Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 58-15 (2024) 253–258

State Derivative Normalization for Continuous-Time Deep Neural Networks

Jonas Weigand * **Gerben I. Beintema** ** **Jonas Ulmen** *** **Daniel Görges** *** **Roland Tóth** ** **Maarten Schoukens** ** **Martin Ruskowski** *

* *Chair of Machine Tools and Control Systems, RPTU Kaiserslautern, and the German Research Center for Artificial Intelligence, Kaiserslautern, Germany (e-mail: jonas@weigand-weigand.de and martin.ruskowski@rptu.de, ORCID: 0000-0001-5835-3106, 0000-0002-6534-9057)*

** *Control Systems (CS) Group at the Department of Electrical Engineering, Eindhoven University of Technology, Netherlands. R. Tóth is also affiliated to the Systems and Control Laboratory at the Institute for Computer Science and Control, Budapest, Hungary (e-mail: g.i.beintema@tue.nl, r.toth@tue.nl and m.schoukens@tue.nl, ORCID: 0000-0002-7822-6283, 0000-0001-7570-6129, 0000-0002-4904-1255)*

*** *Institute for Electromobility, RPTU Kaiserslautern, Germany (e-mail: jonas.ulmen@rptu.de and daniel.goerges@rptu.de, ORCID: 0000-0003-1597-1523, 0000-0001-5504-0972)*

Abstract: The importance of proper data normalization for deep neural networks is well known. However, in continuous-time state-space model estimation, it has been observed that improper normalization of either the hidden state or hidden state derivative of the model estimate, or even of the time interval can lead to numerical and optimization challenges with deep learning based methods. This results in a reduced model quality. In this contribution, we show that these three normalization tasks are inherently coupled. Due to the existence of this coupling, we propose a solution to all three normalization challenges by introducing a normalization constant at the state derivative level. We show that the appropriate choice of the normalization constant is related to the dynamics of the to-be-identified system and we derive multiple methods of obtaining an effective normalization constant. We compare and discuss all the normalization strategies on a benchmark problem based on experimental data from a cascaded tanks system and compare our results with other methods of the identification literature.

Keywords: System Identification, Continuous-Time, Neural Ordinary Differential Equations, Nonlinear State-Space

1. INTRODUCTION

Machine learning is a powerful tool for time series modeling and system identification (Schoukens and Ljung, 2019). Models can be categorized into *Discrete-Time* (DT) models and *Continuous-Time* (CT) models such as *Neural Ordinary Differential Equations* (NODE) (Chen et al., 2018), *Runge-Kutta Neural Networks* (RKNN) (Wang and Lin, 1998), Deep Encoder Networks (Beintema et al., 2023) or *Liquid Time Constant Neural Networks* (LTC) (Hasani, 2020). In control engineering, CT models are often preferred, as they are closely related to physical models, they are often found more attractive for designing controllers since shaping performance with such models is more intuitive, and they can be used under irregular sampling.

Data normalization is a key data preprocessing step in modern machine learning approaches. The core idea of normalization is to map the input and output data to a numerically favorable range (e.g. zero-mean and standard deviation of 1) (Ba et al., 2016). This is important to (*i*) comply with the underlying assumption in many parameter initialization strategies for *Neural Networks* (NN's) (e.g., Xavier initialization (Glorot and Bengio, 2010)) and to (*ii*) numerically improve the effective numerical range of the gradients for the nonlinear activation functions (Ioffe and Szegedy, 2015).

Prior research demonstrates the significant enhancement in model performance through *State Derivative Normalization* (SDN) as evidenced by (Weigand et al., 2021; Beintema et al., 2023). Despite these advancements, a comprehensive exploration and analysis of SDN's mechanisms remain absent from published literature. In addition, practical methodologies to estimate the normalization factor are missing in the literature.

- Consequently, this paper analyses the state, the state derivative, and the time normalization as well as their coupling. Additional graphical interpretation and the connection to *Ordinary Differential Equations* (ODE) solvers are presented.

⋆ The project RACKET supported this research under grant 01IW20009 by the German Federal Ministry of Education and Research.

10.1016/j.ifacol.2024.08.537

- Furthermore, three approaches are proposed to estimate this normalization factor and these approaches are compared on a well-studied benchmark example.

The remainder of the paper is structured as follows. Section 2 introduces the considered identification problem. We present multiple interpretations of the use of a normalization constant in Section 3. Section 4 defines the criteria for an effective normalization and provides three methods for practical implementation. Section 5 applies the method to a real-world benchmark problem, and in Section 6 concluding remarks on the established results are made.

2. THE IDENTIFICATION PROBLEM

2.1 Data-generating System

Consider a nonlinear system that is represented by a nonlinear state-space representation

$$\dot{x}(t) = f\left(x(t), u(t)\right) \tag{1a}$$
$$y_k = h(x_k, u_k) + w_k, \tag{1b}$$
$$x(0) = x_0, \tag{1c}$$

where $t \in \mathbb{R}_{\geq 0}$ is the continuous time, $x(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the state associated with (1) with $n_\mathrm{x} \in \mathbb{N}$, $x_0 \in \mathbb{R}^{n_\mathrm{x}}$ being the initial state, $u(t) \in \mathbb{R}^{n_\mathrm{u}}$ is the exogenous input, and $f : \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}} \to \mathbb{R}^{n_\mathrm{x}}$ is locally Lipschitz continuous ensuring that all solutions of x are forward complete. With respect to the output equation, $y(k) \in \mathbb{R}^{n_y}$ stands for the sampled system output at time moments kT_s with $T_\mathrm{s} > 0$ being the sampling interval and $k \in \mathbb{Z}_{\geq 0}$, $x_k = x(kT_\mathrm{s})$ and $u_k = u(kT_\mathrm{s})$, while $w_k \in \mathbb{R}^{n_y}$ is an i.i.d. zero-mean white noise process with finite variance Σ_w. Such a hybrid process and noise model is often used in CT nonlinear system identification (Schoukens and Ljung, 2019).

2.2 Identification

For the considered system class, the CT model identification problem can be expressed for a given data set of measurements

$$D_N = \{(u_0, y_0), (u_1, y_1), ..., (u_{N-1}, y_{N-1})\}, \tag{2}$$

generated by (1), with unknown w_k, $x(t)$, $\dot{x}(t)$, and initial state x_0, as the following optimization problem (a.k.a. ℓ_2 simulation loss minimization)

$$\begin{aligned} \min_{\theta, \hat{x}_0} \quad & \frac{1}{N} \sum_{k=0}^{N-1} \|y_k - \hat{y}_k\|_2^2, \\ \text{s.t.} \quad & \hat{y}_k = h_\theta(\hat{x}(kT_\mathrm{s})), \\ & \dot{\hat{x}}(t) = f_\theta(\hat{x}(t), u(t)), \end{aligned} \tag{3}$$

where $\hat{x}(t) \in \mathbb{R}^{n_\mathrm{x}}$ is the model state, h_θ and f_θ are the output and state-derivative functions parameterized by $\theta \in \mathbb{R}^{n_\theta}$. These two functions are represented by multi-layer feedforward NN's within this paper. The state network $f_\theta(\cdot)$ input layer nodes are considered as model state $\hat{x}(t)$ and exogenous variables $u(t)$, and the output layer nodes are considered as model state derivative $\dot{\hat{x}}(t)$ (Schoukens, 2021; Suykens et al., 1995). However, the presented results also hold when other function approximators such as polynomials are considered.

3. INTERPRETATION OF THE NORMALIZATION

State Derivative Normalization **(SDN)** The idea is the introduction of a normalization factor τ, which scales the state-derivative equation of the state-space model during training:

$$\dot{\hat{x}}(t) = \frac{1}{\tau} f_{\text{NN}}(\hat{x}(t), u(t)) \tag{4}$$

The normalization can be implemented as a scalar normalization $\tau \in \mathbb{R}_{>0}$, or as a vector $\tau \in \mathbb{R}_{>0}^{n_\mathrm{x}}$ corresponding to elements of the output of the hidden state network f_{NN}. In the latter case, (4) is defined as $\dot{\hat{x}}(t) = \mathrm{diag}(\frac{1}{\tau_1}, \ldots, \frac{1}{\tau_{n_\mathrm{x}}}) f_\theta(\hat{x}(t), u(t))$. This linear scaling can also be interpreted as changing the weight initialization of the output layer of $f_{\text{NN}}(\cdot, \cdot)$.

It is clear that τ in (4) scales the hidden state derivative. Hence, it can be used to normalize the state derivatives. Next, we show that the derivative normalization is inherently coupled to the scaling of the state x and the scaling of the time t.

State Normalization. It is also possible to rewrite (4) as:

$$\frac{d(\tau \hat{x}(t))}{dt} = f_{\text{NN}}(\hat{x}(t), u(t)) \tag{5a}$$
$$\tilde{x}(t) \triangleq \tau \hat{x}(t) \tag{5b}$$
$$\frac{d\tilde{x}(t)}{dt} = f_{\text{NN}}(\tilde{x}(t)/\tau, u(t)) \tag{5c}$$

In this way, normalization can be interpreted as the normalization of the magnitude of the state.

Time-Based Normalization. We can rewrite (4) as

$$\frac{d\hat{x}(t)}{d(t/\tau)} = f_{\text{NN}}(\hat{x}(t), u(t)), \tag{6a}$$
$$\tilde{t} \triangleq t/\tau, \tag{6b}$$
$$\frac{d\hat{x}(\tilde{t}\tau)}{d\tilde{t}} = f_{\text{NN}}(\hat{x}(\tilde{t}\tau), u(\tilde{t}\tau)), \tag{6c}$$

which suggests that the normalization can be viewed as a scaling of time by a factor τ.

This time rescaling can also be viewed as changing the effective integration length in ODE solvers. For instance,

$$\hat{x}(t) = \hat{x}(0) + \int_0^t \frac{1}{\tau} f_{\text{NN}}(\hat{x}(t'), u(t'))\, dt' \tag{7a}$$
$$= \hat{x}(0) + \int_0^{t/\tau} f_{\text{NN}}(\hat{x}(\tau\tilde{t}'), u(\tau\tilde{t}'))\, d\tilde{t}' \tag{7b}$$

using the substitution of $\tilde{t}' = t'/\tau$. This integration by substitution shows that the normalization can be viewed as rescaling the effective integration time from t to t/τ. This can be translated directly to a wide range of numerical integration schemes by rescaling the integration length and step size simultaneously.

Graphical intuition about the time rescaling can be gained from output data of the *Cascaded Tank System* (CTS) benchmark in Fig. 1 (Schoukens et al., 2016). The measured data (black) of the water level is given at a sample rate of $T_\mathrm{s} = 4.0\,\mathrm{s}$, the scaled data (red) is transferred to a sample time of $T_m = T_\mathrm{s}/\tau = 1.0$, while the number of time

steps remains unchanged. The original data is transferred to the scaled time domain, where the model is trained and evaluated, and the results are transferred back to the original time grid.

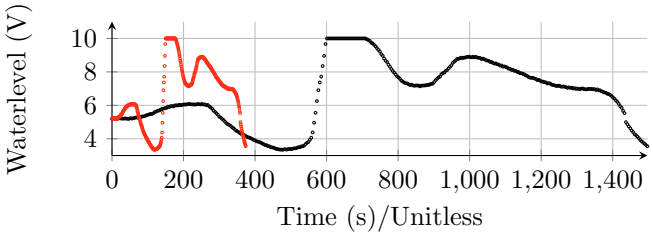

Fig. 1. Time scaling of the output measurement of the *Cascaded Tank System* (CTS) benchmark (Schoukens et al., 2016). Black: Original measurement (unit seconds). Red: Time-domain scaled data (unitless).

It is known that the stiffness of an ODE system is subject to temporal variation. However, employing a constant (linear) normalization across all stiffness regions is essential from a practical model inference point of view. Moreover, averaging across multiple regions increases the robustness of the estimation by providing a broader range of suitable normalization factors. Our subsequent experiments demonstrate that there exists a broad valley of good normalization factors (Fig. 3). Remarkably, all normalization factors within this range lead to high-performance models.

The key takeaway of representations (4), (6) and (5) is that SDN collectively influences the hidden state, the hidden state derivative, and the scaling of time t.

4. ESTIMATION OF THE NORMALIZATION FACTOR

Since the system dynamics are unknown a priori, selecting a normalization factor τ that results in a favorable numerical performance of the training algorithm is challenging a priori, it has to be estimated during training, from the available data, or by heuristics. We propose three methods:

- make the normalization factor a trainable parameter,
- estimate τ using cross-validation,
- use a heuristic-based approach starting from a linear approximation of the system.

Trainable Parameter. The normalization factor τ can be added to the set of parameters present in the optimization problem (3). This results in a simultaneous optimization of τ, θ, and $\hat{x}_0$ under the ℓ_2 simulation loss. For this method, a vectorized normalization leads to more optimization parameters than using a single scalar.

In terms of implementation, $\tau > 0$ should hold to fulfil its role as a magnitude normalization. As most NN libraries do not provide constrained optimization one can implement

$$\hat{\tau} = \max(\epsilon, \tau) \tag{8}$$

with a small positive number $\epsilon \in \mathbb{R}_{>0}$. Empirical evidence demonstrates that the trainable normalization quickly reaches a stationary value.

From a pure mathematical perspective, making τ a trainable parameter may seem to have little impact because the expressive power of the NN could easily compensate for it. Practically, the optimizer would need to adapt a large quantity of parameters instead of a single normalization factor. Additionally, traditional NN initialization methods assume inputs and outputs are within a specific range. We consolidated the normalization dependency into a single trainable parameter τ, making the training process simpler and more robust, as demonstrated in our experiments.

Overall, this method offers high adaptability across different data sets and NN configurations, an easy implementation, and demands only little additional computational load.

Cross-validation. The normalization factor τ can be considered as a hyperparameter and estimated using cross-validation methods. For example, (3) is optimized using a training data set for a chosen set of values of τ, a validation data set is used subsequently to determine the best performing τ value. This method is universally applicable to nonlinear systems, is easy to implement if a hyperparameter tuning is already available, and, given sufficient function evaluations, ensures global optimality.

Heuristic approach based on an approximate model. This heuristic is derived based on a guarantee of the existence of a normalized model formalized by (Beintema et al., 2023) which states;

Theorem 1. Given a input trajectory $u(t)$, a non-constant state trajectory $x(t)$ which satisfies $\dot{x}(t) = f(x(t), u(t))$ as in (1) for all $t \in [0, L]$ than there exists a $\tau \in \mathbb{R}^+$ and a scalar state transformation $\gamma\hat{x}(t) = x(t)$ such that both the model state trajectory $\hat{x}(t)$ and model derivative f_{NN} given by $\dot{\hat{x}}(t) = \frac{1}{\tau} f_{\text{NN}}(\hat{x}(t), u(t))$ as in (4) are normalized as

$$\operatorname{var}(\hat{x}) \triangleq \frac{1}{L}\int_0^L \frac{1}{n_{\text{x}}}\|\hat{x}(t)\|_2^2 dt = 1, \tag{9a}$$

$$\operatorname{var}(f_{\text{NN}}(\hat{x}, u)) = 1. \tag{9b}$$

Proof. With

$$\gamma = \sqrt{\operatorname{var}(x)} \tag{10}$$

$$\tau = \sqrt{\operatorname{var}(x)/\operatorname{var}(\dot{x})} \tag{11}$$

the normalization conditions are satisfied, as shown below

$$\operatorname{var}(\hat{x}) = \operatorname{var}(x/\gamma) = \operatorname{var}(x)/\gamma^2 = 1,$$

$$\operatorname{var}(f_{\text{NN}}(\hat{x}, u)) = \operatorname{var}(\tau\dot{\hat{x}}) = \operatorname{var}(\tau\dot{x}/\gamma) = \tau^2/\gamma^2\operatorname{var}(\dot{x}) = 1.$$

We base our heuristic on the relationship of (11). We make two alterations *(i)* the integral over time can be approximated by a summation over the time samples to make this tractable and *(ii)* in (11) we use an approximate model instead of the system equations.

We have observed that using a linear approximate model based on the *Best Linear Approximation* (BLA) of a nonlinear system can get a sufficiently accurate estimate of the optimal τ. The BLA offers a linear approximation of a nonlinear system, best in the mean-squared-error sense, based on measured input-output data (Pintelon and Schoukens, 2012). Estimating a BLA of a potentially nonlinear system is commonly done using *Prediction Error Minimization* (PEM), minimizing a least squares prediction error. Using an estimated BLA, an estimate of the state and state-derivatives, x_{BLA} and $\dot{x}_{\text{BLA}}$, can be obtained to compute τ through (11):

$$\tau_{\text{BLA}} = \sqrt{\text{var}(x_{\text{BLA}})/\text{var}(\dot{x}_{\text{BLA}})}. \tag{12}$$

This method is especially valuable for dynamic systems that can be reasonably well represented by a linear model for the considered range of excitation. Also note that any linear transformation that can be present in the BLA state and state derivative estimate does not affect the resulting τ estimate in (12). Additionally, besides normalization, the BLA can be used as a good NN weight initialization candidate (Schoukens and Tóth, 2020).

Frequency-domain analysis of this approach offers additional insight into the optimal choice of the normalization factor. Consider that the input signal has a periodicity of N samples, then one can decompose the input into its *Discrete Fourier Transform* (DFT) components as:

$$u_k = \frac{1}{N}\sum_{m=0}^{N-1} U_m e^{j\frac{2\pi}{N}mk} \tag{13}$$

$$U_m = \sum_{k=0}^{N-1} u_k e^{-j\frac{2\pi}{N}mk} \tag{14}$$

Both the variance of the state and state-derivative can be expressed using these DFT components as:

$$\text{var}(x) = \frac{1}{L}\int_0^L \|x(t)\|_2^2 dt \sim \sum_{m=-\infty}^{\infty} \|X_m\|_2^2 \tag{15a}$$

$$\text{var}(\dot{x}) = \frac{1}{L}\int_0^L \|\dot{x}(t)\|_2^2 dt \sim \sum_{m=-\infty}^{\infty} \omega_m^2\|X_m\|_2^2 \tag{15b}$$

where $\omega_m = 2\pi\frac{m}{NT_s}$.

Since we are considering the BLA, the Fourier components of the state can be written in terms of the input-to-state frequency response function $X_m = G(j\omega_m)U_m$, assuming a single input system for simplicity. Therefore, substituting this and (15) into (12) results in

$$\tau_{\text{BLA}} = \sqrt{\frac{\sum_{m=-\infty}^{\infty} U_m^H G^H(j\omega_m)G(j\omega_m)U_m}{\sum_{m=-\infty}^{\infty} \omega_m^2 U_m^H G^H(j\omega_m)G(j\omega_m)U_m}}, \tag{16}$$

where $\cdot^H$ denotes the Hermitian operation. Hence, if the signal $u(t)$ only consists of a single sine wave (i.e. only one nonzero U_m) then $\tau_{\text{BLA}} = 1/\omega_m$. Furthermore, if $u(t)$ or $G(j\omega_m)$ has a finite bandwidth bounded by Ω then $\tau_{\text{BLA}} \geq 1/\Omega$.

5. EXPERIMENTAL RESULTS

Benchmark. We apply the proposed methods to the CTS benchmark (Schoukens et al., 2016). The setup is depicted in Fig. 2. It consists of two vertically mounted tanks, where the upper one has a water inflow using a pump, and the water flows from the upper tank into the lower one. The task is to estimate a model that can predict the water level of the lower tank given a pump input sequence. The experiment incorporates an overflow of the tank, which introduces a hard saturation function.

Model Configuration. We chose the fixed-step Runge-Kutta 4 ODE solver. The network $f_{\text{NN}}(\cdot,\cdot)$ is chosen to consist of linear layers with matrices $A \in \mathbb{R}^{n_\text{x}\times n_\text{x}}$, $B \in \mathbb{R}^{n_\text{x}\times n_\text{u}}$ and two residual hidden layers with a Leaky ReLU activation function $\sigma(\cdot)$ and 64 hidden units for each layer. Weight matrices are denoted as $W_{(\cdot)}$ and bias terms as

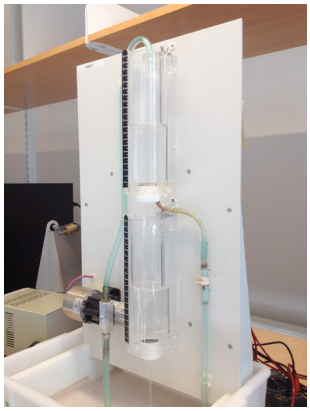

Fig. 2. Picture of the *Cascaded Tank System* (CTS) (Schoukens et al., 2016).

$b_{(\cdot)}$ with appropriate dimensions. Bias terms for the linear layer are disabled. An output network $g_{\text{NN}}(\cdot,\cdot)$ is also used with the same structure as the state network $f_{\text{NN}}(\cdot,\cdot)$. The overall NN model can be written as:

$$\begin{aligned}\frac{dx(t)}{dt} &= f_{\text{NN}}(x(t),u(t)) = Ax(t)+Bu(t) \\ &+ W_{F1}\sigma\left(W_{F2}\sigma\left(W_{F3}\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+b_{F3}\right)+b_{F2}\right)\end{aligned} \tag{17a}$$

$$\begin{aligned}y(t) &= g_{\text{NN}}(x(t),u(t)) = Cx(t)+Du(t) \\ &+ W_{G1}\sigma\left(W_{G2}\sigma\left(W_{G3}\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+b_{G3}\right)+b_{G2}\right)\end{aligned} \tag{17b}$$

More details on the general model structure are given in (Schoukens, 2021; Suykens et al., 1995). White box modeling of the CTS would lead to 2 states. Nevertheless, we set the number of states $n_\text{x} = 4$. It is observed that this results in better learning behavior. This is motivated by the effect of state augmentation (Dupont et al., 2019), corresponding to the fact that increasing the state dimension allows to describe the system behavior with less complex nonlinearities involved (also expressed by the immersion concept in nonlinear system theory). The initial hidden states are obtained using a *Deep Encoder Network* (Beintema et al., 2023) $e_{\text{NN}}(\cdot,\cdot)$ with $n_\text{a} = n_\text{b} = 5$. Weight matrices are initialized with a small random number chosen from a uniform distribution $\mathcal{U}[-0.01, 0.01]$, while bias terms are initialized to zero. The scalar normalization is initialized with 1 and the vectorized normalization with $1/n_\text{x}$. A barrier function L_N is applied to the linear layer A of the state network to ensure negative definiteness (Weigand et al., 2021). This barrier minimizes drifts over long simulation horizons and is estimated using the differentiable Sylvester Criterion. Furthermore, we apply a *Differential Algebraic Equations* (DAE) network

$$\begin{aligned}&d_{\text{NN}}(x(t),u(t)) = \\ &W_{D1}\sigma\left(W_{D2}\sigma\left(W_{D3}\begin{bmatrix}x(t)\\u(t)\end{bmatrix}+b_{D3}\right)+b_{D2}\right)+b_{D1}\end{aligned} \tag{18}$$

to account for the hard state saturation, which is trained using an additional penalty function L_D. It does not affect the forward model evaluation. The optimization problem is given by

$$
\begin{aligned}
\min_{\theta} \quad & L_{\mathrm{N}} + L_{\mathrm{W}} + \\
& \sum_{k=\max(n_{\mathrm{a}},n_{\mathrm{b}})}^{K-J} \left(\sum_{j=0}^{J-1} \|y_{k+j} - \hat{y}_{k+j|k}\|_2^2 + L_{\mathrm{D},j} \right), \\
\text{s.t.} \quad & \hat{y}_{k+j|k} = g_{\mathrm{NN}}(x_{k+j|k}), \\
x_{k+j+1|k} &= \mathrm{ODE\_Solve}\left(x_{k+j|k}, u_{k+j}, \frac{1}{\tau} f_{\mathrm{NN}}(\cdot,\cdot), T_{\mathrm{s}} \right), \\
& x_{k|k} = e_{\mathrm{NN}}(u_{k-1}, ..., u_{k-n_b}, y_{k-1}, ..., y_{k-n_a}), \\
& L_{D,j} = \lambda_{\mathrm{D}} \left(d_{\mathrm{NN}}(x_{j|j}, u_{j|j}) \right)^2, \\
& L_{\mathrm{N}} = \begin{cases} 0 & \text{if } A \prec 0 \\ \lambda_{\mathrm{N}} & \text{otherwise.} \end{cases}, \\
& L_{\mathrm{W}} = \lambda_{\mathrm{W}} \|\theta\|_2^2
\end{aligned}
$$

with $\lambda_{\mathrm{N}} = 10^{12}$ and $\lambda_{\mathrm{D}} = 10^3$. The bar notation $x_{k+j|k}$ reads as "The simulated state at x_{k+j} starting at k with initial state $x_{k|k}$". The notation $\|\cdot\|_2^2$ stands for the squared Euclidean norm. The sampling time is $T_{\mathrm{s}} = 4$ s.

The implementation is written in Python, using PyTorch. Furthermore, the ADAM optimizer is applied with unmodified configuration except for the learning rate, which is set to 0.003 for the first 1,000 steps, and to 0.0009 until step 3,000, and to 0.00027 for all subsequent iterations. The maximum number of optimization steps is set to 20,000. Regularization is obtained using weight decay of $\lambda_{\mathrm{W}} = 10^{-8}$ and early stopping when the best validation error does not improve for 2,000 iterations. We do not apply *Batch-Norm* or *Dropout*. Input and output data are z-score normalized. As no explicit validation data set is given for CTS, we apply the first 512 time steps of the test data for early stopping. Test data is not accessed for any other reason. Training is performed in 64 mini-batches with a sequence length of $J = 128$ steps using *Truncated Simulation Error Minimization* (TSEM) (Forgione and Piga, 2021).

Effect of Normalization. We analyze the effect of the normalization factor τ given the CTS benchmark, a fixed NN configuration in simulation mode (free-run simulation/simulation error), and a fixed training pipeline throughout all experiments. In addition to the proposed state derivative normalization, the magnitude of the data (waterlevel) is normalized, too. Performance is measured in terms of the *Root Mean Squared Error* (RMSE) of the simulation error w.r.t. the test data:

$$
e_{\mathrm{RMSE}} = \sqrt{\frac{1}{K - \max(n_{\mathrm{a}}, n_{\mathrm{b}})} \sum_{k=\max(n_{\mathrm{a}},n_{\mathrm{b}})}^{K-1} \|y_k - \hat{y}_k\|_2^2}.
$$

Fig. 3 displays a grid search for different scalar values of T_{s}/τ fixed prior to the model training on a log scale ($T_{\mathrm{s}} = 4\,\mathrm{s}$ for CTS). Each experiment is repeated 20 times to account for and observe the effect of randomness in the initial weights. We observe in Fig. 3 that both the performance and variance of the results get worse for large and small values of τ. Furthermore, Fig. 3 indicates that there exists a desirable optimum. The optimum is different from the unscaled function estimation with $\tau = 1$, which corresponds to $T_{\mathrm{s}}/\tau = 4.0$ for CTS in Fig. 3. The median RMSE for $T_{\mathrm{s}}/\tau = 4.0$ is $0.75(V)$ and the mean RMSE is $1.98(V)$, considerably larger than the normalized result.

The model outputs obtained with the proposed estimators for τ are displayed in Figure 4.

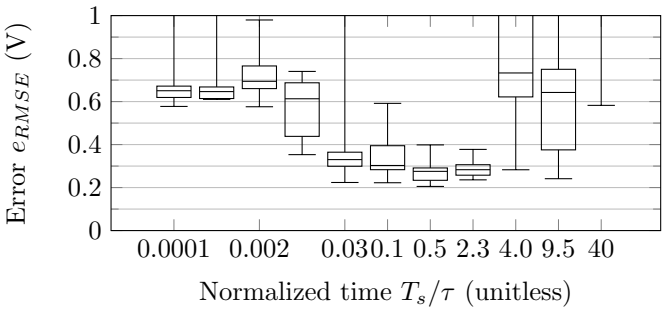

Fig. 3. 200 independent experiments, each with the same configuration except for the random weight initialization and a fixed scalar normalization factor. We tested 10 different normalization factors repeated 20 times each. The box plot displays the median, lower quartile, upper quartile, minimum and maximum values. Note that experiments with a normalization $T_{\mathrm{s}}/\tau = 40$ sometimes lead to unstable results, with RMSE $> 10^9$.

Using the cross-validation method to estimate the normalization, we observe in Fig. 3 good results between $T_{\mathrm{s}}/\tau = 0.031$ and $T_{\mathrm{s}}/\tau = 2.276$, with a best RMSE at $T_{\mathrm{s}}/\tau = 0.543$. For the trainable normalization method, we define a normalization vector τ as a trainable parameter and implement (8). It is not optimized with hyperparameter tuning but jointly trained with all NN weights, using the same learning rate. The normalization is initialized to $T_{\mathrm{s}}/\tau = 0.1$. After training 20 models, we obtain an average normalized time constant $T_{\mathrm{s}}/\tau = 0.072 \pm 0.048$ (mean $\pm$ std). Using the BLA and (12), we estimate a normalization factor $T_{\mathrm{s}}/\tau = 0.054$, which matches the results in Fig. 3.

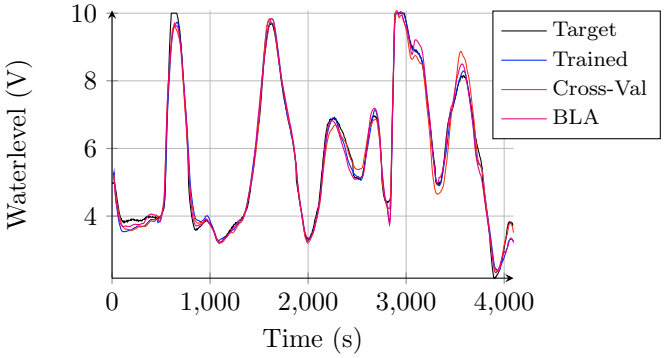

Fig. 4. Simulation results on the CTS benchmark (Schoukens et al., 2016). Results of the best models obtained with the trained normalization factor, the cross-validation method, and the *Best Linear Approximation* are displayed.

Comparison to Literature. Since several black-box identification methods have been applied to the CTS in the literature, we can compare our method to the black-box approaches with the best performance (in terms of RMSE on the test data). This comparison is provided in Table 1. The following approaches are compared: (Relan et al., 2017) estimates a BLA and develops an unstructured *Nonlinear State Space Model* (NLSS) with different initialization schemes. A nonparametric Volterra series model

is estimated in (Birpoutsoukis et al., 2018). A nonlinear state-space model based on Gaussian processes is applied in (Svensson and Schön, 2017). As a direct comparison, continuous-time NN on this modeling problem has been applied in (Beintema et al., 2023; Forgione and Piga, 2021; Mavkov et al., 2020; Weigand et al., 2021). These works emphasize initial state estimation, fitting criteria, and model stability differently. It can be observed that the proposed approach outperforms all other black-box identification approaches.

Table 1. Results for the Cascaded Tank Benchmark.

Method	Test data e_{RMSE} (V)
Best Linear Approximation	0.75
Truncated Volterra Model	0.54
State-space with GP-inspired Prior	0.45
Integrated Neural Networks	0.41
Soft-constrained Integration Method	0.40
Stable Runge-Kutta Neural Network	0.39
Nonlinear State Space Model	0.34
Truncated Simulation Error Minimization	0.33
Deep Subspace Encoder	0.22
ours (SDN, Trained Parameter, best model)	0.2151
ours (SDN, Trained Parameter, mean ± std)	0.2977 ± 0.1259
ours (SDN, Cross-Validation, best model)	0.2054
ours (SDN, Cross-Validation, mean ± std)	0.2777 ± 0.052
ours (SDN, BLA, best model)	0.2253
ours (SDN, BLA, mean ± std)	0.2633 ± 0.0284

6. CONCLUSION

We have shown the importance of proper state normalization when considering continuous-time modeling with state-space NN's. This is handled by introducing a normalization constant in front of the state derivative network. We have provided a state domain, a state derivative domain, and a time domain interpretation of this concept, and showed the beneficial effects of such a normalization. To estimate the appropriate normalization constant, three approaches based on a trainable parameter, cross-validation, and BLA have been proposed to ensure the practical applicability of the normalization. Based on simulation studies, we have shown that the proposed methodologies enable to improve model estimation with NN-based state-space modeling methods.

REFERENCES

Ba, J.L., Kiros, J.R., and Hinton, G.E. (2016). Layer normalization. URL `http://arxiv.org/pdf/1607.06450v1`.

Beintema, G.I., Schoukens, M., and Tóth, R. (2023). Continuous-time identification of dynamic state-space models by deep subspace encoding. In *The Eleventh International Conference on Learning Representations*.

Birpoutsoukis, G., Csurcsia, P.Z., and Schoukens, J. (2018). Efficient multidimensional regularization for volterra series estimation. *Mechanical Systems and Signal Processing*, 104, 896–914. doi:10.1016/j.ymssp.2017.10.007.

Chen, R.T.Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *Advances in neural information processing systems*, 31. URL `http://arxiv.org/pdf/1806.07366v5`.

Dupont, E., Doucet, A., and Teh, Y.W. (2019). Augmented neural ODEs. *Advances in neural information processing systems*, 32. URL `http://arxiv.org/pdf/1904.01681v3`.

Forgione, M. and Piga, D. (2021). Continuous-time system identification with neural networks: Model structures and fitting criteria. *European Journal of Control*, 59, 69–81. doi:10.1016/j.ejcon.2021.01.008.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Hasani, R. (2020). *Interpretable Recurrent Neural Networks in Continuous-time Control Environments*. Ph.d. thesis, TU Wien.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. PMLR.

Mavkov, B., Forgione, M., and Piga, D. (2020). Integrated neural networks for nonlinear continuous-time system identification. *IEEE Control Systems Letters*, 1. doi:10.1109/LCSYS.2020.2994806.

Pintelon, R. and Schoukens, J. (2012). *System identification: a frequency domain approach*. John Wiley & Sons.

Relan, R., Tiels, K., Marconato, A., and Schoukens, J. (2017). An unstructured flexible nonlinear model for the cascaded water-tanks benchmark. *IFAC-PapersOnLine*, 50(1), 452–457. doi:10.1016/j.ifacol.2017.08.074.

Schoukens, J. and Ljung, L. (2019). Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6), 28–99.

Schoukens, M. (2021). Improved initialization of state-space artificial neural networks. In *2021 European Control Conference (ECC)*, 1913–1918. IEEE.

Schoukens, M., Mattsson, P., Wigren, T., and Noel, J.P. (2016). Cascaded tanks benchmark combining soft and hard nonlinearities. *Workshop on Nonlinear System Identification Benchmarks, Brussels, Belgium*.

Schoukens, M. and Tóth, R. (2020). On the initialization of nonlinear LFR model identification with the best linear approximation. *IFAC-PapersOnLine*, 53(2), 310–315. doi:10.1016/j.ifacol.2020.12.142.

Suykens, J.A., De Moor, B.L., and Vandewalle, J. (1995). Nonlinear system identification using neural state space models, applicable to robust control design. *International Journal of Control*, 62(1), 129–152.

Svensson, A. and Schön, T.B. (2017). A flexible state–space model for learning nonlinear dynamical systems. *Automatica*, 80, 189–199. doi:10.1016/j.automatica.2017.02.030.

Wang, Y.J. and Lin, C.T. (1998). Runge-kutta neural network for identification of dynamical systems in high accuracy. *IEEE Transactions on Neural Networks*, 9(2), 294–307.

Weigand, J., Deflorian, M., and Ruskowski, M. (2021). Input-to-state stability for system identification with continuous-time runge–kutta neural networks. *International Journal of Control*, 1–17. doi:10.1080/00207179.2021.1978555.