Neural Learning of Vector Fields for Encoding Stable Dynamical Systems

A. Lemme, K. Neumann, R. F. Reinhart, J. J. Steil

Research Institute for Cognition and Robotics (CoR-Lab)
Bielefeld University - Germany

Abstract

The data-driven approximation of vector fields that encode dynamical systems is a persistently hard task in machine learning. If data is sparse and given in form of velocities derived from few trajectories only, state-space regions exists, where no information on the vector field and its induced dynamics is available. Generalization towards such regions is meaningful only if strong biases are introduced, for instance assumptions on global stability properties of the to-be-learned dynamics. We address this issue in a novel learning scheme that represents vector fields by means of neural networks, where asymptotic stability of the induced dynamics is explicitly enforced through utilizing knowledge from Lyapunov's stability theory, in a predefined workspace. The learning of vector fields is constrained through point-wise conditions, derived from a suitable Lyapunov function candidate, which is first adjusted towards the training data. We point out the significance of optimized Lyapunov function candidates and analyze the approach in a scenario where trajectories are learned and generalized from human handwriting motions. In addition, we demonstrate that learning from robotic data obtained by kinesthetic teaching of the humanoid robot iCub leads to robust motion generation.

Keywords: Extreme learning machine, neural network, vector fields, dynamical systems learning, motion generation, movement generation

1. Introduction

The approximation of vector fields from sparse data that represent dynamical systems, e.g. to encode quantitative flow visualization [1], optical flow in computer vision [2] or force fields in motor control [3, 4], is an important but also persistently hard task for learning algorithms. In recent work, vector fields were applied to learn and generate complex motions for robots [5, 6]. In such scenarios, training data typically consist of only few trajectories and thus leave many regions in the state space with no information of the desired vector field. Generalization towards regions subject to sparse sampling is challenging, because small errors in the approximation of the vector field can get amplified during integration and can lead to diverging behavior of the dynamical system.

Thus, a strong model bias is needed for generalization which has to be derived from prior knowledge about the underlying dynamics. In [7], a superposition of irrotational basis fields is used to approximate a variety of vector patterns, where it is assumed that the data originate from the gradient of a potential function. Kuroe and Kawakami introduced a combination of neural networks to reconstruct vector fields where prior knowledge of inherent vector field properties is used to enhance the accuracy [8, 9].

Wave propagation [10], which was introduced to dynamic path planning in [11], can be used to create potential fields, which in turn can be adapted to demonstrations [12]. Following the respective gradients then leads to an inherently stable dynamical system.

Motion generation methods, developed in computational imitation learning and programming by demonstration, appear to be promising in order to generalize to unseen areas in the workspace by providing stable solutions [13, 14, 15]. The stability of the motion is ensured by a linear spring damper system, which generates a straight line with a biologically plausible velocity profile. The shape is then induced by adding a perturbation force term. These time-dependent perturbations are learned by means of a mixture of Gaussian functions. The force term is suppressed at the end of the motion ensuring stability, because only the linear components drive the dynamical system. An alternative approach to the standard DMP approach is the task-parameterized Gaussian mixture model (TpGMM [16]), where the parameterization of the motion is variable. It models a second order dynamical system, which uses a probabilistic representation of the demonstrations. This representation can be parameterized either time-dependent, task-dependent or in combination.

For motion generation from vector fields, one promi-

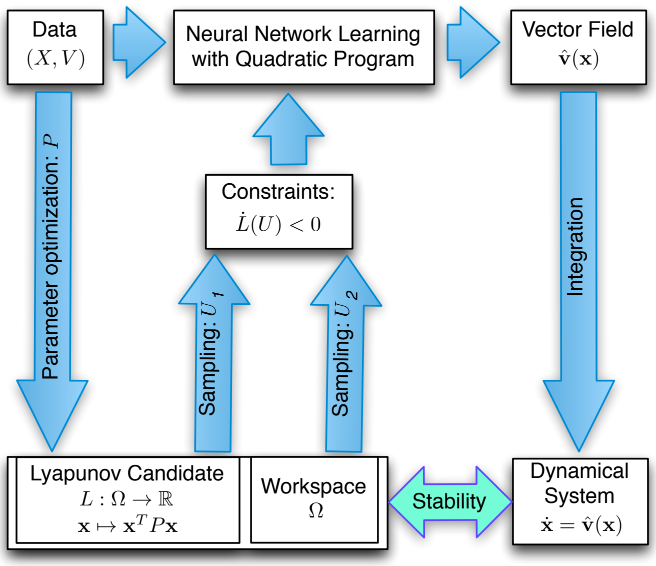
Email addresses: alemme@cor-lab.uni-bielefeld.de (A. Lemme), kneumann@cor-lab.uni-bielefeld.de (K. Neumann), freinhar@cor-lab.uni-bielefeld.de (R. F. Reinhart), jsteil@cor-lab.uni-bielefeld.de (J. J. Steil)

nent approach is the stable estimator of dynamical systems (SEDS [17]). This learning approach represents vector fields by a Gaussian mixture of linear dynamical systems. Learning is achieved by solving a nonlinear constrained optimization problem formulated as a quadratic program. The learned dynamical system then complies to a specific quadratic Lyapunov function. The main advantage of this method is that the learned dynamics are provably globally asymptotically stable. On the downside, the stability constraints may be too restrictive with respect to the motion that shall be learned. If the training data and the stability constraints contradict, accurate learning of the desired motion is prevented.

An extension of SEDS called SEDS-II was very recently published in [18] and implements less conservative stability conditions as compared to SEDS. This extension relies on a stabilization approach called control Lyapunov function derived from Artstein and Sontag's stability theory [19]. Such functions are used to stabilize nonlinear dynamical systems through online corrections at runtime and interfere with the learned dynamical system. This methodology can be applied in combination with any learning approach to represent the training data and leaves the stability issue to the online correction mechanism. However, the learning of dynamics that satisfy desired Lyapunov functions and guarantees stability without interfering with the data or requiring online corrections is so far only solved for special cases and remains difficult in case of using a dynamical systems represented by vector fields.

These issues are partly addressed in [20]. Here a neural network approach is used to learn from demonstrations and to generate motions for the humanoid robot iCub. The accuracy performance and the stability are addressed by two separately trained but superimposed neural networks. The first network approximates the data while the second network addresses stability by learning a velocity field, which implements a contraction towards the desired movement trajectory. However, the superposition of two networks seems complex for representing only one motion. Additionally, no guarantee for stable motion generation is given.

The contribution of this paper is the introduction of Lyapunov theory in neural networks learning for stable motion generation. Therefore, we extend the ideas recently published in [21] and propose a novel learning approach. This approach is based on the idea to represent time-independent vector fields in one neural network that lead to asymptotically stable dynamics in a predefined workspace. The learning is separated into three steps: First, construct a suitable Lyapunov candidate through parameter optimization towards the data. Second, use the constructed Lyapunov candidate to obtain inequalities constraints for learning. Third, add inequality constraints which ensure that the dynamics cannot leave a predefined region. The inequality constraints are implemented by a quadratic program, which minimizes the error between the training data and the output of the network. To keep the



Data
(X, V)
Neural Network Learning with Quadratic Program
Vector Field
$\hat{\mathbf{v}}(\mathbf{x})$
Constraints:
$\dot{L}(U) < 0$
Parameter optimization: P
Sampling: U_1
Sampling: U_2
Integration
Lyapunov Candidate
$L : \Omega \to \mathbb{R}$
$\mathbf{x} \mapsto \mathbf{x}^T P \mathbf{x}$
Workspace
Ω
Stability
Dynamical System
$\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$

Fig. 1: Schematic view of the proposed approach to learn vector fields. The learning is separated into two main steps: i) predefine a proper Lyapunov candidate through parameter optimization and ii) use this function to sample inequality constraints that are implemented by a quadratic program learning the data and iii) add constraints to restrict the motion to stay in the defined workspace. The resulting dynamical system approximates the data and is asymptotically stable in the defined workspace after learning.

amount of used constraints to a minimum, a sampling algorithm identifies problematic regions and adds constraints until the dynamical system is stabilized. Thus, the resulting vector field induces stable dynamics by construction. This approach is schematically illustrated in Fig. 1.

The reminder of this paper is organized as follows. In Sec. 2 we explain the theoretical basis of training neural networks with stability constraints. In Sec. 3 we show that the accuracy of the estimates is highly dependent on the applied Lyapunov candidate and show two different candidates in comparison. A rigorous analysis conducted here evaluates the relation between the regularization of the weights, the obtained errors, and the number of sampled constraints needed to implement stability. Additionally, it is demonstrated that the approach generates smooth and accurate motions in several experiments including also a kinesthetic teaching scenario with the humanoid robot iCub. Before we conclude this work in Sec. 5, we discuss the main features of this approach in Sec. 4.

2. Extreme Learning Machine for Estimation of Vector Fields

We consider trajectory data that are driven by time independent vector fields:

$$\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}) \quad , \mathbf{x} \in \Omega \quad , \tag{1}$$

where a state variable $\mathbf{x}(t) \in \Omega \subseteq \mathbb{R}^d$ at time $t \in \mathbb{R}$ with dimensionality d, defines a state trajectory in the workspace

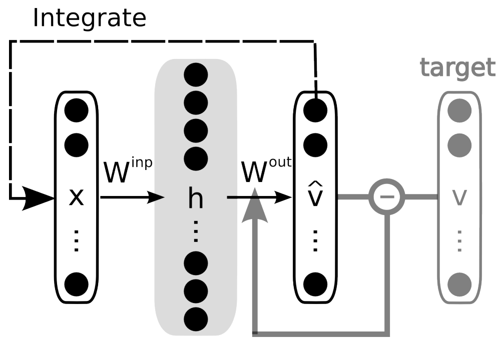


Fig. 2: ELM with its three layer structure used in an integration loop. Only the read-out weights are trained.

Ω. It is assumed that the vector field $\mathbf{v}(\mathbf{x})$ is nonlinear and continuous with a single asymptotically stable point attractor $\mathbf{x}^*$ with $\mathbf{v}(\mathbf{x}^*) = 0$ in Ω. The limit of each trajectory in Ω thus satisfies:

$$\lim_{t\to\infty} \mathbf{x}(t) = \mathbf{x}^* : \forall \mathbf{x}(0) \in \Omega \ . \tag{2}$$

The key question of this paper is how to learn $\mathbf{v}$ as a function of $\mathbf{x}$ by using demonstrations for training and ensure its asymptotic stability at target $\mathbf{x}^*$ in Ω. The estimate is denoted by $\hat{\mathbf{v}}$ in the following. The evolution of motion can then be computed by numerical integration of $\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$, where $\mathbf{x}(0) \in \Omega$ denotes the starting point of the motion.

Consider the neural architecture depicted in Fig. 2 for estimation of $\mathbf{v}$. The figure shows a single hidden layer feed-forward neural network: $\mathbf{x} \in \mathbb{R}^d$ denotes the input, $\mathbf{h} \in \mathbb{R}^R$ the hidden, and $\hat{\mathbf{v}} \in \mathbb{R}^d$ the output neurons. The input is connected to the hidden layer by the input matrix $W^{\mathbf{inp}} \in \mathbb{R}^{R\times d}$. The read-out matrix is given by $W^{\mathbf{out}} \in \mathbb{R}^{d\times R}$. For input $\mathbf{x}$, the output of the ith neuron is thus given by:

$$\hat{v}_i(\mathbf{x}) = \sum_{j=1}^{R} W^{\mathbf{out}}_{ij} f(\sum_{n=1}^{d} W^{\mathbf{inp}}_{jn} x_n + b_j) \ , \tag{3}$$

where $i = 1, \dots, d$ and b_j is the bias for neuron j. $f(x) = 1/(1 + \exp(-x))$ denotes the activation function applied to each neuron in the hidden layer. The components of the input matrix and the biases are drawn from a random distribution and remain fixed after initialization. This approach is called extreme learning machine (ELM) [24]. The following experiments reveal that this network architecture is particularly well suited for incorporation of stability constraints in the learning.

Let $D = (\mathbf{x}(k), \mathbf{v}(k)) : k = 1 \dots N_{\text{tr}}$ be the data set for training where N_{tr} is the number of samples in the data set. Supervised learning for ELMs is restricted to the read-out weights $W^{\mathbf{out}}$ and is usually done by ridge regression in a computationally cheap fashion:

$$W^{\mathbf{out}} = \left(HH^T + \varepsilon I\right)^{-1} HV^T \ , \tag{4}$$

where $H = (\mathbf{h}(\mathbf{x}(1)), \dots, \mathbf{h}(\mathbf{x}(N_{\text{tr}}))) : k = 1 \dots N_{\text{tr}}$ is a matrix harvesting the hidden states for each input $\mathbf{x}(k)$ in the training data set, I is the identity matrix, and $\varepsilon > 0$ is a regularization parameter.

2.1. Asymptotically Stable Dynamics

Learning a vector field from few training trajectories gives only sparse information about the shape of the vector field in the entire state space. Therefore, considerable need for generalization towards regions subject to sparse sampling is obvious. Recent studies have emphasized that stability of a predefined target plays an important role for the generalization ability in robotics tasks [17].

In order to analyze the stability of the dynamical system induced by the neural network, we recall the conditions for asymptotic stability of arbitrary dynamical systems found by Lyapunov: A dynamical system is asymptotically stable at fixpoint $\mathbf{x}^* \in \Omega$ in the compact and positive invariant region $\Omega \subset \mathbb{R}^d$ if there exists a continuous and continuously differentiable function $L : \Omega \to \mathbb{R}$

$$\begin{aligned} &(\mathbf{i}) \ \ L(\mathbf{x}^*) = 0 \ , \\ &(\mathbf{ii}) \ \ L(\mathbf{x}) > 0 : \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{x}^* \ , \\ &(\mathbf{iii}) \ \ \dot{L}(\mathbf{x}^*) = 0 \ , \\ &(\mathbf{iv}) \ \ \dot{L}(\mathbf{x}) < 0 : \forall \mathbf{x} \in \Omega, \mathbf{x} \neq \mathbf{x}^* \ . \end{aligned} \tag{5}$$

We assume that a function L satisfying condition (**i**)-(**iii**) is given. In order to obtain a learning algorithm for $W^{\mathbf{out}}$ that also respects condition (**iv**) of the Lyapunov function L, we analyze this condition by taking the time derivative of L and inserting the network's output $\hat{\mathbf{v}}_i(\mathbf{x})$:

$$\begin{aligned} \dot{L}(\mathbf{x}) &= \frac{\mathrm{d}}{\mathrm{d}t} L(\mathbf{x}) \\ &= (\nabla_{\mathbf{x}} L(\mathbf{x}))^T \cdot \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = (\nabla_{\mathbf{x}} L(\mathbf{x}))^T \cdot \hat{\mathbf{v}} \\ &= \sum_{i=1}^{d} (\nabla_{\mathbf{x}} L(\mathbf{x}))_i \cdot \sum_{j=1}^{R} W^{\mathbf{out}}_{ij} \cdot f_j(\mathbf{x}) < 0 \ . \end{aligned} \tag{6}$$

Note that $\dot{L}$ is linear in the output parameters $W^{\mathbf{out}}$ irrespective of the form of the Lyapunov function L. For a given point $\mathbf{x} \in \Omega$, condition (**iv**) in Eq. (6) defines a linear inequality constraint $\dot{L}(\mathbf{x}) < 0$ on the read-out parameters $W^{\mathbf{out}}$, which can be implemented by a quadratic programming scheme as first introduced for ELMs in [23].

Hence, the read-out weights are trained by solving a quadratic program optimizing $W^{\mathbf{out}}$ subject to a set of collected point constraints $U = \{\mathbf{u}^1, \dots, \mathbf{u}^{N_{\text{u}}}\} : \mathbf{u} \in \Omega$ and additional weight regularization:

$$\begin{aligned} W^{\mathbf{out}} &= \arg\min_{W} (\|W \cdot H(X) - V\|^2 + \varepsilon \|W\|^2) \\ &\text{subject to: } \dot{L}(U) < 0 \ , \end{aligned} \tag{7}$$

where the matrix $H(X) = (\mathbf{h}(\mathbf{x}(1)), \dots, \mathbf{h}(\mathbf{x}(N_{\text{tr}})))$ again collects the hidden layer states obtained from a given data set $D = (X, V)$ for inputs X and the corresponding output vectors V. The parameter ε again denotes the regularization parameter, and $\dot{L}(U) < 0$ defines the collected inequality constraints for a set of points U.

2.2. Positive Invariant Regions

The previous section introduced the implementation of inequalities, which are derived from a Lyapunov candidate L, at discrete points. These points are element of a predefined region Ω. However, stability according to condition (**iv**) is not stringently achieved in the entire space Ω without consideration of L. Even if the condition (**iv**) is valid in Ω, the state of the dynamical system $\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$ can possibly cross the border of Ω during numerical integration. In this case, stability cannot be guaranteed, since those parts of the input space outside Ω are not considered in the sampling process. Stability is, in principle, enforced in the largest level set region of the candidate function L that is completely intersecting with the sampling region Ω. Each initial point outside this region is potentially subject to divergence. The identification of such regions is difficult for arbitrary Lyapunov candidates.

We therefore introduce an additional constraint defined on the surface of Ω (denoted by $\partial\Omega$) which forces the learned dynamics to stay in the predefined region. It will be shown in the following sections that this technique also stabilizes the dynamics by implementing a positive invariant region Ω. This region can principally have arbitrary shapes, however, we only use hypercubes where each point $\mathbf{x} \in \Omega$ on the surface of this cube $\partial\Omega$ is mapped onto an uniquely defined normal vector $\mathbf{n}(\mathbf{x}) \in \mathbb{R}^d$ with $\|\mathbf{n}(\mathbf{x})\| = 1$ that point outward of the cube. The resulting constraints are expressed as a scalar product between the normal vector $\mathbf{n}(\mathbf{x})$ and the network's output $\hat{\mathbf{v}}(\mathbf{x})$:

$$L_\partial(\mathbf{x}) := \mathbf{n}(\mathbf{x})^T \cdot \hat{\mathbf{v}}(\mathbf{x}) \leq 0 : \mathbf{x} \in \Omega \ . \tag{8}$$

Note that the scalar product has the same form as Eq. (6) and is thus linear in $W^{\mathbf{out}}$. It forces the network's dynamics to stay inside the hypercube implementing a positive invariant region.

2.3. Sampling Strategy

We introduce the following sampling strategy in order to minimize the number of samples needed for generalization of the local constraints towards the continuous region Ω. The data set D for training and the region Ω where the constraints are supposed to be implemented are assumed to be given. As a first step ($k = 0$), the network is initialized and trained without any constraints (i.e. the sample matrices $U_i^k = U_i^0 = \emptyset$ are empty). In this case learning can be accomplished by ridge regression, which is the standard learning scheme for ELMs, see Eq. (4).

In the next step, N_C samples $\hat{U} = \{\hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \ldots, \hat{\mathbf{u}}^{N_\mathrm{C}}\}$ are randomly drawn from a uniform distribution in Ω and $\partial\Omega$, respectively. Afterwards, the number of samples ν_1 fulfilling (**iv**) of Lyapunov's conditions of asymptotic stability (see Eq. (6)) and the number of samples ν_2 satisfying Eq. (8) are determined.

The sampling algorithm stops if more or equal than p percent of these samples fulfill the continuous constraints, i.e. $\nu_i/N_C \geq p$. Otherwise, the most violating sample $\hat{\mathbf{u}}$ for each constraint is added to the respective sample pool $U_i^{k+1} = U_i^k \cup \hat{\mathbf{u}}$ to constitute the new sample set U_i^{k+1}. The most violating sample $\hat{\mathbf{u}}$ thus maximizes either $\dot{L}(\hat{\mathbf{u}})$ or $L_\partial(\hat{\mathbf{u}})$ with respect to the Lyapunov candidate L. The obtained set of samples is then used for training. A pseudo code of the learning procedure is provided in Alg. 1.

Algorithm 1 Sampling Strategy

```
Require: data set $D$, region $\Omega$, counter $k = 0$, sample pools $U_i^k = \emptyset$, and ELM $\hat{\mathbf{v}}$ trained with $D$
  repeat
    draw $N_C$ samples $\hat{U} = \{\hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \ldots, \hat{\mathbf{u}}^{N_\mathrm{C}}\}$
    $\nu_1$ = no. of samples in $\hat{U}$ fulfilling condition (iv)
    $\nu_2$ = no. of samples in $\hat{U}$ fulfilling Eq. (8)
    if $p > \frac{\nu_1}{N_\mathrm{C}}$ then $U_1^{k+1} = U_1^k \cup \arg\max_{\mathbf{u} \in \hat{U}} \dot{L}(\mathbf{u})$
    if $p > \frac{\nu_2}{N_\mathrm{C}}$ then $U_2^{k+1} = U_2^k \cup \arg\max_{\mathbf{u} \in \hat{U}} L_\partial(\mathbf{u})$
    train ELM with $D$ and $U_i^{k+1}$
  until $p \leq \frac{\nu_i}{N_\mathrm{C}} : \forall i$
```

Note that the constraints at points U and the input samples X of the training data are not the same. For learning according to Eq. (7), it remains to select a set of samples U in addition to the training inputs X. It was already shown in [23] that a well-chosen sampling of points U is sufficient for generalization of the incorporated discrete constraints towards a continuous region Ω and that stability can be proven ex-post after learning. This ex-post verification algorithm is based on Taylor approximation of the learned function.

This is possible since the neural network method allows analytical differentiation and exploits the linear dependency on the learning parameters in order to use a worst case approximation by means of Taylor polynomials. The approximation of the constraint surface is, however, computationally costly and should only be applied if required. It is also important to note that, in absence of the verification processes, stability is only given with a certain probability which is related to the value of p. We use $p = 1$ throughout the paper. Note further that sampling and ex-post verification is only possible in finite regions and can only enforce local asymptotic stability.

2.4. Learning Lyapunov Candidates

The previous section introduced the learning process and showed how to use a Lyapunov candidate function in order to construct stable dynamics. This raises the important question what kind of Lyapunov candidate is desirable?

Fig. 3 illustrates the impact of different Lyapunov candidate functions on the approximation ability of the proposed neural network approach for an exemplary motion. The first row of Fig. 3 shows the used Lyapunov candidates

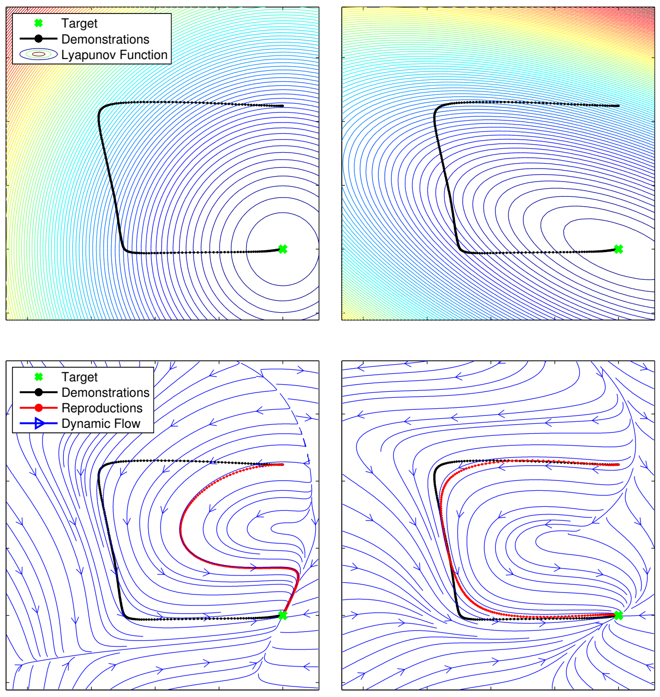


Fig. 3: The impact of different Lyapunov candidate functions on the learning of dynamical systems from data. The first row contains the respective Lyapunov candidates while the second row depicts the learned dynamics. The vector field obtained for a quadratic Lyapunov candidate $L = \mathbf{x}^2$ (first column) and the result of learning when using a proper Lyapunov candidate (second column).

while the second row visualizes the obtained vector fields. The first column of Fig. 3 shows the results for a simple candidate function $L_q = \mathbf{x}^T\mathbf{x}$. This Lyapunov candidate is not well suited to approximate the example motion (black line), because it is not possible to find any vector field that approximates the data and simultaneously satisfies the constraints given by the Lyapunov candidate function L_q. Part of the training samples $(\mathbf{x}(k), \mathbf{v}(k))$ are violated by L_q, where violation means that the angle between the negative gradient of the Lyapunov candidate at point $\mathbf{x}$ and the velocity of the data sample $\mathbf{v}$ is bigger than 90°:

$$\sphericalangle(-\nabla L(\mathbf{x}(k)), \mathbf{v}(k)) > 90^\circ \Leftrightarrow (\nabla L(\mathbf{x}(k)))^T \cdot \mathbf{v}(k) > 0 \tag{9}$$

Note that the derivation in Eq. (6) shows that this is in contradiction to condition (**iv**) of Lyapunov's theorem. Fig. 3 (second column) shows the learning result for a more suitable Lyapunov candidate L. The conflict between the training data and the Lyapunov candidate L is resolved which leads to an accurate dynamical estimate of this motion. This example illustrates the importance of selecting a proper Lyapunov candidate L that matches the target motion. Otherwise, a mismatch between data approximation and stability constraints remains and the reproduced trajectory does not follow the target motion (see first example), because the constraints overrule the training data in the learning process. The second example shows that a suitable Lyapunov candidate resolves this mismatch such that the learned dynamics closely follow the demonstrations.

In the following, the violation of the training data by a Lyapunov candidate is formalized and it is shown how a parameterized Lyapunov candidate can be fitted to the data in order to minimize this violation.

The violation of the training data D by a Lyapunov candidate $L(\mathbf{x})$ can be formalized by generalizing condition (**iv**) of Lyapunov's theorem and defining the following measure:

$$M(L) = \frac{1}{N_{\mathrm{tr}}} \sum_{k=1}^{N_{\mathrm{tr}}} \Theta \left[(\nabla L(\mathbf{x}(k)))^T \cdot \mathbf{v}(k) \right] \quad , \tag{10}$$

where Θ is the ramp function. Only samples $(\mathbf{x}(k), \mathbf{v}(k))$ where the scalar product between $\nabla L(\mathbf{x}(k))$ and $\mathbf{v}(k)$ is positive are counted in M.

We exemplary consider parameterized Lyapunov candidates $L(\mathbf{x})$ of the following quadratic form in the experiments similar to the Lyapunov functions in [25]:

$$L_P(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \cdot P \cdot (\mathbf{x} - \mathbf{x}^*) \quad . \tag{11}$$

This Lyapunov function L_P will be employed for learning to implement asymptotic stability at the known fixpoint $\mathbf{x}^*$. Note that (**i**) - (**iii**) describe the general form of a Lyapunov function and are fulfilled by Eq. (11) if P is positive definite and symmetric.

We use the measure M to adapt P in order to prevent a strong violation of the training data D. If a data set D is given, P can be chosen as:

$$P = \arg\min_{P \in \mathcal{P}} M(L) \quad \text{with}$$
$$M(L) = \frac{1}{N_{\mathrm{tr}}} \sum_{k=1}^{N_{\mathrm{tr}}} \Theta \left[(\mathbf{x}^i(k))^T \cdot P \cdot \mathbf{v}^i(k) \right] \quad , \tag{12}$$

The minimization operator on the right hand side of Eq. (12) can be formulated as a nonlinear program. We use successive quadratic programming based on quasi-Newton methods for optimization of Eq. (12) [26]. To prevent an infinite stretching of P through minimization of M we restrict the eigenvalues λ_i of P to $\alpha \leq \lambda_i \leq 1, \forall i = 1 \ldots K$, where $\alpha = 0.1$ in the following experiments.

Since Eq. (6), which is used to enforce asymptotic stability, is linear in the read-out parameters $W^{\mathbf{out}}$ and at the same time independent of the form of the respective Lyapunov candidate L, it is also possible to use more complex Lyapunov candidates of higher complexity. A recently published study in [22] shows how to learn flexible Lyapunov candidates that can be used to enforce asymptotic stability for complex and robust motion generation in robotic imitation learning. We nevertheless use Lyapunov candidates of quadratic form for the sake of clarity in this paper.

3. Learning Stable Point-to-Point Motions

In order to analyze the impact of the learning with constraints, we perform experiments where the networks learn from human-demonstrated handwriting motions collected for movement primitive learning [27][1]. We first consider a single motion from this data set which is composed of three S-like trajectories with 250 samples each and the end-point located at the origin (see Fig. 4). The trajectories are in millimeters and get scaled to decimeters for the following experiments.

The ELM is initialized with $R = 100$ neurons in the hidden layer. The biases b_i and components of $W^{\mathbf{inp}}$ are initialized randomly drawn from the uniform distribution on $[-1, 1]$. The regularization parameter is set to $\varepsilon = 10^{-4}$. The parameterized Lyapunov candidate function is adapted to the training data according to Eq. (12). The set of constraints U_1 consists of samples drawn from the region $\Omega = [-0.9, 2.5] \times [-2.0, 3.4]$, which covers the relevant region of the task space, while the sample pool U_2 is located on the surface of Ω. The sampling strategy used $N_C = 1000$ and $p = 1$ to find regions that violate the constraints.

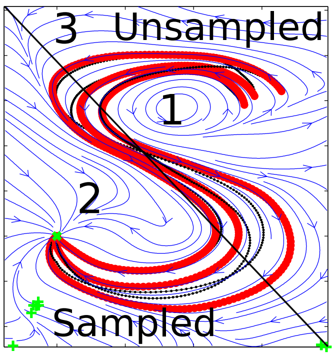


Fig. 5: The impact of sampling stability constraints for the learning only in one half of the workspace.

Fig. 4 (left) illustrates an example of learning the S-like trajectories by an ELM without the usage of stability constraints. In the areas close to the demonstrations, the trajectories converge to an attractor next to the target. In other regions of the space, they either converge to spurious attractors or diverge. In contrast, Fig. 4 (center) and Fig. 4 (right) show the same network setups trained with the stabilization method using Eq. (11) with metric P chosen according to Eq. (12). The center plot shows the learning results with consideration of the constraints drawn from the Lyapunov candidate function without application of the border constraints given by Eq. (8). The right plot visualizes the additional enforcement of the positive invariance of the workspace Ω by sampling it's surface $\partial\Omega$ according to Eq. (8). The generated trajectories converge to the target, because the constrained learning process enforces asymptotic stability at target $\mathbf{x}^*$. This ensures that the target is reached when starting from any point in the workspace Ω. The adopted Lyapunov function candidate enables the accurate modeling of the dynamics. Furthermore, Fig. 4 demonstrates that the sampling of the workspace border ensures the convergence of the state to the target attractor irrespective of the starting point in the workspace Ω. Note that only a few constraints are necessary to incorporate stability (green and red '+' in Fig. 4). The sampling algorithm adds constraints only in critical regions which results in a non-uniform distribution of constraint samples in U_1 and U_2.

3.1. Systematic Evaluation

In the following a systematic evaluation of the new stability mechanism is conducted using four performance measures. The first measure is the root mean square error $E_{\text{tr}} = \sqrt{\frac{1}{N_{\text{tr}}} \sum_k \|\mathbf{v}(k) - \hat{\mathbf{v}}(\mathbf{x}(k))\|^2}$ evaluated on the training data, which quantifies the ability to approximate the training data. The second and third measure quantify the stability of the dynamical system numerically. For these measures, we simulate the motion generation by choosing $N_{\text{s}} = 100$ starting points uniformly drawn in Ω and iterate the dynamical system for $N_{\max}$ time steps with step size $\Delta_t = 0.1$. If the end-point $\mathbf{x}(N_{\max})$ of the reproduced trajectory is in the vicinity of the desired attractor $\mathbf{x}^* = \mathbf{0}$, the learned dynamics are rated as converged to the target (i.e. $\|\mathbf{x}(N_{\max}) - \mathbf{x}^*\| < \delta = 1$), otherwise as diverged. The distance dist $= \|\mathbf{x}(N_{\max}) - \mathbf{x}^*\|$ of the end-point to the attractor for each converged trajectory constitutes the second measure. The amount of the converged trajectories S divided by the number of starting points N_{s} quantifies the stability of the system as a third measure. In addition, the CPU Time used for training conducted on a x86_64 linux machine with at least 4 GB of memory and a 2+ GHz intel processor in matlab R2010a[2] is recorded. All results are averaged over 10 different network initializations.

The results of the experiments for networks with and without stabilization for different regularization parameters ε are shown in Tab. 1. The performance of implementing the desired attractor is given by the Euclidean distance (dist). The ratio S/N_{s} shows how many motions generated according to the discretization of the dynamics are close to the desired target. Note that $S/N_{\text{s}} = 1$ holds for all constrained networks. The results show that the stability ratio increases with increasing regularization ε for networks trained without constraints. This reveals a trade-off between stability and accuracy for the unconstrained

[1]The data set was collected at the LASA institute at EPFL.

[2]MATLAB. Natick, Massachusetts: The MathWorks Inc., 2010.

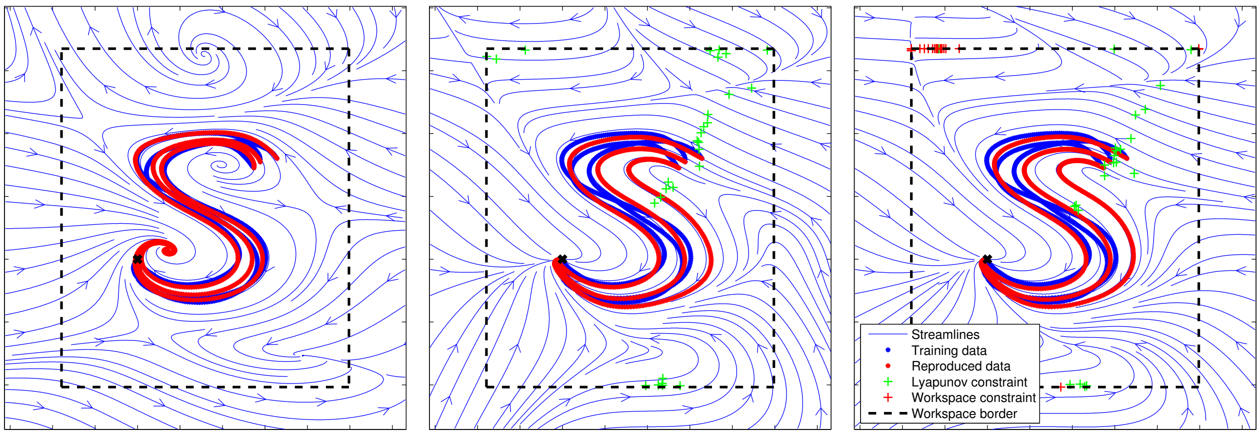


Fig. 4: Impact of incorporating asymptotic stability into the learning. The figure shows the dynamics of a network trained without stability constraints (left), the dynamics of a network trained solely with stability constraints derived from the Lyapunov function (center), and the dynamics of the same network trained with constraints for stabilization and restriction to the workspace (right).

without constraints				
$\lg \varepsilon$	dist	S/N_s	E_tr	CPU Time [s]
-8	.431±.067	.686±.053	.21±.0006	0.22 ± 0.001
-6	.431±.018	.813±.031	.22±.0008	0.21 ± 0.001
-4	.401±.013	.917±.053	.24±.0027	0.22 ± 0.001
-2	.382±.010	.965±.044	.30±.0028	0.20 ± 0.001

with constraints						
$\lg \varepsilon$	dist	S/N_s	E_tr	$\#\ U_1$	$\#\ U_2$	CPU Time [s]
-8	.049±.014	1	.26 ± .0008	52.1 ± 6.1	47.8 ± 8.4	638.6 ± 184.2
-6	.043±.016	1	.28 ± .0011	32.7 ± 5.9	26.4 ± 4.7	249.1 ± 80.4
-4	.055±.014	1	.31 ± .0014	23.3 ± 4.4	12.4 ± 4.6	139.2 ± 44.9
-2	.020±.012	1	.38 ± .0015	8.5 ± 0.7	13.6 ± 3.6	59.4 ± 18.6

Tab. 1: Network learning with and without constraints compared for different regularization parameters ε averaged over 10 trails. Additional information about the learning time and the amount of derived constraints from the Lyapunov function ($\#U_1$) and from the workspace constraint ($\#U_2$) is given. S/N_s is the ratio of converged motions.

networks. For the explicitly stabilized ELMs, the trade-off is resolved, because all networks converge to the target independent of the regularization while the network can still approximate the training data accurately for less regularization. Thus, the target attractor is imprinted very close to the desired attractor (cf. dist in Tab. 1). The successive implementation of the constraints is computationally more costly (this is strongly depending on the number of constraint samples $\#U_1$ and $\#U_2$ that are used for training) but can be reduced by increasing regularization. The main reason for this effect is that the generalization of the constraint becomes easier with lower model complexity (larger ε). Note that the positioning of the samples is non-uniform, because the samples are only placed in regions that are more prone to instability with respect to the Lyapunov candidate and the border of the workspace Ω.

Fig. 5 demonstrates the locality of the proposed approach, please compare to Fig. 4. In this experiment, only the region below the black stroke is subject to constraint sampling and no border constraint is considered. Three regions in this experiment are interesting: the spiral repeller in region 1 is still active since the data close to this region forces the dynamics to this behavior. The spiral and spurious attractor in region 2, that appears in the left plot of Fig. 4, is deleted due to the sampling of constraints. Region 3 shows that the stability constraint can be generalized towards regions not subject to sampling, since no data is present in this region, thanks to the strong model bias induced by the stability constraints. This example shows that the locality of the approach is no disadvantage. Instead, locality adds high flexibility to the approach: The application of different constraints in selected regions is possible.

3.2. Results on the LASA Data Set

This section extends the evaluation of the proposed approach to experiments performed on the entire data set of hand-writing motions from [27]. Fig. 6 shows an investigation of several measures on each of the twenty motion shapes separately. The plot in the upper part of the figure shows the value for M defined in Eq. (10) for each motion of the data set. It is revealed that the simple quadratic

Lyapunov candidate $L_q = \mathbf{x}^T\mathbf{x}$ strongly violates the training data for some of the motions. This violation can be relaxed by introduction of the matrix P. Some of the motions are not violated to a high degree anymore after optimization of L_P - e.g. the sharp-C shaped motion. These results support the idea that optimized Lyapunov candidates enhance the class of accurately learnable motions. Fig. 6 (second row) shows the training error obtained in this experiment. It is observable that the training error is high for motions which also produce a high value of violation M. This is expected because the networks are forced to implement the Lyapunov candidate. Thus, if the Lyapunov candidate violates the training data, the network dynamics will not accurately reproduce the demonstrations. It is also revealed that the training error is decreasing for lower regularization parameters ε. The additional experiments show that the regularization parameter is still important to tune. However, the stability of the resulting estimate is mainly influenced by the sampling scheme. The plot in the third row of Fig. 6 shows the number of samples $\#U_1$ drawn form the Lyapunov candidate in the learning phase until implementation of asymptotic stability. The third plot in the figure shows the number of samples $\#U_2$ drawn on the border of Ω to enforce this region to be positive invariant. The last plot in Fig. 6 contains the time needed for learning. These plots summarize the results for networks with different regularization parameters. Two results can be deduced from this experiment. First, there is a clear correlation between difficult motions - in the sense of M - and the number of applied samples: difficult motions need more samples for implementation of stability. Second, networks with stronger regularization need less constraints to enforce stability. Note that the number of samples for some motions could be reduced significantly by increasing the regularization parameter from $\varepsilon = 10^{-8}$ to $\varepsilon = 10^{-2}$. This is advantageous, because the time for learning strongly depends on the required number of constraints which is related to the number of iterations of the quadratic program. Higher regularization leads obviously to a reduced need for constraint sampling and therefore significantly accelerates the learning process. Regularization is an appropriate means to tackle the curse of dimensionality of the constraint sampling in higher dimensional spaces Ω.

Fig. 7 shows the motions learned from human demonstrations (black) from the LASA benchmark data set with the reproduced trajectories (red) and the surrounding vector field (blue). Note that all networks produce stable and accurate motions which converge to the given target point attractor due to the combination of the optimized Lyapunov candidate and constrained learning.

3.3. Generating Motions for iCub

The data considered in the previous experiments are 2D trajectories. In this section, we show that the approach is also suited for 3D trajectories recorded in a programming by demonstrations scenario with the humanoid robot iCub [28]. Humanoid robots are typically designed to solve service tasks in environments where a high flexibility is required. Reliable and robust adaptability by means of learning is thus a prerequisite for such systems.

The experimental setting is illustrated in Fig. 8. A human tutor physically guides iCub's left arm using a recently established force control on the robot [29, 30]. The tutor can thereby actively move the joints of the arm to place the end-effector at a desired position. Beginning on the right side of the workspace, the tutor moves the arm in a half circle towards the left side of the workspace were the motion stops. This procedure is repeated five times, where the position of the end-effector of the robot is recorded. Each of the five recorded trajectories has be-

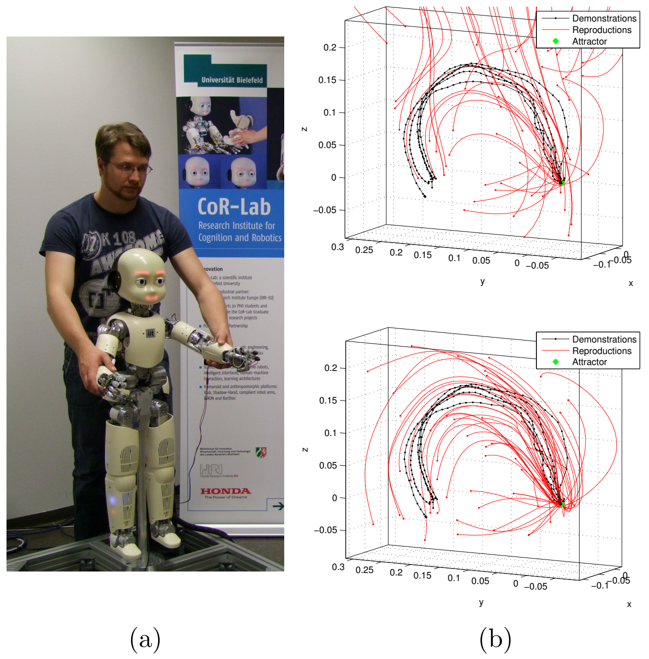


Fig. 8: Kinesthetic teaching of the humanoid robot iCub (a) and learned motions for iCub (b). Dynamical estimate with demonstrations and reproductions without stabilization (b, top) and with stabilization through application of a Lyapunov candidate (b, bottom).

tween $N_{\text{traj}} = 125$ and $N_{\text{traj}} = 167$ discrete data points, which are used for learning without further preprocessing. The hidden layer of the networks estimating the motion by means of a dynamical system consists of $R = 100$ neurons and the regression parameter is $\epsilon = 10^{-5}$ in this experiment. The networks' weights and biases are initialized randomly from a uniform distribution in the interval $[-10, 10]$ due to the small ranges of the motion in meters. The results of the experiment are visualized in Fig. 8. The figure shows the impact of the explicit stabilization on the estimation of the dynamics by visualizing the reproduction of the motions by the robot iCub. The estimation of the networks trained without explicit stabilization is prone to instabilities even if the motions stay close to the demonstrations in the beginning, see Fig. 8

Important parameters for learning	
L	Suitable Lyapunov candidate to draw constraints from
Ω	Workspace to implement constraints
N_C	Amount of samples drawn from region Ω used in Alg. 1
ε	Regression parameter
R	Number of neurons in the hidden layer
W^{inp}	Initialization range of the input weight matrix (input scaling)
a, b	Initialization ranges of the nonlinearity see Eq. (4)

Tab. 2: Important parameters which need to be set by the user before network learning. The last set of parameters are needed already for the standard ELM approach. The first part of the table gives the new parameters important for inducing stability.

(upper right). The networks trained with the Lyapunov candidate from Eq. (11) achieve a good performance and stability, see Fig. 8 (lower right).

4. Discussion

The methods described above ensure the resulting dynamical system to be asymptotically stable in a predefined workspace describing a local region of the state space. The main advantage of the proposed approach is the inherently decoupled process of finding a proper Lyapunov candidate and implementing the dynamical systems itself in one neural network. In principle it allows the implementation of arbitrary Lyapunov candidates and can thus lead to flexible, robust, and accurate dynamical systems learned from data. However, to reach a good approximation ability of the training data, it is essential that the Lyapunov function does not contradict the training data. Therefore, finding a suitable Lyapunov candidate is one of the key steps in this approach, because a quadratic energy function is for many motions too restrictive and results in poor reproduction of the targeted dynamics. The idea of using Lyapunov stability constraints and incorporate them into the model is not new. In the SEDS approach [17] a quadratic Lyapunov function is used, but arbitrary Lyapunov functions cannot be exploited. The SEDS-II approach [18] tackles this issue by correcting the dynamical system online according to a Lyapunov function adapted to the data. The main difference to the proposed approach is that in SEDS-II the adapted Lyapunov function is not incorporated into the vector field representation itself, but is applied in parallel. Our approach is capable of learning complex dynamics in a neural network which satisfy arbitrary Lyapunov functions without an additional correction mechanism.

The learning process proposed in this paper uses constraints that are generated by a sampling strategy. It identifies potential unstable regions and uses them to constructively enforce stability into the learned dynamics. The locality of the approach allows to choose desired regions for the implementation of constraints. This flexibility is advantageous in cases where a variety of different stability constraints has to be satisfied. They can appear in several bounded regions in possibly lower dimensional subregions of the input space. Also, multiple attractors can be implemented with this approach. Although, the locality of this approach prevents a direct stability guarantee for a previously defined basin of attraction, asymptotic stability can be effectively proven ex-post if needed. The prove exploits the linear dependency on the learning parameters and uses a worst case approximation by means of Taylor polynomials (see [23]).

An important difference of related approaches to the presented approach is the guarantee of *global* asymptotic stability. The main advantage of global stability is the missing requirement of a predefined local workspace. However, the relevant region of the task space, in which the dynamical system is applied to generalize and perform motions, is in many applications bounded and known a priori. In case of a strong perturbation, pushing the state of the dynamical system outside of the stable region, a high-level controller can react and apply counter measures (e.g. emergency stop of the robot or switch to another motion).

The presented results show that the required number of samples and thus the time for training can be significantly reduced by increasing the regularization parameter of the ELM. This becomes important if we consider the scalability of this approach, because scaling the approach to higher dimensions depends on the number of samples needed to implement stability.

In Tab. 2 we summarize the important parameters, which are set before the learning process starts. Note that only three additional parameters are added (top) to the standard ELM parameters (bottom). The Lyapunov candidate L is the most important variable to reach good performance in approximating the training data. The size of the workspace is mostly task dependent and needs to be determined accordingly. The amount of samples N_C drawn from this region is a crucial parameter with respect to computation time. If too few samples are drawn, the probability to implement asymptotic stability decreases, because regions for which the learner violates the stability constraints maybe overlooked. Thus, this parameter needs to be linked to the size of the workspace Ω.

5. Conclusion

In this paper we proposed a novel scheme for learning vector fields from sparse data by using Lyapunov's stability theory. Due to the strong bias applied to the model, induced by stability assumptions, save and stable motion generation in the workspaces becomes feasible. The learning scheme incorporates linear inequality constraints, derived from a Lyapunov candidate function, which are efficiently implemented by quadratic programming at discrete points of the workspace. Despite the fact that the constraints are only satisfied at discrete points by construction, they can be generalized towards a continuous region by using a simple sampling strategy also presented in this work.

In addition, the accuracy of the estimated dynamical systems is enhanced by using a parameterized Lyapunov candidate, which is adapted to the data. In fact, the proposed scheme allows the integration of more complex Lyapunov candidates [22]. Further experiments show the impact of regularization regarding performance and the number of samples needed for learning stable dynamics. Finally, we demonstrate that the learning scheme can cope with data obtained by kinesthetic teaching of the humanoid robot iCub and that the learned dynamical system is able to generate smooth and accurate reproductions in a three-dimensional task space.

Acknowledgment. All authors contributed to research design and paper writing. A.L., K.N. and F.R. conducted the core research on the learning method. A.L., F.R. and K.N. conducted the robotic experiments. A.L. and F.R. are funded by FP7 under GA. No. 248311-AMARSi. K.N. and F.R. are funded by the German BMBF within the Intelligent Technical Systems Leading-Edge Cluster Competition and managed by the Project Management Agency Karlsruhe (PTKA).

[1] M. Gharib, F. Pereira, D. Dabiri, and D. Modarress. Quantitative flow visualization. *Annals of the New York Academy of Sciences*, 972(1):1–9, 2002.

[2] A. Verri and T. Poggio. Motion field and optical flow: qualitative properties. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):490–498, 1989.

[3] Ferdinando A Mussa-Ivaldi and Simon F Giszter. Vector field approximation: a computational paradigm for motor control and learning. *Biological Cybernetics*, 67(6):491–500, 1992.

[4] Simon F. Giszter and William J. Kargo. Modeling of dynamic controls in the frog wiping reflex: Force-field level controls. *Neurocomputing*, 38(0):1239–1247, 2001.

[5] Jochen Heinzmann and Alexander Zelinsky. Quantitative safety guarantees for physical human-robot interaction. *The International Journal of Robotics Research*, 22(7-8):479–504, 2003.

[6] B. Corteville, E. Aertbelien, H. Bruyninckx, J. De Schutter, and H. Van Brussel. Human-inspired robot assistant for fast point-to-point movements. In *IEEE International Conference on Robotics and Automation*, pp. 3639–3644, 2007.

[7] Ferdinando A Mussa-Ivaldi. From basis functions to basis fields: vector field approximation from sparse data. *Biological Cybernetics*, 67(6):479–489, 1992.

[8] Yasuaki Kuroe and Hajimu Kawakami. Vector field approximation by model inclusive learning of neural networks. In *Proc. ICANN*, pp. 717–726, 2007.

[9] Yasuaki Kuroe and Hajimu Kawakami. Estimation method of motion fields from images by model inclusive learning of neural networks. In *Proc. ICANN*, pp. 673–683, 2009.

[10] J. Barraquand, B. Langlois and J-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[11] Dmitry V. Lebedev, Jochen J. Steil and Helge J. Ritter. The dynamic wave expansion neural network model for robot motion planning in time-varying environments. *Neural Networks*, 18(3):267–285, 2005.

[12] J. V.Go'mez; D.lvarez; S.Garrido; L.Moreno. Kinesthetic Teaching via Fast Marching Square. In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 1305–1310, 2012.

[13] Auke Ijspeert, Jun Nakanishi, Peter Pastor, Heiko Hoffmann and Stefan Schaal. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25:328–373, 2013.

[14] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 358(1431):537–547, 2003.

[15] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. *Handbook of Robotics*, chapter Robot Programming by Demonstration. MIT Press, 2008.

[16] S. Calinon, T. Alizadeh, and D. G. Caldwell. Improving extrapolation capability of task-parameterized movement models. In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 610–616, 2013.

[17] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[18] S. M. Khansari-Zadeh. *A Dynamical System-based Approach to Modeling Stable Robot Control Policies via Imitation Learning.* PhD thesis, EPFL, 2012.

[19] E D Sontag. A universal construction of artstein's theorem on nonlinear stabilization. *Systems & control letters*, 13(2):117–123, 1989.

[20] F.R. Reinhart, A. Lemme, and J.J. Steil. Representation and generalization of bi-manual skills from kinesthetic teaching. In *Proc. Humanoids*, pp. 560–567, 2012.

[21] A. Lemme, K. Neumann, F.R. Reinhart, and J.J. Steil. Neurally imprinted stable vector fields. In *Proc. ESANN*, pp. 327–332, 2013.

[22] K. Neumann, A. Lemme, and J.J. Steil. Neural Learning of Stable Dynamical Systems based on Data-Driven Lyapunov Candidates. In *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 1216–1222, 2013.

[23] K. Neumann and J. Steil. Reliable integration of continuous constraints into extreme learning machines. *Journal of Uncertainty, Fuzziness & Knowledge-Based Systems*, 21(supp02):35–50, 2013.

[24] Z. Huang and C. Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(1–3):489–501, 2006.

[25] Laurent E. Ghaoui, Eric Feron, and Vendataramanan Balakrishnan. *Linear Matrix Inequalities in System & Control Theory*, vol. 15. Society for Industrial and Applied Mathematics (SIAM), 1994.

[26] M. S. Bazaraaa, H. Sherali, and C. Shetty. *Nonlinear programming: Theory and Algorithms*. Ed. John Wiley & Sons, 2006.

[27] S.M. Khansari-Zadeh. `http://www.amarsi-project.eu/open-source`, 2012.

[28] N.G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A.J. Ijspeert, M.C. Carrozza, et al. iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, 21(10):1151–1175, 2007.

[29] M. Fumagalli, S. Ivaldi, M. Randazzo, and F. Nori. Force control for iCub. `http://eris.liralab.it/wiki/Force_Control`, 2011.

[30] A. Parmiggiani, M. Randazzo, L. Natale, G. Metta, and G. Sandini. Joint torque sensing for the upper-body of the iCub humanoid robot. In *Proc. Humanoids*, pp. 15–20, 2009.

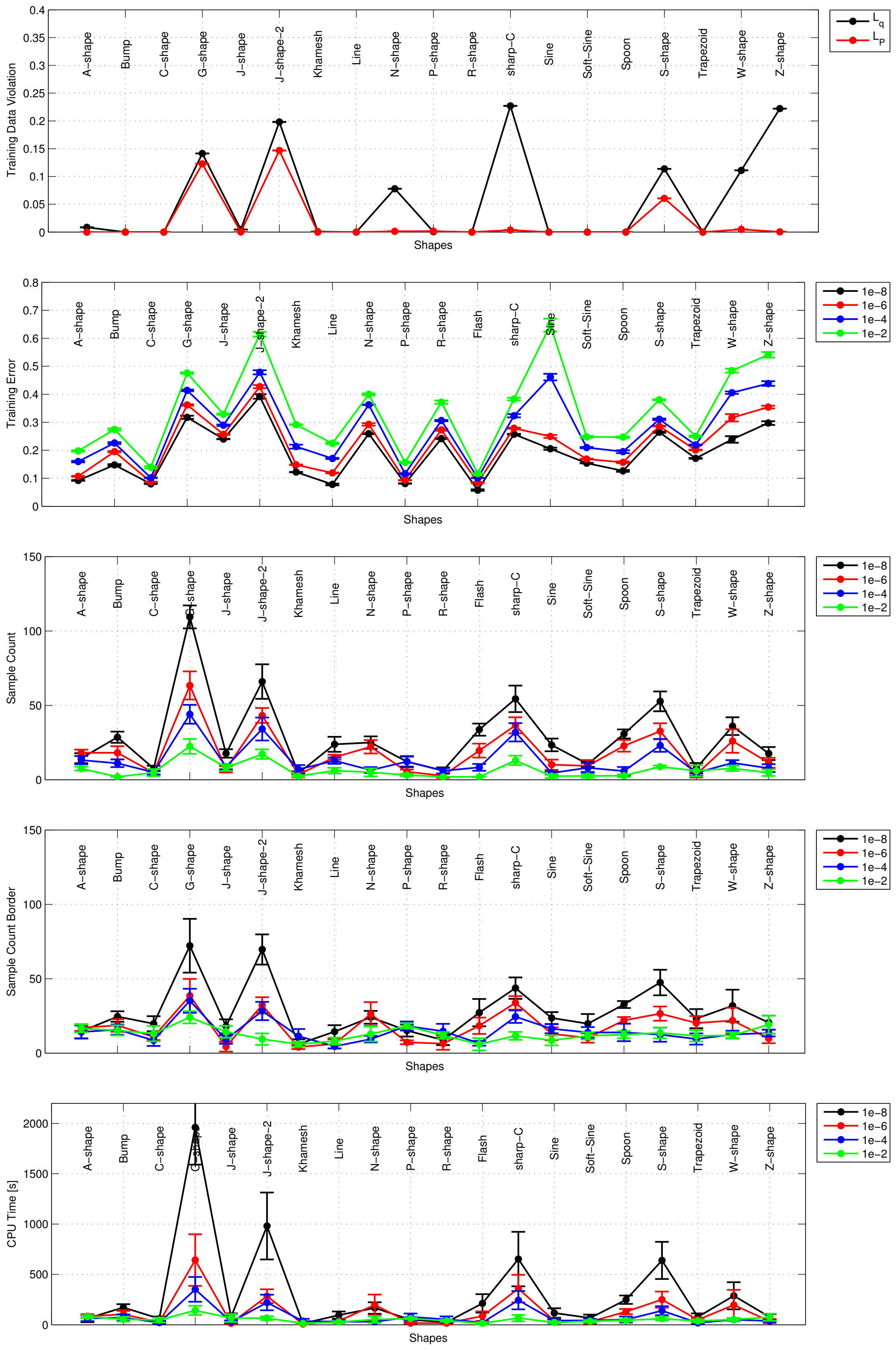


Fig. 6: Investigation on 20 motion shapes of the LASA data set. Evaluation of the violation measure M on each motion (first row) comparing the quadratic Lyapunov candidate L_q with the parameterized version L_p. The following plots show the results using L_p and different regularizations ε. The second row show the the training errors. The next two rows show the amount of samples used for learning without and with border constraints. At last we show the CPU time during learning when using L_p with both types of constraints.

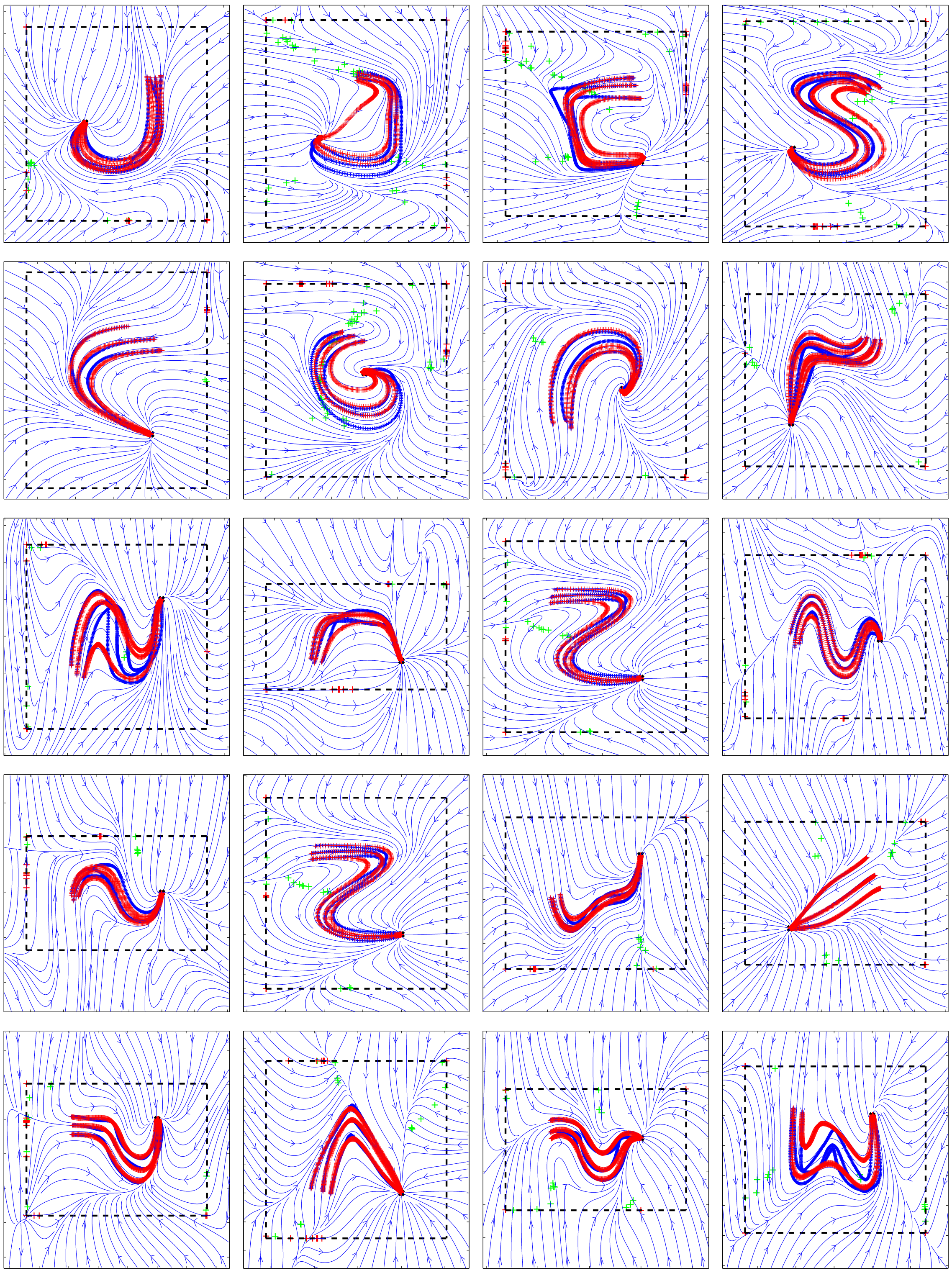


Fig. 7: Dynamical system estimates for 20 motions from the LASA benchmark data set. The black lines represent the training data and the red lines are the reproduced motions. The learned vector fields are depicted in blue. The green and red '+' are the constraints in U_1 and U_2 after the final iteration of Algorithm 1. The borders of regions Ω are shown by dashed lines.