Improved Initialization of State-Space Artificial Neural Networks

Maarten Schoukens

Abstract—The identification of black-box nonlinear state-space models requires a flexible representation of the state and output equation. Artificial neural networks have proven to provide such a representation. However, as in many identification problems, a nonlinear optimization problem needs to be solved to obtain the model parameters (layer weights and biases). A well-thought initialization of these model parameters can often avoid that the nonlinear optimization algorithm converges to a poorly performing local minimum of the considered cost function. This paper introduces an improved initialization approach for nonlinear state-space models represented as a recurrent artificial neural network and emphasizes the importance of including an explicit linear term in the model structure. Some of the neural network weights are initialized starting from a linear approximation of the nonlinear system, while others are initialized using random values or zeros. The effectiveness of the proposed initialization approach over previously proposed methods is illustrated on two benchmark examples.

I. INTRODUCTION

Due to increasing performance demands, the introduction of more light-weight structures and increasing constraints on energy consumption of systems, nonlinear models and nonlinear control has become increasingly important [1]. Since many nonlinearities, such as hysteresis or friction, are very difficult to model starting from first principles only, improved data-driven modelling approaches are required. A wide range of nonlinear modelling approaches and frameworks is available to the user: NARMAX, block-oriented and nonlinear state-space approaches to name a few [1]–[3].

Due to their flexibility and their ability to model a wide range of nonlinear behavior, the identification of nonlinear state-space models has been of high interest over the last decade. This resulted in a variety of different approaches such as the polynomial nonlinear state-space method [4], [5], fully probabilistic nonlinear state-space identification using particle filter approaches [6], [7], subspace approaches [8], and state-space neural network (SS-NN) methods [9]–[11].

Artificial neural networks are known for their ability to represent multidimensional static nonlinear functions [12]. This makes them a more suitable candidate than multivariate polynomial functions to represent the state and the output equation that are present in a nonlinear state-space model. Furthermore, [9] shows how a nonlinear state-space model equation that are present in a nonlinear state-space model can easily be represented as a recurrent neural network. This makes them a more suitable candidate than multivariate polynomial functions to represent multidimensional static nonlinear functions [12].

One can separate a linear part in eq. (1) without loss of generality:

\[
x(k + 1) = f(x(k), u(k)), \quad y_0(k) = g(x(k), u(k)),
\]

where \( u(k) \in \mathbb{R}^{n_u \times 1} \), is the system input, and \( y_0(k) \in \mathbb{R}^{n_y \times 1} \) is the noiseless system output, \( x(k) \in \mathbb{R}^{n_x \times 1} \) denotes the system states and \( k \) denotes the sample index.

One can separate a linear part in eq. (1) without loss of generality:

\[
x(k + 1) = Az(k) + Bu(k) + \hat{f}(x(k), u(k)), \quad y_0(k) = Cx(k) + Du(k) + \tilde{g}(x(k), u(k)),
\]

with the matrices \( A \in \mathbb{R}^{n_x \times n_x} \), \( B \in \mathbb{R}^{n_x \times n_u} \), \( C \in \mathbb{R}^{n_y \times n_x} \), \( D \in \mathbb{R}^{n_y \times n_u} \). The functions \( \hat{f}(\cdot) \) and \( \tilde{g}(\cdot) \) are again static nonlinear functions. This split between nonlinear and linear dynamics automatically results when a polynomial representation is used for \( f(\cdot) \) and \( g(\cdot) \) [4]. Such a separation of the linear and nonlinear dynamics has also been reported in [11], however, no results on the impact of explicitly including such a linear part have been reported there.

In this work, the static nonlinearities are represented by a feedforward neural network with one hidden layer with

This research was supported by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Fellowship (grant agreement nr. 798627).

M. Schoukens is with the Control Systems group, Department of Electrical Engineering, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands m.schoukens@tue.nl
neural network (gR-SS-NN): done for eq. (2), denoted as a generalized residual state-space as a state-space neural network (SS-NN). The same can be when written down as a recurrent neural network: are depicted in Figure 1. Equation (1) becomes the following network, denoted as a state-space neural network (SS-NN). can be interpreted as a specific form of a recurrent neural network to represent the state and output equations all with appropriate dimensions. A similar representation is used for \( g(\cdot) \), \( f(\cdot) \) and \( \hat{g}(\cdot) \).

A zero-mean, possibly colored, additive noise source \( v(k) \) is assumed to be present at the output only \( y(k) = y_0(k) + v(k) \). The noise source is assumed to have a finite variance. In case the noise is white, this corresponds to the classical output-error noise framework.

B. Uniqueness of the Parametrization

The considered model representation is not unique. Beyond the well-known arbitrary state-space transformation that defines an equivalence class of models around [1] and [2], also the neural network representation of the static nonlinearity is not uniquely parametrized. A simple permutation of the inner and other weights and biases can result in the same input-output behavior of the neural network [3]. Beyond the non-uniqueness of the state space and the static nonlinear functions separately, the linear terms denoted by the state-space matrices \( A, B, C \) and \( D \) can also be incorporated in the static nonlinear functions \( \hat{f} \) and \( \hat{g} \) in eq. (2).

C. State-Space Neural Network

As is argued in [9], a state-space representation using a neural network to represent the state and output equations can be interpreted as a specific form of a recurrent neural network, denoted as a state-space neural network (SS-NN). The recurrent neural networks representing eq. (1) and eq. (2) are depicted in Figure 1. Equation (1) becomes the following when written down as a recurrent neural network:

\[
\begin{align*}
x(k + 1) &= W_x\sigma \left( [W_{fx} & W_{fu}] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + b_f \right) + b_x, \\
y_0(k) &= W_y\sigma \left( [W_{gx} & W_{gu}] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + b_g \right) + b_y,
\end{align*}
\]

(4)

the model type described by these equations are denoted as a state-space neural network (SS-NN). The same can be done for eq. (2), denoted as a generalized residual state-space neural network (gR-SS-NN):

\[
\begin{align*}
x(k + 1) &= [A & B] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \\
&\quad + \tilde{W}_x\sigma \left( [\tilde{W}_{fx} & \tilde{W}_{fu}] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \tilde{b}_f \right) + \tilde{b}_x, \\
y_0(k) &= [C & D] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} \\
&\quad + \tilde{W}_y\sigma \left( [\tilde{W}_{gx} & \tilde{W}_{gu}] \begin{bmatrix} x(k) \\ u(k) \end{bmatrix} + \tilde{b}_g \right) + \tilde{b}_y,
\end{align*}
\]

(5)

Observe that the inclusion of an explicit linear part in eq. (2) and (5) can be interpreted as a generalized form of a so-called residual network or resNet [17]. Instead of including a unity weighting matrix in the linear layers, a fully connected linear weighting is employed in this paper to model the ‘linear’ dynamics, the residuals that are present on top of these linear dynamics are captured by the neural network layers that have nonlinear activation functions (layers 1 and 3 in the bottom Figure 1). It is illustrated in this paper that, together with an improved initialization, the inclusion of the explicit linear part improves the estimation of the SS-NN model significantly.

III. ESTIMATING A SS-NN MODEL

A. Cost Function Optimization

The model parameters are obtained as the minimization of the mean squared simulation error:

\[
V_N(\theta) = \frac{1}{N} \sum_{k=1}^{N} (y(k) - \hat{y}(k|\theta))^2,
\]

(6)

\[
\hat{\theta} = \arg\min_{\theta} V_N(\theta),
\]

(7)

where \( \hat{y}(k|\theta) \) is the simulated output of the SS-NN model given the parameter vector \( \theta \) and \( y(k) \) is the noisy system output. The parameter vector \( \theta \) contains all the model parameters: the state-space matrix entries and the weights and biases of the neural network representing the static nonlinearities \( f(\cdot) \) and \( g(\cdot) \) or \( \hat{f}(\cdot) \) and \( \hat{g}(\cdot) \) respectively. \( N \) represents the total number of samples over which the cost function is computed.

Eq. (6) is nonlinear in the parameters and typically not convex but its gradients can be efficiently computed. Hence the Levenberg-Marquardt algorithm [18], a gradient descent-type algorithm, is used to minimize the cost function. The required gradients can efficiently be obtained using the backpropagation-through-time algorithm [19]. A ‘data-driven coordinate frame’ is used to get rid of equivalent gradient directions (resulting in a rank deficient Jacobian) due to the non-uniqueness of the model representation. In practice this
is achieved using the singular value decomposition of the Jacobian [20].

IV. Initialization of SS-NN

Due to the use of the Levenberg-Marquardt (or any other gradient-based) optimization algorithm, an initial guess of the parameter values is required. The subsections below describe the use of 3 different parameter initialization schemes.

The initialization is performed under the assumption that the input signal is zero mean and has been normalized to have a standard deviation equal to 1.

A. Random Initialization

Random initialization of the parameters is the standard approach in most neural network training algorithms. However, such a random initialization needs to be performed with caution as one can easily obtain unstable initial models, which most often lead to bad final estimates, or simply cause the algorithm to stop the optimization due to exploding gradients, or due to an exploding model output.

Here we consider the random initialization of the layer weights and biases using a uniform distribution with a support from -1 to 1 (denoted by \(U(-1, 1)\)). Except for some layer weights of eq. (5), they are initialized as zeros to avoid unstable initial estimates. The uniform random initialization of the parameters is the common approach when training a neural network [21]. Table I gives a detailed overview of the random weights and bias initialization scheme. The normalization of some weights with \(\sqrt{n}\) or \(\frac{1}{\sqrt{n}}\) ensures that the input of the nonlinear activation function does not grow too large for an increasing state or input dimension.

| Table I | Random Weights and Bias Initialization. All Weights and Bias Matrices Are of Appropriate Dimension. |
|---------|---------------------------------------------------------------|
| \(W_a\) | \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) |
| \(W_{fx}\) | \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) |
| \(W_{fy}\) | \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) |
| \(W_{gx}\) | \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) |
| \(W_{gy}\) | \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) \(U(-1, 1)\) |
| \(b_x\) | 0 \(b_f\) \(b_y\) \(b_q\) |
| \(b_f\) | 0 \(b_f\) \(b_y\) \(b_q\) |
| \(b_y\) | 0 \(b_f\) \(b_y\) \(b_q\) |
| \(b_q\) | 0 \(b_f\) \(b_y\) \(b_q\) |
| \(A\) | / \(U(-1, 1)\) / / |
| \(B\) | / \(U(-1, 1)\) / / |
| \(C\) | / \(U(-1, 1)\) / / |
| \(D\) | / \(U(-1, 1)\) / / |

B. Initialization using a Linear Approximation

An alternative initialization scheme was proposed in [9] for a SS-NN following eq. (1). This scheme makes use of the well known LTI identification tools that are available in the literature [22], [23]. These tools can obtain a linear approximation of the nonlinear system, expressed in the state-space matrices \(A_{LTI}, B_{LTI}, C_{LTI}, D_{LTI}\). The Best Linear Approximation framework [23]–[25] is a well established framework that can guide the practitioner in obtaining such a linear approximate model of a nonlinear system. The state-space matrices of the linear approximate model are normalized such that each of the states has a standard deviation equal to 1 [26].

1) Random Weights in the Linear Layers: Suykens and co-authors [9] observed that such a linear initial guess can be incorporated in a SS-NN by making use of the linear regime of some typical activation functions such as the hyperbolic tangent function if the number of neurons in the nonlinear layers is larger than the number of states in the system-to-be-modeled. Table II details the initialization of a SS-NN using a previously obtained LTI approximation of the nonlinear system, represented in state-space form. The factor \(\gamma\) is chosen such that the nonlinear activation function operates in a linear regime for the given training dataset. Additional random values are used to initialize some of the weights of the layers with linear activation functions, while still ensuring that the initial model behaves as the linear approximation we started from.

2) Random Weights in the Nonlinear Layers: Observe that [9] initializes the linear layer with random numbers, and uses zero weights for the nonlinear layer weights that are not initialized by the linear approximation. This often results in a suboptimal initial estimate. This paper proposes to do this the other way around, random weights in the nonlinear layer and zero weights in the linear layers, works better in the benchmark examples in Section V of this paper. Intuitively one can understand this by observing that using random weights and biases in the nonlinear layer generates a pool of nonlinearly transformed outputs which the estimator can pick from using the linear weights during optimization. Again, the initialization scheme is detailed in Table II.

3) Explicit Linear Dynamics - generalized Residual Network: The gR-SS-NN structure offers a natural way to initialize the neural network using a linear approximation. The linear state-space matrices are directly used to initialize the \(A, B, C, D\) matrices in eq. (5). This leaves quite some flexibility in initializing the weights and biases of the nonlinear layers. As in the previous initialization approach, random weights and biases in the nonlinear layer generates a pool of nonlinearly transformed outputs which the estimator can pick from using the linear weights during optimization. The full initialization scheme is detailed in Table II.

Since this initialization scheme does not make explicit use of the linear region of the chosen activation function, a broader set of activation functions, such as radial basis functions or rectified linear units, can be considered for the gR-SS-NN structure.

V. Benchmark Results

The proposed model structures are compared on two different nonlinear system benchmarks: the Bouc-Wen [13], [14] and the Wiener-Hammerstein [15] datasets. The effectiveness of the different initialization schemes are studied using a Monte-Carlo simulation on both benchmark datasets.
A. Bouc-Wen Benchmark

1) System and Data: The Bouc-Wen system is a mass-spring-damper system with a hysteretic nonlinearity. It is available as a simulation script allowing users to generate their own data for identification. We used one period of a random phase multisine input signal [23]. The signal was $N = 8192$ samples long and it excites the full frequency grid between 5 Hz and 150 Hz. The input signal amplitude is 50 Nms.

Two test datasets are available for the Bouc-Wen benchmark system: a multisine and a sinesweep dataset. The multisine test output is obtained as the steady-state response of the system excited by a random phase multisine of $N = 8192$ samples long, with the same frequency range and amplitude as the input used during estimation. The sinesweep test output is obtained by exciting the system, starting from zero initial conditions, with a sinesweep signal with an amplitude of 40 N. The frequency band from 20 to 50 Hz is covered at a sweep rate of 10 Hz/min.

A detailed description of the Bouc-Wen benchmark system is provided in [13], [14], the data can be obtained from www.nonlinearbenchmark.org.

2) Model Settings: A 3rd order state-space model ($n_x = 3$) is used, in line with previous findings [27]. A total of 15 neurons ($n_n = 15$, tansig activation function) are used to represent the static nonlinearities of both the state and the output equation.

The LTI approximation used to initialize the weights and biases of the SS-NN is obtained by the time-domain prediction error method as implemented in Matlab by the function ssest. This function initializes the parameter estimates using either a subspace approach or an iterative rational function estimation approach. The state-space matrices are refined subsequently using the prediction error minimization approach [22].

3) Results: As a measure of model quality, the simulation RMSE (root mean squared error) on the multisine and sinesweep test dataset is reported:

$$e_{\text{RMSE}} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} (y(k) - \hat{y}(k|\theta))^2}, \quad (8)$$

where $y(k)$ is the observed test output and $\hat{y}(k|\theta)$ is the simulated output using the estimated model. To make sure the model output is in steady state for the multisine dataset two periods are simulated and the RMSE is calculated on the second period. The sinesweep data is not in steady state, hence, the first 2000 samples are ignored when calculating the RMSE to allow the transient to decay.

Since all of the proposed initialization schemes contain some random initial parameters, a Monte Carlo simulation has been performed of 100 Monte Carlo runs.

The obtained gR-SS-NN models using linear approximation initialization obtain state of the art results, with an error which is almost half of the one reported in [27] ($7.14 \times 10^{-6}$ versus $1.2 \times 10^{-5}$ RMSE on the multisine test data). The error in both the time- and frequency domain of the median gR-SS-NN model (in RMSE sense) is shown in Figure 2. The multisine test signal. These figures also illustrates that the obtained has an RMSE of almost a factor 100 lower than a LTI model.

Furthermore, the gR-SS-NN model with linear approximation initialization outperforms all the other SS-NN models using various initialization schemes. This can be observed from the boxplots shown in Figure 3. The fully random initialization methods occasionally result in a high quality estimate, but the median and the spread of the resulting models is very large. The three models based on a linear approximate model have a much smaller spread, and in the case of the gR-SS-NN and the SS-NN improved initialization (Table II column 3 and 2 respectively) this results also in a much lower median RMSE.

The introduction of the generalized residual network, in other words the explicit inclusion of linear dynamics in the SS-NN, clearly improves the model behavior for both the random and linear approximate-based initialization methods. This can both be observed from the boxplots in Figure 3 but also in the evolution of the RMSE over the training epochs shown in Figure 4. The RMSE of the gR-SS-NN models starting from a linear approximation decay much faster, resulting in the need for much less training epochs. Since the gR-SS-NN models only have slightly higher complexity compared to the SS-NN models, the training time can be reduced significantly.

B. Wiener-Hammerstein Benchmark

1) System and Data: The Wiener-Hammerstein benchmark [15] features a Wiener-Hammerstein block-oriented

| TABLE II |
| --- |
| **Initialization starting from a linear approximation. All weights and bias matrices are of appropriate dimension.** |

|   | SS-NN following [9] | SS-NN this paper | gR-SS-NN this paper |
|---|---|---|---|
| $W_z$ | $\frac{1}{2}[I_{n_z} \ 0]$ | $\frac{1}{2}[I_{n_z} \ 0]$ | 0 |
| $W_{fx}$ | $[\gamma_{ALTI} \ 0]$ | $[\gamma_{ALTI} \ 0]$ | $U(-1,1)/\sqrt{n_u}$ |
| $W_{fu}$ | $[\gamma_{BLTI} \ 0]$ | $[\gamma_{BLTI} \ 0]$ | $U(-1,1)/\sqrt{n_u}$ |
| $W_y$ | $\frac{1}{2}[I_{n_y} \ 0]$ | $\frac{1}{2}[I_{n_y} \ 0]$ | 0 |
| $W_{gx}$ | $[\gamma_{C_{LTI}} \ 0]$ | $[\gamma_{C_{LTI}} \ 0]$ | $U(-1,1)/\sqrt{n_u}$ |
| $W_{gy}$ | $[\gamma_{D_{LTI}} \ 0]$ | $[\gamma_{D_{LTI}} \ 0]$ | $U(-1,1)/\sqrt{n_u}$ |
| $b_x$ | 0 | 0 | 0 |
| $b_f$ | 0 | 0 | $U(-1,1)$ |
| $b_y$ | 0 | 0 | $U(-1,1)$ |
| $A$ | 1 | 1 | $ALTI$ |
| $B$ | 1 | 1 | $BLTI$ |
| $C$ | 1 | 1 | $C_{LTI}$ |
| $D$ | 1 | 1 | $D_{LTI}$ |

The models are trained using the Matlab Deep Learning toolbox. The developed algorithm is available through: https://gitlab.tue.nl/mschouke/ss-nn
Fig. 2. Time- and frequency-domain test results of the median gR-SS-NN model initialized using a linear approximation on the Bouc-Wen multisine test data. The true system output is shown in red, the residuals obtained with a 3rd order LTI model are shown in blue and the residuals obtained with the gR-SS-NN model are shown in green.

electronic circuit: a one-sided saturation diode-resistor non-linearity is sandwiched in between two 3rd-order low-pass linear filters. Only noise at the output of the system is present (SNR≈60dB). The training and test input data consist of a filtered white Gaussian noise sequence (10kHz cut-off frequency). The benchmark data consist of and more information on the benchmark system and previous benchmark results is available through: www.nonlinearbenchmark.org.

2) Model Settings: A 6th order state-space model (n_x = 6) is used, in line with the system description [15]. A total of 15 neurons (n_n = 15, tansig activation function) are used to represent the static nonlinearities of both the state and the output equation.

The LTI approximation used to initialize the weights and biases of the SS-NN is obtained in the same way as outlined for the Bouc-Wen benchmark results.

3) Results: The general observations are in line with the observations obtained for the Bouc-Wen benchmark: the use of an linear approximation to initialize the SS-NN model greatly reduces the spread of the quality of the obtained models, while the introduction of the explicit linear part (generalized residual net) lowers the RMSE obtained by these models significantly (Figure 6) and reduces the training time (Figure 4).

Furthermore, it can be observed that the best obtained RMSE (0.34 mV) is quite close to the best result reported in the literature (0.278 mV, [28]). The performance of a gR-SS-NN model with the median RMSE can be observed in Figure 5. This figures also illustrates that the best obtained model has an RMSE of almost a factor 100 lower than a LTI model.

VI. CONCLUSIONS

This paper proposed two important improvements in the initialization of state-space neural network models for dynamical systems, compared to the existing approaches outlined in the literature. Firstly, starting from a linear approximate model, some weights and biases of the SS-NN can be initialized, while the other weights are initialized by random numbers. It turns out to be valuable to randomly initialize the weights and biases of the nonlinear layers, rather than the linear ones. Secondly, the inclusion of an explicit linear term in parallel to the nonlinear function results in a significantly improved training behavior, the resulting SS-NN is denoted as a generalized residual state-space neural network. This is in line with the positive results obtained using residual neural networks in the machine learning literature. These observations have been obtained by studying Monte-Carlo simulation results on two well-established benchmark systems.

REFERENCES

[1] J. Schoukens and L. Ljung. Nonlinear system identification: A user-oriented road map. IEEE Control Systems Magazine, 39(6):28–99, Dec 2019.
[2] S.A. Billings. Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains. John Wiley & Sons, Ltd., West Sussex, United Kingdom, 2013.
[3] M. Schoukens and K. Tiels. Identification of block-oriented nonlinear systems starting from linear approximations: A survey. Automatica, 85:272–292, 2017.
[4] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, and R. Pintelon. Identification of nonlinear systems using polynomial nonlinear state space models. Automatica, 46(4):647–656, 2010.
[5] K. Tiels. PNLS 1.0: A polynomial nonlinear state-space toolbox for Matlab. Technical report, Vrije Universiteit Brussel, 2018.
Fig. 4. Evolution of the RMSE on the training dataset over the training epochs for both the Bouc-Wen and the Wiener-Hammerstein benchmark. The thick green line visualizes the median error of the training of a gR-SS-NN model with a linear approximation initialization (Table II, column 3), the green zone around it shows the 10% - 90% region of the RMSE. The thick blue dashed line and the colored zone around visualizes the same for the training of a SS-NN model using the improved linear approximation initialization (Table II, column 2).

Fig. 5. Time- and frequency-domain test results of the median gR-SS-NN model initialized using a linear approximation on the Wiener-Hammerstein test data. The true system output is shown in red, the residuals obtained with a 6th order LTI model and with the gR-SS-NN model are shown in bulle and green respectively.

Fig. 6. Boxplots of the test results of the Monte Carlo simulation obtained SS-NN models using the various initialization methods: gR-SS-NN LTI init. (Table II, column 3), SS-NN LTI improved init. (Table II, column 2), gR-SS-NN LTI init. (9) (Table I, column 1), gR-SS-NN Random init. (Table II, column 2), SS-NN Random init. (9) (Table I, column 1).

[6] T.B. Schön, A. Wills, and B. Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

[7] T.B. Schön, A. Svensson, L. Murray, and F. Lindsten. Probabilistic learning of nonlinear dynamical systems using sequential Monte Carlo. *Mechanical Systems and Signal Processing*, 104:866–883, 2018.

[8] J.P. Noël and G. Kerschen. Frequency-domain subspace identification for nonlinear mechanical systems. *Mechanical Systems and Signal Processing*, 40(2):701–717, 2013.

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

[10] A. Marconato, J. Sjöberg, and J. Schoukens. Initialization of nonlinear state-space models applied to the Wiener-Hammerstein benchmark. *Control Engineering Practice*, 20(11):1126–1132, 2012.

[11] M. Forgione and D. Piga. Model structures and fitting criteria for system identification with neural networks. *arXiv preprint: arXiv:1911.13034*, 2019.

[12] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[13] J.P. Noël and M. Schoukens. Hysteretic benchmark with a dynamic nonlinearity. In *4TU.ResearchData*, 2016. Dataset: [https://doi.org/10.4121/12967592](https://doi.org/10.4121/12967592).

[14] M. Schoukens and J.P. Noël. Three benchmarks addressing open challenges in nonlinear system identification. In *20th World Congress of the International Federation of Automatic Control*, pages 446–451, Toulouse, France, 2017.

[15] J. Schoukens, J.A.K. Suykens, and L. Ljung. Wiener-Hammerstein benchmark. In *15th IFAC Symposium on System Identification (SYSID)*, Saint-Malo, France, July 2009.

[16] M. Hudson Beale, M.T. Hagan, and H.B. Demuth. *Neural Network Toolbox User’s Guide.* The MathWorks, Inc., 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of Applied Mathematics*, 2(1):164–168, 1944.

[19] P.J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356, 1988.

[20] A. Wills and B. Ninness. On gradient-based search for multivari-