technische universität dortmund

Process Dynamics and Operations

Technical University Dortmund
Department of Biochemical and Chemical Engineering
Process Dynamics and Operations Group
Prof. Dr.-Ing. Sebastian Engell

Bachelor Thesis

Improving the Generalisation of NARX Models by the Integration of Physical Constraints

by

Ümmühan Magdalena Strebel

Supervision: Jens Ehlhardt, M. Sc.

Examiner: Prof. Dr.-Ing. Sebastian Engell

Dortmund, 04.07.2022

Acknowledgments

This work was done under the direction of Prof. Dr.-Ing. Sebastian Engell in the Process Dynamics and Operations Group of the Department of Biochemical and Chemical Engineering at the Technical University Dortmund. I would like to thank him for his constructive feedback in the midterm.
Special thanks go to my supervisor Jens Ehlhardt for his guidance throughout this thesis. Without his patience and support, this thesis would not have been possible.

Abstract

Nowadays, neural networks find more and more application in engineering problems. In practice, however, it is often not so easy to obtain training data of appropiate quality and sufficient quantity. The goal of this bachelor thesis is to extend the generalization of a neural network to simulate well even with poor data availability. This should be realised with the help of a physics-based loss function. The approach of this problem was tested and evaluated on a simple one-tank system from [1]. Overall, an improvement of the generalization was observed, however, depending on the perturbation factor, not always every physical constraint led to an improvement. This had the consequence that those physical constraints got a lower influence. Consequently, the effect of different formulations of physics constraints was analyzed and the algorithm tuned accordingly.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Neural networks are able to represent any continuous non-linear function based on input-output data as long as the network structure is large enough. But since there is not always enough data of sufficient quality available a neural network is often no longer adequate.[3] To prevent this problem, an approach is to extend the neural network with engineering or scientific knowledge. The extension of the neural network takes place with the help of embedding physical constraints into the training of neural networks. The physical constraints are modelled in the loss function which is to be minimized. [4] [5] Because of the additional penalty term in the loss function, these constraints are called *soft constraints*. They are not necessarily always satisfied, but in case of non-compliance with the constraint, an effort is made to minimize the resulting error. [6]

In chapter two, the theoretical background of this thesis is propounded and the concept of a neural network is explained in detail. Furthermore, the later used non-linear autoregressive models with exogenous inputs are explained. Afterwards, the concept of physics-guided neural network is presented in which it is described how the physical constraints are added into the training of neural networks.
In chapter three, the case study, that will be considered, is discussed. In addition to that, the structure of the physics-guided neural network is presented in more detail.
In chapter five the generated results are presented and classified.
In chapter six there is a final evaluation of the results and an outlook for future research topics.

Chapter 2

Theoretical Background

The theoretical background of this thesis is explained in this chapter. First it is defined what artificial neural networks are. This includes the question of what they consist of and how they are trained. Subsequently, it is explained how nonlinear autoregressive models with exogenous inputs based on artificial neural networks are set up. Finally the process embedding physical constraints into the loss function of an artificial neural network is explained.

2.1 Neural Network

The idea of an artificial neural network, often only called neural network (NN), originates from biological neural networks in the nervous system of living organisms. The neurons themself act as information transmitters between the brain and the nervous system. In the context of Machine Learning a NN is a mathematical (nonlinear) function. NN's are used in a wide range of applications such as signal processing, pattern recognition and regression. [7]

The structure is composed of so called *neurons* that are grouped in *layers*. The mathematical basics are also explained in the following sections which are important to understand how the output of each layer is generated and how the training and therefore the adjustment of the unknown parameters works.

2.1.1 Single Neuron

The single neuron is the simplest case of a NN as it consists of only one neuron with the inputs $\vec{x}$, weights $\vec{w}$ and an output $\hat{y}$. Although figure 2.1 depicts one output, the amount of inputs and outputs in general is arbitrary.

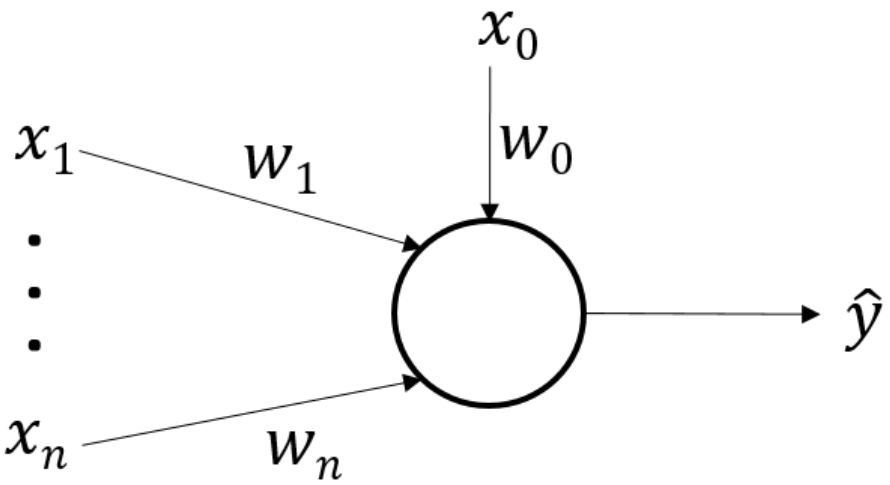

Figure 2.1: Simplified model of a single neuron. (Graphic adapted from [2, p. 452, Fig. 2(b)])

$$\vec{x} = \begin{pmatrix} x_0 \\ \vdots \\ x_n \end{pmatrix}, \vec{w} = \begin{pmatrix} w_0 \\ \vdots \\ w_n \end{pmatrix} \tag{2.1}$$

n is the number of inputs, x_0 is equal to 1 and w_0 is the bias.
The mathematical notation of the output $\hat{y}$ is as follows:

$$\hat{y} = g\left(\sum_{i=0}^{n} w_i x_i\right) \tag{2.2}$$

The function $g(\cdot)$ is a nonlinear activation function, as it will be explained in section 2.1.3. In figure 2.2, the mathematical layout of a single neuron is given. [8] [2]

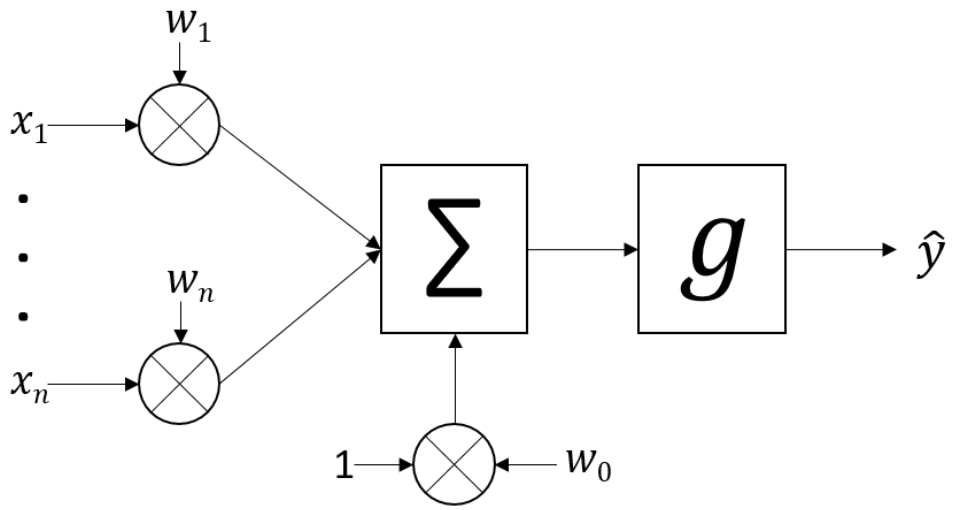

Figure 2.2: Detailed model of a single neuron system. (Graphic adapted from [2, p. 452, Fig. 2(a)])

2.1.2 Multi-layer Perceptron

Typically, mutliple neurons are put into one (*shallow*) or more layers (*deep*). With a single layer, it is called a *single-layered perceptron* and with more than one layer it turns into a *multi-layered perceptron*. [9]
The input of one layer is the output of the previous one and can be described as

$$a_j^{(l)} = g\left(\sum_{i=0}^{n_{(l-1)}} w_{j,i}^{(l-1)} \cdot a_i^{(l-1)}\right) \tag{2.3}$$

with l as the number of the layers, j the number of neurons, $w_{j,0}^{(1)}$ as the bias, $w_{j,i}^{(1)}$ as the weights and a_j as the activation of the neuron. The activation is transformed by using a nonlinear activation function $g(\cdot)$. The output layer also is simply an addition of the multiplication $w_{j,i}^{(j)} \cdot x_i$ without consideration of an activation function. [10]
The number of layers and neurons are hyperparameters. Hyperparameters are changeable and adapted to the NN of the observed case study. Its selection is a user's choice to customize the NN.

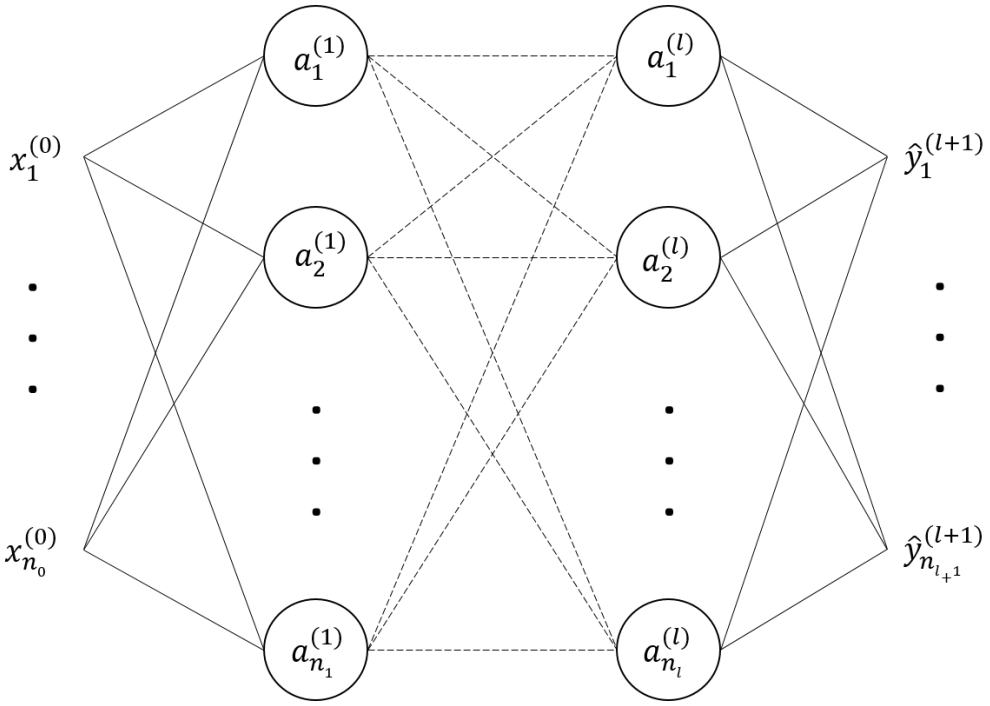

Figure 2.3: Model of a multi-layeedr perceptron with inputs $\vec{x}$ and outputs $\vec{\hat{y}}$.

2.1.3 Activation Functions

The activation function is a biologically inspired mechanism where neurons are activated by different stimuli. This can be represented by artificial neurons: if the value $a_j^{(l)}$ is closer to one the neuron is more likely to be activated. With a value of $a_j^{(l)}$ closer to zero the neuron is less activated. The activation function is a hyperparameter. Following activation functions are possible options:

1. *Sigmoid* function
2. *Hyperbolic Tan* function
3. *ReLU* function.

The activation function of a neuron defines the output given the inputs. This happens by transforming the sum of weighted outputs of the previous layer to a value between a lower and upper limit. The *sigmoid*, *hyperbolic tan* or *ReLU* function are used in this thesis and therefore are presented in the following section.

2.1.3.1 Sigmoid Function

The *sigmoid* function maps the inputs value in the range [0, 1]. Whereby, large negative inputs result in a value close to 0. Vise versa, large positive inputs result in a value close to one. [7]

$$g(x) = \frac{1}{1 + exp(-\sigma x)} \tag{2.4}$$

$$\frac{dg(x)}{dx} = \sigma \cdot g(x) \cdot [1 - g(x)] \tag{2.5}$$

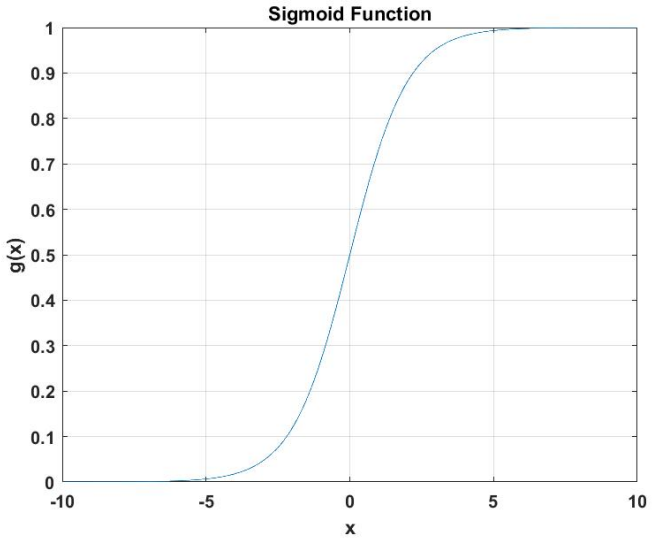

Figure 2.4: The *sigmoid* activation function.

2.1.3.2 Hyperbolic Tan Function

Compared to the *sigmoid* function, the *tanh* function compresses the input values into a range of [-1 1]. [7, p. 19]

$$g(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} = \frac{1 - exp(-2x)}{1 + exp(-2x)} \tag{2.6}$$

$$\frac{dg(x)}{dx} = [1 + g(x)]\,[1 - g(x)] \tag{2.7}$$

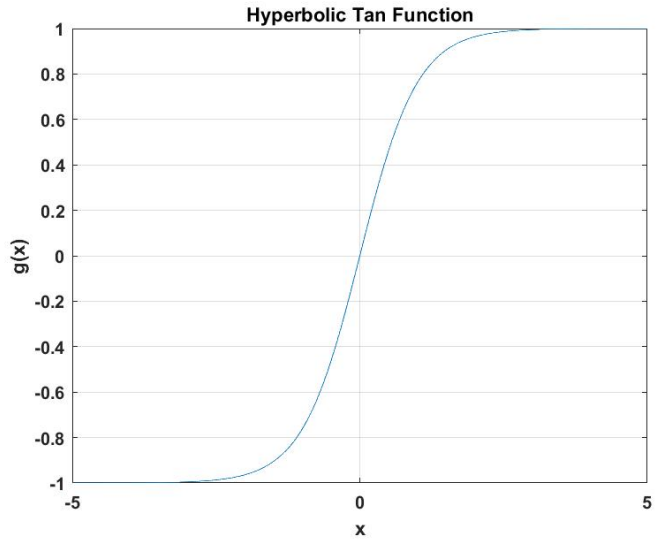

Figure 2.5: The *hyperbolic tan* activation function.

2.1.3.3 ReLU Function

ReLU is an abbreviation and stands for *Rectified Linear Unit*. The *ReLU* function does not squish the value into a range, but will set all negative inputs to zero. The upper and lower limit is therefore $[0, +\infty]$. The *ReLU* function has become popular especially in the context of deep learning in recent use. Additionally, it will be used for the embedding of the physical constraints. [11]

$$g(x) = ReLU(x) = max(0, x) \tag{2.8}$$

$$\frac{dg(x)}{dx} = \begin{cases} 1, x > 0 \\ 0, x < 0 \end{cases} \tag{2.9}$$

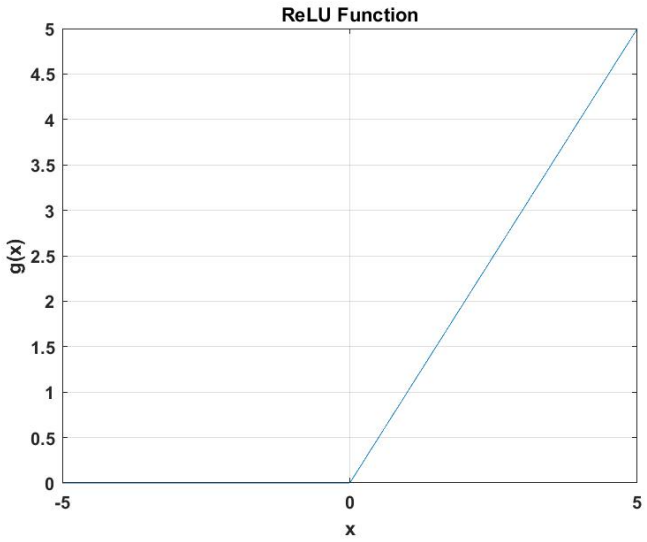

Figure 2.6: The *ReLU* activation function.

2.1.4 Training

The parameters of a neural network are randomly initialized in the beginning of its training. In regression problems, the goal of the training is to adapt the weights $\vec{w}$ of the neurons such that it represents the data accurately. Therefore, a sum of errors between the data and the predictions of the NN, which is the so called loss function $J(\vec{w})$, is minimized. It is common to use data that has not been used in the training to validate a trained network. This way the accuracy of the NN and its generalisation capabilities can be checked. Training and validation data sets can be obtained either by splitting one large data set or by generating two different kind of sets. [10] [12]

2.1.4.1 Loss Function

As stated above, the essence of the training is considered as minimizing the sum of the errors. Mathematically this is achieved by minimizing the loss function $J(\vec{w})$.

$$J(\vec{w}) = \underbrace{\frac{1}{m}\sum_{i=1}^{m}\left(\hat{y}(\vec{w}, x^{(i)}) - y^{(i)}\right)^2}_{MSE} \tag{2.10}$$

With m as the number of sampling points and y as the actual data. A common choice in regression problems is the "mean squared error" (MSE) (2.10).

2.1.4.2 Gradient Descent

Gradient descent is used as a numerical computation of an optimization problem in which the goal is to find the local minimum of a loss function J. The solution of gradient descent is based on following iteration process:

$$x_{k+1} = x_k + \alpha d_k \tag{2.11}$$

With x_{k+1} as the next iterate, x_k the current iterate, α as the step size and d_k as the current search direction. The search direction is given by the negative gradient of the loss function with respects to the parameters of the neural network. In figure 2.7, the gradient descent iterates approach the minimum of a quadratic function and is shown as following equation: [13]

$$d_k = - \bigtriangledown f(x_k) \tag{2.12}$$

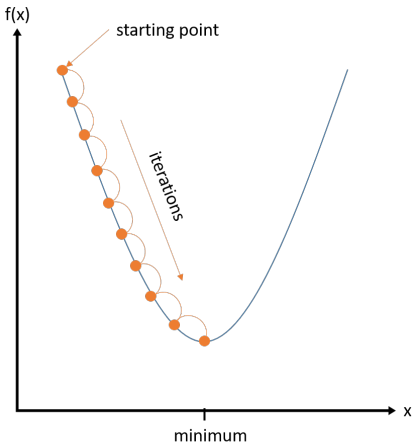

Figure 2.7: Simplified visual representation of gradient descent.

2.1.4.3 Gauss-Newton Algorithm

The *Gauss-Newton* method is used to numerically solve nonlinear optimization problems. These are of the type of least-squares problems where the sum of squared errors are minimized. While performing this method, the objective function is linearised.

$$\min_{x \in \mathbb{R}} \| f(x_k) + J(x_k)(x - x_k) \|_2^2 \tag{2.13}$$

The next iterate x_{k+1} is calculated as follows:

$$x_{k+1} = x_k - \alpha_k \left((J(x_k)^T J(x_k))^{-1} J(x_k)^T \cdot f(x_k) \right) \tag{2.14}$$

With α_k as the step size and J as the *jacobian* matrix of f.

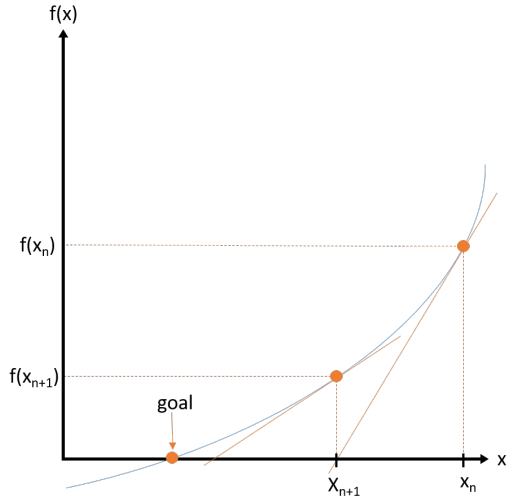

Figure 2.8: Simplified visual representation of Newton method.

2.1.4.4 Levenberg-Marquardt Algorithm

Levenberg-Marquardt is an algorithm for solving nonlinear least-squares problems and a combination of the just described gradient descent and *Gauss-Newton* algorithm. The iteration from the *Gauss-Newton* algorithm is extended as follows: the term $J(x_k)$ is extended by Δ: The iteration is similar to the one in the *Gauss-Newton* algorithm.

$$x_{k+1} = x_k - \alpha_k \left((J(x_k)^T J(x_k) + \Delta)^{-1} J(x_k)^T \cdot f(x_k) \right) \tag{2.15}$$

Δ is a diagonal matrix which makes sure the term $J(x_k)^T J(x_k)$ stays positive definite and α_k is the step size. In each iteration, the value of Δ is adapted based on the success of the previous iteration. If Δ has a small value, the iteration step is computed through *Gauss-Newton* method and if it is large, it is computed by gradient descent. [14]

2.2 Physics Guided Neural Networks

The kind of physics-guided neural networks (PGNN) used in this thesis add knowledge in the loss function of a NN, which is why the loss function is also

called physics-based loss function. It connects the ability to train with a high amount of data while taking physical conditions into account. In conclusion the advantage of PGNN is that the outputs are not only being based on training data, but also on constraints that are reformulated physical relations. Those relations can be bounds on outputs, balance equations, cause-effect relationships in which the change of one parameter influences the change of another.

The development of a PGNN can be divided into two parts: First setting up the neural network with its loss function and afterwards embedding the physical constraints into it. The loss function is extended with an term L_{Phy} as follows:

$$Loss = \underbrace{L_{STD}(y, \hat{y})}_{Empirical\ loss} + \underbrace{L_{Reg}(\theta_{net})}_{Regularisation} + \underbrace{L_{Phy}(\hat{y})}_{Physical\ constraints} \tag{2.16}$$

Since the physical constraints are assumed to be equally well on unseen data. The training data can be extended by unlabelled data sets. This can help achieving better generalisation performance even when the collected data is small and not fully representative. The loss function can be minimzied the same as in the chapter 2.1.4 explained. [5]
The term of the physical constraints L_{Phy} is formulated as follows:

$$L_{Phy}(\hat{y}) = \sum_{i=1}^{n} \lambda_{Phy,i} L_{Phy,i} \tag{2.17}$$

Each physical constraint is formulated in a term $L_{Phy,i}$ and weighted with a tuning parameter $\lambda_{Phy,i}$. The constraints are defined as equalities or inequalities:

$$g(\hat{y}, z) = 0 \tag{2.18}$$

$$h(\hat{y}, z) \leq 0 \tag{2.19}$$

Lastly they are embedded in the loss function as follows:

$$L_{Phy,i}(\hat{y}) = \| \, g(\hat{y}, z) \, \| + ReLU(h(\hat{y}, z)) \tag{2.20}$$

2.3 NARX Model

The *Nonlinear AutoRegressive model with eXogenous inputs* (NARX) is a neural network that can be used for a dynamic time series prediction. Exogenous inputs are external variables that influence the system. For example, an exogenous input can be a valve position and the predicted output can be a height. The NARX models used in this thesis can be evaluated in two configurations. In the closed-loop form past values of the current outputs are fed back as inputs to the NARX model (fig. 2.9). All of the inputs are fed in with a certain delay which include current and past values of the inputs. In training however, the open-loop configuration is used in which the NARX model is solely evaluated on measured data. [15] [16]

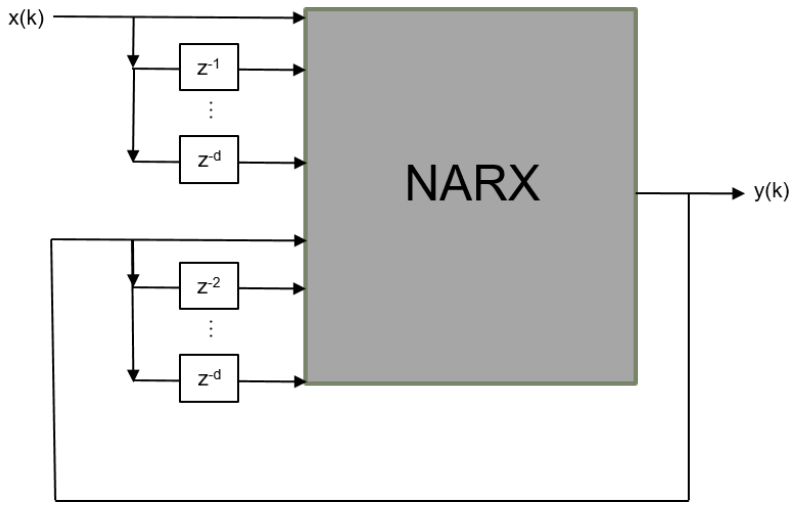

Figure 2.9: Simplified model of a closed-loop NARX model with exogenous inputs x, the output y and a delay d.

The mathematical equation of a NARX neural network can be stated as following:

$$\vec{\hat{y}}(k) = F\left(\underbrace{\vec{\hat{y}}(k-1), ..., \vec{\hat{y}}(k-d_y)}_{Autoregression\ of\ outputs}, \underbrace{x(k), x(k-1), ..., x(k-d_x)}_{Weighted\ moving\ average\ of\ inputs}\right) \tag{2.21}$$

With d_x and d_y as some delays, $\vec{\hat{y}}$ as outputs, x as the current and past exogenous inputs and k as the discrete time steps. Delays are hyperparameters and therefore a user's choice. [17]
The function F represents the NN. NARX networks are widely used for non-linear system identification such as waste water treatment plants ([18], [19]) or heat exchangers ([15]).

Chapter 3

Case Study: One Tank System

Goal of this chapter is to build up a nonlinear autoregressive exogenous model with a PGNN as its function. The considered case study is a one-tank system. First, optimal hyperparameters of the NARX model need to be determined. Therefore, the range of the individual hyperparameters are predetermined and subsequently the grid search is carried out to obtain optimal hyperparameters. The effectiveness of the physical constraints are tested by manipulating training and validation data of the PGNN.

The PGNN is implemented in MATLAB. The implementation of the system is divided into the NN and PGNN part, which are trained separately from each other. The optimization method is a *Levenberg-Marquardt* optimizer which was explained in section 2.1.4.4. First, the neural network is created without physical constraints to generate model parameters which are passed to the PGNN as initial model parameters.

3.1 Model of the One Tank System

The one tank system, as it can be seen at figure 3.1, has an inflow stream F_{in} and an outflow stream F_{out} which is regulated by a valve with the valve position u. The tank is filled up with a liquid to the height h.

The change of the height h over the time can be formulated in the following differential equation, adapted from [1]:

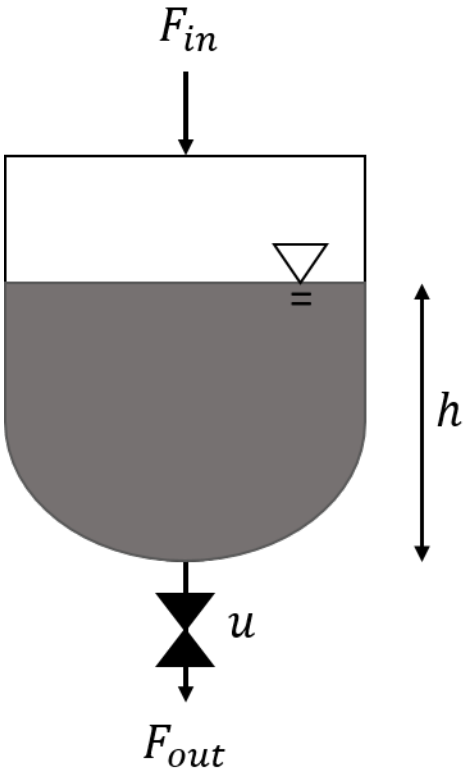

Figure 3.1: Schematic model of the one tank system with the inflow F_{in} and regulated outflow F_{out} through a valve with the position u.

$$\frac{dh}{dt} = \frac{F_{in}}{A} - \frac{1}{A} \cdot \underbrace{\frac{u \cdot \sqrt{h}}{\sqrt{c_{13} + u^2 + c_{23}}}}_{F_{out}} \tag{3.1}$$

The parameter A is the cross sectional area of the tank, the parameters c_{13} and c_{23} are parameters defining the behaviour of the valve.

3.2 Application in a NARX Model

As it was explained in 2.3 a NARX model uses current and past values of the output and exogenous variables. In this case the exogenous inputs are the

valve positions u and the output is the level of the liquid h. For validation, the NARX model is run in closed-loop configuration. The hyper parameters of the network will be determined by a grid search.

3.2.1 Optimal Hyper Parameter

3.2.1.1 Grid Search

To reduce the large amount of hyper parameter combinations a preliminary study was conducted. The goal of this study was to obtain a smaller range of hyperparameters in which a grid search is reasonable. The analysed ranges of the preliminary study reach from a neuron number between $[1, 100]$, a hidden layer number between $[1, 5]$ and a delay between $[1, 1000]$ and was observed by a random search where randomized combinations were viewed. This study was carried out with the *sigmoid*, *hyperbolic tan* and $ReLU$ function.

The observed ranges for the grid search and the chosen optimal hyper parameters can be seen on table 3.1. The decision of the small ranges results from the reason that the considered one-tank system has a low complexity.

Two criterions were taken into account when deciding for the optimal hyper parameters: the value of the MSE and the complexity of the network. Although the chosen hyperparameters did not have the lowest value of the MSE, the complexity of the system is relatively low. The schema of the NARX model with the optimal hyperparameters can be seen at figure 3.2.

For each combination of hyperparameters, five networks are trained and evaluated on validation data. The resulting MSE is the mean of the five MSE's which were computed before.

Table 3.1: Ranges of the grid search for finding optimal hyper parameters for the NARX model of the one tank system and its chosen hyper parameters.

	Activation Function	Layer number	Neuron number	Delay
Ranges[a]	*sigmoid*, *tanh*, $ReLU$	$[1, 3]$	$[1, 10]$	$[1, 5]$
Values[b]	*tanh*	1	3	2

[a] The values of the range limits of the grid search.
[b] The values of the chosen hyper parameters.

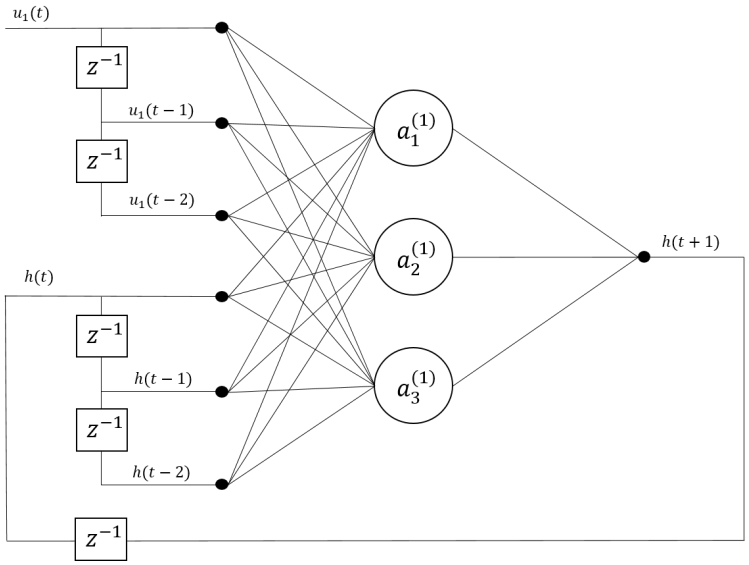

Figure 3.2: Specific NARX model of the one tank system with optimal hyper parameters.

3.3 Application in a NARX Model with Physical Constraints

To finalize the PGNN the NARX model of section 3.2 is extended by including the physical constraints. As in section 2.2 explained, these are formulated as terms $L_{Phy,i}$ which are added to the loss function.

3.3.1 Height Constraint

The height constraint is a limit of the level of the liquid in the tank. It cannot reach a negative value since the height starts with the bottom of the tank.

$$h \geq 0 \tag{3.2}$$

Transformed into an inequality constraint, the term reads as follows:

$$-h \leq 0 \tag{3.3}$$

The term is embedded into a ReLU function and can be inserted into the loss function in the following form:

$$L_{Phy,1} = ReLU(-h(\hat{y})) \tag{3.4}$$

3.3.2 Cause-and-Effect Constraint

When changing the valve position, an observation about an immediate change in height becomes detectable. The reason for this is that there is a direct correlation between the valve position u and fluid height h. These two parameters have a cause-and-effect relationship.

$$\overbrace{\frac{\Delta u}{\Delta t}}^{\uparrow} \Rightarrow \underbrace{\frac{\Delta h}{\Delta t}}_{\downarrow} \quad \wedge \quad \overbrace{\frac{\Delta u}{\Delta t}}^{\downarrow} \Rightarrow \underbrace{\frac{\Delta h}{\Delta t}}_{\uparrow} \tag{3.5}$$

Since the change of the valve position triggers a change of the liquid height that is opposite in sign, a multiplication of both will always result as a negative value:

$$\frac{\Delta u}{\Delta t} \cdot \frac{\Delta h}{\Delta t} < 0 \tag{3.6}$$

The formulated term as an inequality constraint for the loss function is

$$L_{Phy,2} = ReLU\left(\frac{\Delta h}{\Delta t} \cdot \frac{\Delta u}{\Delta t}\right) \tag{3.7}$$

The rate of change of the height is approximated using forward finite differences. The rate of change of the valve position is evaluated on a moving window. Therefore, the neural network is evaluated with the current paramater values on an additional amplitude modulated PRBS signal.

3.3.3 Speed Constraint

There are three approaches on how to consider the speed of the system by physical constraints. All of them are tested individually and collectively.

For the explanation of the speed constraints, a schematic step response of the one tank system with an entered valve signal is shown.

3.3.3.1 Speed Constraint 3.1

The first speed constraint contains the slope of the step response curve at T_{min} and T_{max} which can be seen on figure 3.3. The constraint is stated as follows:

$$\left|\frac{\Delta h}{\Delta t}\Big|_{T_{min}}\right| > 0 \quad \wedge \quad \left|\frac{\Delta h}{\Delta t}\Big|_{T_{max}}\right| = 0 \tag{3.8}$$

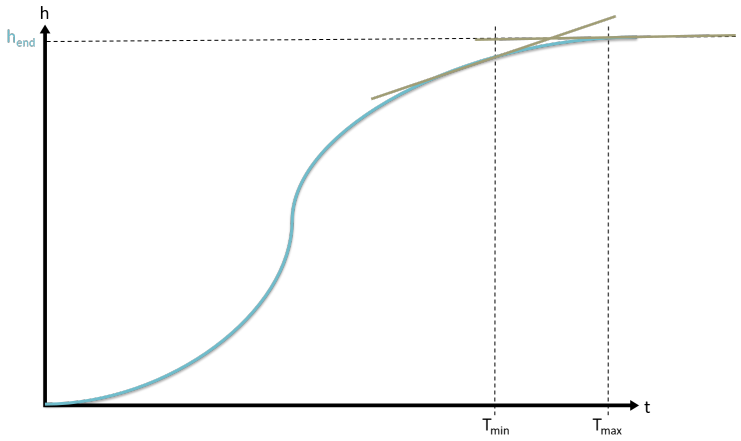

Figure 3.3: Schema of a step response of the one tank system with tangents at points T_{min} and T_{max} for the first speed constraint.

The term which can be inserted into the loss function reads as follows:

$$L_{Phy,3,1} = ReLU\left(\left|\frac{\Delta h}{\Delta t}\Big|_{T_{min}}\right|\right) + \left\|\frac{\Delta h}{\Delta t}\Big|_{T_{max}}\right\|_2 \tag{3.9}$$

3.3.3.2 Speed Constraint 3.2

The second constraint shows that the ratio between the values of the height at T_{min}, $h(T_{min})$ and the height at steady state h_{end} is less than an arbitrary percentage value, in this case 95 %, and can be formulated as the term stated in (3.11):

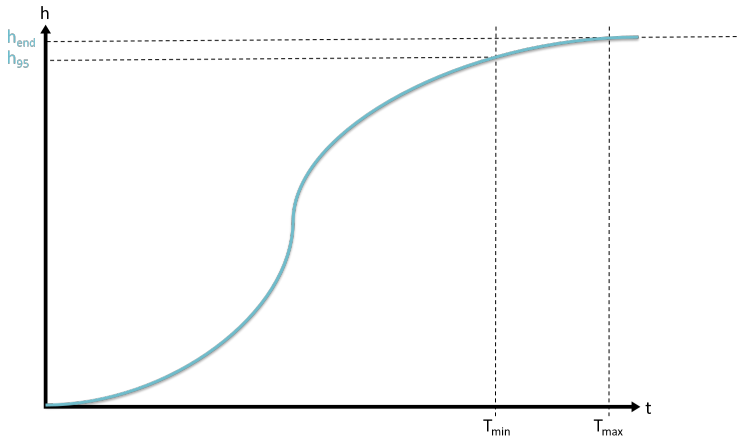

Figure 3.4: Schema of a step response of the one tank system with a tangent at point T_{min} for the second speed constraint.

$$\frac{h(T_{min})}{h_{end}} \leq 0.95 \quad \Leftrightarrow \quad \frac{h(T_{min})}{h_{end}} - 0.95 \leq 0 \tag{3.10}$$

$$L_{Phy,3,2} = ReLU\left(\frac{h(T_{min})}{h_{end}} - 0.95\right) \tag{3.11}$$

3.3.3.3 Speed Constraint 3.3

The third speed constraint divides the step response curve by a specific T_m and specifies that the areas marked in fig. 3.5 are of the same size.

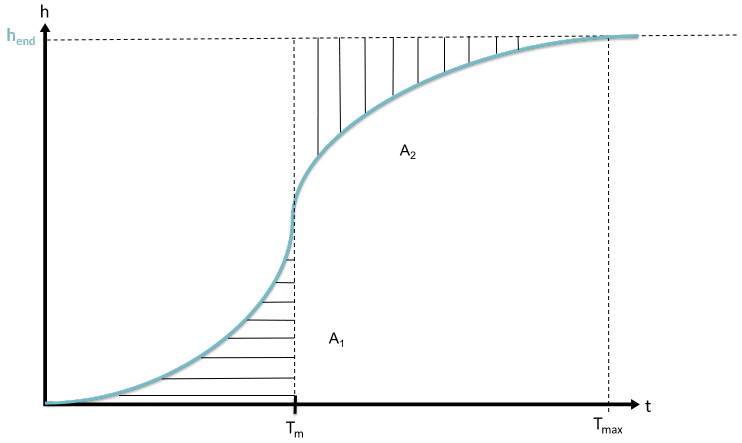

Figure 3.5: Schema of a step response of the one tank system with a specific T_m splitting the curve into the marked areas A_1 and A_2.

$$A_1 = A_2 \quad \Leftrightarrow \quad A_1 - A_2 = 0 \tag{3.12}$$

To calculate T_m, the sum of time constants is made.

$$T_m = \sum_{i=1}^{n} T_i \tag{3.13}$$

The time constants origin from the transfer function G within the *Laplace-domain* of the step response. It visualizes the relation between the in- and outcoming values of the signals: [20]

$$G = \frac{K}{\prod_{i=1}^{n} (1 + T_i s)} \tag{3.14}$$

with K as the gain, T_i as the time constant and s as the Laplace variable. [21, lect. 1, sl. 19]

Due to the lack of knowledge of the time constants, T_m cannot be calculated exactly. However, every speed constraint is evaluated on a simulated step response. For the third speed constraint specific T_m were determined in which the areas from fig. 3.5 were approximately equal.

To include the third speed constraint in the loss function, the equality constraint must be formulated as a *euclidean norm*.

$$L_{Phy,3,3} = \| A_1 - A_2 \|_2 = \sqrt{A_1^2 - A_2^2} \tag{3.15}$$

The values used for the training of the physical constraints on a simulated step response are shown in the tables 6.1, 6.2 and **??** in the appendix.

3.4 Comparison of the NN with the PGNN

The NN and PGNN of the one-tank system are going to be compared with each other. The data will be manipulated in various ways to test the performances of the NN and the PGNN.

3.4.1 Base Case

First, the system is trained with inputs $u \in [0.2, 0.4]$ and validated on inputs of the same range. Therefore the training and validation data set are generated in the range of $[0.2, 0.4]$ using an amplitude modulated pseudo-random binary stream (PRBS) signal.

3.4.2 Extrapolation of Validation Data

3.4.2.1 Extended Extrapolation

This manipulation of the validation data contains the extrapolation in an extended range $[0.1, 0.9]$. The range of the training data set ($[0.2, 0.4]$) is

included and already known by the system, because it was trained in it.
The extrapolated validation signal is also generated by an amplitude modulated PRBS signal.

3.4.2.2 Extrapolation on Foreign Data

The second possibility to extrapolate validation data is to extrapolate it onto a foreign area. The three observed foreign areas are going to be $u \in [0.5, 0.7], [0.7, 0.9]$ and $[0.5, 0.9]$. In this way, it is going to be observed on how well the PGNN will simulate on foreign data with different range sizes. It also allows the comparison between data which is more far away to the one the NN has trained with.

3.4.3 Reduction of Training Data

Another way to manipulate data is to reduce the training data down to 10 %. The NN simulated well with reduction down to 20 %, which is why only the reduction down to 10 % is analyzed. The test is made to look if the PGNN can still simulate well on a very much reduced amount of training data.

3.4.4 Gaussian White Noise

Lastly *Gaussian White Noise* (GWN) is added to the input training data. The signal-to-noise ratio used in this thesis is 20 %.

Chapter 4

Results

In the following, the graphs of the validation data over time are compared and presented with those of the simulated data of the individual neural networks for each case. The values of the MSE for all cases can be find in the Appendix.

4.1 One Tank System

4.1.1 Base Case

In figure 4.1 the NN is shown without physical constraints and no manipulation of the data sets. The simulation and validation data lie almost exactly on top of each other.

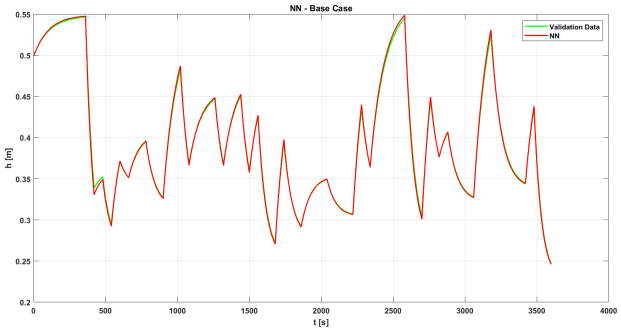

Figure 4.1: Validation and simulation data of the one-tank system. The input signal of the training and validation data is in the range of $[0.2, 0.4]$

Figure 4.2 shows the different plots of the base case condition which means no manipulation of data computed with different combinations of physical

constraints. The simulated values fit the validation values well, but the plot out of the computation with the speed constraint 3.3 (3.3.3.3) makes an exception.

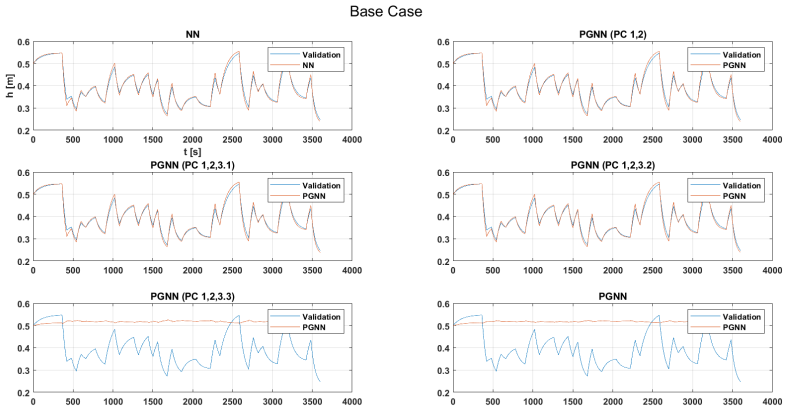

Figure 4.2: Validation and simulation data over time for the PGNN's with no manipulation of the validation or training data.

4.1.2 Extrapolation of Validation Data

4.1.2.1 Extended Extrapolation

In figure 4.3 the neural network is shown without physical constraints, but already with the extrapolation of the the validation data up to $[0.1, 0.9]$. In order to enable a verification of the individual constraints, they are tested separately, as it is shown in figure 4.5.

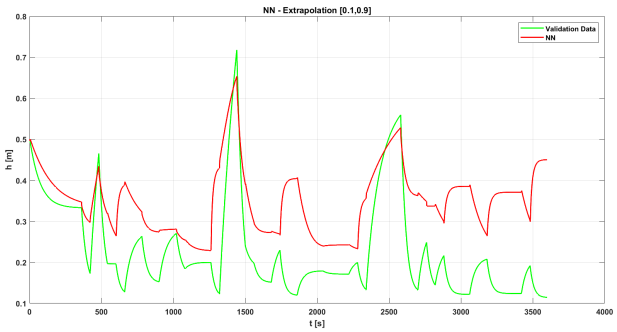

Figure 4.3: Validation and simulation data over time of the NN of the one-tank system with an extrapolation of the validation set ($[0.1, 0.9]$).

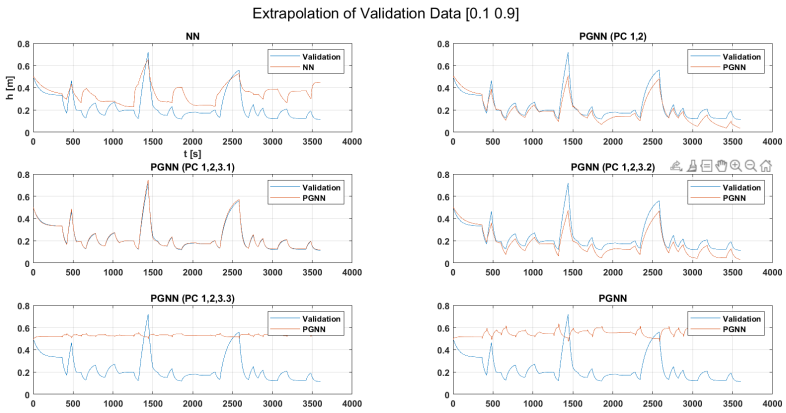

Figure 4.4: Validation and simulation data over time for the PGNN's with the extrapolation of the validation data ($[0.1, 0.9]$).

The PGNN with an extrapolation of the validation data to an enlarged region $[0.1, 0.9]$ can be improved by a maximum with the constraints 1,2 and 3.1 which were explained in section 3.3. However, it does not improve with the third speed constraint. Combining it results in a deterioration of the PGNN's performance.
The improved PGNN is as follows:

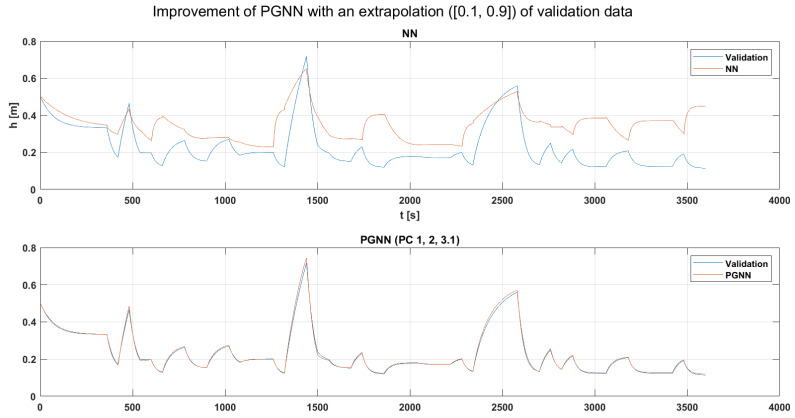

Figure 4.5: Comparison of the validation and simulation data over time for the NN and PGNN with an extrapolation of the validation data ($[0.1, 0.9]$). The constraints with the best results are (PC 1, 2, 3.2)

The MSE of the system is decreasing by adding the physical constraints to it except for the third speedconstraint.

4.1.2.2 Extrapolation in External Area

Extrapolating the validation to foreign data in the range of $[0.5, 0.7]$ showed an improvement of the first two and the second speed constraints. The constraints however did not show much effect on the NN where the extrapolated data was in the range of $[0.5, 0.9]$ and $[0.7, 0.9]$. The comparison of the plots can be seen as follows (4.6, 4.7, 4.7):

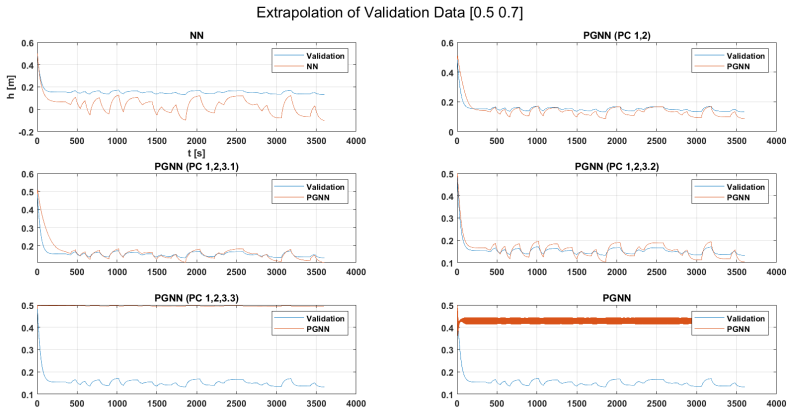

Figure 4.6: Validation and simulation data over time for the PGNN's with the extrapolation of the validation data ($[0.5, 0.7]$).

Considering the constraints that lead to an maximum improvement, the generalization of the NN which validation data was extrapolated to the range of $[0.5, 0.7]$ can be enhanced as follows:

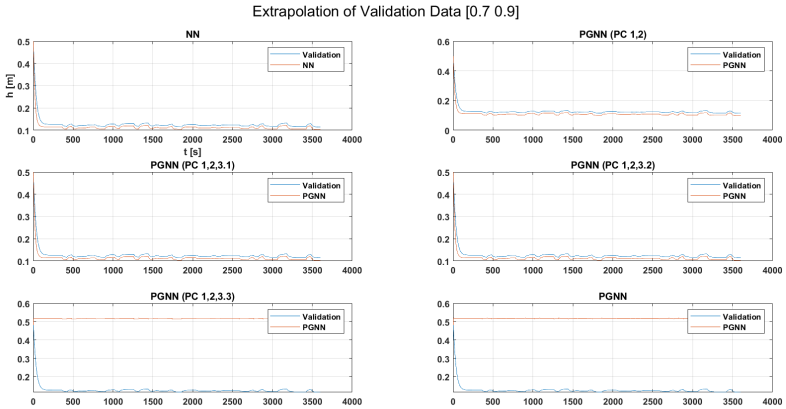

Figure 4.7: Validation and simulation data over time for the PGNN's with the extrapolation of the validation data ($[0.7, 0.9]$).

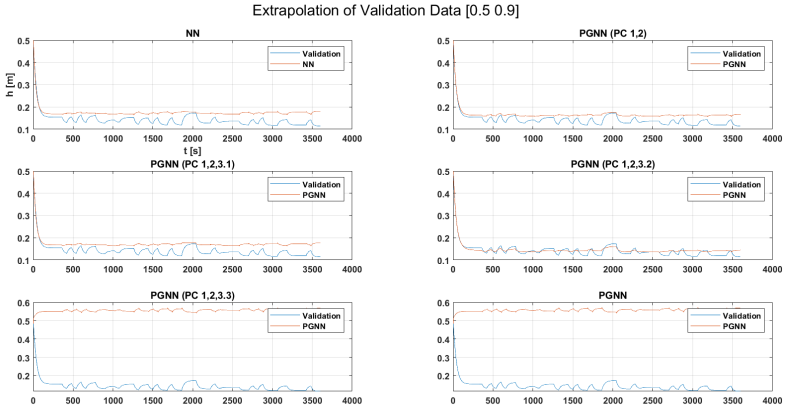

Figure 4.8: Validation and simulation data over time for the PGNN's with the extrapolation of the validation data ($[0.5, 0.9]$).

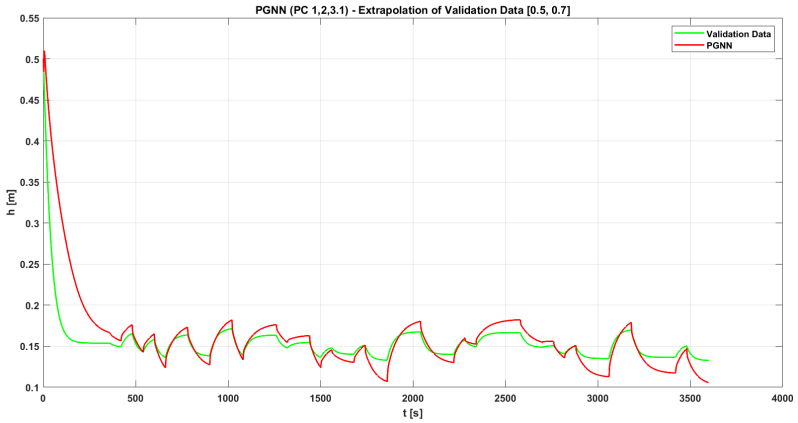

Figure 4.9: Validation and simulation data over time for the PGNN's with the extrapolation of the validation data ($[0.5, 0.7]$). The constraints with a maximum effort are considered (PC 1, 2, 3.2)

4.1.3 Reduction of Training Data

Training with reduced training data initially showed not much improvement, but the first speed constraint paired with the first two physical constraints do lead to an improved PGNN. The second physical constraint paired with the first two constraints does not improve much, but still shows a better validation set.

Figure 4.10: Validation and simulation data over time for the PGNN's with the reduction of the training data down to 10 %.

4.1.4 Gaussian White Noise

The addition to the training data of the PGNN with the GWN does not lead to an improvement. On the contrary the physical constraints *(3.2)* and *(3.3)* worsen it by adding them up to the NN.

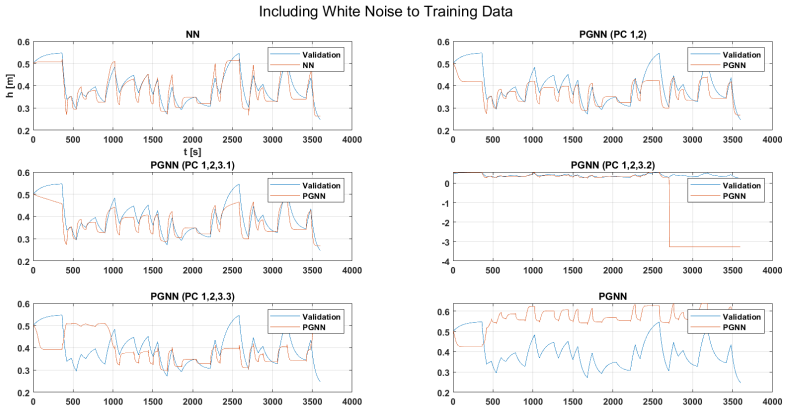

Figure 4.11: Validation and simulation data over time for the PGNN's with the addition GWN to the training data.

Chapter 5

Conclusion and Outlook

In this thesis, various NARX models with and without physical constraints have been developed and evaluated. Different formulations of physical constraints have been developed and tested on a case study.

The height and cause-effect constraint were extending the neural network usefully in all cases and did help generalising the NN. The speed constraints 3.1 and 3.2 improved the generalisation of the PGNN in most cases. The third speed constraint 3.3 however has worsened the generalisation.
Even when the constraint was correct by verifying it on a step response of a linear system, the physical constraint of it was not helpful for the generalisation. The reason could be the sensitivity towards T_m which needs to be manually determined for this speed constraint. Even small variations can have big influence on the resulting areas. In addition, the areas A_1 and A_2 from section 3.3.3.3 are approximated by numerical integration which may lead to an error. Furthermore, the sum of time constants rule is only valid for linear systems. As the one-tank system is nonlinear this may lead to errors.
Nonetheless it is shown that the extension of the loss function of NN's by physical constraints can improve their generalisation capabilities. This can be seen from the fact that the simulated data almost lie on top of the validation data or approaches it. It also shows lower MSE values as without physical constraints in general.

Since the search for the hyper parameters in this work were determined manually via a grid search, this can be improved, for example, with the deter-

mination by a *Bayesian optimizer*.

In this thesis, the focus was limited on three constraints and a single-input single-output system. Further constraints that focus on different aspects of system dynamics such as oscillations may be investigated further. In addition, more complex multiple-input single output or multiple-input multiple-output systems may be investigated in the future.

Chapter 6

Appendix

Table 6.1: Specific parameters T_{min} and T_{max} for the first speed constraint for the one-tank system on changing valve signals u.

Signal u	0.3	0.6	0.9
T_{min} $[s]$	250	250	250
T_{max} $[s]$	2300	2100	2100

Table 6.2: Specific parameters T_{min} for second speed constraint for the one-tank system with changing valve signals u.

Signal u	0.3	0.6	0.9
T_{min} $[s]$	190	90	70

Table 6.3: Values of the MSE for the NN in comparison to each other.

PC[a]	Base Case	$[0.1, 0.9]$[b]	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT (10%)[c]	GWN[d]
MSE	$3.07 \cdot 10^{-10}$	$5.77 \cdot 10^{-6}$	$1.45 \cdot 10^{-6}$	$4.58 \cdot 10^{-6}$	$4.77 \cdot 10^{-6}$	$4.85 \cdot 10^{-8}$	$6.53 \cdot 10^{-7}$

[a] Abbreviation: PC = Physical Constraint
[b] Ranges mean the extrapolation of validation data.
[c] Abbreviation: RT = Reduced Training Data
[d] Abbreviation: WN = *Gaussian White Noise*

Table 6.4: Values of the MSE for the PGNN (1,2).

	Base Case	$[0.1, 0.9]^a$	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT $(10\%)^b$	GWN^c
MSE	$3.64 \cdot 10^{-11}$	$2.94 \cdot 10^{-5}$	$1.85 \cdot 10^{-7}$	$4.97 \cdot 10^{-7}$	$3.36 \cdot 10^{-6}$	$1.5 \cdot 10^{-5}$	$6.83 \cdot 10^{-6}$

[a]Ranges mean the extrapolation of validation data.
[b]Abbreviation: RT = Reduced Training Data
[c]Abbreviation: WN = *Gaussian White Noise*

Table 6.5: Values of the MSE for the PGNN (1,2, 3.1).

	Base Case	$[0.1, 0.9]^a$	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT $(10\%)^b$	GWN^c
MSE	$3.65 \cdot 10^{-11}$	$1.82 \cdot 10^{-6}$	$3.43 \cdot 10^{-7}$	$5.81 \cdot 10^{-7}$	$3.85 \cdot 10^{-6}$	$3.32 \cdot 10^{-5}$	$8.94 \cdot 10^{-6}$

[a]Ranges mean the extrapolation of validation data.
[b]Abbreviation: RT = Reduced Training Data
[c]Abbreviation: WN = *Gaussian White Noise*

Table 6.6: Values of the MSE for the PGNN (1,2, 3.2).

	Base Case	$[0.1, 0.9]^a$	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT $(10\%)^b$	GWN^c
MSE	$3.65 \cdot 10^{-11}$	$1.79 \cdot 10^{-6}$	$1.32 \cdot 10^{-7}$	$3.58 \cdot 10^{-7}$	$4.79 \cdot 10^{-7}$	$1.13 \cdot 10^{-5}$	$6.82 \cdot 10^{-6}$

[a]Ranges mean the extrapolation of validation data.
[b]Abbreviation: RT = Reduced Training Data
[c]Abbreviation: WN = *Gaussian White Noise*

Table 6.7: Values of the MSE for the PGNN (1,2, 3.3).

	Base Case	$[0.1, 0.9]^a$	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT $(10\%)^b$	GWN^c
MSE	$4.46 \cdot 10^{-8}$	$4.05 \cdot 10^{-4}$	0.002	0.0039	$8.1 \cdot 10^{-5}$	0.0055	$9.03 \cdot 10^{-6}$

[a]Ranges mean the extrapolation of validation data.
[b]Abbreviation: RT = Reduced Training Data
[c]Abbreviation: WN = *Gaussian White Noise*

Table 6.8: Values of the MSE for the PGNN.

	Base Case	$[0.1, 0.9]^a$	$[0.5, 0.7]$	$[0.7, 0.9]$	$[0.5, 0.9]$	RT $(10\%)^b$	GWN^c
MSE	$4.9 \cdot 10^{-8}$	$1.13 \cdot 10^{-4}$	0.007	0.003	$1.06 \cdot 10^{-4}$	0.011	$7.35 \cdot 10^{-6}$

[a] Ranges mean the extrapolation of validation data.
[b] Abbreviation: RT = Reduced Training Data
[c] Abbreviation: WN = *Gaussian White Noise*

Bibliography

[1] Sergio Lucia, Alexandru Tătulea-Codrean, Christian Schoppmeyer, and Sebastian Engell. An environment for the efficient testing and implementation of robust nmpc. In *2014 IEEE Conference on Control Applications (CCA)*, pages 1843–1848, 2014.

[2] Hind Taud and Jean-François Mas. Multilayer perceptron (mlp). 2018.

[3] Laura Von Rueden, Sebastian Mayer, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed machine learning–towards a taxonomy of explicit integration of knowledge into machine learning. *Learning*, 18:19–20, 2019.

[4] Samuel J. Raymond and David B. Camarillo. Applying physics-based loss functions to neural networks for improved generalizability in mechanics problems, 2021.

[5] Arka Daw, Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling, 2017.

[6] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[7] Laurene V Fausett. *Fundamentals of neural networks: architectures, algorithms and applications*. Pearson Education India, 2006.

[8] Kevin Gurney. *An Introduction to Neural Networks*. Taylor and Francis, Inc., USA, 1997.

[9] Chih-Hung Chang. Deep and shallow architecture of multilayer neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 26(10):2477–2486, 2015.

[10] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 1 edition, 2007.

[11] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. pages 1–8, 07 2015.

[12] Maarten R Dobbelaere, Pieter P Plehiers, Ruben Van de Vijver, Christian V Stevens, and Kevin M Van Geem. Machine learning in chemical engineering: strengths, weaknesses, opportunities, and threats. *Engineering*, 7(9):1201–1211, 2021.

[13] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. Frankfurt am Main.

[14] JorgeJ. Moré. The levenberg-marquardt algorithm: Implementation and theory. In G.A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin Heidelberg, 1978.

[15] Sheng Chen, Stephen A. Billings, and Peter M. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51:1191–1214, 1990.

[16] K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[17] I. J. LEONTARITIS and S. A. BILLINGS. Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328, 1985.

[18] Hong-Te Su and Thomas J. McAvoy. Identification of chemical processes using recurrent networks. In *1991 American Control Conference*, pages 2314–2319, 1991.

[19] Hong Te Su, Thomas J. McAvoy, and Paul Werbos. Long-term predictions of chemical processes using recurrent neural networks: a parallel training approach. *Industrial & Engineering Chemistry Research*, 31(5):1338–1352, 1992.

[20] Bernd Girod, Rudolf Rabenstein, and Alexander Stenger. *Einführung in die Systemtheorie - Signale und Systeme in der Elektrotechnik und Informationstechnik (3. Aufl.).* Teubner, 2005.

[21] Sebastian Engell. Data-based dynamic modeling. Lecture Notes, Technical University Dortmund, 2022.

Eidesstattliche Versicherung

(Affidavit)

Strebel, Ümmühan Magdalena
Name, Vorname
(surname, first name)

193393
Matrikelnummer
(student ID number)

☒ Bachelorarbeit
(Bachelor's thesis)

☐ Masterarbeit
(Master's thesis)

Titel
(Title)

Improving the Generalisation of NARX Models by the Integration of Physical Constraints

Ich versichere hiermit an Eides statt, dass ich die vorliegende Abschlussarbeit mit dem oben genannten Titel selbstständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate kenntlich gemacht. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

I declare in lieu of oath that I have completed the present thesis with the above-mentioned title independently and without any unauthorized assistance. I have not used any other sources or aids than the ones listed and have documented quotations and paraphrases as such. The thesis in its current or similar version has not been submitted to an auditing institution before.

Dortmund, 04.07.2022
Ort, Datum
(place, date)

Ümm. Strebel
Unterschrift
(signature)

Belehrung:
Wer vorsätzlich gegen eine die Täuschung über Prüfungsleistungen betreffende Regelung einer Hochschulprüfungsordnung verstößt, handelt ordnungswidrig. Die Ordnungswidrigkeit kann mit einer Geldbuße von bis zu 50.000,00 € geahndet werden. Zuständige Verwaltungsbehörde für die Verfolgung und Ahndung von Ordnungswidrigkeiten ist der Kanzler/die Kanzlerin der Technischen Universität Dortmund. Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling zudem exmatrikuliert werden. (§ 63 Abs. 5 Hochschulgesetz - HG -).

Die Abgabe einer falschen Versicherung an Eides statt wird mit Freiheitsstrafe bis zu 3 Jahren oder mit Geldstrafe bestraft.

Die Technische Universität Dortmund wird ggf. elektronische Vergleichswerkzeuge (wie z.B. die Software „turnitin") zur Überprüfung von Ordnungswidrigkeiten in Prüfungsverfahren nutzen.

Die oben stehende Belehrung habe ich zur Kenntnis genommen:

Official notification:
Any person who intentionally breaches any regulation of university examination regulations relating to deception in examination performance is acting improperly. This offense can be punished with a fine of up to EUR 50,000.00. The competent administrative authority for the pursuit and prosecution of offenses of this type is the Chancellor of TU Dortmund University. In the case of multiple or other serious attempts at deception, the examinee can also be unenrolled, Section 63 (5) North Rhine-Westphalia Higher Education Act (*Hochschulgesetz, HG*).

The submission of a false affidavit will be punished with a prison sentence of up to three years or a fine.

As may be necessary, TU Dortmund University will make use of electronic plagiarism-prevention tools (e.g. the "turnitin" service) in order to monitor violations during the examination procedures.

I have taken note of the above official notification:*

Dortmund, 04.07.2022
Ort, Datum
(place, date)

Ümm. Strebel
Unterschrift
(signature)

***Please be aware that solely the German version of the affidavit ("Eidesstattliche Versicherung") for the Bachelor's/ Master's thesis is the official and legally binding version.**