UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Identification of Dynamical System's Parameters using Neural Networks

Author:
Dmytro KUSHNIR

Supervisor:
Dr. Lyubomyr DEMKIV

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2019

Declaration of Authorship

I, Dmytro KUSHNIR, declare that this thesis titled, "Identification of Dynamical System's Parameters using Neural Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"... all models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind...."

George E.P. Box, Statistician

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Identification of Dynamical System's Parameters using Neural Networks

by Dmytro KUSHNIR

Abstract

Parameter identification of Dynamical systems examined in the context of its value for the DNN training process facilitation and mitigation of the data imbalance. In the result of the theoretical analysis of the stated problem's methodological roots, were proposed the approach of augmenting usual NN model with part responsible for explicit representation of required parameters. Moreover, this approach involves the change of a learning procedure towards indirect supervision setting, where the NN responsible for system modeling, in form of time series prediction, does not observes the dataset at all but is being taught via the intermediate step of identification of target system parameter knowing its physical interpretation. Work considered the dynamical process on the example of DC motor described by the system of the ordinary nonlinear differential equations.

Acknowledgements

I am gracefull for all who supported me in the course of my academical efforts, from my loving family who gave me the chance to follow this path, to my teachers and colleagues who guided me. Also to my scientific advisor, who introduced me to the field of robotics and showed the whole new world, the world of challenging problems, but fruitful applications.

Contents

List of Figures

List of Tables

List of Abbreviations

ROS	**R**obot **O**perating **S**ystem
ML	**M**achine **L**earning
(A)NN	(**A**rtificial) **N**eural **N**etworks
RNN	**R**ecurrent **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
EMF	**E**lectro **M**agnetic **F**orce
WB	**W**hite **B**ox model
BB	**B**lack **B**ox model
GB	**G**ray **B**ox model
NLDS	**N**on**L**inear **D**ynamic **S**ystem
GD	**G**radien **D**escent
MLE	**M**ost **L**ikelyhood **E**stimator
SGD	**S**tochastic **G**radient **D**escent
ODE	**O**rdinnary **D**ifferential **E**quations

List of Symbols

$\underline{\parallel}$	'independant'	
a	distance	m
P	power	W
ω	angular frequency	rad
v	linear velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
R	electrical resistance	Ω
V	electrical voltage	V
i	electrical current	A
H	inductance of the conductor	H

Dedicated to my patient loved ones . . .

Chapter 1

Introduction

People are eager to change the environment, make things, and express their creativity in their deeds. Innovations in tools and machines that augment people creative power, giving them new abilities for altering things had been the milestones of history. From the simplest instruments up to the great machines of the late industrial period, all of those required artisan master or experienced human-operator for targeting and moving each machine component and processed detail.

But with the progress of technology and the rise of the complexity of such systems, direct control became a limitation, and more systems got controller as the middle part between the operator influence and occurring process. Studies in a field of related problems gave rise to the Control theory, separate field of mathematics, which is especially actively developing since the 50th of the XX century.

With the modern state of technology and progress towards digitalization not only industrial processes but even casual life, controllers had become much more sophisticated and now have their place in lots of everyday things.

But despite computing technology progress, the fundamental challenges of building controllers remain, as they depend on knowledge of controlled system state. When there is a real-world mechanical system, it is essentially impossible to grasp all its properties with computer simulation. Problems of stability of control, reliability of the model, lay a significant limitation on present controllers.

To eliminate issues concerning system state observations, separate algorithms called observers are used. Their task is to estimate directly unavailable or inaccurate parts of the system's state. An observer can identify essential system parameters to fill the gaps in knowledge about the system and its current state in the environment.

But the problem of observer synthesis is not trivial at all and requires extensive physical modeling and involves sophisticated mathematical apparatus.

Recent progress in a field of machine learning gives rise to adaptations of those new methods to the control theory problems. Shift from the computer systems programming, requiring strict, rigorous abstractions of system components to the teaching of such a system can help a lot with control problems. But as the biggest problem of programming is not in computational abilities but in the expressiveness of abstractions provided by the human, the same we have with 'teaching' approach, where now the edge is shifted towards the data and statement of the purpose of the learning process.

Attempt to address that problem is the motive behind this work, here we will review an approach to increase the expressiveness of the system modeling task statement using the identification of underlying system parameters, try to incorporate them to the usual nowadays methods and estimate the resulting benefits on an example of modeling the nonlinear dynamical system.

Chapter 2

Background information

2.1 Systems modeling

When natural sciences deal with a real-world object to describe it, each scientific field uses abstractions relating to their subject matter field. Describing those objects through the lens of natural laws which governs them allows operating with their adapted representations. Such representations are usually grasped in some symbolic form within mathematical expressions. That kind of expressions and equations are translating observed real-world phenomenons to a strictly founded sequence of computations reproducing the described processes but in abstract form. Those forms do not reproduce all properties, but only those are interesting for a particular case. As those representations are modeling the part of the object, they are called models.

Following the famous statistician's aphorism *"All models are wrong, but some of them are useful"* (Box, 1976), it worth underlining that practical models of real-world systems cannot grasp all its aspects. In mathematical model included only a small relevant portion of attributes from a real system. Models are an accurate representation of real systems only to some extent of uncertainties and only in ranges of real-world data it was validated against.

The most general field of knowledge spreading towards common characteristics of such objects' representations regardless of its disciplinary origin is system theory. The systems theory is focused on the connections, dependencies, and emerging characteristics of an object's components and evolution of that object as a system concerning time and interactions between its parts or with the external environment.

When the model of some system considered it is usually represented in terms of their interactive properties as some rules which under external influences (inputs) exhibits specific behavior (outputs). Such abstraction allows for describing a system in the form of mathematical functions which provides a mapping from input to output values.

Within the model real-world object properties represented as vectors of variables, where the range of each variable's values represents possible states of corresponding modeled parameter and dimensionality of a vector, - the number of modeled properties. Equivalently, such variables vector can be treated as coordinates of a point in n-dimensional space, where n - is the dimensionality of a vector.

Due to referred imperfectness of representation and inherent measurement errors and noisiness of real-world data, the best model we can hope for is the approximation which provides a representation of the targeted system with some inherent small stable random error.

Using error for model validation: To state that some model has the best fit to the modeled system, we can explicitly assume the modeled process includes some inherent noise part $v(t)$. Afterward, we can construct the function measuring the

error between the data provided by a real system and data its model produces as following (2.1):

$$\begin{aligned} &F(u) \to y + v(t), \\ &\hat{F}(u) \to \hat{y} \\ &e(y, \hat{y}) = v(t) \Rightarrow \hat{F}(u) = \hat{F}^*(u) \end{aligned} \tag{2.1}$$

Where:

u = input signal
y = output signal
$\hat{y}$ = estimation of signal y
$v(t)$ = noise
e = error function
$F(u)$ = real system
$\hat{F}(u)$ = our system estimation
$\hat{F}^*(u)$ = correct model estimation

With such an approach, we can verify that our model-candidate is a best-capable of predicting real system reaction onto inputs. Another problem is how to identify an optimal model among infinitely many possible? That requires additional consideration of the search procedure.

2.2 Systems properties

To reason about systems' properties we are interested in those work, we need to introduce symbolic abstractions for them. Denote system as F and time of experiment as t. Interaction with the system will be described in forms of Inputs and outputs at the moment will be described as x_t and y_t.

For the needs of this works, we have to consider a system abstractly, as we frequently have to address both real-world objects and their mathematical models simultaneously. Therefore let's consider a System like the previously defined Model of a System, as an entity that perceives influence (as input) and outputs some information. In such terms, most of the considerations can be performed with all forms of the considered object or its abstractions interchangeably and don't require particular clarifications.

2.2.1 Systems classification

In the context of interactions causality of the results, all systems can be divided into stochastic and deterministic. Stochastic systems under the same input x_i produce a random value from some probability distribution Y_i. Deterministic systems, in turn, produce only one possible value. Since the deterministic output can be represented as a degenerate case of a random variable from a distribution with probability of a single value, equal 1, further we will implicitly consider them both if other is not specified.

We can denote system as mapping from the domain of input values to its outputs as $F(x_t) \rightarrow y_t$.

For static systems equal inputs always produce the same distribution of outputs Y_t (2.2).

$$if\ x_{t_i} = x_{t_j}, t_i \neq t_j\ \forall i, j \Rightarrow Y_{t_i} = Y_{t_j} \tag{2.2}$$

The concept of a *'time'* does not play a significant role in the description of a static system. A trial performed at t_i for the static system will be predetermined by x_i only ignoring all the history of interactions with the system from time t_0 to the t_i. We can say that the static system does not *'remember'* the previous trials.

As an example of such a system in the real world, we can take a series of coin tossing experiments. Here the outcome of the experiment will result in sequence $P(heads|tails)$ sampled from some probability distribution. From our experience, we know that the coin does not *'remember'* what was the results of previous experiments, and all the trials are independent of the outcome of the others.

The notions of time and the system's memory gain meaning in the course of consideration of dynamical systems. Here each input results not only in the change of output but altering the system itself. Model of such a system has to have some own variable attributes, as well as input and output objects. Those attributes are called system state, and their possible values are system state space.

The output of the dynamical system in a moment of time t is determined by both state and input (2.3).

$$F(x_t, s_t) \rightarrow Y_t\ or\ F(x_t)|s_t \rightarrow Y_t \tag{2.3}$$

System's change in response to influences is reflected in those states changes. The law, governing those changes, is called the system evolution rule (2.4).

$$Ev(s|t_{current}) \longrightarrow s|t_{next} \tag{2.4}$$

Evolution rule also can be either stochastic or deterministic, but for our research, we will have a focus on the systems with deterministic evolution.

As evolution rule governs a system to transition $t_{current} \rightarrow t_{next}$ on each moment and "time" here considered as an abstract parameter which "moves" only in one direction and determine the order of events.

Knowing all the sequence of transitional conditions, deterministic system's state can be traced from the episode at t_0 via those transitions through all the intermediate states to every t_i. This property is called the causality property (axiom) of the dynamical systems (Hinrichsen and Pritchard, 2005). It formalizes that in each moment of time system is affected by interactions that happened in the past, but not in the future. From this same principle arises requirement that description of a dynamical system requires an explicit statement of the state at the beginning moment of time $s|t_0$.

Property of causality means that the model of a dynamic system requires only the definition of the state-transition rule and initial state to be able to describe all its time-states. Therefore such evolution rule in a mathematical model of a real-world system for continuous processes usually is represented by the differential equations.

The equation (2.4) rewrites as (2.5):

$$\frac{dx}{dt} = Ev(x, u) \tag{2.5}$$

or as difference equation for discrete-time models (2.6):

$$x_{t+1} = A(x_t, u_t) \tag{2.6}$$

where A is a transition matrix

2.2.2 Linearity and Nonlinearity

Certain systems' properties are arising from the characteristics of their mathematical representation. One of the most prominent of them is the property of system linearity. Those are systems properties of which can be modeled by the linear equations, having all the terms introduced in the first degree.

Defining properties for the linear systems are additivity and homogeneity:

Additivity: If system produces output y_a under input u_a, and y_b under u_b, it has to produce $y_a + y_b$ under the input $(u_a + u_b)$ (2.7):

$$F(u_a + u_b) = y_a + y_b \tag{2.7}$$

Additivity determines the property of **superposition** of linear systems, meaning that several linear systems can be represented as one system representing the combined effect of them. In case of superposition of inputs it looks like (2.8):

$$F(u_a) + F(u_b) = y_a + y_b = F(u_a + u_b) \tag{2.8}$$

Homogeneity: If we scale input by k, the output will also be scaled by k:

$$F(ku) = ky \tag{2.9}$$

Linear models can describe only a limited subset of phenomenons due to limitations on its mathematical form. The linearity is considered a nice property for modeling of the real systems, as characteristics following from it eases the computing task and allows better solutions. But this property not only desirable but also seldom. It is hard to find a completely linear real-world system (Boeing, 2016). A much more frequent case is modeled nonlinear system that can be linearized (represented and solved as linear) under some assumptions or conditions for the particular case (Krener and Respondek, 1985).

Most of the nontrivial systems observed in real-world are both nonlinear and dynamical (Boeing, 2016). As nonlinear phenomenons are more common in nature, it is an active research field not only amongst natural science disciplines and engineering but also in nontechnical studies as psychology (Guastello, 2013), economics (Puu, 2003), finances (Chen, 2008), medicine (Hoshi et al., 2013) and urban planning(Ostwald, 2013). Therefore research of flexible methods for modeling of NLDS bears a prominent value for the wide specter of disciplines.

2.2.3 DC Motor as an example of a nonlinear system

Looking inside such ubiquitous for modern days system as the DC motor, used in fields varying from electric toys to the power plant turbine, we can observe an entangled system of co-occurring physical processes. Here we see the fusion of mechanical and electrical systems, where electrical current sent to the motor winding is being transformed to the mechanic rotation of the shaft.

Under applied voltage to the motor's coil, it starts to feel the moving force from the magnetic field it bears in. When the voltage becomes large enough to overcome friction and inertia of rest, coil starts its rotations, thereby driving the motor shaft. With the increase of the rotational velocity rotation inertia of moving parts grows, it became harder to stop it now. Moreover, velocity gain continues increasing for a while. But with the increase of the coil rotation speed, other forces are gaining effect: as DC motor can be used as a generator producing electricity when experiencing shaft rotation, those principles are inherent to this system. Therefore with rotation in a magnetic field, inductive coils suppose to have electrical current emerging inside and opposing to its rotation, but as applied voltage has the opposite direction, this system starts to suffer from emerging reverse electromagnetic force (EMF). It slows the velocity gain and consequently balances the moving force, and rotor speed stabilizes while the voltage remains stable.

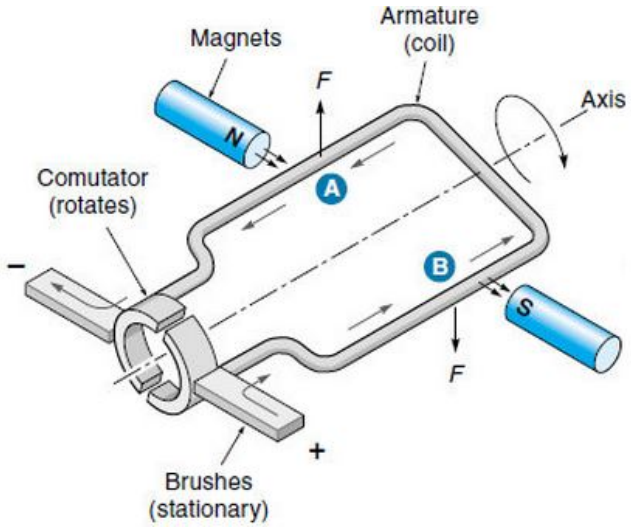

FIGURE 2.1: Principal scheme of DC motor physical concepts involved. Picture from (*DCMotorScheme*)

Besides those mentioned processes DC motor system is affected by other nonlinear processes, as magnetic loss in the form of hysteresis (shift in a phase of the output to the input) and eddy currents emerging in the rotor core. And with the increase of the system's scale number of processes worth consideration also increases, making the task of modeling even for such well-known system a nontrivial problem.

At the same time, with a glance from the outside on such a motor, we observe only phenomenons: we see how the rotation of the shaft increases in the response of supplied electrical voltage growing. We can notice that while the change of voltage can be performed almost instantly, the shaft rotation speeding up or slowing down requires some time.

Also, it is notable without any measurements that electrical can be arbitrary, from 0 to arbitrarily large values, but any particular motor will always have operating mode nonlinearities. There are at least several of them: while under too low voltage

shaft is not rotating at all, there is 'dead-zone' nonlinearity, as well as when and above some threshold voltage rotation speed slows its rotation speed gain, and we observe the 'saturation' nonlinearity effect.

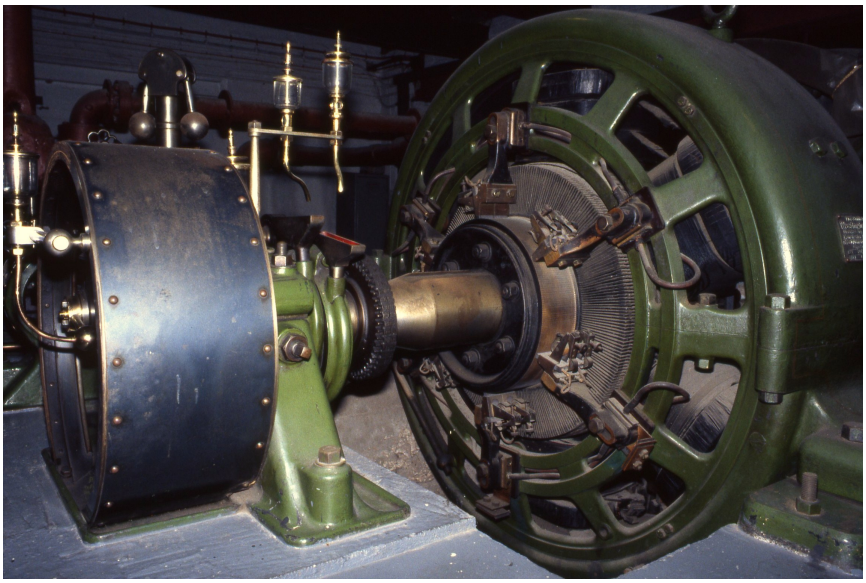

FIGURE 2.2: Westingson DC generator (1907), Museum of Science Industry, Birmingham. Image from wiki (*Westingson DC generator*)

But such observations give us no insight about involved physical processes; knowledge from such observation are bare values, therefore obtained knowledge is quantitative. But for such system modeling, we still can collect diverse enough set of experimental data for all available regimes of functioning and build a detailed map of states and transition without discovering any detail of governing rules for this system.

Moreover, in a combination with a known theoretical model of this system describing underlying physical processes, data collected from system behavior observation can improve modeling accuracy. With its help, we can perform model verification procedure, and therefore decide what are exact coefficients should be plugged in the equations of a model to make it describe this particular physical system optimally.

We think that such an approach can help in the creation of both precise and interpretable models. Property of NN to be the universal approximator can be a good foundation for creations of a model, flexible enough to fit all possible nonlinearities. But while computational complexity of NN-based modeling could be overcome via adaptation of architecture to a specific problem and growing availability of computational resources, the lack of such model interpretability and robustness to uncertainties are the inherent flaw for such approach to the modeling. This limits such approaches from application in risk-related tasks and fields requiring active decision-making where exposure of bases for such decisions is necessary.

This research is aimed exactly on the methods of introduction expert knowledge about the modeled system, that should help to solve those flaws.

2.3 Control theory

Real-world physical processes could be represented by the systems of differential equations, being built according to physical laws governing the process. Resulting dynamical system of equations, if precise enough, being solved concerning time gives us the states of the referral system as the solution. Using such a model, we can predict a system's behavior under certain actions from outside. Those approaches, utilized in fields of engineering, gave rise to the **control theory**, the field of applied math whose aim is modeling the control influence and improving the designing principles for controller synthesis.

2.3.1 Control theory for dynamic systems

Unfortunately, solving such systems in the closed (analytic) form usually is very hard or even impossible. Therefore the only available tool is a numerical solution. Computational capabilities achieved required abilities only recently to model such systems with sufficient accuracy (Torokhti and Howlett, 2007).

Despite its reach mathematical apparatus, control theory not always can propose an approach for robust and reliable control. There are cases when the accuracy of simulation and precision of control influence has to be impossibly high; mostly, those cases are concerned with controlled systems instability (Minorsky., 2009).

Lots of NLDS are not stable; this means that introducing small influence can result in a significant change in the system state. Unstable systems illustration is a known concept of the butterfly effect when even minuscule change in initial conditions results in a drastic change in the trajectory of the model states (Lorenz, 1963). Facing that, reliable modeling in such cases with reliable predictions is an unsolved problem of high importance as such unstable systems are usual in meteorology, finances and, of course, engineering.

One of the best possible scenarios to handle a system which has unstable states is to prevent it from reaching those dangerous state-space regions (Rieger, Gertman, and McQueen, 2009). This lays more requirements on the controller, which has to possess accurate information about controlled system state description. For such a complicated task, as a rule, used closed-loop controllers, where control process includes some feedback about the state of the systems.

Preventing the system from instability is only a specific use case for closed-loop control, as in general case it has even wider application for providing reliable control.

Despite its benefits, such control methodology is limited by the availability of system state data. Providing advanced control influence requires even more additional terms included in state space.

For example, we can consider the case when we want to control the rate of system state change, in such case we need to augment the state space with the first derivative of the mentioned state $\frac{dx}{dt}$. Similarly, for the systems where the momentum of such parameter plays the role, the controller requires the second derivative in the system's observation: $\frac{dx}{dt^2}$

For such cases, it is customary to isolate the task of tracking state values and approximating their derivatives (Goodwin, Graebe, and Salgado, 2001). Accompanied by unobserved parameters estimation task, those tasks are separated to the separate actor of the closed-loop control system, called **observer**.

Following the notation for dynamical systems (2.5) we can denote linear DS with closed-loop controller as (2.10) and then it's observer in an explicit linear matrix form as the Luenberger observer (CICCARELLA, MORA, and GERMANI, 1993) (2.11):

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \tag{2.10}$$

Where:

x – state vector

y – output vector

u – input vector

A, B, C, D – –corresponding transformation matrices

$$\begin{aligned} \frac{d\hat{x}}{dt} &= A\hat{x} + Bu + L(y - \hat{y}) \\ \hat{y} &= C\hat{x} + Du \end{aligned} \tag{2.11}$$

Where:

L – Observer gain matrix chosen such that $(y - \hat{y}) \to 0$ asymptotically

$\hat{x}$ – estimate of system's state x by the Observer

$\frac{d\hat{x}}{dt}$ – estimate of system evolution rule $\frac{dx}{dt}$ by the Observer

$\hat{y}$ – estimate of system output y by the Observer

But for the cases of nonlinear systems we cannot present any explicit matrix form. In order to build such observer we have either to linearise such a system if it is possible (Krener and Respondek, 1985) or use more sophisticated Observer synthesis methods applicable only for certain class of systems, as high gain observers (Bernat and Stepien, 2015), sliding mode observers (Drakunov, 1992) or other.

Therefore our goal of estimation parameters for NLDS is covering this problem of NLDS observers creation by using NN, as the system modelling tool agnostic to underlying system structure.

2.4 Models classification based on information source attitude

For classification of modeling approaches, we will use prior information criteria as it was done in the (Hauth, 2008). Every model created is based on some priors which can be divided into three primary types: either prior knowledge about the real system, underlying structure or physical phenomenon used as K, or information of systems behavior obtained from collected exploitation data D, or assumptions on properties and adequacy of our proposed model A. As the author states in (Hauth, 2008),

assumptions are an inherent part of any model and can be tested on adequacy, while the choice of using knowledge and data as grounds for modeling is the fundamental difference between modeling approaches.

There are opposite methodologies: data-based, called Black Box models, and knowledge-based, called White-box models.

2.4.1 Black and White box concepts

Concepts of black-box and white-box models: Naming of the models or concept as '-boxes' in the scientific field started at the 50th of 20 century. Mostly discourse was devoted with 'black-box,' concept, which meant 'some process or system with unobservable parts' and counterparts to its concept of white-box, or 'glass-box', where the underlying process is evident.

As an abstract term definition in cybernetics field notion of "black box" were introduced by Ross Ashby in 1956 (Ashby, 1956). Description of this concept later was provided in 1961 by Norbert Wiener as an unknown system which has to be identified by system identification special methods (Wiener, 1965). Those ideas were expanded by other scientist and philosophers in 60th, epistemologist Mario Bunge detailed and strengthened the formal foundation of this concept by producing the general Black Box theory (Bunge, 1963).

Black box models are relying on Data from an experiment, which secures them from false assumptions provided by human-in-the-loop of model design . The main principle they follow is popular nowadays when computing power is widely accessible, and data amounts became Big, is: 'let the data speak for themselves.' But, the relative simplicity and universality of those methods are challenged by their drawbacks:

Lack of results interpretability: a black box model aim is to imitate the process being led by its imprint in data. The goal of the Black box model is to provide expected output, not to express how this output was produced. As there exist infinitely many mappings $X \to Y$ when $|X|$ and $|Y|$ are finite . Then we can expect more than one possible process to have the same imprints. Therefore we can't say with confidence that resulting black box model represents the aimed system, but only its data-represented characteristics.

Data-intensive

Dependence on provided data: model cannot provide any insight from data if relevant information is not there. Also, any bias in provided data and possible flaws in the training process can threaten to achieve reliable results.

Those two properties result in vulnerability to unseen data and outliers in provided inputs.

But nowadays we observe that this methodology is on the top of its popularity, as an illustration, can be considered a hype on neural network modeling and arise of deep learning, we will return to their characteristics later and show why they are vivid examples of black-box modeling methodology.

On the other side of the modeling methodologies spectrum lies models, where the model foundation is made by expressing knowledge about governing physical laws and system properties in the form of the equations. Those models usually expressed in terms of the differential equations (or the time-difference equations for discrete-time processes) coupled with a prescribed set of rules for specific cases. Those approach to modeling covers some flaws mentioned for Black boxes, but impose their own shortcomings:

Expertise demand: Beside of knowledge about physical processes requirement, which is already a substantial limitation for applying this methodology, the building of a reliable model also requires some experience with complex computations.

Discrete calculations: Overall computing machines are inherently operating with discrete values, as for continuous processes we have to operate with Real values, possessing an infinite number of digits, we are limited to operate only with finite digits-number approximations. It means that all continuous processes models are dealing with their discrete approximations. Especially sensitive cases for this problem are unstable systems, where roundoff error can cause a significant change in modeling result and when there is a goal of long term modeling, where the accumulation of error results in a deviation from correct states-trajectory.

Overspecification of models: Models produced in such a method can achieve excellent results in a specific case when lots of factors specific for a given system in given conditions. But it means that the change of problem sets will require a restart of the modeling process from scratch. Moreover, with an increase of components number model's equations become cumbersome in manipulation and computing.

Idealistic modelings: If instead of model specification we will follow the generalization strategy, the resulting nice looking model will rather model perfect machine then real-world aggregate with its wear, noisy signal, and flaws.

Upon those categories became widely used in fields of philosophy, social science, and what most essential for us to, cybernetics and control theory, this discourse had produced derived categories, like 'grey-box' concept.

2.4.2 Gray box

Both approaches White-box and Black-box are both extreme abstractions of the opposite extremes; they possess descriptiveness for theoretical reasoning, but not for the real-world applications. Most of the created models are encompassing to some extent both Data and Knowledge in some form: either it would be data processing based on insights from their real-world nature, or tuning the parameters of equations in white-box models based on experimental data.

Altogether a combination of two approaches can help to negotiate the flaws of either.

Those models from *'in between'* part of the specter are collectively referred to as **Gray box (GB)** models. As they give optimal practical use, practical application of such methodology in industrial systems came into wide practice up to the development of special design procedures and guidelines (BOHLIN, 1994). To describe them briefly we can cite short principle, listed by Bohlin himself which is: *"don't estimate what you already know but test your knowledge against experimental data."*

Often modeled system posses some obvious properties and including this knowledge to Black box model can give it a boost in performance, but that how to do this becomes a challenge by its own. How to incorporate the best of two approaches is one of this research aims.

Advantages from use of GB approaches depend on each particular situation, but several most obvious of them can be listed for general cases:

- allows incorporating prior knowledge
- allows explicit formulations of nonlinearities and dynamical properties
- data imbalance impact diminishes
- additional variables not available from data alone could be estimated

- both statistical modeling tools and subject field apparatus can be applied
- results could be interpreted by experts from a field
- suits combined teams of both, knowledge field specialists and modeling experts
- the model requires fewer parameters

2.4.3 Classification of Gray-box models

From what is stated up to this point Gray Box methods is a vague notion for a group of modeling methods that are neither purely White box nor Black box. Clarifying this notion is accompanied with a problem that gray-box methodologies does not possess any firm distinction as white-box or black-box models, but just denotes the grouping attribute of diverse approaches. To designate this concept further, we will follow a descriptive approach by considering its constituting groups of modeling methods. Following by classification proposed by (Sohlberg and Jacobsen, 2008) GB models can be split into five classes:

1. **Constrained Black Box identification**: approach where a-priory knowledge is incorporated in the form of additional restrictions laid upon the original Black Box model methodology.
2. **Semi-physical modeling:** when system input and output data can be additionally transformed with a known nonlinear transformation occurring in the underlying process.
3. **Mechanistic modeling:** where the original white-box model is expanded by hypothesis testing process and information about model prediction errors.
4. **Hybrid modeling:** when the model is explicitly divided into black-box and white box parts.
5. **Distributed parameters systems:** approaches aimed to solve the challenge of distinguishing the errors produced by model reduction and errors from provided data and assumed system properties.

2.5 Optimization

A search of the best model parameters can be formulated in terms of a mathematical optimization problem. For setting the notation and definitions for research we will outline its basic notions as we use them:

Parameters: If the model uses parameters (values that characterizes the modeled process) then optimization process can help to select the optimal set of parameters for the model to fit the given data. Define that all parameters are described in one vector term as θ. Then model that provides outputs using parameters will be defined as $F(u, \theta) = y$. Alternatively when the same set of parameters is used for several different inputs we can use notation for *parameterized function* $F_\theta(u_1) = y_1$

Likelihood: Likelihood is the joint probability distribution of one parameter, conditioned on the other. Likelihood function provides a probability of one one parameter distribution, given distribution of another. In our case, we will use it like $\mathcal{L}(\theta|y)$, for evaluating the probability of parameters given our observations from

the model. This function defines probabilities for possible values of the searched parameter and drives all procedure of parameters estimation, as according to likelihood principle all the relevant information for inference of parameters is contained in likelihood function (Edwards, 1984).

Estimator: Is the function of data, that estimates the searched parameter θ. Estimator is the function which maps distribution of observed data to certain parameter value – the estimand (estimate) (2.12) (Kosorok, 2008). For the use of notation, we can denote estimator $\hat{\theta}(X)$ and its estimand value θ can both be denoted the same as $\hat{\theta}$.

$$\hat{\theta}(X) \to \hat{\theta} \tag{2.12}$$

Where:

X – given data distribution

$\hat{\theta}(X)$ – estimator

$\hat{\theta}$ – estimate (estimand)

Estimate is sampled from vector space of parameter values Θ. Difference between the estimate and actual values of estimated parameter is error of estimation (2.13):

$$e(X) = \hat{\theta}(X) - \theta \tag{2.13}$$

If estimated parameter θ maximize the likelihood of observing data, X such Estimator is Maximum Likelihood Estimator (MLE) (2.14) (Aldrich, 1997).

$$|L(\hat{\theta}_{MLE}|X) - L(\theta)| \to 0 \tag{2.14}$$

Loss function: Since the estimation error defined as the difference of real and estimated values, it belongs to the same vector space $\bar{\theta}$. For a model of a system, it is the space of modeled values, which means that the meaning of '*distances*' in this such vector space is vague as it depends on introduced modeling abstractions. To clarify the meaning of a distance and define which prediction is the best in the sense of the lowest error has to be stated distance function, which in optimization task context is called loss function.

The loss function is denoted as (2.15) and the lower value of the loss function means the better fit of parameters estimate to the formulated problem with given data. Specifically, for each problem, it can be augmented with additional restrictions laid upon estimated parameters or other task-specific limitations. Loss function can include terms facilitating optimization process while targeting the optimization method, not optimization goal, called *auxiliary loss functions* (Du et al., 2018).

$$L(X, \hat{X}) \to \mathbb{R} \tag{2.15}$$

If loss function has at least one local minimum, we can find the optimal local estimator for our estimated parameters. If this local minimum value is the lowest

function value for all parameters space it is global optimum estimator $\hat{\theta}^*$ (2.16). Our goal in this works is to find such estimator for parameters values.

$$L(\hat{\theta}^*(X), \theta) \leq L(\hat{\theta}(X), \theta), \forall \hat{\theta} \tag{2.16}$$

We will come back to the question of loss functions design during the description of our modeling process with a neural network, as it is also the cornerstone of the NN training process.

Summarizing: Our task of parameter idenfication can be formulated in the terms of optimization problem for parameter θ value as the search of such estimator $\hat{\theta}(X)$ which estimate minimizes loss function defined for function of θ and data X.

After being formulated, the optimization problem can be solved with different approaches. We will concentrate our attention in this work on gradient-based methods of parameter identification.

Gradient descent: Continuum of the loss function values constitutes some surface of loss function values, and aim of the learning process can be considered as to find the minimum of the *'height'* on this surface. To steadily progress toward this value the iterative gradient-based approach to optimization is used. On each iteration, the gradient *(derivative)* of a loss function calculated in the circumference of the current parameter estimation $\hat{(\theta_i)}$ defining the direction of the 'downhill' step towards new, improved estimate:

$$\hat{\theta}_{i+1} = \hat{\theta}_i + \gamma \nabla_\theta L(F(x, \hat{\theta}_i), y) \tag{2.17}$$

Where γ is a small value of a *'step'* being made.

This process will eventually converge to some local minimum, but convergence to global optimum is not guaranteed and strongly depend on starting parameter values initialization, the shape of the loss function 'landscape' and allows the use of various techniques for optimization problem facilitation (Bottou, 2012).

$$L_\theta(F(\hat{\theta}(X), y) \rightarrow min \tag{2.18}$$

2.6 Neural networks

2.6.1 Concept and properties

Artificial Neural Networks can be defined as a framework for different machine learning algorithms. For shorthand, we will refer to them simply as Neural Networks (NN). They have widely inspired by neural science, therefore naming of concepts was also brought from that field (Marblestone, Wayne, and Kording, 2016). NN consists of individual nodes, called neurons, and connection between them called edges. They have floating point values called weights, which are the learnable parameters for the NN and are adjusted in course of the NN 'Training'. Those weights are involved in computations with processed data and determine the resulting NN properties. Weights increase or decrease the power of a signal (called activation) going through.

For better abstraction of model structure groups of neurons are combined in layers. NN consisting of a large number of neurons are called Deep Neural Networks (DNN) (LeCun, Bengio, and Hinton, 2015).

On the picture (2.3) you can see a simple example of Feed Forward NN with fully connected layers where processing happens in a straight manner from the input layer via a hidden layer towards the output layer.

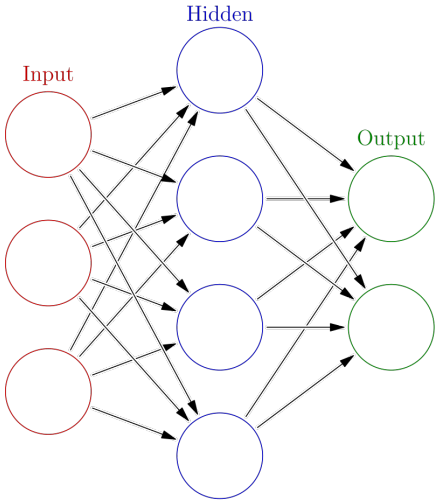

FIGURE 2.3: Abstract representation of NN. Image from (*Conceptual scheme NN*)

NN provides only a wireframe or a canvas for building a model to be filled by defining the loss function – expressing the goal of modeling and embodied via training procedure. Training is stated in terms of the optimization task (2.5) goal of training has to be expressed as the loss function of NN output values and desired values $L(\hat{y}, y)$, where y stands for the desired values and $\hat{y}$ for NN-predicted values (**??**), where $F(\bullet)$ is function representing NN and x are input values.

To achieve this goal training algorithm should find the optimal set of weights θ^* for NN $F(x, \hat{\theta}) \rightarrow \hat{y}$ which will minimize the loss function given the distribution of training data:

$$L(F(x, \hat{\theta}^*), y) < L(F(x, \hat{\theta}), y),\ \forall \hat{\theta},\ \hat{\theta}^* != \hat{\theta} \tag{2.19}$$

A training of NN is also the form of *loss function minimization*. Usually it is being done with gradient-based optimization (2.5). As NN training often is a data-hungry procedure, compute gradient on each iteration using all the available training data at once is impractical. Therefore is commonly used *stochastic gradient descent*-based (Bottou, 1998) (SGD) optimization method, where only a small part of data is used on each iteration to make the optimization step. As each used part of data are not absolutely representative, the computed gradient can deviate from the real one, but with smaller optimization steps being made but in larger quantity, SGD-based algorithm eventually converge to a minimum of the loss function (Bottou, 2012).

The loss function is determined in terms of predictions $\hat{y}$, not by weights of NN nodes connections themselves. Therefore training goal is to translate and transfer the error from NN output vector to weights responsible for those estimate. This transition $e(\hat{\theta}) \rightarrow e(W_i)$ is performed with the error Backpropagation (Werbos, 1975) algorithm based on the derivative chain rule. Use of this algorithm made the large NN training computationally feasible, as it benefits largely of calculating each individual derivative in only once. After it was shown by Rumelhart, Hinton and Williams (Rumelhart, Hinton, and Williams, 1986) that it is an assured method for

deriving the useful representation of data in hidden layers of a NN the backpropagation procedure became the driver of modern ANN advancements (LeCun et al., 2012).

Data splitting. Neural networks as universal approximator can fit any provided data in case of sufficient NN size. This issue shapes a training procedure because inherent noises present in data could be learned as well as useful information. To avoid this overfitting problem data available for model training are split onto disjoint subsets (Ripley, 2007):

- The **training** set: The biggest part of data, which is used for the training itself, and two smaller parts, validation set, and test set.
- **Validation set.** The validation set is a part of the data which is hidden from the training algorithm but used to evaluate the model performance. Error calculated on the validation set is not propagated through the network but only used to tune hyperparameters and detect whether the model capable of generalizing its assumptions on the validation set as well. The lowest value of **Error value** indicates a training moment when the desired optimum is achieved, and further training should be stopped.
- **Test set** As we specially tune some hyperparameters concerning validation set performance, it can result in another case of overfitting by hyperparameters selection for the validation set. To secure from such a bias can be introduced dedicated test set, absolutely excluded from the training procedure and used for final evaluation only.

2.6.2 RNN

Recurrent neural networks (RNN) are a special class of NN architectures which have a recursive connection made to transport activation from the previous input to the next one. Such RNN structure can be unfolded to the oriented graph of hidden state transition that helps to grasp dependencies in sequential data, as it is shown on [picture] In case of dealing with a time-organized data this property allows to capture dynamic properties from observed data (Miljanovic, 2012).

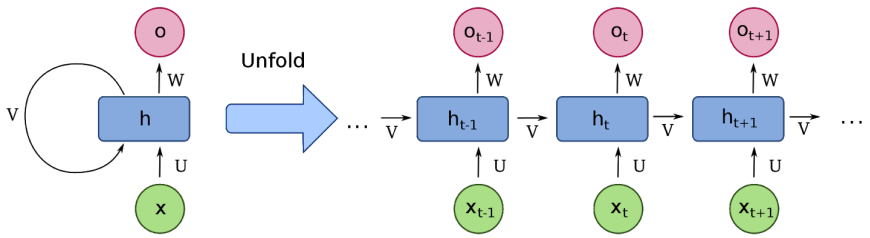

FIGURE 2.4: RNN can be *"unfolded"* into a sequential representation. Image from (*Unfolded representation of RNN*)

Those properties hidden activation in RNN resembles the process of dynamical systems evolution with the inner state transition from one timestep to another. Because of those properties, RNN gains popularity in time series studies (Che et al., 2018), dynamical system modeling and processing of other linearly structured

data as natural language processing (Plank, Søgaard, and Goldberg, 2016) or images frames in video streams .

It is hard to define RNN properties more precisely, as it is the name for whole class of NN. As an example we will consider Elman and Jordan (Pham and Karaboga, 1999) networks, they have simple idea lying in their foundation. Both of them store values from previous iteration in dedicated parameter neurons called *'context neurons'*.

$$\begin{aligned} h_t &= \sigma_h(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= \sigma_y(W_{yh}h_t + W_{hy}y_{t-1} + b_y \end{aligned} \tag{2.20}$$

Where:

$$\begin{aligned} t &= \text{moment at time} \\ x_t &= \text{inputs vector} \\ h_t &= \text{hidden layer vector} \\ y_t &= \text{output vector} \\ W_{q_1q_2} &= \text{parameters matrix} \\ q_1, q_2 &= \text{matrix transformation spaces, } q_2 \xrightarrow{W} q_1 \\ b &= \text{bias parameters vector} \\ \sigma_h, \sigma_y &= \text{nonlinear activation functions} \end{aligned}$$

Equations for Jordan NN are barely the same as for Elman NN (2.20) with difference in just one term of first the first equation: h_{t-1} is replaced by y_{t-1} (2.21). This shows that context neurons in Elman NN are storing values from previous timestep internal state, but in Jordan NN they store the previous step output value.

$$h_t = \sigma_h(W_h x_t + W_h y_{t-1} + b_h) \tag{2.21}$$

Methods for training RNN are based on the same backpropagation algorithm that for general NN, but the sequential character of data determines the special technique called *backpropagation through the time* which inherently works like the usual backpropagation algorithm, but requires a representation of RNN in the unfolded form as shown at picture (2.4) . Such representation treats a whole range of T inputs and T outputs as like they are available simultaneously (Werbos, 1988).

The requirement of Backpropagation through the time application revealed the limitation of the classical error backpropagation algorithm, called *the vanishing gradient problem*. As a number of considered timesteps rise, the chain of derivatives becomes longer eventually, resulting in a numerical value of gradient drops to zero. Popular activation functions like *tanh* have values range $[0, 1]$ and value the product of such values decreases exponentially with the number of cofactors grows. This issue makes it harder to track dependencies between remote inputs (Bengio, Simard, and Frasconi, 1994).

2.6.3 LSTM

The Long Short Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997) were originally designed to overcome the vanishing gradient issue. They include separate *track* for passing the memorized cell state between the cells and separate nodes with nonlinear activations responsible for managing those cell state by learning when to '*remember*' or '*forget*' long-term states.

This cell state passing track is depicted on the (2.5) as a straight horizontal line on the top of the cell. From the equation (2.22) it can be seen that state c_t *does not pass via any activation function*, but only multiplied with forget gate f_t values 'filtering' out all unnecessary states and being add with the value from update gate o_t. Therefore if nonlinear activation representing forget function is chosen such that we can expect to have values very close to 1 then state c_{t-1} is passed through the cell without significant changes.

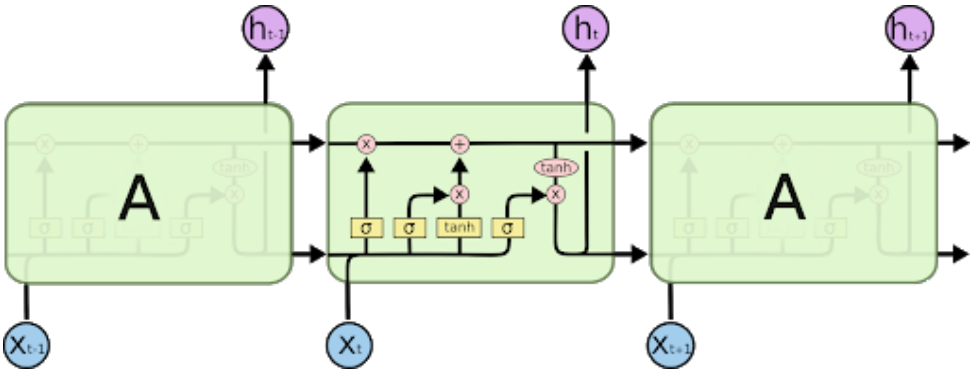

FIGURE 2.5: Internal structure of single a LSTM node. Image from (*Understanding LSTM Networks*)

LSTM is the model class name itself, as it has several variations in architectures. We follow the classical approach and using LSTM with forget gates (Gers, Schmidhuber, and Cummins, 2000). It is represented in terms of the following equations (2.22):

$$
\begin{aligned}
&f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
&i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
&g = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
&c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \\
&h_t = o_t \circ tanh(c_t) \\
&h_0 = 0, c_0 = 0
\end{aligned}
\tag{2.22}
$$

Where:

$\circ$ stands for elementwise multiplication

$t -$ moment of time

$\sigma -$ nonlinear function applied

$h_t -$ hidden state

$c_t -$ cell state

$x_t -$ input

$i_t -$ input gate

$f_t -$ forget gate

$g_t -$ cell memory gate

$o_t -$ output gate

$W_{q_1 q_2} -$ parameters matrix, mapping $q_2 \xrightarrow{W} q_1$

$f_t, i_t, o_t, h_t, c_t, b_{q_1 q_2} \in \mathbf{R}^{dim(h)}$

$x_t \in \mathbf{R}^{dim(x)}$

As we see, there are way more parameters and, therefore, memorizing capabilities, by this reason and broad applicability to learning on sequential data we had selected it as the component of choice for incorporating into our architecture.

Chapter 3

Related works

In a discourse of exploitation, the dynamical system interaction with controller works as following: signal emitted by the controller makes its influence onto the evolution of the system state. To maintain the desired behavior of the system controller has to operate in a 'feedback loop' relations with the systems state, to when the result of provided control influence is exhibited back to the controller via sensors observations or deducted by other means.

Regarding the governed system state observation, controllers can be created with the 'White-box' modeling approach, where the system is assumed perfectly observable with all its intermediate steps and transitions occurring in the process of system transitions from one state to another, in this case, a controller operates with detailed state description provided in 'disclosed' form. On the opposite part of the specter lies a 'Black-box' modeling: when the internal system state is presumed unobservable and controller relies on state estimation based on external with respect to the system state data and no interpretable system state description produced as a byproduct of control procedures. Lastly, the 'Gray-box' concept lies between those concepts and refers to the systems which are observed only partially or with the limited degree of confidence.

As the system we are dealing with has generally known structure we can use this knowledge to introduce into the creation of the governing entities. Hence it is possible to create something more transparent then pure black-box. But those governing equations for our system are parametrized by with some time-evolving parameters of a kind $\mu(t)$ as well as nonlinearities arising from the process flow. Therefore the pure, predictable white-box model is impossible for our case.

Therefore best we can expect of our model is to lie somewhere on the specter between 'white-box' and 'black-box' models, belonging to the family of 'gray-box' systems.

3.1 Grey box models concept

The idea of creating a Grey-box model observer of Nonlinear Dynamic System is not new, as example there are several works written by Cen (Cen, Wei, and Jiang, 2013), (Cen, Wei, and Jiang, 2011) on this topic.

As mentions in his paper Cen (Cen, Wei, and Jiang, 2011), the task of modeling is NDS incorporates two major challenges: modeling of nonlinearities and dynamic simultaneously. As nonlinearity reflects the static and dynamics reflects dynamic behavior. This problem is feasible to track if observed object or system is well-known; in such case, we can parametrize the mathematical model describing the system and produce a so-called 'white-box' model. But actual mechanical system observed in automation and robotics problems is not entirely deterministic, despite underlying processes are known.

On the other hand, such systems can be modeled with a "black-box" model approach, with any of discriminative ML methods , neural networks are belonging to as well. Those approach does not require the introduction of any knowledge about the modeled system in advance but can be purely data-driven instead. But the complexity of task lies in the idea of supervised learning itself: training sample is a static snapshot of the object behavior. Therefore only those dynamical properties can be learned which are represented in the training dataset.

Because those limitations Cen proposed Gray-Box Neural Network Model (GBNNM), an approach taking advantages of white-box and black-box approaches. According to (Mandic, 2001), the Gray-box model is the one obtained from black-box with introduction of some a priory knowledge, whether it is some PDF, statistics of observed data, attractor geometry, etc. Sjorberg (Sjöberg et al., 1995) describes two classes of gray-box models: physical models where physical process is used for designing the state-space, as semi-physical, where knowledge about process is used for defining some non-linear combination of data structures later estimated with black-box methodology.

While analyzing two abovementioned works regarding the gray-box modeling, Cen proposes a question-based plan for systematization such projects description:

- What kind of NN are used in our gray-box model and why?
- NN is used to identify nonlinearity, dynamics of the process or both?
- Which parts of the studied object are known and which is unknown?
- What is approximation ability of used NN? To what, extent is it possible to model the process?
- What is the training procedure for the mentioned GBNN model?
- What is the performance of the obtained GBNN?

As those questions are providing a and clear structure, we will follow them in the corresponding part of our research during the resulting model description.

3.2 Dynamical System Observation

As mentioned Vantsevich (V. Vantsevich, 2018), controller for the automatized system requires as many parameters as possible with best possible precision, but there lies major drawback concerning the price and complexity of required sensor system to obtain them. Instead of deployment of real sensors can be used observer – a mathematical algorithm that estimates unknown parameters based on the subset of known parameters of the system and observed parameters from sensors. It can be found in control systems as part of a feedback loop. However, the author empathizes that there are lots of serious drawbacks and limitations in this approach: additional computational resources, correct configuration parameters, problems with handling noisy data, etc.

Awareness of those warnings will be used in the course of our model synthesis and testing, as we are striving exactly to one of the stated aims of Vansevich's research highlighted by the author: estimating unmeasured system parameters in the exploitation discourse. But while the author uses Extended Kalman Filter (Kalman, 1960) for extracting detailed structured information, this method has problems with

an increasing number of estimated parameters as it implies including even more variables to the system state vector, increasing dimensionality of the problem implying demands on both measurements collection and computing parts of the problem. Also, Kalman Filter relies on the linear solving method, which implies the linearization of the observed system. At the same time, NNs are better suited to dealing with multidimensional optimisation as well as estimating nonlinearities, and as those are inherent components of the stated problem, we will use the NN-based approximation method.

3.3 Decomposition methodology

For dynamical system identification, there were lots of approaches produced and trialed. As Puscasu (Puscas et al., 2009) stated. Currently, there are two major distinct approaches:

1. Concentrated on computing parameters of a neural network to approximate linear or nonlinear mapping between the inputs and outputs of the modeled system, as researches on networks with memory neurons (Sastry, Santharam, and Unnikrishnan, 1994), Elman neural networks and Jordan neural networks (Pham and Karaboga, 1999).

2. Based on the neural network (Adeli, Asce, and Jiang, 2006) structural organization which itself represents structural dependencies of system components (Sastry, Santharam, and Unnikrishnan, 1994). These structures contain themselves system state representation in discrete time steps, where on each step k which is mapped onto the k+1 step state using some nonlinear mapping. This mapping can be approximated using NN, as the universal approximator.

   The general idea of NN representation is to approximate system of governing equations (3.1) with neural networks (3.2).

$$\begin{cases} \mathrm{x(k)} = \mathrm{f}_{NL}(x(k-1), u(k-1)) \\ \mathrm{y(k)} = \mathrm{g}_{NL}(x(k), u(k)) \end{cases} \tag{3.1}$$

   Where:
   $x(k)$ – state vector
   $y(k)$ – output vector
   $u(k)$ – control (input) vector

Those nonlinear mappings (3.1) can be approximated suing the neural networks in following manner (3.2):

$$\begin{cases} \mathrm{x(k)} = \mathrm{N}_f(x(k-1), u(k-1)) \\ \mathrm{y(k)} = \mathrm{N}_g(x(k), u(k)) \end{cases} \tag{3.2}$$

where N_f and N_g stand for the neural network approximators for the nonlinear functions f_{NL} and g_{NL}

Mentioned Puscasu (Puscas et al., 2009) work itself incorporates those approaches, therefore creating another method of grey-box system representation using internal recurrent neural networks (IRNN). The author uses the representation of a studied system based on the structural decomposition onto parts which have its own isolated peculiarly engineered functionality exploiting some physical principles (as hydraulic, mechanical, electric-based systems, etc.). As any decently complex system of the car or robotic aggregate complexity level can be decomposed in numerous ways because of each structural subsystem has it own subsystems as well author uses approaches described by (Yue and Schlueter, 2005) (Sjövall and Abrahamsson, 2008) for appropriate system parts identification and isolation for the following modeling using separate NN with prescribed inputs and outputs.

We see the potential in Pucasu's approach, yet in our work on this stage of the research we will use more traditional architecture in order to evaluate the general impact of used approach on the modeling experience and decompose model only on parametrised and parameter-free components.

3.4 NN for the similar problem

Branch of research devoted to using NN with memory cells for time series analysis nowadays came the long way and predominantly converged to use one of two pretty similar approaches: first, concerned with use of recurrent neural networks (RNN) with of Long-short term memory cells (LSTM) (Hochreiter and Schmidhuber, 1997) or the second one, devoted to networks with Gate Recurrent Unit (GRU)(Cho et al., 2014). The application of those NN architectures is researched by Ogunmolu (Ogunmolu et al., 2016). As he mentions, ordinary multilayered neural networks are proven to suit well for identification and control of static and dynamic neural networks simple nonlinear systems (Narendra and Parthasarathy, 1992), (Narendra and Parthasarathy, 1990). But for the time-series (Graves, Mohamed, and Hinton, 2013) and in dynamic identification and control of nonlinear systems (Wang and Chen, 2006), (Dinh, Bhasin, and Dixon, 2011) RNN took its place, as their property of representing associative memory is crucial for problems with long-term dependencies in data.

Ogunmolu in his paper studied how different kinds of RNN (ordinary RNN, LSTM, Fast-LSTM, and GRU-RNN) perform on nonlinear system identification. He concluded that such self-organized Deep NN suit good for nonlinear parameter estimation. In our work, we will also use this class of neural network replacing with the low-level details of implementation the individual NN for the approximation of each individual subsystem model on which we decompose our general one. Used in that paper datasets from DaISy database, containing both SISO and SIMO datasets of nonlinear systems, will also be used for functional verification of our model's individual subsystems approximators and the whole NNs combination.

3.4.1 Goal Statement

Founding on a problem field highlighted at the Background Overview chapter (2) we decided to aim onto the stated in the title of this research topic of a parameter identification from the specific angle. Despite the overall importance of parameter identification for the NLDS control and operation, this task inevitably faces another issue: as identified parameters are used for parameterization of the system's model

equations, there is always standing requirement of building that system's model itself.

We are addressing issues of model building issues concerned with the applied methods itself. Inherent flaws of the Black box (2.4.1) modeling does not allow the resulting estimator to be robust and fully accountable. At the same time, the White box methods require extensive knowledge about target system constituting processes which complicate the task and conflicts with the requirement of model adaptation to the individual system.

Following the highlighted in Related Works experience (2.4.1) as well as our clauses mentioned in the course of those works analysis, we will state our general goal as the identification of dynamical systems parameters and using them to address the flaws of usual NN-based modeling.

To achieve those goals we use analysis of methods for addressing the similar problem, then formulate the proposal of our own method and implement it. The evaluation will be performed using the exemplary nonlinear dynamical system.

Chapter 4

Modeling experience

4.1 Object of modelling

4.1.1 DC motor description

For providing control to such a system, underlying parameters have to be identified for each particular case, as in the course of exploitation and under different conditions, their properties can change. Interactive Observers should provide this functionality. To make a step towards the synthesis of such Observer, we studied methods of estimation on modeled example of NLDS representing such a motor. For model building was used Simulink as a tool for visual system constructor provided with a set of prepared solvers for differential equations systems.

To reproduce a DC motor system behavior, we took the representation described at (Kara and Eker, 2004)

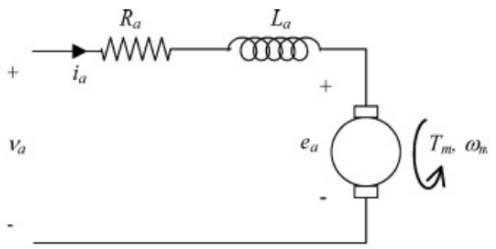

FIGURE 4.1: Principal scheme of DC motor. Picture from (Kara and Eker, 2004)

The system is described by the following ODE:

$$U_a = R_a i_m(t) + L_a \frac{di_m}{dt}(t) + e(t) \tag{4.1}$$

Where :
U_a (V) is the motor armature voltage
R_a (Ω) is an armature coil resistance
L_a (H) is an armature coil inductance
i_a (A) stands for armature current

And

$$e_a = K_b w_m \tag{4.2}$$

e_a (V) is back electromotive force
K_b (rad/Vs) is the EMF constant
w_m (rad/s) is a rotational speed of the motor

moreover

$$T_m = K_t i_a \tag{4.3}$$

T_m (Nm) is a Torque produced by the motor
K_t (Vs/rad) is motor's torque constant

As this basic representation has only the components of the first order, this is a simplified linear model of DC motor. It, like any real-world mechanical system characterized by numerous nonlinearities, an example could be mentioned in Background overview dead-zone nonlinearity example . For training procedure generalization, we introduce the nonlinear term representing voltage loss due to magnetic hysteresis. This part of DC motor model described in the context of similar problem consideration by (V. Vantsevich, 2018) were considered as nonlinearity example satisfying two crucial criterions: makes an influence in any part of state space and have big enough effect on the system to detect it in data.

$$u_{HS} = K_{HS} w_m(t) i_a(t) \tag{4.4}$$

Where introduced K_{HS} is a magnetic hysteresis constant.

Its nonlinearity may be not obvious as it consists of different terms w_m and i_a, in our model they relate as $w_m = i_a K_t$. In result we have obvious nonlinear term in equations rewritten as $u_{HS} = K_{HS} K_b i_a(t)^2$

In the DC motor system, U_a plays the role of control influence and $\frac{di_a}{dt}$ describes system evolution rule and dynamical properties arising.

$$\begin{cases} \frac{di_m}{dt}(t) = \frac{U_a - R_a i_m(t) - e(t) - u_{HS}}{La} \\ e_a = K_b w_m \\ T_m = K_t i_a \\ u_{HS} = K_{HS} K_b i_a^2(t) \end{cases}$$

4.1.2 Data generation

For a generation of sample data were used Simulink, the system for a visual building of mathematical models. Modeled system gains representation in visual blocks, as shown at picture (4.2). The Simulink model is composed of two primary components: **blocks** graphical abstractions for some underlying operation and **connections** (lines and arrows) representing the passage of data (signal) between them.

Diagram (4.2) shows the scheme of a build model we used for simulating the behavior of the NLDS representing DC motor.

In the left part of the diagram are grouped blocks responsible for generation input voltage for DC motor.

Generating block which uses random signal generators denoted as *sig_value* used for sampling value of a signal and *sig_duration* for the duration of which chosen signal value will be transmitted. Those blocks are sampling their output values from

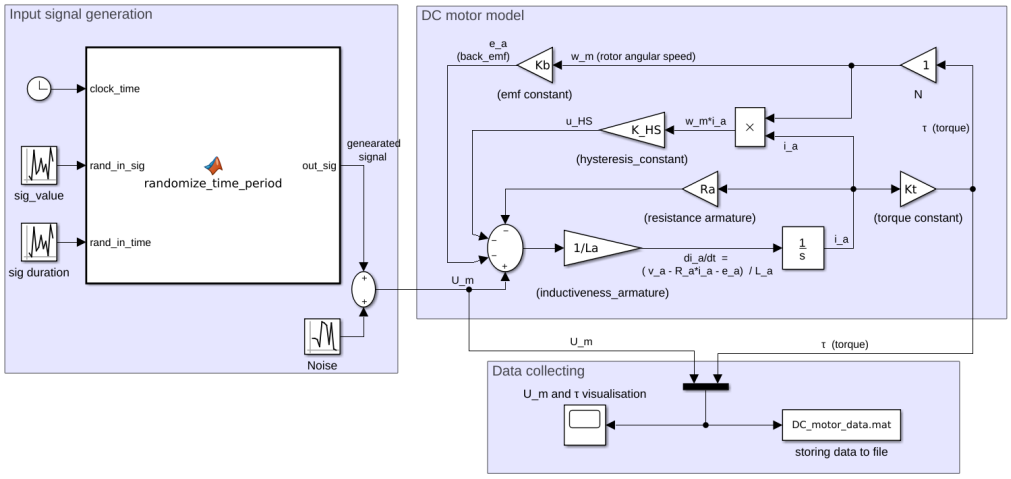

FIGURE 4.2: Organization of DC simulation in Simulink

uniform distribution bounded by p_{max} and p_{min} parameters for each block. Block denoted as *randomized_time_period* represents the programmatic implementation of combining those two signals into resulting random-valued, random-period signal.

Also, we introduced uncertainties if a form of additive noise and other distortions in the course of this research to evaluate the robustness of NN to such phenomenons, as an example on the picture is shown block denoted as 'Noise'.

The largest part of the scheme is devoted to DC motor model itself placed at the top right part. The core part of this scheme is the bottom signal line with integrator marked with $\frac{1}{s}$. This is the block which treats input as derivative value and outputs value of the corresponding integral in the current moment of time.

Upper three signal flows are representing processes describing computations for the members of differential equations according to their marks: Hysteresis, EMF, the resistance of the motor armature.

To simulate system the ***solver*** algorithm is being used. A solver is a method for the numerical solution of a given system. It is responsible for the transition of states in the model concerning time. We've chosen the *'ode3 (Bogacki-Shampine)'* solver (Shampine and Reichelt, 1997).

To achieve sufficient accuracy, we chosen solver timestep of 0.01 sec. The value was taken due to experimental experience, showed that larger values of solving step results in values transitions between solution steps being not smooth enough to represent state values with the precision required for parameters identification or even causes significant deviations from the expected system behavior.

On the other hand, decreasing timestep more is also impractical, as even with this step size, we had to select only each k-th time record, dropping all the others. On the stage of parameters identification, we treat that time-data as a discrete states sequence and selected timestep is a compromise between sequence long enough to grasp properties of underlying process and computing time spent.

To grasp the context of occurring dynamics, the RNN we are using in our architecture processes, the data split into small sequential periods. We picked step for sampling those periods such that the number of states seen by the model in each subsequent period would cover the expected length of the transitional phase met in data. In this way, we ensure that single data samples will be representative regarding the dynamical properties of a system.

At the bottom part of the diagram placed blocks for visualizing and storing values from signal lines to file.

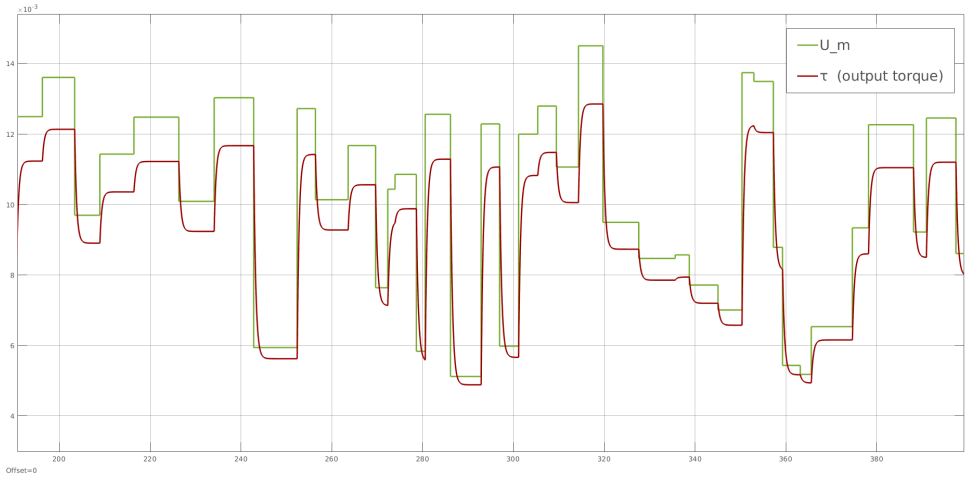

FIGURE 4.3: Example of generated data. Voltage in blue and resulting torque in green. Data re-scaled to fit the same plot.

Example of generated data used for evaluation of our model is shown at (4.3). Control voltage on this example is rescaled to match the scale of resulting output torque. On the plot, you can see the curvature of the output plot showing the process of system transition to the new stable state.

4.2 Optimization goal

Modeling of a system implies reproduction of its properties. For this reason, our NN-based model should represent the same data transformation that can be observed from the original system . In our case, it should predict values of system output y_t, knowing information from previous timesteps and control signal x_t. Such a problem statement literally defines the problem of TS prediction (Madsen, 2007).

Core goal stated as the topic of this work is the identification of parameters describing the underlying physical process. We state it as search for vector of variables θ such that function representing the model of our dynamical system parametrized by this vector will produce the observed mapping from inputs to outputs. $F(x_t, \theta) = y_t$

4.2.1 Parameter Identifiability prerequisites

Following the (Newey and McFadden, 1994), to ensure that formalization of the parameter identification problem could be formulated as an optimization task, we have to clarify several assumptions that should be satisfied:

We presume that θ_i **uniquely identifies the system** in sense that no two different parameter values result in the same output sequence (4.5).

$$\theta_i \neq \theta_j \Leftrightarrow F(\bullet, \theta_i) \neq F(\bullet, \theta_j) \tag{4.5}$$

For the successful identification of the parameters, *loss function* defined for system properties have to satisfy the following criterions:

- smoothness in the neighbourhood of θ^*:

$$\exists \nabla_\theta L(F(\theta^* + \epsilon, X), Y)\, \epsilon \to 0 \tag{4.6}$$

- have minimum in the neighbourhood of θ^*:

$$L(F(\theta^*, X), Y) < L(F(\theta^* + \epsilon, X), Y) \tag{4.7}$$

If requirements above (4.2.1) are satisfied then the solution of optimization problem will correspond to identifiable parameters of modeled dynamical system, as (Hengl et al., 2007)

Those assumptions being held result in that true system parameters θ from given data, given data X for the system $F_\theta(X)$ will infer, as certain data distribution maximizes likelihood (2.5) of only one possible θ_i (4.8).

$$\mathcal{L}(\theta_0|Y_0) > \mathcal{L}(\theta|Y_0),\ \theta_0 \neq \theta \tag{4.8}$$

Equivalently in terms of loss function it implies, that only one parameter value corresponds to the loss function global optimum given sample data (4.9).

$$L(F(\theta_0, X_0), Y_0) \leq L(F(\theta, X_0), Y_0),\ \theta_0 \neq \theta \tag{4.9}$$

Two separate goals standing before our model could be formalized in the form of optimization task as a search for the following estimators (4.10):

$$\begin{aligned} &\hat{F}(x, \theta) \to F(x, \theta) \\ &F(x, \hat{\theta}) \to F(x, \theta) \\ &Where: \\ &F - original\ function \\ &\hat{F} - its\ estimator \end{aligned} \tag{4.10}$$

Here we see the problem, that we do not have the original function available. To overcome this we could use its estimate instead, but then as we have it as the optimization goal in other function, our goal function system will reduce to single expression as $\hat{F}(x, \hat{\theta}) = F(x, \theta)$.

$$\begin{aligned} L(\hat{Y}, Y) &= L(\hat{F}(X, \hat{\theta}), F(X, \theta)) \\ &\to \hat{F} = \hat{F}(X, Y, \hat{\theta}),\ \hat{\theta} = \hat{\theta}(X, Y, \hat{F}) \end{aligned} \tag{4.11}$$

Problem is that such optimization task is ill-specified as it includes two different optimization criterions and used estimators have a recursive dependency in their definition. There are two cases which resolve this issue:

- Both optimization criterions are independent; then the problem can be split into two independent subproblems (4.12).

$$\begin{aligned} L(\hat{Y}, Y) &= L(\hat{F}(X, \hat{\theta}), F(X, \theta)) \\ \hat{\theta} \perp\!\!\!\perp \hat{F} \Rightarrow &= L_F(\hat{F}(X, \theta), F(X, \theta)) + L_\theta(F(X, \hat{\theta}), F(X, \theta)) \\ &\rightarrow \hat{F} = \hat{F}(X, Y),\ \hat{\theta} = \hat{\theta}(X, Y) \end{aligned} \tag{4.12}$$

- Optimization criterions are interdependent. Moreover, optimality of one implies the optimality of another. Then it should be possible to rewrite it in terms of combined criterion optimization (4.13).

$$\hat{F}^* \Leftrightarrow \hat{\theta}^* \Rightarrow \hat{F}(X, Y, \hat{\theta}) = \hat{F}_{\hat{\theta}}(X, Y) \tag{4.13}$$

According to our previous assumptions we consider functions which are uniquely determined by its parameters (4.2.1), it means that precondition of estimators independence is not held, as one criterion defines the other uniquely. At the same time, unique identifiability indicates that the parameters identify the function: $\theta \Rightarrow F_\theta$. Therefore, such a formulation of the optimization problem in some general case is correct and decidable, because if we can find the correct representation of θ, this implies that a real is necessarily derived from it, but the search for θ is an ordinary task optimization of one parameter. From here we conclude that there is such a consistent pair of estimates that are simultaneously optimal: $\exists(\hat{F}^*, \hat{\theta}^*)$:

Since such a problem is solvable in general case, as the next step, we should analyze the problem in the context of our precise task and consider if optimum of that problem will be the one we need. Not-unique value for $\hat{F}$ satisfies conditions of our task, while they have the required dynamical properties but the system parameter identification problem expects $!\exists\theta$.

It would be a trivial task in case if this problem has only one optimum $\exists!(\hat{\theta}^*, \hat{F}^*)$.

From uniqueness criterion (4.2.1) follows naturally demand of its exclusiveness (4.14):

If exists any $\tilde{F}$ that is parametrized by some $\tilde{\theta}$ is valid representation of our data our assumption of single optimum does not hold.

$$\begin{aligned} if\ \exists\tilde{F}, \exists\tilde{\theta},\ \tilde{F} \neq \hat{F}, \tilde{\theta} \neq \hat{\theta},\ such\ that\ \tilde{F}(X, \tilde{\theta}) = \tilde{Y}, \\ e(\tilde{Y}, Y) \leq e(\hat{Y}, Y)\ \Rightarrow \exists\ \not{!}\hat{\theta}^*. \end{aligned} \tag{4.14}$$

We have assumption that $\theta \Rightarrow F$, this mean not only that guarantees on $\hat{\theta}$ will lead us to the valid solution of both problems $\hat{\theta} \rightarrow \hat{\theta}^*$ (4.15), but also guarantees on F, as having F we can use it to check θ (4.16).

$$if\ \hat{\theta}^* = \theta \Rightarrow F = \hat{F}(\theta) \tag{4.15}$$

$$if\ \hat{F}^* = F, \hat{\theta} \rightarrow \tilde{F}, \Rightarrow \hat{\theta} = \theta \Leftrightarrow \tilde{F} = \hat{F}^* \tag{4.16}$$

Since parameter values and observed TS are both just sets of values, and unique identifiability statement is nothing more than a requirement on the mapping between those sets to be bijective, we can think of the function we are parameterizing, as that bijective mapping. As several possible bijective mappings between two sets can be infinite (we can represent a mapping from the set A to the set B: $A \longrightarrow B$ as $A \longrightarrow Z \longrightarrow B$ where Z is arbitrary set), therefore exists lots of such functions that could be parametrized by θ and produce the observed data (X, Y).

$$\text{if} \quad A \xrightarrow{F_{AB}} B, and \; \exists A \xrightarrow{F_{AQ}} Q \xrightarrow{F_{QB}} B$$

$$\exists \not! Q \rightarrow F_{AB}\, F_{AQ}(F_{QB}), \exists \not! F_{AB}$$

$$\exists F_{aq}(A) = Q, F_{qb}(Q) = B, F_{ab} = F_{qb}(F_{aq}),$$
$$\exists \not! Q \Rightarrow \exists \not! F_{ab}$$

Therefore as the actual form of equation ("template" to insert parameters) is not known and our optimality criterion consists of both estimators, for the function and its parameters, we cannot expect that result of optimization will be exactly targeted dynamical system model.

It means that we should impose additional restrictions to find our exact solutions from the space of optimal estimators. As follows from (4.15), (4.16) we could restrict only one part of solution: either $\hat{F}$ or $\hat{\theta}$. We will consider both options:

Uniqueness of $\hat{F}^*$ because of several reasons:

- Functions of the same restricted form: As we stated in the dynamical systems overview, different processes can leave the equivalent data imprint, therefore exists many configurations for nonlinear dynamical systems that satisfy our optimality condition. As it is hard to provide all available state space for the model in a finite training set even for an ideal problem. For the case of the real-world mechanical systems, it is truly impossible, due to:
  - unknown disturbances from the external environment in the course of exploitation
  - noises from sensors and analogue components
  - inherent flaws of measurements approximations and calculations, etc.
- Even form of the function cannot be guaranteed: As we have the NN as a Black box-based estimator included in our model, we cannot control explicitly what parametrized system is being approximated by a particular model instance.

Therefore we decided to follow the path of restricting $\hat{\theta}$ and then rely on it to optimize $\hat{F}(X, Y, \hat{\theta})$. Restricting of $\hat{\theta}$ will be achieved by introduction explicit dynamical model of our goal system in form of governing differential equations (4.2.2) building for it a White box model.

Isolation of TS prediction and DS parameter identification provides us with various possible ways to achieve them. Models could be trained in a sequential manner, while one is the supervisor for another. But we decided to build a combined model because such setting gives us wider applications cases:

- Could be used in an interactive manner as the Observer part for the real-world system, identifying additional system parameters.
- Parameter identification could be used as the auxiliary loss for the TS-prediction NN, responsible for the 'modeling' of the system itself.
- Translating problem from pure black-box setting to Gray-box we see as the very promising methodology and used in this work approach can be one of the examples of such approach benefits.

Order of accomplishing: As we've already stated that TS prediction part cannot be restricted effectively in case of treating it as a Black-box problem. Therefore restrictions would be lay on the parameter identification task. As the source of the data, we can take either available training data (X, Y) or the TS estimate from BB model $(X, \hat{Y})$. We decided to use the original data, as then it will break the recursive-reference in optimization problem as stated in (4.13).

We also want to use the obtained White-box representation with identified parameters to use it as the system dynamical properties representation. Having a form of the system, we can perform property-based learning procedure of the Black Box model with an aim to improve its robustness to data change and push towards learning the nonlinearities of target system instead of the memorizing data mapping from the training procedure.

4.2.2 System dynamics as the optimization criterion

As a model of interest posses certain nonlinear properties described by a system of ODE parametrized by some parameters θ, solving such a system means finding system states in each moment of time. As we see, solving such a system corresponds to the time-series prediction problem. To combine those two sides of the same problem we use the explicit evolutionary rule of a system in a form of state derivative to predict values. For the case of generated data, it is the same differential equations we used to generate the data (perfect White Box case).

To translate our physical equations into the abstract model for the optimization problem, we will rewrite it in terms of used variables:

As the DC motor speed is governed by the control voltage U_m. It is natural to choose it as x_t standing for the input for our system in the time moment t. The goal of the motor is to transform input current into the output torque. Therefore we use the motor produced torque τ_m as the output of the system y_t. Constant parameters chosen for identification our system are parameters denoting behaviour of the DC motor as the electrodynamical system:

1. armature coil inductance of the motor: $L_a = \theta_1$
2. armature coil resistance: $R_a = \theta_2$
3. constant parameter of the Hysteresis process: $K_{HS\theta 3}$

$$\begin{cases} e_a = K_b w_m \\ y(t) = K_t i_a \\ u_{HS} = \theta_3 K_b i_a^2(t) \\ \frac{dy}{dt}(t) = K_t \frac{di}{dt}(t) = K_t \frac{x(t) - \theta_1 i_m(t) - e(t) - u_{HS}}{\theta_2} \end{cases} \tag{4.17}$$

Where :

$x(t) - U_a$

$y(t) - T_m$

$\theta_1 - R_a$

$\theta_2 - L_a$

$\theta_3 - K_{HS}$

Restriction on individually identifiable parameters: As you can see, all these parameters are chosen in such a way that we cannot rewrite them in a form that is non-linear with respect to the terms θ. This is done intentionally, as if we had members of θ_i and θ_j that were included in only one factor, such as $a\theta_i b\theta_j$, the final model would be absolutely correct for the purposes of the model, but not so useful for representing the identification of individual parameters, the model will be limited only to this value of the product, but not to the individual values of θ_i, θ_j.

Such form of the derivative will be used in the loss functions for subtasks, parameter identification, and TS estimation. We use this method to impose a requirement of the dynamic properties resulting system approximation has to satisfy.

4.2.3 Loss function

As we stated in time series prediction formalization and providing solutions for differential equations with respect to time are equivalent tasks. We use it to formulate our problem in term of both. Differential equations are used to introduce known physical model (WB) and TS prediction – for learning NLDS properties from data representation (BB). We designed our loss function to involve those approaches in combination while also allowing to use only one of them in order to explore the impact of each method used.

In time series analysis the area of interest usually on dynamical properties of TS. They can be extracted not from original TS, but from its "first difference" form madsen2007timeSeries: the difference between every two consequent elements of series (4.18).

$$D(Y) = Y_t - Y_{t-1} \tag{4.18}$$

Equivalently we can get the difference of time series $D(Y)$ in a manner of numerical integration calculating the derivative in the current moment and adding it multiplied by data timestep we time-difference for our time series (4.19) .

$$\tau(Y_t) = time\,of\,Y_t\,observation \tag{4.19}$$

$$h_t = \tau(Y_t) - \tau(Y_{t-1} \tag{4.20}$$

$$\frac{dy}{dt}(t) = \frac{dy}{dt}(x_t, \theta) \tag{4.21}$$

$$D(Y) \approx h_t \frac{dy}{dt}(t) = \hat{D}(Y) \tag{4.22}$$

$$\tag{4.23}$$

Combining those two representations, we can define loss function for our problem as absolute value of plain vector difference (l^1 norm) between estimated and target values (4.24):

$$L(D(Y), \hat{D}(Y)) = |D(Y) - \hat{D}(Y)| \tag{4.24}$$
$$= |(Y_t - Y_{t-1}) - \frac{d\hat{y}}{dt}(t)h_t| \tag{4.25}$$
$$= |(Y_t - Y_{t-1}) - \frac{dy}{dt}(x_t, \hat{\theta})h_t| \tag{4.26}$$
$$= L(D(Y), \frac{d\hat{y}}{dt}(x_t, \hat{\theta})) \tag{4.27}$$

Which can be equivalently used as Loss function for initial TS (4.28):

$$L(Y, \hat{Y}) = |Y - \hat{Y}| \tag{4.28}$$
$$= |(Y_{t-1} + D(Y)) - (Y_{t-1} + \hat{D}(Y))| \tag{4.29}$$
$$= |D(Y) - \hat{D}(Y)| = (4.24) \tag{4.30}$$

We used L1 instead of L2 error measures, despite L2 loss function curve behaves better for gradient-based optimization , but the problem of the multiscaled parameters identification (4.3.2) we are facing, required introduction of balancing weights. If we use them together with L2 error measure our resulting estimate of $\hat{\theta}$ will be harmed, as dependence $Y(\theta)$ is nonlinear and under introducing square measure will result in bias of L2-loss-based estimator $\hat{\theta}_{L_2}(Y) \not\to^* (Y)$.

4.2.4 Loss application

As we showed form from (4.24) is ill-defined, as result we have to split it in two stages. In section dedicated to formalization of our task in terms of optimization problem (2.5) we made the conclusion that only valid way of achieving our goal is to solve our problem as two separate.

To achieve those two stated goals (4.2.2) this purposes we split our learning procedure in two stages repeated on each iteration.

First: Using target values from Y calculate loss for and $\hat{Y}$ resulting in parameters update $\hat{\theta}$. There are no computed gradient on TS predictor as no predicted data is involved in computing:

$$L(D(Y), \frac{dy}{dt}(x_t, \hat{\theta})h_t) \to e_{\hat{\theta}}(Y_D, Y_{\hat{\theta}}) \to \nabla_{\hat{\theta}} Y \tag{4.31}$$

Second: Then against $\hat{Y}_\theta$ and $\hat{Y}_D$ with updated $\hat{\theta}$ and $\hat{Y}_\theta$. As, while $\hat{\theta}$ is frozen. In both samples we use difference of differential equation-based computed derivative and explicit difference of a y time series.

$$L((\hat{Y}_t - \hat{Y}_{t-1}), \frac{d\hat{y}}{dt}(x_t, \hat{\theta})) \to e_{\hat{Y}}(\hat{Y}_D, \hat{Y}_{\hat{\theta}}) \to \nabla_{\hat{Y},\hat{\theta}} \hat{Y} \tag{4.32}$$

Here we will have involved gradients on both $\hat{\theta}$ and $\hat{Y}$. On a stage of programatical implementation we choose to 'freeze' our Parameter Context Nodes and don't

change them on Backpropagation of this error. Eventually it will converge to the same values, but notation for such setting will be clearer and it will much harder to impose some strict explanations on learning process.

4.3 Architecture of Neural Network

To achieve the desired properties following the gray box modeling approach, we decided to follow the Hybrid approach exactly and build parallel models responsible for white-box and black-box modeling parts. It is done by incorporating in same NN the Black-box part represented by LSTM, and special Parameter Context Neurons (PCN) dedicated for explicit parameter identification. A number of these nodes is the same as the number of parameters required for storing a vector of model parameters.

The idea of parameter storing nodes was inspired by Jordan RNN described by . Jordan NN has nodes of a similar type used for remembering previous timestep outputs and passing them as inputs for next timestep, in this way achieving RNN-specific properties of sequential memory.

This approach gives us following benefits:

- **White box isolation:** $\hat{\theta}$ in nodes stored explicitly as their real values grasped in float-point weights in those nodes. We do not use any nonlinearity or hidden layer calculations for these parameters. We performed only gradient propagation with for the target data, therefore this approach does not uses any NN-specific properties. This mean that we can insist on explicity of the identefication procedure and treat it as a White box model.

- **Constant:** As we pursuing estimation of constant terms inherent for the process, no need in activation-based identification dependant of each specific batch of data and involving both additional computations and errors from varying predictions.

- **Clearer Loss:** In case of including $\hat{\theta}$ to output of LSTM we will have to impose constantness of parameters explicitly, what mean additional terms in lost function and more computation involved.

Dataflow: Input for the model x_t is a control signal, when the output is composed of two parts $\hat{y}_t$ – resulting system response and $\hat{\theta}$ estimation of requested dynamical system parameters. Estimated parameter values also used as additional inputs for LSTM, to enforce (not guarantee) the parameter-driven inference. I this role we see such parameter estimation as an auxiliary task for our time series prediction (Trinh et al., 2018).

Parameters identification

As we stated before in it is impossible to build the model of parametrized function with BB and estimate its parameters simultaneously.

Because of this reason we are solving the parameter identification to and learn them from the target data directly applying knowledge about the underlying physical model of a relevant system to overcome the absence of the system.

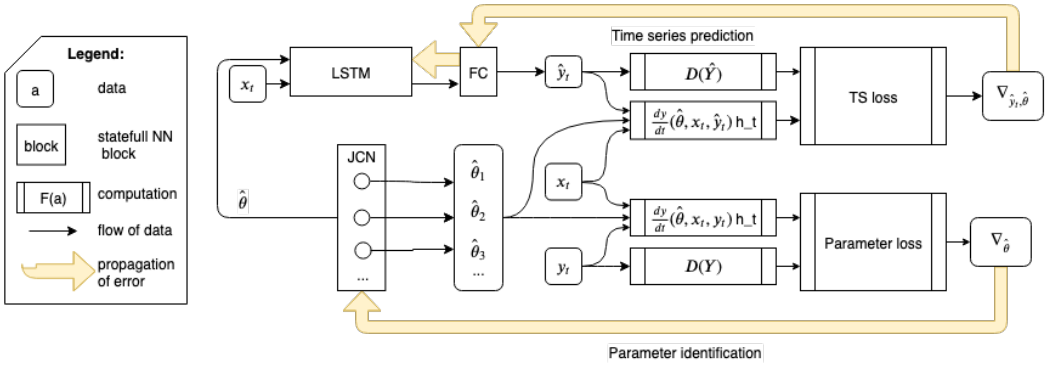

FIGURE 4.4: Dataflow in our main neural network architecture

Modeling of a system

We have already mentioned above NN approximation power causes the problem of training data overfitting. The same problem illustrates that in a common direct supervision approach desired output can be memorized without inferring generalization from data (Arpit et al., 2017). If provided data complexity is low enough and the capacity of weights allows We have already mentioned above NN approximation power causes the problem of training data overfitting. The same problem illustrates that with the directly-supervised learning approach when the training goal is formulated in terms of reproducing desired output if provided data have a low enough diversity for weights to store it by enumeration, for NN is easier to learn a direct mapping between inputs and outputs, than underlying consistent patterns.

As we are addressing the problem of NN memorizing data instead of modeling behavior of the system of interest, we tried to implement learning process without direct supervision, when predicted data are compared against the target data and the resulting error used for learning. Instead, we try to achieve behavior-based modeling by using some representation of desired system dynamical properties in the loss function.

In our problem setting, TS prediction is driven by the dynamical properties of the system in the form of:

- identified parameters of the targeted system
- form of the original system dynamical model
- data of system dynamics

In the result, we expect the NN-based model to express the same dynamical properties, as the original system.

4.3.1 Restrictions on parameters

According to (Sohlberg and Jacobsen, 2008), introduced knowledge about the system follows the methodology of Restricted Black Box model approach. Mentioned restrictions should be laid according to known physical properties of a system. Such an approach also can help us to achieve better estimate convergence, as loss function expressing some real-world physical process could be well-behaved (be differentiable and smooth) only in parameter ranges corresponding to meaningful parameter values.

An illustration of such problem was met in the course of this research: as it can be seen from pic loss function shape around 0 for L_a, the parameter is not determined, as L_a stands in the denominator of our calculated loss. If the initial value of parameter L_a would be randomly initialized with a negative value, there is no way when a gradient-based approach will lead to the global minimum of this function. In general case for NN, similar kind of a problem is not a critical as weights representing parameters of a model give us parameter space of very high dimensionality, but in our case, where the problem was formulated in such low-dimensional space intentionally this requires special attention. We are allowed to introduce such limitation of being positive to all values we estimate in this problem, as it follows their physical sense.

4.3.2 Multiscaled parameters - learning procedure extension and complication

In pursuit of identifying several system parameters at once, we met the problem of combined convergence rate. As parameters stated for learning have a different magnitude of influence on resulting time series their convergence procedure interfere with each other. At the time when parameters with a higher-order magnitude of impact still in the process of converging to its stable minimum, they shadow the benefit of lower-order parameters improved estimates, which does not allow them to converge effectively.

As we use batch-learning procedure and data in each batch can't grasp all system properties accurately, each batch has a different optimal parameter value. The aforementioned stochastic learning process results in that even after estimator convergence to its optimum we still can observe the error oscillations around that parameter value which also require special treatment to diminish its harmful effects on the learning process.

To address this issue and facilitate the parameter identification process, we've introduced an adaptive learning rate into the training process and instance-wise representativeness-based weight coefficients into loss function.

An adaptive loss rate is a common technique for NN training used especially for the oscillating optimal value estimates [(Bengio, 2012),(Smith, 2018)]. We've introduced a decrease in learning rate upon loss function value stops decreasing and starts to oscillate as it decreases the amplitude of gradient-based steps, therefore also dumping the oscillation.

On the same time, introducing weights into a task is not just following the common practice, but subject matter base driven decision. After the change of control signal, it takes some time for the dynamical system output value to reach a new stable state. This period in between steady states is called transitions period . Usually observed in real-world mechanical systems are spending most of the functioning time in a steady state mode. But dynamical properties are tightly concerned with system state memory, therefore are most prominent where this states changes in the course of a transition period.

Some parameters from governing ODE influence the stable state value, and accordingly can be derived from it. Other affects mostly or exclusively on the behavior at the transition periods conditions. Because of it, different exploitation data samples have different significance for different parameter identification .

Therefore disparity of available exploitation data for parameter identification is a usual problem . A similar issue is even more prominent for system failure prediction task, as failure-based data are both inaccurate and expensive to obtain .

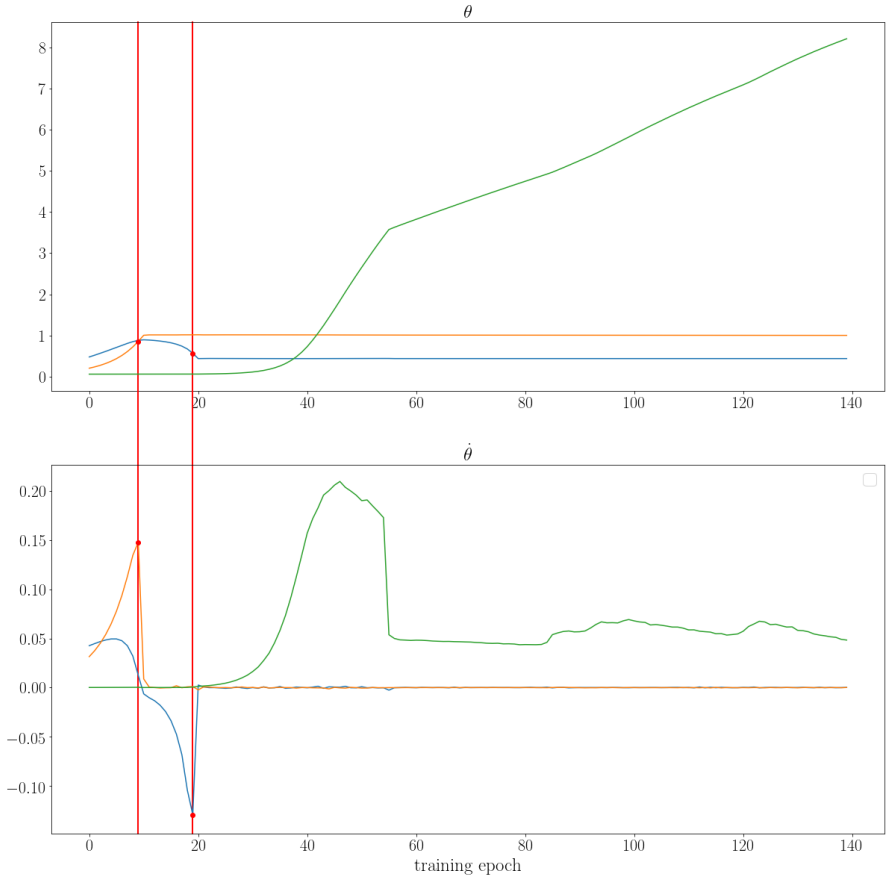

FIGURE 4.5: Convergence of different parameters. Top: parameter values, Bottom: Rate of change. Red lines showing the moment when senior parameter had converged

To address this issue were introduced, weight coefficients are giving more attention to transition period data against steady-state periods. As we already compute derivatives in our loss functions and derivative value by its physical sense is a measure of underlying function rate of change, we used absolute values of derivative $|\frac{dy}{dt}|$ as the basis for our introduced weights. We normalized it between 0 and 1 using as weights coefficients for the errors. To prevent severe unbalance in favor of transition-state based parameters against stable-state ones, we also added some small constant value to weights not to zero-out stable state info at all.

4.3.3 Data normalization-denormalization procedure

One of proposed method issue is that loss function formulated in terms of the time-difference derivative does not grasp *initial state problem* of differential equations. Any TS with corresponding *curvature* will satisfy the optimality criterion despite shift on a constant value.

But in the context of ML problems and time series prediction, especially it is a common procedure to normalize data to 0-1 diapason (Madsen, 2007). Hence data normalization removes an added constant problem before giving it to our model and denormalization after normalizing inputs and denormalizing outputs.

Also, we use denormalization on the stage of parameter-driven loss calculation, as due to the nonlinearity of the system, its identified parameters values does preserve under normalization-denormalization transformation.

Chapter 5

Experimental results

5.1 Experiment description

As the basis for comparison, we will propose four variations of the same model with the identical NN architecture and data used for training. We restricted the difference between the experiments only to the used loss approach, in all cases were used the same architecture, dataset, learning rate, and the number of epochs. Error measure L^1 was preserved the same for all loss functions. We used an excessive number of 800 training epochs to let the slowest converging instance fit the data.

The difference will be in loss functions and loss application procedures: (5.1).

1. The main model: Learning parameters for WB, then using only to train the BB model. Model is agnostic for the data themselves, being led only by identified dynamical properties of the system.

2. Same as (1) but in the process of BB model training use the goal data for the evaluation of dynamics-based loss. Model is aware both of data and separated dynamical properties.

3. WB and BB parts both are learning from data in a conventional, directly supervised manner. BB has WB predicted parameters on input. Usual model, but with additional inputs passively introducing system parameters, but none insights provided about the system form.

4. BB model is learned from data directly, no parameter identification included. Ordinary TS learning as the baseline.

5.1.1 Experiment parameters

While for the models training we use different loss functions, the comparison is performed using only the ordinary error between predicted and target values $|D(Y) - D(\hat{Y})|$. That gives the advantage for model baseline model 3, which were trained under the same conditions.

All data were obtained from the same Matlab model of DC motor described in (4.1.2). Following parameter values of DC motor were used:

$$L_m = 0.5\,\mathrm{H}$$
$$R_m = 1\,\Omega$$
$$K_{HS} = 10$$
$$K_t = 0.01$$
$$K_b = 0.01$$

As we mentioned in (4.2.2), parameters used for identification are L_m, R_m, K_{HS}.

For the evaluation, we chose four additional datasets and added a performance on training set evaluation (train / test split preserved). Special sets are the following. $R[a,b]$ - stands for the range of control signals, $D[a,b]$ - durations of steady-value signals:

1. $R[1,2]D[3,3]$: The one used for training the models. The most constrained depleted set of values, has steady values of signals with constant duration of 3 of values in range of [1,2].
2. $R[5,15]D[3,3]$: evaluation set with depleted steady signal period only.
3. $R[1,2]D[1,20]$: evaluation set with depleted signal variance only.
4. $R[5,15]D[1,20]$: signal with moderate values. All estimator instances trained on such set have approximately equal ability to predict data from all the other proposed sets.
5. $R[5,15]D[1,20] + sin(t)$: Added sinusoidal signal.This dataset represents the signal where the system permanently remains in a transition state.
6. $R[5,15]D[1,20] + v(t)$: where $v(t)$ stands for noise.
7. $R[0,33]D[1,20]$: increased range of duty cycle voltage values. With this dataset, we want to test the robustness to changes in observed data patterns. As the normalized data are insensitive to a steady value shift, we changed the data distribution.
8. $R[5,15]D[.001,2]$: altered range of steady signal duration, present values which does not allow system to reach the new steady-state value before signal changes. This set should underline the transition state importance in expressed data.

5.1.2 NN architecture parameters

To struggle the memorization of data we used the LSTM with an intentionally small number of hidden layer weights. This also should show that it is possible to use such an approach when computing power is expensive or unavailable, as in the example of on-board systems for integrated observers.

Performance of the proposed method best seen in the cases when the training set consists of an intentionally depleted data sample. In this case, the model has to make a conclusion from data about underlying nonlinearities, rather than fit the observed curvature regardless of context. This setting corresponds to a real-world scenario when mechanical systems are usually functioning in limited narrow space from all possible states. To describe the system better with the data we have to

collect them in all specter of exploitation conditions, including extreme and harmful for the real-world system itself. Therefore such experiment setting corresponds to the real-world challenges, especially in cases of system fault prediction tasks and experiments availability.

5.1.3 Results

Results of our modeling experiment provided in following table (5.1). Columns denote the different used data during the experiment, while lines regard to modifications of the model:

Experimental datasamples descriptions in table are following: $V[v_{min}, v_{max}]$ stands for signal values sampling range and $D[D_{min}, D_{max}]$ for the steady states durations. Additional markings are $+sin(t)$ - stands for added sin-wave signal and $+noise$ for added noise.

In the following table (**??**) we placed the obtained results of our models perfromance evaluation. All values in the table is divided by the minimal error among the experiment participating models, therefore .0 means model 100% of lowest error. Models in table are listed in the same order as in listing above (5.1).

NN type	R[1,2] D[3,3]	R[5,15] D[3,3]	R[1,2] D[1,20]	R[5,15] D[1,20] $+sin(t)$	R[5,15] D[1,20] $+v(t)$	R[0,33] D[1,20]	R[5,15] D[.001,2]	R[5,15] D[1,20]
Main	1.26	.021	1.78	.0	.0	.207	.0	.0
Sees the Data	.850	.042	1.18	.104	.050	.365	.247	.106
Supervised + Parameters	.0	.008	.0	.176	.129	.163	.138	.219
Supervised	.144	.0	.174	.229	.152	.0	.072	.270
Baseline error	428.2	3399.9	258.4	4455.6	20867.5	8485.4	4052.0	3241.5

TABLE 5.1: Comparative results of testing. Units: relevant to minimal result column-wise

From results in the table (5.1) it can be seen that there are no uniform arrangements among model performance at the different tasks.

It is illustrative that models performing worse on the training set are the leaders by results of evaluations on more complex datasets.

Our main model is lagging behind in case with training set or similar sets. But the size of this handicap is not dramatic, you can see that the R[1,2]D[3,3] model being trained on and its modification with varying dataset R[1,2]D[1,20] are both having baseline error much lower 428.2 and 258.4 accordingly. On the other hand modification of the training data with higher variability of control signal R[5,15]D[3,3] has significant baseline error of 3399.9, but instead, deviations from the best result are very small, as the biggest deviation is only 4.2% for the main model modification which observed data directly.

Compariosn of the main model results and its modification with direct data observation availabel shows that the isolation of training data gave a clear advantage, as all evaluations resulted in main model performed better.

As we can explain it, duration signal effects the most to the data distribution, as well as the adding $sin(x)$. With those changes system remains in the transition state

more than in training data, therefore model which had shifted the focus of interest fit those insignificant regarding the representation part in training set data, produced the best results.

From this, we can conclude that the current experiment showed that the proposed method can help to address the issue of focusing the model attention on the precise mode of behavior. Therefore the requirement on behavior can be imposed on the model via the parameter identification and introduction of system-specific loss function involving a physical description of the modeled process.

As we see the model had been trained in a directly supervised manner with the introduction of the identified parameters behaves more like the baseline model. It can be interpreted that the training process of such a model is being determined by the TS predictor training procedure.

The general decision in favor of introduction identified parameters to directly supervised model is not absolutely clear from those results. Better performance of supervised model supplied with evaluated parameters on the training-set related examples can be the negative clue, meaning the larger data distribution overfitting. But in the cases of more complicated examples, its performance is considerably higher than the baseline model. The only complication is in extreme cases with very high range or very low duration values, but explanations for those results require extensive analysis on a wide range of different data distributions.

In general, we consider examples on data which includes sin(x) and wider sampling ranges for durations and value ranges as the most prominent arguments in favor of our proposed NN training method.

5.2 Conclusion

Therefore in the course of this research, we showed that the proposed method for indirectly supervised NN training can be used for the controlling of the Black box model remembering the training set, instead of inferencing nonlinearities of the underlying system.

The proposed methodology and architecture could be farther improved and applied to the practical problems, as current versions being made without additional components used to improve the performance of such systems, but potentially undermining the reliability of experimental results.

Also, farther research for better training parameters values should help to specify conditions under which observed the increase in performance remains and provide better-argument results.

Bibliography

Adeli, Hojjat, F Asce, and Xiaomo Jiang (2006). "Dynamic Fuzzy Wavelet Neural Network Model for Structural System Identification". In: *Journal of Structural Engineering-asce - J STRUCT ENG-ASCE* 132. DOI: 10.1061/(ASCE)0733-9445(2006)132:1(102).

Aldrich, John (1997). "R.A. Fisher and the making of maximum likelihood 1912-1922". In: *Statistical Science* 12.3, pp. 162–176. DOI: 10.1214/ss/1030037906. URL: https://doi.org/10.1214/ss/1030037906.

Arpit, Devansh et al. (2017). "A Closer Look at Memorization in Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, pp. 233–242. URL: http://proceedings.mlr.press/v70/arpit17a.html.

Ashby, William Ross. (1956). *An introduction to cybernetics*. J. Wiley, chap. chapter 6: The black box, 86 – 117. DOI: 10.5962/bhl.title.5851. URL: https://doi.org/10.5962/bhl.title.5851.

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. DOI: 10.1109/72.279181. URL: https://doi.org/10.1109/72.279181.

Bengio, Yoshua (2012). "Practical Recommendations for Gradient-Based Training of Deep Architectures". In: *Neural Networks: Tricks of the Trade*, 437–478. ISSN: 1611-3349. DOI: 10.1007/978-3-642-35289-8_26. URL: http://dx.doi.org/10.1007/978-3-642-35289-8_26.

Bernat, Jakub and Slawomir Stepien (2015). "Multi-modelling as new estimation schema for high-gain observers". In: *International Journal of Control* 88.6, pp. 1209–1222. DOI: 10.1080/00207179.2014.1000380. eprint: https://doi.org/10.1080/00207179.2014.1000380. URL: https://doi.org/10.1080/00207179.2014.1000380.

Boeing, Geoff (2016). "Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction". In: *Systems* 4.4, p. 37. DOI: 10.3390/systems4040037. URL: https://doi.org/10.3390/systems4040037.

BOHLIN, TORSTEN (1994). "Derivation of a 'designer's guide' for interactive 'grey-box' identification of nonlinear stochastic objects". In: *International Journal of Control* 59.6, pp. 1505–1524. DOI: 10.1080/00207179408923143. eprint: https://doi.org/10.1080/00207179408923143. URL: https://doi.org/10.1080/00207179408923143.

Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". In: *Online Learning and Neural Networks*. Ed. by David Saad. revised, oct 2012. Cambridge, UK: Cambridge University Press. URL: http://leon.bottou.org/papers/bottou-98x.

Bottou, Léon (2012). "Stochastic Gradient Descent Tricks". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 421–436. DOI: 10.1007/978-3-642-35289-8_25. URL: https://doi.org/10.1007/978-3-642-35289-8_25.

Box, George E. P. (1976). "Science and Statistics". In: *Journal of the American Statistical Association* 71.356, pp. 791–799. DOI: 10.1080/01621459.1976.10480949. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1976.10480949. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1976.10480949.

Bunge, Mario (1963). "A General Black Box Theory". In: *Philosophy of Science* 30.4, pp. 346–358. ISSN: 00318248, 1539767X. URL: http://www.jstor.org/stable/186066.

Cen, Z., J. Wei, and R. Jiang (2011). "A Grey-Box Neural Network based identification model for nonlinear dynamic systems". In: *The Fourth International Workshop on Advanced Computational Intelligence*, pp. 300–307. DOI: 10.1109/IWACI.2011.6160021.

Cen, Zhaohui, Jiaolong Wei, and Rui Jiang (2013). "A gray-box neural network-based model identification and fault estimation scheme for nonlinear dynamic systems". In: *International journal of neural systems* 23, p. 1350025. DOI: 10.1142/S0129065713500251.

Che, Zhengping et al. (2018). "Recurrent Neural Networks for Multivariate Time Series with Missing Values". In: *Scientific Reports* 8.1. DOI: 10.1038/s41598-018-24271-9. URL: https://doi.org/10.1038/s41598-018-24271-9.

Chen, Wei-Ching (2008). "Nonlinear dynamics and chaos in a fractional-order financial system". In: *Chaos, Solitons & Fractals* 36.5, pp. 1305–1314. DOI: 10.1016/j.chaos.2006.07.051. URL: https://doi.org/10.1016/j.chaos.2006.07.051.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

CICCARELLA, G., M. DALLA MORA, and A. GERMANI (1993). "A Luenberger-like observer for nonlinear systems". In: *International Journal of Control* 57.3, pp. 537–556. DOI: 10.1080/00207179308934406. URL: https://doi.org/10.1080/00207179308934406.

Conceptual scheme NN. URL: https://en.wikipedia.org/wiki/Artificial_neural_network#/media/File:Colored_neural_network.svg. (accessed: 15.04.2019).

Crossing, Gillett. *Westingson DC generator*. URL: https://commons.wikimedia.org/wiki/File:Westinghouse_DC_generator_(1907)_driven_by_Ruston_Proctor_engine,_Museum_of_Science_\%26_Industry,_Birmingham_21.1.1995_Scans124_(11634042363).jpg. (accessed: 15.04.2019).

Dinh, Huyen, Shubhendu Bhasin, and W.E. Dixon (2011). "Dynamic Neural Network-based Robust Identification and Control of a class of Nonlinear Systems". In: pp. 5536 –5541. DOI: 10.1109/CDC.2010.5717445.

Drakunov, S. V. (1992). "Sliding-mode observers based on equivalent control method". In: *[1992] Proceedings of the 31st IEEE Conference on Decision and Control*, 2368–2369 vol.2. DOI: 10.1109/CDC.1992.371368.

Du, Yunshu et al. (2018). "Adapting auxiliary losses using gradient similarity". In: *arXiv preprint arXiv:1812.02224*.

Edwards, A.W.F. (1984). *Likelihood*. Cambridge science classics. Cambridge University Press. ISBN: 9780521318716. URL: https://books.google.com.ua/books?id=LL08AAAAIAAJ.

François, Deloche. *Unfolded representation of RNN*. URL: https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg. (accessed: 15.04.2019).

Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins (2000). "Learning to Forget: Continual Prediction with LSTM". In: *Neural Computation* 12.10, pp. 2451–

2471. DOI: 10.1162/089976600300015015. URL: https://doi.org/10.1162/089976600300015015.

Goodwin, G.C., S.F. Graebe, and M.E. Salgado (2001). *Control System Design*. Prentice Hall. ISBN: 9780139586538. URL: https://books.google.com.ua/books?id=7dNSAAAAMAAJ.

Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton (2013). "Speech Recognition with Deep Recurrent Neural Networks". In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 38. DOI: 10.1109/ICASSP.2013.6638947.

Guastello, S.J. (2013). *Chaos, Catastrophe, and Human Affairs: Applications of Nonlinear Dynamics To Work, Organizations, and Social Evolution*. Taylor & Francis. ISBN: 9781134787852. URL: https://books.google.com.ua/books?id=9U2fVd0Q8soC.

Hauth, Jan (2008). "Grey-Box Modelling for Nonlinear Systems". doctoralthesis. Technische Universität Kaiserslautern. URL: http://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-22879.

Hengl, S. et al. (2007). "Data-based identifiability analysis of non-linear dynamical models". In: *Bioinformatics* 23.19, pp. 2612–2618. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm382. eprint: http://oup.prod.sis.lan/bioinformatics/article-pdf/23/19/2612/16861405/btm382.pdf. URL: https://doi.org/10.1093/bioinformatics/btm382.

Hinrichsen, Diederich and Anthony J. Pritchard (2005). *Mathematical Systems Theory I*. Springer Berlin Heidelberg. DOI: 10.1007/b137541. URL: https://doi.org/10.1007/b137541.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hoshi, Rosangela Akemi et al. (2013). "Poincaré plot indexes of heart rate variability: Relationships with other nonlinear variables". In: *Autonomic Neuroscience* 177.2, pp. 271–274. DOI: 10.1016/j.autneu.2013.05.004. URL: https://doi.org/10.1016/j.autneu.2013.05.004.

Kalman, R. E. (1960). "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1, p. 35. DOI: 10.1115/1.3662552. URL: https://doi.org/10.1115/1.3662552.

Kara, Tolgay and İlyas Eker (2004). "Nonlinear modeling and identification of a DC motor for bidirectional operation with real time experiments". In: *Energy Conversion and Management* 45.7-8, pp. 1087–1106. DOI: 10.1016/j.enconman.2003.08.005. URL: https://doi.org/10.1016/j.enconman.2003.08.005.

Kosorok, Michael R. (2008). *Introduction to Empirical Processes and Semiparametric Inference*. Springer New York. DOI: 10.1007/978-0-387-74978-5. URL: https://doi.org/10.1007/978-0-387-74978-5.

Krener, A. and W. Respondek (1985). "Nonlinear Observers with Linearizable Error Dynamics". In: *SIAM Journal on Control and Optimization* 23.2, pp. 197–216. DOI: 10.1137/0323016. eprint: https://doi.org/10.1137/0323016. URL: https://doi.org/10.1137/0323016.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

LeCun, Yann A. et al. (2012). "Efficient BackProp". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

Lorenz, Edward N. (1963). "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Sciences* 20.2, pp. 130–141. DOI: 10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2. URL: https://doi.org/10.1175/1520-0469(1963)020<0130:dnf>2.0.co;2.

Madsen, H. (2007). *Time Series Analysis*. Chapman & Hall/CRC Texts in Statistical Science. CRC Press. ISBN: 9781420059687. URL: https://books.google.com.ua/books?id=WAHLBQAAQBAJ.

Mandic, Danilo (2001). *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Wiley. ISBN: 0471495174. URL: https://www.xarg.org/ref/a/0471495174/.

Marblestone, Adam H., Greg Wayne, and Konrad P. Kording (2016). "Toward an Integration of Deep Learning and Neuroscience". In: *Frontiers in Computational Neuroscience* 10. DOI: 10.3389/fncom.2016.00094. URL: https://doi.org/10.3389/fncom.2016.00094.

Miljanovic, Milos (2012). "Comparative analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction". In: *Indian Journal of Computer Science and Engineering* 3.

Minorsky., N. (2009). "DIRECTIONAL STABILITY OF AUTOMATICALLY STEERED BODIES". In: *Journal of the American Society for Naval Engineers* 34.2, pp. 280–309. DOI: 10.1111/j.1559-3584.1922.tb04958.x. URL: https://doi.org/10.1111/j.1559-3584.1922.tb04958.x.

Narendra, Kumpati and K Parthasarathy (1990). "Identification and Control of Dynamical System Using Neural Networks". In: vol. NN-1, 1737 –1738 vol.2. DOI: 10.1109/CDC.1989.70448.

Narendra, Kumpati and Kannan Parthasarathy (1992). "Neural networks and dynamical systems". In: *Int. J. Approx. Reasoning* 6, pp. 109–131. DOI: 10.1016/0888-613X(92)90014-Q.

Newey, Whitney K. and Daniel McFadden (1994). "Chapter 36 Large sample estimation and hypothesis testing". In: *Handbook of Econometrics*. Elsevier, pp. 2111–2245. DOI: 10.1016/s1573-4412(05)80005-4. URL: https://doi.org/10.1016/s1573-4412(05)80005-4.

Ogunmolu, Olalekan et al. (2016). *Nonlinear Systems Identification Using Deep Dynamic Neural Networks*. arXiv: 1610.01439 [cs.NE].

Olah, Christopher. *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. (accessed: 15.04.2019).

Ostwald, Michael J (2013). "The Fractal Analysis of Architecture: Calibrating the Box-Counting Method Using Scaling Coefficient and Grid Disposition Variables". In: *Environment and Planning B: Planning and Design* 40.4, pp. 644–663. DOI: 10.1068/b38124. URL: https://doi.org/10.1068/b38124.

Pham, D and Dervis Karaboga (1999). "Training Elman and Jordan networks for system identification using genetic algorithms". In: *Artificial Intelligence in Engineering* 13, pp. 107–117. DOI: 10.1016/S0954-1810(98)00013-2.

Plank, Barbara, Anders Søgaard, and Yoav Goldberg (2016). "Multilingual Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Models and Auxiliary Loss". In: *CoRR* abs/1604.05529. arXiv: 1604.05529. URL: http://arxiv.org/abs/1604.05529.

Puscas, Gheorgh et al. (2009). "NONLINEAR SYSTEM IDENTIFICATION BASED ON INTERNAL RECURRENT NEURAL NETWORKS". In: *International Journal of Neural Systems* 19.02. PMID: 19496207, pp. 115–125. DOI: 10.1142/S0129065709001884. eprint: https://doi.org/10.1142/S0129065709001884. URL: https://doi.org/10.1142/S0129065709001884.

Puu, Tönu (2003). *Attractors, Bifurcations, & Chaos*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-24699-2. URL: https://doi.org/10.1007/978-3-540-24699-2.

Rieger, Craig G., David I. Gertman, and Miles. A. McQueen (2009). "Resilient control systems: Next generation design research". In: *2009 2nd Conference on Human System Interactions*. IEEE. DOI: 10.1109/hsi.2009.5091051. URL: https://doi.org/10.1109/hsi.2009.5091051.

Ripley, B.D. (2007). *Pattern Recognition and Neural Networks*. Cambridge University Press. ISBN: 9780521717700. URL: https://books.google.com.ua/books?id=m12UR8QmLqoC.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

Sastry, P. S., G. Santharam, and K. P. Unnikrishnan (1994). "Memory neuron networks for identification and control of dynamical systems". In: *IEEE Transactions on Neural Networks* 5.2, pp. 306–319. ISSN: 1045-9227. DOI: 10.1109/72.279193.

Shampine, Lawrence F. and Mark W. Reichelt (1997). "The MATLAB ODE Suite". In: *SIAM Journal on Scientific Computing* 18.1, pp. 1–22. DOI: 10.1137/s1064827594276424. URL: https://doi.org/10.1137/s1064827594276424.

sherrellbc. *DCMotorScheme*. URL: https://i.stack.imgur.com/s37EI.jpg. (accessed: 15.04.2019).

Sjöberg, Jonas et al. (1995). "Nonlinear black-box modeling in system identification: a unified overview". In: *Automatica* 31.12. Trends in System Identification, pp. 1691 –1724. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(95)00120-8. URL: http://www.sciencedirect.com/science/article/pii/0005109895001208.

Sjövall, Per and Thomas Abrahamsson (2008). "Substructure system identification from coupled system test data". In: *Mechanical Systems and Signal Processing - MECH SYST SIGNAL PROCESS* 22, pp. 15–33. DOI: 10.1016/j.ymssp.2007.06.003.

Smith, Leslie N. (2018). *A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay*. arXiv: 1803.09820 [cs.LG].

Sohlberg, B. and E.W. Jacobsen (2008). "GREY BOX MODELLING – BRANCHES AND EXPERIENCES". In: *IFAC Proceedings Volumes* 41.2. 17th IFAC World Congress, pp. 11415 –11420. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20080706-5-KR-1001.01934. URL: http://www.sciencedirect.com/science/article/pii/S1474667016408025.

Torokhti, Anatoli and Phil Howlett (2007). *Computational Methods for Modeling of Nonlinear Systems*. 1st. San Diego, USA: Elsevier Science. ISBN: 0444530444, 9780444530448.

Trinh, Trieu H. et al. (2018). *Learning Longer-term Dependencies in RNNs with Auxiliary Losses*. arXiv: 1803.00144 [cs.LG].

V. Vantsevich D.Gorsich, A.Lozinsky L.Demkiv T.Borovets (2018). "STATE OBSERVERS: AN OVERVIEW AND APPLICATION TO AGILE TIRE SLIPPAGE DYNAMICS". In: Trends in System Identification. URL: https://www.istvs.org/10th-asia-pacific-conference-kyoto.

Wang, Jeen-Shing and Yen-Ping Chen (2006). "A fully automated recurrent neural network for unknown dynamic system identification and control". In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 53, pp. 1363 –1372. DOI: 10.1109/TCSI.2006.875186.

Werbos, Paul J. (1988). "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural Networks* 1.4, pp. 339–356. DOI: 10.1016/

0893-6080(88)90007-x. URL: https://doi.org/10.1016/0893-6080(88)90007-x.

Werbos, P.J. (1975). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University. URL: https://books.google.com.ua/books?id=z81XmgEACAAJ.

Wiener, N. (1965). *Cybernetics Or Control and Communication in the Animal and the Machine*. DE-601)251474038: MIT paperback series. M.I.T. Press. ISBN: 9780262730099. URL: https://books.google.it/books?id=NnM-uISyywAC.

Yue, Meng and Robert Schlueter (2005). "An Algorithm and Properties Enabling Identification of Bifurcation Subsystems". In: *Electric Power Components and Systems* 33.6, pp. 611–628. DOI: 10.1080/15325000590885243. eprint: https://doi.org/10.1080/15325000590885243. URL: https://doi.org/10.1080/15325000590885243.