Stable and robust neural network controllers

Camilla Sterud*,†, Signe Moe† and Jan Tommy Gravdahl*
camilla.sterud@sintef.no, signe.moe@sintef.no, jan.tommy.gravdahl@ntnu.no

Abstract— **Neural networks are expressive function approimators that can be employed for state estimation in control problems. However, control systems with machine learning in the loop often lack stability proofs and performance guarantees, which are crucial for safety-critical applications. In this work, a feedback controller using a feedforward neural network of arbitrary size to estimate unknown dynamics is suggested. The controller is designed for solving a general trajectory tracking problem for a broad class of two-dimensional nonlinear systems. The controller is proven to stabilize the closed-loop system, such that it is input-to-state and finite-gain $\mathscr{L}_p$-stable from the neural network estimation error to the tracking error. Furthermore, the controller is proven to make the tracking error globally and exponentially converge to a ball centered at the origin. When the neural network estimate is updated discretely, or the state measurements are affected by bounded noise, the convergence bound is shown to be dependent on the Lipschitz constant of the neural network estimator. In light of this, we demonstrate how regularization techniques can be beneficial when utilizing deep learning in control. Experiments on simulated data confirm the theoretical results.**

I. Introduction

For the last three decades, researchers have been eager to utilize the enormous amounts of information that exists after years of collecting sensor data for process monitoring and control purposes [1]. In the field of machine learning, several data-hungry approaches have benefitted greatly from the data and inexpensive computing power that have become available. In particular, the subfield of machine learning known as deep learning, which is concerned with large black-box models called deep neural networks (DNNs), or simply neural networks, has grown. Deep learning algorithms have been suggested and successfully implemented for natural language processing, object classification, and solving simulated continuous control tasks, among other things [2]–[4].

A challenge with assuring safety in systems using DNNs is that the internal workings of the networks are generally hard to analyze, and their behavior is challenging to predict. This is exemplified by DNNs' well known weakness to adversarial attacks [5]. However, masures can be taken to increase the robustness of DNN predictions, for instance by applying suitable regularization techniques [6], [7].

Lately, there has been an increasing initiative to combine control theory and machine learning in order to bring rigorous mathematical proofs and reasoning based on physics into data-based methods. Many have proposed ways to incorporate learning into the popular model predictive control scheme, for instance, to achieve stability with pre-learned models, and safe exploration for online model learning [8], [9]. Lyapunov theory has been applied in learning systems to ensure safe exploration in reinforcement learning settings, as done by Berkenkamp et al. [10]. Richards et al. use a particular class of DNNs, dubbed Lyapunov neural networks, to estimate safe sets for dynamical systems [11]. Others have let the notion of closed-loop stability inspire new DNN architectures [12]–[14].

In [15], stability and convergence in a drone path-following problem are proven for a controller where a DNN learns unknown dynamics. They claim to be the first to provide stability guarantees for a DNN-based feedback controller that can utilize arbitrarily large networks. This last work has inspired the learning controller that is proposed here.

This work contributes to the understanding of how data-driven learning and control theory can be safely and productively combined by proposing a controller that gives the closed-loop system attractive stability traits, and provably bounds the error in a trajectory tracking problem. The controller incorporates a feedforward DNN of arbitrary size for estimating unknown model dynamics, and makes the closed-loop system input-to-state and finite-gain $\mathscr{L}_p$-stable from the DNN estimation error to the tracking error. In particular, the Lipschitz constant of the DNN is proven to influence the robustness of the controller, showing that the regularization technique known as spectral normalization can be useful when using DNNs as estimators in control.

II. Background theory

In this work, $\|\cdot\|$ denotes the 2-norm.

A. Feedforward neural networks

The simplest form of a neural network is called a feedforward network (FFN). A FFN takes an input $\boldsymbol{x}$ and maps it to an output $\hat{\boldsymbol{y}} = f(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ denotes the network parameters. The mapping can be seen as L nonlinear mappings applied in succession. Each nonlinear mapping is referred to as a layer.

The output of layer i, $\boldsymbol{u}^i$, is computed as follows

$$\boldsymbol{u}^i = a^i(W^i \cdot \boldsymbol{u}^{i-1} + \boldsymbol{b}^i), \tag{1}$$

where W^i is the weight matrix, $\boldsymbol{b}^i$ is the bias, and a^i is the activation function. Often, in regression problems,

*Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway
†Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway

the last layer has no activation, so that the output is not restricted.

A DNN is trained by computing its output on a set of data points $\boldsymbol{x}_i$ and comparing the network outputs $\hat{\boldsymbol{y}}_i$ to the corresponding ground truth points $\boldsymbol{y}_i$. The weights and biases of the network are usually adjusted with gradient descent methods applied to an objective function that measures the distance between the network predictions and the ground truth.

B. The Lipschitz constant

The Lipschitz condition is given by $\|f(s_1)-f(s_2)\| \leq \Lambda\|s_1-s_2\|$, where $\Lambda>0$. If this condition holds for all s_1, s_2 in a neighborhood of some point s_0, the function f is said to be Lipschitz continuous in this neighborhood. Every Λ that satisfies the Lipschitz condition in some neighborhood, is called a Lipschitz constant of f in that neighborhood. The smallest Lipschitz constant gives a bound on how much the output of f can change, given some change in the input [16].

C. Naive upper bound on the Lipschitz constant of feed forward networks

Consider the FFN layer from (1). The typical DNN activation functions, sigmoid, ReLU, and tanh, are all Lipschitz continuous with derivatives that are contained in the interval $[0,1]$. Using this, it is easily proven that the norm of the derivative of a layer with respect to its input is upper bounded by the norm of the layer weight matrix: $\left\|\frac{\partial \boldsymbol{u}^i}{\partial \boldsymbol{u}^{i-1}}\right\| \leq \|W^i\|$.

If $\left\|\frac{\partial f}{\partial \boldsymbol{x}}\right\| \leq \Lambda$, then Λ is a Lipschitz constant for f, where $f:\mathbb{R}^n \to \mathbb{R}^m$ [16]. Further, two transformations g and h with Lipschitz constants Λ_g and Λ_h form a Lipschitz continuous composition $g\circ h$ with Lipschitz constant $\Lambda_g\Lambda_h$ [16].

From this we can conclude that the FFN layer (1) has a Lipschitz constant $\Lambda^i=\|W^i\|$, and that a FFN with L layers is Lipschitz continuous in its input space with a naive upper bound on the Lipschitz constant given by

$$\Lambda_U=\prod_{i=1}^{L}\|W^i\|. \tag{2}$$

D. Spectral normalization

Spectral normalization, as suggested in [6], can be applied to fix the naive upper bound on the Lipschitz constant of a FFN to some chosen value $\gamma>0$.

The 2-norm, also called the spectral norm, of a matrix is equal to the largest singular value $\sigma_{\max}$ of the matrix. Hence, the naive upper bound (2) is simply a product of the largest singular values of the layer weight matrices of the FFN. Normalizing the weight matrices by their largest singular value results in a spectral norm equal to one,

$$\bar{W}=\frac{W}{\sigma_{\max}(W)} \Rightarrow \|\bar{W}\|=1.$$

If all weight matrices are normalized, the Lipschitz constant of a FFN with L layers is upper bounded by 1, in accordance with (2). By multiplying each normalized weight matrix by $\gamma^{\frac{1}{L}}$, the upper bound on the Lipschitz constant becomes γ:

$$\Lambda_U=\prod_{i=1}^{L}\left\|\frac{W^i}{\sigma_{\max}(W^i)}\right\|\gamma^{\frac{1}{L}}=\gamma.$$

Thus, by normalizing and multiplying all weight matrices by $\gamma^{\frac{1}{L}}$ during training, the upper bound γ on the Lispchitz constant of the FFN becomes a hyperparameter that can be freely chosen. The largest singular values of the weight matrices can be estimated using the power iteration method, as suggested in [6].

E. Robustness of regularized neural networks

A DNN $\hat{d}(\cdot)$, with Lipschitz constant Λ, estimates a function $d(\cdot)$ which is Lipschitz continuous with Lipschitz constant L. Let $\hat{\varepsilon}$ be the largest estimation error on some test set, observed at test sample $\hat{x}$. When the DNN is presented with a new, unseen observation $\tilde{x}$, the estimation error satisfies the following:

$$\begin{aligned}\left\|d(\tilde{x})-\hat{d}(\tilde{x})\right\| &= \left\|d(\tilde{x})-d(\hat{x})+d(\hat{x})+\hat{d}(\hat{x})-\hat{d}(\hat{x})-\hat{d}(\tilde{x})\right\| \\ &\leq \left\|d(\tilde{x})-d(\hat{x})\right\|+\left\|d(\hat{x})-\hat{d}(\hat{x})\right\| \\ &\quad+\left\|\hat{d}(\hat{x})-\hat{d}(\tilde{x})\right\| \\ &\leq (L+\Lambda)\left\|\tilde{x}-\hat{x}\right\|+\hat{\varepsilon}.\end{aligned}$$

Hence, a DNN with a smaller Lipschitz constant gives a stronger guarantee in terms of robustness to pertubations in the input data. This is assuming the decrease in Lipschitz constant does not hamper the accuracy of the network. Decreasing Λ makes the DNN less flexible, as the variance is reduced, and the bias increased, which might contribute to an increase of $\hat{\varepsilon}$.

III. Stable neural network controllers

A. Trajectory tracking problem

Consider all systems of the form

$$\begin{aligned}\dot{x}_1 &= c_{11}x_1+c_{12}x_2 \\ \dot{x}_2 &= d(\boldsymbol{x},u)+bu,\end{aligned} \tag{3}$$

where c_{11}, c_{12} and b are known constants, and $d(\boldsymbol{x},u):\mathbb{R}^2\times\mathbb{R}\to\mathbb{R}$ is a continuous, unknown, nonlinear mapping. The objective is to make $x_1(t)$ follow the trajectory $x_1^r(t)$. It is assumed that $x_1^r(t)$ is twice continuously differentiable, and that its derivatives are known.

If $\dot{x}_2$ can be measured, data points $\begin{bmatrix}\boldsymbol{x}^T(t_i) & u(t_i)\end{bmatrix}$ and corresponding ground truth points $d(\boldsymbol{x}(t_i),u(t_i))$ can be gathered, as $bu(t_i)$ is known. A FFN that makes an estimate $\hat{d}(\boldsymbol{x},u)$ of $d(\boldsymbol{x},u)$ can then be trained on the resulting dataset.

B. Feedback controller

Making x_1 track $x_1^r(t)$ is equivalent to driving $z_1 = x_1 - x_1^r(t)$ to zero. We define the reference value $x_2^r(t)$ as

$$x_2^r(t) = -\frac{1}{c_{12}}(c_{11}x_1 - \dot{x}_1^r(t) + \Gamma z_1),$$

such that the composite variable z_2 is

$$z_2 = x_2 - x_2^r(t) = \frac{1}{c_{12}}(\dot{z}_1 + \Gamma z_1),$$

where Γ is a freely chosen parameter.

It follows that the dynamics of $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^T$ can be written as

$$\dot{\boldsymbol{z}} = A\boldsymbol{z} + B(u - \frac{1}{b}[-d(\boldsymbol{x},u) + \dot{x}_2^r(t)]), \tag{4}$$

where

$$A = \begin{bmatrix} -\Gamma & c_{12} \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

Equation (4) is a form of error dynamics of (3), when the objective is trajectory tracking of $x_1^r(t)$. The pair (A,B) is controllable for all $c_{12} \neq 0$.

Let $\hat{d}(\boldsymbol{x},u)$ be the output of a DNN estimating $d(\boldsymbol{x},u)$ by mapping the states and input to a real number. Define the estimation error of the DNN at time t as $\tilde{d}(t) = d(\boldsymbol{x}(t),u(t)) - \hat{d}(\boldsymbol{x}(t),u(t))$. Assuming that new estimates are continuously available, the controller

$$u = -K\boldsymbol{z} - \frac{1}{b}\left[\hat{d}(\boldsymbol{x}(t),u(t)) - \dot{x}_2^r(t)\right], \quad K = \begin{bmatrix} k_1 & k_2 \end{bmatrix}, \tag{5}$$

where k_1 and k_2 are scalar gains, is proposed. Applying this controller to (4) yields the closed-loop dynamics

$$\dot{\boldsymbol{z}} = (A - BK)\boldsymbol{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\tilde{d}(t). \tag{6}$$

Even though the forward pass of a DNN is relatively quick, the assumption that the controller in (5) can be updated continuously might be violated. Therefore, the scenario where a new estimate can only be made every $\Delta t_{\max}$ seconds is considered. This results in the controller

$$u(t) = -K\boldsymbol{z}(t) - \frac{1}{b}[\hat{d}(\boldsymbol{x}(t-\Delta t), u(t-\Delta t)) - \dot{x}_2^r(t)], \tag{7}$$

where $\Delta t \leq \Delta t_{\max}$ is the time since the last estimate was made. It is assumed that the linear term of the controller can be updated continuously. Applying (7) to (4) yields the perturbed system

$$\dot{\boldsymbol{z}} = (A - BK)\boldsymbol{z} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t), \tag{8}$$

where $w(t) = d(\boldsymbol{x}(t),u(t)) - \hat{d}(\boldsymbol{x}(t-\Delta t), u(t-\Delta t))$.

C. Stability properties

The stability and convergence properties of the closed-loop systems (6) and (8) are summarized in the two following theorems. The validity of assumption 1) in Theorem 1 is discussed in Section VI.

Theorem 1 (Closed-loop stability and convergence of (6).) *Assume that*

1) *The error of the DNN estimate is bounded: $\|\tilde{d}(t)\| \leq \varepsilon$.*
2) *The DNN estimate of the unknown dynamics can be updated continuously.*

Let the input u of (3) *be given by the continuous DNN controller* (5)*, where K and Γ satisfy* (9).

$$k_1 = -\frac{c_{12}}{b}, \quad \Gamma > 0, \quad \Gamma k_2 > \frac{c_{12}^2}{b} \tag{9}$$

Then, the resulting closed-loop system (6) *is input-to-state and finite-gain $\mathscr{L}_p$-stable from $\tilde{d}(t)$ to $\boldsymbol{z}(t)$.*

The error norm converges globally and exponentially to a ball centered at the origin:

$$\lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq \left|\frac{\varepsilon}{\lambda}\right|, \tag{10}$$

where λ denotes the least negative eigenvalue of $A - BK$.

Further, if the measurements of $\boldsymbol{x}$ are affected by some additive bounded noise $\boldsymbol{q}^x = [q_1 \quad q_2]^T$ satisfying $|q_1| \leq q_1^m$, $|q_2| \leq q_2^m$, the error norm converges to

$$\lim_{t\to\infty} \|\boldsymbol{z}(t)\| \leq \left|\frac{1}{\lambda}\left(\Lambda \|\boldsymbol{q}^x\| + B^q + \varepsilon\right)\right| \tag{11}$$

where Λ is the Lipschitz constant of the DNN estimator and $B^q = |c_{12} + \frac{bk_2+1}{c_{12}}(c_{11} + \Gamma)|q_1^m + |k_2 b|q_2^m$.

Theorem 2 (Closed-loop convergence of (8).) *Assume that*

1) *The error of the DNN estimate is bounded: $\tilde{d} \leq \varepsilon$.*
2) *The change in the system state and input is bounded: $\|\boldsymbol{x}(t) - \boldsymbol{x}(t-\Delta t)\| \leq \rho^x$, $\|u(t) - u(t-\Delta t)\| \leq \rho^u$ for some small time-step $\Delta t \leq \Delta t_{max}$.*

Let u be given by the discretely updated DNN controller (7) *where K and Γ satisfy* (9)*. Then, the resulting closed-loop system* (8) *is input-to-state and finite-gain $\mathscr{L}_p$-stable from $w(t)$ to $\boldsymbol{z}(t)$.*

The error norm converges globally and exponentially to an error-ball centered at the origin:

$$\lim_{t\to\infty} \|z(t)\| \leq \left| \frac{(\Lambda(\rho^x+\rho^u)+\varepsilon)}{\lambda} \right|. \quad (12)$$

Further, if the measurements of $\boldsymbol{x}$ are affected by some additive bounded noise, the error norm converges to

$$\lim_{t\to\infty} \|z(t)\| \leq \left| \frac{1}{\lambda}\left(\Lambda(\rho^x+\rho^u+\|\boldsymbol{q}^x\|)+B^q+\varepsilon\right) \right|. \quad (13)$$

IV. Proof of Theorem 1 and 2

It is sufficient to only prove Theorem 2, as Theorem 1 is just the special case when $\Delta t_{\max}=0$.

When K and Γ are chosen such that (9) is satisfied, $A-BK$ is real symmetric negative definite, and hence, its eigenvalues are real and negative. The perturbation term in (8) satisfies

$$\begin{aligned}\|w(t)\| &= \left\|d(\boldsymbol{x}(t),u(t)) - \hat{d}(\boldsymbol{x}(t-\Delta t),u(t-\Delta t))\right\| \\ &\leq \left\|d(\boldsymbol{x}(t),u(t)) - \hat{d}(\boldsymbol{x}(t),u(t))\right\| \\ &\quad + \left\|\hat{d}(\boldsymbol{x}(t),u(t)) - \hat{d}(\boldsymbol{x}(t-\Delta t),u(t-\Delta t))\right\| \\ &\leq \varepsilon + \Lambda\|\boldsymbol{x}(t)-\boldsymbol{x}(t-\Delta t)\| + \Lambda\|u(t)-u(t-\Delta t)\| \\ \|w(t)\| &\leq \varepsilon + \Lambda(\rho^x+\rho^u).\end{aligned}$$

Define the Lyapunov function $V=\frac{1}{2}\boldsymbol{z}^T\boldsymbol{z}$. Then

$$\begin{aligned}\dot{V} &= \boldsymbol{z}^T(A-BK)\boldsymbol{z} + \boldsymbol{z}^T\begin{bmatrix}0 & 1\end{bmatrix}^T w(t) \\ &\leq \lambda\boldsymbol{z}^T\boldsymbol{z} + \|\boldsymbol{z}\|(\Lambda(\rho^x+\rho^u)+\varepsilon) \\ \dot{V} &\leq 2\lambda V + \sqrt{2V}(\Lambda(\rho^x+\rho^u)+\varepsilon).\end{aligned}$$

Now, let $W=\sqrt{V}=\frac{1}{\sqrt{2}}\|\boldsymbol{z}\|$, so that

$$\dot{W} = \frac{\dot{V}}{2\sqrt{V}} \leq \lambda W + \frac{\Lambda(\rho^x+\rho^u)+\varepsilon}{\sqrt{2}}. \quad (14)$$

With initial condition $\boldsymbol{z}(t_0)=\boldsymbol{z}_0$, the solution to the differential inequality (14) is

$$W(t) \leq \tfrac{1}{\sqrt{2}}(\|\boldsymbol{z}_0\| + \tfrac{\Lambda(\rho^x+\rho^u)+\varepsilon}{\lambda})e^{\lambda(t-t_0)} - \tfrac{\Lambda(\rho^x+\rho^u)+\varepsilon}{\sqrt{2}\lambda}.$$

As $\frac{\mathrm{d}}{\mathrm{d}t}\|\boldsymbol{z}\| = \sqrt{2}\dot{W}$, it can be concluded by the comparison lemma, Lemma 3.4 from [16], that

$$\|\boldsymbol{z}(t)\| \leq \|\boldsymbol{z}_0\| e^{\lambda(t-t_0)} + \frac{\Lambda(\rho^x+\rho^u)+\varepsilon}{\lambda}(e^{\lambda(t-t_0)}-1), \quad (15)$$

which proves that $\lim_{t\to\infty}\|\boldsymbol{z}(t)\| \leq |(\Lambda(\rho^x+\rho^u)+\varepsilon)/\lambda|$.

As (15) holds, it must also be true that

$$\|\boldsymbol{z}(t)\| \leq \|\boldsymbol{z}_0\| e^{\lambda(t-t_0)} - \frac{e}{\lambda},$$

where $e=\Lambda(\rho^x+\rho^u)+\varepsilon$. The function $\gamma(e)=-e/\lambda$ belongs to class $\mathscr{K}$, and $\beta(\|\boldsymbol{z}_0\|, t-t_0)=\|\boldsymbol{z}_0\|e^{\lambda(t-t_0)}$ to class $\mathscr{KL}$. As $e=\sup_{t_0\leq\tau\leq t}\|w(\tau)\|$, system (6) is input-to-state stable from $w(t)$ to $\boldsymbol{z}(t)$ by Definition 4.7 from [16].

The system also satisfies all criteria of Theorem 5.1 from [16] and is therefore finite-gain $\mathscr{L}_p$ stable from $w(t)$ to $\boldsymbol{z}(t)$.

Assume the measurement of $\boldsymbol{x}$ is affected by additive noise $\boldsymbol{q}^x=\begin{bmatrix}q_1 & q_2\end{bmatrix}^T$, where $|q_1|\leq q_1^m$ and $|q_2|\leq q_2^m$. The noisy input is

$$\begin{aligned}u(\boldsymbol{x}+\boldsymbol{q}^x) = -K\boldsymbol{z} - \frac{1}{b}(\hat{d}(\boldsymbol{x}+\boldsymbol{q}^x,u)-\dot{x}_2^r) - \frac{c_{12}}{b}q_1 \\ -(k_2+\frac{1}{b})(\frac{1}{c_{12}}(c_{11}+\Gamma)q_1+q_2) + \frac{1}{b}q_2,\end{aligned}$$

and it follows that

$$\dot{V} \leq \lambda\boldsymbol{z}^T\boldsymbol{z} + \|\boldsymbol{z}\|(\Lambda\|\boldsymbol{q}^x\| + B^q + \Lambda(\rho^x+\rho^u)+\varepsilon),$$

and, by the same reasoning as earlier, (13) holds.

V. Simulations and Results

Two simulations are conducted on a mass-spring-damper system (MSD) to test the controllers (5) and (7). The controllers and the test system were implemented in MATLAB with Simulink, while the deep learning was done in Python 3.6 using Keras [17]–[19].

A. Experiment setup

The MSD dynamics are described by

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{b}{m}u - \frac{1}{m}c_d x_2 - \frac{1}{m}c_s x_1 + d(\boldsymbol{x},u),\end{aligned}$$

where $d(\boldsymbol{x},u)=\sin(u)(\sin(x_1)+\sin(x_2))$ is considered an unknown disturbing force. For this system, the DNN controller (5) is given by

$$u = -K\boldsymbol{z} - \frac{m}{b}[\hat{d}(\boldsymbol{x}(t),u(t)) - \frac{1}{m}c_d x_2 - \frac{1}{m}c_s x_1 - \dot{x}_2^r(t)].$$

The MSD is simulated 500 times for 20 s while a PID controller attempts to make x_1 track a randomly generated sine wave. The period and amplitude of the sine waves are uniformly sampled from the intervals [1, 50] and [0, 5], respectively. The simulation is sampled at 10 Hz, so 100 000 data points $(\boldsymbol{x}(t),u(t))$ with corresponding ground truth points $d(\boldsymbol{x}(t),u(t))$ are available for learning. 25% are set aside for testing, and 15% are used for validation during training.

Three FFNs are trained on this data. All have four tanh-activated layers of size 32, 16, 8 and 1. Spectral normalization is applied to two of the networks, dubbed SpectNorm 10 and SpectNorm 1, restricting their Lipschitz constants to below 10 and 1, respectively. The network without regularization is called FFN. All networks are trained with a mean squared error (MSE) loss for 1600 epochs using the Adam optimizer with batch size 32 and learning rate 0.001.

The controllers were tested on 500 trajectory tracking episodes lasting 20 s, where the objective is to track randomly generated sine waves. Zero mean noise generated by a truncated normal distribution is added to the measurements of $\boldsymbol{x}$. The distribution is truncated at one standard deviation $\pm\sigma$. Two simulated experiments are conducted: In Experiment 1 the DNN estimates are calculated every 0.01 s, which is considered as continuously. In Experiment 2, the DNN estimates are only calculated every second.

B. Results

The performance of the controllers are summarized in Tables I and II, and visualized in Figure 1. Here, the expected estimation error is calculated by computing the empirical average over the 500 test episodes. The controller without an estimator ($\hat{d} = 0$ always) is also tested on the MSD for comparison. A representative example of the controller behaviors is shown in Figure 2, and the expected error norm and tracking error over the 500 test episodes is shown in Figure 3.

Figure 3 shows the empirical mean of the error norms with error bands and bars that are two standard deviations wide. FFN has both the lowest expected error and the lowest standard deviation, which indicates consistent performance, even though it has the highest convergence bound. When noise is introduced, as visualized in Figures 1 and 4, the behavior of SpectNorm 1 changes less than that of FFN and SpectNorm 10 from the noiseless case.

In order to estimate the convergence bounds (11) and (13), ε was set to be the largest prediction error each FFN makes on the test data. The upper bound on the rate of change in state and input ρ^x and ρ^u were set to be the largest observed difference between consecutive controller updates in Experiment 2.

The results confirm that the convergence bounds (11) and (13) apply in practice. The controllers outperform the pure feedback controller that does not compensate for the unknown dynamics, even when the DNN estimate is rarely updated, and the state measurements are affected by noise.

VI. Discussion

Figure 1 illustrates how SpectNorm 1 is relatively less affected by the switch from continuous and discrete operation than the other two controllers. It is also relatively less affected by measuring noise. However, its expected prediction

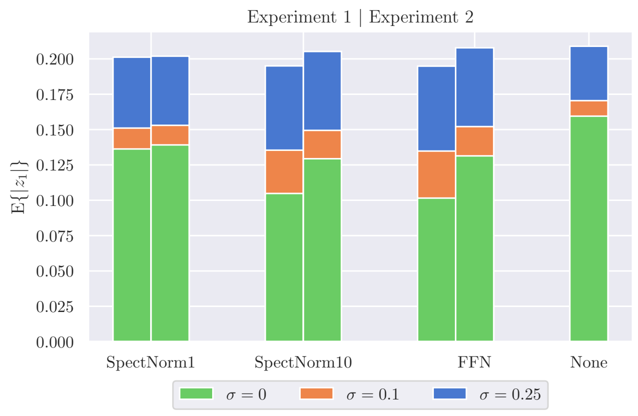

Fig. 1: A visualization of the data from Tables I and II. The left part of each bar presents the results from Experiment 1, while the right presents results from Experiment 2.

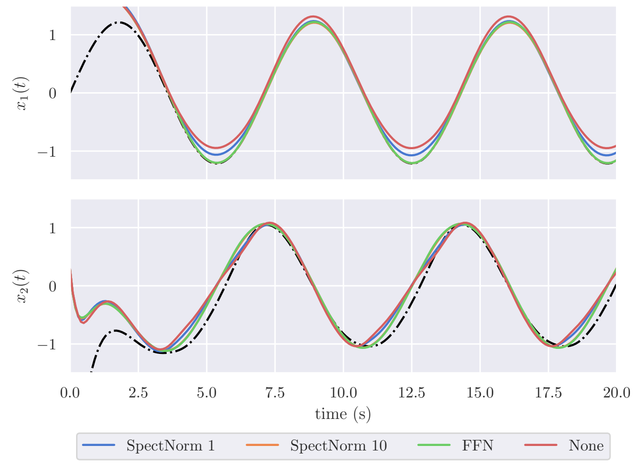

Fig. 2: Example behaviour of the controllers on a random test example without noise. The black stapled lines indicate the reference values $x_1^r(t)$ and $x_2^r(t)$.

error is the highest, and as exemplified in Figure 2, SpectNorm 1 is outperformed by the two other DNN controllers when no measuring noise is present. This illustrates the bias-variance tradeoff which is always present when applying regularization techniques.

SpectNorm 10 and FFN perform very similarly, as is clear from the results in Tables I and II. Yet, the convergence bound of SpectNorm 10 is dramatically lower than that of FFN, and hence, Spectnorm 10 might be preferable as it gives stronger robustness guarantees.

The experiments reveal that the convergence bounds are quite conservative and that a lower bound does not necessarily imply better performance overall. Nonetheless, the experiments indicate that a lower Lipschitz constant reduces the relative effect of rarely updating the DNN estimate, as well as the controller's sensitivity to measuring noise. Hence, regularization schemes that lower the Lipschitz constant of a DNN can be beneficial when tackling noisy data.

TABLE I: Experiment 1: Expected mean error and retrospective convergence bounds when the DNN estimate is updated continuously.

Noise limit	$\sigma=0$			$\sigma=0.1$			$\sigma=0.25$		
DNN estimator	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound
None	0.159	0.325		0.170	0.355		0.209	0.455	
FFN	**0.102**	0.179	2.020	**0.135**	0.264	12.467	**0.195**	0.417	28.143
SpectNorm 10	0.105	0.187	1.606	0.136	0.266	3.154	**0.195**	0.417	5.474
SpectNorm 1	0.136	0.269	1.855	0.151	0.308	2.801	0.201	0.434	4.219

TABLE II: Experiment 2: Expected mean error and retrospective convergence bounds when the DNN estimate is updated every second.

Noise limit	$\sigma=0$			$\sigma=0.1$			$\sigma=0.25$		
DNN estimator	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound	$\mathbb{E}\{\lvert z_1\rvert\}$	$\mathbb{E}\{\lVert\mathbf{z}\rVert\}$	Bound
FFN	0.131	0.265	24.744	0.152	0.318	72.124	0.208	0.458	137.852
SpectNorm 10	**0.129**	0.259	3.509	**0.149**	0.310	7.422	0.205	0.450	13.250
SpectNorm 1	0.139	0.279	2.226	0.153	0.315	3.286	**0.202**	0.436	5.081

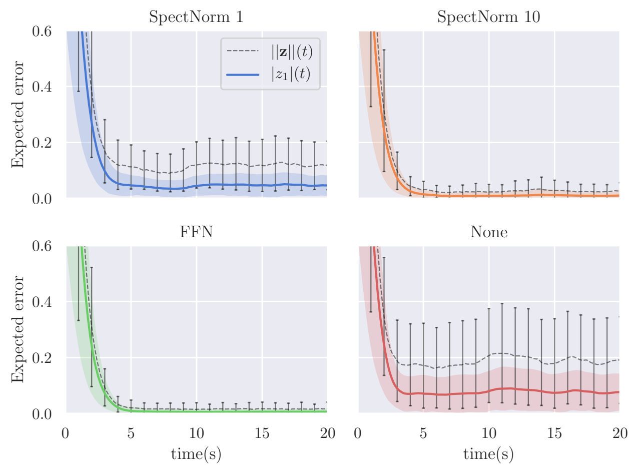

Fig. 3: The expected error of each controller at every time-step of the test episodes. The colored lines represent the absolute error in the first state, while the stapled lines are the mean squared error of the state vector. The error bands and bars represent one standard deviation above and below the mean. No measurement noise is present.

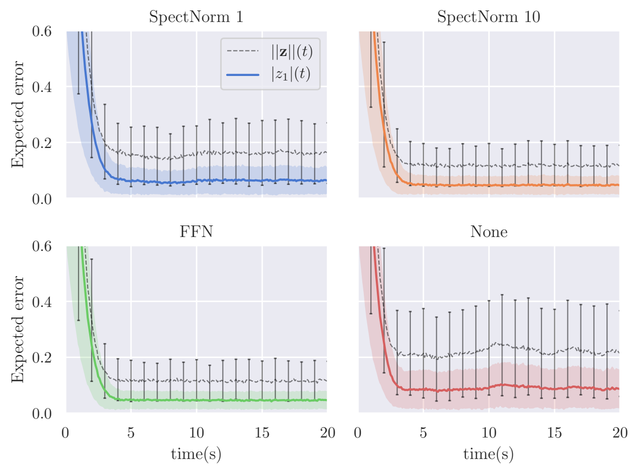

Fig. 4: The expected error of each controller at every time-step of the test episodes. The colored lines represent the absolute error in the first state, while the stapled lines are the mean squared error of the state vector. The error bands and bars represent one standard deviation above and below the mean. Here, the noise standard deviation and truncation point is $\sigma = 0.1$.

One of the cornerstone results of this work is the boundedness of the estimation error, which was assumed in both Theorem 1 and Theorem 2. In the experiments, it was assumed that ε could be approximated by the maximum prediction error on the test set. This assumption is reasonable when the samples in the test are representative of the data the estimator will encounter during operation. However, if the test set is small, or gathered from operation in a limited area of the state space, this approximation might be optimistic.

Another challenge is the assumptions made in Theorem 2 that the convergence bounds are dependent on the maximum rate of change of the state and input. It is relatively common to assume that the input change is bounded, but for the state change to be upper bounded, it must be required that the dynamics are bounded as well. This does not hold globally when parts of the dynamics are, for instance, linear or polynomial. If applying this controller, one must be certain that either, the update rate of the DNN estimate is faster than the system dynamics, or the dynamics are bounded.

A significant advantage of using this controller for trajectory tracking is that $\dot{x}_2$ can be measured at a high sampling rate with an accelerometer. From these measurements, it is possible to get ground truth values of the unkown dynamics, by utilizing $d(\boldsymbol{x}, u) = bu - \dot{x}_2$. This real life data would undoubtedly be affected by measuring noise, which is not considered in the training data in this work. However, it is it reasonable to believe that regularization by spectral normalizaion will be beneficial when measuring noise is present, as it prevents overfitting.

In addition to measuring noise, it is probable that we would have imprecice knowledge of b when working with a physical system. This would have to be taken into account, as it would affect the current stability analysis.

VII. Conclusion and Further Work

This work attempts to further the sound unification of data-driven methods and traditional control theory.

A feedback controller with a neural network estimating unknown dynamics is suggested. The controller is proven to stabilize the closed-loop error dynamics of a wide class of two-dimensional systems in a trajectory tracking problem.

Moreover, the controller globally and exponentially drives the error to a ball centered at the origin. A specific upper bound on the radius of the error ball is found. Measuring noise and the update rate of the neural network estimate influence the size of the convergence bound. So does the Lipschitz constant of the neural network, and we therefore argue that the regularization approach known as spectral normalization can be beneficial when utilizing neural networks for control.

Simulations of a mass-spring-damper system affected by an unknown input nonlinearity confirm the theoretical findings. Experiments show that the Lipschitz constant influences the noise rejection capabilities of the controller, as well as its ability to handle a low update rate of the neural network estimate. The experiments also demonstrate the tradeoff between bias and variance when restricting the Lipschitz constant; a lower Lipschitz constant bound allows for less flexibility and might affect network accuracy.

In this work, only two-dimensional systems in strict-feedback form are considered. In further work, we recommend investigating how the proofs given here can be extended to a wider class of systems. Also, imprecisely known input gains is a phenomenon that will likely be present when working with physical systems. Therefore, considering this in the stability analysis would be an advantageous extension of the current work.

Acknowledgments

This work was supported by the industry partners Borregaard, Elkem, Hydro, Yara and the Research Council of Norway through the project TAPI: Towards Autonomy in Process Industries, project number 294544.

References

[1] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensors in the process industry," *Computers & Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[2] L. Deng and Y. Liu, "Deep Learning in Natural Language Processing", 1st ed. Springer Singapore, 2018.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 2012 Annual Conference on Neural Information Processing Systems 2012*, 2012.

[4] T. P. Lillicrap, J. J. Hunt, A. e. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", Sep. 2015. arXiv: `1509.02971 [cs.LG]`.

[5] A. F. S. Moosavi-Dezfooli and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proc. 2016 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2016.

[6] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *Proc. 2018 International Conference on Learning Representations*, OpenReview.net, 2018.

[7] T. B. H. Shiori Sagawa Pang Wei Koh and P. Liang, "Distributionally robust neural networks," in *Proc. 2020 International Conference on Learning Representations*, OpenReview.net, 2020.

[8] D. Limon, J. Calliess, and J. Maciejowski, "Learning-based nonlinear model predictive control," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7769–7776, 2017.

[9] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. 2018 IEEE Conference on Decision and Control*, IEEE, 2018.

[10] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. 2017 Annual Conference on Neural Information Processing Systems*, 2017.

[11] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proc. 2018 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 2018.

[12] S. Moe, F. Remonato, E. I. Grøtli, and J. T. Gravdahl, "Linear antisymmetric recurrent neural networks," in *Proc. 2020 Annual Conference on Learning for Dynamics and Contro*, ser. Proceedings of Machine Learning Research, vol. 120, PMLR, 2020.

[13] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Proc. 2018 Annual Conference on Neural Information Processing Systems*, 2018.

[14] M. Ciccone, M. Gallieri, J. Masci, C. Osendorfer, and F. Gomez, "Nais-net: Stable deep networks from non-autonomous differential equations," in *Proc. 2018 Annual Conference on Neural Information Processing Systems*, 2018.

[15] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *Proc. 2019 International Conference on Robotics and Automation*, IEEE, 2019.

[16] H. K. Khalil, "Nonlinear Systems", 3rd ed. Edinburgh Gate: Pearson, 2015, vol. 4.

[17] "Matlab version 9.7.0 (r2019b)", Natick, Massachusetts, USA, 2019.

[18] "Python 3.6", `https://www.python.org/`.

[19] F. Chollet *et al.*, "Keras 2.3.1", `https://keras.io`, 2015.