Neural Learning of Stable Dynamical Systems based on Data-Driven Lyapunov Candidates

Klaus Neumann[1], Andre Lemme[1] and Jochen J. Steil[1]

***Abstract*— Nonlinear dynamical systems are a promising representation to learn complex robot movements. Besides their undoubted modeling power, it is of major importance that such systems work in a stable manner. We therefore present a neural learning scheme that estimates stable dynamical systems from demonstrations based on a two-stage process: first, a data-driven Lyapunov function candidate is estimated. Second, stability is incorporated by means of a novel method to respect local constraints in the neural learning. We show in two experiments that this method is capable of learning stable dynamics while simultaneously sustaining the accuracy of the estimate and robustly generates complex movements.**

I. INTRODUCTION

Nonlinear dynamical systems appear to be one of the most promising candidates as computational basis for exploitation of flexible motor capabilities featured by modern humanoid robots [1], [2], [3]. For instance, point-to-point movements modeled by autonomous dynamical systems can provide a library of basic building blocks called movement primitives [4] which are very successfully applied to generate movements in a variety of manipulation tasks [5], [6].

Recently, several studies emphasized that stability plays an important role for such tasks besides the undoubted modeling power of dynamical systems [7]. Reinhart et al. [8], e.g., used a neural network approach to generate movements for the humanoid robot iCub [9]. The performance and the stability are addressed by two separately trained but superpositioned networks. Another widely known approach is called dynamic movement primitives (DMP) [10] which is a very successful technique to generate motions with dynamical systems. It provides a very accurate non-linear estimate of a given trajectory robust to perturbations while ensuring global stability at the target attractor. The stability is enforced through a stable linear dynamical system which suppresses a nonlinear perturbation at the end of the motion. The smooth switch from non-linear to linear dynamics is controlled by a phase variable. The phase variable can be seen as external stabilizer which in return distorts the temporal pattern of the dynamics. This leads to the inherent inability of DMP to generalize well outside the demonstrated trajectory [11].

This directly motivates methods which are capable of generalizing to unseen areas. Such methods are time-independent and thus preserve the spatio-temporal pattern. They became of special interest by focusing on the "what to imitate" problem [12], [13].

[1]The authors are with the Research Institute for Cognition and Robotics (CoR-Lab), Faculty of Technology, Bielefeld University. Universitätsstr. 25, 33615 Bielefeld, Germany. (kneumann,alemme,jsteil)@cor-lab.de

However, the incorporation of stability into non-linear dynamical systems is inherently difficult due to their high complexity, in particular, if systems are supposed to approximate given data sets. On the one hand an integration through conservative stability constraints often leads to a poor reproduction performance. On the other hand a too strong focus on the accurate reproduction of the data lead to a weak robustness to perturbations which might end in divergence. The trade-off between stability and performance is often resolved for the benefit of stability in return for losing accuracy. This resolution is undesired if the complex parts of underlying dynamics are of major interest. How to imprint stability while simultaneously sustaining the complexity of the dynamical system is thus a key question to tackle.

Several solutions to that question have been developed in previous years. The common basis of these approaches is Lyapunov's theory which is a powerful tool to analyze the stability of such systems. One statement is that asymptotic stability of a fixed-point is equivalent to the existence of a Lyapunov function. However, most of the approaches for estimation of dynamical systems only deal with simple and data-independent Lyapunov functions which makes the finding of a satisfactory solution to the trade-off difficult. An appealing approach that aims at ensuring robustness to temporal perturbations by learning vector fields from demonstrations is the stable estimator of dynamical systems (SEDS) [13]. It is based on a mixture of Gaussian functions and respects correlation across several dimensions. It is shown, that SEDS is globally asymptotically stable but restricted to approximate contractive dynamics corresponding to a quadratic Lyapunov function [13]. An extension of SEDS called SEDS-II was very recently published in [14] and implements less conservative stability conditions as compared to SEDS. However, this extension relies on an explicit stabilization approach called control Lyapunov function derived from Artstein and Sontag's stability theory [15]. Such functions are used to stabilize nonlinear dynamical systems through online corrections at runtime and interfere with the dynamical system. They cannot directly guarantee stability of the learned system and therefore are not directly comparable to the other approaches which provide stable estimates by construction. However, the learning of dynamics that satisfy desired Lyapunov functions which guarantees stability without interfering with the data is so far only solved for special cases and remains difficult [14].

The application of standard candidates, i.e. quadratic Lyapunov candidates as in Fig. 1 (left) or matrix parametrized candidates visualized in Fig. 1 (center) is not satisfactory

when the tasks demands an appropriate complexity. In theory, much more complex functions are possible and also desired, see Fig. 1 (right), but there is no constructive or analytic way to derive such a candidate function directly.

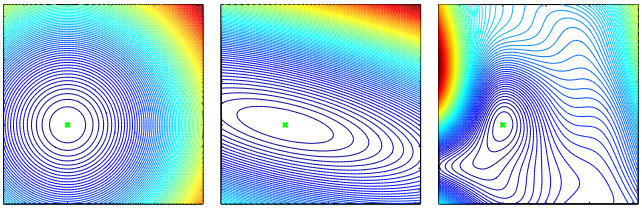

Fig. 1. Level sets of three different Lyapunov candidates $L = x^2$, $L = x^T Px$, and L =?, but which one to apply for learning?

We extend the ideas recently published in [16]. This learning approach is based on the idea to learn time-independent vector fields while the learning is separated into two main steps: i) predefine a proper Lyapunov candidate and ii) use this function for sampling inequality constraints during the learning process in order to obtain a stable dynamical estimate. This separation has several advantages: it allows for the application of arbitrary Lyapunov candidates and leads to flexible and accurate solutions. Note, that the locality of this approach prevents a direct stability guarantee, but the used neural approach allows analytical differentiation and it can be constructively and effectively proven ex-post if needed, see [17]. Very similar to the already mentioned methods, Lemme et al. [16] only proposes the implementation of *simple* Lyapunov candidates, compare to Fig. 1 (left, center). For this reason our contribution is to propose a method to learn *highly flexible* Lyapunov candidates from data. We then compare our method to the state of the art and show in a detailed evaluation that its incorporation strongly reduces the trade-off between stability and accuracy, which allows robust and flexible movement generation for robotics.

II. Neurally-Imprinted Stable Vector Fields

This section briefly introduces a technique to implement asymptotic stability into neural networks via sampling based on the recently published paper by Lemme et al. [16]. Stability is implemented by sampling linear constraints obtained from a Lyapunov candidate. The paper suggested to use quadratic functions for stabilization which are in fact relatively simple when considering complex robotic scenarios and thus raises the question: what Lyapunov candidate to apply instead?

A. Problem Statement

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 $t \in \mathbb{R}$ defines a state trajectory. It is assumed that the vector field $\mathbf{v} : \Omega \to \Omega$ is nonlinear, continuous, and continuously differentiable 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:

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

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.

B. Neural Network for Estimating $\mathbf{v}(\mathbf{x})$

Consider the neural architecture depicted in Fig. 2 for estimation of $\mathbf{v}$. The figure shows a single hidden layer feed-forward neural network called extreme learning machine (ELM) [18] comprising three layers of neurons: $\mathbf{x} \in \mathbb{R}^d$ 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 through 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:

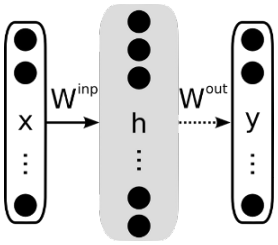

Fig. 2. Extreme learning machine with three layer structure. Only the read-out weights are trained supervised.

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

where $i = 1, \ldots, d$, b_j is the bias for neuron j, and $f(x) = \frac{1}{1+e^{-x}}$ denotes the Fermi 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.

Let $D = (\mathbf{x}^i(k), \mathbf{v}^i(k)) : i = 1 \ldots N_{\text{traj}}, k = 1 \ldots N^i$ be the data set for training where N_{traj} is the number of demonstrations. N_{ds} denotes the overall number of samples in D. Supervised learning is restricted to the read-out weights $W^{\mathbf{out}}$ and is done by ridge regression in computationally cheap fashion. This well known procedure defines the read-out weights as:

$$W^{\mathbf{out}} = \arg\min_{W} (\|W \cdot H(X) - V\|^2 + \varepsilon_{\text{RR}} \|W\|^2) \tag{4}$$

where ε_{RR} is the regularization parameter, $H(X)$ is the matrix collecting the hidden layer states obtained for inputs X, and V is the matrix collecting the corresponding target velocities. The evolution of motion can be computed by numerical integration of $\dot{\mathbf{x}} = \hat{\mathbf{v}}(\mathbf{x})$, where $\mathbf{x}(0) \in \mathbb{R}^d$ denotes the starting point.

C. Asymptotic Stability and Lyapunov's Stability Theory

Dynamical systems implemented by neural networks have been advocated as a powerful means to model robot motions [8], [19], but "the complexity of training these networks to obtain stable attractor landscapes, however, has prevented a widespread application so far" [10]. Learning a vector field

from a few training trajectories gives only sparse information on the shape of the entire vector field. There is thus a desperate need for generalization to spatial regions where no training data reside.

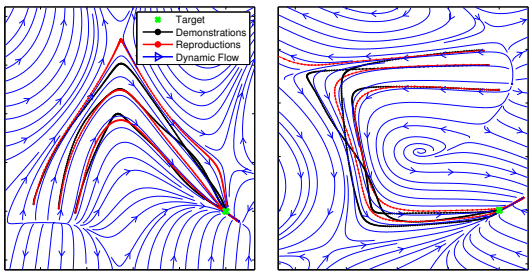

Fig. 3. Unstable estimation of dynamical systems through demonstrations. Data were taken from the LASA data set: A-shape (left) and sharp-C (right).

Fig. 3 illustrates two examples of unstable estimation of a nonlinear dynamical system using an ELM. The networks were each trained with three trajectories obtained from a human demonstrator. The left figure shows that the reproduced trajectories converge to a spurious attractor in the immediate vicinity of the target or diverge. The right figure demonstrates that the motion either converges to other spurious attractors far away from the target or completely diverge from it. This illustration shows that learning this attractor landscape is particularly hard without explicit consideration of stability.

In order to stabilize the dynamical systems estimate $\hat{\mathbf{v}}$, we recall the conditions for asymptotic stability of arbitrary dynamical systems found by Lyapunov: a dynamical system is asymptotically stable at fixed-point $\mathbf{x}^* \in \Omega$ in the compact and positive invariant region $\Omega \subset \mathbb{R}^d$ if there exists a continuous and continuously differentiable function $L : \Omega \to \mathbb{R}$ which satisfies the following conditions:

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

We assume that we have a function L that satisfies condition (**i**)-(**iii**) and call it: *Lyapunov candidate*. Such a function can be used to obtain a learning algorithm that also satisfies condition (**iv**) w.r.t. the estimated dynamics $\hat{\mathbf{v}}$. We therefore re-write condition (**iv**) by using the ELM learner defined in Eq. (3):

$$\begin{aligned} \dot{L}(\mathbf{x}) &= \frac{\mathrm{d}}{\mathrm{d}t} L(\mathbf{x}) = (\nabla_{\mathbf{x}} L(\mathbf{x}))^T \cdot \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = (\nabla_{\mathbf{x}} L(\mathbf{x}))^T \cdot \hat{\mathbf{v}} \\ &= \sum_{i=1}^{d} (\nabla_{\mathbf{x}} L(\mathbf{x}))_i \cdot \sum_{k=1}^{R} W_{ij}^{\mathbf{out}} \cdot f\Big(\sum_{k=1}^{d} W_{jk}^{\mathbf{inp}} x_k + b_j\Big) < 0 \ . \end{aligned} \tag{6}$$

Note that $\dot{L}$ is linear in the output parameters $W^{\mathbf{out}}$ irrespective of the form of the Lyapunov function L. Eq. (6) defines a linear inequality constraint $\dot{L}(\mathbf{u}) < 0$ on the read-out parameters $W^{\mathbf{out}}$ for a given point $\mathbf{u} \in \Omega$ which can be implemented by quadratic programming [20], [21]. The training of the read-out weights $W^{\mathbf{out}}$ given by Eq. (4) is rephrased as a quadratic program subject to constraints given by L:

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

where the matrix V contains the corresponding target velocities of the demonstrations and $U = \{\mathbf{u}(1), \ldots, \mathbf{u}(N_s)\}$ is the matrix collecting the discrete samples. Note that it was already shown in [17] that a well-chosen sampling of such points collected in the set U is sufficient to generalize the incorporated discrete inequalities to continuous regions. However, an ex-post process is needed to verify that the constraint holds in the continuous region [17].

D. Quadratic Lyapunov Candidate

The most commonly used Lyapunov function is given by $L_q = (\mathbf{x} - \mathbf{x}^*)^T I (\mathbf{x} - \mathbf{x}^*)$, where I denotes the identity matrix. This function is data independent, quadratic and fulfills conditions (**i**)-(**iii**). This function is also a valid Lyapunov function for estimates produced by SEDS [13].

In [16], Lyapunov candidates of the following form are considered:

$$L_P(\mathbf{x}) = \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)^T \cdot P \cdot (\mathbf{x} - \mathbf{x}^*) \ . \tag{8}$$

Note that (**i**)-(**iii**) are fulfilled if P is positive definite and symmetric. It is defined as:

$$P = \arg\min_{P \in \mathscr{P}} M(L_P) \ . \tag{9}$$

We restrict the possible matrices to be an element of $\mathscr{P} := \{P \in \mathbb{R}^{d \times d} : P^T = P,\ \lambda_i \in [\alpha, 1],\ \lambda_i \text{ is EV of } G\}$[1], where $\alpha = 0.1$ is a small and positive scalar. M is defined according to the following sum over the training data:

$$M(L_P) = \frac{1}{N_{\mathrm{ds}}} \sum_{i=1}^{N_{\mathrm{traj}}} \sum_{k=1}^{N^i} \Theta\left[(\mathbf{x}^i(k))^T \cdot P \cdot \mathbf{v}^i(k) \right] \ , \tag{10}$$

where Θ denotes the ramp function. The minimization operator in Eq. (9) can be formulated as a nonlinear program. We use successive quadratic programming based on quasi Newton methods for optimization [22].

E. Sampling Constraints from the Lyapunov Candidate

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.

The data set D for training and the region $\mathscr{S}$ where the constraints are supposed to be implemented are assumed to be given. As a first step ($k = 0$), the network is initialized randomly and trained without any constraints (i.e. the sample matrix $U^k = U^0 = \emptyset$ is empty). In this case learning can be accomplished by ridge regression - 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. Afterwards, the number of samples ν fulfilling (**iv**) of Lyapunov's conditions of asymptotic stability

[1] EV is used as abbreviation of eigenvalue.

is determined. The sampling algorithm stops if more then p percent of this samples fulfill the constraint. Otherwise, the most violating sample $\hat{\mathbf{u}}$ of condition (**iv**) is added to the sample pool: $U^{k+1} = U^k \cup \hat{\mathbf{u}}$. 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 $\mathscr{S}$, counter $k = 0$, sample pool $U^k = \emptyset$, and ELM $\hat{\mathbf{v}}$ trained with D
  repeat
    draw samples $\hat{U} = \{\hat{\mathbf{u}}^1, \hat{\mathbf{u}}^2, \ldots, \hat{\mathbf{u}}^{N_C}\}$
    $v =$ no. of samples in $\hat{U}$ fulfilling (**iv**)
    $U^{k+1} = U^k \cup \arg\max_{\mathbf{u} \in \hat{U}} \dot{L}(\mathbf{u})$
    train ELM with D and U^{k+1}
  until $p > \frac{v}{N_C}$

III. What is a good Lyapunov Candidate for Learning?

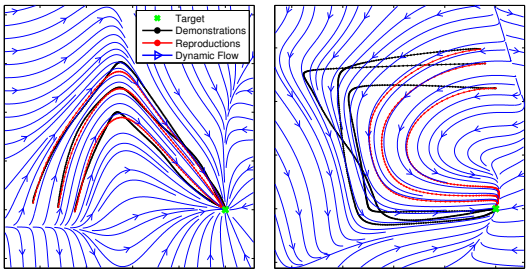

Fig. 4. Stable estimation of A-shape (left) and sharp-C (right) by applying L_q as Lyapunov candidate. Compare with Fig. 3

Fig. 4 shows the same demonstrations as in Fig. 3 but with additional consideration of stability by using L_q. It is shown that the form of the Lyapunov function candidate has a great impact on the resulting shape of the estimated dynamics. The data-independent candidate function L_q is sufficient to estimate the A-shape (Fig. 4, left) with good generalization. It is less suited to approximate the sharp-C shape (Fig. 4, right).

Note that it is impossible to find an estimate for the sharp-C that approximates the data accurately while simultaneously fulfilling the quadratic constraint given by the Lyapunov function candidate 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°:

$$\sphericalangle(-\nabla L(\mathbf{x}(k)), \mathbf{v}(k)) > 90^\circ \Leftrightarrow (\nabla L(\mathbf{x}(k)))^T \cdot \mathbf{v}(k) > 0 \quad (11)$$

Note that the derivation in Eq. (6) shows that this is in contradiction to condition (**iv**) of Lyapunov's theorem.

In order to prevent a violation of the training data D by a given Lyapunov candidate L we generalize the measure M in Eq. (10) to arbitrary Lyapunov candidates. The following functional defines this measure of violation:

$$M(L) = \frac{1}{N_{\text{ds}}} \sum_{i=1}^{N_{\text{traj}}} \sum_{k=1}^{N^i} \Theta\left[(\nabla L(\mathbf{x}^i(k)))^T \cdot \mathbf{v}^i(k)\right] \ , \quad (12)$$

only those samples $(\mathbf{x}^i(k), \mathbf{v}^i(k))$ are counted in M where the scalar product between $\nabla L(\mathbf{x}^i(k))$ and $\mathbf{v}^i(k)$ is positive and thus violating Lyapunov's condition (**iv**).

A good Lyapunov candidate does not violate the training data to a high degree (i.e. M is small). We cannot guarantee this for data-independent functions in general. This raises the question: which Lyapunov candidate function is suitable for learning dynamical systems from complex data?

A. Neural Architecture for Learning Lyapunov Candidates

The Lyapunov candidate functions L_q and L_P are only able to capture a limited class of dynamics. We therefore suggest a candidate which is more flexible but in turn still feasible for implementation.

Consider an ELM architecture which defines a scalar function $L_{\text{ELM}} : \mathbb{R}^d \to \mathbb{R}$. Note, that this ELM also contains a three-layered and random projection structure with only one output neuron. The read-out matrix becomes $W^{\mathbf{out}} \in \mathbb{R}^R$. The main goal is to minimize the violation of the training data measured by Eq. (12) by making the negative gradient of this function follow the training data closely. We again define a quadratic program:

$$\frac{1}{N_{\text{ds}}} \sum_{i=1}^{N_{\text{traj}}} \sum_{k=1}^{N^i} \left(\| -\nabla L_{\text{ELM}}(\mathbf{x}^i(k)) - \mathbf{v}^i(k) \|^2 + \ldots + \varepsilon_{\text{RR}} \|W^{\mathbf{out}}\|^2 \right) \to \min_{W^{\mathbf{out}}} \ , \quad (13)$$

subject to the following equality and inequality constraints corresponding to Lyapunov's conditions (**i**)-(**iv**) such that L_{ELM} becomes a valid Lyapunov function candidate:

$$\begin{array}{llll} (\mathbf{a}) & L_{\text{ELM}}(\mathbf{x}^*) = 0 & (\mathbf{b}) & L_{\text{ELM}}(\mathbf{x}) > 0 : \mathbf{x} \neq \mathbf{x}^* \\ (\mathbf{c}) & \nabla L_{\text{ELM}}(\mathbf{x}^*) = 0 & (\mathbf{d}) & \dot{L}_{\text{ELM}}(\mathbf{x}) < 0 : \mathbf{x} \neq \mathbf{x}^* \end{array} \quad (14)$$

where the time derivative in (**d**) is defined w.r.t the stable system $\dot{\mathbf{x}} = -\mathbf{x}$. The constraints (**b**) and (**c**) define inequality constraints which are implemented by sampling these constraints. The gradient of the scalar function defined by the ELM is linear in $W^{\mathbf{out}}$ and given by:

$$(L_{\text{ELM}}(\mathbf{x}))_i = \sum_{j=1}^{R} W_j^{\mathbf{out}} f'\left(\sum_{k=1}^{d} W_{jk}^{\mathbf{inp}} x_k + b_j\right) \ , \quad (15)$$

where f' denotes the first derivative of the Fermi function. We define a sampling strategy very similar to the one already defined in Sect. II-E. This strategy is described in Alg. 2.

IV. Benchmarking on the LASA Data Set

In the previous sections, three different Lyapunov candidates L_q, L_P, and L_{ELM} are suggested for learning. The performance of these candidates in comparison to SEDS[2] are

[2]We used the SEDS software by Khansari-Zadeh et al. [23]

Algorithm 2 Lyapunov Function Learning

Require: data set D, region $\mathscr{S}$, counter $k=0$, sample pools $U_1^k=\emptyset$ and $U_2^k=\emptyset$

Require: $L_{\text{ELM}}:\mathbb{R}^d\to\mathbb{R}$ trained with D w.r.t. (**a**) and (**c**)

repeat

draw samples $\hat{U}=\{\hat{\mathbf{u}}^1,\hat{\mathbf{u}}^2,\ldots,\hat{\mathbf{u}}^{N_C}\}$

$\nu_1=$ no. of samples in $\hat{U}$ fulfilling (**2**)

$\nu_2=$ no. of samples in $\hat{U}$ fulfilling (**4**)

if $p>\frac{\nu_1}{N_C}$ **then** $U_1^{k+1}=U_1^k\cup\arg\max_{\mathbf{u}\in\hat{U}} L_{\text{ELM}}(\mathbf{u})$

if $p>\frac{\nu_2}{N_C}$ **then** $U_2^{k+1}=U_2^k\cup\arg\min_{\mathbf{u}\in\hat{U}} \dot{L}_{\text{ELM}}(\mathbf{u})$

train ELM with D, U_1^{k+1} and U_2^{k+1} w.r.t. (**a**) and (**c**)

until $p>\frac{\nu_1}{N_C}$ and $p>\frac{\nu_2}{N_C}$

evaluated on a library of 20 human handwriting motions from the LASA data set [24]. The local accuracy of the estimate is measured according to [13]:

$$E_{\text{velo}}=\frac{1}{N_{\text{ds}}}\sum_{i=1}^{N_{\text{traj}}}\sum_{k=1}^{N^i}\left(r\left(1-\frac{\mathbf{v}^i(k)^T\hat{\mathbf{v}}(\mathbf{x}^i(k))}{\|\mathbf{v}^i(k)\|\|\hat{\mathbf{v}}(\mathbf{x}^i(k))\|+\varepsilon}\right)^2+q\frac{(\mathbf{v}^i(k)-\hat{\mathbf{v}}(\mathbf{x}^i(k)))^T(\mathbf{v}^i(k)-\hat{\mathbf{v}}(\mathbf{x}^i(k)))}{\|\mathbf{v}^i(k)\|\|\mathbf{v}^i(k)\|+\varepsilon}\right)^{\frac{1}{2}}, \tag{16}$$

which quantifies the discrepancy between the direction and magnitude of the estimated and velocity vectors for all training data points[3]. The accuracy of the reproductions is measured according to

$$E_{\text{traj}}=\frac{1}{T\cdot N_{\text{ds}}}\sum_{i=1}^{N_{\text{traj}}}\sum_{k=1}^{N^i}\min_l\|\hat{\mathbf{x}}^i(k)-\mathbf{x}^i(l)\|\ , \tag{17}$$

where $\hat{\mathbf{x}}^i(\cdot)$ is the equidistantly sampled reproduction of the trajectory $\mathbf{x}^i(\cdot)$ and T denotes the mean length of the demonstrations. We nevertheless believe that the trajectory error is a more appropriate measure than the velocity error to quantify the quality of the dynamical estimate.

The results for each shape are averaged over ten network initializations. Each network used for estimation of the dynamical system comprises $R=100$ hidden layer neurons, $\varepsilon_{\text{RR}}=10^{-8}$ as regularization parameter, and $N_C=10^5$ samples for learning. The networks for Lyapunov candidate learning also comprise $R=100$ neurons in the hidden layer and $\varepsilon_{\text{RR}}=10^{-8}$ as regularization parameter. The SEDS models where initialized with $K=5$ Gaussian functions in the *mse* mode and trained for maximally 500 iterations. Two movements from the data set are taken to analyze the performance of the methods in detail: A-shape and J-2-shape. This demonstrations are shown in Fig. 3 (left) and Fig. 5, respectively. The experimental results are provided in Tab. I. The table contains the time used for computation in seconds, the measure of violation M, the velocity error according to Eq. (16), and the trajectory error defined in Eq. (17). All experiments have been accomplished on a x86_64 linux machine with at least 4 GB of memory and a 2+ GHz intel processor in matlab R2010a[4]. The overall results for the LASA data set are collected in Tab. I (last tabular). Fig. 5 visualizes the estimation of the J-2-shape and the respective Lyapunov candidates.

TABLE I

RESULTS FOR A-SHAPE, J-2-SHAPE, AND THE ENTIRE DATA SET.

A-shape	Time $[s]$	M	E_{velo}	E_{traj}
SEDS	22.1 ± 1.3	0.008	$.113\pm.0029$	$.0168\pm.0004$
L_q	9.2 ± 0.9	0.008	$.106\pm.0006$	$.0081\pm.0003$
L_P	$\mathbf{7.8\pm0.8}$	**0.000**	$\mathbf{.096\pm.0015}$	$\mathbf{.0053\pm.0002}$
L_{ELM}	50.5 ± 4.6	0.000	$.103\pm.0023$	$.0066\pm.0004$
J-2-shape	Time $[s]$	M	E_{velo}	E_{traj}
SEDS	$\mathbf{23.0\pm1.4}$	0.198	$.119\pm.0008$	$.0338\pm.0001$
L_q	49.8 ± 3.5	0.198	$.093\pm.0015$	$.0298\pm.0032$
L_P	36.4 ± 2.7	0.147	$.143\pm.0011$	$.0241\pm.0004$
L_{ELM}	55.2 ± 4.8	**0.000**	$\mathbf{.069\pm.0013}$	$\mathbf{.0079\pm.0004}$
LASA all	Time $[s]$	M	E_{velo}	E_{traj}
SEDS	22.6 ± 1.3	0.0582	$.109\pm.0020$	$.0169\pm.0003$
L_q	$\mathbf{14.3\pm1.5}$	0.0582	$.114\pm.0003$	$.0218\pm.0005$
L_P	15.9 ± 2.3	0.0180	$.110\pm.0004$	$.0177\pm.0003$
L_{ELM}	52.4 ± 4.0	**0.0001**	$\mathbf{.103\pm.0006}$	$\mathbf{.0145\pm.0004}$

The numerical results in Tab. I show that the function used for implementation of asymptotic stability has a strong impact on the approximation ability of the networks. The A-shape (see first tabular) was accurately learned by all models. The reason is that the demonstrations do not violate the respective Lyapunov candidates to a high degree which is indicated by small value of M. The J-2-shape (see second tabular of Tab. I) is one of the most difficult shapes among the shapes in the LASA data set. The experiments reveal that the networks trained with L_{ELM} candidate stabilization have the best velocity and trajectory error values. This is due the high flexibility of the candidate function which eliminates the violation of the training data. The third tabular in Tab. I shows the results for the whole data set. It shows that the differences in the velocity error are marginal in comparison to the values for the trajectory error. The networks trained with the quadratic Lyapunov candidate L_q perform the worst because the function cannot capture the structure in the data. SEDS performs in the range of the quadratic functions due the conservative stability constraints. The method using L_{ELM} has the lowest trajectory error values which is due to the high flexibility of the candidate function. Methods which cannot avoid the violation of the training data are, in principle, not able to approximate a given data set accurately while providing stable dynamics. The experiments thus reveal that a flexible, data-dependent construction and application of a Lyapunov function candidate is indispensable to resolve the trade-off between stability and accuracy. However, it is important to note that SEDS has the very appealing feature that the stability is guaranteed by construction of the model which is not possible with the proposed approach. Nevertheless, as the neural network method allows analytical differentiation, asymptotic stability can be constructively and effectively proven ex-post if needed [17] but with the disadvantage of computational expensiveness. The consumption

[3]Measure and values ($r=0.6$, $q=0.4$) taken from [13]. We set $\varepsilon=10^{-6}$.

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

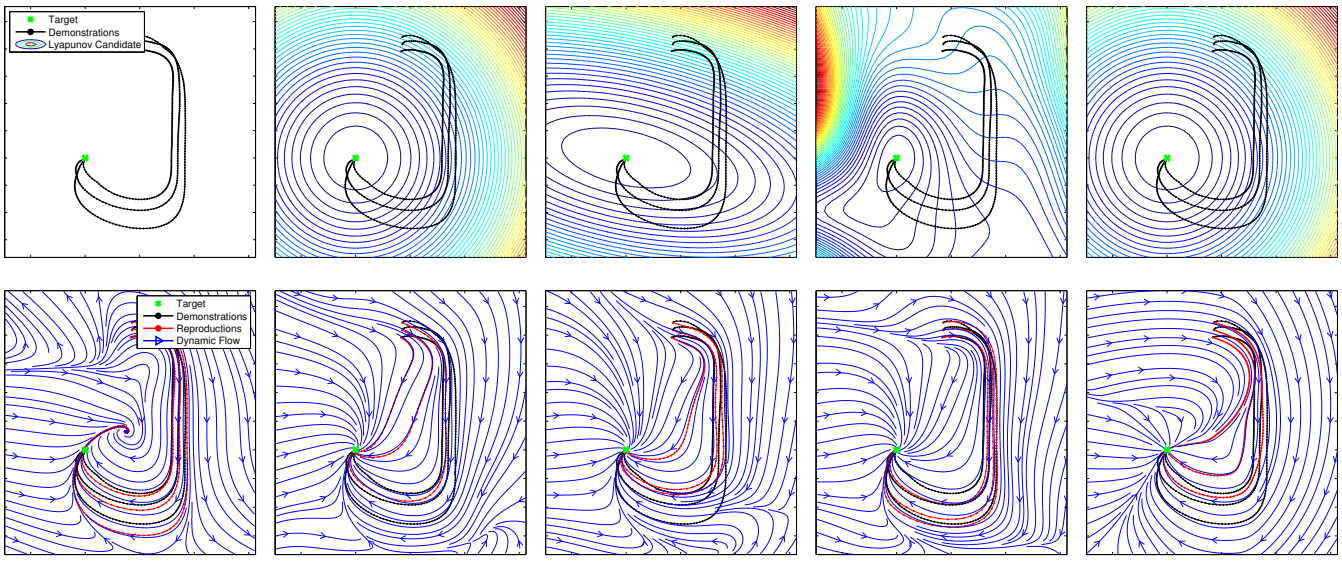

Fig. 5. Estimates of the J-2-shape and respective Lyapunov candidates. The J-2-shape approximated without explicit stabilization (first column), with L_q (second column), L_P (third column), L_{ELM} as Lyapunov candidate (fourth column). The SEDS estimate (fifth column, first row). The stability conditions in SEDS are derived based on a quadratic energy function [14] (fifth column, second row).

of computational time is also given in Tab. I.

In addition to the experiments, Fig. 5 shows the estimations of the J-2-shape and their respective Lyapunov candidates. The left column of the figure contains the estimation result obtained for an ELM without regards to stability. Obviously, even reproductions starting in the vicinity of the demonstrations are prone to divergence. The second column of Fig. 5 illustrates the results for networks trained with respect to L_q. It is shown that this Lyapunov candidate introduces a very strict form of stability, irrespective of the demonstrations. The reproductions are directly converging towards the attractor. This is due to the high violation of the demonstrations by L_q close to the start of the demonstrations. Column three of Fig. 5 shows the results for L_P. This Lyapunov candidate is data-dependent but still too limited to capture the full structure of the J-2 demonstrations. The fourth column of the figure illustrates the performance of the networks trained by L_{ELM}. The Lyapunov candidate is strongly curved to follow the demonstrations closely (first row, fourth column). The estimate leads to very accurate reproductions with good generalization capability. The results for SEDS are shown in the fifth column of the figure. As mentioned, SEDS is subject to constraints corresponding to a quadratic Lyapunov function L_q. The results are very similar to the results for the networks applying L_q or L_P as Lyapunov candidate.

V. Kinesthetic Teaching of iCub

We apply the presented Lyapunov approach in a real world scenario involving the humanoid robot iCub [9]. Such robots are typically designed to solve service tasks in environments where a high flexibility is required. Robust adaptability by means of learning is thus a prerequisite for such systems. The experimental setting is illustrated in Fig. 6. A human tutor

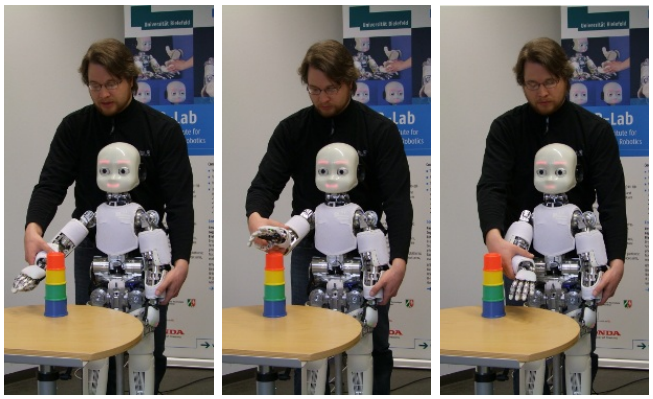

Fig. 6. Kinesthetic teaching of iCub. The tutor moves iCub's right arm from the right to the left side of the small colored tower.

physically guides iCubs right arm in the sense of kinesthetic teaching using a recently established force control on the robot [25]. The tutor can thereby actively move all joints of the arm to place the end-effector at the desired position. Beginning on the right side of the workspace, the tutor first moves the arm around the obstacle on the table, touches its top, and then moves the arm towards the left side of the obstacle were the movement stops. This procedure is repeated three times.

The recorded demonstrations comprise between $N_{\mathrm{traj}} = 542$ and $N_{\mathrm{traj}} = 644$ samples. The hidden layer of the networks estimating the dynamical system consists of $R = 100$ neurons and the regression parameter is $\varepsilon_{\mathrm{RR}} = 10^{-5}$ in the experiment as well as for the network used for L_{ELM}. The networks' weights and biases are initialized randomly from a uniform distribution in the interval $[10, 10]$ due to the low ranges of the movement. The results of the experiment are visualized in Fig. 7. The figure shows the impact of the different Lyapunov candidates on the estimation of the

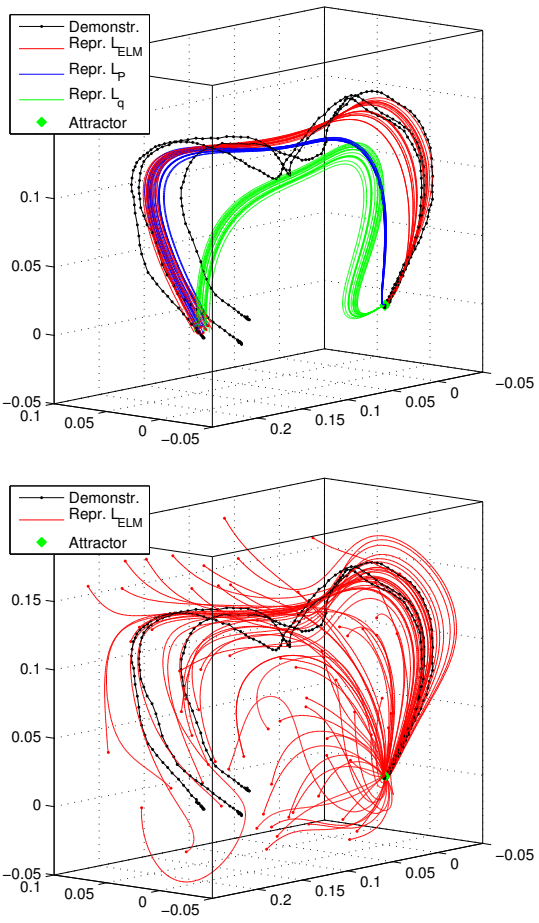

Fig. 7. Results of the iCub teaching experiment. Reproduction of the learned trajectories in meter according to the Lyapunov candidates L_q, L_P, and L_{ELM} (first). Reproductions according to L_{ELM} subject to random spatial perturbations (second).

dynamics. The estimation of the networks trained by the quadratic function L_q is not able to capture the complex shape of the dynamics. The networks trained with the L_P function provides a better performance. The networks using the L_{ELM} function yields a good estimate.

The second plot in Fig. 7 illustrates the robustness of the learned dynamics against spatial perturbations. Therefore, $N = 75$ starting points are randomly drawn from a uniform distribution in $\Omega = [-0.05, 0.1] \times [-0.5, 0.25] \times [-0.05, 0.2]$. Even the trajectories which start far away from the demonstrations converge to the target attractor and thus underline the robustness of the proposed learning method.

VI. CONCLUSIONS

In this paper, we presented a learning approach which is able to estimate vector fields used for movement generation. The method is based on the idea of separating the learning into two main steps: i) learning of a highly flexible Lyapunov candidate from data and ii) implementation of asymptotic stability by means of this function. We demonstrate that this approach strongly reduces the trade-off between stability and accuracy, which allows robust and complex movement generation in robotics.

ACKNOWLEDGMENT

This research is funded by FP7 under GA. No. 248311-AMARSi and the German BMBF within the Intelligent Technical Systems Leading-Edge Cluster.

REFERENCES

[1] M. Mühlig, M. Gienger, and J. J. Steil, "Interactive imitation learning of object movement skills," *Autonomous Robots*, vol. 32, no. 2, pp. 97–114, 2012.

[2] A. Pistillo, S. Calinon, and D. G. Caldwell, "Bilateral physical interaction with a robot manipulator through a weighted combination of flow fields," in *IEEE Conf. IROS*, 2011, pp. 3047–3052.

[3] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, *Robot programming by demonstration*. Springer, 2008, vol. 1.

[4] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[5] F. L. Moro, N. G. Tsagarakis, and D. G. Caldwell, "On the kinematic motion primitives (kmps) - theory and application," *Frontiers in Neurorobotics*, vol. 6, no. 10, 2012.

[6] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[7] S. M. Khansari-Zadeh and A. Billard, "BM: An iterative algorithm to learn stable non-linear dynamical systems with gaussian mixture models," in *IEEE Conf. ICRA*, 2010, pp. 2381–2388.

[8] F. Reinhart, A. Lemme, and J. Steil, "Representation and generalization of bi-manual skills from kinesthetic teaching," in *IEEE Conf. Humanoids*, 2012, pp. 560–567.

[9] 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*, vol. 21, no. 10, pp. 1151–1175, 2007.

[10] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *NIPS*, 2003, pp. 1523–1530.

[11] E. Gribovskaya and A. Billard, "Learning nonlinear multi-variate motion dynamics for real-time position and orientation control of robotic manipulators," in *IEEE Conf. Humanoids*, 2009, pp. 472–477.

[12] K. Dautenhahn and C. L. Nehaniv, "The agent-based perspective on imitation," *Imitation in animals and artifacts*, pp. 1–40, 2002.

[13] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[14] S. M. Khansari-Zadeh, "A dynamical system-based approach to modeling stable robot control policies via imitation learning," Ph.D. dissertation, EPFL, 2012.

[15] E. D. Sontag, "A universal construction of artstein's theorem on nonlinear stabilization," *Systems & control letters*, vol. 13, no. 2, pp. 117–123, 1989.

[16] A. Lemme, K. Neumann, R. F. Reinhart, and J. J. Steil, "Neurally imprinted stable vector fields," in *ESANN*, 2013, in press.

[17] K. Neumann, M. Rolf, and J. J. Steil, "Reliable integration of continuous constraints into extreme learning machines," *Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, in press.

[18] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[19] R. Reinhart and J. Steil, "Neural learning and dynamical selection of redundant solutions for inverse kinematic control," in *IEEE Conf. Humanoids*, 2011, pp. 564–569.

[20] P. Gill, W. Murray, and M. Wright, *Practical Optimization*. Academic Press, 1981.

[21] T. F. Coleman and Y. Li, "A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables," *SIAM Journal on Optimization*, vol. 6, no. 4, pp. 1040–1058, 1996.

[22] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms*. Wiley-interscience, 2006.

[23] S. M. Khansari-Zadeh, http://lasa.epfl.ch/people/member.php?SCIPER=183746/, 2013.

[24] ——, http://www.amarsi-project.eu/open-source, 2012.

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