

**UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS**

**Arthur Maciel Goulart de Oliveira Bertemes**

**Identification of an Adaptative Model for an  
Articulated robot:  
A black-box approach**

Florianópolis  
2019



**Arthur Maciel Goulart de Oliveira Bertemes**

**Identification of an adaptative model for articulated  
robot: A black box approach**

Report submitted to the Universidade Federal de Santa Catarina as requirement in order to be approved in the Control and Automation Engineering course **DAS 5511: Projeto de Fim de Curso.**

Advisor: Prof. Dr.-Ing. Marcelo Ricardo Stemmer

Florianópolis  
2019



**Arthur Maciel Goulart de Oliveira Bertemes**

**Identification of an adaptative model for articulated robot: A  
black box approach**

This monograph was evaluated in the context of the course “DAS 5511: Projeto de Fim de Curso” and approved in its end form by the Control and Automation Engineering Course.

Florianópolis, 29th of July of 2019

**Examination Board:**

M.Sc. Christoph Storm

Laboratory Advisor

Werkzeugmaschinenlabor WZL der RWTH Aachen

Prof. Dr.-Ing. Marcelo Ricardo Stemmer

Advisor in the Course

Universidade Federal de Santa Catarina

Prof. Dr.-Ing. Eduardo Camponogara

Appraiser

Universidade Federal de Santa Catarina

Eduardo Campos Kneipp

Debater

Universidade Federal de Santa Catarina

Ramon Jesuino Dettmer

Debater

Universidade Federal de Santa Catarina



## ACKNOWLEDGEMENTS

I would like to thank my family for all their love, affection, support and encouragement during the completion of this work and the graduation course.

I would like to thank all my friends and colleagues who were with me on this journey. Especially Bruno Rodrigues Battistotti, without him, this work would not be possible.

I would like to thank Christoph C. Storm for having accepted me and given me the opportunity to work at the WZL. The personal and professional growth that this experience provided was extremely rewarding.

I would like to thank the Brazilians in Aachen: Matheus, Igor, Henrique, Eduardo and others. Adapting to a completely new environment was much easier thanks to you.

I would also like to thank *Universidade Federal de Santa Catarina, Departamento de Automação e Sistemas* and all its teachers and employees for providing me with a high-quality degree.





## RESUMO

O conceito emergente e cada vez mais consolidado da indústria 4.0 traz consigo em um de seus pilares a rápida adaptação aos anseios do mercado, sendo necessário prover cada vez mais produtos mais individualizados e customizados, fazendo da flexibilidade, uma das características chaves para prosperar nesse ambiente.

Nesse contexto, a montagem final de produtos de larga escala, como por exemplo, na indústria automotiva, apresenta um ambiente bastante flexível devido ao baixo número de produtos e a imensa gama de variações dos mesmos. Essa flexibilidade, porém, é obtida ao custo de um baixo nível de automação no ambiente da montagem final.

Manipuladores robóticos apresentam-se como elementos bastante flexíveis: apresentam alto grau de liberdade de movimentação e capazes de atuar na execução das mais diversas tarefas. Tradicionalmente, estes são empregados em um layout celular que permitem um alto grau de versatilidade.

A fim de se diminuir o ciclo de tempo da montagem final nas linhas de produção, cada vez mais, opta-se por um layout de fluxo contínuo, ininterrupto, capaz de reduzir em mais de 63% os tempos da montagem.

Assim, o Werkzeugmaschinenlabor -WZL (Laboratório de Máquinas-Ferramenta) da Universidade Técnica da Renânia do Norte-Vestefália em Aachen através do projeto FASIM - *Final Assembly in Motion* (Montagem Final em Movimento) busca solucionar toda a problemática envolvida na sincronização dos manipuladores robóticos, com o restante dos componentes da linha de produção em um ambiente de movimento contínuo.

Através de diversos trabalhos envolvendo anos de pesquisa, o laboratório optou por uma sincronização realizada através de um controle preditivo (*Model Predictive Control* - MPC) capaz de: garantir a sincronização requerida ao passo que compensa interferências das vibrações do sistema de movimentação e; lidar com o tempo de zona morta proveniente da comunicação entre sistema de controle, manipuladores robóticos e o sistema de medição.

Como qualquer abordagem de controle clássica, para um devido ajuste e um bom resultado do sistema de controle, é preciso antes de mais nada um bom modelo que represente o sistema. Durante as etapas mais recentes do projeto, o modelo do sistema foi obtido através de uma estrutura caixa-preta utilizando a captação de dados reais de entrada e saída do sistema.

Esse trabalho se propõe, então, a identificar um modelo de um manipulador robótico, acoplado ao sistema de medição de larga escala, através de uma abordagem caixa-preta, que gere resultados mais próximos ao sistema real que o modelo até então obtido pelo WZL.

O trabalho se centrou em pesquisar diversas técnicas de identificação e possíveis ferramentas de implementação que pudesse proporcionar uma integração rápida ao ambiente do laboratório. Indo desde identificação usando-se de técnicas de aprendizado de máquina, otimização a estimação online de parâmetros do sistema. Visou-se estudar a possibilidade de identificação de um modelo adaptativo capaz de aproximar a dinâmica do sistema real em pose do laboratório a fim de melhorar os resultados do controle projetado pelo mesmo.

**Palavras-chave:** Identificação de sistemas. Estimação de parâmetros de sistemas. Estimação online. Modelos caixa-preta.

## ABSTRACT

The growing desire for more individualized products requires from the industry a high degree of flexibility and shorter production times. In this context, in order to achieve the required quality and time standards in the final assembly in the automotive industry, the process is done through a high degree of manual work in continuous assembly line.

Seeking to create a more automated production environment while maintaining the same levels of flexibility and quality, the Werkzeugmaschinenlabor -WZL through the FASIM (Final Assembly in Motion) project, studies the possibility of employing robotic manipulators synchronized with the movement of the product in the continuous production line.

Synchronization is performed through a model predictive control (MPC) capable of compensating for deviations of the manipulator system and conveyor system while, rejecting system's disturbances and dealing with the dead-time delays from the robot and measurement system .

In order for the control to have an adequate behavior, it needs a good model of the system. Thus, this work aims to study methods and tools capable of providing a more accurate model than the current one in the possession of the laboratory. Several methods and tools were researched, which could provide an adaptive model for the robotic system.

It focused on evaluating the possibility of implementing a neural network model and the implementation of an online estimator of system parameters.

**Keywords:** Systems identification. Estimation of system parameters. Online estimation. Black box models.



## FIGURE LIST

Figure 1- Production Engineering Cluster at RWTH Aachen Campus.....	25
Figure 2 - Exemplary of standard cell in matrix production .....	31
Figure 3 - 6-DOF Robot Manipulator Scheme .....	32
Figure 4 -Forward kinematics (left) and inverse kinematics examples scheme.....	35
Figure 5 - Two-mass flexible joint model for robot arm.....	36
Figure 6 - Classification of large-volume measuring systems .....	38
Figure 7 - Examples of large-volume metrology applications .....	38
Figure 8 – Basic structure of MPC .....	39
Figure 9 - State-space representation.....	40
Figure 10 - Feedforward network with two hidden layers .....	46
Figure 11 - Recurrent Network scheme.....	47
Figure 12 - Multilayer Perceptron network scheme .....	53
Figure 13 - Genetic Algorithm: Crossover example .....	56
Figure 14 - Genetic Algorithm: Mutation Example .....	56
Figure 15 - Overview of the development environment.....	57
Figure 16 - Overview scheme of whole system .....	57
Figure 17 - Structure of EGM controller .....	59
Figure 18 - EGM speed characteristic curve .....	59
Figure 19 – iGPS .....	60
Figure 20 - Overfitting example: Input, Output, Step trained NN, Sine trained NN .....	64
Figure 21 - Simulation of NARX NN model with WLZ's model - Step Input .....	65
Figure 22 - Simulation of NARX NN model with WLZ's model - Sine Input .....	65
Figure 23 - Simulation of NARX NN model with WLZ's model – Uniform Random Signal.	66
Figure 24 - Simulation Results of NARX NN with the Real System .....	66
Figure 25 - Proposed scheme to neural network for adapt SS model.....	67
Figure 26 - Optimization tool interface .....	69
Figure 27 - Genetic Algorithm results: Step input, Sine input. Position, error .....	69
Figure 28 - Simulation of ARX adaptive model approximating WZL's model .....	71
Figure 29 – Simulation of ARX adaptive model approximating the real system .....	71
Figure 30 - Evolution of non-zero parameter of Matrix B .....	72
Figure 31 - Order selection section.....	73

Figure 32 - Simulation with WZL's model as base .....	74
Figure 33 - Nonzero parameters from $B$ matrix – Off/On-line estimation .....	74
Figure 34 - Gradient methods in comparison.....	75
Figure 35 - Example of final validation of models. Output from models. Error .....	77
Figure 36 - Nonzero parameters of $B$ matrix varying over time.....	78
Figure 37 - Mean Square Error – Longer Simulation .....	79
Figure 38 - Mean Square Error – Short Simulation .....	79

## TABLE LIST

Table 1 - Nonlinear models' regressors .....	44
Table 2 - Key figures of the iGPS .....	61
Table 3 - iGPS characteristics observed by WZL .....	61





## **ACRONYMS AND ABBREVIATIONS LIST**

UFSC – Universidade Federal de Santa Catarina

DAS – Departamento de Automação e Sistemas

WZL – Werkzeugmaschinenlabor

RWTH – Rheinisch-Westfälische Technische Hochschule Aachen

GPS – Global Positioning System

iGPS – indoors GPS

MPC – Model Predictive Control

EGM – External Guided Motion

6-DOF – Six Degrees of Freedom

TCP – Tool Center Point

FASIM – Final Assembly in Motion

AGV – Automated Guided Vehicles

LS – Least Squares

SMR – Spherical Mirror Reflector

TF – Transfer Function

MSE – Mean Square Error



# CONTENTS

<b>1</b>	<b>INTRODUCTION .....</b>	<b>21</b>
1.1	OVERVIEW AND MOTIVATIONS .....	21
1.2	OBJECTIVE.....	22
1.3	LOCATION OF THE SUBJECT WITHIN THE CONTROL AND AUTOMATION ENGINEERING COURSE .....	22
1.4	ADOPTED METHODOLOGY .....	23
1.5	DOCUMENT STRUCTURE .....	23
<b>2</b>	<b>THE LABORATORY .....</b>	<b>25</b>
2.1	CHAIR OF METROLOGY AND QUALITY MANAGEMENT .....	25
2.1.1	<i>Model-based Systems Department.....</i>	<i>26</i>
<b>3</b>	<b>THE PROBLEM.....</b>	<b>27</b>
3.1	MAIN OBJECTIVE.....	28
3.2	SPECIFIC OBJECTIVES .....	28
3.3	PROBLEMS IDENTIFIED IN THE CURRENT STATE OF THE PROJECT .....	28
<b>4</b>	<b>THEORETICAL BACKGROUND.....</b>	<b>31</b>
4.1	FLEXIBLE PRODUCTION SYSTEMS .....	31
4.2	SYNCHRONIZATION FOR AUTONOMOUS ASSEMBLY IN MOTION.....	31
4.3	ROBOT MANIPULATORS OVERVIEW.....	32
4.3.1	<i>Robot dynamics and non-linearities .....</i>	<i>33</i>
4.3.1.1	Actuators .....	33
4.3.1.2	Sensors.....	33
4.3.1.3	Kinematics .....	34
4.3.1.3.1	Position Kinematics .....	34
4.3.1.3.2	Velocity Kinematics .....	35
4.3.1.4	Rigid Body Dynamics.....	35
4.3.1.5	Flexible Body Dynamics .....	36
4.4	GLOBAL REFERENCE SYSTEMS IN PRODUCTION .....	37
4.5	LARGE-VOLUME METROLOGY .....	37
4.5.1	<i>Laser based non-contact measurement methods.....</i>	<i>39</i>
4.6	MPC'S MODEL .....	39
4.7	DYNAMIC SYSTEM IN STATE SPACE.....	40
4.8	MACHINE LEARNING .....	40
4.9	SYSTEM IDENTIFICATION.....	41
4.9.1	<i>The identification problem.....</i>	<i>41</i>
4.9.2	<i>Regression Possibilities .....</i>	<i>43</i>
4.9.2.1	Linear case .....	43
4.9.2.2	Nonlinear case .....	44
4.9.3	<i>Nonlinear Mappings .....</i>	<i>44</i>

4.9.3.1	Sigmoid Neural Networks.....	46
4.9.3.2	Multi-layer Networks .....	46
4.9.3.3	Recurrent Networks .....	46
4.10	LINEAR REGRESSION FOR PARAMETER ESTIMATION AND THE LEAST-SQUARES METHOD.....	47
4.10.1	<i>Weighted least-squares criterion</i> .....	48
4.11	RECURSIVE ESTIMATION METHODS .....	48
4.11.1	<i>Recursive Algorithms for Online Estimation</i> .....	49
4.11.2	<i>Matlab's implementation of recursive algorithms for online estimation</i> .....	51
4.11.3	<i>Forgetting Factor</i> .....	52
4.11.4	<i>Kalman Filter</i> .....	52
4.11.5	<i>Normalized and Unnormalized Gradient</i> .....	53
4.12	MULTILAYER PERCEPTRON (MLP) - BACKPROPAGATION NETWORK FOR REGRESSION.....	53
4.13	OPTIMIZATION AND GENETIC ALGORITHM .....	55
4.13.1	<i>Genetic Algorithm</i> .....	55
<b>5</b>	<b>SYSTEM DESCRIPTION.....</b>	<b>57</b>
5.1	ABB IRB-4600 / IRC5 .....	58
5.1.1	<i>EGM – Externally Guided Motion</i> .....	58
5.2	NIKON iGPS.....	60
<b>6</b>	<b>IMPLEMENTATION.....</b>	<b>63</b>
6.1	APPLYING MACHINE LEARNING TO THE IDENTIFICATION PROBLEM.....	63
6.1.1	<i>Adapting the NN structure for the WZL case</i> .....	66
6.2	AN OPTIMIZATION APPROACH .....	68
6.3	RECURSIVE ALGORITHMS FOR ONLINE ESTIMATION .....	70
<b>7</b>	<b>FINAL RESULTS .....</b>	<b>77</b>
<b>8</b>	<b>FINAL CONSIDERATIONS AND PERSPECTIVES .....</b>	<b>81</b>
<b>9</b>	<b>REFERENCES.....</b>	<b>83</b>
APPENDIX A	MATLAB FUNCTION: FITNESS FUNCTION TO OPTIMUM TOOL .....	87
APPENDIX B	SIMULATION RESULTS TO FORGETTING FACTOR ALGORITHM.....	89
APPENDIX C	MSE COMPARISON: WZL'S MODEL WITH ONLINE ESTIMATION .....	91
APPENDIX D	MSE COMPARISON: DIFFERENT ORDERS MODELS.....	93
ANNEX A	NEURAL TIME SERIES TOOL .....	95
ANNEX B	RECURSIVE POLYNOMIAL MODEL ESTIMATOR BLOCK INTERFACE .....	97
ANNEX C	SYSTEM IDENTIFICATION USER INTERFACE .....	99

## 1 INTRODUCTION

The work that is described along this thesis was developed in the Model-based Systems Department of the Chair of Metrology and Quality Management from the Laboratory of Machine Tools and Production Engineering (Werkzeugmaschinenlabor – WZL), of the RWTH Aachen University [1].

It consists mainly in the research and development of an adaptive model for a robotic manipulator. It seeks to observe the applicability of machine learning methods, in the form of a model based on a neural network, or to improve an existing linear model of the robotic system. This work would benefit the research and design of the articulated robot control system.

### 1.1 Overview and motivations

Nowadays the industry undergoes a new set of changes, the strengthening of the concept of the industry 4.0 brings with it a new collection of challenges. In this context, a highly agile and flexible environment is required to meet the demands of high product variability and short product life cycles [2].

In the case of the automobile industry, more specifically in the assembly of large components in a car or truck production line, the high degree of flexibility that can provide the desired production time and the required tolerance is given by a large percentage of manual process [3]. This is noticed as the final assembly features a level of automation between 20%, while other production phases have a level of around 90% [4].

Industrial articulated robots are primarily used to increase productivity, save costs and eliminate dangerous and laborious work [5] and, in this context, can be seen as a highly flexible tool that could be used to raise the level of automation, providing the desirable flexibility and keeping up with the needed requirements at the same time.

Aiming to provide an autonomous system capable of dealing with the large-components assembly in motion, the Metrology-assisted Assembly Group and WZL, through a series of researches, developed the FASIM (Final Assembly in Motion) project. The current phase of the project draws on an MPC control approach to handle the synchronization between robots and the product's motion system.

For its control approach to be successful, it is firstly necessary to have a good model of the system that the controller could base itself on. In the case of the FASIM project, the system is composed by an industrial robotic manipulator merged with the large-volume

metrology system chosen. However, identifying robot models is a challenging task since an industrial robot is a multivariable, nonlinear, unstable and resonant system [5].

At the moment, WZL has identified a linear model with a black-box approach that represents the dynamics of the system and the metrology system, at the same time, it seeks to improve it to achieve better results with its controller that would correspond to the high demands of the final assembly environment.

## 1.2 Objective

The main objective of this work, inserted in the context above mentioned, is to research and provide an improved model for the FASIM project, that could provide a better approximation to the real systems than the current model.

A black box model was selected because of its convenient identification form (it requires no previous knowledge of the system), and because WZL utilizes a control interface provide by the robot's manufacturer.

One of the minor objectives is to evaluate the possibility of applying machine learning methods, in the form of neural networks, to provide an adaptive model for the controller, that would adjust itself in system's runtime.

## 1.3 Location of the subject within the Control and Automation Engineering course

Research and identify methods capable of providing a better model for the control system presents itself as a broad problem with many correlations with the areas of expertise of a control and automation engineer. In general, the engineer graduated in the control and automation course of the *Universidade Federal de Santa Catarina* (UFSC) presents a deepen knowledge in three main areas: Process Control; Industrial Informatics and; Manufacturing Automation.

In the Process Control field, the most notable correlation is the process modeling, even so, the knowledge in analysis, design and implementation of controllers for linear and non-linear systems are totally valid and highly helpful as this work took place in parallel with the development of a newer version of the WLZ's controller.

In the informatics field, the subject of Artificial Intelligence and Machine Learning are approached as it is studied the possibility of applying them in the identification of the system

Finally, the field of manufacturing automation is addressed as knowledge in, automated systems programming, production management, process planning, integrated

manufacturing systems, modeling and performance evaluation of manufacturing systems, promote a knowledge base for understanding the problem.

In conclusion, the whole scope of the project synthesizes the work environment of a control and automation engineer as it inserts itself on the three main fields of learning of the course. At the same time, all mathematical and physical expertise, together with a base knowledge in computer science and mechanics fundamentals are extremely useful for the development of this work.

#### **1.4 Adopted methodology**

Given that this work will be characterized as a research work, several technologies and tools will be addressed. The initial goal is to evaluate if the use of neural networks is possible and for that point look for alternatives. That being said, the plan adopted for the workflow can be summarize in the following steps:

1. Search of literature regarding the state of art of a given approach;
2. Evaluation if it is appropriate for the identification problem;
3. Evaluation of implementation tools, for the chosen approach, that would be suitable for the project;
4. Implement the approach selected aiming to identify a model that can be approximated to the WZL's current model;
5. If the previous step presents a good result, capture real data from the system to use as and identification and test data set;
6. Identify a model that can approximate the real data of the system;
7. Finally, if the identified model presents an acceptable results, test the given approach with WZL's controller.

#### **1.5 Document structure**

This document is divided in a total of 8 chapters. Firstly, the Laboratory where this work took place in is presented. Afterwards, a more in-depth discussion about the problem and all its context is provided. Following, it presents all the theoretical background and then, the system in study. Finally, the developed activities are commented before presenting the final conclusions.





## 2 THE LABORATORY

With more than 100 years of history, the Werkzeugmaschinenlabor (WZL), translated as Laboratory of Machine Tools and Production Engineering, of the RWTH Aachen University is a research institute globally recognized for its pioneering research, forward-thinking and highly successful innovations in the field of production engineering.

The WZL is focused mainly in four chairs, being these: Production metrology and quality management; Production Engineering; Manufacturing Technology; and, Machine tools. It also retains hundreds of employees and maintain committed with research activities related with fundamental theories, and applications of these findings in industrial context.

Researches carried out on WZL characterized itself by the combination of various disciplines in both a pure research and/or applied development environment as well with the collaboration with numerous industrial companies that provide a rapid application in industrial context.

Figure 1- Production Engineering Cluster at RWTH Aachen Campus



Source: WZL [1]

### 2.1 Chair of Metrology and Quality Management

The research area of Metrology and Quality Management focus primarily in explore problems related with the mastery of production process and management of inspection task as well as, error-free, robust and efficient processes that would provide unique products that appeal to the costumer.

Within the research of these chair are included: the development and optimization of measurement processes and equipment; production-integrated metrology; machine-oriented quality control loops; quality management systems; knowledge, innovation and optimization management; including also, quality management methods and computer aided quality management.

### **2.1.1 Model-based Systems Department**

The Model-based Systems Department is one of three different departments included in the chair of Metrology and Quality Management, and where this work has taken place. Driven with the search to attain success in today's competition, the Model-base Systems Department concern itself with the challenges of control of production processes and the efficient management of inspection tasks.

The department endeavor itself in developing and optimizing modern quality-oriented solutions considering organizational and methodical information-technical aspects while engage in design, advancement and optimization of inspection processes and sensors as well.

This works took place within the Metrology-assisted Assembly Group that deals with the conception, realization and validation of integrated metrologically monitored assembly systems.

### 3 THE PROBLEM

For more than fifty decades the introduction of automated process sought to increase industrial production with efficient costs. Throughout this time, the machinery took the form of robots that enhanced productivity and provided comfort and safety for human operators. As the demand of performance increase, it's necessary to implement adaptive controllers for the robotics manipulators, which have the capability to handle the complex dynamics of these systems. As a result, a mathematical description of the system is needed for the design of these controllers [6].

In parallel, in times of the *Industry 4.0*, the progressive need of “individualization” is changing the production line. Due to the volatility of current markets, it is becoming more important to produce small size lots of variants of one product [7]. In such a context, it's important to possess a flexible manufacturing environment to keep up with the markets' evolution.

At the same time, the main design of technologically challenging assembly process such as the assembly of large-scale components in the automotive industry, took the form of pulsed lines. Aiming to reduce the production time, a continuous flow assembly is preferred. In comparison with stationary assembly points, the production time can be reduce by up to 64%. ([3] *apud*. Prash, 2010).

The highly flexible environment, capable of maintaining the high levels of requirements in these kind of production lines are, however, obtained through a high degree of manual process. In comparison to the other stages of the production process, the final assembly can present a level of automation 450% smaller. These manual assembly processes can be used to compensate for variant-specific differences in production.

In the industrial environment, the tool capable to meet these requirements are the robot manipulators. Industrial robots can be seen as a pretty flexible tool for manufacturing, capable to perform a series of task like the ones stated in [5]: spot welding, arc welding, assembly, handling materials, gluing, painting, cutting, and so on. Yet, this tool are traditionally applied in fixed workstations and therefore, the product have to be transported to the workstation [8]. This transport is not cost and time effective [8].

In this context, in development in WZL throughout a series of works [3, 4, 9, 10], the FASIM project seeks to provide a suitable continuous flow autonomous assembly for large components employing the Nikon's large-volume metrology systems indoors GPS to provide support and guidance.

Through the researches developed in WZL, the synchronization between the product and the handling system required a use of a controller to guarantee a safe environment and the compensation of interferences: in one hand, vibrations of the conveyor system that affect the accuracy during assemble, in the other, the uncertainty and the dead time of the measure system. For these reasons and seeking a robust control, the laboratory chose the approach of Model Predictive Control (MPC) [9].

The control approach selected relies on a prior model of the system, since the quality of the control is based on the future behavior of the system that is predicted through this model. The WLZ have currently available a prior model, but seeks ways to improve it, which would provide better conditions to enhance the overall control synchronization

### 3.1 Main Objective

Considering the above mentioned problem and taking into account what is stated in [9] that, “the model is the core element of the MPC”, the main objective of this works is to provide a suitable model to WZL for its MPC application, researching the viability of applying machine learning algorithms to identify and improve the model in system’s run time. The physical systems subject to the modeling promoted by this work are the same as those discussed in [9]: the robot manipulator and the measurement system used.

### 3.2 Specific Objectives

- Evaluate the viability, and subsequent implementation of machine learning algorithms to identify physical systems;
- Implement online algorithm to improve model in run time;
- Compare the proposed approach with other methods;
- Identify and evaluate suitable implementations tools for the chosen approaches;
- Validate the implementation with the real system.

### 3.3 Problems identified in the current state of the project

The current model provided by WZL to the control research was obtained throughout the *System Identification Toolbox*<sup>TM</sup> as described in [9]. Seeking ways to improve it, it was implemented a PID control approach to estimate the parameters associated with the dynamics of the inputs of the system during the system’s operation.

This strategy showed itself insufficient as it wasn't capable to provide a good estimation for different kinds of inputs to the system, probably, because of the fact that the estimation only updates the input dynamics of the system.



## 4 THEORETICAL BACKGROUND

The following chapter aims to cover the state of the art of the project and the subsequent issues of this work with in it.

### 4.1 Flexible production Systems

In the field of flexible production Systems, KUKA [7], presents an example of its vision for the industry 4.0, the matrix production. The concept of matrix production consists on categorized, standardized production cells. Arranged in a grid layout, with any given number of cells. These cells are equipped with turntables for storing the components, tool holders and the robots that carry out the process.

The components and tools reach each of the cells through Automated Guided Vehicles (AGV). The AGVs can pick up and transport different components or tools via configurable load handling devices.

Logistics process and production are decoupled from one another in matrix production, this allows the production chain to be reconfigurable as demands need it. Cells could easily be integrated or removed in a given process without interrupting the value chain.

Figure 2 - Exemplary of standard cell in matrix production



Source: [7]

### 4.2 Synchronization for autonomous assembly in motion

Synchronization can be given by three principles as described by [9]: mechanical synchronization, conveyor-tracking, or by a feedback control loop.

Mechanical synchronization is the simplest one. Usually done with a rail guide in parallel to the conveyor system where the whole manipulator system could move and track the product. The synchronization is done by physical contact of the conveyor system.

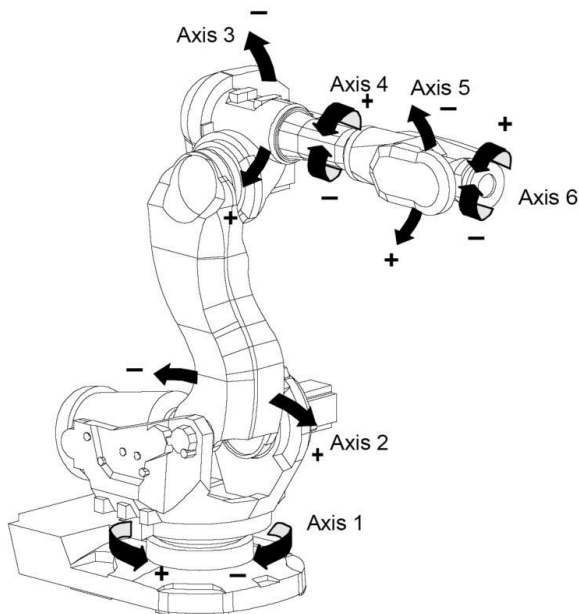
The conveyor-tracking is done by measuring the speed of the conveyor system and feedback into the handling system controller. For heavy and large-scale components, this type can cause deviations of several millimeters due to vibrations of the motion system not compensated.

The feedback control loop relies on the measurement of the actual position values to compensate deviations between the conveyor and handling systems. To achieve this method, however, a sensor that measure the entire system's dynamics is necessary.

### 4.3 Robot manipulators overview

The mechanical structure of a robot manipulator consists of a sequence of rigid bodies connected by rotatory or prismatic joints also called links and axes respectively. Each joint is actuated by an electric motor combined with a gearbox. Prismatic joints give relative translational motion between links while rotatory joints provide rotational motion between them [5].

Figure 3 - 6-DOF Robot Manipulator Scheme



Source: Wernholt [5]



### 4.3.1 Robot dynamics and non-linearities

The idea of this section is to provide an overview of the system and its components, and not to focus on a precise modeling of the whole physical system as presented in [5]. Instead, a summary of the modeling conducted by [5] is presented.

One usual robot manipulator is composed of a series of links, actuators sensors and a robot controller.

#### 4.3.1.1 Actuators

Actuators are the elements responsible to provide motion between the links of the robot, basically composed by a motor attached, or not, to a transmission gearbox, a break for emergency stops and power supply unit. The gearbox, in the same time that provides the high torques and low speeds demanded by the joint, reduces the nonlinear coupling in the dynamic model, with the disadvantage of introducing flexibilities, backlash and friction. [5]

Robotic Manipulators' actuators are mainly electric motors. AC permanent magnet motors are fast, compact and robust. A drawback, however, is that the generated torque changes periodically dependent of the rotor position. Since the torque ripple is periodic in the motor angular position  $q_m$ , it can be modeled as a generic sum of sinusoids like: [5]

$$v_\tau(t) = \sum_{n \in \mathbb{N}_a} a_n \sin(nq_m(t) + \phi_{a,n}) + \tau_c(t) \sum_{n \in \mathbb{N}_b} b_n \sin(nq_m(t) + \phi_{b,n}) \quad 4-1$$

Where  $\mathbb{N}_a$  and  $\mathbb{N}_b$  are associated with the number of components and depend on the specific type of motor.

#### 4.3.1.2 Sensors

Sensors are often divided in proprioceptive sensors or heteroceptive, that measure respectively, the internal state of the robot like or the surrounding environment. Encoders and resolvers used for joint position measure and tachometers for joint velocity measure are examples of proprioceptive sensors. For heteroceptive, sensors for end effector force measure and vision sensors for environment inspection can be cited.[5]

Because of transmission and other sources of flexibilities these components require advanced dynamics models to accurately estimate the robot's movement. Wernholt (*apud*. Hanselman, 1990) exemplify this complexity by modeling the position error of a non-ideal resolver as a sum of sinusoids: [5]

$$v_{qm}(t) = \sum_{n \in \mathbb{N}_c} c_n \sin(nq_m(t) + \phi_{c,n}) \quad 4-2$$

### 4.3.1.3 Kinematics

Kinematics of a robot is the description of the geometric relationship between joints variables and the end effector position and orientation in task space. Usually the motion of the end effector is described in Cartesian coordinates with respect to a reference frame. Frequently, for convenience, the robot kinematics is defined adopting a series of coordinate systems, where each link, the base (reference) and the end effector have its own coordinate frame. [5]

The coordinates of the  $n$  joints can be given by a vector  $q_a$ . One realization of  $q_a$  is named *configuration* of the robot: [5]

$$q_a = (q_{a1}, q_{a2}, \dots, q_{an})^T \quad 4-3$$

The position of the tool frame, or the tool center point (TCP) and its orientation may be determined, respectively, by a vector  $x \in \mathbb{R}^3$  and a rotation matrix  $R \in \mathbb{R}^{3 \times 3}$ . [5]

#### 4.3.1.3.1 Position Kinematics

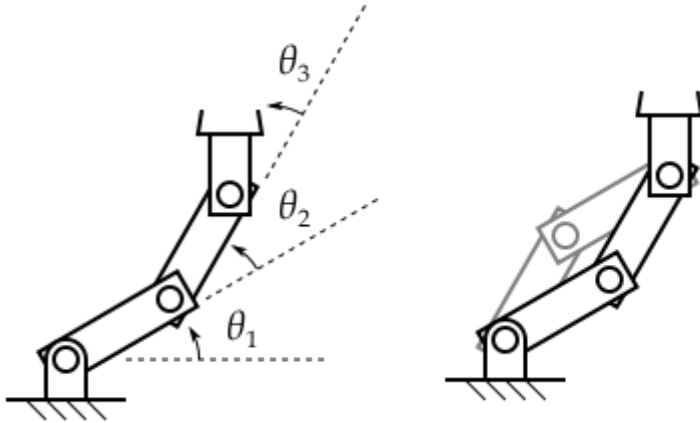
The positioning kinematics can be divided in two problems, the *forward kinematics* and *inverse kinematics*. The first one is to determine the following mapping from the joint space to task space with  $X$  being named the robot *location*:

$$X = \begin{bmatrix} x(q_a) \\ \gamma(q_a) \end{bmatrix} = f_{kin}(q_a) \quad 4-4$$

In other words, the *forward kinematics* problem refers to calculate the position of the TCP in Cartesian coordinates given in the joints' angles. The determination of this function is straightforward in a serial link robot, for example, and it can be done iteratively from the base frame to the first link and then sequentially until the TCP, determining the relation by geometric properties of the links and a single joint variable. [5]

The *inverse kinematics*, on the other hand is to determine the inverse of the mapping in 4-4, i.e. determinate the corresponding joint configuration given a position and rotation of the TCP. In general, this case is a much harder problem as it is not always possible to find a closed-form solution and it may exist multiple or infinitely solutions.

Figure 4 -Forward kinematics (left) and inverse kinematics examples scheme



Source: [11]

#### 4.3.1.3.2 Velocity Kinematics

*Velocity kinematics* express the relationship between joint velocities and the end effector linear and angular velocities and can be written as: [5]

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix} = J(q_a) \dot{q}_a \quad 4-5$$

Where,

$$J(q_a) = \frac{\partial f_{kin}}{\partial q_a}(q_a) \in \mathbb{R}^{6 \times n} \quad 4-6$$

is the manipulator Jacobian,  $V$  represents the linear and angular velocities of the tool frame in respect to the base frame. The Jacobian is a key element in the analysis and control of the robots' motion. As it is a function of the *configuration* of the robot, those configurations, in which it loses rank are called *singularities* and can be interpreted as points in the task space where the robot manipulator loses one or more degree of freedom. [5]

#### 4.3.1.4 Rigid Body Dynamics

The dynamic model of a robot describes the evolution of the robot's joints position in time as a function of the applied torques and forces. The identification of such model is the main focus of the work in [5] and in its simple form presented as:

$$M_a(q_a) \ddot{q}_a + c_a(q_a, \dot{q}_a) + g_a(q_a) + \tau_{fa}(\dot{q}_a) = \tau_a \quad 4-7$$

Where:  $M_a(q_a)$  is the inertia matrix;  $c_a(q_a, \dot{q}_a)$  is the velocity dependent term, which contains the centrifugal and Coriolis effects;  $g_a(q_a)$  is the gravitational term;  $\tau_{fa}(\dot{q}_a)$  the friction torque, and;  $\tau_a$  is the applied torque. [5]

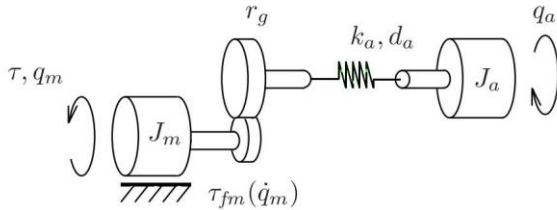
#### 4.3.1.5 Flexible Body Dynamics

Another approach for dynamic model used for control aside from the entirely Rigid Body Dynamics model is to consider flexible joint models as well, *i.e.* elastic gear transmissions and rigid links ([5] *apud.* Albu-Schäffer & Hirzinger, 2000, Spong, 1987).

Besides, because of the trend of using lighter robots that could perform the same tasks while promoting cost and power reduction and a safer environment (reduced mass of moving parts), an elastic model is also convenient. Lighter robots result in a weaker and more complex mechanical structure because of the enhanced elastics effects of the materials. Wernholt [5] presents a simple overview of the flexible joint models and elastics models.

Considering a single joint, the flexible joint model results in a two-mass flexible model as shown in Figure 5.  $J_m$  and  $J_a$  are respectively the moments of inertia of the motor and the arm,  $r_g$  the (inverse) gear ration,  $k_g$  the spring stiffness,  $d_g$  the spring damping,  $\tau_{fm}(\dot{q}_m)$  the motor friction and finally,  $\tau$  is the motor torque.

Figure 5 - Two-mass flexible joint model for robot arm



Source: [5]

The equations that describe the dynamics are listed below:

$$\begin{aligned} J_m \ddot{q}_m + \tau_{fm}(\dot{q}_m) + r_g \tau_g &= \tau \\ J_a \ddot{q}_a &= \tau_g \\ k_g (r_g q_m - q_a) + d_g (r_g \dot{q}_m - \dot{q}_a) &= \tau_g \end{aligned} \quad 4-8$$

The combination of the rigid body dynamics with the generalized two-mass flexible model is then given by:

$$M_m \ddot{q}_m + \tau_{fm}(\dot{q}_m) + r_g \tau_g = \tau \quad 4-9$$

$$\begin{aligned}
M_a(q_a)\ddot{q}_a + c_a(q_a, \dot{q}_a) + g_a(q_a) &= \tau_g \\
k_g(r_g q_m - q_a) + d_g(r_g \dot{q}_m - \dot{q}_a) &= \tau_g
\end{aligned}$$

where  $\tau$  is now a vector of applied torques,  $q_a$  is the vector of arm joint variables, and  $q_m$  the vector of motor joint variables.

As for the elastics' models, Wernholt [5] considers the *lumped parameters* approach base on Khalil and Gautier's work (2000), by considering that each link is divided into a number of rigid bodies connected by spring-damper pairs. The flexible joint model 4-9 is then extended giving the following:

$$\begin{aligned}
M_m \ddot{q}_m + \tau_{fm}(\dot{q}_m) + r_g \tau_g &= \tau \\
M_{ae}(q_a, q_e) \begin{bmatrix} \ddot{q}_a \\ \ddot{q}_e \end{bmatrix} + c_{ae}(q_a, q_e, \dot{q}_a, \dot{q}_e) + g_{ae}(q_a, q_{ee}) &= \begin{bmatrix} \tau_g \\ \tau_e \end{bmatrix} \\
k_g(r_g q_m - q_a) + d_g(r_g \dot{q}_m - \dot{q}_a) &= \tau_g \\
-k_e q_e - d_g \dot{q}_e &= \tau_e
\end{aligned} \tag{4-10}$$

The additional variable  $q_e$  describe the angular motion between the rigid bodies due to elastic effects,  $k_e$  and  $d_e$  took a similar definition as  $k_g$  and  $d_g$ .

The above equation represents a simple but a profound overview of the modeling of a robot manipulator, it could be adapted depending of its degree of freedom and could also be transformed in a nonlinear state-space model ideal for some kind of control approaches.

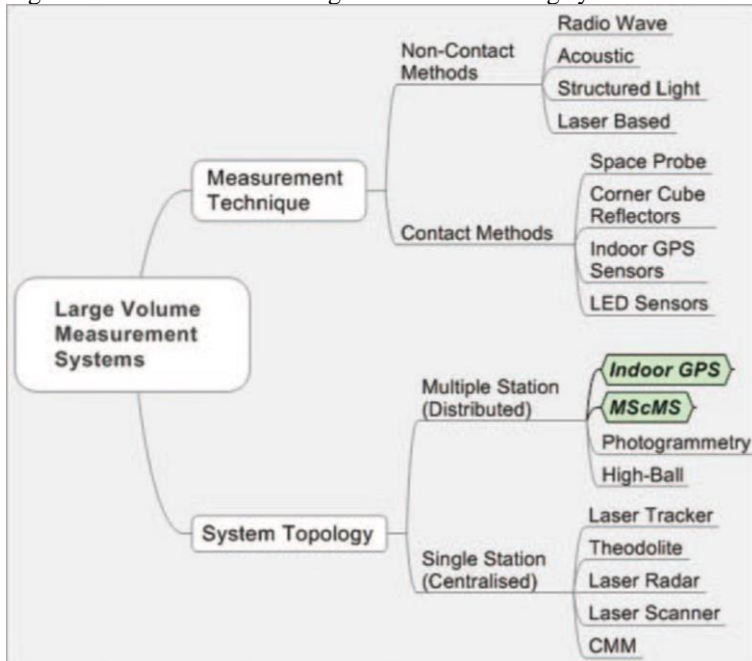
#### 4.4 Global Reference Systems in production

Global Reference System (GRS) is similar to the well know Global Positioning System (GPS). [3, 9] In a production context is a common coordinate system to all present elements. A GRS can also provide a reference system to both simulation and production for which compensation for the deviations could be applied. [3] In other words, a GRS can provide an object's location in the production line through time. A GRS can be established through the application of Large-volume metrology systems.

#### 4.5 Large-volume metrology

Large-volume metrology deals with the measurement of large machines and structures, in which the linear dimensions range from tens to hundreds of meters. ([12] *apud*. Puttock, 1978). The figure below can summarize the measuring systems for large volumes.

Figure 6 - Classification of large-volume measuring systems



Source: [12]

Maisano *et al.* [12] classifies the large-volume metrology systems in two categories: *distributed* and *centralized* systems. Centralized systems (*e.g.* Laser Tracker) are basically a single unit that work independently to measure a spatial co-ordinate of a point on a object's surface. Distributed systems, like iGPS, differently, need a series of measure units working cooperatively to determine the co-ordinates of a point in an object's geometry and, although, generally, an individual unit is not capable of ascertain the position of a point, their light weighted peripheral and portable devices can easily be moved to the point of interest.

Figure 7 - Examples of large-volume metrology applications



Source: [13]

There is an increasing interest in accurate measurement of three-dimensional co-ordinates on industries such as aircraft and ship construction. [12] These systems are capable of keep tracks of, for example, an aircraft fuselage and its wings and the tools responsible to perform an autonomous assembly. [13]

### 4.5.1 Laser based non-contact measurement methods

The above section presents briefly two laser based non-contact large-scale measurement systems: iGPS and laser tracker.

Laser tracker works utilizing the principle interferometry. It generates a coherent laser beam of known wavelength and passes it through a beam-splitter. One beam is reflected back within the system, the other is aimed at a Spherical Mirror Reflector (SMR). Constructive and destructive interference at the laser wavelength can be observed by the detector and thus the distance from the SMR could be determined. [14]

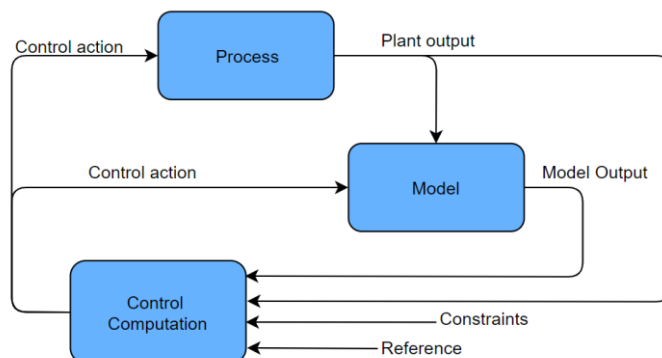
Laser trackers are considered one of the most reliable, well established metrology systems. Their main disadvantage is that a line of sight from the laser tracker and the SMR must be clear at all time, and that only one point of interest can be tracked at a time.

The iGPS from Nikon (see 5.2 for details) uses laser transmitters that create a measurement field that encompasses a field as large as the room facility. [13] Laser detectors scattered through the working determine its position through a GRS prior established by the known position of the transmitters.

## 4.6 MPC's Model

A Model Predictive Controller (MPC) does not correspond to any specific control strategy but a range of methods that make explicit use of a model of the process to obtain the control signal by minimizing an objective function. Each approach will diverge itself from other by the model used and the cost function to be minimized. [15]

Figure 8 – Basic structure of MPC



Source: [15]

The above figure represents a basic structure of an MPC algorithm and all its common elements. The process model plays a decisive role in the controller because it is necessary to predict the future outputs of the system. [15]

The basic operation of an MPC requires the observation of the futures outputs of the system in a certain time frame, this predicted observations are then provided by the controller's model.

#### 4.7 Dynamic system in state space

The behavior of linear (time-invariant) systems over time can be described throughout differential equations. Over the frequency domain, it could be described as a transfer function  $G(s)$  that specify the relation between the input  $U(s)$  and output  $Y(s)$ . [9]

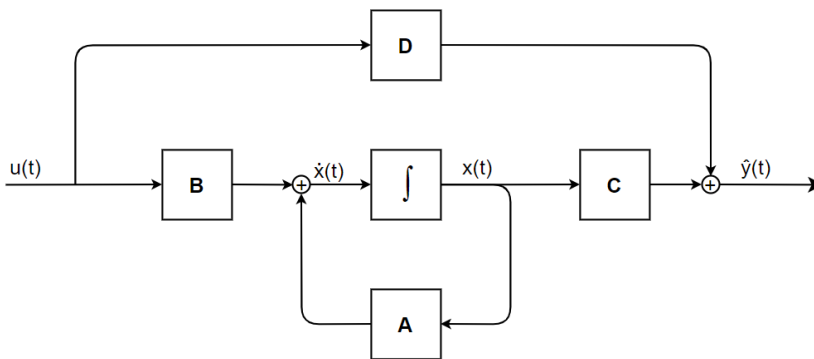
$$s^n Y(s) + a_{n-1} s^{n-1} Y(s) + \dots + a_1 s Y(s) + a_0 Y(s) = b_m s^m U(s) + \dots + b_0 U(s) \quad 4-11$$

$$G(s) = \frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} \quad 4-12$$

Unfortunately, this kind of representation do not provide any kind of information regarding the internal dynamics of the system operating between the input and output. One representation that takes these dynamics in account is the state-space representation. [9]

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = C^T x(t) + Du(t) \end{cases} \quad 4-13$$

Figure 9 - State-space representation



Source: author

#### 4.8 Machine Learning

Machine Learning is a subfield of computer science [16] and Artificial Intelligence [17]. Often referred as predictive analytics or predictive modeling, its goal is to build and/or enhance existing algorithms that, learning from data, could provide generalizable models that give accurate predictions. [16] Common Machine Learning applications involve natural



language processing, recommendation systems, univariate and multivariate regression and many others.

From a systems identification point of view, regressions problems are the most appropriate case. Regression models are characterized by its ability to assign a continuous output to an observed data. [16] That means that a regression machine learning model could predict a given system's output or identify the change in its parameters in time.

Regression models are adjusted only by supervised learning, that is, the algorithm is trained with already correct labeled data. [16, 18] Some algorithms that could be mentioned in this class are linear regression, decision tree or forest regression, random forest, neural networks, and many others.

## 4.9 System Identification

The main problem in system identification is to select a suitable structure in which a good model could be found, the later problem, that is, estimating the parameters for the given structure tend to be a lesser problem. It's common knowledge that one should prioritize prior knowledge and physical insights about the system when selecting such structures. [19]

As presented in [19], the basics structures can be divided in tree levels of prior knowledge: White Box Models, Grey Box Models, Black Box Models, each representing respectively: a model constructed from full knowledge of the system; a model with some physical insight with some parameters determined from observed data; and, a model with no physical knowledge at all.

Linear black-box models are tasked to approximate the system's frequency response, Sjöberg *et al.* [19] define that as a "modest approximation problem". For nonlinear black-box models however, the problem is more complicated. The main reason is that nothing can be excluded, and it is necessary to handle a large spectrum of possible models' description.

### 4.9.1 The identification problem

Given the inputs  $u(t)$  and outputs  $y(t)$ , from a dynamical system:

$$u^t = [u(1), u(2), \dots u(t)] \quad 4-14$$

$$y^t = [y(1), y(2), \dots y(t)] \quad 4-15$$

The problem of identification is defined by [19] as to find a relationship between past data  $[u^{t-1}, y^{t-1}]$  and the future output  $y(t)$ :

$$y(t) = g(u^{t-1}, y^{t-1}) + v(t) \quad 4-16$$

The term  $v(t)$  is added because of the fact that the output  $y(t)$  will not be an exact function of past data, though, the objective is that  $v(t)$  would be small enough so that the function  $g(u^{t-1}, y^{t-1})$  would be considered a good approximation of  $y(t)$ .

To find the function  $g$  in 4-16 it is necessary to approximate it by parameterizing  $g$  by a finite dimensional vector  $\theta$ :

$$g(u^{t-1}, y^{t-1}, \theta) \quad 4-17$$

Chosen a good structure and collected the data set, the quality of the parameters vector  $\theta$  could be measured through the fitting between the model and the data recorded, using the norm for example:

$$\sum_{t=1}^N \|y(t) - g(u^{t-1}, y^{t-1}, \theta)\|^2 \quad 4-18$$

Because of the generality of 4-17, usually, it is convenient to write  $g$  as a concatenation of two mappings: one that takes the past data  $u^t$  and  $y^t$  and maps them into a vector  $\varphi(t)$  with finite dimensions; one that takes  $\varphi$  to the space of outputs:

$$g(u^{t-1}, y^{t-1}, \theta) = g(\varphi(t), \theta) \quad 4-19$$

where vector  $\varphi$  is defined as:

$$\varphi(t) = \varphi(u^{t-1}, y^{t-1}) \quad 4-20$$

The vector defined in 4-20 is called *regression vector*.

Summarizing, the problem of identification as two partial problems:

1. Choose the regression vector  $\varphi(t)$ ;
2. Choose the mapping  $g(t)$  from the regressor space to the output space

## 4.9.2 Regression Possibilities

### 4.9.2.1 Linear case

The simplest dynamical model addressed by [19] is the Finite Impulse Response model:

$$\begin{aligned} y(t) &= B(q)u(t) \\ &= b_1u(t-1) + \dots + b_nu(t-n) + e(t) \end{aligned} \quad 4-21$$

Where,  $B(q)$  is a polynomial in  $q^{-1}$  and  $q$  denote the shift operator. The noise term is modeled as  $e(t)$  [19].

The predictor for the above model is:

$$\hat{y}(t|\theta) = B(q)u(t) \quad 4-22$$

And thus, based on the following regressor:

$$\varphi(t) = [u(t-1), u(t-2), \dots, u(t-n)] \quad 4-23$$

With  $n$  tending to infinity, it is possible to describe a wide range of linear systems dynamics [19].

In practice, the most common linear black-box structures are variations of 4-21 with different ways to define the poles of the system and describe the noise characteristics, but can all be summarized by the following (Ljung, 1987 *apud* [19]):

$$A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t) \quad 4-24$$

The known special cases are:

1. *ARX*:  $C(q) = D(q) = F(q) = 1$ ;
2. *ARMAX*:  $D(q) = F(q) = 1$ ;
3. *Box-Jenkins* (BJ):  $A(q) = 1$ ;
4. *Output-Error* (OE):  $A(q) = C(q) = D(q) = 1$

The regressors are usually given by:

1.  $u(t-k)$ : associated with the  $B$ -polynomial;
2.  $y(t-k)$ : associated with the  $A$ -polynomial;

3.  $\hat{y}_u(t - k | \theta)$ : simulated outputs from past inputs and associated with the F-polynomial;
4.  $\varepsilon(t - k) = y(t - k) - \hat{y}(t - k | \theta)$ : Prediction errors and associated with the C-polynomial;
5.  $\varepsilon_u(t - k) = y(t - k) - \hat{y}_u(t - k | \theta)$ : Simulation errors, associated with the D-polynomial;

#### 4.9.2.2 Nonlinear case

The nonlinear cases follow similar structures of the linear ones. In the work of Sjöberg *et al.* [19] it is adopted the following:

$$\hat{y}(t|\theta) = g(\varphi(t), \theta) \quad 4-25$$

Where  $g$  corresponds to a nonlinear function parameterized by  $\theta$ . The components of  $\varphi(t)$  are similar to the ones described to the linear case:

Table 1 - Nonlinear models' regressors

Structures	Regressors
NFIR-models	$u(t - k)$
NARX-models	$u(t - k)$ and $y(t - k)$
NOE-models	$u(t - k)$ and $\hat{y}_u(t - k   \theta)$
NARMAX-models	$u(t - k)$ , $y(t - k)$ and $\varepsilon(t - k   \theta)$
NBJ-models	$u(t - k)$ , $\hat{y}(t - k   \theta)$ , $\varepsilon(t - k   \theta)$ and $\varepsilon_u(t - k   \theta)$
Non-linear state space	<i>e.g.</i> : virtual outputs, internal

Source: [19]

#### 4.9.3 Nonlinear Mappings

Sjöberg *et al* [19] presents the nonlinear mapping as:

$$g(\varphi, \theta) \quad 4-26$$

Where for any given  $\theta$ , goes from  $R^d$  to  $R^p$  with  $\varphi \in R^d$ .

The function  $g$  is then, parameterized as a function expansion as follows [20]:

$$g(\varphi, \theta) = \sum_{k=1}^n a_k g_k(\varphi) \quad \theta = [a_1, \dots, a_n]^T \quad 4-27$$

$g_k$  is referenced as *basis functions* because of its similarity to a functional space basis. The equation 4-27, together with the regression vector summarize most known nonlinear black-box model structures, where  $g_k$  is obtained by parameterizing a single “mother basis function” usually denoted by  $\kappa(x)$  [19].

$$g_k(\varphi) = g_k(\varphi, \beta_k, \gamma_k) = \kappa(\beta_k(\varphi - \gamma_k)) \quad 4-28$$

With  $\beta_k$  and  $\gamma_k$  representing a dilation parameter and a translation parameter respectively. Both works [19, 20], state that the equation above should be interpreted symbolically and specify it through out a series of examples. The most interest one for the sake of this works is the *sigmoid* function:

$$\kappa(x) = \sigma(x) = \frac{1}{1 + e^x} \quad 4-29$$

For a multidimensional case in other words, for  $g_k$  as a function of several variables, in most black-box models cases, their structure is constructed from a single-variable function  $\kappa$ . One simple case is Ridge construction [19, 20]:

$$g_k(\varphi) = g_k(\varphi, \beta_k, \gamma_k) = \kappa(\beta_k^T \varphi + \gamma_k), \varphi \in R^d \quad 4-30$$

The ridge function bases itself in the idea of letting its value depend only on the distance of  $\varphi$  to a given hyperplane, hence, for all  $\varphi$  in the hyperplane its value is constant ( $\varphi \in R^d : \beta_k^T \varphi = \text{constant}$ ). The consequence is that if the mother basis function  $\kappa$  has local support, the basis function  $g_k$  will have unbounded support in this subspace. [20]

Thus, for the multidimensional case, the resulting model becomes:

$$g(\varphi, \theta) = \sum_{k=1}^n a_k \kappa(\beta_k(\varphi - \gamma_k)) \quad 4-31$$

The work of Sjöberg *et al.* [19] in addition, correlates the basics of the non-linear mapping to some known named structures as the followings:

### 4.9.3.1 Sigmoid Neural Networks

The combination of the model expansion 4-27, with the sigmoid choice for mother function 4-29, and with a ridge construction method 4-30, provide the known *one hidden layer feed-forward sigmoid neural net*. [19, 20]

### 4.9.3.2 Multi-layer Networks

A network-like structure is more noticeable when the basic mappings are convoluted with each other. Let the output of the basis function be:

$$\varphi_k^{(2)}(t) = g_k(\varphi(t)) = \kappa(\varphi(t), \beta_k, \gamma_k) \tag{4-32}$$

Group them into a vector:

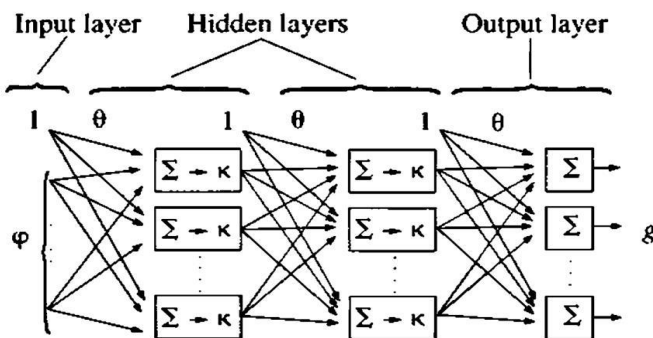
$$\varphi^{(2)}(t) = [\varphi_1^{(2)}(t), \dots, \varphi_n^{(2)}(t)] \tag{4-33}$$

Finally, instead of considering the linear combination  $\varphi^{(2)}$  as the output of the model, it is possible to treat it as new regressor inserting it into another layer of basis functions forming a second expansion.

$$g(\varphi, \theta) = \sum_l \alpha_l^{(2)} \kappa(\varphi^{(2)}, \beta_l^{(2)}, \gamma_l^{(2)}) \tag{4-34}$$

Where  $\theta$  represents the whole collection of involved parameters.

Figure 10 - Feedforward network with two hidden layers



Source: [20]

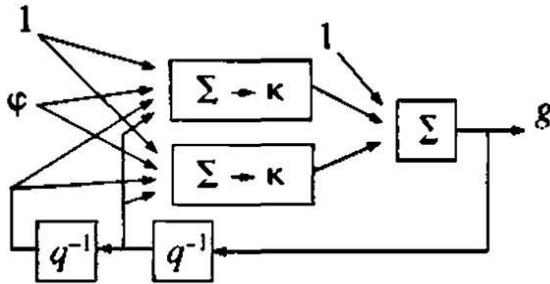
### 4.9.3.3 Recurrent Networks

Recurrent Networks states for networks structures in which some of the regressors used at time  $t$  are previous outputs from the model. [20]

$$\varphi_k(t) = g(\varphi(t - k), \theta)$$

4-35

Figure 11 - Recurrent Network scheme



Source: [20]

#### 4.10 Linear regression for parameter estimation and the least-squares method

The parameter estimation problem can be briefed as: given a chosen model structure  $\mathcal{M}$ , the objective is to find a model  $\mathcal{M}(\theta)$ , with the parameters  $\theta$ , that would provide a small prediction error  $\varepsilon$  in a certain data set  $Z^N$ , where  $N$  represents the number of elements of the data set [20]. In this context, the problem consists in minimizing the prediction error.

Lennart [20] describes the linear regression applied to linear model structures employing the following predictor:

$$\hat{y}(t|\theta) = \varphi^T(t)\theta + \mu(t) \quad 4-36$$

In the above case,  $\mu$  is a known dependent vector, but for simplicity is taken as null, the prediction error is then given by:

$$\varepsilon(t, \theta) = y(t) - \varphi^T(t)\theta \quad 4-37$$

Given this problem, the *least squares criterion* for the linear regression in 4-36 is defined by:

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \frac{1}{2} [y(t) - \varphi^T(t)\theta]^2 \quad 4-38$$

This criterion is deduced as an especial case norm that, for a given  $Z^N$ , is a well-defined scalar-value function of the parameters  $\theta$  and therefore a natural measure of the validity of the model  $\mathcal{M}(\theta)$ . The estimation of the parameters can be defined then as the following minimization:

$$\hat{\theta}_N = \hat{\theta}_N(Z^N) = \arg \min V_N(\theta, Z^N) \quad 4-39$$

$$\hat{\theta}_N^{LS} = \left[ \frac{1}{N} \sum_{t=1}^N \varphi(t) \varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^N \varphi(t) y(t) \quad 4-40$$

In 4-39, “arg mim” stands for “the minimizing argument of the function”. The 4-40 represents the specific representation for the *least-squares* case.

#### 4.10.1 Weighted least-squares criterion

The Least-Squares (LS) can also assume a weighted form from the concept of time-varying Norms. This allows the LS to associate different weights to each measure of the system in regard to its reliability. For example, a measure with a high noise component can be given a lower weight.

$$V_N(\theta, Z^N) = \frac{1}{N} \sum_{t=1}^N \beta(N, t) [y(t) - \varphi^T(t)\theta]^2 \quad 4-41$$

The weighted version of the LS is presented above (4-41) with the weighting being determined by the weighting function  $\beta(N, t)$ .

Rewriting the estimation expression 4-40 for the weighted case:

$$\hat{\theta}_N^{LS} = \left[ \frac{1}{N} \sum_{t=1}^N \beta(N, t) \varphi(t) \varphi^T(t) \right]^{-1} \frac{1}{N} \sum_{t=1}^N \beta(N, t) \varphi(t) y(t) \quad 4-42$$

According to Lennart [20] the LS method presents a series of advantages, being the most notable one the fact that a global minimum can be found efficiently and unambiguously (without local minima other than the global exists).

#### 4.11 Recursive Estimation Methods

The identification techniques that provides an online computation of the model in a way that the processing of measurements from one sample can, with certainty be completed during one sampling interval are called *recursive identification methods*. [20] Ljung [20] states that this methods are “quite competitive” alternatives for parameters estimation as in off-line situations.



### 4.11.1 Recursive Algorithms for Online Estimation

A general identification problem can be defined as a mapping from the data set  $Z^t$  to the parameter space, as stated in [20]:

$$\hat{\theta}_t = F(t, Z^t) \quad 4-43$$

The above general expression is not suitable for a recursive algorithm as the evaluation of  $F$  cannot be guaranteed at the next sample time, thus, the following format is more suitable: [20]

$$\begin{aligned} X(t) &= H(t, X(t-1), y(t), u(t)) \\ \hat{\theta}_t &= h(X(t)) \end{aligned} \quad 4-44$$

Where  $X(t)$  is a vector with fixed dimensions that holds some information of the system. In this case,  $H$  and  $h$  are expressions that can be evaluated with a fixed amount of calculations. Considering that the system's information usually consists of the input  $u(t)$  and output  $y(t)$  the expression can be rearranged as: [20]

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + \gamma_1 Q_\theta(X(t), y(t), u(t)) \\ X(t) &= X(t-1) + \mu_t Q_X(X(t-1), y(t), u(t)) \end{aligned} \quad 4-45$$

Lennart [20] takes the least-squares method as a simple archetypal case to deduce other forms of algorithms. Considering the weighted LS criterion presented in both 4-41 and 4-42, the estimated parameters in one time step can be given by:

$$\hat{\theta}(t) = \arg \min \sum_{k=1}^t \beta(t, k) [y(k) - \varphi^T(k)\theta]^2 \quad 4-46$$

Restructuring, we have:

$$\hat{\theta}(t) = \bar{R}^{-1}(t) f(t) \quad 4-47$$

$$\bar{R}(t) = \sum_{k=1}^t \beta(t, k) \varphi(k) \varphi^T(k) \quad 4-48$$

$$f(t) = \sum_{k=1}^t \beta(t, k) \varphi(k) y(k) \quad 4-49$$

The solution of 4-47, at a given time  $t$ , provides the parameters of the system from a data set  $Z^t$ . At first sight there is no explicit relation between the  $\hat{\theta}(t)$  and a past value  $\hat{\theta}(t-1)$ . Lennart [20] deduce a relation as follows, supposing that the weighting sequence has the following property:

$$\begin{aligned} \beta(t, k) &= \lambda(t) \beta(t-1, k), & 0 \leq k \leq t-1 \\ \beta(t, t) &= 1 \end{aligned} \quad 4-50$$

This can be rewritten as:

$$\beta(t, k) = \prod_{j=k+1}^t \lambda(j) \quad 4-51$$

Resulting in:

$$\bar{R}(t) = \lambda(t) \bar{R}(t-1) + \varphi(t) \varphi^T(t) \quad 4-52$$

$$f(t) = \lambda(t) f(t-1) + \varphi(t) y(t) \quad 4-53$$

Applying the two above equations on 4-47 the following could be reached:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + \bar{R}^{-1}(t) \varphi(t) [y(t) - \varphi^T(t) \hat{\theta}(t-1)] \quad 4-54$$

$$\bar{R}(t) = \lambda(t) \bar{R}(t-1) + \varphi(t) \varphi^T(t) \quad 4-55$$

The two above equations characterize a recursive algorithm that complies with the 4-44 requirement where, the parameters are given by a function of  $X(t)$ , a finite-dimension vector

Aiming to avoid inverting  $\bar{R}(t)$ , Lennart [20] conveniently assumes:

$$P(t) = \bar{R}^{-1}(t) \quad 4-56$$

And utilizes the matrix inversion lemma:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[DA^{-1}B + C^{-1}]^{-1}DA^{-1} \quad 4-57$$

Assuming:

$$A = \lambda(t)\bar{R}(t-1) \quad 4-58$$

$$B = D^T = \varphi(t) \quad 4-59$$

$$C = 1 \quad 4-60$$

Finally resulting into the summarized algorithm:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)[y(t) - \varphi^T(t)\hat{\theta}(t-1)] \quad 4-61$$

$$L(t) = \frac{P(t-1)\varphi(t)}{\lambda(t) + \varphi^T(t)P(t-1)\varphi(t)} \quad 4-62$$

$$P(t) = \frac{1}{\lambda(t)} \left( P(t-1) - \frac{P(t-1)\varphi(t)\varphi^T(t)P(t-1)}{\lambda + \varphi^T(t)P(t-1)\varphi(t)} \right) \quad 4-63$$

#### 4.11.2 Matlab's implementation of recursive algorithms for online estimation

The general form of a recursive estimator adopted in [21] are defined below:

$$\hat{\theta} = \hat{\theta}(t-1) + K(t)(y(t) - \hat{y}(t)) \quad 4-64$$

Where, for recall,  $\hat{\theta}$  is the estimated parameters at the time  $t$ ,  $y(t)$  and  $\hat{y}(t)$  are respectively, the observed output, and the prediction of the system at time  $t$ . The gain  $K$ , corresponds to the proportion in which the current prediction error  $y(t) - \hat{y}(t)$  affects the update of the parameter estimation. The focus of the algorithm is to minimize de error  $y(t) - \hat{y}(t)$ .

The gain  $K$  in the other hand is defined with the following form:

$$K(t) = Q(t)\Psi(t) \quad 4-65$$

The different recursive algorithms available in [21] differ on the choosing form of  $Q(t)$  and the computing of  $\Psi(t)$  that, represents the gradient of the predicted output  $\hat{y}(t | \theta)$  with respect to the parameter  $\theta$ . In comparison with Lennart's [20] deduction, these algorithms diverge themselves on the function  $Q_\theta$  in 4-45.

The role of  $\Psi(t)$  is better described by [21] using a linear regression form:

$$y(t) = \Psi^T(t)\theta_0(t) + e(t) \quad 4-66$$

In 4-66,  $\Psi(t)$  correspond to the *regression vector* which is calculated based on previous values of inputs and outputs;  $\theta_0(t)$  is the true parameters;  $e(t)$  is the noise source, that is assumed to be white noise. The predicted output could be given then by:

$$\hat{y}(t) = \Psi^T(t)\hat{\theta}(t-1) \quad 4-67$$

The exact form of  $\Psi(t)$  will depend of the structure of the polynomial model.

### 4.11.3 Forgetting Factor

Forgetting Factor is one kind of recursive estimation algorithm and as stated in [21] can be summarized by the equations 4-64, 4-65, 4-67 with the following additions:

$$Q(t) = \frac{P(t-1)}{\lambda + \Psi^T(t)P(t-1)\Psi(t)} \quad 4-68$$

$$P(t) = \frac{1}{\lambda} \left( P(t-1) - \frac{P(t-1)\Psi(t)\Psi^T(t)P(t-1)}{\lambda + \Psi^T(t)P(t-1)\Psi(t)} \right) \quad 4-69$$

$Q(t)$  is then obtained by minimizing the following function at time  $t$ :

$$\sum_{k=1}^t \lambda^{t-k} (y(k) - \hat{y}(k))^2 \quad 4-70$$

The forgetting factor method discount old measurements exponentially in a way that an observation  $\tau$  samples old have a weight equal to  $\lambda^\tau$  times the weight of the most recent one.  $\lambda$  is called the forgetting factor and  $\tau = 1/(1 - \lambda)$  represents the *memory horizon* of the algorithm.

### 4.11.4 Kalman Filter

The second recursive estimation is, as well, defined by the equations 4-64, 4-65, 4-67

$$Q(t) = \frac{P(t-1)}{R_2 + \Psi^T(t)P(t-1)\Psi(t)} \quad 4-71$$

$$P(t) = P(t-1) + R_1 - \frac{P(t-1)\Psi(t)\Psi^T(t)P(t-1)}{R_2 + \Psi^T(t)P(t-1)\Psi(t)} \quad 4-72$$

This algorithm assumes the linear-regression form given by 4-66, and  $Q(t)$  is calculated using a Kalman Filter.

**4.11.5 Normalized and Unnormalized Gradient**

In the linear regression case, the gradient methods are also known as the *least mean squares* (LMS) methods. [21] This algorithm bases itself on 4-64, 4-65, 4-67 as well.

The unnormalized gradient has  $Q(t)$  given by:

$$Q(t) = \gamma^* \Psi(t) \tag{4-73}$$

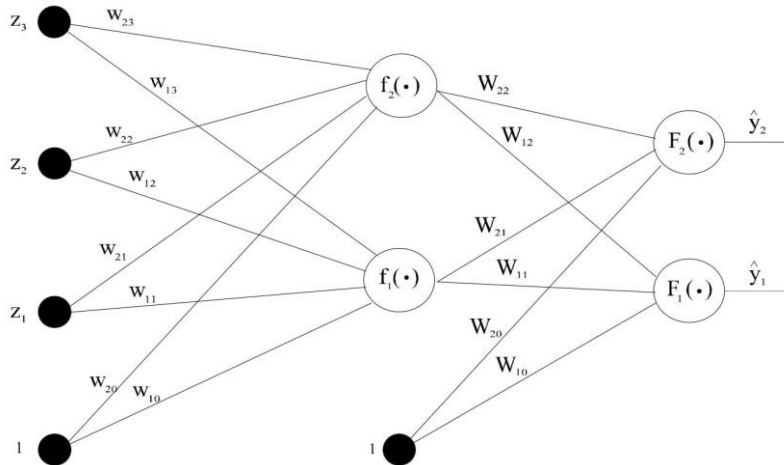
The normalized gradient on the other hand:

$$Q(t) = \frac{\gamma^* \Psi(t)}{|\Psi(t)|^2} \tag{4-74}$$

**4.12 Multilayer Perceptron (MLP) - Backpropagation network for regression**

A multilayer perceptron network has the following scheme:

Figure 12 - Multilayer Perceptron network scheme



Source: [22]

Drifting from the principles of the multivariable mapping (4-31), for a convoluted in layers case, the output of an MLP is given by:

$$\hat{y}_i(w, W) = F_i \left( \sum_{j=1}^q W_{ih} h_j(w) + W_{i0} \right) = F_i \left( \sum_{j=1}^q W_{ij} f_j \left( \sum_{l=1}^m w_{jl} z_l + w_{j0} \right) + W_{i0} \right) \tag{4-75}$$

The above equation states that the output of an MLP is a function of its inputs  $z$  and the weights  $w$  and  $W$ . Neural networks used for system identification (and other cases) relies on supervised training, in other words, the weights of a neural network are adjusted by applying a set of labeled training samples. [23]

Several works address the system identification problem using a neural network structure. Qin *et al.* [24] compare four different neural network structure to identify dynamics systems. Chen *et al.* [25] propose a use of a NARMAX model structure identified using a neural network. Nørgaard [22] implements a whole system identification toolbox using neural networks. Ljung [20] stated that that: “one hidden layer is usually sufficient for modeling most types of systems”. The idea remains the same, determinate a map from the training data set  $Z^N$ , to the set of possible parameters  $\hat{\theta}$ , in this case, the weights of the MLP.

One of the most common algorithm to update the weights of the neural network is the backpropagation algorithm. Used in Narendra *et al.* [26] and Nørgaard [22] the backpropagation algorithm is a simple application of the gradient of an error function.

Suppose for example an error function giver by the mean square error:

$$V_N(\theta, Z^N) = \frac{1}{2n} \sum_{t=1}^N [y(t) - \hat{y}(t|\theta)]^2 \quad 4-76$$

The weights are then selected by:

$$\hat{\theta} = \arg_{\theta} \min V_N(\theta, Z^N) \quad 4-77$$

Nørgaard [22] simply describe this process with the following iterative scheme:

$$\hat{\theta}^{(i+1)} = \hat{\theta}^{(i)} + \mu^{(i)} f^{(i)} \quad 4-78$$

Where  $\hat{\theta}^{(i)}$  specifies the current parameters in the iteration,  $f$  is the search direction and  $\mu$  the step size. The direction can be determined by the gradient of the function  $V_N(\theta, Z^N)$ . For a single output network, the gradient is given by the partial derivatives of the error function with respect of each weight of the network.

$$\text{grad } V_N(\theta, Z^N) = \left\langle \frac{\partial V_N(\theta, Z^N)}{\partial w_{11}}, \dots, \frac{\partial V_N(\theta, Z^N)}{\partial w_{qm}}, \frac{\partial V_N(\theta, Z^N)}{\partial W_{11}}, \dots, \frac{\partial V_N(\theta, Z^N)}{\partial W_{qm}} \right\rangle \quad 4-79$$

### 4.13 Optimization and Genetic Algorithm

Arora [27] defines the problem of systems design as an *problem optimization*, stating that “several systems can usually accomplish the same task”. Also is stated that, any problem in which certain parameters need to be determined to satisfy some constraints can be seen as an optimization problem. In other words, an optimization problem is the search of best suitable variables that can maximize or minimize the value of a given function.

In this context, the problem of system identification could also be seen as an optimization problem. Considering that a given structure of a model with variable parameters, can have multiple models deriving from it, with all being able to approximate a certain system, the optimization approach would consider which parameters would give the model with the best “performance”, for example, less deviation for the real system measured by an error function.

#### 4.13.1 Genetic Algorithm

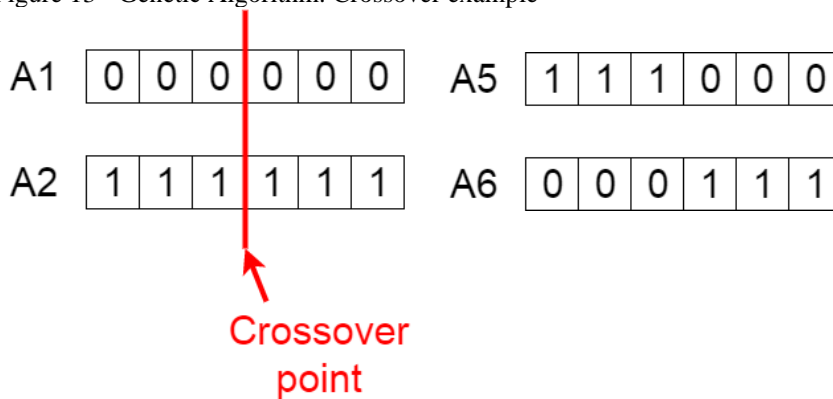
Genetic algorithm is a computational model inspired by the principle of evolution, and natural selection, where the fittest individuals (solutions) are selected for reproduction to generate the next generation. The concept consists in encapsulate the solution of a problem in a “chromosome-like” data structure, that characterize an *individual*. A collection of *individuals*, (the solution candidates) are called *population*. [28]

The Genetic Algorithm is a method for solving constrained and unconstrained optimization problems. [29] It continually modifies the population selecting random individuals in each generation to be parents of new generation solutions. Over successive generations, the population "evolves" toward an optimal solution. [29]

The algorithm follows a set of three rules to create each generation:

- **Selection:** Select the parents that will contribute to the population for the next generation. The individuals with better performance have better chances to be selected
- **Crossover:** combine two different parents to form the children for the next generation. Figure 13 - Genetic Algorithm: Crossover example exemplifies that by showing the generation of two news binary-structured chromosomes (A5 and A6) from the crossover of its parents (A1, A2);

Figure 13 - Genetic Algorithm: Crossover example



Source: [28]

- **Mutation:** applies random mutation to individuals to maintain diversity within the population (Figure 14 - Genetic Algorithm: Mutation Example).

Figure 14 - Genetic Algorithm: Mutation Example  
Before Mutation

A5 [1 | 1 | 1 | 0 | 0 | 0]

After Mutation

A5 [1 | 1 | 0 | 1 | 1 | 0]

Source: [28]

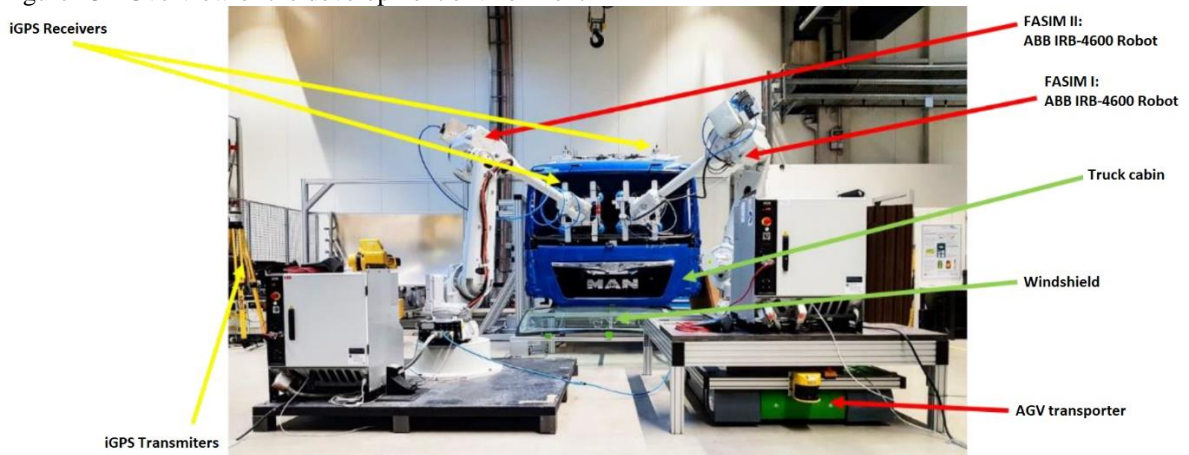
The performance of each individual is given by a so-called *fitness function*. The selection can be made throughout a series of different methods, as for example the stochastic uniform selection. This method uniformly lays the individuals in a line, in which each individual corresponds to a section of it, proportional to its fitness value. The algorithm moves along the line with a fixed step size selecting the parent that it lands on.



## 5 SYSTEM DESCRIPTION

This chapter aims to present the specific the elements that compose the system to be identified. The research environment is constituted in a way to simulate a continuous flow assembly line. Composed by two ABB IR-4600 robots, a truck cabin and windshield provide by WZL's partners, an AGV transporter and NIKON's iGPS system transmitters, spread through the shop floor, and receivers located on the truck's cabin and robot's actuators (Figure 15).

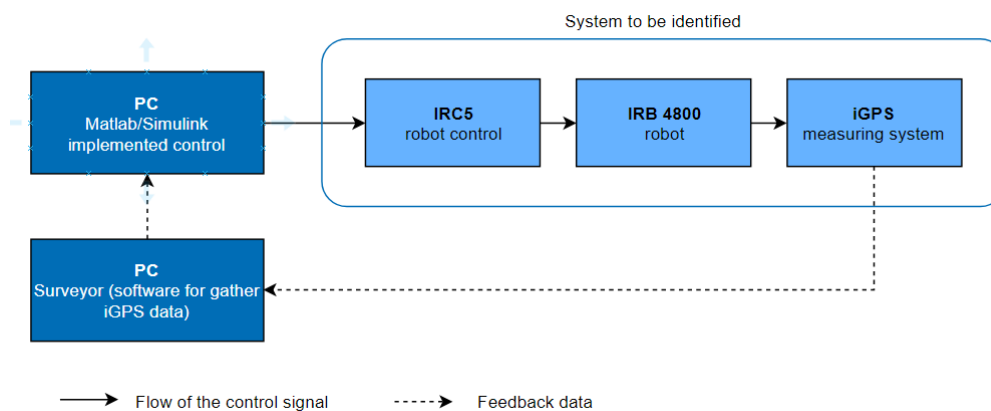
Figure 15 - Overview of the development environment



Source: [4]

The focus of this work it's to identify a better model of the ABB robot using its provided control interface, merged with the iGPS large volume metrology device. Figure 16 exemplify the elements of interest.

Figure 16 - Overview scheme of whole system



Source: author, [9]

## 5.1 ABB IRB-4600 / IRC5

The ABB IRB 4600 is one 6-DOF robot manipulator model, manufacture by ABB, suitable for material handling, machine tending, laser- and water jet cutting, dispensing, measuring, assembly and welding applications. [30] The IRC5 correspond to the control unit shipped with the articulated robot.

### 5.1.1 EGM – Externally Guided Motion

The *Externally Guided Motion* is the communication interface that conducts the communication of the robot controller with the central control unity. Provided by the robot manufacturer. EGM offers two different features: EGM Position Guidance and EGM Path Correction, the first being the most suitable one for the problem. [31]

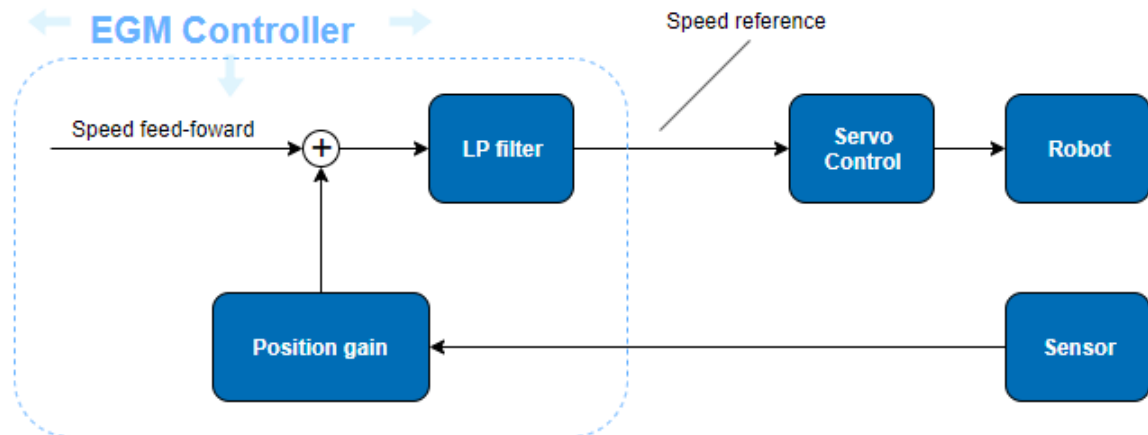
The purpose of EGM Position Guidance is to generate the path of the robot through the use of an external device to generate position data for one or several robots. [31] In this context, EGM it's a valid approach to the project as it's capable to handle the position data provided from a metrology system and feed data to multiple robots

EGM Position Guidance is ideal to applications when high responsive robot movements are needed as it provides a fast reading and writing of positions to the motion system (4 ms) reaching a control lag of 20 ms. [31]

The operation of EGM can be summarized by its control loop (Figure 17):

1. Set point for speed of the robot to be controlled is determine with the help of an external sensor;
2. Send the setpoint to EGM controller through its interface;
3. EGM controller:
  - a. generates the manipulated variable for speed through a position gain;
  - b. superimposing the manipulated variable with value of the feedforward control for the speed;
  - c. filter the speed performance, generating the speed reference to the servo control;
4. The robot is moved via the servo drives;
5. Position is detected by the sensor and the loop comes back to 2.

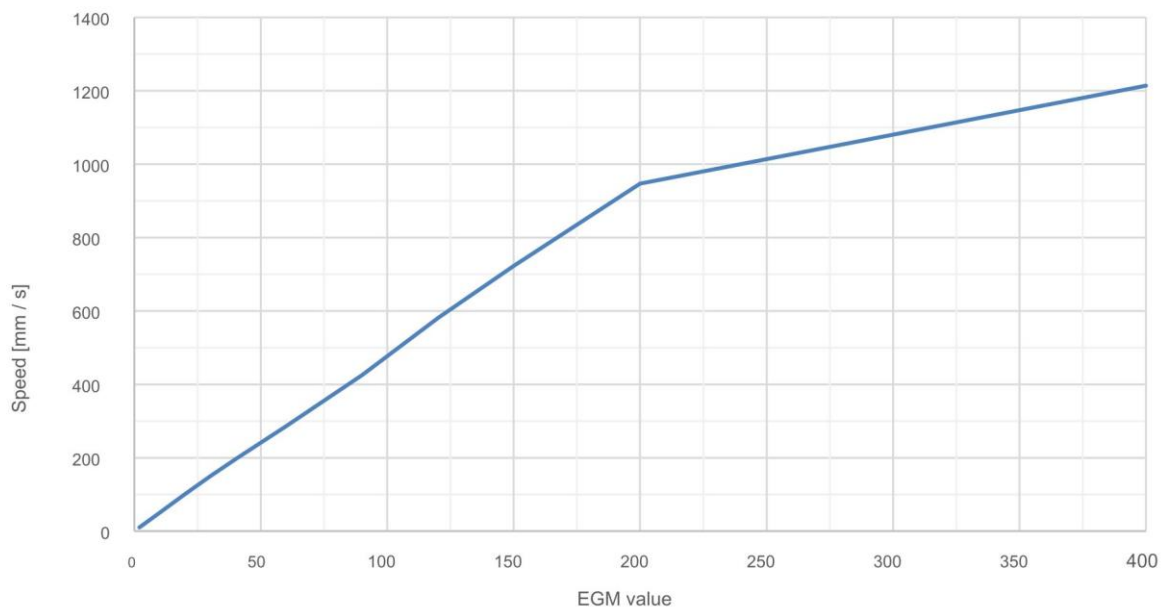
Figure 17 - Structure of EGM controller



Source: [31]

One of its limitations is that it is not possible to perform linear movements using EGM Position Guidance, since EGM Position Guidance does not contain interpolator functionality. [31] Even so, throughout the observations made in [9] and in this work, EGM showed a strong sense of linear behavior in the desired operation range (Figure 18) and presents a lower dead time making it the selected interface by WZL.

Figure 18 - EGM speed characteristic curve



Source: [9]

Given the defined methodology, identifying a model close to the current laboratory model is one of the first steps in implementation. That said, this work assumes the same assumptions as in [9]: the robot modes at a maximum speed of 200 *mm/s*; the object to be

tracked moves at a continuous speed through the working area of the robot without string delays and standstills.

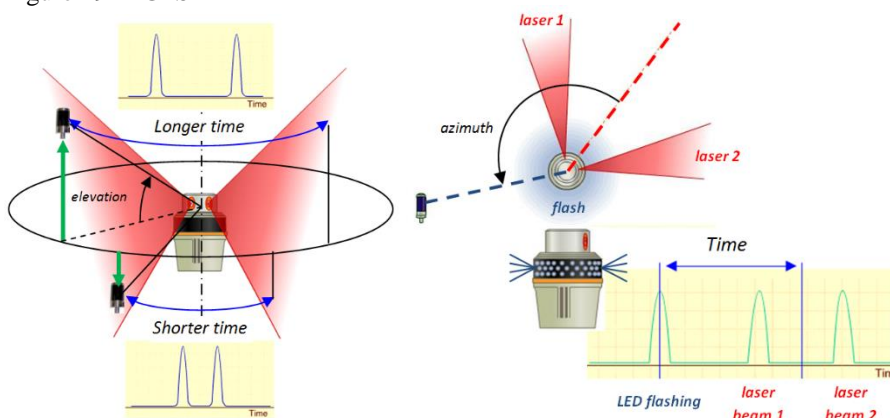
## 5.2 Nikon iGPS

In a flexible production system environment with several robots it is necessary to keep track of the position of the robots as well other elements of the production. As presented in section 4.5.1, the iGPS from Nikon [13] is a modular, distributed, laser-base non-contact, large-volume metrology system capable of defining the position of several elements in a production environment. Its hardware is divided in two types: optical sensors and transmitters

The transmitters are placed in the working area and continuously generate three signals; two infrared fanned beams rotating in the head, and an infrared LED strobe. [14] These signals are converted into timing pulses through a photo detector and the so-called, strobe signal, marks the reference time, which is analog to the zero degree angle of the transmitter in the axis of rotation [3]. The rotation speed of the head are different in each transmitter to distinguish one from another.

The receivers can then be set on the point of interest to determine its position in the working area. Due to the tilt of the two laser beams it is possible to determine the vertical position of the sensor. As the beams are close to each other below the head, a smaller time between the detection of the two lasers means that the sensor is located below it (Figure 19). The time between the LED signal and the midpoint of the two lasers signals is used to calculate the horizontal position.

Figure 19 – iGPS



Source: [32]

Each transmitter-receiver pair can identify the position of the sensor along a line, but its distance is unknown, for that at least two transmitters are required. [32] The Global Reference

System can be conceived by measuring a number of point inside the working area and applying scaling information between the points.

The process of determine the position of the sensor is determined by a so-called Position Calculation Engine (PCE), part of the sensors' hardware and able to communicate with a PC through an industrial network, wired or wirelessly. In order to identify the elements of the system, a frame consisting of four receivers with a PCE unit is mounted on the manipulators' actuator and the product. [3]

Prior analyzes have determined [9, 14] that the iGPS meets the systems requirements in both static and dynamic conditions with the advantage of being capable to identify multiple points at the same time at a lower cost. [14] To speeds bellow  $100\text{ mm/s}$  is possible to achieve a deviation of less than  $0.5\text{ mm}$  with a standart deviation of  $\pm 3\sigma$  (Table 2).

Table 2 - Key figures of the iGPS

Position update rate		40 HZ
Static accuracy		$\sim 0.05\text{ mm}$
Dynamic accuracy	Speed < $100\text{ mm/s}$	< $0.5\text{ mm}$
	Speed = $1000\text{ mm/s}$	3 to 4 mm
Standard deviation		$\pm 3\sigma$
Transmitter range		2 to 55 m
Transmitter frequency		40 to 55 Hz
Deadtime delay		$\sim 200\text{ ms}$

Source: [9]

Tests carried out by WZL [9] showed the following deviations (Table 3) for both static and dynamic conditions. Under dynamics conditions, 70% of the measurement where to be found in the interval of the desired specification of  $\pm 1\text{ mm}$  [9].

Table 3 - iGPS characteristics observed by WZL

Standard deviation static	0.053
Standard deviation Dynamic ( $v = 100\text{ mm/s}$ )	0.719
Dead time	$\sim 180\text{ ms}$ (max. $200\text{ ms}$ , min $100\text{ ms}$ )

Source: [9]

The data collected for this study showed a mean standard deviation for some of the data about 0.200. The procedure to calculate the standard deviation was performed by defining an interval with a good linear behavior of the system output and applying a linear regression as the mean of the measurements. The lower deviation between this and [9]works could have been

given due to the low speed during tests, since under dynamic conditions the system is less precise [9, 14].

## 6 IMPLEMENTATION

Section 4.3 showed the principles of modeling the components and the robot manipulator, presenting an overview of the nonlinearity's and uncertainties presenting in the physical system. That being said, a model that describes the entire dynamics of the system would be very complex and time consuming to be found.

Like shown in the Figure 16 on chapter 5, the system which this works seeks to identify consists of a closed loop control system of the robotic manipulator integrated with the chosen measurement system. Consequently, it's safe to assume that a model with such complexity is already defined by the robot manufacturer as pointed out in section 5.1.1. The EGM interface provide a velocity control, leaving the laboratory to define the path of the actuator, through its controller.

In this case, aiming to avoid any kind of rework in the physical modeling of the system, and due to the lack of knowledge dynamics of the ABB controller, it has been chosen a black-box model structure approach to identify the system in study.

The current implementation of the controller from WLZ was carried out in Matlab's Simulink. The selection of the implementing tools took this strongly into account.

The present section aims to describe the activities carried out through the development of this work, presenting the steps taken, the tools used and its partial results.

### 6.1 Applying machine learning to the identification problem

Focusing to provide a different approach that the one utilized by WZL, this work addressed the problem initially, by using machine learning methods for identifying the system. Chapter 4 showed that the most appropriate machine learning algorithm for identification are neural networks due to their application as mapping in the theory of systems identification. That being said, the firsts steps of this works were to identify a NN model that could, at least, approximate the current WZL model.

Following the adopted methodology, after determining that a given method is appropriate for the identification problem, it is necessary to identify a suitable environment to implement it. In this case, the Matlab's Deep Learning Toolbox [33] Neural Time Series Tool was selected.

Firstly, it was investigated if it would fit to approximate the currently model provided by WZL. To do this a series of signals were generate as input to the model (a step input, a sine

input, and a uniformly distributed random signal), its output was then recorded and used as a pair input-output data for the training of the network.

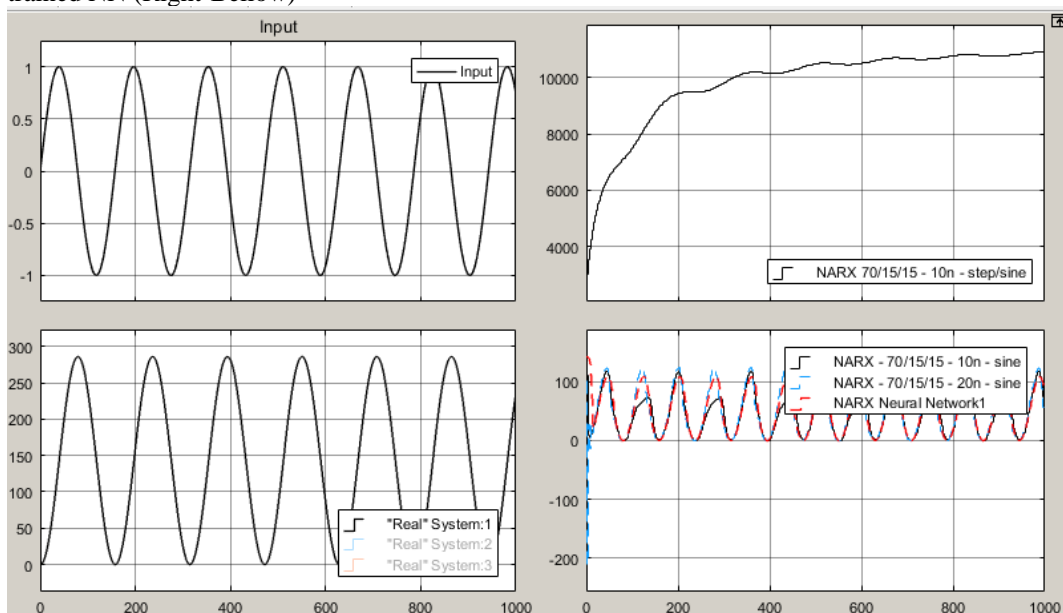
The user interface of the tool can be seen in ANNEX A. It was selected a Nonlinear Autoregressive with External (Exogenous) Input structure - NARX (Table 1). The other configuration was maintained in the preconfiguration of Matlab's, these encompass: the division of the data set ( 70% of the data set for training data, 15% for validation and, 15% for testing); the number of neurons, and delayed outputs that are used as inputs to the NN; and the training algorithms.

The tool provide an option to generate a Simulink block after the training is finished. With this block the NN was used in a feedforward configuration (the inputs of the network were inputs and past outputs of the system model), and simulated in the simulink enviroment.

The use of the step and sine inputs present some case of overfitting, when the model "memorizes" the training data and fails to generalize well the behavior of the system. An example of this could be seen in Figure 20, where a NN that was trained with just the step input-output of WZL model (Figure 20, Right-Above) shows a output with strong resemble to the step response of the system.

The simulation presented below and the subsequent ones are simulations of the models' behavior over time.

Figure 20 - Overfitting example: Input(Left-Above), Output(Left-Bellow), Step trained NN(Right-Above), Sine trained NN (Right-Bellow)



Source: author

Figure 20 also exemplify the poor choosing of the NN's sample time in the response of the sine-trained NN that presented a bad result in post simulation (Right-bellow). Matlab's

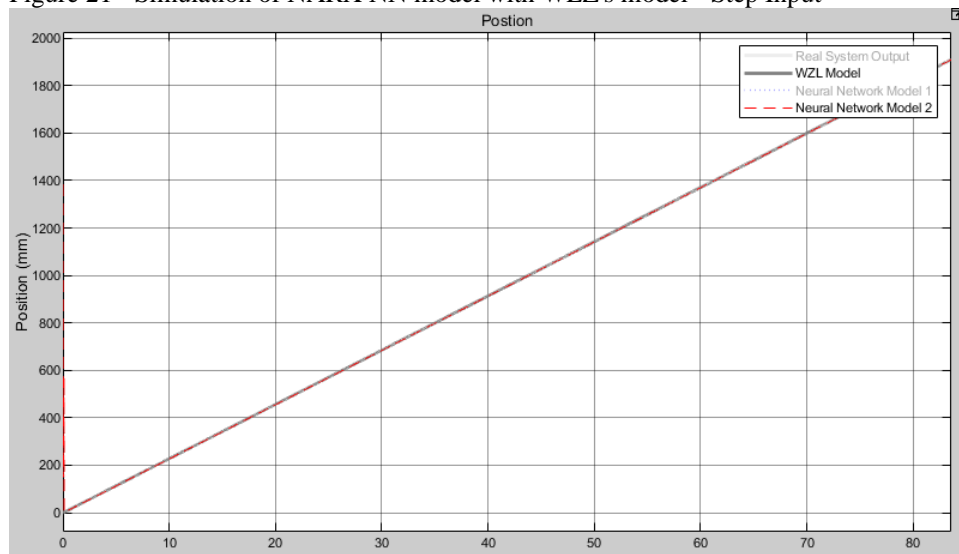


tool generate a Simulink block for use with WZL's model, but generate it with a pre-established sample time.

Training the NN with a uniformly distributed random signal promoted better results as it was understood that the data set would present a more general view of WZL's model. The configuration of the time-sample prior to the creation of the Simulink block also correct the bad post simulations problem above mentioned.

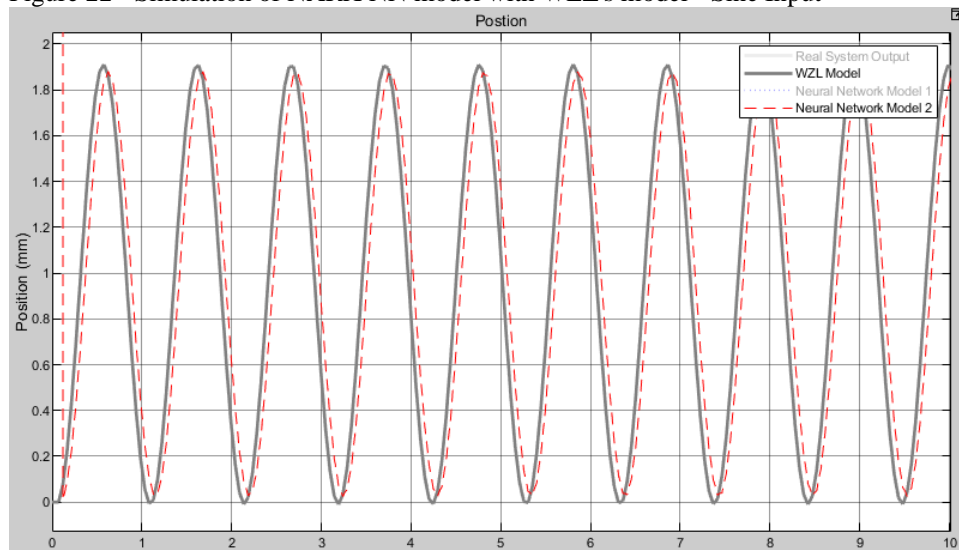
The results of a series of different signals could be seen following:

Figure 21 - Simulation of NARX NN model with WLZ's model - Step Input



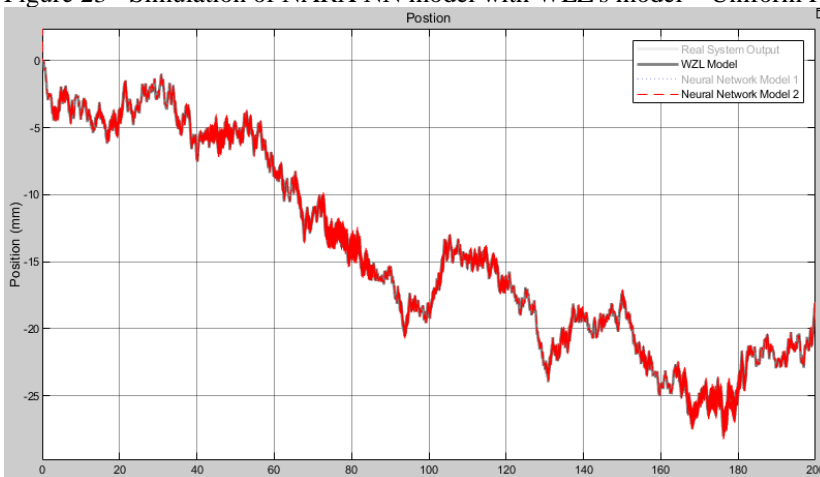
Source: author

Figure 22 - Simulation of NARX NN model with WLZ's model - Sine Input



Source: author

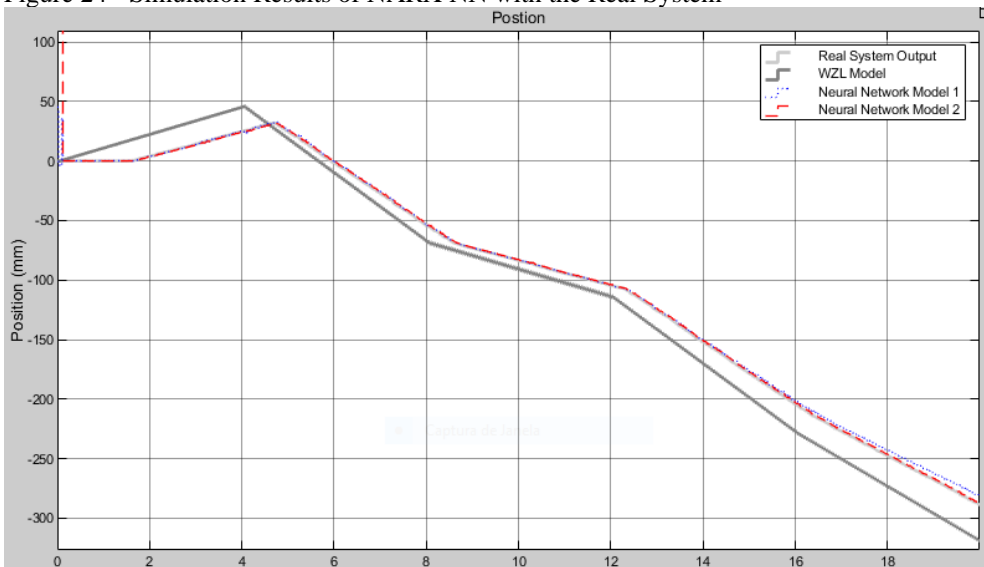
Figure 23 - Simulation of NARX NN model with WLZ's model – Uniform Random Signal



Source: author

The above results proved that the time series networks were able to capture the dynamics of the model from WZL. The following picture Figure 24 demonstrate also that the model trained NN could also approximate the real system with relatively good results if its inputs are changed to the real system data.

Figure 24 - Simulation Results of NARX NN with the Real System



Source: author

### 6.1.1 Adapting the NN structure for the WZL case

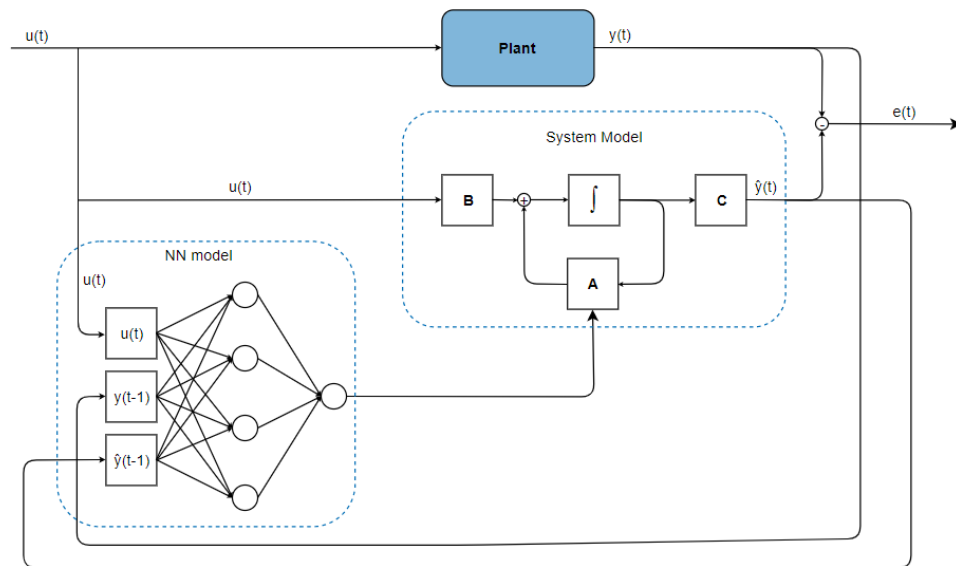
The above results showed that the NN model is possible and presents good results in dealing with predict the system's output during run time. Unfortunately, the above implementation wouldn't be well suited to WZL. The controller was in an advanced process of development and it required a State Space structured model (4.7).

Because of this, the following activities focused on evaluate the possibility to adapt this approach to the requirements from WZL, and to locate a better tool to use as an implementation starting point since the adaptation of the NN provided by the Time Series Network was limited.

In this context, the Neural Network based System Identification Tool Box provided by Nørgaard [22] seemed to be an good starting point. Its recursive version of the back-propagation algorithm was select as the starting point. The idea was to use regression capability of a NN making its output (4-75) a function to determine the parameters that would provide the best results desired.

The training would be given the same way as for any other NN, providing as training data, the input and output of WZL's model, the difference will be that the output of the NN model in training, would not be the output of the NN but of a state space structure. Figure 25 demonstrates a concept of the idea proposed.

Figure 25 - Proposed scheme to neural network for adapt SS model



Source: author

During the evaluation and adaptation of Nørgaard's tool [22], the problem with this approach was identified. For the back-propagation training algorithm to work, its necessary to calculate de gradient of the error function in respect to each weight that would be adjusted, like demonstrated in 4-75 and presented again bellow.

$$\text{grad } V_N(\theta, Z^N) = \left\langle \frac{\partial V_N(\theta, Z^N)}{\partial w_{11}}, \dots, \frac{\partial V_N(\theta, Z^N)}{\partial w_{qm}}, \frac{\partial V_N(\theta, Z^N)}{\partial W_{11}}, \dots, \frac{\partial V_N(\theta, Z^N)}{\partial W_{qm}} \right\rangle \quad 6-1$$

The problem here is that the error would no longer be a function of system's output and neural network's output. For this case, the error would be a function of the state space representation output, which would imply being a function of the system dynamics as well, that, will vary over time due to updating of the parameters. This makes the calculation of the gradient, before simple to define, become extremely complex and time consuming.

Due to this problem, other approaches being studied during the work were selected as the main approach.

## 6.2 An optimization approach

Following what is said in 4.13, the system identification problem can be seen as a optimization problem, even more, when the structure of the model is well defined, in this case, the state space structure used by WZL's MPC. Following the definition in [27], the problem of system identification as an optimization one was formulated as following.

The purpose of this problem is to design a model for a robot arm using the states space representation structure, that provide a good approximation with the real system. The focus is to minimize the error between the output of the real system and the output of the model. The error function utilized was the mean square error:

$$MSE = \frac{1}{N} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad 6-2$$

The limitations are provided by the state space model. There are no clear restrains for the variables as them are parameters of the matrix A of the system, and it's assumed that any real value is applicable. The B, C, and D matrix were defined by converting the TF WZL's model to a state space representation. The size of the matrix A was defined similarly.

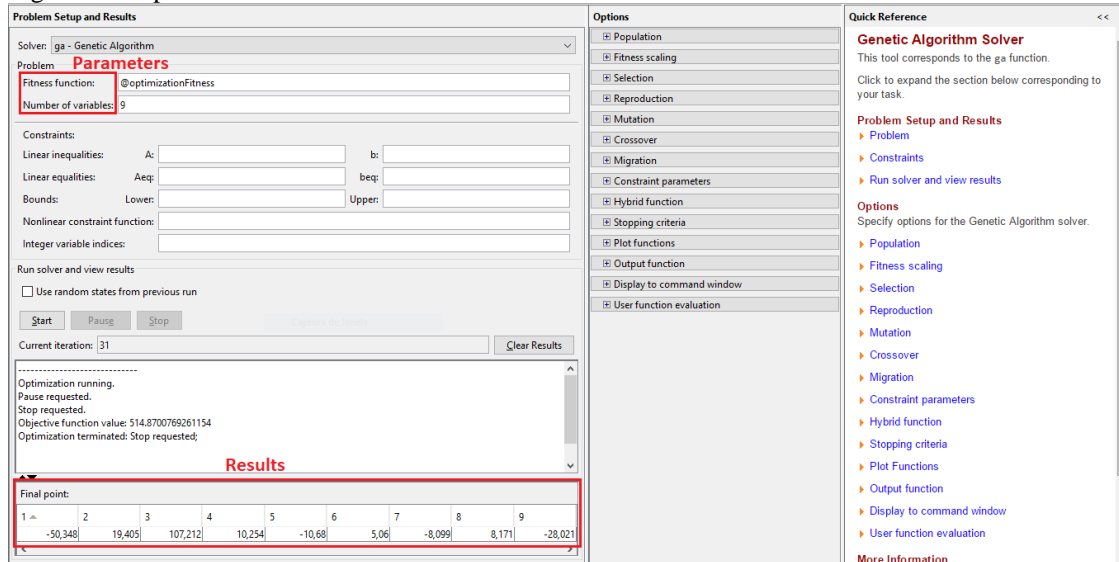
The chosen optimization algorithm was Genetic Algorithm. The implementation was taken by using Matlab's Global Optimization Tool [29]. The user interface of the tool is shown below (Figure 26): In this works case, only the parameters fields were determined, even so, the tool provide a series of configurations for the algorithm.

The fitness function field takes a Matlab function file that establish the GA fitness function and the variables field takes the number of desirable parameters to determine. The population type was defined as a vector of double variables.

The Matlab function file (APPENDIX A) generates both a step and a sine signal that is passed to the WZL's model and the output is stored. Later, the same signals are passed to the

GA models and the prediction is also stored. Finally, the score of the fitness function is calculated as the MSE.

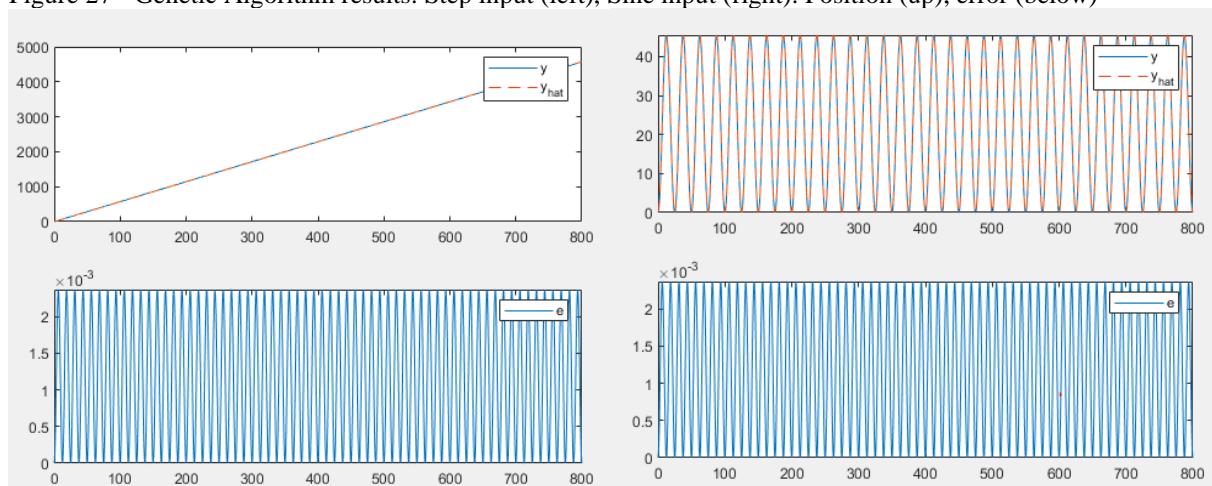
Figure 26 - Optimization tool interface



Source: [29]

With the parameters established by the optimization tool, they were then, passed to an state space model using the *ss* command [34], and then the model is simulated side by side with WZL's models. Figure 27 shows the results of these simulation. The results were promising and showed a satisfactory approximation from the GA model to WLZ's, with the error oscillating from 0 to 0.003.

Figure 27 - Genetic Algorithm results: Step input (left), Sine input (right). Position (up), error (below)



Source: author.

This approach was not carried out through the rest of the established methodology because it presented an only offline identification approach, meaning that it only could identify the system with previous collected data, and not provide an adaptive model as the system works.

Even so, the optimization approach with genetic algorithm was implemented relatively fast with the Matlab's Optimization Tool and was taken alongside the development of 6.1.1. It showed that the neural network approach would present itself as a challenging approach to adjust to the WLZ' problems, while better and more appropriate options to solve the problem would exist.

### 6.3 Recursive Algorithms for Online Estimation

Seeking a more suitable and simpler method to provide a better model, the research role of this works explored the applications of online estimation methods. Designed for this sole purpose of parameter estimation, seemed to be the most reasonable approach for WZL's problem.

The System Identification Toolbox from Matlab was previous applied in [9] to identify a model that provide sufficient approximation to prove that the problem of the controller synchronization was possible using a MPC. However, it was not good enough to keep up with the high standards of the flexible production line in the automotive industry for example.

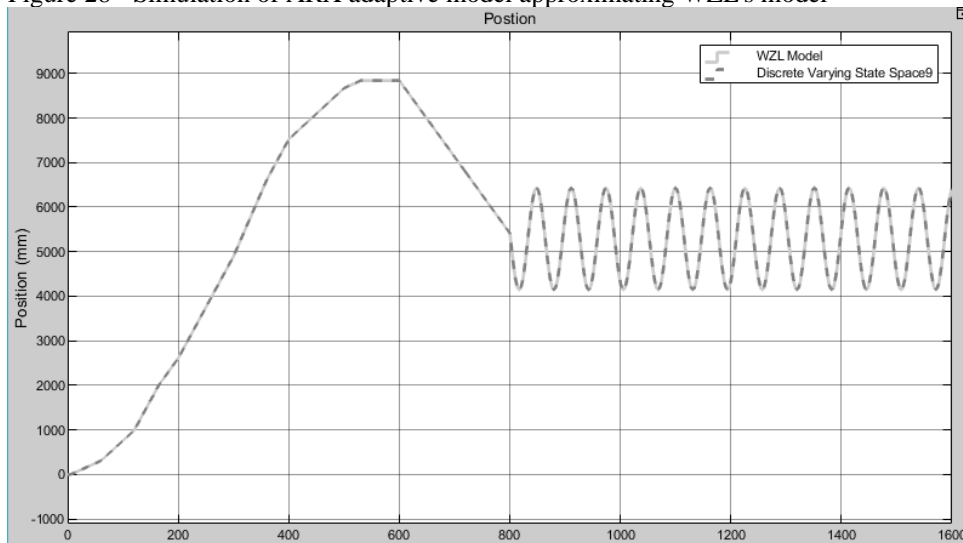
Even so, the same toolbox provides tools for online estimation with easy and fast implementation. Being able to address the specific problem of state space parameters estimation with a small number of Simulink blocks and parameters. For that reason, this tool was used for further implementations.

The Recursive Polynomial Model Estimator (ANNEX B) presents itself as an approach to identify adaptive linear polynomial models (see 4.9.2.1 for the structures available) within a closed time window or a continuous estimation during the system's runtime. The tool also provided a converter for the state space representation natively

Because of the strong sense of linear behavior of the EGM (5.1.1), and considering that lower order models are preferred, the firsts implementations of the on Recursive Polynomial Model Estimator were made with the simplest model possible: an ARX polynomial model, with a first order  $A$  and  $B$  polynomial. The input delay was considered 1, meaning that, the model considers that it takes 1-time step for the input affects the output.

Initial results showed that it was capable of approximating WZL's models with no prior knowledge of the model. Figure 28 exhibit that by showing the track of the adaptive model for cases of steps and sines inputs. The algorithm applied was the Forgetting Factor (4.11.3), and the factor itself was configured in a manner to present a fast "learning", in other words, a small memory horizon.

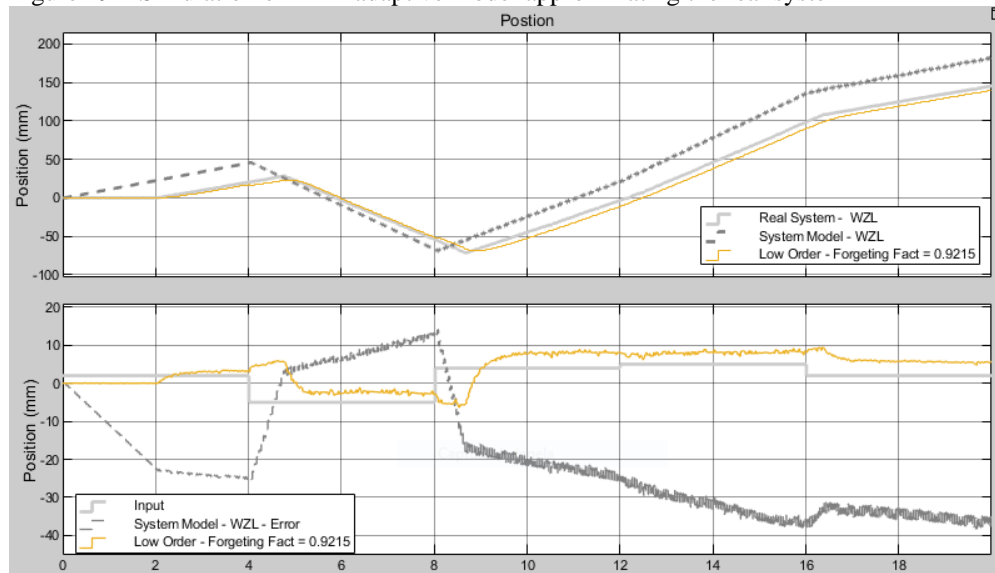
Figure 28 - Simulation of ARX adaptive model approximating WZL's model



Source: author

Being able to provide a great approximation to the original model, this approach was selected between the others to pass through real data tests. The parameters selected were the same as the as previous described. Simulation results (Figure 29) showed that the online estimation provide better results than the model from WZL.

Figure 29 – Simulation of ARX adaptive model approximating the real system

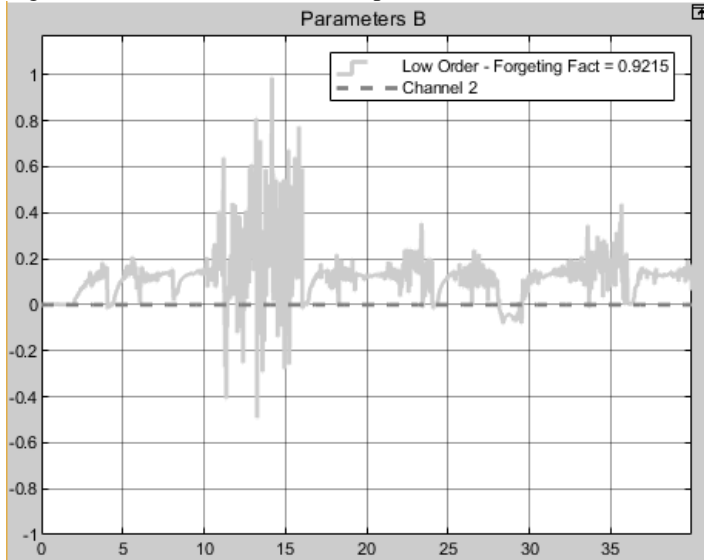


Source: author

Tests results with the real system however were flawed. The unforeseen variation the  $A$ ,  $B$ ,  $C$  and  $D$  matrices' parameters led to misleading control signals that moved the robot in the wrong direction. More specifically, the parameters of the  $B$  matrix, the one tied to the input dynamics, presented a fast change in its value, and presenting some cases of zero crossing

(Figure 30). In the experiment, this resulted in both cooperative robots moving in the opposite direction

Figure 30 - Evolution of non-zero parameter of Matrix B



Source: author

Because of this, it was necessary to step back in the adopted methodology to better adjust the parameters of the online estimation.

Regarding the Forgetting Factor algorithm (APPENDIX B), it was identified that it wasn't possible to avoid the oscillation in the parameters without losing in accuracy. Models with less oscillation provide poor accuracy in comparison to WZL's model. It was also noticed that the algorithm provides no convergence.

To solve this problem, two approaches were adopted: reinvestigate the other algorithms to identify a model with less accuracy, but with less oscillation or with convergence; provide an offline-estimated model as an initial approximation to the online algorithm.

The first one did not present conclusive results by itself. For all algorithms (4.11.2) of the Recursive Polynomial Model Estimator's (ANNEX B) a good better approximation than the original model came at the cost of variation in the parameters and no convergence.

For the offline estimation, the Matlab's System Identification Toolbox [21] App (ANNEX C) was used. This tool, the same used by WZL to identify the prior model, also provide command lines code that let the user create scripts for more specific problems.

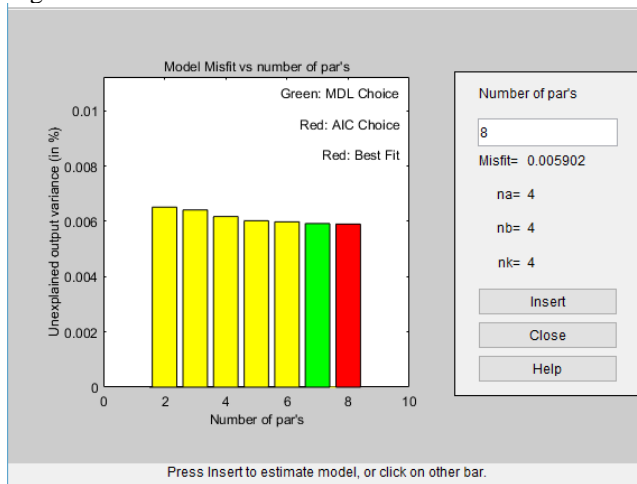
The tool also provides options to select the most appropriate order for the model. By estimating ARX models for a range of structures of different orders and delays comparing each order and highlighting the structures with best results in three different best-fit criterion.

- Red: Mean Square Error;



- Green: Rissanen MDL criterion.
- Blue: Akaike AIC criterion.

Figure 31 - Order selection section

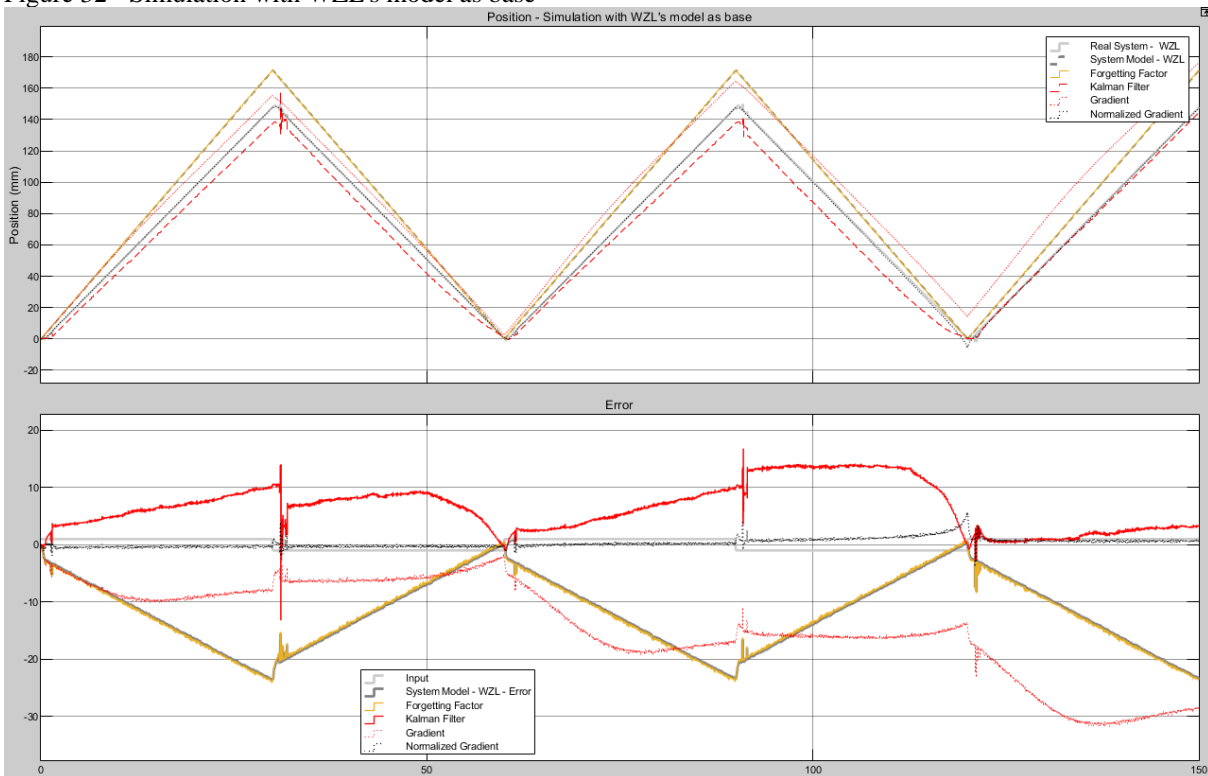


Source: author, [21]

Testing with a series of offline-estimated models combined with different online estimation algorithms made possible to conclude that the Gradient and the Normalized Gradient provide an overall better results. The following figures Figure 32 and Figure 33 exemplify this by showing results of the four available algorithms with the same base model (the prior, identified by WZL).

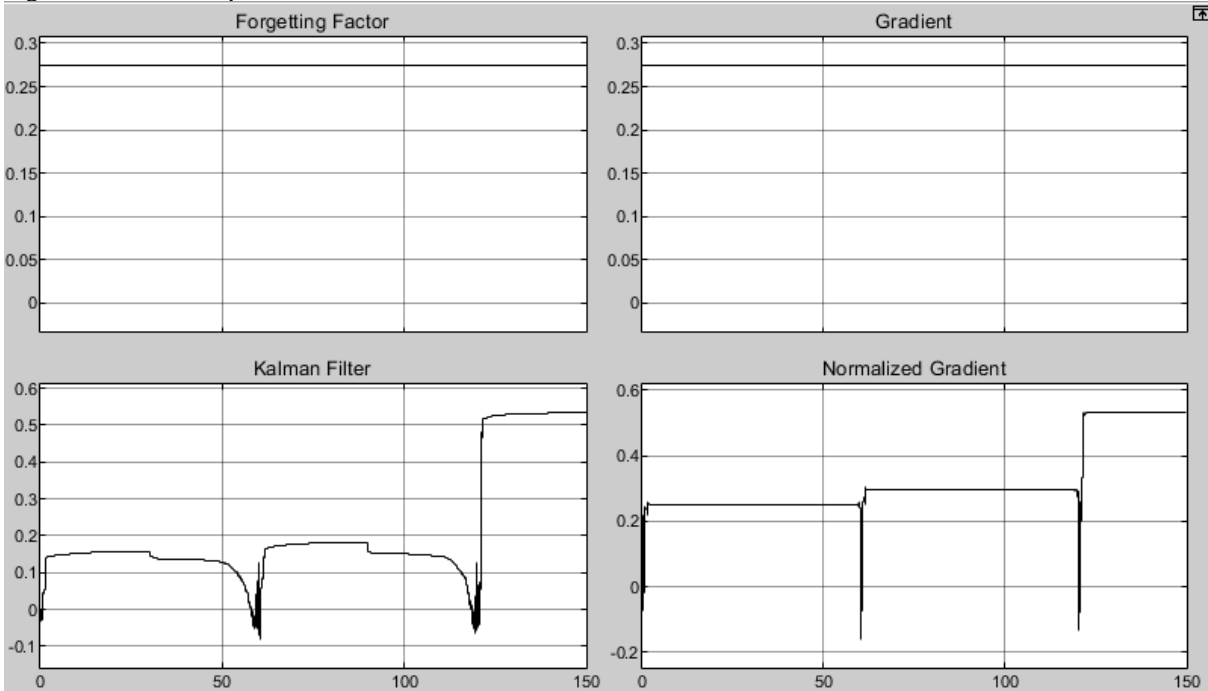
In general, the Kalman Filter provide good results if well adjusted, but the oscillation in the parameters were always higher. The Forgetting Factor wasn't able to approximate the system with an offline combination. The Normalized Gradient seemed to stabilize but presented a "step-like" variation in the parameters. Finally, the Gradient approach was the only one that showed some kind of convergence, but, wasn't able to give a better model than the initial one.

Figure 32 - Simulation with WZL's model as base



Source: author

Figure 33 - Nonzero parameters from  $B$  matrix – Off/On-line estimation



Source: author

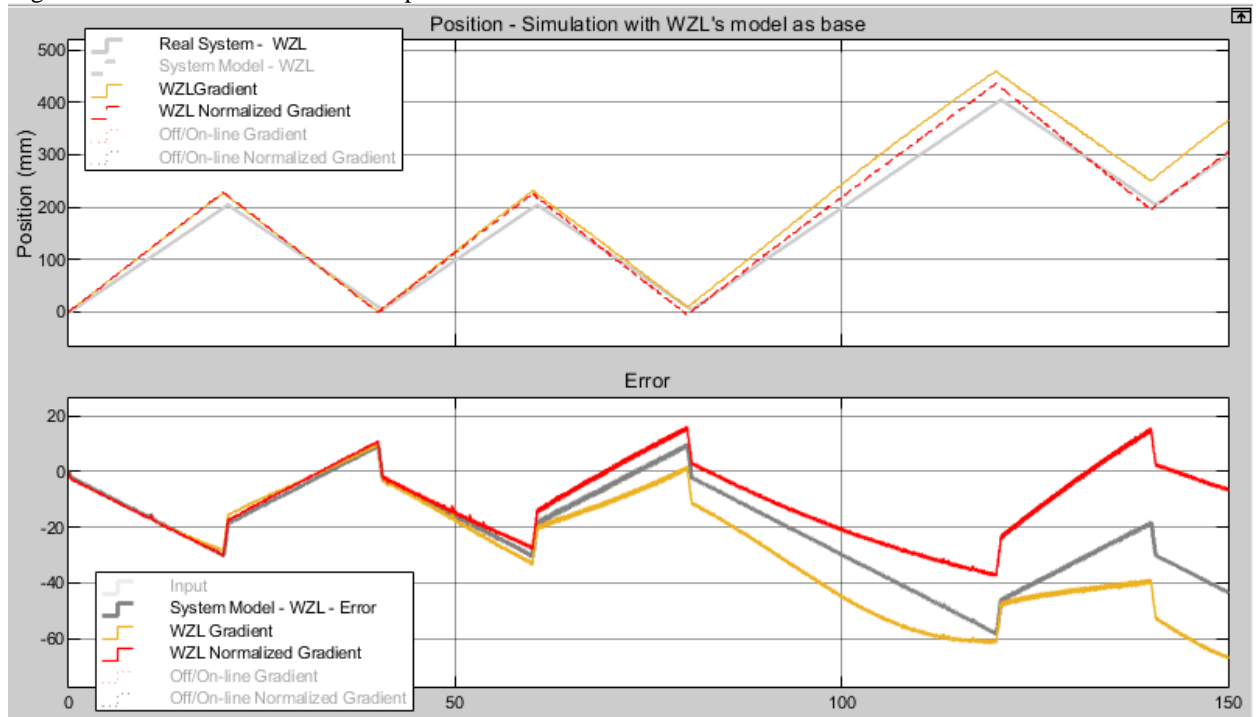
With what was presented, two approaches were selected for further implementations: The Gradient and Normalized Gradient.

Simulation tests showed that the models estimated offline in this works had better approximation than the original WZL mode, however, the data for estimating and testing the

models where captured within a short period of time, this could mean that the current state of the system has changed and the data used for WZL's model no longer generalize well the system as the data set used here. Or, that the current data is misleading.

For this reason, it was selected the algorithm that better managed to detach itself from the initial parameters. In this matter, the Normalized Gradient showed a more accurate and precise results (see APPENDIX C) in most cases. Figure 34 exemplifies that demonstrating that the Normalized Gradient managed to track the model's outputs with time.

Figure 34 - Gradient methods in comparison



Source: author

With the approach selected, and with the algorithm determined, it was possible to develop the final solution for WZL's problem.

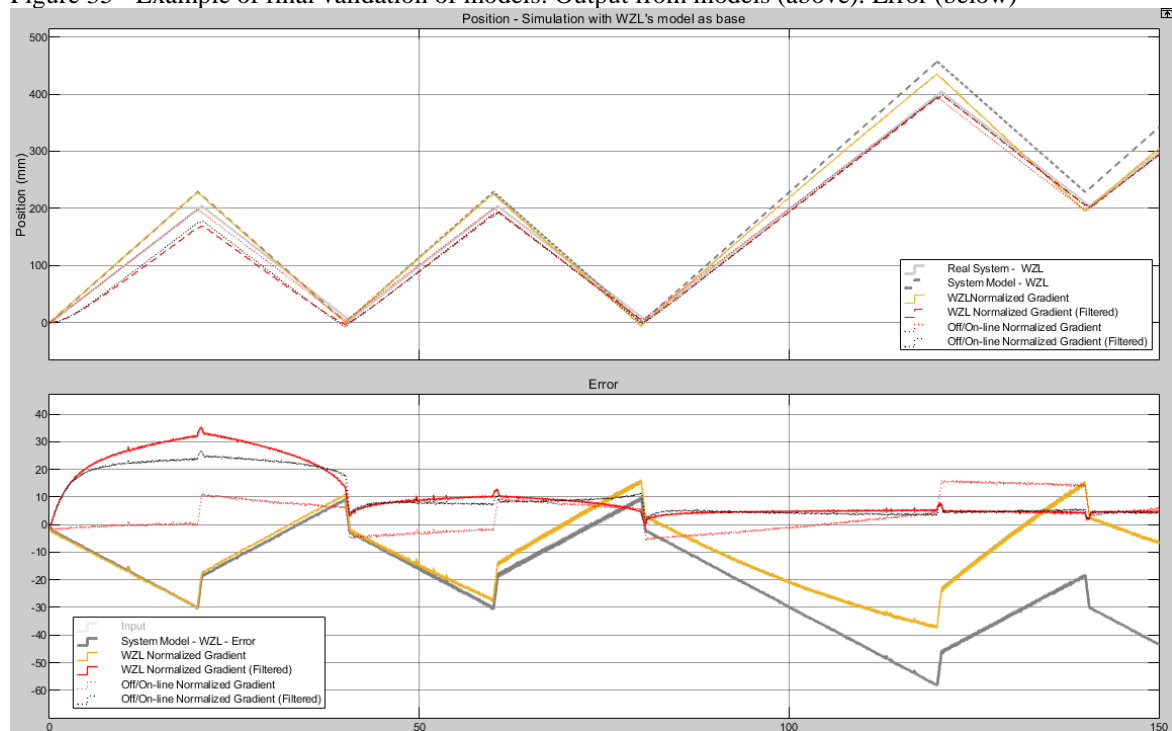


## 7 FINAL RESULTS

The previous chapter provided the steps taken and its subsequent results throughout the development of this work in seeking a suitable approach, and its implementation to solve WZL's identification problem. Through the described steps it was concluded that the online estimation tool from Matlab, combined with an offline estimated model could provide an adaptive model for the articulated robot using EGM and iGPS.

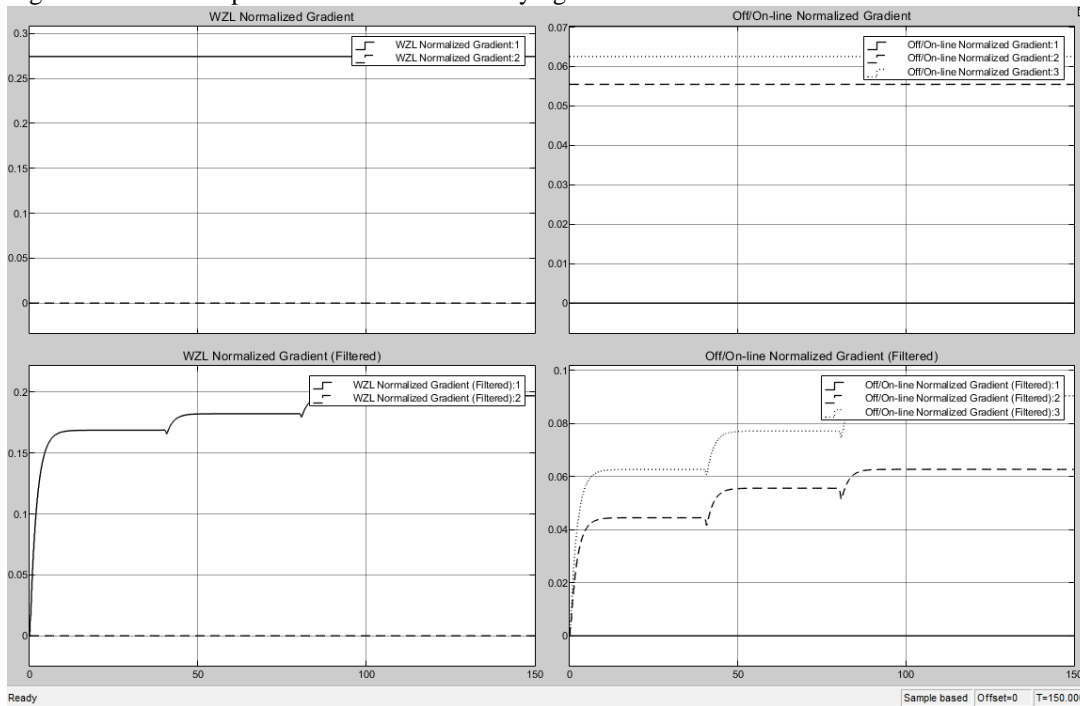
The integration with the control was tested in the real system, but without measuring the quality of the model in this environment. Thus, the results were considered using strictly a series of different test data sets in a simulation environment. The capacity of the models to follow the real position of the system and its relative error (Figure 35) while maintaining a smooth parameter variation (Figure 36) was evaluated.

Figure 35 - Example of final validation of models. Output from models (above). Error (below)



Source: author

It was also studied a application of a filter with a slower time constant between the estimator's converter and the model to evaluate the possibility to provide more aggressive estimation. Even so the parameters showed a slower variation (Figure 36), it was not capable to provide any better results (Figures Figure 35, Figure 37 and Figure 38)

Figure 36 - Nonzero parameters of  $B$  matrix varying over time

Source: author

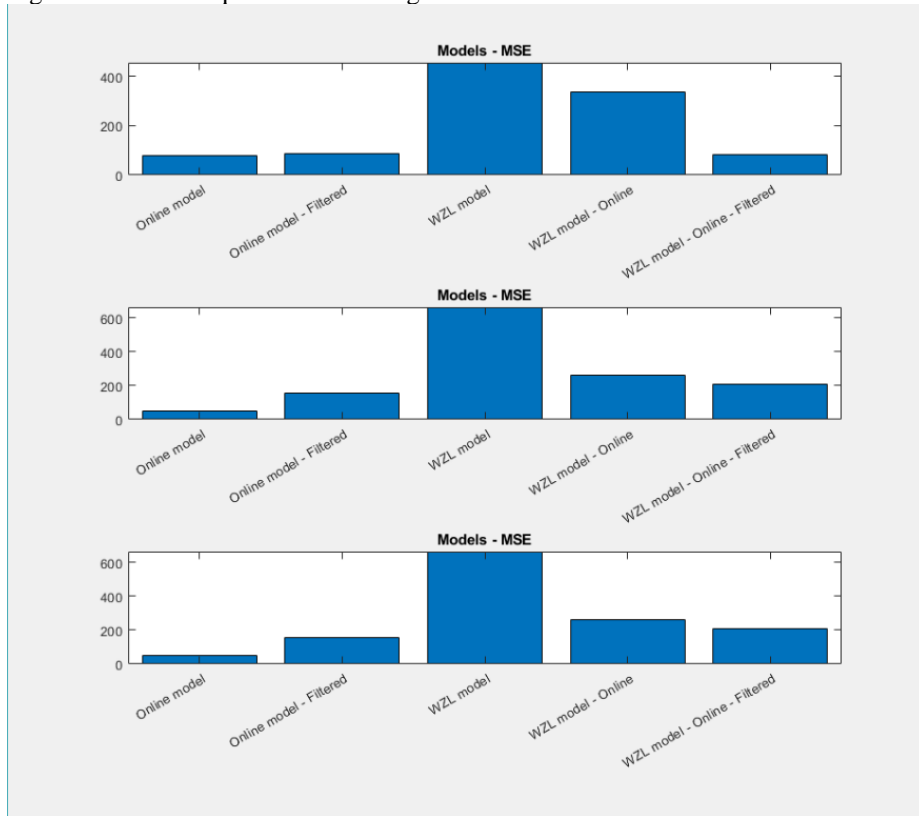
It was also considered an performance parameter to evaluate the quality of the models identified. The Mean Square Error was used for this matter. FiguresFigure 37 and Figure 38 presents the results for seven different datasets comparing the performance of each model.

These results showed that the combination of offline and online estimation provided by Matlab is an adequate tool capable of providing an adaptive model that promotes more accurate results than the current one, while presenting an easy integration to the environment and current control.

Unfortunately, the low order model requested by the control design proved that the online estimation does not handle oscillating inputs very well. For more accurate results, a higher order model would be preferable (APPENDIX D).

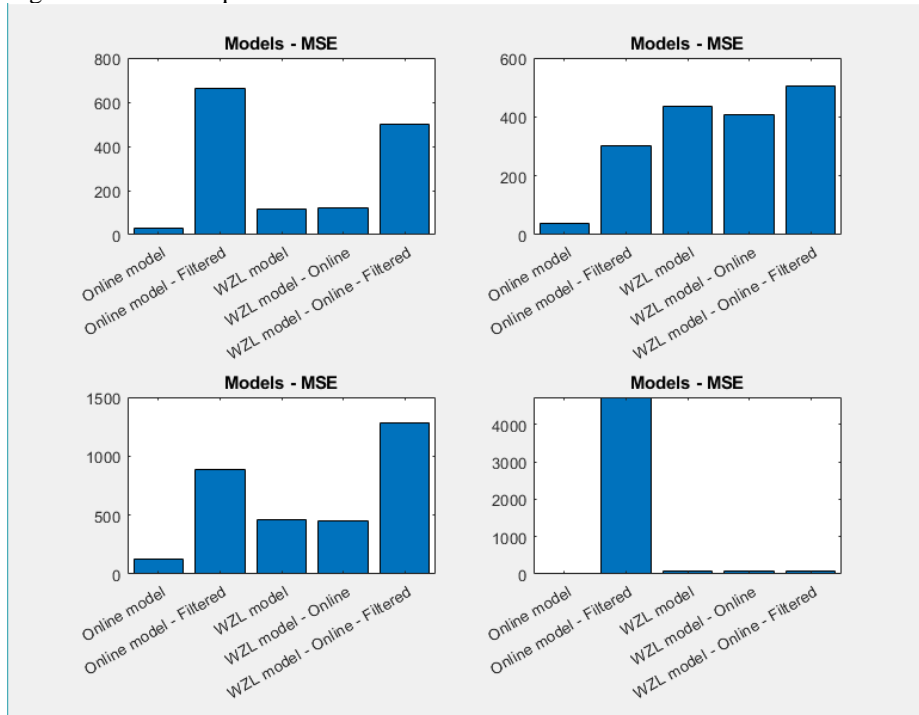
Regardless, it is possible to assume, that, the main goal of this work was achieved. The following MSE charts proved that it's possible to provide adaptative models with better approximation than that which was obtained by WZL. And, at the very least, it can improve the current WZL's model (APPENDIX C).

Figure 37 - Mean Square Error – Longer Simulation



Source: author

Figure 38 - Mean Square Error – Short Simulation



Source: author





## 8 FINAL CONSIDERATIONS AND PERSPECTIVES

Through the research carried out it was possible to identify and implement three different techniques for identify a model for an articulated robot with a black box approach.

Machine Learning represented by the use of Neural Networks are an appropriate method to identify a system, unfortunately, the structures observed through the conception of this work are not well suited for WLZ's control method at its current state and, are too difficult or time-consuming to adapt.

Yet, the NN model provide a good approximation and as stated in [15]: any model that can predict the output of the system in the future time window of the MPC could be used in this control approach. If desired by WZL to explore others MPC control approaches, this model identification technique could prove itself very useful if not better.

The optimization approach using Genetic Algorithm was implemented relatively fast with the help of the Mathworks' *Matlab*'s Optimization Tool [29], however it presents no major advantage to the remaining alternatives. It only can be used as an offline approach to identify the system and, produces results too similar to the obtained firstly by WZL in previous works, and to the offline estimated models from System Identification Tool Box [21].

For the optimization approach to be interesting to the scope of the FASIM project it would be necessary to investigate any other optimization algorithm or an online option. In this scenario Hasan's *et. al.* work's [35] could it be an alternative for futures investigation.

With that said, online estimation prove itself as a simpler, and more convenient approach to the system identification problem and, made specifically to these kind of problems.

Throughout the whole research period of this work, the online estimation provided by Mathworks' *Matlab*'s, System Identification Toolbox [21] produced the most interesting results. Simulations have proved that the tool is capable of providing more faithful approximations to the real system than the WZL's previously identified model.

Unfortunately, the best results achieved in simulation were not viable in real implementation. The high frequency in which the parameters of the model varied provoked a misled control signal that forced the work to redesign the models. In this context, finding the optimum parameters proved to be a challenging task, as it was necessary to keep in mind that faster, more accurate and more precisely models, came with the cost of fast change in its parameters.

Even so, the online estimation was able to provide better approximation than the original model, with no high frequency parameters' change, when combined with an offline approach as the one used by WZL previously.

The fact that it wasn't possible to achieve convergence in any available method, implies that the nonlinearity from EGM's is more present than expected, making the algorithms needs constant adjustment. Also, the lower order requirement by the controller design has promoted limitations to the performance of the models. Larger order models were able to present better results and the study of their application in the future of the project is suggested.

Other alternatives that counteract this work that may be cited are: further investigation of the Deep Learning Toolbox™ [33] from Mathworks' Matlab, and a grey-box modeling but may involve the need to change the structure of the MPC.

The Deep Learning Toolbox also provide the possibility to identify three neural networks architectures for prediction and control of a system, including as well a MPC control approach. Even though this feature was not explored in this work, it could be an interesting alternative tool to investigate in future works on the FASIM project.

One other path would it be the modeling of a grey-box model. The work from Wernholt [5] would fit as a great stating place. It focusses on a grey-box approach, using physical insights to stablish a structure more faithful to the real system and relays on estimation methods to identify the parameters as was done in this work.

In conclusion, the accuracy of MPC approaches depends, in some extent, of a good prediction of the system. Better models contribute to the optimum control signal that would bring the desired future output in dead-time delayed systems. In this context, any improvement in the adopted model is valid.

Two of the three implementations here developed were able to predict the position with better accuracy and precision to the real system as the previous method. The online estimation prove itself as a fast deployable, simple, that could be easily integrated in the WZL's control environment and capable of improving a linear black-box model for an extremely complex, nonlinear system.

## 9 REFERENCES

- [1] WZL W 2019 *WZL - Werkzeugmaschinenlabor der RWTH AACHEN - Deutsch*  
[https://www.wzl.rwth-aachen.de/cms/www\\_content/en/index.htm](https://www.wzl.rwth-aachen.de/cms/www_content/en/index.htm) (accessed 12 Mar 2019)
- [2] Stephan Weyer, Mathias Schmitt, Moritz Ohner, Dominic Gorecky Towards Industry 4.0 - Standardization as the crucial challenge for highly modular, multi-vendor production systems 2015
- [3] Storm C, Schönberg A and Schmitt R H Model Predictive Control Approach for assembling Large Components in Motion 2017 *Prod. Eng. Res. Devel.* **11** 167–73
- [4] Xia M 2019 Entwicklung einer modellprädiktiven Regelung für zwei kooperierende Roboter *Master Thesis* WZL - Werkzeugmaschinenlabor der RWTH AACHEN, RWTH - Rheinisch-Westfälische Technische Hochschule Aachen
- [5] Wernholt E Multivariable Frequency-Domain Identification of Industrial Robots 2007
- [6] Karnik A M and Sinha N K Modelling a Robot Arm from Sampled Input-Output Data 1985 *IFAC Proceedings Volumes* **18** 189–94
- [7] 2016 *Industrie 4.0: Matrix-Produktion - KUKA AG* <https://www.kuka.com/de-de/branchen/loesungsdatenbank/2016/10/solution-systems-matrix-produktion> (accessed 9 Jun 2019)
- [8] U. Berger, D. T. Le, W. Zou, C. Lehmann, J.P. Städter, A. Ampatzopoulos Development of a mobile robot system for assembly task on continuous conveyor 2015
- [9] Nicksch C 2017 Entwicklung einer modellprädiktiven Regelung für einen messtechnisch geführten Industrieroboter zur Montage einer Windschutzscheibe in Bewegung *Master Thesis* WZL - Werkzeugmaschinenlabor der RWTH AACHEN, RWTH - Rheinisch-Westfälische Technische Hochschule Aachen
- [10] *Fasim XL: Automated assembly of large components under flow conditions*  
<http://www.fasim-xl.de/index.php?id=2> (accessed 30 Jun 2019)

- [11] Wikibooks, open books for an open world 2019 *Robotics Kinematics and Dynamics/Serial Manipulator Position Kinematics*  
[https://en.wikibooks.org/wiki/Robotics\\_Kinematics\\_and\\_Dynamics/Serial\\_Manipulator\\_Position\\_Kinematics](https://en.wikibooks.org/wiki/Robotics_Kinematics_and_Dynamics/Serial_Manipulator_Position_Kinematics) (accessed 23 Jun 2019)
- [12] Maisano D A, Jamshidi J, Franceschini F, Maropoulos P G, Mastrogiacomo L, Mileham A R and Owen G W A comparison of two distributed large-volume measurement systems: The mobile spatial co-ordinate measuring system and the indoor global positioning system 2009 *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **223** 511–21
- [13] *iGPS / iGPS / Nikon Metrology* <https://www.nikonmetrology.com/en-gb/product/igps> (accessed 15 May 2019)
- [14] Wang Zheng, Mastrogiacomo Luca, Maropoulos Paul, Franceschini Fiorenzo Experimental Comparison of Dynamic Tracking Performance of iGPS and Laser Tracker 2011 *International Journal of Advanced Manufacturing Technology*
- [15] Normey-Rico J E and Camacho E F 2007 *Control of dead-time processes (Advanced textbooks in control and signal processing)* (London: Springer)
- [16] Castrounis A *Machine Learning: An In-Depth Guide - Overview, Goals, Learning Types, and Algorithms* <https://www.innoarchitech.com/blog/machine-learning-an-in-depth-non-technical-guide> (accessed 29 Jun 2019)
- [17] Valchanov I *Machine Learning: An Overview*  
<https://www.datascience.com/blog/machine-learning-overview> (accessed 29 Jun 2019)
- [18] Castle N *Supervised vs. Unsupervised Machine Learning*  
<https://www.datascience.com/blog/supervised-and-unsupervised-machine-learning-algorithms> (accessed 29 Jun 2019)
- [19] Sjöberg J, Zhang Q, Ljung L, Benveniste A, Deylon B, Glorennec P-Y, Hjalmarsson H and Juditsky A Nonlinear black-box modeling in system identification a unified overview 1995
- [20] Lennart Ljung - *System Identification\_ Theory for the User*-Prentice Hall (1999)

- 
- [21] The MathWorks I 2019 *System Identification Toolbox: User's Guide*  
[https://www.mathworks.com/help/pdf\\_doc/ident/ident.pdf](https://www.mathworks.com/help/pdf_doc/ident/ident.pdf) (accessed 30 Jun 2019)
- [22] M. Nørgaard Neural Network Based System Identification Toolbox 2000
- [23] Horváth G Neural Networks in System Identification 2002
- [24] Qin S Z, Su H T and McAvoy T J Comparison of four neural net learning methods for dynamic system identification 1992 *IEEE transactions on neural networks* **3** 122–30
- [25] CHEN S, BILLINGS S A and GRANT P M Non-linear system identification using neural networks 1990 *International Journal of Control* **51** 1191–214
- [26] Kumpati D. Narendra, Kannan Parthasarathy Neural Networks and Dynamical Systems 1992 *International Journal of Approximate Reasoning*
- [27] Arora J 2016 *Introduction to optimum design* 4th edn (Cambridge MA: Elsevier)
- [28] Mallawaarachchi V 2017 *Introduction to Genetic Algorithms — Including Example Code* <https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3> (accessed 30 Jun 2019)
- [29] The MathWorks I 2019 *Global Optimization Toolbox: User's Guide*  
[https://www.mathworks.com/help/pdf\\_doc/gads/gads\\_tb.pdf](https://www.mathworks.com/help/pdf_doc/gads/gads_tb.pdf) (accessed 30 Jun 2019)
- [30] Robotics ABB Product specification - IRB 4600
- [31] Robotics ABB Application manual - Controller software IRC5
- [32] d'Affaires C S *Newsletter details* <http://www.amrikart.com/infoletter-details/article/2013-02-11/The-iGPS-system/The-iGPS-system> (accessed 23 Jun 2019)
- [33] The MathWorks I 2019 *Deep Learning Toolbox: User's Guide*  
[https://www.mathworks.com/help/pdf\\_doc/deeplearning/nnet\\_ug.pdf](https://www.mathworks.com/help/pdf_doc/deeplearning/nnet_ug.pdf) (accessed 30 Jun 2019)
- [34] *Create state-space model, convert to state-space model - MATLAB ss*  
<https://www.mathworks.com/help/control/ref/ss.html> (accessed 26 Jun 2019)

- [35] Ali Hussein Hasan, Aleksandr N. Grachev, Saad Jabbar Abbas State space parameter estimation using online genetic algorithms 2014

## APPENDIX A Matlab Function: Fitness function to optimum tool

```

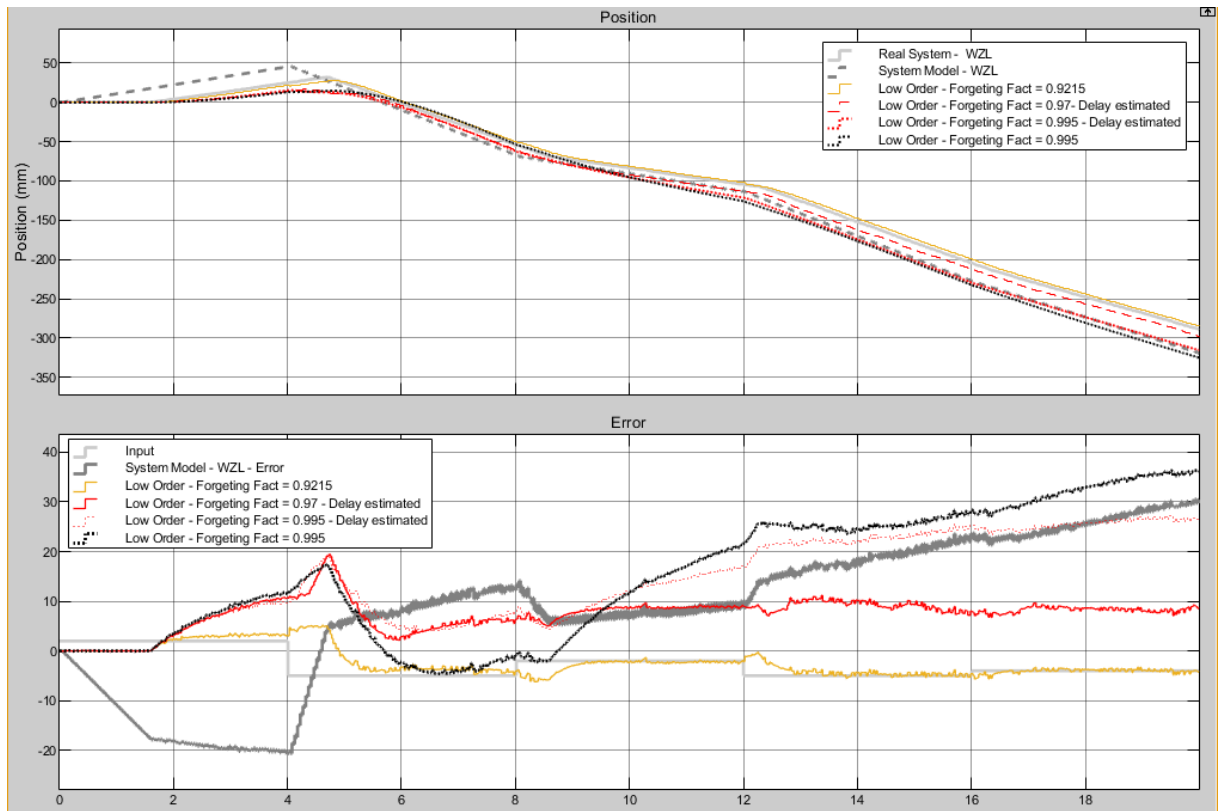
Editor - C:\Users\Arthur\Desktop\PFC\Implementações\optimizationFitness.m
EDITOR PUBLISH VIEW
optimizationFitness.m x +
1 function scores = optimizationFitness(x)
2   ts_egm = 0.004; % has to be a multiple of 4 ms
3   ts_mpc = 6*ts_egm;
4   |
5   T = 100;
6
7   %%
8   %%Transfer function of system
9   num = [5.71516496487132 1175.13344453930 97927.7870449423];
10  den = [1 -9.16532241463984e-13 17134.7282375379 2.44662370285048e-10];
11  sys = tf(num,den);
12  sysd = c2d(sys, ts_mpc); %discrete transfer function of WZL's model
13
14  %%State Space representaion
15  [A, B, C, D] = tf2ss(num, den);
16  A0 = zeros(3,3);
17  A0(1,:) = [x(1), x(2) x(3)];
18  A0(2,:) = [x(4), x(5) x(6)];
19  A0(3,:) = [x(7), x(8) x(9)];
20
21  model = ss(A0,B,C,D);
22  modeld = c2d(model, ts_mpc);
23
24  %% Genetate the input signal
25  t = 0:ts_mpc:T; % Times samples
26
27  %%Sine signal parameters
28  F = 0.04;
29  Am = 1;
30  u_sine = sin(2*pi*Am*F*t);
31
32  %%Sine input simulation
33  [y] = lsim(sysd, u_sine);
34  [y_hat] = lsim(modeld, u_sine);
35
36  %%Step input simulation
37  % y = step(sysd,t);
38  % y_hat = step(modeld,t);
39
40  scores = (1/length(t))*(sum((y_hat - y).^2));
41  end
optimizationFitness Ln 4 Col 1

```

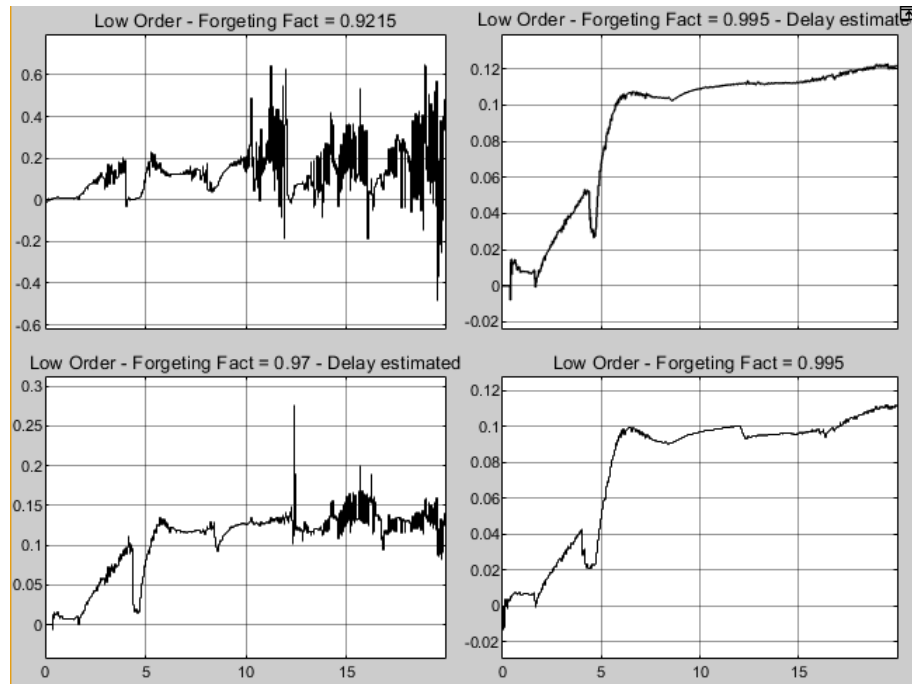




**APPENDIX B Simulation results to Forgetting Factor algorithm**



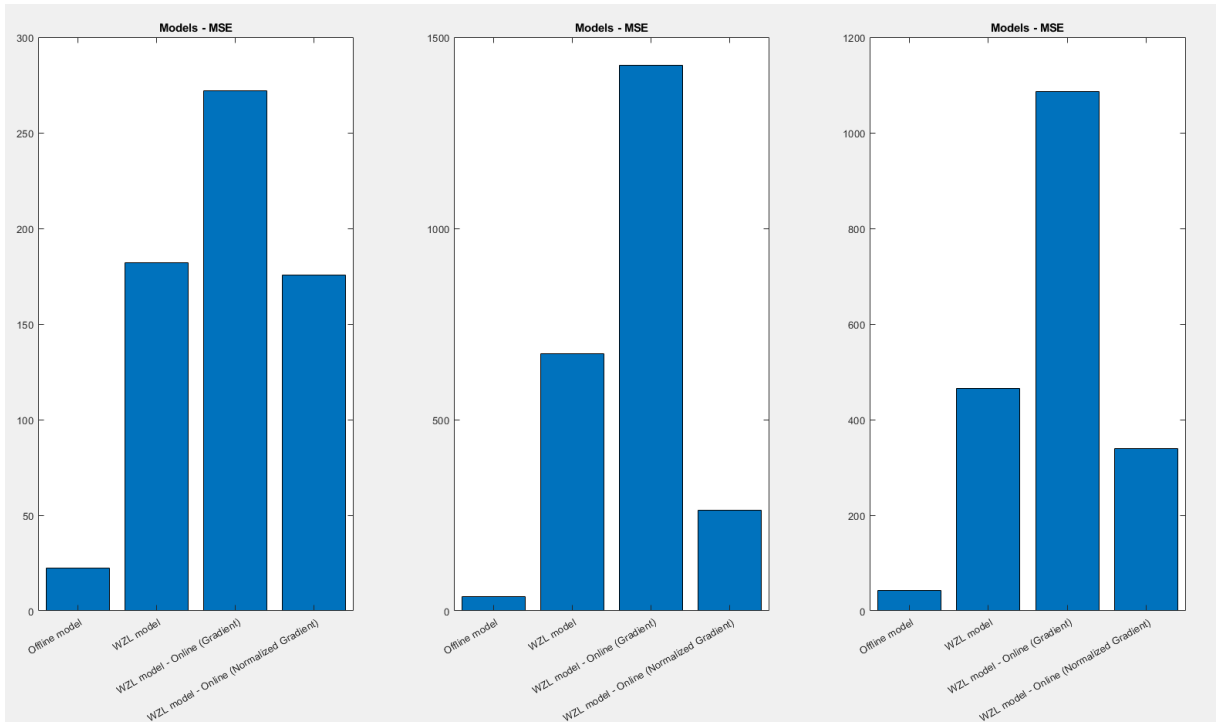
APPENDIX B I - System's and Models' outputs and errors



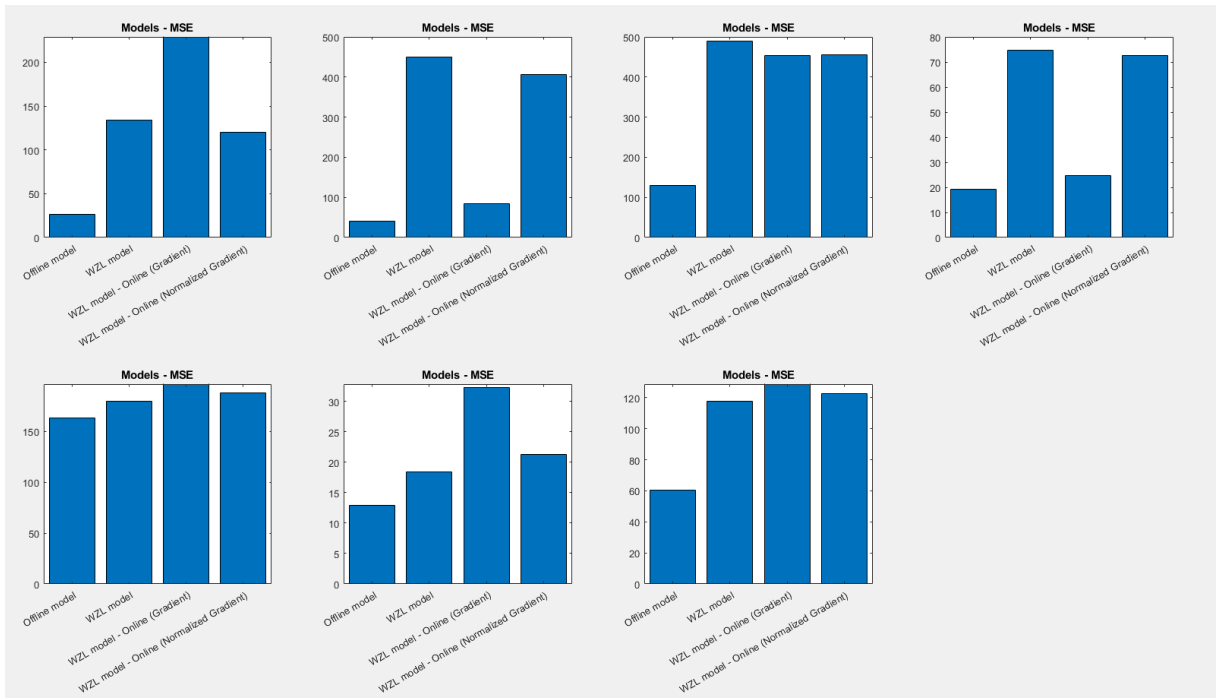
APPENDIX B II – Variation of nonzero parameters of *B* matrix



**APPENDIX C MSE Comparison: WZL's model with online estimation**



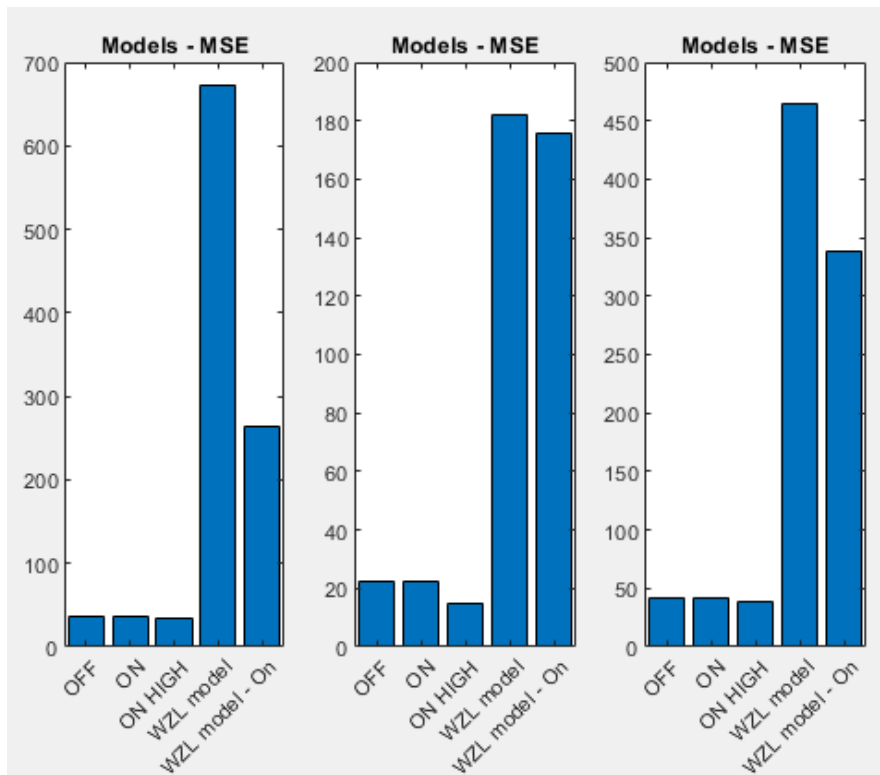
**APPENDIX C I - MSE Comparison with WZL's model with online estimation – Longer simulation**



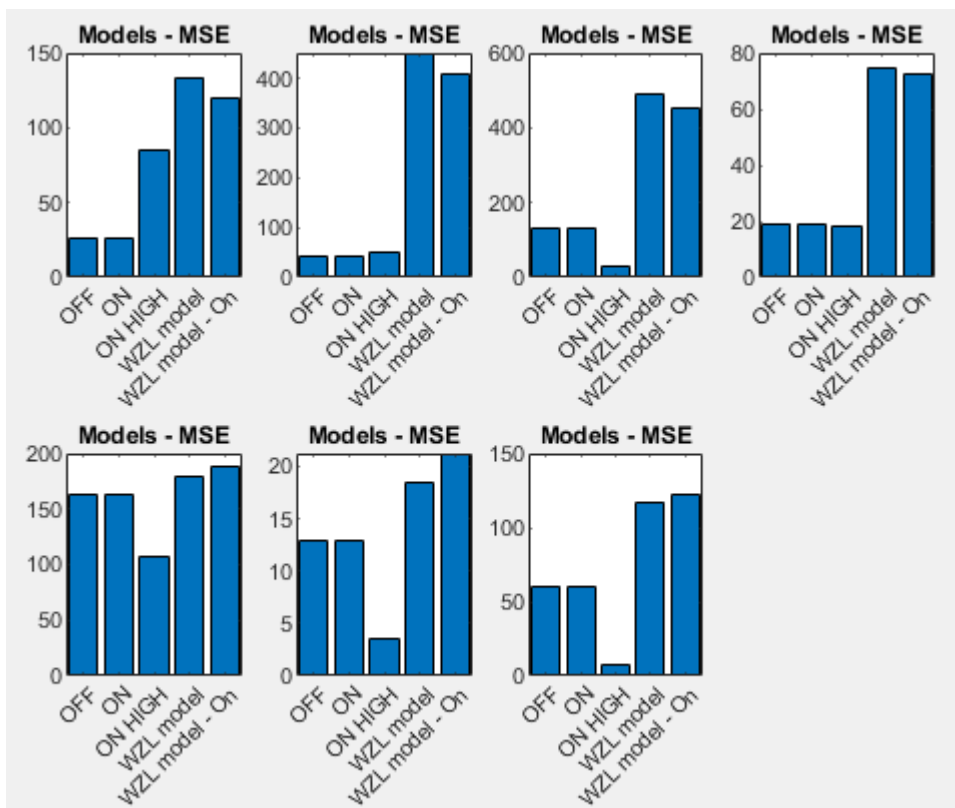
**APPENDIX C II - MSE Comparison with WZL's model with online estimation – Shorter simulation**



**APPENDIX D MSE Comparison: different orders models**



APPENDIX D I - MSE comparison in different order models – Longer simulation



APPENDIX D II - MSE comparison in different order models – Shorter simulation



## ANNEX A Neural Time Series Tool

Neural Time Series (ntstool)

**Welcome to the Neural Network Time Series app.**  
Solve a nonlinear time series problem with a dynamic neural network.

is a kind of dynamic filtering, in which past values of one or more series are used to predict future values. Dynamic neural networks which include tapped delay lines are used for nonlinear filtering and prediction.

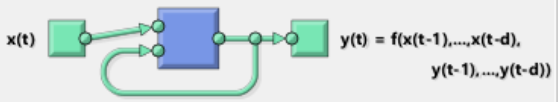
has many applications for prediction. For example, a financial analyst might use it to predict the future value of a stock, bond or other financial instrument. An engineer might want to predict the impending failure of a system.

Dynamic models are also used for system identification (or dynamic modeling), in which you build dynamic models of physical systems. These models are important for analysis, simulation, monitoring and a variety of systems, including manufacturing systems, chemical processes, robotics and aerospace systems.

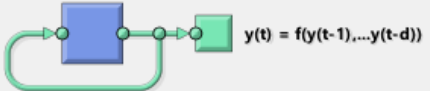
This tool allows you to solve three kinds of nonlinear time series problems in the right panel. Choose one and click [Next].

**Select a Problem**

Nonlinear Autoregressive with External (Exogenous) Input (NARX)  
Predict series  $y(t)$  given  $d$  past values of  $y(t)$  and another series  $x(t)$ .



Nonlinear Autoregressive (NAR)  
Predict series  $y(t)$  given  $d$  past values of  $y(t)$ .



Nonlinear Input-Output  
Predict series  $y(t)$  given  $d$  past values of series  $x(t)$ .

**Important Note:** NARX solutions are more accurate than this solution. On use this solution if past values of  $y(t)$  will not be available when deployed. ▾

To continue, click [Next].

Neural Network Start   Welcome   Back   **Next**   Cancel





## ANNEX B Recursive Polynomial Model Estimator Block Interface

Block Parameters: Recursive Polynomial Model Estimator

Recursive Polynomial Model Estimator

Estimate discrete-time, polynomial models of AR, ARX, ARMA, ARMAX, BJ or OE structures.

Model Structure: ARX

Parameters

Model Parameters    Algorithm and Block Options

Model Description

Initial Estimate: None

Number of Parameters in  $A(q)$  ( $n_a$ ): 1

Number of Parameters in  $B(q)$  ( $n_b$ ): 1

Input Delay ( $n_k$ ): 1

Parameter Covariance Matrix:  $1e4$

Input Processing: Sample-based

Sample Time (-1 for inherited): -1

OK    Cancel    Help    Apply



### ANNEX C System Identification user interface

