf{0}\}$. For notational convenience, we collectively use $\mathcal{X}^{\mathbf{0}}$ for all convergence sets in the following.

[1] *Notation:* Lower and upper case bold symbols denote vectors and matrices, $\mathbb{R}_+ / \mathbb{N}_+$ all real/integer positive numbers, $\boldsymbol{I}_n$ the $n \times n$ identity matrix, $\|\cdot\|$ the Euclidean norm, $|\cdot|$ the absolute value, and $\#(\cdot)$ the cardinality of a set. $\mathcal{L}$ is the set of Lipschitz continuous functions, $\mathcal{D}$ the set of diffeomorphic maps, and p-times continuously differentiable functions are denoted by $\mathcal{C}^p$. The composition of two functions is written $f \circ g = f(g(\cdot))$ and the nested composition of functions is denoted by $\bigcirc_{n=1}^{N} f_n = f_N \circ \cdots \circ f_1$.



(a) State space $\boldsymbol{x}$ (b) Invertible (c) Non-invertible

Figure 1: Transformations of a 2D space (a) by an invertible (b) and a non-invertible map (c). Invertability requires that the map is unique for each point.

Diffeomorphism. A mapping $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ is bijective, if it's inverse $\phi^{-1}(\cdot)$ is guaranteed to exist. If the mapping $\phi(\cdot)$ and its inverse $\phi^{-1}(\cdot)$ are further smooth, it is referred to as a *diffeomorphism*, defined as follows.

Definition 2.3 (Boumal (2023))**.** A diffeomorphism $\phi : U \to V$ with open sets $U, V \subseteq \mathbb{R}^n$ is a bijective and smooth map whose inverse ϕ^{-1} is also smooth.

We denote the set of diffeomorphic maps with $\phi \in \mathcal{D}$. The requirement of $\phi(\cdot)$ being smooth allows the mapping between two *differentiable manifolds*. Since a differentiable manifold is additionally equipped with a differential structure (Lee, 2012), it gives rise to the tangent space required to define gradients, which are necessary for any gradient-based analysis framework, such as Lyapunov stability analysis. Conveniently, diffeomorphic maps preserve the topology of objects, such as functions or differential equations. Intuitively, two sets are *topologically equivalent*, if a mapping between the two can be established, with the map and its inverse being continuous (Lee, 2000). Figure 1 illustratively depicts the difference between a topology-preserving and a non-topology-preserving transformation.

3 DIFFEOMORPHIC LYAPUNOV FUNCTIONS

Directly searching for a Lyapunov function is difficult since constraints (2b) and (2c) need to hold for an uncountable, infinite set of states. To overcome this, we propose to exploit the topological equivalence of Lyapunov functions shown by Grüne et al. (1999) to reformulate the function approximation problem to an optimization over state-space transformations.

3.1 Lyapunov Function Hypothesis Space

The primary challenge in synthesizing a Lyapunov function is guaranteeing that the gradient $\nabla_x V(\cdot)$ fulfills the descent condition (2c). Typically, the dynamics $f(\cdot)$ are not known analytically. Thus, in our considered scenario we only have access to trajectory samples $\mathbb{D} = \{\boldsymbol{x}^{(i)}, \dot{\boldsymbol{x}}^{(i)}\}_{i=1}^N$. However, we may still derive shape-constraints that any potential Lyapunov function candidate has to adhere to locally, since

$$\nabla_x^\intercal V(\boldsymbol{x}) f(\boldsymbol{x}) < 0 \Longrightarrow \nabla_x V(\boldsymbol{x}) \neq \mathbf{0}, \;\; \forall \boldsymbol{x} \in \mathcal{X} \backslash \{\mathcal{X}^{\mathbf{0}}\}. \quad (3)$$

Consequently, a positive definite function $V(\cdot)$ with non-vanishing gradient $\nabla_x V(\boldsymbol{x}) \neq \mathbf{0}$ is always a valid Lyapunov function for *some* stable system. Therefore, a diffeomorphic transformation that preserves these topological properties is guaranteed to generate an output that remains in the space of Lyapunov functions, which we demonstrate in the following:

Proposition 3.1. *Consider a smooth function $V : \mathbb{R}^n \mapsto \mathbb{R}$ and let N_V denote the number of unique gradient roots of $V(\cdot)$*

$$N_V = \#(\mathcal{S}_V), \quad \textit{with} \;\; \mathcal{S}_V = \{\boldsymbol{x} | \nabla_{\boldsymbol{x}} V(\boldsymbol{x}) = \mathbf{0}\} \quad (4)$$

where $\#(\mathcal{S})$ denotes the cardinality of the set $\mathcal{S}$. Next, let $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ be an orientation-preserving diffeomorphism, i.e., its Jacobian $\boldsymbol{J}_\phi \in \mathbb{R}^{n \times n}$ satisfies

$$\det(\boldsymbol{J}_\phi(\boldsymbol{x})) > 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^n. \quad (5)$$

Then, the number of gradient roots is invariant under the map $\phi(\cdot)$ and $N_V = N_U$, where $U := V \circ \phi$.

Proof. Due to (5), $\boldsymbol{J}_\phi$ is full rank $\forall \boldsymbol{x} \in \mathbb{R}^n$, and consequently, the nullspace $\text{null}(\boldsymbol{J}_\phi(\boldsymbol{x}))$ only contains the trivial solution (Strang, 2019). The same holds for the transpose, since $\det(\boldsymbol{J}_\phi(\boldsymbol{x})) = \det(\boldsymbol{J}_\phi(\boldsymbol{x})^\top) > 0$. Thus, the *left* nullspace (Strang, 2019) also only contains the trivial solution. Applying the chain rule

$$\nabla_x U(\boldsymbol{x}) = \frac{\partial}{\partial \boldsymbol{x}} V(\phi(\boldsymbol{x})) = \boldsymbol{J}_\phi(\boldsymbol{x})^\top \nabla_{\boldsymbol{x}} V(\boldsymbol{x}). \quad (6)$$

From (5) and (6), it trivially follows that $\#(\mathcal{S}_V) = \#(\mathcal{S}_U)$, which concludes the proof. □

Thus, it follows that candidate Lyapunov functions are diffeomorphic to one another and consequently that any Lyapunov functions can be transformed into a simple $\mathcal{K}_\infty$ function under a change of coordinates, as proposed in Grüne et al. (1999). For a single point attractor system for instance, the time derivative of a valid Lyapunov function $V(\cdot)$ has to decrease along the trajectories, thereby necessitating non-vanishing gradients outside of the equilibrium (3). Therefore, each contour line of any $V(\cdot)$ is topologically equivalent to a sphere, hence, admitting a diffeomorphic transformation to one another.

3.2 Formulation as Diffeomorphic Learning Problem

We propose to search over the space of Lyapunov functions by finding an appropriate diffeomorphic transformation without the need to explicitly incorporate shape constraints. The data-driven, diffeomorphic Lyapunov learning problem is formalized as follows:

Definition 3.2. Given a dataset $\mathbb{D} = \{\boldsymbol{x}^{(i)}, \dot{\boldsymbol{x}}^{(i)}\}_{i=1}^{N}$ generated by an unknown stable system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and any initial Lyapunov-like function $V_b \colon \mathbb{R}^n \to \mathbb{R}$ with

$$V_b(\boldsymbol{x}) > 0 \ \wedge \ \nabla_x V_b(\boldsymbol{x}) \neq \mathbf{0}, \ \ \forall \boldsymbol{x} \in \mathcal{X} \setminus \{\mathcal{X}^{\mathbf{0}}\}, \tag{7}$$

find a diffeomorphism

$$\phi^* = \underset{\phi \in \mathcal{D}}{\operatorname{argmin}} \, L(V_\phi, \mathbb{D}) \tag{8a}$$

$$\text{s.t.} \ \nabla_{\boldsymbol{x}}^{\intercal} V_\phi(\boldsymbol{x}_t) \dot{\boldsymbol{x}}_t < 0 \quad \forall t \in [1, \dots, T] \tag{8b}$$

where $V_\phi(\boldsymbol{x}) = V_b \circ \phi(\boldsymbol{x})$ with loss function $L(\cdot)$.

Intuitively, (8) reformulates the search for a Lyapunov function, which is a functional optimization problem, as a diffeomorphic optimization problem. First, a simple base function $V_b(\cdot)$ is specified that adheres to the topological properties of a Lyapunov candidate function (7). Then a diffeomorphism is constructed such that the function under the diffeomorphic transformation $V_\phi(\cdot)$ satisfies the Lyapunov conditions on the samples (8b). This is convenient, since the geometric properties of a Lyapunov function are well known, and therefore, the surrogate function can be easily specified, e.g., to $V_b(\boldsymbol{x}) = \boldsymbol{x}^{\intercal}\boldsymbol{x}$ for a single attractor system.

Note: The distinct advantage of encoding geometric knowledge through a base function $V_b(\cdot)$ becomes even more apparent when considering general system classes with different attractor landscapes. Typical function approximation approaches, do not readily extend to more involved attractor landscapes, since the new *Lyapunov-like* function requires different geometric constraints. In contrast, our proposed diffeomorphic learning framework merely requires an appropriate base function $V_b(\cdot)$, that encodes the topology of the desired attractor landscapes.

4 DEEP DIFFEOMORPHIC RBF NETWORK

While the formulation in (8) provides a convenient framework to infer Lyapunov functions through a diffeomorphic reformulation, it still requires optimizing over the space of diffeomorphisms $\mathcal{D}$. In particular, the evaluation of constraint (8b) necessitates an analytical expression of the Jacobian $\boldsymbol{J}_\phi(\cdot)$ to evaluate the gradient $\nabla_{\boldsymbol{x}}^{\intercal} V_\phi(\cdot)$, which some popular invertible modelling frameworks, such as NODEs (Chen et al., 2018) or invertible ResNets (Behrmann et al., 2019) do not provide. While alternative approaches based on partitioned transformations with affine coupling layers (ACLs) (Dinh et al., 2015, 2016; Kingma and Dhariwal, 2018; Rana et al., 2020) admit an analytic expression of the Jacobian, this is at the cost of limited flexibility regarding coordinate-wise transformations.

Table 1: Comparison of diffeomorphism constructions.

	ACL	NODE	i-ResNet	**Our**
Analytic Forward	✓	✗	✓	✓
Analytic Jacobian	✓	✗	✗	✓
Freeform Jacobian	✗	✓	✓	✓
Sup-Universality	✓	✓	**N/A**	✓

To this end, we propose a novel approach to construct diffeomorphisms using layers of bijective RBF maps, which we call **D**eep **D**iffeomorphic **RBF** **N**etwork (DD-RBFN). The principle architecture follows the residual learning paradigm (He et al., 2016), where we use kernel machines to learn a residual component that is added to an identity mapping. By an appropriate choice of activation function, i.e., smooth function with analytical, partial derivative, such as a Gaussian kernel, we are able to derive simple box constraints for the network weights that guarantee invertibility of the learned map. Figure 2 illustrates the working principle of our proposed DD-RBFN and Table 2 compares our method to other invertible network architectures.

4.1 Bijective Residual RBF Layer

Each layer of the DD-RBFN induces a mapping

$$\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{9}$$

$$= \boldsymbol{x} + \begin{bmatrix} \psi_1(\boldsymbol{x}) \\ \vdots \\ \psi_n(\boldsymbol{x}) \end{bmatrix}, \tag{10}$$

with

$$\boldsymbol{\psi}_j(\boldsymbol{x}) = \sum_{i}^{N} w_{i,j} k(\boldsymbol{x}, \boldsymbol{c_i}) \tag{11}$$

where N is the number of neurons in the hidden layer, $\boldsymbol{c}_i \in \mathbb{R}^n$ denotes the center point of the i-th neuron, $\boldsymbol{W} \in \mathbb{W}^{n \times N} \subset \mathbb{R}^{n \times N}$ represents the weight matrix of the output layer with entries $w_{i,j}$, and $\boldsymbol{K} \in \mathbb{R}^{N \times 1}$ is the vector of radial basis functions with entries defined as $K_i(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{c}_i)$. In particular, we deploy multivariate Gaussian kernels as activation functions

$$k(\boldsymbol{x}, \boldsymbol{c}_i) = \exp\Big(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{c}_i)^{\top} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{c}_i) \Big), \tag{12}$$

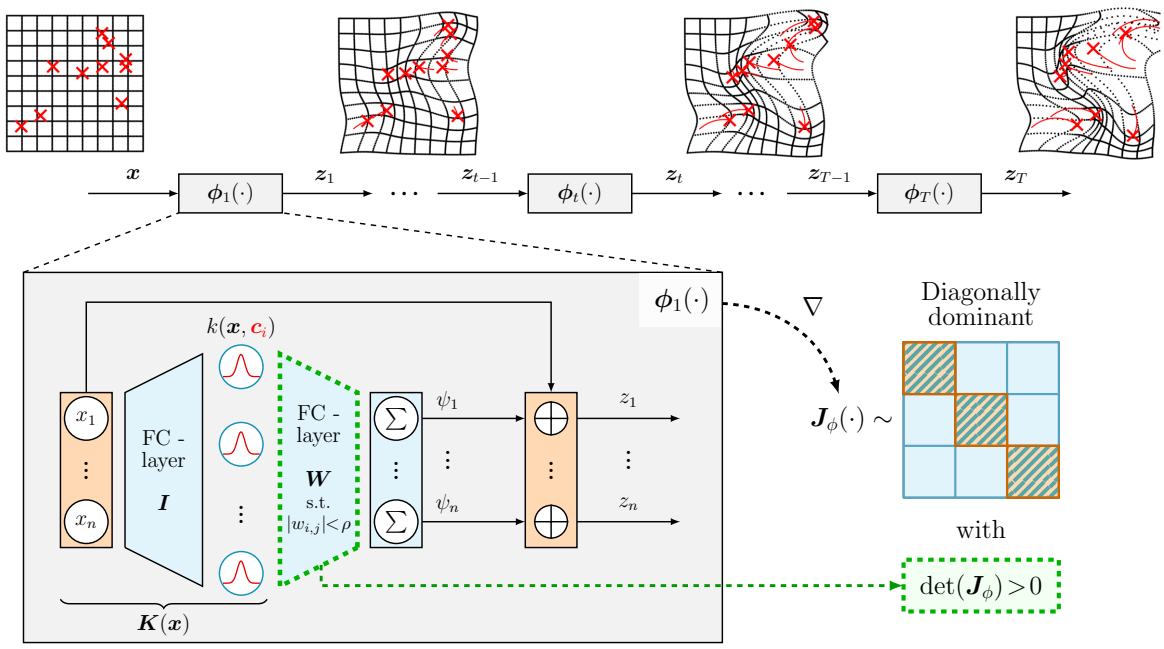


Figure 2: Depiction of the proposed **D**eep **D**iffeomorphic **RBF** **N**etwork (DD-RBFN) architecture. In each layer, a residual RBF mapping is applied which induces a state-space transformation $\boldsymbol{\phi}(\cdot)$ with a diagonally dominant Jacobian $\boldsymbol{J}_\phi(\cdot)$. Through appropriate weight bounds ρ, the bijectivity of the mapping is guaranteed.

where $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ is the positive-definite, symmetric covariance matrix. In order to guarantee bijectivity for the mapping (9), we make use of a finding regarding *determinants of diagonally dominant matrices.*

Theorem 4.1 (Determinant of Diagonally dominant matrices (Ostrowski, 1937))**.** *If $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{E}$ is a real $n \times n$ matrix with the elements of $\boldsymbol{E}$ being bounded in absolute value by $\epsilon \leq \frac{1}{n}$, then a lower bound of the determinant of $\boldsymbol{A}$ is given by:*

$$\det(A) \geq 1 - n\epsilon \tag{13}$$

By exploiting Theorem 4.1, we derive conditions for the weight matrix $\boldsymbol{W}$ under which the invertibility of our mapping (9) is guaranteed.

Theorem 4.2 (Bijective Residual RBF Map)**.** *Consider a mapping $\boldsymbol{\phi}(\cdot)$ as in (9) with multi-variate Gaussian kernel $k(\cdot,\cdot)$ (12) as an activation function. If the entries $w_{i,j}$ in the weight matrix $\boldsymbol{W}$ adhere to the box-constraints*

$$|w_{i,j}| < \frac{1}{nN \sum_{l=1}^{n} |Q_{j,l}| e^{-\frac{1}{2}} \sqrt{D_{j,j}}} := \rho(n, N, \boldsymbol{\Sigma}) \quad \forall i \in [1, \dots, N], j \in [1, \dots, n], \tag{14}$$

where $\boldsymbol{Q}$ and $\boldsymbol{D}$ are due to the eigendecomposition $\boldsymbol{\Sigma}^{-1} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top$, then $\boldsymbol{\phi}(\cdot)$ is a diffeomorphism.

Proof. To enforce bijectivity of the mapping $\boldsymbol{\phi}(\cdot)$, we make use of the residual structure in (9), since it induces a Jacobian $\boldsymbol{J}_\phi(\cdot)$ that is decomposable into an identity matrix $\boldsymbol{I}_n$ and a *disturbance* matrix $\boldsymbol{E} \in \mathbb{R}^{n \times n}$

$$\boldsymbol{J}_\phi(\boldsymbol{x}) = \begin{bmatrix} 1 + \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial \psi_2(\boldsymbol{x})}{\partial x_1} & 1 + \frac{\partial \psi_2(\boldsymbol{x})}{\partial x_2} & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial \psi_n(\boldsymbol{x})}{\partial x_1} & & & 1 + \frac{\partial \psi_n(\boldsymbol{x})}{\partial x_n} \end{bmatrix} := \boldsymbol{I}_n - \boldsymbol{E}(\boldsymbol{x}). \tag{15}$$

From Theorem 4.1 we have that the determinant $\det(\boldsymbol{J}_\phi)$ of such a decomposable matrix can be bounded from below

$$\det(\boldsymbol{J}_\phi(\boldsymbol{x})) \geq 1 - n\epsilon > 0, \tag{16}$$

if the absolute value of the elements of the disturbance matrix $\boldsymbol{E}$ admit a bound ϵ as such:

$$\epsilon = \max_{l,j}(\|E_{l,j}\|) \leq \frac{1}{n}, \quad \forall l \in [1, \dots, n], j \in [1, \dots, n]. \tag{17}$$

Since the elements of $\boldsymbol{E}$ consist of partial derivatives of the residual terms (11), i.e., $E_{l,j} = -\frac{\partial \psi_l(\boldsymbol{x})}{\partial x_j}$, the inequality (17) directly translates to a bound on the weighted sum of partial derivatives of the RBFs

$$\max_{i,j} \left(\sum_{i=1}^{N} |w_{i,j}| \left\| \frac{\partial k(\boldsymbol{x}, \boldsymbol{c_i})}{\partial x_j} \right\| \right) \leq \frac{1}{n}. \tag{18}$$

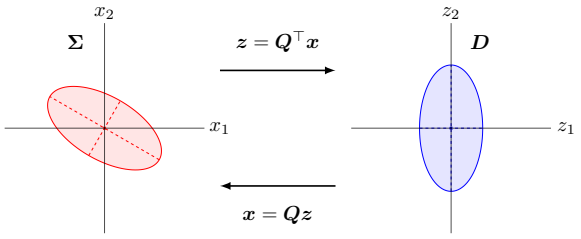


Figure 3: Visualization of bound derivation for Gaussian kernel with symmetric covariance. The kernel with diagonal covariance $\boldsymbol{D}$ is equivalent to k under a state-space transformation. This transformation is given by the matrix $\boldsymbol{Q}$ under which the original $\boldsymbol{\Sigma}$ is axis-aligned with the new coordinate axis $\boldsymbol{z} = \boldsymbol{Q}^\top \boldsymbol{x}$.

For the choice of Gaussian kernel $k(\cdot, \cdot)$ (12) the maximum value of $\|\frac{\partial k}{\partial x_j}\|$ is bounded, can be computed in closed-form, and only depends on the entries of $\boldsymbol{\Sigma}$. Thus, it suffices to appropriately bound the weights $|w_{i,j}|$ to guarantee the positive-definiteness of $\det(\boldsymbol{J}_\phi)$ and thereby the invertibility of the mapping $\boldsymbol{\phi}(\cdot)$. □

Theorem 4.2 provides a bound on the maximum, relative change that the state-space transformation $\boldsymbol{\phi}(\cdot)$ may induce such that bijectivity is ensured. The corresponding weight bound (14) is inversely proportional to the state dimension n, the number of RBF centers N, and the shape of the RBF activation functions encoded by $\boldsymbol{\Sigma}$. Therefore, intuitively the admissible deformation decreases, if there are many basis functions with small bandwidths, since many local changes are more likely to induce a non-bijective mapping. Additionally, the eigendecomposition of $\boldsymbol{\Sigma}^{-1}$ in Theorem 4.2 facilitates bounding the maximum value of $\|\frac{\partial k}{\partial x_j}\|$ for the more general symmetric covariance case, as illustrated in Figure 3.

Note: The derived constraint (14) is conservative, as Theorem 4.2 imposes the same absolute bound on all coefficients instead of considering the weighted sum as shown in (18). Advantageously, this leads to simple box-constraints that only depend on a-priori known quantities facilitating a linear optimization problem. Critically, the bound for each weight $w_{i,j}$ is independent of the other coefficients, thereby admitting an optimization over all weights in a distributed fashion.

4.2 Composition of Network Layers

While the formulation in Theorem 4.2 is convenient from an optimization perspective, it also imposes a limitation on the admissible network weights and thereby restricts the expressivity of each bijective RBFN layer. To overcome this, we exploit a property of invertible maps, i.e., that bijectivity is preserved under composition. Since the composition of smooth functions is trivially smooth, the same holds for diffeomorphisms, which is formalized in the following.

Proposition 4.3. *The composition of T orientation preserving diffeomorphisms*

$$\boldsymbol{\Phi}(\boldsymbol{x}) \coloneqq \bigcirc_{t=1}^{T} \boldsymbol{\phi}_t(\boldsymbol{x}), \quad t \in \mathbb{N}_+, \tag{19}$$

is also an orientation preserving diffeomorphism.

Thus, we propose to construct compositions of bijective RBFN layers to obtain a more expressive mapping resulting in the DD-RBFN architecture:

$$\begin{aligned} \boldsymbol{\Phi}_{\text{DD-RBFN}}(\boldsymbol{x}) \coloneqq \bigcirc_{t=1}^{T} & \underbrace{\boldsymbol{\phi}_{t-1}(\boldsymbol{x}) + \boldsymbol{W}_t \boldsymbol{K}_t(\boldsymbol{\phi}_{t-1}(\boldsymbol{x}))}_{\coloneqq \boldsymbol{\phi}_t(\boldsymbol{x})}, \\ & \text{s.t. } |w_{i,j,t}| \leq \rho_t(n, N_t, \boldsymbol{\Sigma}_t), \end{aligned} \tag{20}$$

with $\boldsymbol{\phi}_0(\cdot) = \boldsymbol{I}$ and where t denotes the iterator variable over the layers. For each $\boldsymbol{\phi}_t(\cdot)$, the number of center points N_t, the center positions $\boldsymbol{c}_{t,i}$, and covariance matrix $\boldsymbol{\Sigma}_t$ can be chosen freely. Since the coefficient bounds $\rho_t(\cdot)$ in (14) are independent of the center positions, recomputing them is only necessary if N_t or $\boldsymbol{\Sigma}_t$ changes. Figure 2 (top) illustratively depicts the DD-RBFN architecture, where we denote the output of the composition at the t-th layer by $\boldsymbol{z}_t \in \mathbb{R}^n$, e.g., $\boldsymbol{z}_0 = \boldsymbol{x}$ and $\boldsymbol{z}_2 = \boldsymbol{\phi}_2(\boldsymbol{\phi}_1(\boldsymbol{x}))$, for improved readability.

The composite map (20) can also be interpreted as the evolution of a time-varying, discrete-time system

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + \boldsymbol{W}_t \boldsymbol{K}_t(\boldsymbol{z}_t), \quad \text{s.t. } |w_{i,j,t}| \leq \rho_t(n, N_t, \boldsymbol{\Sigma}_t), \tag{21}$$

where the dynamics of the difference equation (21) are driven by the weight matrix $\boldsymbol{W}_t$ and the gram matrix $\boldsymbol{K}_t(\cdot)$. While some approaches exploit the existence and uniqueness of solutions to Lipschitz continuous ordinary differential equations (Khalil, 2002; Chen et al., 2018) to construct diffeomorphims, this property does not hold for discrete-time difference equations in general. Hence, intuitively the weight constraints ρ_t induce a Lipschitz bound such that the difference equation (21) has a unique solution everywhere.

4.3 Flow Endpoints and Sup-Universality

In this section, we analyze the approximation capabilities of DD-RBFNs and demonstrate that our proposed architecture is a sup-universal approximator for $\mathcal{C}^2$ diffeomorphisms.

A well-known result in the literature is that sufficiently wide, one-layer RBFNs are universal approximators on

a compact subset of $\mathbb{R}^n$, under mild conditions (Park and Sandberg, 1991). Similarly, linear combinations of universal kernels, such as Gaussian kernels (12), are known to approximate continuous functions arbitrarily well (Micchelli et al., 2006). This property extends to the residual-based mapping $\boldsymbol{\phi}(\cdot)$ in (9).

Lemma 4.4. *Let $\phi : \mathbb{R}^n \to \mathbb{R}^n$ be defined as*

$$\phi(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{22}$$

with $\boldsymbol{W} \in \mathbb{R}^{n \times N}$ and denote by $\mathcal{M}$ the set of all possible ϕ as defined in (22). Then, $\mathcal{M}$ is a sup*-universal approximator for Lipschitz continuous maps $\mathcal{L}(\mathbb{R}^n)$.*

While Lemma 4.4 establishes the principle expressiveness of a kernel-based residual mapping, it does not readily extend to the bijective formulation in Theorem 4.2 due to the weight constraints (14). More specifically, the mapping is Lipschitz bounded with

$$L \propto \sum_{i=1}^{N} |w_i| \left\| \frac{\partial k(\boldsymbol{x}, \boldsymbol{x}_i)}{\partial x_j} \right\|. \tag{23}$$

Since the partial derivative of a Gaussian kernel is bounded, the box constraints on w_i directly translate into a maximum value of L. In (Teshima et al., 2020), it is established that any $\mathcal{C}^2$-diffeomorphism can be approximated over a compact set by a composition of flows generated by Lipschitz continuous dynamical system $f \in \mathcal{L}(\mathbb{R}^n)$. This property extends to our DD-RBFN architecture, despite the Lipschitz bound (23).

Theorem 4.5 (sup-universality of DD-RBFNs)**.** *Let $\phi_t \in \mathcal{M}_\rho$ define the set of all functions that satisfy Theorem 4.2. Then, the composite model class*

$$\mathcal{M}_{DD\text{-}RBFN} = \left\{ \bigcirc_{t=1}^{T} \phi_t | \phi_t \in \mathcal{M}_\rho, k \in \mathbb{N}_+ \right\} \tag{24}$$

is a sup*-universal approximator for $\mathcal{C}^2$ diffeomorphisms.*

Proof. Exploiting the analogous interpretation of the DD-RBFN architecture as a discrete-time system, the composite mapping (21) is equilivant to an Euler discretization of a continuous-time ODE

$$\boldsymbol{z}_{t+1} = \boldsymbol{z}_t + \delta \boldsymbol{W}_t \boldsymbol{K}_t(\boldsymbol{z}_t), \tag{25}$$

with newly introduced virtual time-step $\delta \in (0, 1]$. Due to δ, the admissible weight bound is adjusted to

$$\rho_\delta(n, N, \boldsymbol{\Sigma}, \delta) = \frac{1}{|\delta|} \rho(n, N, \boldsymbol{\Sigma}), \tag{26}$$

where ρ_δ is the virtual time-step dependent weight bound. Since the coefficient bound ρ_δ becomes arbitrary large in the limit $\delta \to 0$, the instantaneous integration step of any $f \in \mathcal{L}(\mathbb{R}^n)$ can be approximated given a sufficiently small virtual time-step δ. Thus, Lemma 4.4 applies for each layer $\phi_t \in \mathcal{M}_\rho$ in (24). Consequently, a sufficiently long composition $\mathcal{M}_{\text{DD-RBFN}}$ can approximate arbitrary flows of Lipschitz continuous ODEs. □

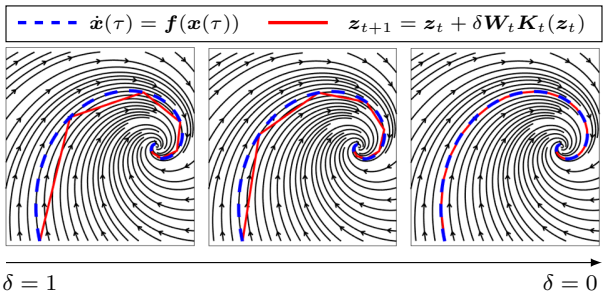


Figure 4: Illustration of virtual time-step δ and resulting implication on the admitted flow-endpoints.

Algorithm 1 MPC-based DD-RBFN optimization

Require: $V_b(\boldsymbol{x})$, $\mathbb{D}$, $\boldsymbol{\Sigma}$, H
for $t = 0, \ldots,$ `#Iterations` **do**
  $\{\boldsymbol{W}_{t+1}, \ldots, \boldsymbol{W}_{t+H}\} \leftarrow$ `solve` (27)
  $\boldsymbol{\Phi} \leftarrow \boldsymbol{\phi}_{t+1} \circ \boldsymbol{\phi}_t$ ▷ `Update` $\boldsymbol{\Phi}$ `with` $\boldsymbol{W}_{t+1}$ only
  $V_{\boldsymbol{\Phi}} \leftarrow V_b \circ \boldsymbol{\Phi}$ ▷ `Update Lyapunov function`
  `discard` $\{\boldsymbol{W}_{t+2}, \ldots, \boldsymbol{W}_{t+H}\}$
end for
return $\boldsymbol{\Phi}$, $V_{\boldsymbol{\Phi}}$

The remainder of the proof is based on the fact that locally supported diffeomorphisms can be described by a finite composition of flow endpoints generated by ODEs integrated over unit time (Teshima et al., 2020; Ishikawa et al., 2023). The implication of considering a virtual time-step δ in (25) is illustratively depicted in Figure 4. Note that the required network depth to reach a flow endpoint increases with decreasing δ, since each integration step is realized by a single layer.

4.4 Learning Diffeomorphic Lyapunov Functions from Data

To finally obtain a Lyapunov function from data, we solve the diffeomorphic optimization problem (8) using our proposed architecture. To this end, we exploit the analogous interpretation of DD-RBFN as a time-discrete system to solve the diffeomorphic optimization problem using tools from optimal control theory.

Specifically, each network layer t corresponds to a time-step, the output at each layer $\boldsymbol{z}_t$ represents the system's state and the network weights $\boldsymbol{W}_t$ correspond to control inputs. Together with the weight constraints ρ_t, it results in an input-constrained system. We deploy *model predictive control* (MPC), which is a com-

mon method to solve nonlinear, non-convex problems by iteratively optimizing the control inputs over a prediction horizon H (Grüne and Pannek, 2017; Kamthe and Deisenroth, 2018). Since MPC facilitates input constraint handling, it is convenient to deploy here. Hence, at each layer, we optimize over the weights

$$\boldsymbol{W}_{t+1} = \underset{\boldsymbol{W}_{t+1},\dots,\boldsymbol{W}_{t+H}}{\operatorname{argmin}} \sum_{h=1}^{H}\sum_{i=1}^{N} \nabla_{\boldsymbol{x}}^{\intercal} V_\Phi(\boldsymbol{z}_{t+h}^{(i)})\dot{\boldsymbol{x}}^{(i)} \quad (27a)$$

$$\text{s.t.} \quad \boldsymbol{z}_{t+1} = \boldsymbol{z}_t + \boldsymbol{W}_t \boldsymbol{K}_t(\boldsymbol{z}_t) \quad (27b)$$

$$|w_{i,j,t+h}| < \rho_{t+h} \quad \forall h \in [1,\dots,H] \quad (27c)$$

$$\boldsymbol{z}_0^{(i)} = \boldsymbol{x}^{(i)} \quad (27d)$$

where (27b) represents the dynamics and (27c) the input constraints. The resulting $\boldsymbol{W}_{t+1}$ is used to iteratively append the DD-RBFN and by minimizing the empirical Lyapunov risk (27a) (Chang et al., 2019), the diffeomorphic optimization is guided in a manner that induces constraint satisfaction on the samples.

5 EVALUATION

To evaluate our proposed approach, we first apply our method to different attractor landscapes, including point attractors, two-attractor systems, and limit cycles. Second, we demonstrate the expressiveness of our approach in a comparison evaluation with related works. An implementation of Algorithm 1 is available at `https://github.com/stesfazgi/Diffeomorphic-Lyapunov-Functions`

5.1 Attractor Landscapes

Single Equilibrium. For this evaluation, we use the LASA handwriting dataset (Khansari-Zadeh and Billard, 2011), which is a popular benchmark in the learning stable dynamical system literature (Perrin and Schlehuber-Caissier, 2016; Rana et al., 2020; Zhang et al., 2023). Figure 5 shows the result of applying the diffeomorphic learning approach to 6 exemplary shapes with a base function $V_b(\boldsymbol{x}) = 0.1\boldsymbol{x}^\top\boldsymbol{x}$. Our proposed approach successfully learns a diffeomorphism such that the transformed function $V_\Phi(\cdot)$ constitutes a valid Lyapunov function for the trajectories.

Two Attractor System. Additionally, we deploy the approach on a dynamical system with two stable equilibria and one unstable equilibrium as depicted by the vector field in Figure 6 (left). For the training data, the system is initialized at 6 different positions and simulated to convergence. The obtained $V_\Phi(\cdot)$ is depicted in Figure 6 (right). The learned diffeomorphic Lyapunov function is consistent with all demonstrations included in the dataset and successfully identified the position of the two stable equilibria.

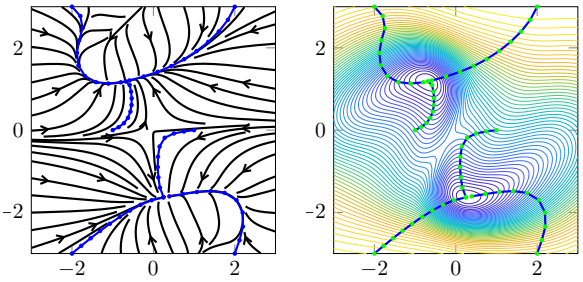


Figure 6: Two equilibria system. **Left:** Vector field of the system (black) and data samples (blue). **Right:** Learned diffeomorphic Lyapunov-like function.

Limit Cycle System. Finally, we use the proposed approach to find a Lyapunov-like function for a limit cycle system. To this end, we consider the well-known Van der Pol oscillator (van der Pol, 1926) depicted in Figure 7 (left) with the stable limit cycle highlighted in red. For training, we sample 20 approximately equally spaced data points along the limit cycle and simulate 4 trajectories starting in the corners of the state space. The sampled data and the resulting diffeomorphic function $V_\Phi(\cdot)$ are shown in Figure 7 (right). It can be seen that the zero gradient contour line of $V_\Phi(\cdot)$ (marked in red) aligns well with the data sampled along the limit cycle (blue dots). Additionally, the Lyapunov constraints are fulfilled along the trajectories converging towards the limit cycle.

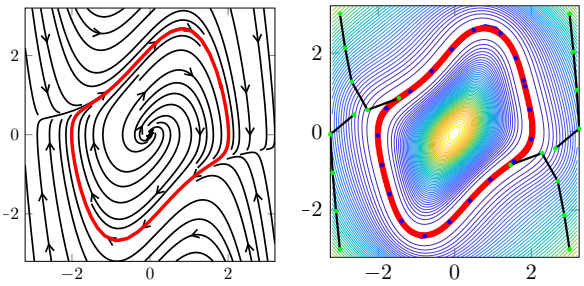


Figure 7: **Left:** Vector field of system (black) with limit cycle (red). **Right:** Contour lines of learned diffeomorphic Lyapunov-like function with sampled data.

5.2 Comparison Evaluation

For the comparison evaluation, we again consider the single-equilibrium case using the LASA dataset. In the dataset, each shape consists of a total of 7 demonstrations with 1000 data points each. We train the methods on a subset of the data of one demonstration and evaluate the generalization capabilities of the learned Lyapunov function on the remaining 6 demonstrations by computing a violation rate, i.e., the percentage of data points on which the Lyapunov conditions are violated. We repeat this process for 5 different seeds.

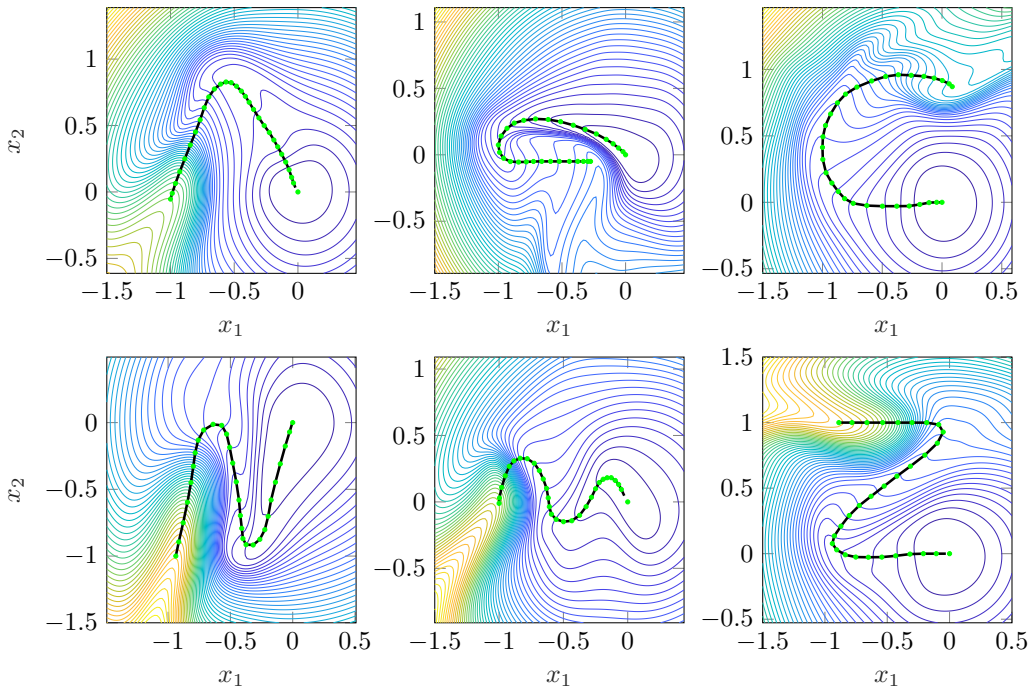


Figure 5: Successfully learned diffeomorphic Lyapunov functions for 6 exemplary shapes of the LASA dataset with data trajectories (black) and samples on which the Lyapunov conditions are satisfied (green).

Table 2: Resulting mean and standard deviation of the violation rate, i.e., the percentage of data points on which the Lyapunov conditions are violated (smaller=better), over 6 shapes of the LASA handwriting dataset.

	MLP	PD-MLP	ACL	i-ResNet	NODE	**DD-RBFN (Our)**
Angle	2.14 ± 0.86	2.25 ± 1.86	2.80 ± 1.61	9.74 ± 9.19	2.58 ± 1.82	$\mathbf{2.00 \pm 0.71}$
C-Shape	7.29 ± 5.35	3.20 ± 0.26	5.58 ± 2.13	4.06 ± 0.41	2.48 ± 0.45	$\mathbf{2.45 \pm 0.86}$
Z-Shape	9.95 ± 7.01	8.44 ± 2.14	6.65 ± 3.92	3.43 ± 4.03	7.02 ± 3.89	$\mathbf{3.26 \pm 2.41}$
N-Shape	15.05 ± 6.52	11.69 ± 3.12	12.45 ± 2.79	23.79 ± 1.92	14.69 ± 4.76	$\mathbf{10.89 \pm 1.30}$
Sine	11.52 ± 8.54	4.55 ± 3.10	2.84 ± 1.18	13.95 ± 11.62	4.18 ± 3.52	$\mathbf{1.41 \pm 0.50}$
Bended line	33.03 ± 11.71	15.66 ± 2.07	$\mathbf{7.78 \pm 7.51}$	38.85 ± 2.70	20.02 ± 15.14	8.44 ± 4.38

For the comparison, we use other diffeomorphism constructions, i.e., ACL (Dinh et al., 2016), i-ResNets (Behrmann et al., 2019), and NODEs (Chen et al., 2018). Moreover, we include two MLP-based Lyapunov function approximators, similar to the learner deployed in (Chang et al., 2019) and (Dawson et al., 2021). In (Dawson et al., 2021), the authors guarantee the positive definiteness of the output by taking the inner product of the last hidden layer. Therefore, we call the architecture PD-MLP in the following.

Table 2 reports the obtained results for the 6 shapes shown in Figure 5. With the exception of the Bended line, the PD-MLP architecture outperforms the naive MLP, which indicates the benefit of prior structural knowledge. Moreover, the generative approaches show a similar performance as the PD-MLP, which indicates the principal expressivity of the indirect, diffeomorphic Lyapunov learning framework. Within the diffeomorphic methods, our DD-RBFN-based approach achieves the best performance for all of the shapes except the Bended line, for which ACL has a slightly lower violation rate.

6 Conclusion

In this work, we present a diffeomorphic learning framework to infer Lyapunov functions from data. By designing simple base functions and optimizing over topology-preserving maps, we successfully encode prior geometrical knowledge during inference. Furthermore, we propose a novel, diffeomorphic modeling approach based on RBFs to facilitate precise, data-driven transformations. Finally. we evaluate our method on systems with different attractor landscapes, including multiple equilibria and limit cycles, and in a comparison evaluation with related works.

Acknowledgements

This work was supported by the European Research Council (ERC) Consolidator grant "CO-MAN" under grant agreement no. 864686.

References

Qingsong Ai, Zemin Liu, Wei Meng, Quan Liu, and Sheng Q. Xie. Machine learning in robot-assisted upper limb rehabilitation: A focused review. *IEEE Transactions on Cognitive and Developmental Systems*, 15(4):2053–2063, 2023.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 146–155, 2017.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *Proceedings of the International Conference on Machine Learning*, pages 573–582, 2019.

Petar Bevanda, Max Beier, Sebastian Kerz, Armin Lederer, Stefan Sosnowski, and Sandra Hirche. Diffeomorphically learning stable koopman operators. *IEEE Control Systems Letters*, 6:3427–3432, 2022.

Jóhann Björnsson, Peter Giesl, Sigurdur F Hafstein, and Christopher M Kellett. Computation of lyapunov functions for systems with multiple local attractors. *Discrete and Continuous Dynamical Systems*, 35, 2015.

Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. In *Advances in Neural Information Processing Systems*, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. In *Advances in Neural Information Processing Systems*, page 8103–8112, 2018.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *5th Annual Conference on Robot Learning*, 2021.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *International Conference on Learning Representations, Workshop Track Proceedings*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2016.

Lars Grüne and Jürgen Pannek. *Nonlinear model predictive control*. Springer, 2017.

Lars Grüne, Eduardo D Sontag, and Fabian R Wirth. Asymptotic stability equals exponential stability, and iss equals finite energy gain—if you twist your eyes. *Systems & Control Letters*, 38(2):127–134, 1999.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Isao Ishikawa, Takeshi Teshima, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Universal approximation property of invertible neural networks. *Journal of Machine Learning Research*, 24(287):1–68, 2023.

Sanket Kamthe and Marc P. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2018.

Hassan K Khalil. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition, 2002.

Seyed Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

John M. Lee. *Introduction to Topological Manifolds*, volume 2. Springer, 2000.

John M Lee. *Smooth manifolds*. Springer, 2012.

Haochen Liu, Zhiyu Huang, Xiaoyu Mo, and Chen Lv. Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2024.

Jun Liu. Converse barrier functions via lyapunov functions. *IEEE Transactions on Automatic Control*, 67 (1):497–503, 2022.

Gaurav Manek and J. Zico Kolter. Learning stable deep dynamics models. In *Advances in Neural Information Processing Systems*, 2019.

Charles A. Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(95):2651–2667, 2006.

Alexander Ostrowski. Über die determinanten mit überwiegender hauptdiagonale. *Commentarii mathematici Helvetici*, 10:69–96, 1937.

J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257, 1991.

Pablo Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

Mauro Patrão. Existence of complete lyapunov functions for semiflows on separable metric spaces. *Far East Journal of Dynamical Systems*, 2011.

Nicolas Perrin and Philipp Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59, 2016.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 2019.

Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In *Proceedings of the Conference on Learning for Dynamics and Control*, volume 120, pages 630–639, 2020.

Hadi Ravanbakhsh and Sriram Sankaranarayanan. Learning control lyapunov functions from counterexamples and demonstrations. *Autonomous Robots*, 2019.

Spencer M. Richards, Felix Berkenkamp, and Andreas Krause. The Lyapunov Neural Network: Adaptive Stability Certification for Safe Learning of Dynamical Systems. In *Conference on Robot Learning*, 2018.

Eduardo D. Sontag. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989.

Gilbert Strang. *Linear algebra and learning from data*. Wellesley-Cambridge Press, Philadelphia, PA, 2019.

Qinqin Sun, Xiuye Wang, Guolai Yang, Ye-Hwa Chen, and Fai Ma. Adaptive robust formation control of connected and autonomous vehicle swarm system based on constraint following. *IEEE Transactions on Cybernetics*, 53(7):4189–4203, 2023.

Andrew R. Teel, João P. Hespanha, and Anantharaman Subbaraman. A converse lyapunov theorem and robustness for asymptotic stability in probability. *IEEE Transactions on Automatic Control*, 59 (9):2426–2441, 2014.

Samuel Tesfazgi, Armin Lederer, and Sandra Hirche. Inverse reinforcement learning: A control lyapunov approach. In *IEEE Conference on Decision and Control (CDC)*, pages 3627–3632, 2021.

Samuel Tesfazgi, Leonhard Sprandl, Armin Lederer, and Sandra Hirche. Stable inverse reinforcement learning: Policies from control lyapunov landscapes. *IEEE Open Journal of Control Systems*, 3:358–374, 2024.

Takeshi Teshima, Koichi Tojo, Masahiro Ikeda, Isao Ishikawa, and Kenta Oono. Universal approximation property of neural ordinary differential equations. In *Advances in Neural Information Processing Systems, Workshop on Differential Geometry meets Deep Learning*, 2020.

Balth. van der Pol. On "relaxation-oscillations". *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):978–992, 1926.

Jiechao Zhang, Hadi Beik-Mohammadi, and Leonel Rozo. Learning riemannian stable dynamical systems via diffeomorphisms. In *Conference on Robot Learning*, page 1211 – 1221, 2023.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes. Analyzed approximation capabilities.]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes.]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes.]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Yes]

   (b) The license information of the assets, if applicable. [Not Applicable]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

   (d) Information about consent from data providers/curators. [Not Applicable]

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Learning Geometrically-Informed Lyapunov Functions with Deep Diffeomorphic RBF Networks: Supplementary Materials

1 EXTENDED RELATED WORK AND BACKGROUND

1.1 Lyapunov Function Approximation

For the extended related work, we consider data-driven identification of Lyapunov functions of the following form: Given a set of N samples $\mathbb{D} = \{\boldsymbol{x}^{(i)}, \dot{\boldsymbol{x}}^{(i)}\}_{i=1}^{N}$ generated by an unknown, asymptotically stable system (in the sense of Definition 2.1), find a function $V : \mathbb{R}^n \mapsto \mathbb{R}$ such that

$$V^* = \min_V q(V) \tag{1a}$$

$$\text{s.t.} \quad V(\boldsymbol{x}) > 0 \qquad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\} \tag{1b}$$

$$\nabla_{\boldsymbol{x}} V(\boldsymbol{x})^\top f(\boldsymbol{x}) < 0 \quad \forall \boldsymbol{x}^{(i)}, \dot{\boldsymbol{x}}^{(i)} \in \mathbb{D} \tag{1c}$$

$$\nabla_{\boldsymbol{x}} V(\boldsymbol{x}) \neq \boldsymbol{0} \qquad \forall \boldsymbol{x} \in \mathbb{R}^n \setminus \{\boldsymbol{0}\}, \tag{1d}$$

where q is some user-defined cost function. However, solving the functional optimization problem (1) is intractable without the use of function approximators. Moreover, the problem is challenging, since the constraints (1b) and (1d) need to hold over an uncountable, infinite set of states. Thus, a well-suited Lyapunov function hypotheses class requires a tractable parameterization of these shape constraints. Given this background, we present a short review of different Lyapunov function approximation approaches in the following.

1.1.1 Parametric Models

Choosing the hypothesis class among the parametric models offers the advantage of exploiting the structure provided by the function class itself to tackle problematic shape-constraints.

Sum-of-Squares (*SOS*) programming (Parrilo, 2000) for instance exploits the polynomial structure to provide globally valid constraints. The main idea is to parameterize the Lyapunov candidate function by a sum of squares polynomial, which is non-negative by construction. The class of SOS-polynomials for a fixed degree $2d$ can be defined as:

$$V_{\text{SOS}}(\boldsymbol{x}) = \boldsymbol{m}(\boldsymbol{x})^\top \boldsymbol{Q} \boldsymbol{m}(\boldsymbol{x}), \quad \boldsymbol{Q} \succeq 0 \tag{2}$$

where $\boldsymbol{m}(\boldsymbol{x})$ is a vector of monomials up to degree d and $\boldsymbol{Q}$ is a positive semi-definite matrix. In Parrilo (2000), it is shown that the requirement of a polynomial to be decomposable into a sum of squares can be enforced by the linear matrix inequality (*LMI*) $\boldsymbol{Q} \succeq 0$, turning the task of finding a Lyapunov function into a convex optimization problem that can be solved efficiently (Lofberg, 2009). However, besides restricting the Lyapunov candidate functions to the space of SOS polynomials, the approach is only admissible for polynomial dynamics (Papachristodoulou and Prajna, 2002). Furthermore, it is shown by Parrilo (2001) that not every positive semi-definite polynomials admit an SOS decomposition.

Weighted sum of asymmetric quadratic functions (WSAQF) (Khansari-Zadeh and Billard, 2014) are another parametric approach to learn Lyapunov functions

$$V_{\text{WSAQF}}(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{P}_0 (\boldsymbol{x} - \boldsymbol{x}^*) + \sum_{i=1}^{C} \beta_i(\boldsymbol{x}) \big((\boldsymbol{x} - \boldsymbol{x}^*) \boldsymbol{P}_i (\boldsymbol{x} - \boldsymbol{\mu}_i - \boldsymbol{x}^*) \big)^2 \tag{3}$$

with

$$\beta_i(\boldsymbol{x}) = \begin{cases} 1, & \forall \boldsymbol{x} : (\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{P}_i (\boldsymbol{x} - \boldsymbol{\mu}_i - \boldsymbol{x}^*) \geq 0 \\ 0, & \forall \boldsymbol{x} : (\boldsymbol{x} - \boldsymbol{x}^*)^\top \boldsymbol{P}_i (\boldsymbol{x} - \boldsymbol{\mu}_i - \boldsymbol{x}^*) < 0 \end{cases} \tag{4}$$

where $\boldsymbol{P}_i \succ 0$, $\boldsymbol{\mu}_i \in \mathbb{R}^n$ and $C \in \mathbb{N}_+$ is the number of asymmetric components. Due to their construction, they satisfy constraints (1b) and (1d) at the cost of being only $\mathcal{C}_1$. However, since WSAQFs are sums of asymmetric quadratic functions, their expressiveness is limited and due to (4), their evaluation requires a case distinction. In addition, the parameters $\boldsymbol{P}_i$ require to fulfill a *strict* matrix inequality, as opposed to sum-of-squares polynomials.

1.1.2 Non-Parametric Models

Kernel machines are a non-parametric learning technique that do not take a predefined structure, but instead constructs one solely from samples by implicitly mapping data vectors $\boldsymbol{x} \in \mathcal{X} \subset \mathbb{R}^n$ into a high- or infinite-dimensional feature space. To achieve this, they rely on a kernel function $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $(\boldsymbol{x}, \boldsymbol{x}') \mapsto k(\boldsymbol{x}, \boldsymbol{x}')$, which defines a structure to the data by acting as a similarity measure between data points. Every kernel defines a so-called reproducing kernel Hilbert space (RKHS)

$$\mathcal{H}_k = \Bigg\{ h : \quad h(\cdot) = \sum_{i=1}^{N} \alpha_i k(\cdot, \boldsymbol{c}_i), \\ \|h\|_k = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} \alpha_i \alpha_j k(\boldsymbol{z}_i, \boldsymbol{z}_j) < \infty \Bigg\}, \tag{5}$$

where $\alpha_i \in \mathbb{R}$ and $\boldsymbol{z}_i \in \mathcal{X}$. The RKHS comprises all the functions that can be approximated through the kernel. Linear combinations of universal kernels are well-suited for function approximation, since their spanned function space $\mathcal{H}_k$ is known to be dense in the continuous functions, thus, allowing to approximate continuous functions arbitrarily well (Micchelli et al., 2006).

Imposing global constraints on models relying on local structure induced by data is in general difficult. A common approach to realize shape constraints, e.g. positive definiteness, on a kernel-based approximator is by means of discretization (Giesl, 2007; Berkenkamp et al., 2016; Lederer and Hirche, 2019; Aubin-Frankowski and Szabó, 2020; Tesfazgi et al., 2021). By introducing inducing points on which appropriately conservative constraints depending on the Lipschitz bound of the system are enforced, constraint satisfaction can be guaranteed for a bounded region of the state space. In particular, inducing points may be arranged in a grid $\mathbb{G}_\varepsilon$ of equally spaced points based on a grid constant ε resulting in

$$V_{\mathcal{H}}(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{6a}$$

with

$$V_{\mathcal{H}}(\boldsymbol{0}) = 0 \tag{7a}$$
$$V_{\mathcal{H}}(\boldsymbol{x}_j) > \eta_1(\varepsilon, \boldsymbol{x}_j), \qquad \forall \boldsymbol{x}_j \in \mathbb{G}_{1,\varepsilon} \subset \mathbb{R}^n \tag{7b}$$
$$\dot{V}_{\mathcal{H}}(\boldsymbol{x}_j) < -\eta_2(\varepsilon, \boldsymbol{x}_j), \quad \forall \boldsymbol{x}_j \in \mathbb{G}_{2,\varepsilon} \subset \mathbb{R}^n \tag{7c}$$

where η_1 and η_2 are positive definite functions depending on the Lipschitz bound and the grid constant ε. Due to practical sampling limitations, the grid is necessarily confined to a bounded subset of the state-space, making it impossible to satisfy constraints globally. Additionally, the approach suffers from the curse of dimensionality in higher dimensional state-spaces. Apart from grid-based shape-constraints, it is also possible to approach the problem via a soft-constraint as in (Shirin et al., 2023). However, this comes at the cost of losing guarantees, since the original constraint may not be satisfied after solving the problem numerically.

1.1.3 Deep Learning

Neural networks (NNs) admit universal approximation capabilities (Hornik, 1991) and are therefore a popular function approximator choice. In (Lutter et al., 2021), the authors propose an architecture to construct Lyapunov functions by a quadrature of lower triangular matrices with positive diagonal parameterized by the NN. By

construction, the positive definiteness constraint (1b) is satisfied, while (1d) is generally not given, thus, the architecture is only able to guarantee a subset of the Lyapunov shape constraints. However, this comes at the cost of limiting the admissible NN architecture and approximation capabilities due to the quadratic structure. Furthermore, depending on the size of the NN, a large amount of data may be required for training.

A different approach is presented in (Chang et al., 2019), which employs satisfiability modulo theories (SMT) solvers to provide the required guarantees. The Lyapunov function itself consists directly of Ψ and no constraints on the NN architecture are imposed a-priori. The Lyaunov conditions are incorporated as a soft-constraint using a hinge-like loss. In an iterative fashion, a search algorithm based on an SMT solver tries to find states at which the Lyapunov constraints are violated, which are then used for a subsequent optimization of Ψ to satisfy the constraints on those states. This has the advantage that the approximation power of the model doesn't have to be constrained beforehand. However, it also induces the additional computational overhead of the SMT solver and requires searching a state-space for counter examples, which increases in complexity for higher dimensions.

The authors in (Richards et al., 2018) propose a Lyapunov Neural Network, which is non-negative (but not necessarily positive definite) on $\mathbb{R}^n$ by construction. Here, the weight matrices, activation functions and dimensions of each layer are designed to guarantee a trivial null-space of each layer. Satisfaction of the decrease condition (1c) is induced through a soft-constraint bound to a hinge-like loss (similar to Chang et al. (2019)). To estimate the region of attraction, a tightened version of (1c) is checked at discrete points, which allows a generalization to the vicinity of each inducing point given a Lipschitz bound.

1.2 Invertible Model Architectures

Guaranteeing invertibility is the primary challenge to obtain a diffeomorphic map. Most method guarantee this bijectivity through non-degeneracy of the Jacobian, which can be achieved in different ways, e.g., restricting the Jacobian to a triangular structure, relying on the connection to dynamical systems or including a Lipschitz bound. In the following, different approaches are presented.

1.2.1 Affine Coupling Layers

The construction of affine coupling layers (ACL) (Dinh et al., 2016) is based on the idea that the determinant of a triangular matrix is given by the product of its diagonal entries. Hence, guaranteeing a non-degenerate jacobian reduces to ensuring that the product of diagonal entries of is non-zero. Let $\boldsymbol{x}_k \in \mathbb{R}^n$, then the forward transformation $\boldsymbol{x}_k \to \boldsymbol{x}_{k+1}$ of an ACL is given by:

$$\boldsymbol{x}_{k+1,1:d} = \boldsymbol{x}_{k,1:d} \tag{8a}$$

$$\boldsymbol{x}_{k+1,d+1:n} = \boldsymbol{x}_{k+1,d+1:n} \odot e^{s(\boldsymbol{x}_{k,1:d})} + t(\boldsymbol{x}_{k,1:d}) \tag{8b}$$

where $\odot$ is the Hadamard product (or elementwise-product) and $d < n \in \mathbb{N}_+$. We will refer to (8) as ϕ_{ACL}. The notation $\boldsymbol{x}_{k,1:d}$ is to be understood as the first d states in $\boldsymbol{x}$. An ACL partitions the coordinates in $\boldsymbol{x}$ into two sets: the first d states are not transformed by an ACL (compare (8a)), while the remaining states (compare (8b)) undergo a transformation, which is only dependent on the first set. The translation function $t : \mathbb{R}^d \mapsto \mathbb{R}^{n-d}$ and the scaling function $s : \mathbb{R}^d \mapsto \mathbb{R}^{n-d}$ define the transformation for the latter set. For the case where $n = 2$, each of the two sets consist of exactly one coordinate of $\boldsymbol{x}$

$$\boldsymbol{J}_{\phi_{\text{ACL}}}(\boldsymbol{x}_k) = \begin{bmatrix} 1 & 0 \\ \frac{\partial \boldsymbol{x}_{k+1,2}}{\partial \boldsymbol{x}_{k,1}} & e^{s(x_1)} \end{bmatrix} \quad \text{with} \quad \det(\boldsymbol{J}_{\phi_{\text{ACL}}}(\boldsymbol{x})) = e^{s(x_1)} > 0 \tag{9}$$

where $\boldsymbol{J}_{\phi_{\text{ACL}}}$ has a triangular form and is guaranteed to have a determinant larger than zero. The functions t and s can take arbitrary form, i.e., NNs in (Dinh et al., 2015, 2016; Kingma and Dhariwal, 2018) or kernel machines in (Rana et al., 2020; English et al., 2024). While the partitioning architecture is restrictive, by considering a layered composition of ACLs, the expressiveness of the overall mapping can be increased. In each layer, the subset of transformed coordinates is changed such that each coordinate undergoes many transformations for a sufficiently deep composition. For this construction, universal approximation power of diffeomorphisms is demonstrated in (Teshima et al., 2020a; Ishikawa et al., 2023).

1.2.2 Neural ODEs

Due to the inherent bijectivity of flows of Lipschitz-continuous ODEs, Chen et al. (2018) propose to model these diffeomorphic maps by learning a dynamical system f_θ, whose flow over some time t defines the approximated map. In (Chen et al., 2018), f_θ is modeled by a NN. Since f_θ is a continuous-time system, calculating the flow requires solving an initial value problem, which is done numerically by an ODE solver, thus, the name Neural ODE (NODE)

$$\boldsymbol{x}_{k+1} = \phi_{\text{NODE}}(\boldsymbol{x}_k) := \int^{\delta t} f_{\text{NODE}}(\boldsymbol{x}(t), t)dt + \boldsymbol{x}_k. \tag{10}$$

NODEs can be thought of the continuous-layer equivalent of residual networks (ResNets) (He et al., 2016), since the number of layers depends on the number of function evaluation of the ODE solver. The work of Teshima et al. (2020b) and Ishikawa et al. (2023) establish the approximation power of NODEs.

1.2.3 Invertible ResNets

Invertible residual networks (i-ResNets) can be viewed as the time-discrete analogue of NODEs. In i-ResNets, a discrete propagation step is modeled by a residual function, which is parameterized by a neural network in (Behrmann et al., 2019) whose activation functions are restricted to contractive functionals. The forward evaluation is then given by

$$\boldsymbol{x}_{k+1} = \phi_{\text{IRN}}(\boldsymbol{x}_k) := \boldsymbol{x}_k + f_{\boldsymbol{W}}(\boldsymbol{x}_k) \tag{11}$$

where $\boldsymbol{W}$ are the weights for the residual block $f_{\boldsymbol{W}}$ and invertibility is guaranteed through a Lipschitz bound on $f_{\boldsymbol{W}}$. In particular, Behrmann et al. (2019) propose to build $f_{\boldsymbol{W}}$ with convolutional layers, which allows to guarantee invertibility by imposing a spectral norm constraint on $\boldsymbol{W}$.

1.2.4 Fast Diffeomorphic Matching

In (Perrin and Schlehuber-Caissier, 2016), the authors present *fast diffeomorphic matching* (FDM), which learns a diffeomorphism by scaling a directional vector $\boldsymbol{v} \in \mathbb{R}^n$ with a smooth, positive definite kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ centered around $\boldsymbol{c} \in \mathbb{R}^n$:

$$\boldsymbol{x}_{k+1} = \phi_{\text{FDM}}(\boldsymbol{x}_k) = \boldsymbol{x}_k + k(\boldsymbol{x}_k, \boldsymbol{c})\boldsymbol{v} \tag{12}$$

where the kernel k is defined as:

$$k(\boldsymbol{x}, \boldsymbol{c}) = e^{-\sigma^2\|\boldsymbol{x}-\boldsymbol{c}\|^2} \quad \text{s.t. } \sigma < \frac{1}{\sqrt{2}\|\boldsymbol{v}\|}e^{\frac{1}{2}} \tag{13}$$

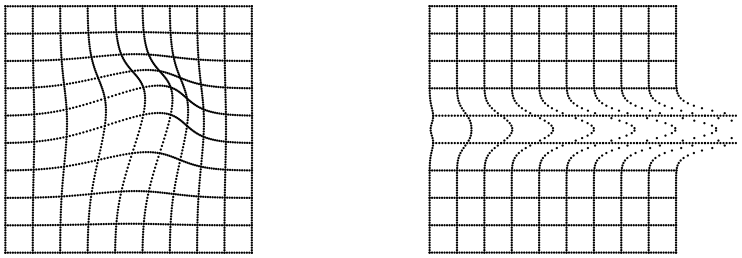
If the problem setup allows to calculate *target* states $\boldsymbol{z}^*$ for each $\boldsymbol{x}$, then the problem can be framed as a *matching* problem to minimize the mismatch $\|\phi(\boldsymbol{x}) - \boldsymbol{z}^*\|$. For each layer ϕ_{FDM}, the center $\boldsymbol{c}$ and direction $\boldsymbol{v}$ are calculated according to an error heuristic and afterwards, optimization is carried out over the bandwidth parameter σ.

1.2.5 Discussion of Architectures

From the presented approaches, NODEs and ACLs guarantee invertibility by construction without the need to impose additional constraints during optimization. However, the triangular form of $\boldsymbol{J}_{\phi_{\text{ACL}}}$ induces unbounded transformations, as depicted in Figure 1, which can be undesirable in a data-driven setting (such as learning Lyapunov functions) where one wishes to only influence a vicinity around each data-point. ACLs, i-ResNets and FDM can be evaluated analytically in the forward direction, however, only ACLs possess an analytical inverse expression. While an analytical inverse is important for density estimation (Dinh et al., 2016), in our considered scenario it is not necessary, since we primarily exploit the topology-preserving property of invertible maps.

The bijectivity of NODEs is theoretically guaranteed by the uniqueness of solutions to Lipschitz-continuous ODEs (Khalil, 2002), however, in practice numerical integration is performed using blackbox ODE solver and the actual flow generated by the ODE can only be evaluated approximately. This causes discretization errors (Zhu et al., 2022), not only in forward and reverse evaluation, but also in evaluation of the Jacobian.

In contrast, i-ResNets and FDMs guarantee their invertibility by posing constraints on the parameters and by appropriate choice of activation functions. For instance i-ResNets require a Lipschitz bound on the residual



(a) Free-form Jacobian (b) Triangular Jacobian

Figure 1: Examples of state-space transformations generated by approaches with free-form Jacobians (Figure 1(a)) and triangular Jacobians (Figure 1(b)). Due to the state pass-through, the transformation under triangular Jacobians causes a significant change along the coordinate axes of coordinates that were transformed.

block, which is not easy to guarantee (Kobyzev et al., 2021). This requires careful choice of the activation functions of the residual block and additional constraints on the parameters. In (Behrmann et al., 2019), the proposed residual block is composed of a convolutional network. For that special case, it is sufficient to restrict the network weights by a spectral norm constraint. However, this constraint does not readily extend to other activation functions and relying on convolutional layers makes an analytic jacobian intractable (Kobyzev et al., 2021), thus, requiring numerical approximations (Behrmann et al., 2019).

FDMs possess an analytical Jacobian since they rely on the Gaussian kernel. While the parameter constraint merely requires the bandwidth of the kernel to be bounded, the optimization is also performed over the bandwidth, which necessarily results in a nonlinear, nonconvex optimization. Thus, the beneficial property of a linear and convex optimization typical for Kernel machines is lost. Additionally, FDM always searches over a single scalar bandwidth, while the directional vector $\boldsymbol{v}$ is chosen in advance. This poses a restriction for multi-dimensional maps with $n > 1$, as the transformation of all coordinates is coupled to a single decision variable during optimization. If one tried to additionally optimize over $\boldsymbol{v}$, the bound for the bandwidth cannot be computed in advance anymore, since it depends on $\boldsymbol{v}$.

For the application of learning Lyapunov functions from data, an analytical expression for the jacobian is a key requirement, as it becomes necessary for evaluating the gradient of V, which makes NODEs and IRNs unsuitable. This leaves ACLs and FDM as potential candidates. They both however have limitations in the granularity of the transformation that they are able to model. ACLs cause an unbounded transformation due to the their triangular jacobian and FDM couples the transformation along all coordinates to a single optimization variable, leaving no flexibility to control the coordinate-wise transformation at optimization time. Our proposed Deep Diffeomorphic RBF Network architecture overcomes these issues. Here, each layer is parameterized by a bounded expansion over Gaussian kernels, which is linear in parameters and only requires simple box constraints in the coefficients to guarantee invertibility. This circumvents the restriction to a triangular Jacobian, enables granular coordinate-wise control over the transformation during optimization. Simultaneously, the architecture allows for an analytical evaluation of the forward map and Jacobian.

2 MISSING AND EXTENDED PROOFS

The section contains detailed proofs of the results that are missing in the main paper and extends some of the proofs shown in the main paper.

2.1 Extended Proof of Theorem 4.1

Before presenting the extended proof of Theorem 4.1, we show the result by Ostrowski (1937) on the determinant of diagonally dominant square matrices, which becomes necessary for the proof:

Theorem 2.1 (Determinant of Diagonally dominant matrices (Ostrowski, 1937)). *If $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{E}$ is a real n by n matrix with the elements of $\boldsymbol{E}$ being bounded in absolute value by $\epsilon \leq \frac{1}{n}$, then a lower bound of the determinant of $\boldsymbol{A}$ is given by:*

$$\det(A) \geq 1 - n\epsilon \tag{14}$$

With this result as context, we present the extended proof of Theorem 4.1.

Theorem 4.1 (Bijective Residual RBF Map). *Consider a mapping $\boldsymbol{\phi}(\cdot)$ as in (9) with multi-variate Gaussian kernel $k(\cdot,\cdot)$ (12). If the entries $w_{i,j}$ in the weight matrix $\boldsymbol{W}$ adhere to the box-constraints*

$$\begin{aligned} |w_{i,j}| < & \frac{1}{nN \sum_{l=1}^{n} |Q_{j,l}| e^{-\frac{1}{2}} \sqrt{D_{j,j}}} := \rho(n, N, \boldsymbol{\Sigma}) \\ & \forall i \in [1, \ldots, N], j \in [1, \ldots, n], \end{aligned} \tag{15}$$

where $\boldsymbol{Q}$ and $\boldsymbol{D}$ are due to the eigendecomposition $\boldsymbol{\Sigma}^{-1} = \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top$, then $\boldsymbol{\phi}(\cdot)$ is a diffeomorphism.

Proof. To show that the mapping ϕ is a diffeomorphism, it has to be continuously differentiable and admit a Jacobian $\det(\boldsymbol{J}_\phi(\boldsymbol{x})) \neq 0 \ \forall \boldsymbol{x} \in \mathbb{R}^n$. The first property is satisfied, since a Gaussian kernel k is continuously differentiable, and thus, a mapping ϕ, which is given by a linearly weighted sum of k plus an additional linear function, is also at least once continuously differentiable. We define the residual term of the j-th entry of ϕ as:

$$\psi_j(\boldsymbol{x}) = \sum_{i=1}^{N} w_{i,j} k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{16}$$

To prove bijectivity, note that the jacobian of ϕ at $\boldsymbol{x}$ can be decomposed into an identity matrix $\boldsymbol{I}$ and a *disturbance* matrix $\boldsymbol{E}$ containing the partial derivatives $\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}$:

$$\boldsymbol{J}_\phi(\boldsymbol{x}) = \begin{bmatrix} 1 + \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_1} & \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_2} & \cdots & \frac{\partial \psi_1(\boldsymbol{x})}{\partial x_n} \\ \frac{\partial \psi_2(\boldsymbol{x})}{\partial x_1} & 1 + \frac{\partial \psi_2(\boldsymbol{x})}{\partial x_2} & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial \psi_n(\boldsymbol{x})}{\partial x_1} & & & 1 + \frac{\partial \psi_n(\boldsymbol{x})}{\partial x_n} \end{bmatrix} := \boldsymbol{I} - \boldsymbol{E}(\boldsymbol{x}) \tag{17}$$

where the entries of $\boldsymbol{E}(\boldsymbol{x})$ are given by $E_{i,j} = -\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}$. From Theorem 2.1 we have that the determinant of $\boldsymbol{J}_\phi$ can be bounded from below if there is an ϵ satisfying:

$$\epsilon \leq \frac{1}{n} \implies \det(\boldsymbol{J}_\phi(\boldsymbol{x})) \geq 1 - n\epsilon \tag{18}$$

where ϵ is the absolute bound of the entries in $\boldsymbol{E}(\boldsymbol{x})$:

$$\epsilon = \max_{i,j}(| - \frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}|) \quad \forall i \in [1, \ldots, n], j \in [1, \ldots, n]. \tag{19}$$

Consequently, ϕ is invertible if (15) guarantees satisfaction of the following bound on the partial derivative of $\psi_i(\boldsymbol{x})$:

$$\max_{i,j}(| - \frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}|) \leq \frac{1}{n} \tag{20}$$

In order to derive a bound on $\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}$, we first derive a bound for $\frac{\partial k}{\partial x_j}$ for the case where $\boldsymbol{\Sigma}$ takes a diagonal structure:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix} \tag{21}$$

with $\sigma_i > 0$. Since $\boldsymbol{\Sigma}$ is diagonal, its inverse $\boldsymbol{\Sigma}^{-1}$ is also a diagonal matrix given by:

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} \sigma_1^{-2} & & 0 \\ & \ddots & \\ 0 & & \sigma_n^{-2} \end{bmatrix} \coloneqq \boldsymbol{H} \tag{22}$$

The first and second partial derivatives of $k(\boldsymbol{x}, \boldsymbol{x}_c)$ with respect to x_j yield:

$$\frac{\partial k}{\partial x_j} = -H_{j,j} k(\boldsymbol{x}, \boldsymbol{x}_c)(x_j - x_{c,j}) \tag{23a}$$

$$\frac{\partial^2 k}{\partial^2 x_j} = -H_{j,j} k(\boldsymbol{x}, \boldsymbol{x_c})\big[- H_{j,j}(x_j - x_{c,j})^2 + 1\big] \tag{23b}$$

where $x_{c,j}$ is the j-th entry of the center point $\boldsymbol{x}_c$. By setting (23b) to zero to find extreme points of (23a) and from the fact that $k(\boldsymbol{x}, \boldsymbol{x_c}) \neq 0 \ \forall \boldsymbol{x} \in \mathbb{R}^n$, we get the following condition for the extreme points:

$$\frac{\partial^2 k}{\partial^2 x_j} = 0 \iff 1 = H_{j,j}(x_j - x_{c,j})^2 \tag{24}$$

and hence $\frac{\partial k}{\partial x_j}$ takes its maximum and minimum values at $(x_j - x_{c,j}) = \pm \frac{1}{\sqrt{H_{j,j}}}$. Condition (24) hence gives us the j-th coordinate x_j at which (23a) takes its extreme values with respect to the center $\boldsymbol{x}_c$. However, it is still necessary to determine the other coordinates of $\boldsymbol{x}$, namely x_i for $i \neq j$. For these other dimensions, we have:

$$\frac{\partial^2 k}{\partial x_j \partial x_i} = H_{j,j} H_{i,i} k(\boldsymbol{x}, \boldsymbol{x}_c)(x_j - x_{c,j})(x_i - x_{c,i}) \tag{25}$$

Since we are interested in the extreme points of (23a), we again set (25) to zero and plug in the above derived solution of $(x_j - x_{c,j}) = \pm \frac{1}{\sqrt{H_{j,j}}}$:

$$\frac{\partial^2 k}{\partial x_j \partial x_i} = 0 \iff 0 = \pm \sqrt{H_{j,j}} H_{i,i} k(\boldsymbol{x}, \boldsymbol{x}_c)(x_i - x_{c,i}) \tag{26}$$

which has its only root at $(x_i - x_{c,i}) = 0$ due to the positive definiteness of k. We are only interested in the largest absolute value and due to the symmetry of k, we choose $(x_j - x_{c,j}) = \frac{1}{\sqrt{H_{j,j}}}$, and we therefore have $\boldsymbol{x} - \boldsymbol{x}_c = \frac{1}{\sqrt{H_{j,j}}} \boldsymbol{e}_j$ with $\boldsymbol{e}_j$ being the unit vector in the j-th dimension. Inserting the maximizer in absolute value into (23a) yields :

$$\| \frac{\partial k}{\partial x_j} \| = \| - H_{j,j} e^{-\frac{1}{2}(\frac{1}{\sqrt{H_{j,j}}} \boldsymbol{e}_j) \boldsymbol{H} (\frac{1}{\sqrt{H_{j,j}}} \boldsymbol{e}_j)} \frac{1}{\sqrt{H_{j,j}}} \| \tag{27a}$$

$$= \| - e^{-\frac{1}{2} \frac{1}{\sqrt{H_{j,j}}} H_{j,j} \frac{1}{\sqrt{H_{j,j}}}} \sqrt{H_{j,j}} \| \tag{27b}$$

$$= e^{-\frac{1}{2}} \sqrt{H_{j,j}} \tag{27c}$$

Concluding the proof for the case where $\boldsymbol{\Sigma}$ is diagonal, we note that the bound on the j-th partial derivative of k is only dependent on the j-th diagonal entry of $\boldsymbol{H}$. To derive a bound of $\| \frac{\partial k}{\partial x_j} \|$ for the case where $\boldsymbol{\Sigma}$ symmetric, we first perform an eigendecomposition of $\boldsymbol{H}$, which is guaranteed to exist since $\boldsymbol{H}$ is positive definite by definition:

$$\boldsymbol{H} = \boldsymbol{Q} \boldsymbol{D} \boldsymbol{Q}^\top \tag{28}$$

Note that the matrix $\boldsymbol{D}$ is diagonal and contains the positive eigenvalues of $\boldsymbol{H}$, hence $\boldsymbol{D}$ is also a valid covariance matrix. We define another kernel parameterized by $\boldsymbol{D}$ on a transformed state $\boldsymbol{z} = \boldsymbol{Q}^\top \boldsymbol{x}$:

$$k_D(\boldsymbol{z}, \boldsymbol{z}_i) = e^{-\frac{1}{2}(\boldsymbol{z} - \boldsymbol{z}_i)^\top \boldsymbol{D} (\boldsymbol{z} - \boldsymbol{z}_i)} \tag{29}$$

This allows us to express $\nabla_{\boldsymbol{x}} k$ through $\nabla_{\boldsymbol{z}} k_D$:

$$\nabla_{\boldsymbol{x}} k = -k(\boldsymbol{x}, \boldsymbol{x}_c)\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_c) \tag{30a}$$
$$= -e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_i)^\top \boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}_i)}\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top(\boldsymbol{x} - \boldsymbol{x}_c) \tag{30b}$$
$$= -e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_i)^\top \boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top(\boldsymbol{x}-\boldsymbol{x}_i)}\boldsymbol{Q}\boldsymbol{D}\boldsymbol{Q}^\top(\boldsymbol{x} - \boldsymbol{x}_c) \tag{30c}$$
$$= -e^{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_i)^\top \boldsymbol{D}(\boldsymbol{z}-\boldsymbol{z}_i)}\boldsymbol{Q}\boldsymbol{D}(\boldsymbol{z} - \boldsymbol{z}_c) \tag{30d}$$
$$= \boldsymbol{Q}\nabla_{\boldsymbol{z}} k_D \tag{30e}$$

From this, it follows that the j-th partial derivative of k is given by:

$$\frac{\partial k}{\partial x_j} = \sum_{i=1}^{n} Q_{j,i} \frac{\partial k_D}{\partial z_j} \tag{31}$$

From (31) we conclude that the partial derivatives of k for the non-diagonal case are weighted partial derivatives of k_D, which possesses a diagonal covariance matrix. This allows us to derive the bound as follows:

$$\|\frac{\partial k}{\partial x_j}\| = \|\sum_{i=1}^{n} Q_{j,i} \frac{\partial k_D}{\partial z_j}\| \tag{32a}$$
$$\leq \sum_{i=1}^{n} |Q_{j,i}| \|\frac{\partial k_D}{\partial z_j}\| \tag{32b}$$
$$\leq \sum_{i=1}^{n} |Q_{j,i}| e^{-\frac{1}{2}} \sqrt{D_{j,j}} \tag{32c}$$

where we used the triangular inequality, the previously derived bound for partial derivatives with diagonal covariance on k_D and the fact that $D_{j,j} > 0$.

Next, we derive a bound for $\|\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}\|$:

$$\| - \frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}\| = \| - \frac{\partial}{\partial x_j}\big(\sum_{i=1}^{N} w_{i,j} k(\boldsymbol{x}, \boldsymbol{x}_i)\big)\| \tag{33a}$$
$$= \| - \sum_{i=1}^{N} w_{i,j} \frac{\partial k}{\partial x_j}\| \tag{33b}$$
$$\leq \sum_{i=1}^{N} |w_{i,j}| \|\frac{\partial k}{\partial x_j}\| \tag{33c}$$

To resolve the sum in (33c), we restrict every $w_{i,j}$ to the same absolute maximum value, which yields:

$$\sum_{i=1}^{N} |w_{i,j}| = N|w_{i,j}|. \tag{34}$$

Hence, we get the following bound for (20):

$$\max_{i,j}(\| - \frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}\|) \leq N|w_{i,j}| \|\frac{\partial k}{\partial x_j}\| \tag{35a}$$

This bound gives the desired constraint (15) after rearranging (20) and inserting (32c):

$$N|w_{i,j}| \|\frac{\partial k}{\partial x_j}\| < \frac{1}{n} \tag{36a}$$
$$|w_{i,j}| < \frac{1}{nN\|\frac{\partial k}{\partial x_j}\|} \tag{36b}$$
$$= \frac{1}{nN(\sum_{l=1}^{n} |Q_{j,l}| e^{-\frac{1}{2}} \sqrt{D_{j,j}})} \tag{36c}$$

which concludes the proof. □

2.2 Proof of Proposition 4.2

Proposition 4.2. *The composition of T orientation preserving diffeomorphisms*

$$\boldsymbol{\Phi}(\boldsymbol{x}) \coloneqq \bigcirc_{t=1}^{T} \boldsymbol{\phi}_t(\boldsymbol{x}), \quad T \in \mathbb{N}_+, \tag{37}$$

is also an orientation preserving diffeomorphism.

Proof. Since each map ϕ_t in (37) is an orientation-preserving diffeomorphism, it follows

$$\det(\boldsymbol{J}_{\phi_i}(\boldsymbol{x})) > 0 \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \forall t \in [1, \dots, T]. \tag{38}$$

The Jacobian of the functional composition is

$$\boldsymbol{J}_{\Phi}(\boldsymbol{x}) = \frac{\partial \phi(\boldsymbol{x})}{\partial \boldsymbol{x}} = \prod_{t=1}^{T} \frac{\partial \phi_t(\boldsymbol{x})}{\partial \boldsymbol{x}} = \prod_{t=1}^{T} \boldsymbol{J}_{\phi_t}(\boldsymbol{x}) \tag{39}$$

with determinant

$$\det(\boldsymbol{J}_{\Phi}(\boldsymbol{x})) = \prod_{t=1}^{T} \det(\boldsymbol{J}_{\phi_t}(\boldsymbol{x})) > 0 \tag{40}$$

and hence $\Phi(\boldsymbol{x})$ is invertible. Finally, since each ϕ_i is a smooth map, their composition is also smooth by the chain rule, which concludes the proof. □

2.3 Proof of Lemma 4.3

For completeness we introduce the following formal notion of approximation capabilities

Definition 2.2 (sup-universality, (Teshima et al., 2020b))**.** Let $\mathcal{M}$ be a model, which is a set of mappings from $\mathbb{R}^m$ to $\mathbb{R}^n$. Let $\mathcal{F}$ be a set of mappings $f : U_f \mapsto \mathbb{R}^n$, where U_f is measurable subset of $\mathbb{R}^m$, which may depend on f. We say that $\mathcal{M}$ is a sup-universal approximator or has the sup-universal approximation property for $\mathcal{F}$, if for any $f \in \mathcal{F}$, any $\epsilon > 0$ and any compact subset $S \subset U_f$, there exists $g \in \mathcal{M}$ such that $\|f - g\|_{\sup,S} < \epsilon$.

Intuitively, this means that the pointwise error over a subset S between a target functional f and the model g can be made arbitrarily small. The limitation of S being bounded is a technicality and not a practical restriction, as S can be chosen arbitrarily large, as long as it is bounded. It is also important to mention that $\mathcal{M}$ being sup-universal does not imply $f \in \mathcal{M}$, as universality is defined with respect to approximation, not equality (Li et al., 2022).

Lemma 4.3. *Let $\phi : \mathbb{R}^n \to \mathbb{R}^n$ be defined as*

$$\phi(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{41}$$

with $\boldsymbol{W} \in \mathbb{R}^{n \times N}$ and denote by $\mathcal{M}$ the set of all possible ϕ as defined in (41)*. Then, $\mathcal{M}$ is a* sup*-universal approximator for Lipschitz continuous maps $\mathcal{L}(\mathbb{R}^n)$.*

Proof. Consider the case $n = 1$

$$\phi(x) = x + \boldsymbol{W}\boldsymbol{K}(x) = x + \sum_{i=1}^{N} w_i k(\boldsymbol{x}, \boldsymbol{x}_i), \tag{42}$$

and define the residual function r

$$r(x) = \sum_{i=1}^{N} w_i k(\boldsymbol{x}, \boldsymbol{x}_i). \tag{43}$$

Hence, (42) can be written as

$$\phi(x) = x + r(x). \tag{44}$$

Note that (43) constitutes a linear combination of universal kernels for which sup-universality for $\mathcal{L}(\mathbb{R})$ was shown in (Micchelli et al., 2006). Thus, it follows for any $\epsilon > 0$ and any $f \in \mathcal{L}(\mathbb{R})$:

$$\|f - r\|_{\sup,S} < \epsilon \tag{45}$$

with S defined as in Definition 2.2. Since the function $h(x) = -x$ is in $\mathcal{L}(\mathbb{R})$, then $r(\cdot)$ must also be sup-universal for $\tilde{f} = f + h$ and we have:

$$\begin{aligned} \|\tilde{f}(x) - r(x)\|_{\sup,S} &= \|f(x) + h(x) - r(x)\|_{\sup,S} && \text{(46a)} \\ &= \|f(x) - x - r(x)\|_{\sup,S} && \text{(46b)} \\ &= \|f(x) - \phi(x)\|_{\sup,S} < \epsilon && \text{(46c)} \end{aligned}$$

Consequently, the set of possible ϕ is sup-universal for $\mathcal{L}(\mathbb{R})$ in the case $n = 1$. From that is follows that the set of functionals $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$, where each coordinate-wise functional ϕ_i with $i \in [1, \ldots, n]$ in (41) is sup-universal for $\mathcal{L}(\mathbb{R})$ is sup-universal for $\mathcal{L}(\mathbb{R}^n)$. That is, each coordinate mapping $\phi_i : \mathbb{R}^n \mapsto \mathbb{R}$ is a sup-universal approximator for $\mathcal{L}(\mathbb{R})$ and independent of ϕ_j with $j \neq i$. □

2.4 Extended Proof of Theorem 4.4

Before presenting the extended proof of Theorem 4.4, we state all the necessary definitions, lemmas and facts which are taken from Teshima et al. (2020b) and Teshima et al. (2020a) for completeness.

Definition 2.3 (Initial Value Problem (IVP), Teshima et al. (2020b))**.** For $f \in \mathcal{L}(\mathbb{R}^n)$, $\boldsymbol{x} \in \mathbb{R}^n$ and $t \in \mathbb{R}$, define:

$$\mathrm{IVP}[f](\boldsymbol{x}, t) := z(t) \tag{47}$$

The IVP of f with $\boldsymbol{z}(0) = \boldsymbol{x}$ is the state $\boldsymbol{z}(t)$ given by integrating f for time t with initial condition $\boldsymbol{x}$. For a given set of Lipschitz continuous systems, define the set of IVP solutions for unit time from Definition 2.3 as:

Definition 2.4 (Autonomous-ODE flow endpoints, (Li et al., 2022; Teshima et al., 2020b))**.** Let $\mathcal{F}$ be a subset of Lipschitz continuous maps. With $\mathcal{F} \subset \mathcal{L}(\mathbb{R}^n)$, define the set of flow endpoints as:

$$\boldsymbol{\Psi}(\mathcal{F}) := \{\mathrm{IVP}[f](\cdot, 1) | f \in \mathcal{F}\} \tag{48}$$

From Definition 2.4 we have that $\boldsymbol{\Psi}(\mathcal{F})$ is a set of maps $\psi : \mathbb{R}^n \mapsto \mathbb{R}^n$ which is generated by the solutions to all IVPs at any state $\boldsymbol{x} \in \mathbb{R}^n$ for all ODEs $f \in \mathcal{F}$ for unit time $t = 1$. One can show universal approximation power of this set with respect to general flow endpoints of Lipschitz continuous maps, if the generating ODE class $\mathcal{F}$ is a universal approximator for $\mathcal{L}(\mathbb{R}^n)$:

Lemma 2.5 (Approximation of Autonomous-ODE flow endpoints, Teshima et al. (2020b))**.** *Assume $\mathcal{F} \subset \mathcal{L}(\mathbb{R}^n)$ is a* sup*-universal approximator for $\mathcal{L}(\mathbb{R}^n)$. Then, $\boldsymbol{\Psi}(\mathcal{F})$ is a* sup*-universal approximator for $\boldsymbol{\Psi}(\mathcal{L}(\mathbb{R}^n))$.*

Further, the proof for universal approximation power relies on compactly supported diffeomorphisms:

Definition 2.6 (Compactly supported diffeomorphisms, Teshima et al. (2020a))**.** A diffeomorphism ϕ on $\mathbb{R}^n$ is compactly supported, if there exists a compact subset $S \subset \mathbb{R}^n$ such that for any $x \notin S, \phi(\boldsymbol{x}) = \boldsymbol{x}$. We denote by $\bar{\mathcal{D}}_c^2$ the set of compactly supported $\mathcal{C}^2$-diffeomorphisms.

Hence, a compactly supported diffeomorphism defaults to the identity map for states outside the compact set S. The following fact establishes that the universal approximation power is retained under composition:

Fact 2.7 (Compatibility of composition and approximation, Teshima et al. (2020a))**.** *Let $\mathcal{M}$ be a set of locally bounded maps from $\mathbb{R}^n$ to $\mathbb{R}^n$, and $F_1, \ldots, F_k$ be continuous maps from $\mathbb{R}^n$ to $\mathbb{R}^n$. Assume for any $\epsilon > 0$ and any compact set $S \subset \mathbb{R}^n$, there exists $\tilde{G}_1, \ldots, \tilde{G}_T \in \mathcal{M}$ such that, for $1 \leq t \leq T, \|F_t - \tilde{G}_t\|_{\sup,S} < \epsilon$. Then for any $\epsilon > 0$ and any compact set $S \subset \mathbb{R}^n$, there exist $G_1, \ldots, G_T \in \mathcal{M}$ such that*

$$\|\bigcirc_{t=1}^{T} F_t - \bigcirc_{t=1}^{T} G_t\|_{\sup,S} < \epsilon \tag{49}$$

where local boundedness of a map means that the image under a map f for a small neighborhood of $\boldsymbol{x}$ is a bounded set.

Lemma 2.8 (Existence of locally supported diffeomorphisms, Teshima et al. (2020a))**.** *Let $B_r \subset \mathbb{R}^n$ be an open ball of radius r with origin 0, and let $\phi : B_r \mapsto \phi(B_r) \subset \mathbb{R}^n$ be a $\mathcal{C}^2$-diffeomorphism onto its image such that $\phi(0) = 0$ and $\boldsymbol{J}_\phi(0)) = I$. Let $\nu \in (0, \frac{r}{2})$. Then there exists $h \in \bar{\mathcal{D}}_c^2$ such that $\phi(x) = h(x)$ for any $x \in B_{r-\nu}$.*

Definition 2.9 (Flow endpoints E^r in $\bar{\mathcal{D}}_c^r$, Teshima et al. (2020b))**.** Let $1 \leq r \leq \infty$. Let $E^r \subset \bar{\mathcal{D}}_c^r$ be the set of diffeomorphisms g of the form $g(\boldsymbol{x}) = \psi(\boldsymbol{x}, 1)$ for a map $\psi : \mathbb{R}^n \times U \mapsto \mathbb{R}^n$ such that:

- $U \subset \mathbb{R}$ is an open interval containing $[0, 1]$.
- $\psi(\boldsymbol{x}, 0) = \boldsymbol{x}$
- $\psi(\cdot, t) \in \bar{\mathcal{D}}_c^r$ for any $t \in U$
- $\psi(\boldsymbol{x}, s + t) = \psi(\psi(\boldsymbol{x}, s), t)$ for any $s, t \in U$ with $s + t \in U$
- ψ is C^r on $\mathbb{R}^n \times U$
- There exists a compact subset $S_\psi \subset \mathbb{R}^n$ such that $\cup_{t \in U} \text{supp} \psi(\cdot, t) \subset S_\psi$

Given these formalizations, the proof for Theorem 4.4 can be presented.

Theorem 4.4 (sup-universality of DD-RBFNs)**.** *Let $\phi_t \in \mathcal{M}_\rho$ define the set of all functions that satisfy Theorem 4.1. Then, the composite model class*

$$\mathcal{M}_{DD\text{-}RBFN} = \left\{ \bigcirc_{t=1}^{T} \phi_t | \phi_t \in \mathcal{M}_\rho, k \in \mathbb{N}_+ \right\} \tag{50}$$

is a sup*-universal approximator for $\mathcal{C}^2$ diffeomorphisms.*

Proof. We first show that a single layer $\phi_i \in \mathcal{M}_\rho$ can equivalently be regarded as an infinitesimal step of a time continuous system. Recall the model derived in Theorem 4.1:

$$\phi_i(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{51}$$

with coefficient bounds:

$$\|w_{i,j}\| < \frac{1}{nN \sum_{l=1}^{n} \|Q_{i,l}\| e^{-\frac{1}{2}} \sqrt{D_{i,i}}} := \rho(n, N, \boldsymbol{\Sigma}) \quad \forall i \in [1, \ldots, N], j \in [1, \ldots, n] \tag{52}$$

Equivalently, it can be interpreted as the discretization of a continuous time system with the *virtual* timestep $\delta \in (0, 1]$:

$$\phi_\delta(\boldsymbol{x}) := \boldsymbol{x} + \delta \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{53}$$

for which up to now we had $\delta = 1$. With the introduction of δ, the coefficient bounds have to be adjusted accordingly:

$$\|\delta w_{i,j}\| < \frac{1}{nN \sum_{l=1}^{n} \|Q_{i,l}\| e^{-\frac{1}{2}} \sqrt{D_{i,i}}} \tag{54a}$$

$$|\delta| \|w_{i,j}\| < \frac{1}{nN \sum_{l=1}^{n} \|Q_{i,l}\| e^{-\frac{1}{2}} \sqrt{D_{i,i}}} \tag{54b}$$

$$\|w_{i,j}\| < \frac{1}{nN|\delta| \sum_{l=1}^{n} \|Q_{i,l}\| e^{-\frac{1}{2}} \sqrt{D_{i,i}}} := \rho_\delta(n, N, \boldsymbol{\Sigma}, \delta) \tag{54c}$$

As $\delta \to 0$, we have $\rho_\delta \to \infty$, which means that for a smaller virtual integration time δ, the absolute coefficient bound becomes arbitrarily large. We can hence approximate the instantaneous integration step of any $f \in \mathcal{L}(\mathbb{R}^n)$ by $\phi_\delta \in \mathcal{M}_\rho$. Since Lipschitz continuity is defined for a bounded Lipschitz constant $L < \infty$, there always exists

a sufficiently small $\delta > 0$ such that the parameter bound ρ_δ is large enough to realize a Lipschitz constant of L. In the limit $\delta \to 0$, Lemma 4.3 applies and $\mathcal{M}_\rho \subset \mathcal{L}(\mathbb{R}^n)$ with a sufficiently small δ is a sup-universal approximator for $\mathcal{L}(\mathbb{R}^n)$. Consequently, the generated set of flow endpoints for unit time $\mathbf{\Psi}(\mathcal{M}_\rho)$ can be realized by a composition of sufficiently many ϕ_δ:

$$\phi_\delta \in \mathcal{M}_\rho \implies \Phi = \bigcirc_{t=1}^{\tilde{T}} \phi_{\delta,t} \in \mathcal{M}_{\text{DD-RBFN}} \tag{55}$$

According to Lemma 2.5, $\mathbf{\Psi}(\mathcal{M}_\rho)$ is thus a sup-universal approximator for $\mathbf{\Psi}(\mathcal{L}(\mathbb{R}^n))$, since $\phi_\delta \in \mathcal{M}_\rho$ is a sup-universal approximator for $\mathcal{L}(\mathbb{R}^n)$ given a sufficiently small δ. The remainder of the proof is analogous to (Teshima et al., 2020b), with the modification that we do not have to consider the affine output layer transformation of NODEs.

Let $F : U \mapsto \mathbb{R}^n$ be any $\mathcal{C}^2$ diffeomorphism, $S \subset U$ be compact and $\epsilon > 0$. Then, according to Lemma 2.8, there exist $G \in \bar{\mathcal{D}}_c^2$ such that:

$$F|_S = G|_S, \tag{56}$$

where $F|_S = G|_S$ denotes that F and G are equivalent on the set S. Note that Lemma 2.8 is only true on some ball B, however, we can simply choose $S \subset B$ which establishes the above equality. Next, let G be defined as a finite composition of flow endpoints (Teshima et al., 2020b)

$$G = \bigcirc_{i=1}^{N} g_i \tag{57}$$

Since each g_i is a flow endpoint, it is generated by the IVP of some $f_i \in \mathcal{L}(\mathbb{R}^n)$ (Teshima et al., 2020b):

$$g_i = \text{IVP}[f_i](\cdot, 1) \tag{58}$$

Invoking Lemma 2.5, Fact 2.7 and the previously derived result that $\mathbf{\Psi}(\mathcal{M}_\rho)$ is a sup-universal approximator for $\mathbf{\Psi}(\mathcal{L}(\mathbb{R}^n))$, there exist $\psi_i \in \mathbf{\Psi}(\mathcal{M}_\rho)$ such that:

$$\| \bigcirc_{i=1}^{N} g_i - \bigcirc_{i=1}^{N} \psi_i \|_{\text{sup},S} < \epsilon \tag{59}$$

and consequently:

$$\| F - \bigcirc_{i=1}^{N} \psi_i \|_{\text{sup},S} < \epsilon \tag{60}$$

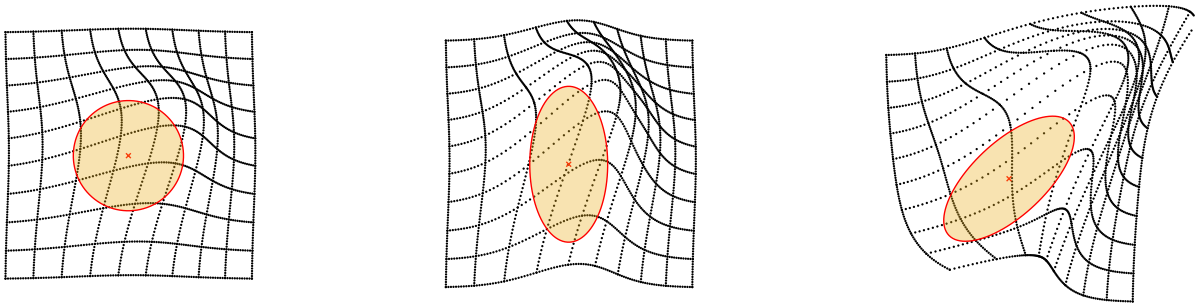
Finally, notice that from (55), we have that each ψ_i can be described by sufficiently many $\phi_{\delta,t} \in \mathcal{M}_\rho$ for a small enough $\delta \in (0, 1]$ and consequently, there exists a number $T = \tilde{T}N$ such that

$$\| F - \bigcirc_{t=1}^{T} \phi_{\delta,t} \|_{\text{sup},S} < \epsilon, \quad \phi_{\delta,t} \in \mathcal{M}_\rho, \tag{61}$$

which concludes the proof. □

3 ADDITIONAL INFORMATION ON DEEP DIFFEOMORPHIC RBF NETWORKS

This section aims to build some intuition about the mappings that DD-RBFNs generate and how kernels in general behave under state-space transformations.



(a) $\boldsymbol{\Sigma} = \boldsymbol{I}_2\sigma$ (b) $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2)$, $\sigma_1 \neq \sigma_2$ (c) $\boldsymbol{\Sigma} \in S_{n+}$

Figure 2: Different choices of covariance matrices, their σ influence radius for the case $n = 2$. The center point is marked with a red cross, while the σ-influence area is shaded in red. The grid deformation shows the transformation $\boldsymbol{z} = \phi(\boldsymbol{x})$ that each sigma choice inflicts. For each figure, the largest allowed $\boldsymbol{W}$ according to (15) has been chosen.

3.1 Covariance Matrix and Kernel Morphing

Intuitively, the covariance matrix $\boldsymbol{\Sigma}$ controls the effective influence of each RBF w.r.t. the center point. Since our formulation allows symmetric covariance matrices, the influence area is not limited to a circle but may be chosen as an arbitrarily rotated ellipsoid, as shown in Figure 2.

Since each ϕ_t transforms the state space in which the kernel of a successor ϕ_{t+1} is defined, we are also changing the effective kernel shape w.r.t. the original state space $\boldsymbol{x}$. To visualize this, recall that the kernel k in $\boldsymbol{x}$ centered around $\boldsymbol{x}_c$ is defined as:

$$k(\boldsymbol{x}, \boldsymbol{x}_c) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_c)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{x}_c)} \tag{62}$$

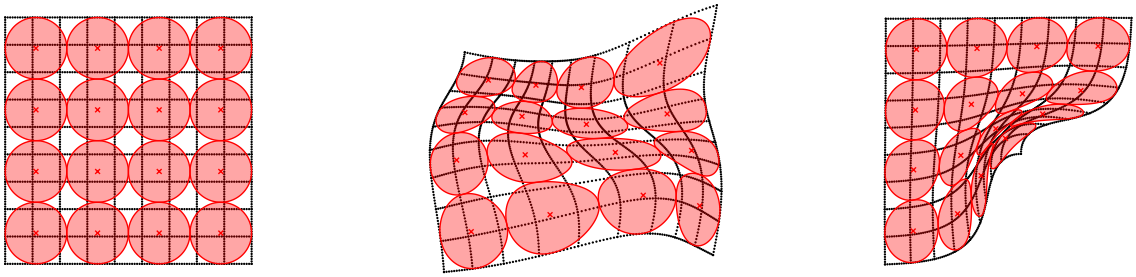
Under the transformation ϕ, we arrive at the transformed states $\boldsymbol{z} = \phi(\boldsymbol{x})$. The original kernel (62) is expressed in morphed coordinates $\boldsymbol{z}$ as follows:

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}_c) &= k(\phi^{-1}(\boldsymbol{z}), \phi^{-1}(\boldsymbol{z}_c)) \\ &= e^{-\frac{1}{2}(\phi^{-1}(\boldsymbol{z})-\phi^{-1}(\boldsymbol{z}_c))^\top \boldsymbol{\Sigma}^{-1}(\phi^{-1}(\boldsymbol{z})-\phi^{-1}(\boldsymbol{z}_c))} \\ &:= k_\phi(\boldsymbol{z}, \boldsymbol{z}_c) \end{aligned} \tag{63a}$$

where $\boldsymbol{z}_c = \phi(\boldsymbol{x}_c)$ is the transformed center. In the space $\boldsymbol{z}$ defined through $\phi(\boldsymbol{x})$, an ellipsoid σ-covering is in general not an ellipsoid anymore. Take for instance σ-covering as depicted in Figure 3(a), where the border of each red circle centered around $\boldsymbol{x}_c$ obeys the equality $\sigma = \|\boldsymbol{x} - \boldsymbol{x}_c\|$. A diffeomorphism ϕ induces a nonlinear, state-space transformation transporting $\boldsymbol{x}$ to $\boldsymbol{z}$. Thus, the same circle can be described in $\boldsymbol{z}$ coordinates by $\sigma = \|\phi^{-1}(\boldsymbol{z}) - \phi^{-1}(\boldsymbol{z}_c)\|$, since we have $\boldsymbol{x} = \phi^{-1}(\boldsymbol{z})$. Due to the nonlinearity, the original circle takes a different shape in $\boldsymbol{z}$ coordinates, as shown in Figure 3(b) and Figure 3(c). The kernel k_ϕ is subject to the same transformation, as it expresses the original kernel in the new coordinates $\boldsymbol{z}$. The topology of the kernel is preserved in the sense that its σ-covering area stays a connected set. If the areas of two neighboring centers do not overlap in $\boldsymbol{x}$, they are also guaranteed to not be overlapping in $\boldsymbol{z}$. This effect is visualized in Figure 3.

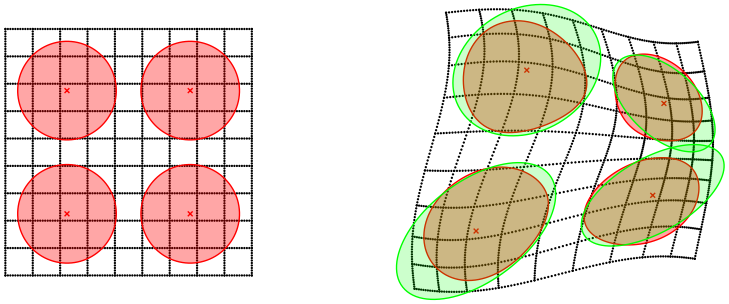
While the depicted kernels k_ϕ provide a good coverage of the transformed state-space, they do not take a Gaussian shape and therefore do not lend themselves to the invertibility bound derived in Theorem 4.1. To overcome this issue, it is possible to use well-known techniques, such as *unscented Kalman filter* (Wan and Van Der Merwe, 2000) (UKF), to estimate the covariance matrix of k_ϕ under the nonlinear propagation of ϕ. In particular, the UKF constructs an estimate $\boldsymbol{\Sigma}_{t+1}$ based on $\boldsymbol{z}_\sigma = \phi_t(\boldsymbol{x}_\sigma)$, which is conceptually depicted in Figure 4.

From Figure 4(b) it can be seen that the estimated covariance may be different for each center. Hence, we extend Theorem 4.1 to the case where different covariance matrices are used for each RBF.



(a) σ-covering of $k(\boldsymbol{x}, \boldsymbol{x}_c)$ (b) σ-covering of $k_\phi(\boldsymbol{z}, \boldsymbol{z}_c)$ (c) σ-covering of $k_{\hat{\phi}}(\boldsymbol{z}, \boldsymbol{z}_c)$

Figure 3: The state-space morphing induced by DD-RBFNs changes the shape of the kernel and its associated distance measure. While ϕ causes a mediocre deformation, $\hat{\phi}$ inflicts a large deformation for the bottom right σ-areas



(a) σ-covering of $k(\boldsymbol{x}, \boldsymbol{x}_c)$ (b) σ-covering of $k_\phi(\boldsymbol{z}, \boldsymbol{z}_c)$ and UKF estimates

Figure 4: Estimation of k_ϕ through an *unscented Kalman filter*. In Figure 4(a) the original $\boldsymbol{\Sigma}$ is shown in red for the space $\boldsymbol{x}$. In Figure 4(b), UKF estimates (green) of the morphed $\boldsymbol{\Sigma}$ (red) in the space $\boldsymbol{z}$ are shown. Depending on the UKF parameters and the morphing, the estimate can be an over- or underapproximation. Additionally, the UKF estimates are not guaranteed to be non-overlapping.

Proposition 3.1 (Multi covariance LDKs)**.** *Let $k_{\Sigma_i} : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}$ be defined as the multi-variate Gaussian kernel associated with covariance $\boldsymbol{\Sigma}_i \in S_{n+}$ and $\boldsymbol{H}_i := \boldsymbol{\Sigma}_i^{-1}$:*

$$k_{\Sigma_i}(\boldsymbol{x}, \boldsymbol{x}_i) = e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_i)^\top \boldsymbol{H}_i (\boldsymbol{x}-\boldsymbol{x}_i)} \tag{64}$$

which is centered around $\boldsymbol{x}_i \in \mathbb{R}^n$. Furthermore, let $\{\boldsymbol{x}_i\}_{i=1}^N$ be a set of N center points $\boldsymbol{x}_i \in \mathbb{R}^n$, each with associated covariance matrix $\boldsymbol{\Sigma}_i$. Define the mapping $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ as:

$$\phi(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{K}(\boldsymbol{x}) \tag{65}$$

where $\boldsymbol{W} \in \mathbb{R}^{n \times N}$, $\boldsymbol{K} \in \mathbb{R}^{N \times 1}$ and the i-th entry of $\boldsymbol{K}$ is given by $K_i(\boldsymbol{x}) = k_{\Sigma_i}(\boldsymbol{x}, \boldsymbol{x}_i)$. If each $w_{i,j}$ obeys

$$|w_{i,j}| < \frac{1}{nN \sum_{l=1}^n |Q_{j_i,l}| e^{-\frac{1}{2}} \sqrt{D_{j_i,i}}} := \rho(n, N, \boldsymbol{\Sigma}_j) \quad \forall i \in [1, \ldots, n], j \in [1, \ldots, N] \tag{66}$$

with $\boldsymbol{H}_j = \boldsymbol{Q}_j \boldsymbol{D}_j \boldsymbol{Q}_j^\top$ being the eigendecomposition of $\boldsymbol{H}_j$ then ϕ is a diffeomorphism.

Proof. Following the same derivations as for Theorem 4.1, we have that by (32) a bound for the partial derivative of k_{Σ_i} is given by:

$$\|\frac{\partial k_{\Sigma_i}}{\partial x_j}\| \leq \sum_{l=1}^{n} |Q_{i_{j,l}}| e^{-\frac{1}{2}} \sqrt{D_{i_{j,j}}} \tag{67}$$

where $D_{i_{j,j}}$ is the j-th diagonal entry of $\boldsymbol{D}_i$. Hence, the bound on the partial derivatives $\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}$ are given by:

$$\|\frac{\partial \psi_i(\boldsymbol{x})}{\partial x_j}\| \leq \sum_{i=1}^{N} |w_{i,j}| \|\frac{\partial k_{\Sigma_i}}{\partial x_j}\| \tag{68}$$

Restricting each element in the sum to the same absolute value we get

$$N|w_{i,j}| \|\frac{\partial k_{\Sigma_i}}{\partial x_j}\| < \frac{1}{n} \tag{69a}$$

$$|w_{i,j}| < \frac{1}{nN\|\frac{\partial k_{\Sigma_i}}{\partial x_j}\|} \tag{69b}$$

$$= \frac{1}{nN(\sum_{l=1}^{n} |Q_{i_{j,l}}| e^{-\frac{1}{2}} \sqrt{D_{i_{j,j}}})} \tag{69c}$$

which concludes the proof. □

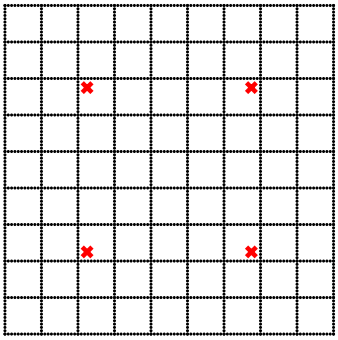
Proposition 3.1 enables the use of different covariance matrices for individual RBFs, which can for example be based on UKF estimates.

3.2 Center Point Morphing

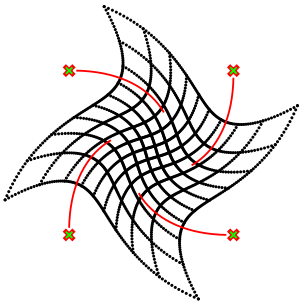
Another design choice for DD-RBFNs are the center points $\boldsymbol{x}_i$ for the kernel expansion. For non-parametric models building their structure from data, it is important to choose these points in such a way, that the data is sufficiently *supported*, e.g. in the σ-influence area around the centers. Since these points are also part of the state-space that is deformed by the model, they are also subject to that deformation and hence it is necessary to potentially consider choosing new center points after each transformation.

For the initial state-space $\boldsymbol{x}$, an obvious choice that guarantees support of the data is to choose each datapoint as an RBF center. However, since an increasing number of centers N leads to more restrictive parameter bounds ρ, one could also apply a k-means clustering approach (Lloyd, 1982) to find a good trade-off. For trajectory data, subsampling is also a possibility that can reduce N while keeping the data supported. A good center choice in $\boldsymbol{x}$ depends on the problem setup and the structure of the data, e.g. trajectories versus point clouds. In our setting however, we need to choose centers for each individual ϕ_t.

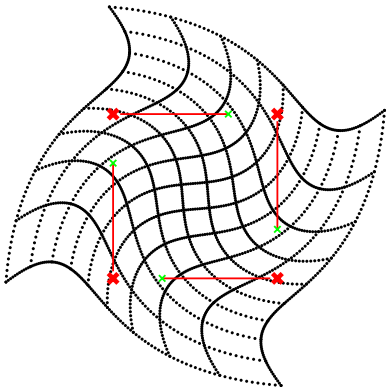
The simplest approach is to keep the centers fixed for all ϕ_t (e.g. $\boldsymbol{z}_{t,i} = \boldsymbol{x}_i$) and hence one does not need to propagate the RBF center positions during training. This comes with the disadvantage that one *loses control* of the transformed data point, once it moves far away from the support spanned by the centers. For a constant $\boldsymbol{W}$ for each layer, the induced morphing is shown in Figure 5(b). Shown in red are the trajectories of the datapoints, which depart from the fixed center position. Another approach is to simply transform the RBF center points and expand on the transformed centers for the next layer (e.g. $\boldsymbol{z}_{t,i} = \phi_t(\boldsymbol{x}_i)$), which is shown in Figure 5(c). This necessitates to propagate the centers during training, however, this does not cause any computational overhead during inference if the center positions are stored for each layer. If every datapoint defines an RBF center in $\boldsymbol{x}$, one is always able to *control* the movement of datapoints. Bijectivity guarantees that two centers will never move exactly on top of each other, however, they can come arbitrarily close to one another. In that case, the effective motion for the same expansion coefficients slows down because they start to influence each other if their distance decreases, shown in Figure 5(d).



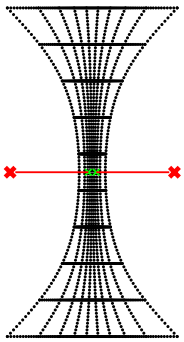
(a) Initial state-space $\boldsymbol{x}$



(b) morphing under fixed $\boldsymbol{W}$ and fixed centers



(c) morphing under fixed $\boldsymbol{W}$ and moving centers



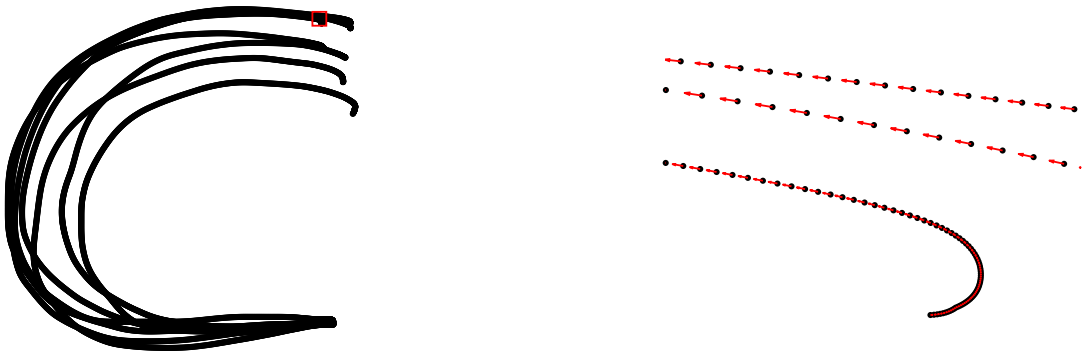
(d) approaching centers

Figure 5: Comparison of the two proposed center choices in morphed layers. Figure 5(a) depicts the initial sate-space $\boldsymbol{x}$ with four centers marked by red crosses. Figure 5(b) shows the morphing induced by repeatedly applying the same $\boldsymbol{W}$ coefficients and keeping the center positions fixed. Figure 5(c) shows the morphing under the same $\boldsymbol{W}$ and layer count, but propagating center positions. Final center positions are shown in green. Note that with fixed centers, control over the movement slowly degrades as the morphed original center point approaches another center, causing a curved path. This is not the case when center points are propagated. A similar effect can be seen if two centers are *pushed* together, as shown in Fig. 5(d).

4 ADDITIONAL EXPERIMENTAL DETAILS AND RESULTS

4.1 Data Processing and Influence of Hyperparameters

The dataset we use (Khansari-Zadeh and Billard, 2011) consists of motions for 30 different shapes sampled from test subjects while writing with a pen on a tablet. For each shape, the dataset gives 7 demonstrations with 1000 samples each. One example shape is shown in Fig. 6. The data is sampled from a noisy system (a human), which can cause contradicting velocities $\dot{\boldsymbol{x}}_i$ on states that are very close, as highlighted in Figure 6(b). This was already addressed by the authors in (Khansari-Zadeh and Billard, 2014), where they propose a *dataset correction step* to mitigate this issue. Hence, we remove a small number of points from the start and end of the trajectory used for training. Furthermore, without loss of generality, we normalize the shape data such that all coordinates of all datapoints are contained within the interval $[-1, 1]$. For our tests, we choose the first demonstration and subsample it to $N = 30$ datapoints after applying the aforementioned preprocessing steps. The resulting training data is shown in Figure 7(a). To solve the optimization problem, we use the general purpose optimizer *fmincon* which is part of the optimization toolbox of *MATLAB R2022b*.



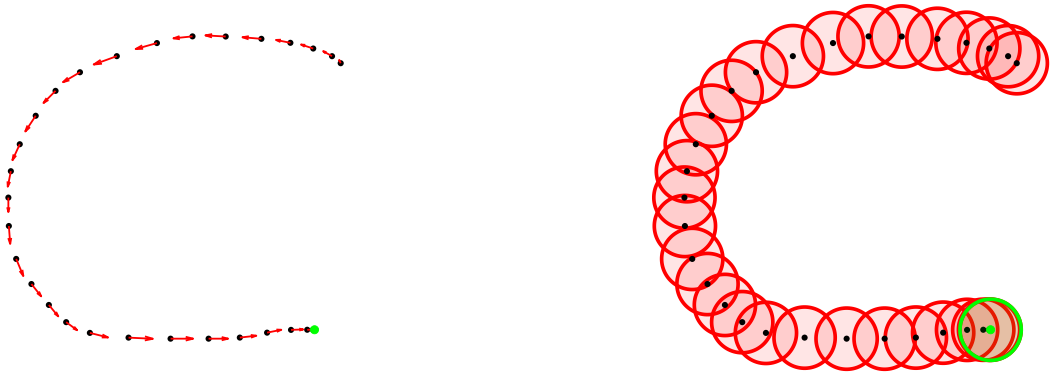
(a) Data $\boldsymbol{x}$ for shape *C-Shape*

(b) Zoomed in section marked in red in 6(a)

Figure 6: Example data from the human handwriting dataset for the *C-Shape*. In Figure 6(a), all 7 demonstrations, each consisting of 1000 datapoints are shown. The trajectories are sampled from noisy dynamical systems. One exemplary problematic section is highlighted in red in Figure 6(a) and enlarged in Figure 6(b). The direction of velocity $\dot{\boldsymbol{x}}_i$ is shown as a red arrow for each $\boldsymbol{x}_i$. The arrows have been scaled for the sake of visibility.

Employing the unscented Kalman filter to estimate $\boldsymbol{\Sigma}_t$ provides a heuristic-free way of choosing the covariance for each MPC iteration. However, we are still left with the choice for the initial covariance matrix. We initialize $\boldsymbol{\Sigma} = \boldsymbol{I}_n\sigma$ as a diagonal matrix with equal diagonal entries, since this represents a simple choice. For evaluating the covariance influence we choose a horizon of $H = 3$, which empirically showed a good trade-off between computational complexity and performance. The *UKF* parameters were chosen as $\alpha_{\text{UKF}} = 1$, $\beta_{\text{UKF}} = 2$ and $\kappa_{\text{UKF}} = 1$ (see (Wan and Van Der Merwe, 2000) for a detailed UKF parameter description). The optimization is carried out for 50 iterations, regardless of the first iteration where all constraints are satisfied.

In Figure 8, the results for two shapes are shown: the *Angle* shape and the *S* shape, each for a choice of $\sigma = 0.3$ and $\sigma = 0.05$. For both shapes, a smaller σ allows for finer *details* in the resulting Lyapunov function, while a larger σ influences a larger area around the datapoints. This is to be expected, as σ directly determines the influence area of each center. A smaller σ causes more restrictive bounds in the parameters $\boldsymbol{W}_t$. The larger covariance in this case hinders morphing finer details of the Lyapunov function to achieve a better fit in the narrow curves of the *S* shape. The *Angle* shape is a relatively simple shape, hence the larger covariance does not prohibit a good fit. In this case, a larger σ is beneficial, as a faster convergence is achieved due to the larger allowed step sizes for $\boldsymbol{W}_t$. The computation time for all 50 iterations was between 7 and 8 minutes for all



(a) Preprocessed dataset

(b) Centers and σ-areas for a diagonal $\mathbf{\Sigma}$

Figure 7: The preprocessed dataset for evaluation consists of a single demonstration normalized to the unitbox $[-1, 1]$, subsampled to 30 datapoints form the original dataset without including the first and last 50 datapoints. The resulting dataset is shown in Figure 7(a), where the black dots represent $\boldsymbol{x}^{(i)}$ and the red arrows depict $\dot{\boldsymbol{x}}^{(i)}$. The origin is marked in green. To support the data, the centers are chosen as the datapoints, as shown in Figure 7(b), where the red areas indicate the individual σ-influence areas for a diagonal choice of $\mathbf{\Sigma}$. One additional center is also positioned at the origin (shown in green) to help satisfy convergence constraint.

experiments. Note however, that feasibile solutions for each problems were obtained as early as in iteration 2 (*Angle* shape with $\sigma = 0.3$) and latest in iteration 30 (S shape with $\sigma = 0.05$).

4.2 Comparison Evaluation

Additionally, we compare our approach with the parametric approach of (Khansari-Zadeh and Billard, 2014) for six different shapes. We parameterize the weighted sum of asymmetric quadratic functions (WSAQF) with 3 components and let the optimization run for a maximum of 1000 iterations. For all shapes, the optimization finished before reaching the maximum allowed number of iterations. We again use $H = 3$ and a maximum number of iterations of 50. As initial guess, we use $V_b(\boldsymbol{x}) = 0.1\boldsymbol{x}^\top\boldsymbol{x}$ and choose $\mathbf{\Sigma} = 0.05\boldsymbol{I}_n$ as initial covariance for all centers. The UKF parameters are chosen identically to the previous experiment. The result is shown in Figure 9, where we show the results after 50 iterations for each shape. Both approaches are able to find valid Lyapunov functions that satisfy the constraints on all datapoints. However, WSAQF based Lyapunov functions are limited in expressiveness since they rely on polynomials. The WSAQF results are $\mathcal{C}_1$ functions which means that their gradient is discontinuous, as opposed to our diffeomorphic solution $V_{\boldsymbol{\phi}}$, which is guaranteed to be $\mathcal{C}_\infty$ everywhere except at the origin. Note that despite the simple initial guess V_b, our approach was able to find valid Lyapunov functions, which in addition possess a good alignment of the descent direction the dynamics at the data, as opposed to the WSAQF solution. Although this alignment is not necessary to fulfill the Lyapunov conditions, it increases robustness towards noise in the observations, as it maximizes the allowed deviation of the direction of $\dot{\boldsymbol{x}}^{(i)}$ without violation of the Lyapunov constraint.

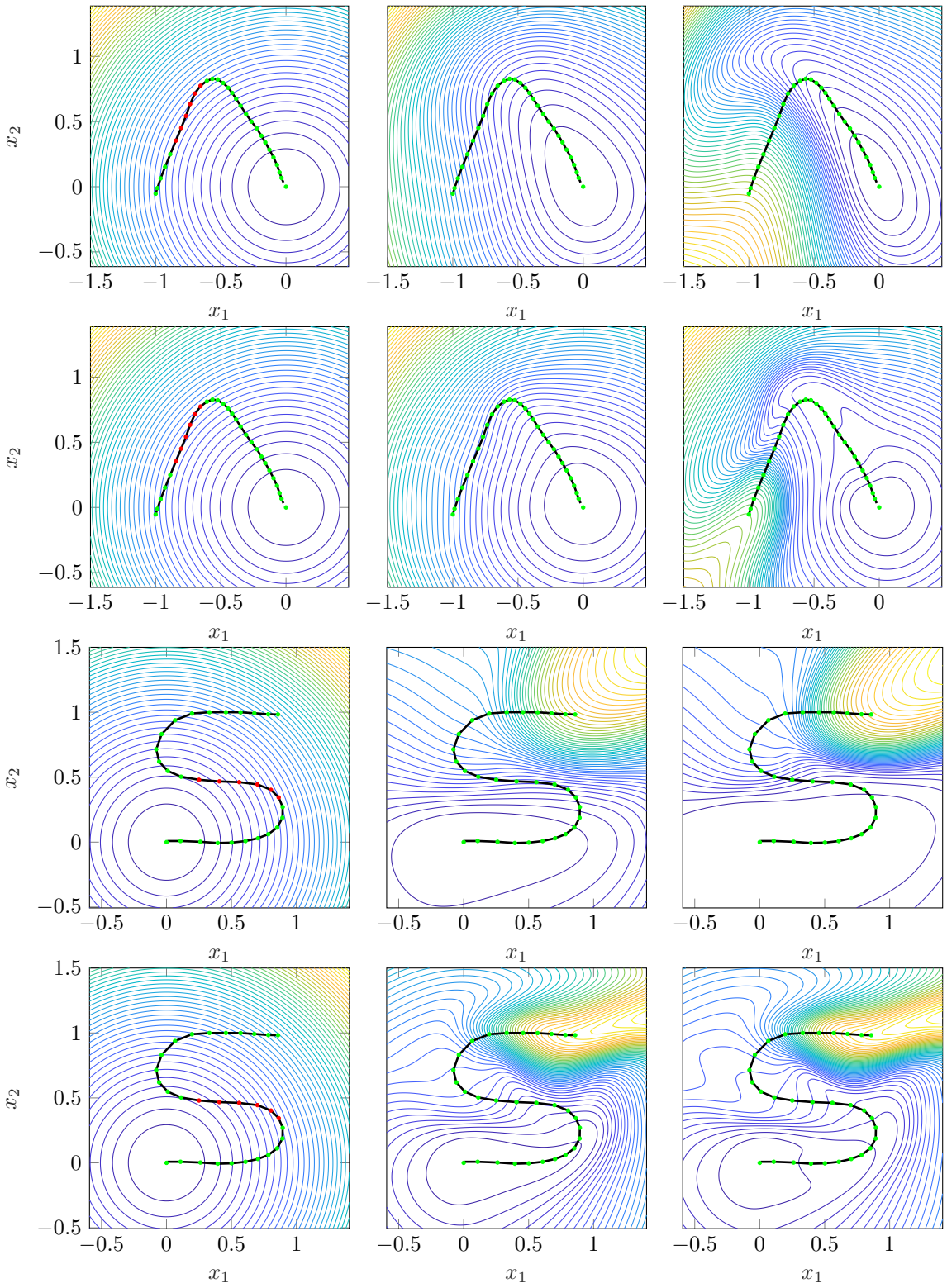
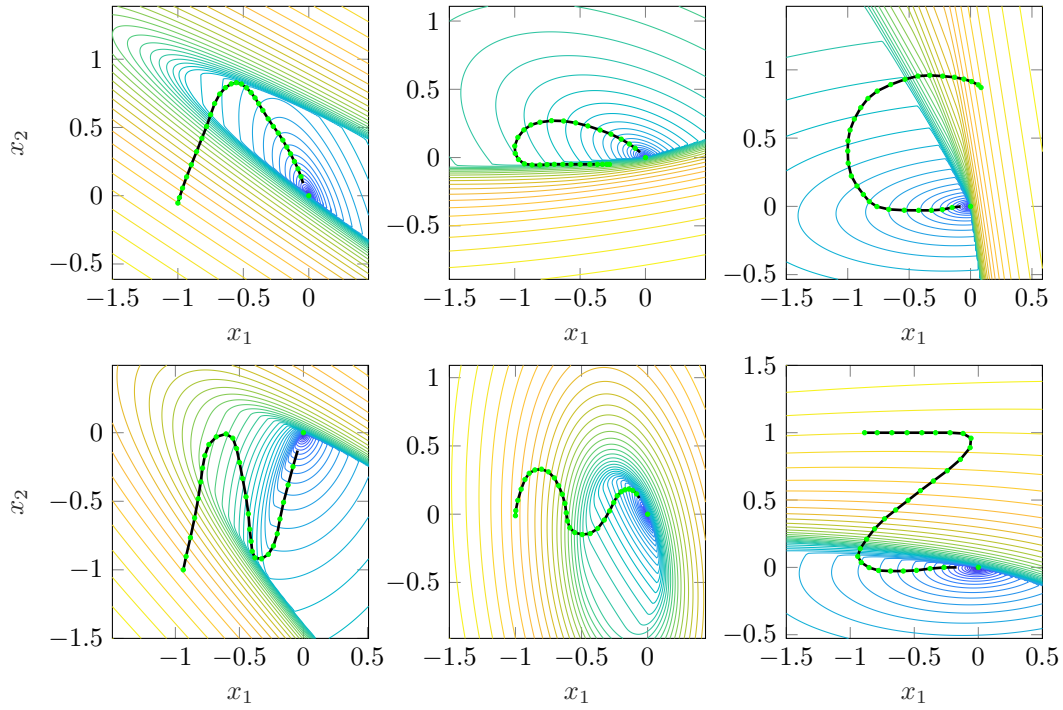
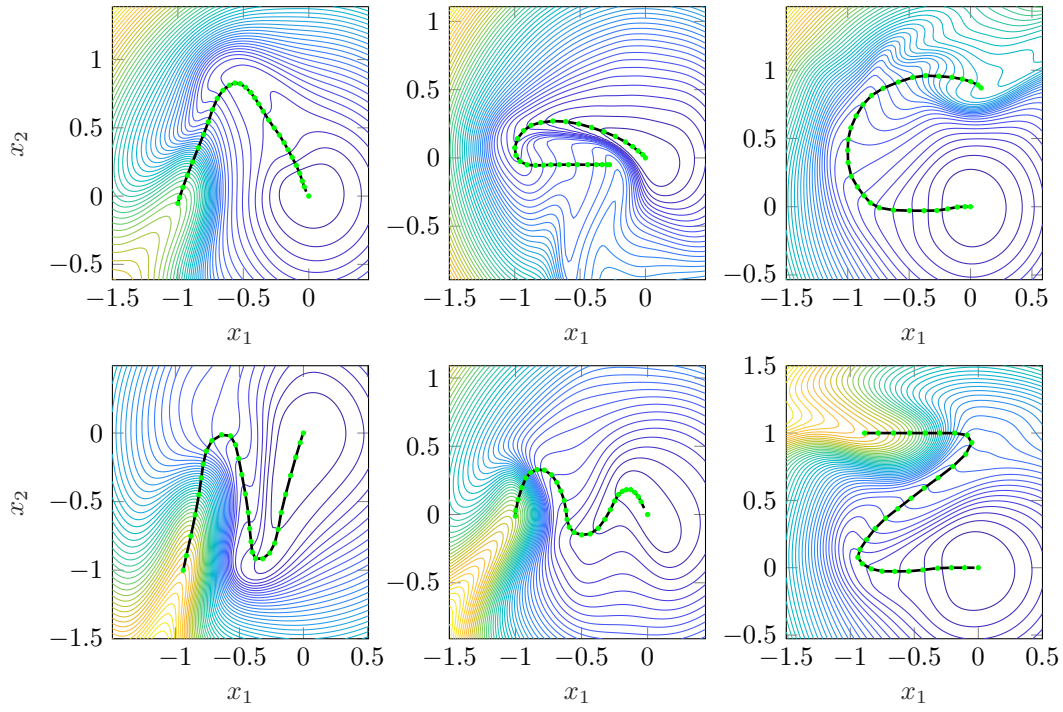


Figure 8: Comparison between two different choices for σ for two shapes. Datapoints on which the Lyapunov constraints are fulfilled are shown in green and datapoints that violate the constraint are shown in red. The columns represent (from left to right): (1) initial guess (2) first iteration with no constraint violations (3) last iteration. The rows represent (from top to bottom): (1) *Angle*-shape with $\sigma = 0.3$ (2) *Angle*-shape with $\sigma = 0.05$ (3) S-shape with $\sigma = 0.3$ (4) S-shape with $\sigma = 0.05$.



(a) WSAQF Lyapunov Functions (Khansari-Zadeh and Billard, 2014)



(b) Diffeomorphic Lyapunov Functions

Figure 9: Learned Lyapunov functions for 6 shapes of the handwriting dataset. Figure 9(a) shows the approach of (Khansari-Zadeh and Billard, 2014), which parameterizes Lyapunov functions through weighted sums of asymmetric quadratic functions (WSAQF). Figure 9(b) shows our approach, where an initial guess $V_b(\boldsymbol{x}) = 0.1\boldsymbol{x}^\top\boldsymbol{x}$ is altered through a diffeomorphism.

References

Pierre-Cyril Aubin-Frankowski and Zoltán Szabó. Hard shape-constrained kernel machines. *Advances in Neural Information Processing Systems*, 33:384–395, 2020.

Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. *Proceedings of the International Conference on Machine Learning*, pages 573–582, 2019.

Felix Berkenkamp, Riccardo Moriconi, Angela Schoellig, and Andreas Krause. Safe Learning of Regions of Attraction for Uncertain, Nonlinear Systems with Gaussian Processes. In *Proceedings of the IEEE Conference on Decision and Control*, pages 4661–4666, 2016.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in Neural Information Processing Systems*, 32, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In *International Conference on Learning Representations, Workshop Track Proceedings*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2016.

Eshant English, Matthias Kirchler, and Christoph Lippert. Kernelised normalising flows. In *International Conference on Learning Representations*, 2024.

Peter Giesl. *Construction of global Lyapunov functions using radial basis functions*, volume 1904. Springer, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Isao Ishikawa, Takeshi Teshima, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Universal approximation property of invertible neural networks. *Journal of Machine Learning Research*, 24(287):1–68, 2023.

Hassan K Khalil. *Nonlinear systems*. Prentice-Hall, Upper Saddle River, NJ, 3rd edition, 2002.

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

Seyed Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, 2021.

Armin Lederer and Sandra Hirche. Local Asymptotic Stability Analysis and Region of Attraction Estimation with Gaussian Processes. In *Proceedings of the IEEE Conference on Decision and Control*, 2019.

Qianxiao Li, Ting Lin, and Zuowei Shen. Deep learning via dynamical systems: An approximation perspective. *Journal of the European Mathematical Society*, 25(5):1671–1709, 2022.

S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Johan Lofberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, 2009.

Michael Lutter, Shie Mannor, Jan Peters, Dieter Fox, and Animesh Garg. Value iteration in continuous actions, states and time. *International Conference on Machine Learning*, pages 7224–7234, 2021.

Charles A Micchelli, Yuesheng Xu, and Haizhang Zhang. Universal kernels. *Journal of Machine Learning Research*, 7(12), 2006.

Alexander Ostrowski. Über die determinanten mit überwiegender hauptdiagonale. *Commentarii mathematici Helvetici*, 10:69–96, 1937.

A. Papachristodoulou and S. Prajna. On the construction of Lyapunov functions using the sum of squares decomposition. In *Proceedings of the IEEE Conference on Decision and Control*, volume 3, pages 3482–3487, 2002.

Pablo Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology, 2000.

Pablo Parrilo. Minimizing polynomial functions. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 2001.

Nicolas Perrin and Philipp Schlehuber-Caissier. Fast diffeomorphic matching to learn globally asymptotically stable nonlinear dynamical systems. *Systems & Control Letters*, 96:51–59, 2016.

Muhammad Asif Rana, Anqi Li, Dieter Fox, Byron Boots, Fabio Ramos, and Nathan Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. *Learning for Dynamics and Control*, pages 630–639, 2020.

Spencer M Richards, Felix Berkenkamp, and Andreas Krause. The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems. *Conference on Robot Learning*, pages 466–476, 2018.

Afroza Shirin, Manel Martínez-Ramón, and Rafael Fierro. Kernel machine to estimate a lyapunov function and region of attraction (roa) for nonlinear systems. *IEEE Access*, 2023.

Samuel Tesfazgi, Armin Lederer, and Sandra Hirche. Inverse reinforcement learning: A control lyapunov approach. In *IEEE Conference on Decision and Control*, pages 3627–3632, 2021.

Takeshi Teshima, Isao Ishikawa, Koichi Tojo, Kenta Oono, Masahiro Ikeda, and Masashi Sugiyama. Coupling-based invertible neural networks are universal diffeomorphism approximators. *Advances in Neural Information Processing Systems*, 33:3362–3373, 2020a.

Takeshi Teshima, Koichi Tojo, Masahiro Ikeda, Isao Ishikawa, and Kenta Oono. Universal approximation property of neural ordinary differential equations. In *Advances in Neural Information Processing Systems, Workshop on Differential Geometry meets Deep Learning*, 2020b.

Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 153–158, 2000.

Aiqing Zhu, Pengzhan Jin, Beibei Zhu, and Yifa Tang. On numerical integration in neural ordinary differential equations. *International Conference on Machine Learning*, pages 27527–27547, 2022.