ROBUST RECURRENT NEURAL NETWORK TO IDENTIFY SHIP MOTION IN OPEN WATER WITH PERFORMANCE GUARANTEES - TECHNICAL REPORT

Daniel Frank, Decky Aspandi
Institute for Parallel and Distributed Systems
University of Stuttgart
{daniel.frank, decky.aspandi-latif}@ipvs.uni-stuttgart

Michael Muehlebach
Max Planck Institute for Intelligent Systems
michael.muehlebach@tuebingen.mpg.de

Benjamin Unger
Stuttgart Center for Simulation Science
benjamin.unger@simtech.uni-stuttgart.de

Steffen Staab
Institute for Parallel and Distributed Systems
University of Stuttgart
steffen.staab@ipvs.uni-stuttgart

ABSTRACT

Recurrent neural networks are capable of learning the dynamics of an unknown nonlinear system purely from input-output measurements. However, the resulting models do not provide any stability guarantees on the input-output mapping. In this work, we represent a recurrent neural network as a linear time-invariant system with nonlinear disturbances. By introducing constraints on the parameters, we can guarantee finite gain stability and incremental finite gain stability. We apply this identification method to learn the motion of a four-degrees-of-freedom ship that is moving in open water and compare it against other purely learning-based approaches with unconstrained parameters. Our analysis shows that the constrained recurrent neural network has a lower prediction accuracy on the test set, but it achieves comparable results on an out-of-distribution set and respects stability conditions.

Keywords Recurrent Neural Networks · System Identification · Linear Matrix Inequality Constraints · Deep Learning

1 Introduction

Traditional system identification often relies on domain-specific knowledge to obtain a representation of the target system. Thereby, complex or unknown effects are often neglected or simplified, which limits the accuracy of the mathematical model. In contrast, purely learning-based approaches have been shown to predict system states of an unknown nonlinear system with high accuracy. Due to a large number of parameters, it is assumed that deep neural networks can learn the internal dynamics, which are not directly visible in the input-output measurements. However, compared to models relying on differential equations, which are derived using traditional modeling techniques, it is much more difficult to satisfy stability requirements on the identified system. Recurrent neural networks (RNN) can handle input sequences and predict future system states of unknown systems, but they generally lack stability guarantees. From a robust control perspective, RNNs can be modeled as linear time-invariant systems with nonlinear disturbances. This perspective allows us to formulate constraints on the learnable parameters that guarantee finite gain and incremental finite gain stability. This is achieved by bounding the nonlinear activation function of the RNN with (incremental) quadratic constraints. We compare the constrained RNN with purely learning-based models such

as long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and unconstrained RNN [Goodfellow et al., 2016] that both achieve high prediction accuracy when learning the ship motion. We evaluate the different systems on within-distribution and out-of-distribution (OOD) test data. Our experiment results indicate that RNNs with parametric constraints do not achieve the same level of accuracy on within-distribution data as purely learning-based models without parametric constrains. However, they guarantee upper bounds on the finite (incremental) stability gain and are on par with an OOD evaluation.

Thus, our main contributions can be summarized as follows: (i) We apply a robust RNN model that guarantees a finite (incremental) stability gain to identify a real-world system (ship motion in open water) with multiple inputs and multiple outputs. (ii) We provide a comprehensive overview of robust RNNs, which is rooted in control theory and leads to guaranteed finite (incremental) stability gains. (iii) We evaluate robust RNNs and comparable learning-based approaches on ship motion data, including OOD measurements and empirically analyze the stability gains to support our theoretical results.

The article is structured as follows: In Section 2, we review existing approaches, whereas Section 3 introduces the theoretical background and the notation that is subsequently used. In Section 4, we introduce our robust RNN model and explain how it is trained. The model is evaluated and compared to other approaches from the literature in Section 5. The article concludes in Section 6.

2 Related Work

Deep recurrent neural network structures showed high prediction accuracy on nonlinear system identification tasks such as quadrotor- [Mohajerin and Waslander, 2019] or ship-motion [Baier et al., 2021]. Compared to classical identification approaches, which are extensively studied in Pillonetto et al. [2022], these learning methods usually do not provide stability guarantees on the input-output behavior. Separating the nonlinear activation function from the linear layers in a neural network allows us to use classical tools from robust control to analyze stability, which was already proposed by Suykens et al. [1995]. More recently, Fazlyab et al. [2019] used semi-definite programming to calculate a Lipschitz gain for deep neural networks. For sequence to sequence models that are used in system identification, Revay et al. [2021a] introduced convex parametric constraints to guarantee finite incremental gain stability for recurrent neural networks. In Revay et al. [2021b], this network structure is generalized to equilibrium networks, which do not require parametric constraints. In [Junnarkar et al., 2022], a recurrent neural network is used to control a partially unknown linear system, whereby closed-loop stability guarantees are established. An extension to recurrent equilibrium models is provided in Gu et al. [2022]. The incremental stability certificates achieved by the aforementioned approaches rely on a contraction argument. In contrast, Pauli et al. [2022] establishes finite gain stability by dissipativity arguments, which also allows for other performance specifications. In this work, we follow the approach of Revay et al. [2021a] and show finite gain stability using arguments from Pauli et al. [2022]. Compared to existing methods, we are evaluating the identification model on a real-world problem with multiple input and multiple outputs and compare against state of the art learning-based approaches [Mohajerin and Waslander, 2019].

3 Background

Notation

With $u = (u^k)_{k=0}^{\infty}$ we refer to a sequence of vectors. We distinguish two sequences u_a, u_b by sub-indices a, b. The k-th vector in the sequence u is denoted by $u^k \in \mathbb{R}^{n_u}$, where k denotes the discrete time index $k \in [0, \dots, T]$. The variables u_m^k and $u_{m,a}^k$ refer to the m-th component of the vectors u^k and $u_{\cdot,a}^k$, respectively. If no sub- or superscript is present, we refer to the entire sequence. In this work, we consider discrete-time systems and square-summable sequences which are from the set $\ell_{2e}^{n_u} := \{(u^k)_{k=0}^{\infty} \mid u^k \in \mathbb{R}^{n_u},\ \sum_{k=0}^{T} \|u^k\|^2 < \infty \text{ for all } T > 0\}$ with squared Euclidean norm $\|u^k\|^2 := \sum_{i=1}^{n_u} (u_i^k)^2 = (u^k)^{\mathrm{T}}(u^k)$. We refer to $\sum_{k=0}^{T} \|\cdot\|^2$ as the squared ℓ_2-norm and use the notation $\Delta u^k = u_a^k - u_b^k$ and $\Delta\psi(z^k) = \psi(z_a^k) - \psi(z_b^k)$ to denote the difference at time step k. With I, we denote the identity matrix of appropriate size. We use $M \succeq 0$ ($M \preceq 0$), to indicate that M is a positive semi-definite (negative semi definite) and $M \succ 0$ ($M \prec 0$), to indicate that M is a positive definite (negative definite). For matrices that can be inferred from symmetry, we use the symbol $\star$.

Model Structure and Robust Performance

The input-output behavior of a dynamical system can be described in various ways and is often useful for characterizing the performance of a system (a more precise definition can be found in Veenman et al. [2016, Def. 4]. Our main result

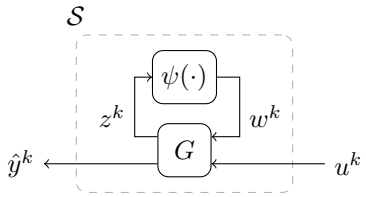


Figure 1: Interconnected system $\mathcal{S}$, consists of a linear system G and nonlinear disturbance $\psi(\cdot)$.

that guarantees a finite (incremental) stability gain for RNN is one specific performance measure of a more general framework used in robust control. We refer to Scherer [2006] for more details and the discussion of other performance measures. For system identification tasks, we focus on the worst possible amplification and the sensitivity, which refers to the finite stability gain and the finite incremental stability gain respectively, both performance measures will be defined in this subsection.

First, we introduce the model structure. We consider the system $\mathcal{S}$ as a linear system G in feedback interconnection with a nonlinearity ψ. The interconnection is shown in Figure 1 and formalized as:

$$G\colon \begin{cases} x^{k+1} = Ax^k + B_1 u^k + B_2 w^k \\ \hat{y}^k = C_1 x^k + D_{11} u^k + D_{12} w^k \ , \\ z^k = C_2 x^k + D_{21} u^k \end{cases} \qquad w^k = \psi(z^k). \tag{1}$$

The system $\mathcal{S}$ maps an input sequence $u \in \ell_{2e}^{n_u}$ to an output sequence $\hat{y} = S(u) \in \ell_{2e}^{n_y}$.

Next, we review the necessary performance conditions that our models should satisfy.

Definition 1 (Finite gain stability) *[cf. Sastry, 2013, Def. 4.14] The system $\mathcal{S}$ is finite gain ℓ_2 stable if there exists $\gamma, \gamma_0 > 0$ such that for a given $u \in \ell_{2e}^{n_u}$ the output $\hat{y} = \mathcal{S}(u) \in \ell_{2e}^{n_y}$ satisfies*

$$\sum_{k=0}^{T} \|\hat{y}^k\|^2 \leq \gamma^2 \sum_{k=0}^{T} \|u^k\|^2 + \gamma_0 \qquad \textit{for all } T > 0. \tag{2}$$

Definition 2 (Incremental finite gain stability) *[cf. Sastry, 2013, Def. 4.16] The system $\mathcal{S}$ is said to be incrementally finite gain stable if there exists $\gamma_{\text{inc}} > 0$ such that for all $u_a, u_b \in \ell_{2e}^{n_u}$*

$$\sum_{k=0}^{T} \|\Delta\hat{y}^k\|^2 \leq \gamma_{\text{inc}}^2 \sum_{k=0}^{T} \|\Delta u^k\|^2 \qquad \textit{for all } T > 0, \tag{3}$$

holds, where $\Delta u = u_a - u_b$ and $\Delta\hat{y} = \mathcal{S}(u_a) - \mathcal{S}(u_b)$.

Remark 3 *Incremental finite gain stability implies finite gain stability. This follows directly from Definition 2 if one input sequence (e.g. u_b) is set to zero.*

To ensure robust performance, we use the following result from Scherer [2006].

Theorem 4 (Robust Quadratic Performance) *The interconnected system $\mathcal{S}$ is well-posed and satisfies robust quadratic performance if there exists $X \succ 0$ and*

$$P = \begin{pmatrix} Q & S \\ S^{\mathrm{T}} & R \end{pmatrix} \qquad \textit{with} \qquad \begin{pmatrix} \psi(z^k) \\ z^k \end{pmatrix}^{\mathrm{T}} P \begin{pmatrix} \psi(z^k) \\ z^k \end{pmatrix} \geq 0 \qquad \textit{for all } z^k \in \ell_{2e}^{n_z} \tag{4}$$

and performance matrices Q_p, R_p, S_p such that

$$\begin{aligned} (\star)^{\mathrm{T}} \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ A & B_1 & B_2 \end{pmatrix} + (\star)^{\mathrm{T}} \begin{pmatrix} Q_p & S_p \\ S_p^{\mathrm{T}} & R_p \end{pmatrix} \begin{pmatrix} 0 & I & 0 \\ C_1 & D_{11} & D_{12} \end{pmatrix} + \\ (\star)^{\mathrm{T}} P \begin{pmatrix} 0 & 0 & I \\ C_2 & D_{12} & 0 \end{pmatrix} \prec 0. \end{aligned} \tag{5}$$

Proof We sketch the key ideas of the proof and refer to Scherer [2006] or more recently Pauli et al. [2022, Thm. 1], for more details. Well-posedness is given for $R_p \succeq 0$ and condition (4). It can be shown that (2) holds by left multiplying (5) with $\begin{bmatrix}(x^k)^{\mathrm{T}} & (u^k)^{\mathrm{T}} & (w^k)^{\mathrm{T}}\end{bmatrix}$ and right multiplying its transpose. The key point is to cancel the term (4) in the resulting inequality. ■

4 Methodology

4.1 Robust System Identification with Linear Matrix Inequality Constraints

In this section, we develop parametric constraints for G such that the stability properties from the previous section are satisfied. We further explain how these constraints are satisfied during training and how the initial state x^0 and the initial parameter set θ^0 is chosen.

Remark 5 *We follow the same ideas as presented in Revay et al. [2021a] and extend the results to guarantee finite gain stability.*

Our goal is to learn the dynamics of a nonlinear system purely from data, while satisfying stability properties according to Definition 1 and 2. This is summarized in the following problem statement.

Problem 6 *Given a dataset $\mathcal{D} = \{(u, y)_i\}$ for $i = 1, \dots, K$ where $u \in \ell_{2e}^{n_u}$ and $y \in \ell_{2e}^{n_y}$ are measurements taken from an unknown dynamical system, find the parameters*

$$\theta = \{A,\ B_1,\ B_2,\ C_1,\ D_{11},\ D_{12},\ C_2,\ D_{21}\}$$

of G that minimize the loss $\sum_{k=0}^{T} \|\hat{y}^k - y^k\|^2$ and satisfy (2) and (3) for some γ^2 and γ_{inc}^2.

To apply Theorem 4, we bound the nonlinearity by linear functions. Nonlinear activation functions commonly used in deep learning like $\mathrm{ReLU}(\cdot)$ or $\tanh(\cdot)$ are sector bounded [e.g. used in Pauli et al., 2022]

$$(\alpha a - \varphi(a))(\varphi(a) - \beta a) \geq 0 \qquad \forall a \in \mathbb{R} \tag{6}$$

and slope restricted [e.g. used in Fazlyab et al., 2019]

$$\begin{aligned}(\mu\Delta_{a,b} - \Delta_{a,b}\varphi)(\Delta_{a,b}\varphi - \eta\Delta_{a,b}) \geq 0,\\ \Delta_{a,b} := a - b,\ \Delta_{a,b}\varphi := \varphi(a) - \varphi(b),\end{aligned} \qquad \forall a, b \in \mathbb{R},\ a \neq b. \tag{7}$$

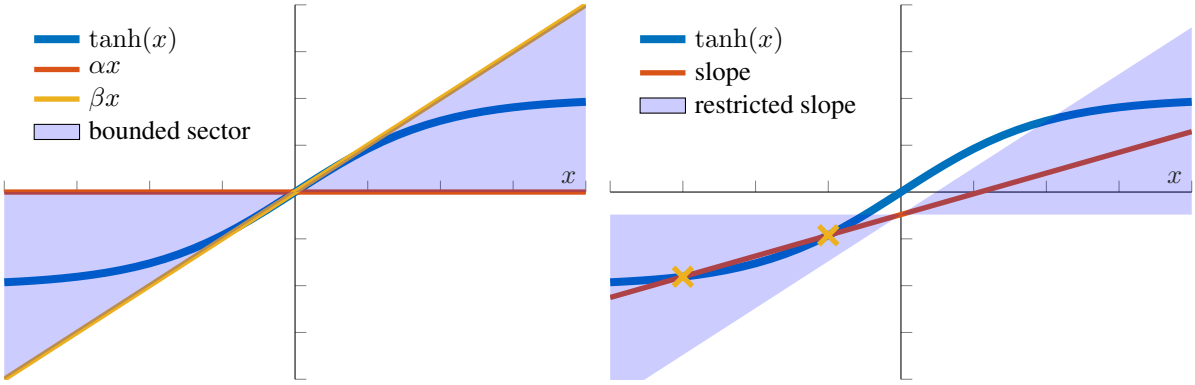
For instance, $\mathrm{ReLU}(\cdot)$ or $\tanh(\cdot)$ satisfies (6) with $\alpha = 0$, $\beta = 1$ and (7) with $\mu = 0$, $\eta = 1$. In this work, we focus on $\varphi = \tanh$ as it is often used for RNNs [Goodfellow et al., 2016]. The sector and slope condition is visualized in Figure 2. Since the activation function $\varphi\colon \mathbb{R} \to \mathbb{R}$ is applied to every neuron in the hidden layer, we define $\psi(z^k) := \left[\varphi(z_1^k), \dots, \varphi(z_{n_z}^k)\right]^{\mathrm{T}}$, which is the disturbance block in Figure 1. Next, we introduce static diagonal multipliers $T = \mathrm{diag}(\lambda_1, \dots, \lambda_{n_z})$, $\lambda_i > 0$, and reformulate condition (6) as a quadratic constraint and condition (7) as an incremental quadratic constraint including multipliers as follows:

$$(\star)^{\mathrm{T}} \underbrace{\begin{pmatrix} -2T & (\alpha+\beta)T \\ (\alpha+\beta)T & -2\alpha\beta T \end{pmatrix}}_{=:P_{\mathrm{sect}}} \begin{pmatrix} \psi(z^k) \\ z^k \end{pmatrix} \geq 0, (\star)^{\mathrm{T}} \underbrace{\begin{pmatrix} -2T & (\mu+\eta)T \\ (\mu+\eta)T & -2\mu\eta T \end{pmatrix}}_{=:P_{\mathrm{slope}}} \begin{pmatrix} \Delta\psi(z^k) \\ \Delta z^k \end{pmatrix} \geq 0. \tag{8}$$

Theorem 4 is classically used to verify robust quadratic performance of a known linear system G. However, for system identification problems, the parameters θ are unknown. Condition (5) is not convex with respect to the parameter set θ and therefore cannot be solved by semi-definite programming. We refer to Boyd and Vandenberghe [2004] for the topic of convex optimization and semi-definite programming. To convexify condition (5), we introduce new parameters $\tilde{\theta} := \left\{\tilde{A}, \tilde{B}_1, \tilde{B}_2, C_1, D_{11}, D_{12}, \tilde{C}_2, \tilde{D}_{21}\right\}$, with $\tilde{A} := XA$, $\tilde{B}_1 := XB_1$, $\tilde{B}_2 := XB_2$, $\tilde{C}_2 := TC_2$, $\tilde{D}_{21} := TD_{21}$, which yields the linear matrix inequality

$$M := \begin{pmatrix} -X & 0 & \tilde{C}_2^{\mathrm{T}} & \tilde{A}^{\mathrm{T}} & C_1^{\mathrm{T}} \\ 0 & -\boldsymbol{\gamma}^2 I & \tilde{D}_{21}^{\mathrm{T}} & \tilde{B}_1^{\mathrm{T}} & D_{11}^{\mathrm{T}} \\ \tilde{C}_2 & \tilde{D}_{21} & -2T & \tilde{B}_2^{\mathrm{T}} & D_{12}^{\mathrm{T}} \\ \tilde{A} & \tilde{B}_1 & \tilde{B}_2 & -X & 0 \\ C_1 & D_{11} & D_{12} & 0 & -I \end{pmatrix} \prec 0, \tag{9}$$

where $\boldsymbol{\gamma} \in \{\gamma,\ \gamma_{\mathrm{inc}}\}$. With the constraints (8) and (9), we can now state the main stability results.



(*a*) tanh is sector bounded by αx and βx.

(*b*) Any slope of tanh is bounded. In this example we picked $a = -1.5$ and $b = -0.5$ and show the bounds for the resulting slope.

Figure 2: Activation function tanh is sector bounded (6) and slope restricted (7)

Corollary 7 *If the parameter set $\tilde{\theta}$ satisfies constraint* (9)*, the system $\mathcal{S}$ defined in* (1) *has a finite ℓ_2-gain γ^2 and satisfies stability condition* (2)*.*

Proof We follow standard (conventional) arguments of the s-procedure [Scherer and Weiland, 2000] to show that (2) holds for the interconnection $\mathcal{S}$ based on the pioneering work by Willems [1972a,b] that was recently also used in [Pauli et al., 2022].

Firstly, we use the Schur complement to obtain:

$$\begin{pmatrix} \tilde{A}^{\mathrm{T}} & C_1^{\mathrm{T}} \\ \tilde{B}_1^{\mathrm{T}} & D_{11}^{\mathrm{T}} \\ \tilde{B}_2^{\mathrm{T}} & D_{12}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} X^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \tilde{A} & \tilde{B}_1 & \tilde{B}_2 \\ C_1 & D_{11} & D_{12} \end{pmatrix} + \begin{pmatrix} -X & 0 & \tilde{C}_2^{\mathrm{T}} \\ 0 & -\gamma^2 I & \tilde{D}_{21}^{\mathrm{T}} \\ \tilde{C}_2 & \tilde{D}_{21} & -2I \end{pmatrix} \prec 0$$

Since (9) is satisfied X, T are invertible and $X = X^{\mathrm{T}} X^{-1} X$ holds, we apply the Schur complement once more utilise use the parameters from (1):

$$\begin{aligned} &\begin{pmatrix} I & 0 & 0 \\ A & B_1 & B_2 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ A & B_1 & B_2 \end{pmatrix} + \\ &\begin{pmatrix} 0 & I & 0 \\ C_1 & D_{11} & D_{12} \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -\gamma I & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} 0 & I & 0 \\ C_1 & D_{11} & D_{12} \end{pmatrix} + \\ &\begin{pmatrix} 0 & 0 & I \\ C_2 & D_{12} & 0 \end{pmatrix}^{\mathrm{T}} P \begin{pmatrix} 0 & 0 & I \\ C_2 & D_{12} & 0 \end{pmatrix} \prec 0. \end{aligned} \tag{10}$$

The variable z^k does not depend on w^k, therefore well-posedness of the interconnection directly follows from (8). We left multiply (5) by $\begin{bmatrix} (x^k)^{\mathrm{T}} & (u^k)^{\mathrm{T}} & (w^k)^{\mathrm{T}} \end{bmatrix}$ and right multiply its transpose to arrive at

$$\begin{pmatrix} x^{k+1} \\ x^k \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -X & 0 \\ 0 & X \end{pmatrix} \begin{pmatrix} x^{k+1} \\ x^k \end{pmatrix} + \begin{pmatrix} u^k \\ y^k \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} -\gamma^2 I & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} u^k \\ y^k \end{pmatrix} + \begin{pmatrix} w^k \\ z^k \end{pmatrix}^{\mathrm{T}} P \begin{pmatrix} w^k \\ z^k \end{pmatrix} < 0.$$

With $V(x) := x^{\mathrm{T}} X x$ and (8) we get

$$-V(x^{k+1}) + V(x^k) - \gamma^2 (u^k)^{\mathrm{T}} (u^k) + (y^k)^{\mathrm{T}} (y^k) < 0.$$

Now we take the sum over $k = 0, 1, \ldots, T$ which yields

$$V(x^0) - V(x^{\mathrm{T}}) + \sum_{k=0}^{T} (y^k)^{\mathrm{T}} (y^k) < \gamma^2 \sum_{k=0}^{\mathrm{T}} (u^k)^{\mathrm{T}} (u^k). \tag{11}$$

The matrix X is positive definite according to (9) and therefore, stability condition (2) follows from (11) with $\gamma_0 = (x^0)^{\mathrm{T}} X (x^0)$, which concludes the proof. ■

Corollary 8 *If the parameter set $\tilde{\theta}$ satisfies the constraint* (9)*, the system $\mathcal{S}$ defined in* (1) *has a finite incremental ℓ_2-gain γ^2_{inc} and satisfies stability condition* (3)*.*

Proof The proof follows the same arguments as Corollary 7, with $\mathbf{\Delta}(z^k) = \Delta\psi(z^k)$, $P = P_{\text{slope}}$. Note that we can assume to have the same initial condition for two different sequences which leads to $\Delta x^0 = (x^0)_a - (x^0)_b = 0$. ■

4.2 Model Optimization

In this section, we explain how we initialize the hidden state x^0, the initial parameter set $\tilde{\theta}^0$ and how we ensure that the parameters satisfy the constraint (9) during training.

4.2.1 Initial hidden state x^0

To initialize the hidden state of the linear system G, we use a long short-term memory (LSTM) network as suggested in Mohajerin and Waslander [2019]. The initializer LSTM is trained separately from the prediction network on the sequence $\xi^k = \begin{bmatrix} u^k & y^{k-1} \end{bmatrix}^{\mathrm{T}}$, for $k = -T_{\text{init}}, -T_{\text{init}}+1, \ldots, 0$ where we assume to know the previous system state y^{k-1}. The initializer LSTM predicts the next system state $\hat{y}^k_{\text{init}}$, the predictor model then takes the final hidden state of the initializer LSTM as the initial state x^0.

4.2.2 Initial parameter $\tilde{\theta}^0$

Before training the predictor network $\mathcal{S}$, we find an initial parameter set $\tilde{\theta}^0$ that satisfies condition (9). This is done by solving a semi-definite program, where the solution is a feasible parameter set $\tilde{\theta}^0$, as also suggested in Revay et al. [2021a], Pauli et al. [2022].

Remark 9 *In this work, we use a trivial objective for the semi-definite program to find a feasible solution. For training, we take the feasible solution as initial parameter set and minimize the prediction error (cf. Section 4.2.3).*

4.2.3 Loss function for predictor $\mathcal{S}$

Assuming that a feasible initial parameter set $\tilde{\theta}^0$ is found, we optimize the parameters $\tilde{\theta}$ to minimize the mean squared error loss. A barrier function is used as regularization to keep the parameters away from the constraint boundary. Thus our loss function is defined as

$$\mathcal{L}(y, \hat{y}) = \frac{1}{T} \sum_{k=0}^{T} \|\hat{y}^k - y^k\|^2 - \nu \log\det(-M), \tag{12}$$

where M is defined as in (9) and we use the logarithmic barrier function, which is often used in interior-point methods [Boyd and Vandenberghe, 2004]. During optimization, it can happen that the new parameters do not satisfy condition (9). Therefore as suggested in Revay et al. [2021a], we apply a backtracking line search algorithm for 100 steps. If no feasible parameter set can be found, training stops. Problem 6 can now be stated as the following optimization problem

$$\min_{\tilde{\theta}, X, T} \mathcal{L}(y, \hat{y}) \qquad \text{such that} \qquad M \prec 0, \tag{13}$$

where $\hat{y} = \mathcal{S}(u)$ and $(u, y) \in \mathcal{D}$. Note that even though the constraints in (13) are convex, the loss function is non-convex with respect to the parameters $\tilde{\theta}$, this directly follow from the recurrent structure of (1). We assume that the step size of the optimization is small enough such that a violation of condition (9) is noticed if one eigenvalue of M flips its sign. The decay parameter ν is decreased by a factor of 10 after 100 training epochs and initialized with 0.001.

5 Numerical Experiment

5.1 Dataset

To generate data for system identification, we use a model of a patrol ship with four-degrees-of-freedom [Perez et al., 2006]. The model is extended to include effects caused by wind [Isherwood, 1972] and waves [Hasselmann et al., 1973]. To navigate the ship, two fixed-pitch rudder propellers are modeled as actuators. Input sequences are generated

via an open-loop controller that performs different maneuvers like turns or circles. In addition to the input sequences that are sent to the actuator, we also assume to have access to simulated wind measurements that include the strength and the angle of attack. The unknown system to be identified has $n_u = 6$ inputs $u^k = \left[\alpha^k_{\sin}\ \alpha^k_{\cos}\ V^k_w\ n^k\ \delta^k_{\text{left}}\ \delta^k_{\text{right}}\right]^{\mathrm{T}}$, namely *angle of attack* ($\alpha^k_{\sin}$ and $\alpha^k_{\cos}$), *wind speed* (V^k_w), *propeller speed* (n^k), and *propeller angle* (left δ^k_{left} and right δ^k_{right}). The predicted output refers to $n_y = 5$ system states $\hat{y}^k = \left[s^k\ v^k\ p^k\ r^k\ \phi^k\right]^{\mathrm{T}}$, the velocity in two dimensions, *surge* (s^k) and *sway* (v^k), the angular velocity of *roll* (p^k) and *yaw* (r^k), and the *roll angle* (ϕ^k).

The dataset $\mathcal{D}$ consists of 96 hours of routine samples and 29 hours of OOD samples [Baier and Staab, 2022]. The measurements are sampled every second. The routine samples are split into 60% *training* $\mathcal{D}_{\text{train}}$, 10% *validation* $\mathcal{D}_{\text{val}}$ and 30% *testing* $\mathcal{D}_{\text{test}}$. The OOD samples $\mathcal{D}_{\text{OOD}}$ are only used for evaluation, and consists of a larger range of the propeller speed and more frequent changes in the rudder angle.

5.2 Models

As baselines for the predictor model, we use purely learning-based approaches that do not have parametric constraints. For learning the initial hidden state, we use the same LSTM architecture for all models (cf. Section 4.2.1). We compare our model $\mathcal{S}$ against: (i) `ltiRNN`, a RNN in LTI structure (1), where we use parameters $\tilde{A}, \tilde{B}_1, \tilde{B}_2, C_1, D_{11}, D_{12}, \tilde{C}_2, \tilde{D}_{21}, X, T$ but no constraints, (ii) `RNN`, a RNN with bias terms [Goodfellow et al., 2016], and (iii) `LSTM`, a LSTM that was proposed by Mohajerin and Waslander [2019] for multi-step prediction of quadcopter motion. We will call the constrained RNN `cRNN` and add the finite stability gain γ^2 in the name. We fix the stability gain γ^2 before training and treat it as a hyperparameter of the model.

5.2.1 Models training

The initializer (cf. Section 4.2.1) is trained on sequences of length $T_{\text{init}} = 50$ and 400 epochs, and we use the same hyperparameters as for the predictor (details on how the hyperparameters are found are provided in Section 5.3). We train the predictor on sequence length $T_{\text{pred}} = 50$ and 2000 epochs. The learning rate is set to 0.0025 and we use the *Adam*-optimizer [Kingma and Ba, 2015]. We apply stochastic gradient descent on batches of size 128 and evaluate the loss function (12) on the entire batch of sequences. Training and testing is performed on a Nvidia A100 GPU [1].

5.3 Evaluation

First, we select the best models by hyperparameter optimization on the validation samples $\mathcal{D}_{\text{val}}$. We then evaluate the models on prediction accuracy of the test samples $\mathcal{D}_{\text{test}}$, empirically evaluate their robustness and further test their generalization ability with the OOD samples $\mathcal{D}_{\text{OOD}}$.

For hyperparameter optimization, we use the root mean squared error (RMSE) that is defined as

$$\operatorname{RMSE}(\mathcal{D}_{\text{val}}, m) := \sqrt{\frac{1}{N_{\text{val}}} \sum_{i=1}^{K_{\text{val}}} \sum_{k=0}^{T_{\text{pred}}} ((y^k_m)_i - (\hat{y}^k_m)_i)^2} \qquad \text{for } m = 1, \ldots, n_y. \tag{14}$$

We perform grid search on the size of the hidden states and the number of recurrent layers for the `LSTM` and the `RNN` model. The RMSE is calculated for each state separately, and we select the model with the lowest mean RMSE over all states given by

$$\overline{\operatorname{RMSE}}(\mathcal{D}_{\text{val}}) := \frac{1}{n_y} \sum_{m=1}^{n_y} \operatorname{RMSE}(\mathcal{D}_{\text{val}}, m). \tag{15}$$

The hyperparameters for each model type are shown in Section A. The best performing models are evaluated for their RMSE and $\overline{\text{RMSE}}$ on the test samples $\mathcal{D}_{\text{test}}$.

For robustness evaluation, we first define a perturbed input sequence as $u_{\text{pert}} := u + \vartheta$ and the corresponding prediction as $\hat{y}_{\text{pert}} = \mathcal{S}(u_{\text{pert}})$. The worst finite stability gain is then defined as

$$\gamma^2_* := \max_{u \in \mathcal{D}_{\text{val}}} \left(\sup_{\vartheta^k \in \mathbb{R}^{n_u}} \frac{\sum_{k=0}^{T} \|\hat{y}^k_{\text{pert}}\|^2}{\sum_{k=0}^{T} \|u^k_{\text{pert}}\|^2} \right). \tag{16}$$

[1] Our code is available on *GitHub* https://github.com/AlexandraBaier/deepsysid

Table 1: Evaluation on test dataset $\mathcal{D}_{\text{test}}$. The best and second-best scores are marked in **bold** and ***bold+italic*** respectively. The last column shows the mean RMSE on the OOD samples.

model	s [m/s]	v [m/s]	p [rad/s]	r [rad/s]	ϕ [rad]	$\overline{\text{RMSE}}(\mathcal{D}_{\text{test}})$	$\overline{\text{RMSE}}(\mathcal{D}_{\text{OOD}})$
cRNN $\gamma^2 = 5$	0.293	0.172	**0.00489**	0.00540	0.01820	0.09878	***0.1984***
cRNN $\gamma^2 = 10$	0.287	0.150	**0.00489**	0.00519	0.01780	0.09306	0.1991
cRNN $\gamma^2 = 20$	0.292	0.138	**0.00489**	0.00525	0.01780	0.09144	0.1999
cRNN $\gamma^2 = 40$	0.294	0.210	***0.00491***	0.00568	0.01870	0.10660	0.2114
ltiRNN	***0.163***	0.059	0.00511	0.00230	0.00753	0.04724	**0.1428**
RNN	0.173	***0.055***	0.00531	***0.00205***	**0.00715**	***0.04845***	0.2057
LSTM	**0.096**	**0.052**	0.00590	**0.00203**	***0.00736***	**0.03259**	0.1992

Table 2: Evaluation on out-of-distribution data $\mathcal{D}_{\text{OOD}}$. The best and second-best scores are marked in **bold** and ***bold+italic*** respectively.

model	s [m/s]	v [m/s]	p [rad/s]	r [rad/s]	ϕ [rad]	$\overline{\text{RMSE}}(\mathcal{D}_{\text{OOD}})$
cRNN $\gamma^2 = 5$	***0.520***	0.414	0.00702	0.0137	0.0375	***0.1984***
cRNN $\gamma^2 = 10$	0.535	0.403	0.00692	0.0139	0.0360	0.1991
cRNN $\gamma^2 = 20$	0.525	0.416	0.00697	0.0145	0.0370	0.1999
cRNN $\gamma^2 = 40$	0.539	0.462	0.00729	0.0129	0.0357	0.2114
ltiRNN	**0.370**	0.311	**0.00599**	***0.00713***	0.0195	**0.1428**
RNN	0.737	**0.261**	0.00632	**0.00573**	**0.0184**	0.2057
LSTM	0.673	***0.293***	***0.00616***	0.00581	***0.0188***	0.1992

We maximize the stability gain γ^2 for each input sequence from the validation set $\mathcal{D}_{\text{val}}$. As optimization variable, we use the perturbation sequence ϑ. This empirically evaluates the stability gain of the trained models. Recall that the ℓ_2-norm is a measure of the energy content of the sequence and therefore the stability gain γ^2 (cf. Definition 1) refers to the worst possible amplification of the model.

For the evaluation of the incremental stability gain (see Definition 2) we define the worst incremental stability gain as

$$\gamma^2_{\text{inc}*} := \max_{u \in \mathcal{D}_{\text{val}}} \left(\sup_{\vartheta^k \in \mathbb{R}^{n_u}} \frac{\sum_{k=0}^{T} \|\hat{y}^k_{\text{pert}} - \hat{y}^k\|^2}{\sum_{k=0}^{\mathrm{T}} \|u^k_{\text{pert}} - u^k\|^2} \right), \tag{17}$$

where $\hat{y} = \mathcal{S}(u)$ is the prediction of the unperturbed input sequence u. For the nonlinear operator $\mathcal{S}$, the incremental stability gain refers to the Lipschitz constant of $\mathcal{S}$ and is considered as a robustness measure of the neural network [Fazlyab et al., 2019]. A model with a large incremental γ^2_{inc}-gain is sensitive to small input changes. The learning rate for the optimization is set to 0.001 and we run 2000 optimization steps for problem (16) and 1000 steps for problem (17).

Lastly, we test the models on OOD samples $\mathcal{D}_{\text{OOD}}$ to evaluate their generalization capacity, note that for evaluation the prediction horizon is set to 900 seconds.

5.4 Experiment Result

Table 1 compares $\text{RMSE}(\mathcal{D}_{\text{test}}, m)$ for $m = 1, \ldots, n_y$ of the predicted states for the selected models. The smallest values refer to the highest score according to (14). In the last column of Table 1 we show the mean squared error loss for the OOD samples $\mathcal{D}_{\text{OOD}}$. Note that all models have a worse mean RSMD for the OOD samples in comparison with the test set $\mathcal{D}_{\text{test}}$, which is expected since the input range of the propeller speed is larger compared to the test set. However, the accuracy drop for the constrained RNN is lower than for the learning-based models without constraints. Despite the higher accuracy of the baseline models (LSTM, RNN, ltiRNN) compared to the constrained RNN (cRNN) no upper bounds on the (incremental) stability gain are available. We show γ^2_* and $\gamma^2_{\text{inc}*}$ and relate it to the prediction accuracy on the test samples in Figure 3. The dashed vertical lines refer to the guaranteed upper (incremental) stability bound for the constrained RNN. Table 2 show the $\text{RMSE}(\mathcal{D}_{\text{OOD}}, m)$ for all states as well as the $\overline{\text{RMSE}}(\mathcal{D}_{\text{OOD}})$

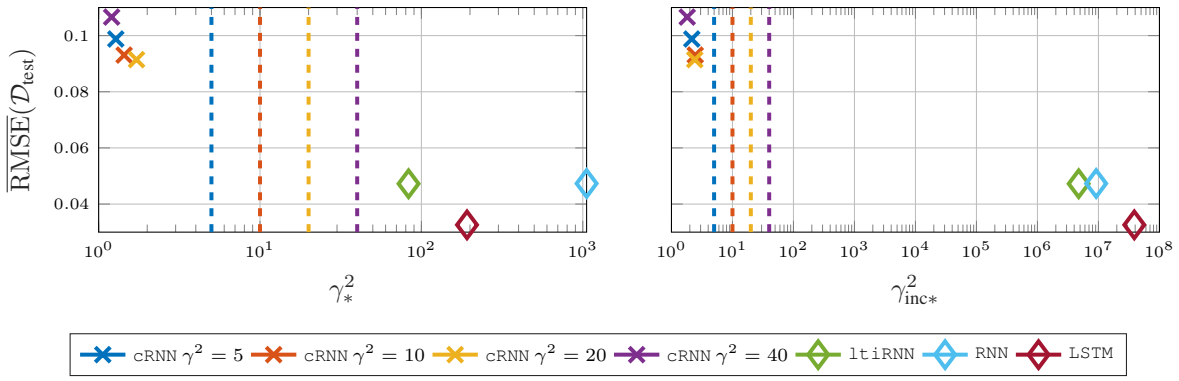
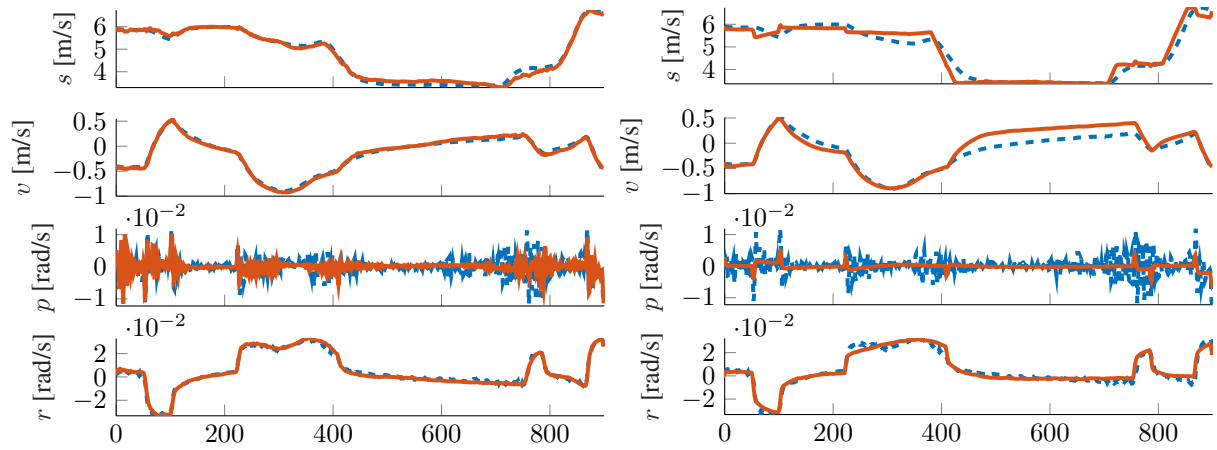


Figure 3: The figure highlights the trade-off between robustness and accuracy when identifying the ship motion for baseline models (LSTM, RNN, ltiRNN) and constrained RNN (cRNN) with different γ^2 values. On the y-axis the $\overline{\text{RMSE}}$ (15) of the test set is shown, on the x-axis, the maximum stability gain γ_*^2 (left) and the maximum incremental stability gain $\gamma_{\text{inc}*}^2$ (right) values are shown.



(a) LSTM (b) constrRNN $\gamma^2 = 20$

Figure 4: Comparison of different identification methods, the dashed red line refers to the true measurement of the velocities and the solid blue line to the prediction of the model.

Figure 4 shows the prediction of the velocities of the best performing constrained RNN (constrRNN $\gamma^2 = 10$) (b)subfigure and the best performing purely learned model LSTM (a)subfigure. For an intuition on the ship movement Figure 5 shows the position trajectory for the different models.

5.5 Discussion

The trade-off between robustness and accuracy of the predictor model is visualized in Figure 3. Here we can see that for the learned models without constraints, we identify worse (incremental) stability values compared to the guaranteed bounds of the cRNN models. As expected, we thus observe a better generalization of the cRNN for the OOD dataset in comparison to the unconstrained approaches, i.e., the ratio $\overline{\text{RMSE}}(\mathcal{D}_{\text{OOD}})/\overline{\text{RMSE}}(\mathcal{D}_{\text{test}})$ is smaller for the cRNN models.

Another observation from Figure 3 is that the upper bounds are not tight. This means that we were not able to find input sequences that lead to an (incremental) stability gain that was close to the upper bound. Because we use linear functions to bound the non-linearity (6), (7) and only consider static multipliers T in (8), our constraints are quite conservative.

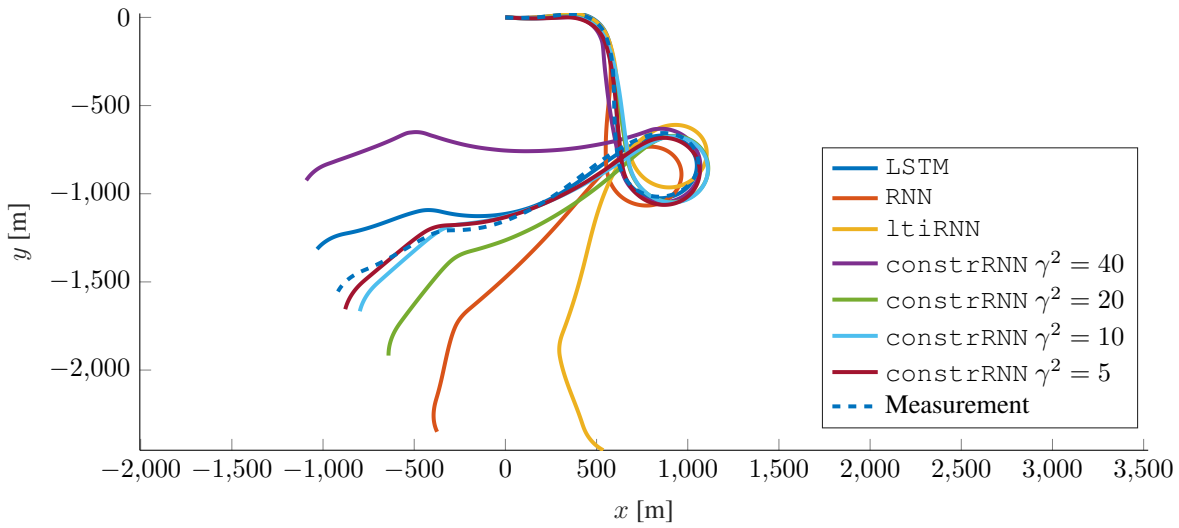


Figure 5: Trajectory of ship in 2D plane.

For the constrained RNN networks, we expect the accuracy to be higher for larger (incremental) stability gains. This hypothesis results from the definition of the allowed parameter set (9) where we have a larger set for higher γ^2 values. In Table 1, one observes that this hypothesis does not hold in our case. A possible explanation for this behavior is that the initial parameter set θ^0 comes from an optimization problem with trivial objective (cf. Section 4.2.2). Since we perform a backtracking line search if parameters are outside the constraint set, training might stop at a local minimum that is close to the constraint boundary. Other initialization methods could improve the prediction results.

We speculate that the roll motion is mostly driven by waves that are induced by wind. As measurement input, we are only able to measure the wind force and leave it to the network to learn an internal wind-to-wave model. The unconstrained models allow for system optimization exploring larger areas of the parameter space compared to the limited region of the constrained RNN and therefore might be able to learn such complex dynamics.

6 Conclusion

In this work, we used constrained RNNs to identify the dynamics of a ship that moves in open water. With a general framework from robust control, we were able to guarantee both a finite stability gain and a finite incremental stability gain, which serves as additional safety measure for the underlying RNN. We compare the constrained RNN against learning-based approaches without constraints. The evaluation shows that there exists a trade-off between robustness and prediction accuracy. We empirically evaluated the stability of the models by finding input sequences that lead to large (incremental) stability gains. These sequences refer to potentially dangerous predictions of the model like swing-up of the roll angle. We further showed that robust RNNs slightly generalize better to unseen input sequences (OOD dataset) compared to LSTMs and RNNs.

Acknowledgments

We thank Alexandra Baier for helpful discussions and her python package *deepsysid* that was used for training, testing, and evaluating our models. This work is funded by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2075 – 390740016. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech).

References

Alexandra Baier and Steffen Staab. A Simulated 4-DOF Ship Motion Dataset for System Identification under Environmental Disturbances, 2022. URL https://doi.org/10.18419/darus-2905.

Alexandra Baier, Zeyd Boukhers, and Steffen Staab. Hybrid Physics and Deep Learning Model for Interpretable Vehicle State Prediction. *arXiv preprint arXiv:2103.06727*, 2021.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. *33rd Conference on Neural Information Processing Systems*, 32, 2019.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Fangda Gu, He Yin, Laurent El Ghaoui, Murat Arcak, Peter Seiler, and Ming Jin. Recurrent Neural Network Controllers Synthesis with Stability Guarantees for Partially Observed Systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5385–5394, 2022.

K. Hasselmann, T. P. Barnett, E. Bouws, H. Carlson, D. E. Cartwright, K. Enke, J. A. Ewing, H. Gienapp, D. E. Hasselmann, P. Kruseman, A Meerburg, A. Müller, D. J. Olbers, K. Richter, W. Sell, and H. Walden. Measurements of Wind-Wave Growth and Swell Decay during the Joint North Sea Wave Project (JONSWAP). *Ergänzungsheft zur Deutschen Hydrographischen Zeitschrift, Reihe A*, 1973.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

R.M. Isherwood. Wind Resistance of Merchant Ships. *The Royal Institution of Naval Architects*, 115:327–338, 1972.

Neelay Junnarkar, He Yin, Fangda Gu, Murat Arcak, and Peter Seiler. Synthesis of Stabilizing Recurrent Equilibrium Network Controllers. *arXiv preprint arXiv:2204.00122*, 2022.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations (Poster)*, 2015.

Nima Mohajerin and Steven L. Waslander. Multistep Prediction of Dynamic Systems With Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3370–3383, 2019.

Patricia Pauli, Julian Berberich, and Frank Allgöwer. Robustness analysis and training of recurrent neural networks using dissipativity theory. *at-Automatisierungstechnik*, 70(8):730–739, 2022.

Tristan Perez, Andrew Ross, and Thor Fossen. A 4-DOF simulink model of a coastal patrol vessel for manoeuvring in waves. In *Proceedings of the 7th IFAC Conference on Manoeuvring and Control of Marine Craft*, pages 1–6. International Federation for Automatic Control, 2006.

Gianluigi Pillonetto, Tianshi Chen, Alessandro Chiuso, Giuseppe De Nicolao, and Lennart Ljung. Classical System Identification. In *Regularized System Identification*, pages 17–31. Springer, 2022.

Max Revay, Ruigang Wang, and Ian R. Manchester. A Convex Parameterization of Robust Recurrent Neural Networks. *IEEE Control Systems Letters*, 5(4):1363–1368, 2021a.

Max Revay, Ruigang Wang, and Ian R. Manchester. Recurrent equilibrium networks: Unconstrained learning of stable and robust dynamical models. In *2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, December 14-17, 2021*, pages 2282–2287. IEEE, 2021b. doi: 10.1109/CDC45484.2021.9683054. URL `https://doi.org/10.1109/CDC45484.2021.9683054`.

Shankar Sastry. *Nonlinear systems: analysis, stability, and control*, volume 10. Springer Science & Business Media, 2013.

Carsten Scherer and Siep Weiland. Linear matrix inequalities in control. *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands*, 3(2), 2000.

Carsten W. Scherer. LMI Relaxations in Robust Control. *European Journal of Control*, 12(1):3–29, 2006.

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

Joost Veenman, Carsten W. Scherer, and Hakan Köroglu. Robust stability and performance analysis based on integral quadratic constraints. *European Journal of Control*, 31:1–32, 2016. doi: 10.1016/j.ejcon.2016.04.004. URL `https://doi.org/10.1016/j.ejcon.2016.04.004`.

Jan C. Willems. Dissipative Dynamical Systems Part I: General Theory. *Archive for Rational Mechanics and Analysis*, 45:321–351, 1972a.

Jan C. Willems. Dissipative Dynamical Systems Part II: Linear Systems with Quadratic Supply Rates. *Archive for Rational Mechanics and Analysis*, 45:352–393, 1972b.

A Hyperparameter Optimization

Hyperparmeter optimization for different model types. The size of the hidden state for `LSTM` and `RNN` is denoted by n_h and the size of $z^k = w^k$ for the linear model G (1) is denoted by n_w. Mean RMSE refers to the mean of $\mathrm{RMSE}_{\mathrm{val}}$ (14) for all predicted states of the validation set, the models are listed in ascending order.

Model type	Mean RMSE	γ^2	n_w	n_h	# rec. layers	pred. steps	t_{train} (hh:mm:ss)	barrier	grad. norm
cRNN	**0.09223**	20	64	-	-	427	03:10:36	-0.001	1.8
cRNN	0.09282	20	128	-	-	349	02:48:13	-0.013	1.8
cRNN	0.09351	20	256	-	-	521	04:18:14	-0.00019	1.8
cRNN	0.09359	20	192	-	-	417	03:47:26	-0.0016	1.8
cRNN	0.09363	10	64	-	-	511	03:52:36	-7.3e-05	2.1
cRNN	0.0956	10	128	-	-	336	03:03:06	-0.0095	2.1
cRNN	0.09584	10	192	-	-	503	04:23:39	-0.00011	2
cRNN	0.09734	10	256	-	-	526	04:20:45	-0.00014	2.4
cRNN	0.09822	5	64	-	-	344	01:51:38	-0.0052	2.7
cRNN	0.102	5	256	-	-	522	04:20:24	-0.0001	2.7
cRNN	0.102	5	128	-	-	446	03:03:06	-0.00069	2.7
cRNN	0.1059	5	192	-	-	512	04:22:33	-8.7e-05	2.4
cRNN	0.1061	40	64	-	-	503	04:03:41	-0.00013	2.3
cRNN	0.1075	40	128	-	-	523	04:45:00	-0.00017	2.3
cRNN	0.1078	40	192	-	-	758	04:52:58	-2.1e-06	2.3
cRNN	0.1078	40	256	-	-	525	04:39:11	-0.00025	2.3
LSTM	**0.03181**	-	-	128	2	2000	00:28:16	NaN	NaN
LSTM	0.03605	-	-	192	2	2000	00:27:44	NaN	NaN
LSTM	0.03725	-	-	128	3	2000	00:27:19	NaN	NaN
LSTM	0.0388	-	-	128	4	2000	00:26:31	NaN	NaN
LSTM	0.04076	-	-	192	3	2000	00:30:21	NaN	NaN
LSTM	0.0409	-	-	64	3	2000	00:28:32	NaN	NaN
LSTM	0.04121	-	-	64	2	2000	00:28:15	NaN	NaN
LSTM	0.04369	-	-	256	2	2000	00:28:39	NaN	NaN
LSTM	0.04406	-	-	192	4	2000	00:33:51	NaN	NaN
LSTM	0.04768	-	-	256	3	2000	00:28:19	NaN	NaN
LSTM	0.05025	-	-	64	4	2000	00:26:06	NaN	NaN
LSTM	0.05125	-	-	256	4	2000	00:29:05	NaN	NaN
RNN	**0.04732**	-	-	64	2	2000	00:31:39	NaN	NaN
RNN	0.04831	-	-	64	3	2000	00:26:14	NaN	NaN
RNN	0.04869	-	-	64	4	2000	00:28:50	NaN	NaN
RNN	0.07736	-	-	128	2	2000	00:18:23	NaN	NaN
RNN	0.08317	-	-	192	2	2000	00:40:30	NaN	NaN
RNN	0.1065	-	-	128	3	2000	00:19:40	NaN	NaN
RNN	0.1066	-	-	128	4	2000	00:21:16	NaN	NaN
RNN	0.11	-	-	192	4	2000	00:27:09	NaN	NaN
RNN	0.1316	-	-	192	3	2000	00:39:56	NaN	NaN
RNN	0.133	-	-	256	4	2000	00:37:54	NaN	NaN
RNN	0.1344	-	-	256	2	2000	00:39:59	NaN	NaN
RNN	0.1365	-	-	256	3	2000	00:16:58	NaN	NaN
ltiRNN	**0.04512**	-	192	-	-	2000	01:23:45	NaN	11
ltiRNN	0.04559	-	256	-	-	2000	01:28:58	NaN	12
ltiRNN	0.04932	-	128	-	-	2000	01:24:11	NaN	9.4
ltiRNN	0.04938	-	64	-	-	2000	01:22:20	NaN	11