Alma Mater Studiorum – Università di Bologna

DOTTORATO DI RICERCA IN

INGEGNERIA BIOMEDICA, ELETTRICA E DEI SISTEMI

Ciclo XXXIII

Settore Concorsuale: 09 / G1 - AUTOMATICA

Settore Scientifico Disciplinare: ING-INF / 04 - AUTOMATICA

ADVANCED CONDITION MONITORING OF COMPLEX MECHATRONICS SYSTEMS BASED ON MODEL-OF-SIGNALS AND MACHINE LEARNING TECHNIQUES

Presentata da: Matteo Barbieri

Coordinatore Dottorato

Prof. Michele Monaci

Supervisore

Prof. Roberto Diversi

Co-supervisore

Prof. Andrea Tilli

Esame finale anno 2021

Abstract

Prognostics and Health Management (PHM) of machinery has become one of the pillars of Industry 4.0. The introduction of emerging technologies into the industrial world enables new models, new forms, and new methodologies to transform traditional manufacturing into intelligent manufacturing. In this context, diagnostics and prognostics of faults and their precursors has gained remarkable attention, mainly when performed autonomously by systems. The field is flourishing in academia, and researchers have published numerous PHM methodologies for machinery components.

The typical course of actions adopted to execute servicing strategies on machinery components requires significant sensor measurements, suitable data processing algorithms, and appropriate servicing choices.

Even though the industrial world is integrating more and more Information Technology solutions to keep up with Industry 4.0 new trends most of the proposed solutions do not consider standard industrial hardware and software. Modern controllers are built based on PCs and workstations hardware architectures, introducing more computational power and resources in production lines that we can take advantage of. This thesis focuses on bridging the gap in PHM between the industry and the research field, starting from Condition Monitoring and its application using modern industrial hardware. The cornerstones of this "bridge" are Model-of-Signals (MoS) and Machine Learning techniques.

MoS relies on sensor measurements to estimate machine working condition models. Those models are the result of black-box system identification theory, which provides essential rules and guidelines to calculate them properly. MoS allows the integration of PHM modules into machine controllers, exploiting their edge-computing capabilities, because of the availability of recursive estimation algorithms. Besides, Machine Learning offers the tools to perform a further refinement of the extracted information, refining data for diagnostics, prognostics, and maintenance decision-making, and we show how its integration is possible within the modern automation pyramid.

Sommario

Le tecniche di diagnosi e prognosi delle condizioni operative delle macchine automatiche, riferite come "Prognostics and Health Management" (PHM) sono uno dei pilastri fondanti le invvoative politiche di Industria 4.0. L'introduzione delle nuove tecnologie informatiche e il potenziamento delle capacità di calcolo e memoria dei computer nel mondo delle macchine automatiche sta rendendo possibile la transizione da metodi di produzione tradizionali a metodi più intelligenti, facilitati da Big-Data e intelligenza artificiale. In questo contesto, molta attenzione è rivolta alla diagnosi e prognosi autonoma dei guasti e dei loro precursori all'interno della linea di produzione. La ricerca in merito è in continua espansione con le pubblicazioni riguardanti nuove metodologie e soluzioni per il PHM di macchine automatiche che aumentano anno per anno.

Alla base di queste metodologie di gestione autonoma dello stato di salute dei sistemi troviamo il "condition monitoring", che si riferisce al monitoraggio continuo delle condizioni operative di macchine e componenti. Le informazioni ottenute attraverso il condition monitoring sono poi utilizzate per programmare la eventuale manutenzione dei sistemi. I passi da seguire per applicare queste procedure sono tipicamente tre e comprendono: l'acquisizione di dati dal sistema da monitorare attraverso sensori di campo, l'elaborazione di questi dati per renderli fruibili e "informanti" riguardo le condizioni della macchina e infine, programmare la manutenzione sulla base delle informazioni precedentemente raccolte. Tuttavia, la maggior parte delle procedure di PHM proposte non tiene conto delle soluzioni hardware e software commerciali a disposizione delle aziende, ma utilizza sistemi di acquisizione ed elaborazione che sono più convenzionali ai ricercatori. Dall'altro lato però, l'industria sta cercando di tenere il passo dell'avanzamento tecnologico, soprattuto per quanto riguarda le soluzioni IT, partendo proprio dall'hardware. I controllori di macchine di nuova generazione sono basati sulle stesse architetture di computer e workstation e hanno a disposizione una capacità di calcolo e memorizzazione senza precendenti, per non parlare dell'aumento esponenziale in termini di connettività con i livelli superiori della piramide dell'automazione.

Questa tesi di dottorato ha lo scopo di sfruttare queste nuove tecnologie a disposizione delle aziende manufatturiere per introdurre metodi di PHM nel settore, riducendo questo gap che si percepisce tra le soluzioni proposte dall'accademia e l'industria. Le tecniche su cui si fonda questo lavoro sono il metodo di monitoraggio conosciuto come Model-of-Signals (MoS) e il machine learning. I segnali misurati a bordo macchina possono essere rappresentati attraverso modelli discreti e questi, a loro volta, contengono informazioni riguardo la condizione operativa della macchina. Il metodo MoS è definito e regolato dalla teoria dell'identificazione dei sistemi, dalla quale provengono gli algoritmi di stima utilizzati per ottenere i modelli dei segnali. In particolare, il fatto che questi algoritmi di stima possano essere implementati in forma ricorsiva apre le porte all'uso dei controllori delle macchine automatiche come unità di edge-computing per porre le fondamenta per soluzioni di PHM. Poi, l'infomazione raccolta dai segnali nei modelli può essere ulteriormente elaborata attaverso tecniche di machine learning e di sintesi in base ai risultati che si vogliono ottenere con la PHM, dalla semplice rivelazione che qualcosa nel macchinario non sta funzionando a dovere alla diagnosi precoce di guasti, fino ad ottenere la predizione di un possibile malfunzionamento. Tutto ciò sempre tentendo presente le tecnologie hardware e software presenti all'interno della piramide dell'automazione.

Infine, i risultati ottenuti attraverso casi di studio e applicazioni industriali delle suddette metologie sono presentati e analizzati includendo anche i contributi teorici alla teoria dell'identificazione dei sistemi che ne hanno permesso la realizzazione.

Acknowledgements

Firstly, I would like to thank my supervisors, Prof. Roberto Diversi and Prof. Andrea Tilli, for mentoring me during my doctoral journey and guiding me with their expertise within the research field and Academia. I also want to mention Prof. Medjaher of ENIT Tarbes (France) for allowing me to work in his research group during my internship abroad period.

Then, I would like to thank all the people involved in my research projects, the colleagues at ACTEMA and LIAM Lab research groups. They provided their knowledge, expertise, and support, which was vital in achieving the dissertation's results.

I would like to thank my family for supporting me in every choice I made until this point. You were always there for me.

A big thank you also to my friends, longtime ones and colleagues, for the support and encouragement throughout this time.

Finally, a special thank you to Sara for being by my side the whole time and sharing all the ups and downs of this journey with me.

You all believed in me more than I did, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The introduction of emerging technologies, such as increasing computer computational capabilities and storage (enabled by modern computing hardware), Big-Data analysis tools, and Artificial Intelligence, into the industrial world is enabling new models, new forms, and new methodologies to transform the traditional manufacturing system into a smart system. Prognostics and Health Management (PHM) of machines has risen, in recent years, as one of the main topics within Industry 4.0 and a relevant factor in firms adopting the main concepts of Smart Factory and Intelligent Manufacturing. Systems with the possibility to autonomously perform the diagnostics and prognostics of faulty conditions are now becoming a source of value and competitive advantage for machine builders and an essential requirement for their customers.

PHM aims to provide users with an integrated view of the health state of a machine or an overall system. It should provide early detection and isolation of the precursors of incipient faults of machinery components or sub-elements. To do that, it should have the means to monitor and predict the progression of those faults. Hence, it should aid in autonomously trigger a maintenance schedule or support asset management decisions or actions. By employing such a system, unnecessary and costly preventive maintenance can be eliminated, maintenance schedules can be optimized, and lead-time for spare parts and resources can be reduced, resulting in significant cost savings (Lee et al., 2011).

With a focus on the industrial automation and manufacturing domain, the involved research field has proposed a wide variety of tools and solutions to apply PHM on machinery (Gouriveau et al., 2016; Atamuradov et al., 2017; Vogl et al., 2019). Typically, these propositions are accurately defined from a methodological perspective, but they lack guidelines for their actual design and implementation in the industrial environment. On the other hand, Industry 4.0 pushes industrial automation suppliers to develop capable hardware, suitable software tools, and ensure reliable inter-system communication to support intelligent manufacturing. In this case, the main issue is in the separation between the methodological aspect and the technological one in the industrial application of PHM. Hence, these two aspects should be simultaneously considered and developed.

The increased computing capacity of machine controllers, particularly in PC-Based Programmable Logic Controllers (PLCs), provides resources for integrating hardware and software solutions to perform PHM on traditional machinery. The intrinsic real-time features those controllers have, permit to handle, through Fieldbus, a large number of sensor measurements to capture information coming from working machine components. The enhanced computational power enables controllers to possibly house part of the processing required for PHM, working as edge-computing units. Besides, the increased connectivity allows the outsourcing of further elaborations to remote computing units, either within the machine supervising network or through cloud-computing.

Given those considerations, PHM methodologies should be studied and structured to consider the possibility of adopting machine controllers as edge-computing units running feasible information processing algorithms. Then, adopt the use of remote-computing units

within the automation pyramid to further elaborate on that information and provide health indicators and guide maintenance decision-making. Lastly, it is necessary to ensure that the systems' interconnection is sustainable by the underlying network. Consequently, the edge-remote data flow should be sustainable for and take advantage of the already in place networking solutions, such as Fieldbuses and LANs. PHM strategies that respect those aspects allow manufacturers to work under their field of expertise, exploiting available hardware and architectures.

This thesis work is devoted to the definition, the implementation, and the validation of PHM strategies that consider all those aspects mentioned above to bridge the gap between the industry and the research field. The result of the analysis and the experiments brought to the proposition of suitable solutions tailored to said available technologies, taking into account their potential and limitations.

In the following, a brief review of the state of the art of PHM methodologies related to the industrial world is presented together with the main reasoning behind MoS-based condition monitoring strategies for machine health management. Then, this work is divided into two parts: Part I "Theoretical Foundations" and Part II "Case Studies and Applications".

In the first part, the theory behind the applications of MoS-based methodologies is depicted. In Chapter 2 the tools and guidelines to exploit the Model-of-Signals technique are presented, discussing the available recursive algorithms for the edge-computing task together with newly developed ones. Then, Chapter 3 introduces, firstly, the health indicators that can be compared to MoS under the edge-computing framework and model distance metrics. Secondly, we present a description of the machine learning algorithm that we made use of during the development of PHM solutions, including Logistic Regression, Support Vector Machines, and Particle Filtering.

In the second part, case studies and applications of the proposed family of MoS-based PHM methodologies are presented together with the involved technological implementation. In Chapter 4 we describe all the projects related to simple condition monitoring and fault detection and isolation, while in Chapter 5 we discuss the methodology related to the tracking of a system performance degradation and the prognostics of incipient faults on run-to-failure examples.

1.1 Methods Review

Tracking equipment condition during operations, on-board the system, is known as Condition Monitoring (CM). It is the basic foundation of modern maintenance and it is required to step up from the fail-and-fix to the prevent-and-predict servicing approach. The knowledge of the current state of health is crucial to introduce advanced CM-based diagnostics and prognostics solutions and consequently drive maintenance.

In particular, diagnostics involves the understanding of the operational state of the machine, while prognostics introduces also the time element, involving the evolution and prediction of the said machine operational state. Both concepts support emerging (almost established, nowadays) servicing strategies, which are known as Condition-Based Maintenance (CBM) (Jardine et al., 2006), and Predictive Maintenance (PM). In general, those procedures consists of the following three main steps:

1. data acquisition;
2. data processing, modelling, and analysis;
3. maintenance decision-making.

This means that CM-based PHM procedures on machinery components require significant sensor measurements, suitable data processing algorithms, and appropriate servicing choices, either automated or with human intervention, with the former being the ultimate goal of smart manufacturing: Autonomous Health Management (AHM) systems. However, to reach this goal, CBM and, in particular, PM has to be considered. In this scenario, laying the foundations of controller-based PHM requires starting from the basics, condition monitoring, and then, looking for what combines or integrates best with the available technology.

For instance, in the literature, the majority of papers cover fault diagnostics and prognostics algorithms of machine's critical components, such as bearings, gears, drive mechanical parts, and electrical equipment (Lee et al., 2014). The main trend is to use available diagnostics and prognostics datasets or application case studies to develop PHM procedures using common tools in the research field, which usually involve National Instruments equipment with LabView for signal sampling and Matlab for data processing. However, most of the existing works do not take into account systems available to manufacturers to perform the task, neither their implementation and deployment in industrial automation platforms.

This work follows the previously mentioned trend to integrate manufacturers' tools and perspectives into the development of PHM procedures. Hereinafter, the three previously mentioned CBM steps are discussed to select suitable solutions that respond to the industrial automation framework requirements.

Data acquisition

The information collected from a system is fundamental to start any task that involves not only the diagnostics or prognostics of its components faults but also its control. Machinery can typically provide measurement data from on-board sensors, such as temperature, current, sound, and vibration, and supervisory data, such as information about products, production, and faults history. So far, The latter is the most used to define the reliability of systems and their servicing (Lee et al., 2014), which is a practice known as preventive maintenance. It relies on the statistics of the historical data regarding a component and schedules its servicing based on its mean lifetime. Besides, the use of on-board sensors to gather data characterizing the machinery's current health state is the foundation of CM-based PHM procedures (Jardine et al., 2006) and introduces the servicing of components only when it is actually needed, optimizing material waste and costs.

Those kinds of measurements (e.g., vibrations, currents and temperatures, and digital sensors of all types) are mainly used by machines to perform the logic control task of plants and for simple diagnostics, following the fail and fix approach. CM-based maintenance focuses on those quantities that are continuously sampled and linked to the operational state of a component. For instance, currents for electric motors and vibrations for mechanical parts. In the literature, this data collection procedure is traditionally performed on external equipment (e.g., National Instruments DAQs). However, in recent years, the technological development regarding sensing equipment for industrial machines provides capable I/O modules in industrial PCs. For instance, Beckhoff EL3632 and B&R X20CM4800X modules can provide vibration sensor measurements under the IEPE standard. Also, the rise of PC-based PLCs and the increased computational capacity of microcontrollers allows the handling of those data streams, even with sampling ratios of up to $50kHz$. Given this observation, the data acquisition task can be designed directly on the machine controller without the addition of external equipment, other than adding the right sensing module within the machine rack, which is easier to integrate. On the other hand, as it will be shown for electric cams, later on, some signals are already available from the devices controlled by the automatic machine, such as torques from electric drives, and their use for diagnostics does not require

further addition of sensors. Considering all these aspects, the use of machine controllers as edge-computing units results promising, the data acquisition step is covered.

Data Processing

Data processing, modelling, and analysis aim to extract useful information from sensor signals. It has a key role in the definition of what PHM can provide for the decision-making stage. The methods employed to achieve this task are usually classified into three groups, depending on how much they exploit (mathematically) the physical knowledge related to the monitored system:

- Model-Based methods.
- Data-Driven methods.
- Hybrid methods.

Model-Based methods (Isermann, 2005; Gertler, 1988), use mathematical approximations of increasing degree to build the process/system model in the input/output or state-space characterization defined from the available signals. In short, this model is used alongside the working system to compare their outputs when subjected to the same inputs to provide information about the machinery's internal state. Physical models result to be very accurate in terms of plant fault detection and isolation (Varga, 2017) and very effective in forecasting failures when also their drifting from the ideal working state is modeled. The drawback of this approach is that its application is very time-consuming due to the complexity of modeling machinery parts and their interconnections in a suitable way to perform diagnosis and prognosis. Moreover, those modeling methodologies may not be robust with respect to nominal operating behavior changes and environmental conditions, even when well thought, some approximations may not embody all the possible outcomes (i.e. thermal models vary from region to region or unexpected disturbance and clearances within the devices). On the other hand, the complexity model-based methods may result prohibitive for edge-computing in the automatic machines field. Machinery controllers' common programming languages have no libraries nor tools to run physical models in parallel with the system, except for some automation suppliers that are trying to integrate Simulink models code generation into their Integrated Development Environments (IDEs) and systems (usually exploiting C/C++ languages). Two examples, even if in the early stages of development are B&R Automation Studio with its Simulink toolbox and Beckhoff Twincat.

Data-Driven methods (Cerrada et al., 2018; Mosallam et al., 2016) exploit directly the monitoring signals to extract information to diagnose faults and/or to forecast them. This framework relies mainly on signal processing and machine learning techniques. Signal processing methods involve measurements statistics and frequency content. They are the most used by practitioners in the industry due to their fairly simple implementation in terms of time and money. In particular, statistical quantities, such as root mean square, skewness, and kurtosis value, are often used. They provide reliable indicators for diagnostics and are simple to implement within machine controllers. However, their use for FDI and prognostics results more complex, thus further refinements are required in this sense. For instance, they are either combined or used as features for machine learning algorithms to produce valuable indicators. On the other hand, signal frequency content can be extracted using tools such as the Fast Fourier Transform (FFT), the HilbertHuang Transform (HHT), and Wavelets. They carry more information about the system than the statistical ones, but in this case, their implementation in machine platforms is complex and even impracticable. There are only newly developed systems that house hardware capable of performing those elaborations without impacting drastically the CPU load. Moreover, in the last decade, neural networks, fuzzy logic,

and machine learning, and related algorithms have been used to achieve the same result. The main criticism that is moved toward those approaches is that they do not provide physical insight on the internal behavior of the system, while still providing information regarding its state of degradation because they "learned" to do so, and no expertise on the monitored system is involved in the definition of the quantities they produce for PHM. Besides, it is required to have at least one example to start the learning process.

Hybrid methods or fusion methods exploit both the previously mentioned ones, combining various techniques to achieve PHM. One approach is to start with the computations of approximate models of the system and then use data-driven techniques to fill the gap between the model uncertainties and the real system, also called gray-box methods (Tulleken, 1993). In other cases, the two methods are used in parallel and their output is combined to obtain useful information for diagnosis and forecasting.

In this dissertation, the Model-of-Signals (MoS) method (Isermann, 2006), based on black-box system identification theory (Söderström et al., 1989), is used as the condition monitoring method and is the foundation of many applications and case studies involving the use of industrial automation hardware and software for PHM deployment to automatic machines that we developed (Barbieri et al., 2018; Barbieri et al., 2019b; Barbieri et al., 2019a; Barbieri et al., 2020b). This method can be considered as a hybrid one because it relies on mathematical expression to model the machinery measurements but in a more data-driven perspective. System identification rules and guidelines allow MoS to provide more inherent information about the system than statistical quantities, and given its nature, it embodies also the frequency content of those modeled signals (i.e., it estimates discrete transfer functions from which the corresponding signal spectrum can be computed). On the practitioner side, this method permits to compress the data contained within the signal stream into model parameters, allowing better handling of their information, and, with the addition of further elaboration, provide significant quantities for PHM. MoS can be seen as an empowered data-driven approach that is effective within the industrial automation pyramid structure since it is designed considering this infrastructure.

Maintenance Decision-Making

At this stage, all the information produced via data processing methods is collected in, what the literature calls, system Health Indicators (HIs) (Atamuradov et al., 2020). The goal of those indicators is to drive maintenance decision-making, which is exactly the purpose of PHM solutions. Typically, health indicators can be discrete or continuous values, from simple binary fault-no fault outputs and component fault severity levels to component degradation trackers and Remaining Useful Life predictors. PHM information is, then, exploited either by human operators or with automated procedures. The latter nowadays has become a fascinating subject that is receiving increasing attention from researchers and firms, pursuing Autonomous Health Management (AHM) systems.

1.2 Proposed Course of Actions

Smart manufacturing sites typically use controllers to handle the production process supervised by LAN-connected PCs and the site mainframes, following the typical structure of the automation pyramid. The technological development of those devices enables the integration of PHM solutions alongside logic control and supervisory tasks. Given these considerations, together with the previously mentioned brief review, this work proposes efficient and practical methodologies, providing guidelines to integrate automated PHM modules in machine controllers while exploiting standard industrial platforms and architectures.

The proposed family of PHM solutions is centered around using machine controllers as edge-computing units and bases its data processing on the Model-of-Signals approach. On top of that, information refining methods provide PHM knowledge to the automation pyramid upper levels. The reference example that can conceptualize the proposed solutions is the most widespread architecture present in production lines: PC supervised PLCs. The general course of action is simple and follows the main health management steps previously depicted.

1. The controller handles the data acquisition task to gather information from the monitored system using available industrial sensing equipment. Device measurements are provided to the main unit through Fieldbus protocols.

2. Despite PHM's goal, being it fault detection and isolation or prognostics with RUL prediction, the first data processing step is about applying Model-of-Signals to the acquired data. Then, the computed models can be refined to produce HI for PHM. This second level can be either computationally simple and housed still within the controller or outsourced to an upper level of the automation pyramid where a more powerful computer can be used.

3. On top of the produced HI, PHM information is defined and provided to the management level for maintenance decision making.

Following these guidelines and the reference example architecture, PC supervised PLCs; this dissertation explores and studies the potentialities of Model-of-Signals within industrial automation. The data used to develop this study are obtained from public benchmark datasets, such as bearing data from (NASA, 2019) and (CWRU, 2014) providing vibration measurements for various failing scenarios, as well as from industrial application case studies where vibrations, currents, and torques are investigated. The monitored components involve bearings, gearboxes, and electric-cam-driven mechanisms.

The contributions provided to the research field cover all the aspects of the presented methodology, both experimental and theoretical. The starting point of this project is the work developed in (Barbieri, 2017) with the definition of the MoS software library for machine controllers. The key health indicators in MoS are model distances. They provide information on how far the current estimated signal model is from a known given reference one. Besides, the model parameters themselves are a valuable set of features that can be exploited. For this reason, various techniques have been studied in combination with MoS, considering the nature of the applications involved and the PHM goal for the particular case study, keeping the technological aspects in mind. MoS-based performance in degradation tracking has been analyzed by comparing model distances and statistical health indicators for gearbox lubrication tracking application (Barbieri et al., 2019a) and for fault detection and isolation using the Logistic Regression (LR) algorithm on a bearing benchmark dataset (CWRU, 2014; Barbieri et al., 2019b). Then, MoS has been tested for fault detection in electric cams introducing a new way to exploit the way mechanism trajectories are designed (Barbieri et al., 2020b). Finally, MoS in combination with SVM to track the play of a paper feeding mechanism (Barbieri et al., 2020a) and with Particle Filtering (PF) to predict bearing (NASA, 2019) Remaining Useful Life (RUL) (Barbieri et al., 2020d).

On the other hand, the adaptation of black-box system identification tools to fulfill the expected MoS properties required the analysis of the current research state of the art. In this fashion, the contributions to the field include: firstly, a novel way to approach mechanisms driven with polynomial trajectories, using ARMA models to retrieve information about the component health state (Barbieri et al., 2020b). Secondly, a new recursive identification algorithm for the estimation of noisy AR signals, which is usually the case for cheap accelerometers, and finally, the definition of a new estimation algorithm for noisy-input-output Finite Impulse Response (FIR) (Barbieri et al., 2020c).

Part I

Theoretical Foundations

Chapter 2

Model-of-Signals

Model-of-Signals (MoS) is a condition monitoring technique that refers to the use of black-box system identification (Söderström et al., 1989) to define and estimate a particular model structure based on a given measurement or set of measurements. Isermann formalized this approach in his book (Isermann, 2006, Chapter 9) following the reasoning that the signals models obtained from a system are useful for diagnostics purposes. Among the various system identification techniques, the case of equation error methods is considered to derive the underlying signal model, under the assumption that it is linear in the parameters. In this scenario, the continuous estimation of the signal models may provide information about the system through their parameters evolution. We took advantage of this concept and developed the use of the Model-of-Signals technique. The sensor signals available in machines, such as currents, vibrations, temperatures, and sounds can be modeled in this framework. Moreover, the availability of recursive estimation algorithms and their computationally efficient forms allow the deployment of such a tool directly in machine controllers (Barbieri, 2017), enabling more involved PHM solution starting from the edge, where measurements are inherently handled. Not only, but the use of Model-of-Signals permits the compression of signal information (as shown later on), enabling a manageable information flow toward higher levels of the automation pyramid. This integration is the core of the PHM course-of-actions presented in this thesis, unlocking the remote computation level. As depicted in the overall procedure section, models can be either used to produce machine health indicators or as input features to machine learning techniques to produce relevant information for PHM and the linked maintenance strategies. To summarize, the properties of Model-of-Signal for PHM are:

- the availability of recursive algorithms for model estimation permits the direct implementation on machine controllers, hence exploiting their edge-computing capabilities.
- allowing the compression of signals information into models, that are easier to handle, and facilitating the distribution of computing loads within the automation pyramid to increase its computational capacities in real-time.
- the developed mathematical models can carry inherent information about system physical characteristics, that is useful to improve the diagnostics and prognostics results.
- models are readily available features for the definition of health indicators, using machine learning related algorithms or other synthesis methods.

This work follows the notion of monitoring by exploiting MoS and controllers as edge-computing units, and in this chapter, we present the theory behind its implementation. The key element that is considered here is the availability of recursive estimation algorithms due to their properties for edge-computing. The way sensor measurements are pre-processed or adapted to fall within the system identification framework will be dealt with in the second

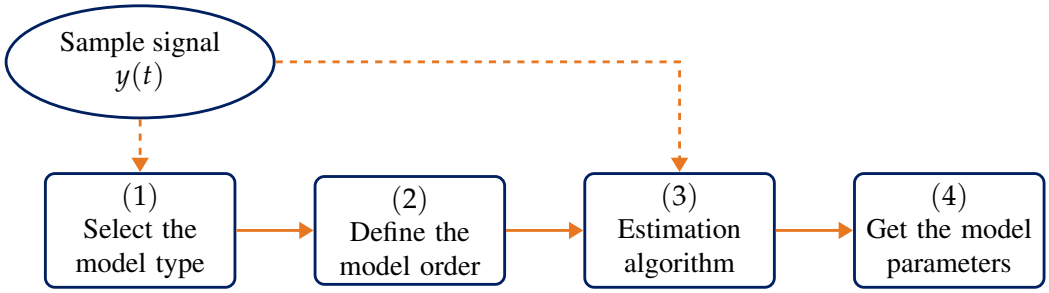

FIGURE 2.1: Summary of Model-of-Signals.

part of this thesis, which is about the various case studies and applications of MoS-based PHM.

Figure 2.1 shows a summary of the main steps to execute MoS following system identification theory and guidelines. It starts with the sampling of the available signal for which the appropriate underlying model parameter structure and the proper model order are selected. Then, the estimation algorithm is defined depending on the picked model structure. In the end, the model's parameters are obtained by injecting the sampled signal into the chosen estimation algorithm.

Before discussing the MoS monitoring problem and its application into case studies, we need to introduce black-box system identification tools and guidelines. The discussion hereinafter refers to this part of discrete-time system identification theory, which is described in detail in (Söderström et al., 1989) and (Ljung, 1999). We describe standard and newly developed estimation algorithms employed to define MoS solutions, with a focus on their recursive forms and how they are suited for the implementation on machine controllers. The first sections will introduce the basics needed to understand the presented recursive algorithms, so readers with expertise in system identification may jump to sections 2.5 and 2.6, where a newly developed recursive algorithm for noisy AR processes and a noisy FIR estimator are presented.

2.1 The Identification Problem

Once the sensor signals are handled and the underlying model structure is selected, the focus shifts toward the estimation of the parameters of such a model. Following this reasoning, and coherently with the MoS applications that will be shown later on, it is assumed that we want to identify an AutoRegressive (AR) process, which is the approximation that typically fits the most signals coming from a working machine in this dissertation. In particular, vibrations are well approximated by AR models since it is possible to assume they are driven by the "operational noise" working machinery generates during production (Mechefske et al., 1992). Then, also currents will be modelled as AR processes, but in some cases, signal pre-processing is involved before the use of the estimation algorithm.

The identification of AR models starts in the framework of equation error models, which means the process is represented in the following way:

$$y(t) + a_1\, y(t-1) + a_2\, y(t-2) + \cdots + a_n\, y(t-n) = e(t) \tag{2.1}$$

or in its compact form:

$$y(t) = \frac{1}{A(z^{-1})} e(t) \tag{2.2}$$

where $e(t)$ in eq. (2.1) is the equation error, that in this case corresponds to the driving noise, a zero-mean white process with variance σ_e^2 and n is the AR system model order. The matrix $A(z^{-1})$ contains the AR system transfer function

$$A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n} \tag{2.3}$$

where z^{-k}, with $k \in \mathbb{N}$, is the backward-shift operator, i.e. $z^{-1}e(t) = e(t-1)$ for $k = 1$, indicating the corresponding sample delay. Then, by casting the model represention of eqs. (2.1) and (2.3) in regression form it is possible to state the identification problem. Thus, let

$$y(t) = \varphi_y^T(t)\theta^* + e(t) \tag{2.4}$$

with

$$\varphi_y(t) = [-y(t-1) \ \ldots \ -y(t-n)]^T \tag{2.5}$$

being the regressor of signal $y(t)$ and

$$\theta^* = [a_1 \ \ldots \ a_n]^T \tag{2.6}$$

being the parameter vector that we want to estimate. Then, the following holds.

Problem 2.1. *Given a set of N measurements of the output $y(t)$ estimate the coefficients $a_1, a_2 \ \ldots \ a_n$ of the AR model, i.e., find θ^*.*

The notation with superscript "$*$", i.e., θ^*, denotes the true parameters of the system, while θ or $\hat{\theta}$ typically denotes its estimate (one of them is used depending on the context). Notice that in this the model order n is assumed known. It is also worth noting that in MoS, the vector θ is the one we refer to when speaking of the model of the signal, and it contains the information we rely on for PHM. The following discussion introduces the solution to the problem and then its recursive forms.

2.2 The Recursive Least Squares

As stated in (Söderström et al., 1989, Chapter 4) the most popular solution of problem 2.1 is defined as the Least Squares (LS) solution and it is the result of an optimization problem. In general, the measurements available to estimate the underlying model suffer from uncertainties, disturbances, and model misfits. In this regard, the availability of a set of measurements allows us to solve an overdetermined problem where we have $N - n$ equations and get an estimate as close as possible to the true model θ^*. We start by pointing out the equation error from eq. (2.4), namely

$$\varepsilon(t) = y(t) - \varphi_y^T(t)\theta, \tag{2.7}$$

which can be computed for each of the $N - n$ observations available and collected as:

$$\varepsilon = [\varepsilon(1)\, \varepsilon(2) \ \cdots \ \varepsilon(N)]^T \tag{2.8}$$

which is also called as the residuals vector, particularly in diagnostics. Then, a compact version of the computation of the residuals of our problem can be written as:

$$\varepsilon = Y - H_y\theta \tag{2.9}$$

where

$$H_y = \begin{bmatrix} y(n) & \cdots & y(1) \\ \vdots & \ddots & \vdots \\ y(N-1) & \cdots & y(N-n) \end{bmatrix} = \begin{bmatrix} \varphi_y(n+1) \\ \vdots \\ \varphi_y(N) \end{bmatrix} \tag{2.10}$$

is defined as the Hankel matrix of the regressor, and Y is the vector containing the samples of $y(t)$ from $n+1$ to N. Finally, the cost function upon with the optimization problem is casted is the following:

$$J(\theta) = \frac{1}{2}\sum_{t=n+1}^{N} \varepsilon^2(t) = \frac{1}{2}\varepsilon^T\varepsilon, \tag{2.11}$$

It takes into account all the measurements involved in the identification problem and is defined in terms of the equation error. Then, with the substitution of ε as in eq. (2.8), one obtains:

$$\min_{\theta} J(\theta) = \frac{1}{2}\left(Y - H_y\theta\right)^T\left(Y - H_y\theta\right), \tag{2.12}$$

which indicates that the parameter vector θ, producing the least amount (in magnitude) of squared residuals, is the best estimation of the true model and it is nothing but the least-squares solution of the problem.

Lemma 2.1. *Given the cost function $J(\theta)$ of eq. (2.12) and assuming that the matrix $H_y^T H_y$ is positive definite. Then $J(\theta)$ has a unique minimum point given by*

$$\hat{\theta} = \left(H_y^T H_y\right)^{-1} H_y^T Y. \tag{2.13}$$

For a detailed proof of lemma 2.1, the reader should refer to (Söderström et al., 1989, Chapter 4).

Remark 2.1. *Under the assumption that the AR system is driven by an ergodic white process and assuming we are able to collect infinite observations of $y(t)$:*

$$\lim_{N\to\infty} \hat{\theta} = \theta^*, \tag{2.14}$$

the estimated model will tend to the true model, which means that the LS estimator is consistent when estimating AR models (and also AR eXogenous ones) driven by white noise.

Remark 2.2. *The LS estimate in eq. (2.13) may be rewritten in the following equivalent form*

$$\hat{\theta} = \left[\sum_{t=n+1}^{N} \varphi_y(t)\varphi_y^{\mathrm{T}}(t)\right]^{-1}\left[\sum_{t=n+1}^{N} \varphi_y(t)y(t)\right] \tag{2.15}$$

This form, despite being less preferred than eq. (2.13) (e.g., MATLAB), is far more interesting in the MoS framework. This formulation does not require big matrices, such as H, to be computed and is the basis for the definition of the recursive forms. This is crucial for edge-computing since the amount of system memory involved in these computations depends only on the model order n, and obviously on the computational power of the CPU and on how the sensor measurements are handled.

The RLS algorithm computes the model parameter estimate $\hat{\theta}$ recursively in time. More precisely, the estimate $\hat{\theta}(t)$ at time t, is obtained on the basis of the previous estimate $\hat{\theta}(t-1)$, the current measurement sample $y(t)$ and the last n sensor measurements $y(t-1),\ldots,y(t-n)$. The LS algorithm can be computed in a recursive fashion starting from eq. (2.15), adjusted for the purpose as

$$\hat{\theta}(t) = \left[\sum_{s=t_0}^{t} \varphi_y(s)\varphi_y^{\mathrm{T}}(s)\right]^{-1}\left[\sum_{s=t_0}^{t} \varphi_y(s)y(s)\right], \tag{2.16}$$

and by defining the following matrix

$$P(t) = \left[\sum_{s=t_0}^{t} \varphi_y(s)\varphi_y^T(s)\right]^{-1}. \tag{2.17}$$

Since eq. (2.17) can be reformulated as

$$P^{-1}(t) = P^{-1}(t-1) + \varphi_y(t)\varphi_y^T(t). \tag{2.18}$$

it follows that

$$\begin{aligned}\hat{\theta}(t) &= P(t)\left[\sum_{s=t_0}^{t-1} \varphi_y(s)y(s) + \varphi_y(t)y(t)\right] \\ &= P(t)\left[P^{-1}(t-1)\hat{\theta}(t-1) + \varphi_y(t)y(t)\right] \\ &= \hat{\theta}(t-1) + P(t)\varphi_y(t)\left[y(t) - \varphi_y^T(t)\hat{\theta}(t-1)\right]\end{aligned} \tag{2.19}$$

which allows us to define the following recursive estimation algorithm

Algorithm 2.1 (RLS I)**.** *Once a new sample of $y(t)$ is available*

1. *Compute the update of $P^{-1}(t) = P^{-1}(t-1) + \varphi_y(t)\varphi_y^T(t)$.*
2. *Invert $P^{-1}(t)$.*
3. *Compute the equation error $\varepsilon(t) = y(t) - \varphi_y^T(t)\,\hat{\theta}(t-1)$.*
4. *Compute the gain factor $K(t) = P(t)\,\varphi_y(t)$.*
5. *Finally, compute the new estimate $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$.*

Remark 2.3. *The RLS algorithm requires the quantities $\hat{\theta}(t_0)$ and $P(t_0)$ as a starting point. They are typically initialized as $P^{-1}(t_0) = \alpha I_n$ and $\hat{\theta}(t_0) = \mathbb{1}$, where $\alpha > 0$ is a scalar and $\mathbb{1}$ is a $n \times 1$ vector whose entries are all equal to 1.*

However, this formulation of the RLS algorithm is not enough for the implementation of machine controllers. The inverse of a matrix requires computational resources that are not available in the platform where we expect the algorithm deployment. The solution to this issue is to avoid the computation of matrix inversions and this is possible by using the Woodbury Identity, also known as the matrix inversion lemma.

Lemma 2.2 (Woodbury Identity)**.**

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1} \tag{2.20}$$

This allows the reformulation of the update of $P(t)$ as

$$P(t) = P(t-1) - \frac{P(t-1)\varphi_y(t)\varphi_y^T(t)P(t-1)}{1 + \varphi_y^T(t)P(t-1)\varphi_y(t)}, \tag{2.21}$$

and the definition of an improved form of RLS.

Algorithm 2.2 (RLS II)**.** *Once a new sample of $y(t)$ is available*

1. *Compute the update of $P(t) = P(t-1) - \frac{P(t-1)\varphi_y(t)\varphi_y^T(t)P(t-1)}{1+\varphi_y^T(t)P(t-1)\varphi_y(t)}$.*

2. *Compute the equation error* $\varepsilon(t) = y(t) - \varphi_y^T(t)\,\hat{\theta}(t-1)$.
3. *Compute the gain factor* $K(t) = P(t)\,\varphi_y(t)$.
4. *Finally, compute the new estimate* $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$.

It is also worth introducing two more refined versions of the algorithm that tackle the fact that $P^{-1}(t)$ grows to infinite as time passes. Both involve the introduction of terms in the LS cost function, enabling robustness in the corresponding RLS forms. The first introduces the time weighting factor, so that

$$J(\theta(t)) = \frac{1}{2}\left(\frac{1}{t}\sum_{s=t_0}^{t}\varepsilon^2(t)\right), \tag{2.22}$$

where, if $t_0 = 1$, the result of $\frac{1}{t}\sum_{s=t_0}^{t}\varepsilon^2(t)$ can be seen as the experimental variance of the residual $\varepsilon(t)$, and tends to σ_e^2 for $t \to \infty$. In particular

$$\lim_{t\to\infty}\frac{1}{t}\sum_{s=t_0}^{t}\varepsilon^2(t) = E\left[e^2(t)\right] \tag{2.23}$$

where $E[\cdot]$ is the statistical expectation operator, which refers, in this case, to the autocorrelation of the residuals. Finally, after computations analogous to eqs. (2.18) and (2.19) one obtains:

Algorithm 2.3 (RLS III)**.** *Once a new sample of* $y(t)$ *is available*

1. *Compute the update of* $P(t) = \frac{t\,P(t-1)}{t-1}\left[I_n - \frac{\varphi(t)\varphi^T(t)P(t-1)}{t-1+\varphi^T(t)P(t-1)\varphi(t)}\right]$.
2. *Compute the equation error* $\varepsilon(t) = y(t) - \varphi_y^T(t)\,\hat{\theta}(t-1)$.
3. *Compute the gain factor* $K(t) = \frac{1}{t}P(t)\,\varphi_y(t)$.
4. *Finally, compute the new estimate* $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$.

This version is particularly useful when the real-time controller has enough memory to save an array of sensor measurements, but not enough CPU power to process each measurement with the frequency it is sampled. In this situation, the best solution is to store a suitable amount of measurements and then process them in the background alongside the control task, allowing the estimation to be still on-line, but not in real-time. In terms of MoS, this is possible because models generation is applied to monitor degradation that occurs slowly over time with respect to signal sampling and processing. This RLS form is typically more robust than algorithm 2.2.

The PHM solution may require a continuous generation of models when new sensor samples are available. This is done typically to track θ coefficients evolution in time when the algorithm is also involved in control tasks, not only in diagnostics. A fourth version of the RLS algorithm is added, this one is able to weigh more recent observations of the system with respect to past ones, enabling the tacking of the model parameter variations continuously. Firstly, the cost function is defined as

$$J(\theta(t)) = \frac{1}{2}\sum_{s=t_0}^{t}\lambda^{t-s}\varepsilon^2(t), \tag{2.24}$$

where λ is seen as the forgetting factor, usually within the range $[0.95 - 0.99]$, and the corresponding RLS version is the following

Algorithm 2.4 (RLS IV). *Once a new sample of $y(t)$ is available*

1. *Compute the update of* $P(t) = \frac{P(t-1)}{\lambda}\left[I_n - \frac{\varphi(t)\varphi^T(t)P(t-1)}{\lambda+\varphi^T(t)P(t-1)\varphi(t)}\right]$.
2. *Compute the equation error* $\varepsilon(t) = y(t) - \varphi_y^T(t)\,\hat{\theta}(t-1)$.
3. *Compute the gain factor* $K(t) = P(t)\,\varphi_y(t)$.
4. *Finally, compute the new estimate* $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\varepsilon(t)$.

Remark 2.4. *Algorithm 2.4 with* $\lambda = 1$ *is equivalent to algorithm 2.2.*

The main difference, beyond the real-time hardware constraints, between algorithm 2.3 and algorithm 2.4 when used in MoS depends on the pace, or "sampling time", with which models are generated, and that is function of the adopted PHM solution. For instance, the case studies proposed in the next part of this dissertation make use of algorithm 2.3.

The derived algorithm formulations are now suitable for machine controllers implementation and their on-line use during operations. As shown in (Barbieri, 2017), the algorithm is numerically validated showing the same results when evaluated within the MATLAB environment as well as on the machine controller.

2.3 The Overdetermined Recursive Instrumental Variable

A second algorithm, whose recursive forms are particularly useful for MoS, is the Instrumental Variable (IV). Its solution involves the use of a vector of instruments φ_i in eq. (2.15) in place of the standard LS regressor.

$$\hat{\theta}_{IV} = \left[\sum_{\tau=n+1}^{N} \varphi_i(\tau)\varphi_y^{\mathrm{T}}(\tau)\right]^{-1}\left[\sum_{\tau=n+1}^{N} \varphi_i(\tau)y(\tau)\right]. \tag{2.25}$$

This solution is particularly used when $e(t)$ is not white and the LS estimation loses consistency in this regard. On the other hand, the IV method can be seen as a generalized version of the LS algorithm, since they coincide when $\varphi_i(t) = \varphi_y(t)$. In this fashion, the vector of instruments should be assembled so that it is uncorrelated with $e(t)$, which means

$$E\left[\varphi_i(t)e(t)\right] = 0 \tag{2.26}$$

The recursive forms of the standard IV algorithm are practically identical to algorithms 2.1 to 2.2 and 2.4, but with the substitution of $\varphi_i(t)$ into $\varphi_y(t)$. The identification problem is the same as problem 2.1, but in this case, the assumption on $e(t)$ that it is a zero-mean process and that eq. (2.26) holds. Besides, make sure that the matrix

$$\left[\sum_{\tau=n+1}^{N} \varphi_i(\tau)\varphi_y^{\mathrm{T}}(\tau)\right] \tag{2.27}$$

is always invertible. For a thorough discussion on IV and Recursive IV refer to (Söderström et al., 1989, Chapter 8 and 9).

Here, in this dissertation, the focus is on the more general form of the IV method, the Overdetermined Instrumental Variable (OIV) and its recursive forms, that introduces the addition of q elements to the instruments vector $\varphi_i(t)$, turning it into an overdetermined problem with an overdetermined solution. Thus, the augmented instruments vector is obtained as

follows

$$\begin{aligned}\bar{\varphi}_i(t) = [\, &-i(t-1) \ \ldots \ -i(t-n) \\ &-i(t-n-1) \ \ldots \ -i(t-n-q)]^T,\end{aligned} \tag{2.28}$$

The main reasons behind the use of this method are two and involve AR and ARMA processes (AutoRegressive Moving Average). Firstly. the recursive versions of the OIV method have increased robustness with the tradeoff of a slightly increased computational burden. The estimation of AR models can be improved by assuming $\bar{\varphi}_y(t)$ as the vector of instrument s$\bar{\varphi}_i(t)$. On the other hand, as will be shown later with the case of current measurements, this algorithm can be employed to identify the AR part of an ARMA process by means of a suitably chosen vector of instruments.

Given those considerations, and following (Friedlander, 1984) reasoning about the derivation of the ORIV method, we define

$$\hat{\theta}_{OIV} = \left[\sum_{s=n+q}^{N} \bar{\varphi}_i(s)\varphi_y^{\mathrm{T}}(s)\right]^{\dagger} \left[\sum_{s=n+q}^{N} \bar{\varphi}_i(s)y(s)\right], \tag{2.29}$$

where the first term is nothing but the pseudo-inverse of a tall matrix. Now, eq. (2.29) can be rewritten in order to introduce the recursive version of the OIV algorithm in the following way (we drop the $_{OIV}$ subscript for brevity):

$$\hat{\theta}(t) = P(t)R^{\mathrm{T}}(t)\rho(t) \tag{2.30}$$

with

$$\rho(t) = \sum_{s=1}^{t} \bar{\varphi}_i(s)y(s) \tag{2.31}$$

$$R(t) = \sum_{s=1}^{t} \bar{\varphi}_i(s)\varphi_y^{\mathrm{T}}(s) \tag{2.32}$$

$$P(t) = \left[R^{\mathrm{T}}(t)R(t)\right]^{-1} \tag{2.33}$$

Then, recursive algorithm is obtained as follows, starting from:

$$\hat{\theta}(t) = \hat{\theta}(t-1) + P(t)R^{\mathrm{T}}(t)[\rho(t) - R(t)\hat{\theta}(t-1)]. \tag{2.34}$$

Then, we analyse the second term starting from $P^{-1}(t)$,

$$\begin{aligned} P^{-1}(t) &= \left[R^{\mathrm{T}}(t-1) + \varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\right]\left[R(t-1) + \bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)\right] \\ &= P^{-1}(t-1) + \varphi_y(t)w^{\mathrm{T}}(t) + w(t)\varphi_y^{\mathrm{T}}(t) + \varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t) \\ &= P^{-1}(t-1) + \begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix} \begin{pmatrix} w^{\mathrm{T}}(t) \\ \varphi_y^{\mathrm{T}}(t) \end{pmatrix} \\ &= P^{-1}(t-1) + \phi(t)\Lambda^{-1}(t)\phi^{\mathrm{T}}(t) \end{aligned} \tag{2.35}$$

with

$$w(t) = R^{\mathrm{T}}(t-1)\bar{\varphi}_i(t) \tag{2.36}$$

$$\phi(t) = \begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix} \tag{2.37}$$

$$\Lambda^{-1}(t) = \begin{pmatrix} 0 & 1 \\ 1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix} \tag{2.38}$$

Then, as explained before, in MoS we want to avoid the use of matrix inversions, so through the matrix inversion lemma 2.2 we get the recursion using directly $P(t)$

$$P(t) = P(t-1) - P(t-1)\phi(t)\left[\Lambda(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right]^{-1}\phi^{\mathrm{T}}(t)P(t-1) \tag{2.39}$$

In the end, by following an analogous reasoning the second part of the term under analysis in eq. (2.34) becomes

$$\begin{aligned}
R^{\mathrm{T}}(t)[\rho(t) - R(t)\hat{\theta}(t-1)] = \\
&= \left[R^{\mathrm{T}}(t-1) + \varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\right]\{\rho(t-1) + \bar{\varphi}_i(t)y(t) \\
&\quad - \left[R(t-1) + \bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)\right]\hat{\theta}(t-1)\} \\
&= \left(\varphi_y(t) \quad R^{\mathrm{T}}(t-1)\bar{\varphi}_i(t) + \varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t)\right) \\
&\quad \times \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\{\rho(t-1) - R(t-1)\hat{\theta}(t-1)\} \\ y(t) - \varphi_y^{\mathrm{T}}(t)\hat{\theta}(t-1) \end{pmatrix} \\
&= \begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix} \\
&\quad \times \left\{ \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\rho(t-1) \\ y(t) \end{pmatrix} - \begin{pmatrix} w^{\mathrm{T}}(t) \\ \varphi_y(t) \end{pmatrix} \hat{\theta}(t-1) \right\} \\
&= \phi(t)\Lambda^{-1}(t)\left[v(t) - \phi^{\mathrm{T}}(t)\hat{\theta}(t-1)\right]
\end{aligned} \tag{2.40}$$

with

$$v(t) = \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\rho(t-1) \\ y(t) \end{pmatrix} \tag{2.41}$$

Finally, by plugging eq. (2.40) in eq. (2.34) and, by making use of the Push-through identity

Lemma 2.3 (Push-through identity)**.**

$$(A + UCV)^{-1}U = A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}C^{-1} \tag{2.42}$$

from which one gets

$$P(t)\phi(t) = P(t-1)\phi(t)\left[\Lambda(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right]^{-1}\Lambda(t), \tag{2.43}$$

the Overdetermined Recursive Instrumental Variable (ORIV) is, then, the following:

Algorithm 2.5 (ORIV I)**.** *Once a new sample of $y(t)$ is available together with the corresponding instrument $i(t)$, compute*

1. $w(t) = R^{\mathrm{T}}(t-1)\bar{\varphi}_i(t)$

2. $\phi(t) = (w(t) \quad \varphi_y(t))$
3. $\Lambda(t) = \begin{pmatrix} -\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) & 1 \\ 1 & 0 \end{pmatrix}$
4. $v(t) = \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\rho(t-1) \\ y(t) \end{pmatrix}$
5. $K(t) = P(t-1)\phi(t)\left[\Lambda(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right]^{-1}$
6. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\left[v(t) - \phi^{\mathrm{T}}(t)\hat{\theta}(t-1)\right]$
7. $R(t) = R(t-1) + \bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)$
8. $\rho(t) = \rho(t-1) + \bar{\varphi}_i(t)y(t)$
9. $P(t) = P(t-1) - K(t)\phi^{\mathrm{T}}(t)P(t-1)$

Where its initialization has to be taken into account and is defined as follows:

$$\begin{array}{ll} \hat{\theta}(0) = 0 & P(0) = \psi I \\ r(0) = 0 & R(0) = 0 \end{array} \tag{2.44}$$

with ψ any large positive number.

Remark 2.5. *The algorithm does not need any matrix inversion of dimension $n \times n$, however, it requires the inverse of a 2×2 matrix at step 5. This can be easily tackled during the algorithm implementation by defining*

$$\Gamma = \left[\Lambda(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right] = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{bmatrix}, \tag{2.45}$$

and then, by the well known formula obtain the inverse:

$$\Gamma^{-1} = \frac{1}{\Gamma_{11}\Gamma_{22} - \Gamma_{12}\Gamma_{21}} \begin{bmatrix} \Gamma_{22} & -\Gamma_{12} \\ -\Gamma_{21} & \Gamma_{11} \end{bmatrix}. \tag{2.46}$$

The previously defined algorithm does not take into account the expected values of the quantities involved and may cause numerical problems (as said for algorithm 2.1), with matrices containing either too big or too small numbers. To avoid this issue, we redefine $\rho(t)$ $R(t)$ and $P(t)$ as follows:

$$\begin{aligned} \rho(t) = E\left[\bar{\varphi}_i(t)y(t)\right] &= \frac{1}{t-n-q}\sum_{s=n+q}^{t} \bar{\varphi}_i(s)y(s) \\ &= \frac{1}{t-n-q}\left[\sum_{s=n+q}^{t-1} \bar{\varphi}_i(s)y(s) + \bar{\varphi}_i(t)y(t)\right] \\ &= \frac{1}{t-n-q}\left[\frac{t-n-q-1}{t-n-q-1}\sum_{s=n+q}^{t-1} \bar{\varphi}_i(s)y(s) + \bar{\varphi}_i(t)y(t)\right] \\ &= \frac{t-n-q-1}{t-n-q}\rho(t-1) + \frac{1}{t-n-q}\bar{\varphi}_i(t)y(t) \\ &= \frac{\tau-1}{\tau}\rho(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)y(t), \quad \text{with} \quad \tau = t-n-q. \end{aligned} \tag{2.47}$$

and analogously:

$$R(t) = \frac{\tau-1}{\tau}R(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t) \tag{2.48}$$

While, $P(t)$ is computed with the same course of actions previously shown, so:

$$\begin{aligned}
P^{-1}(t) &= \left[\frac{\tau-1}{\tau}R(t-1)^{\mathrm{T}} + \frac{1}{\tau}\varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\right]\left[\frac{\tau-1}{\tau}R(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)\right] \\
&= \frac{(\tau-1)^2}{\tau^2}P^{-1}(t-1) + \frac{\tau-1}{\tau^2}\varphi_y(t)w^{\mathrm{T}}(t) + \frac{\tau-1}{\tau^2}w(t)\varphi_y^{\mathrm{T}}(t) \\
&\quad + \frac{1}{\tau^2}\varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t) \\
&= \frac{(\tau-1)^2}{\tau^2}P^{-1}(t-1) + \\
&\quad + \frac{1}{\tau^2}\begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix}\begin{pmatrix} 0 & \tau-1 \\ \tau-1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix}\begin{pmatrix} w^{\mathrm{T}}(t) \\ \varphi_y^{\mathrm{T}}(t) \end{pmatrix} \\
&= \frac{(\tau-1)^2}{\tau^2}P^{-1}(t-1) + \frac{1}{\tau^2}\phi(t)\Lambda^{-1}(t)\phi^{\mathrm{T}}(t)
\end{aligned} \tag{2.49}$$

with $\Lambda^{-1}(t)$ changed as:

$$\Lambda^{-1}(t) = \begin{pmatrix} 0 & \tau-1 \\ \tau-1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix} \tag{2.50}$$

and therefore

$$\begin{aligned}
P(t) &= \frac{\tau^2}{(\tau-1)^2}P(t-1) \\
&\quad - \frac{1}{(\tau-1)^2}P(t-1)\phi(t)\left[\Lambda(t) + \frac{1}{(\tau-1)^2}\phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right]^{-1} \\
&\quad \times \frac{\tau^2}{(\tau-1)^2}\phi^{\mathrm{T}}(t)P(t-1)
\end{aligned} \tag{2.51}$$

where given:

$$\Lambda(t) = -\frac{1}{(\tau-1)^2}\begin{pmatrix} -\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) & \tau-1 \\ \tau-1 & 0 \end{pmatrix} = -\frac{1}{(\tau-1)^2}\bar{\Lambda}(t) \tag{2.52}$$

we have

$$\begin{aligned}
P(t) &= \frac{\tau^2}{(\tau-1)^2}P(t-1) \\
&\quad - \frac{\tau^2}{(\tau-1)^2}P(t-1)\phi(t)\left[\phi^{\mathrm{T}}(t)P(t-1)\phi(t) - \bar{\Lambda}(t)\right]^{-1}\phi^{\mathrm{T}}(t)P(t-1).
\end{aligned} \tag{2.53}$$

At this point the remaining part of the update of $\hat{\theta}$ becomes

$$\begin{aligned}
&R^{\mathrm{T}}(t)[\rho(t) - R(t)\hat{\theta}(t-1)] = \\
&= \left[\frac{\tau-1}{\tau}R^{\mathrm{T}}(t-1) + \frac{1}{\tau}\varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\right]\{\frac{\tau-1}{\tau}\rho(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)y(t) \\
&\quad - \left[\frac{\tau-1}{\tau}R(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)\right]\hat{\theta}(t-1)\} \\
&= \frac{1}{\tau^2}\left(\varphi_y(t) \quad (\tau-1)R^{\mathrm{T}}(t-1)\bar{\varphi}_i(t) + \varphi_y(t)\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t)\right) \\
&\quad \times \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\left(\rho(t-1) - (\tau-1)R(t-1)\hat{\theta}(t-1)\right) \\ y(t) - \varphi_y^{\mathrm{T}}(t)\hat{\theta}(t-1) \end{pmatrix} \\
&= \frac{1}{\tau^2}\begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix}\begin{pmatrix} 0 & \tau-1 \\ \tau-1 & \bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) \end{pmatrix} \\
&\quad \times \left\{\begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\rho(t-1) \\ y(t) \end{pmatrix} - \begin{pmatrix} w^{\mathrm{T}}(t) \\ \varphi_y(t) \end{pmatrix}\hat{\theta}(t-1)\right\} \\
&= \frac{1}{\tau^2}\phi(t)\Lambda^{-1}(t)\left[v(t) - \phi^{\mathrm{T}}(t)\hat{\theta}(t-1)\right]
\end{aligned} \tag{2.54}$$

then by using the Push-through lemma 2.3 with $\frac{1}{\tau^2}P(t)\phi(t)$ in eq. (2.34) with the new defined quantities one obtains

$$\frac{1}{\tau^2}P(t)\phi(t) = P(t-1)\phi(t)\left[\phi^{\mathrm{T}}(t)P(t-1)\phi(t) - \bar{\Lambda}(t)\right]^{-1}\Lambda(t) \tag{2.55}$$

which results in the RIV II:

Algorithm 2.6 (ORIV II)**.** *Once a new sample of $y(t)$ is available together with the corresponding instrument $i(t)$, compute*

1. $w(t) = R^{\mathrm{T}}(t-1)\bar{\varphi}_i(t)$
2. $\phi(t) = \begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix}$
3. $\bar{\Lambda}(t) = \begin{pmatrix} -\bar{\varphi}_i^{\mathrm{T}}(t)\bar{\varphi}_i(t) & \tau-1 \\ \tau-1 & 0 \end{pmatrix}$
4. $v(t) = \begin{pmatrix} \bar{\varphi}_i^{\mathrm{T}}(t)\rho(t-1) \\ y(t) \end{pmatrix}$
5. $K(t) = P(t-1)\phi(t)\left[\phi^{\mathrm{T}}(t)P(t-1)\phi(t) - \bar{\Lambda}(t)\right]^{-1}$
6. $\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\left[v(t) - \phi^{\mathrm{T}}(t)\hat{\theta}(t-1)\right]$
7. $R(t) = \frac{\tau-1}{\tau}R(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)\varphi_y^{\mathrm{T}}(t)$
8. $\rho(t) = \frac{\tau-1}{\tau}\rho(t-1) + \frac{1}{\tau}\bar{\varphi}_i(t)y(t)$
9. $P(t) = \frac{\tau^2}{(\tau-1)^2}\left[P(t-1) - K(t)\phi^{\mathrm{T}}(t)P(t-1)\right]$

Where the initial step is defined as follows

$$\begin{aligned} \hat{\theta}(0) &= 0 \quad P(0) = \psi I \\ r(0) &= 0 \quad R(0) = 0 \end{aligned} \tag{2.56}$$

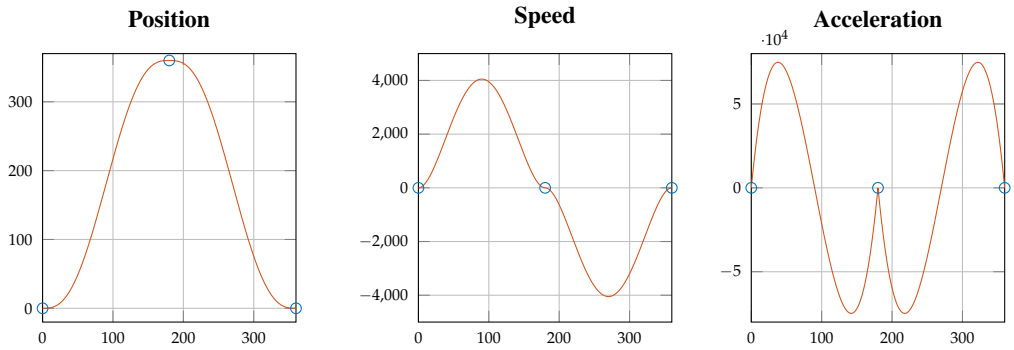

FIGURE 2.2: Example of piece-wise polynomial cam: via-points in blue, master and slave positions are in degrees $[deg]$, while the acceleration is in $[deg^{-1}]$

with ψ any large positive number.

This concludes the MoS tools that are already available in literature (Söderström et al., 1989; Ljung, 1999; Friedlander, 1984), however, the concepts described here are suitable for both practitioners that are interested in the application of this condition monitoring technique, with a focus on the implementation of recursive algorithms for edge computing, and for researchers in the field of PHM and System Identification. Also, ORIV algorithm 2.6 is the foundation of a new contribution in the estimation of AR systems that are sampled through sensors with measurement noise (AR plus noise).

2.4 MoS in trajectory-driven mechanisms

Before exploring the advancement obtained in black-box system identification, we want to introduce a MoS application where a novel way to model the torque signal coming from a trajectory-driven mechanism is developed.

The majority of industrial machines rely on cams to perform complex tasks that require synchronisation among the various mechanisms involved. Cams can be divided into mechanical and electrical. The use of the latter to perform synchronised operations is increasing in the last decades due to their comparable precision and greater flexibility, with respect to mechanical ones. Electric cams allow the coordination of the motion of different mechanisms independently driven by electrical motors. This is possible because servo drives have become able to precisely track given position profiles commanded via Fieldbus by the PLC, allowing the synchronisation of movements via software.

Electric cams are performed by linking together the trajectories of the different motors involved in the synchronised task: a leader, known as master, performs the guiding trajectory while one or more followers, called slaves, move accordingly. The coupling is established geometrically so that any given master trajectory point corresponds to a given slave trajectory point. This coupling is usually programmed by the user on the PLC vendor Integrated Development Environment (IDE). The typical implementations rely on the definition of via-points within the trajectory, which are then connected through mathematical functions that depend on the trajectory constraints. In most cases, polynomial functions, with their smoothness degree dependent on the number of constraints, are used. The constraints, in this case, originate from the required trajectory derivatives at those via-points. For instance, to build a master-slave synchronisation we need the master trajectory in position, $p(\cdot)$ and the relative slave position evolution, defined as $q\left(p(\cdot)\right)$. This definition allows to geometrically connect the

two trajectories, while time enters indirectly with the master position, allowing speed variations without affecting synchronisation. Obviously, also the physical limits of the system affect the trajectory (e.g. the maximum allowed speed and acceleration) and have to be taken into account during the design phase. An example of synchronisation definition procedure is given as follows:

$$\begin{aligned} &q_1(0^\circ) = 0^\circ, \quad q_2(180^\circ) = 360^\circ, \quad q_3(360^\circ) = 0^\circ, \\ &\dot{q}_1(0^\circ) = 0, \quad \dot{q}_2(180^\circ) = 0, \quad \dot{q}_3(360^\circ) = 0, \\ &\ddot{q}_1(0^\circ) = 0^{\circ^{-1}}, \quad \ddot{q}_2(180^\circ) = 0^{\circ^{-1}}, \quad \ddot{q}_3(360^\circ) = 0^{\circ^{-1}}, \end{aligned} \tag{2.57}$$

in which q_1 is connected to q_2 with a polynomial function of order 5 since there is a total of 6 constraints. The same reasoning can be done for the cam piece between q_2 and q_3, with the final result shown in Fig.2.2. If we assume that master speed is constant (as typically happens in real applications), $\dot{p}(t) = const = V_p$, then the x-axis can be directly translated in time by means of $t = p(t)/V_p$. We refer to (Biagiotti et al., 2008) for a complete discussion on how trajectories are generated.

The Torque for Monitoring

Suppose that the controller of the motors we want to synchronise is correctly designed and tuned. The master operates at constant speed followed by the slave with a trajectory defined as in (2.57) driving a linear mechanism. The ideal torque required to perform the task in this case is:

$$\tau(t) = J\alpha(t) = Jq\ddot{(}t), \tag{2.58}$$

where $J > 0$ is the moment of inertia and $\alpha(t) = \ddot{q}(t)$. As a consequence, $\tau(t)$ is a piecewise polynomial trajectory based on M couples of master-slave points and their constraints with the former converted into their time counterparts $t \in [t_1 \ \dots \ t_M]$ following the constant speed assumption. Therefore, ideal torque trajectory segments correspond to the second derivative of the related position profile piece scaled by the inertia factor J. This can be formally described as follows:

$$\tau(t) = \begin{cases} \mathcal{P}^1_{k(1)}(t) & t \in [t_1, t_2] \\ \quad \vdots & \quad \vdots \\ \mathcal{P}^m_{k(m)}(t) & t \in [t_m, t_m + 1] \\ \quad \vdots & \quad \vdots \\ \mathcal{P}^{M-1}_{k(M-1)}(t) & t \in [t_{M-1}, t_M] \end{cases}, \tag{2.59}$$

where $m = 1, \dots, M-1$ is the index of the polynomial piece represented as $\mathcal{P}^m_{k(m)}(t)$ with degree $k(m) = d(m) - 2$, where $d(m)$ is the degree of the respective position polynomial.

The torque measurement from the slave axis is readily available in PLCs implementing electrical cams. Typically, this signal carries the ideal torque profile required by the mechanism, as in (2.58), with parametric uncertainties in J in addition to unmodelled ones (e.g friction, control adjustments, and induced vibrations). As stated in the introduction, our conjecture is that information about the machine state of health is contained in this unknown part. If we are able to take out the ideal cam contribution, the remaining signal can then be used in the Model-of-Signals fashion to perform diagnosis. Our proposition starts from the idea that computing the $(k+1)^{th}$ difference of the ideal torque profile (with k the maximum

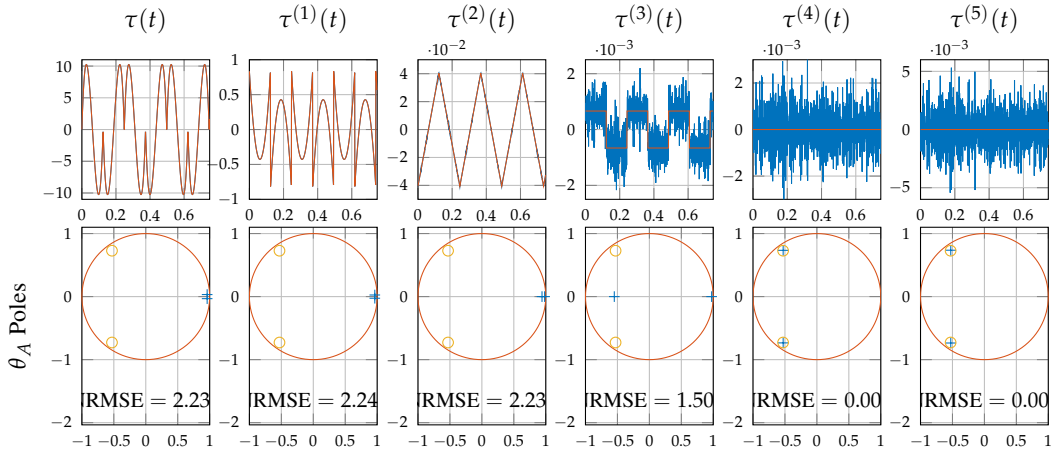

FIGURE 2.3: [Top] Torque signal obtained as in (2.58) of 3 cam periods until the 5^{th} difference. The ideal signal is in orange and the noisy one in blue. [Bottom] Poles of the identified $\hat{\theta}_A$ for the noisy signal (blue), of the applied noise (violet). Notice: the difference signals are processed so to avoid impulses to appear on the cam via-points.

degree of the polynomials in $\tau(t)$), will result in a zero signal. Then, in the real case, we propose to model the torque measurement as an AR process containing information about the system plus the cam nominal torque, which we can get rid of by computing the $(k+1)^{th}$ difference. Therefore we consider the AR process as a representative of the machine health state. The difference computations affect the AR process turning it into an ARMA process. Nevertheless, it is still possible to extract that piece of information with recursive system identification algorithms, as we show in the next section. Notice that it may be advisable, in the difference calculation, to avoid the cam via-points, where the discontinuities generate steps and impulses. This filtering will be used in the simulation, while in the real case it is not required since the physics of the system directly filters this contribution.

Definition and Identification of the Signal Model

Following our reasoning we assume the torque signal $\tau(t)$ to be composed by polynomials with the addition of an AR process:

$$\tau(t) = P^m_{k(m)}(t) + e(t), \tag{2.60}$$

with $P^m_{k(m)}(t)$ compactly denoting (2.59) and

$$e(t) = -a_1 e(t-1) - \cdots - a_n e(t-n) + w(t) = \frac{w(t)}{A(z^{-1})}, \tag{2.61}$$

is an AR process of order n with driving white noise $w(t)$ and $A(z^{-1}) = 1 + a_1 z^{-1} + \cdots + a_n z^{-n}$, where z^{-1} is the backward-shift operator, i.e. $z^{-1}e(t) = e(t-1)$. From now on we drop the symbol $k(m)$ to address the polynomial degree and just use k for the sake of clarity. This modelling permits to take into account the uncertainties derived from the real case. Now, if we apply the difference operator $(1 - z^{-1})$ to (2.58) we obtain

$$(1 - z^{-1})\,\tau(t) = (1 - z^{-1}) \left[P^m_k(t) + \frac{w(t)}{A(z^{-1})} \right] \tag{2.62}$$

which becomes

$$\tau^{(1)}(t) = P_{k-1}^{m}(t) + \frac{(1-z^{-1})w(t)}{A(z^{-1})}. \tag{2.63}$$

Then, by applying again the difference operator k times, we get the $(k+1)^{th}$-order difference of $\tau(t)$:

$$\tau^{(k+1)}(t) = \frac{(1-z^{-1})^{k+1}w(t)}{A(z^{-1})}, \tag{2.64}$$

in which the polynomial contribution disappears and the remaining part is nothing but the following ARMA model:

$$y(t) = \tau^{(k+1)}(t) = \frac{D(z^{-1})}{A(z^{-1})}w(t), \tag{2.65}$$

where the coefficients of $D(z^{-1})$ are known and correspond to the ones of the binomial expansion of $(1-z^{-1})^{k+1}$.

At this point, the problem to be solved consists in estimating the coefficients of $A(z^{-1})$ that, as already said, provide information about the health state of the machine. This identification problem can be solved by employing the Instrumental Variable (IV) method (Söderström et al., 1989; Ljung, 1999). The adopted identification procedure can be summarised as follows. First, the signal $y(t)$ is obtained by performing the $(k+1)^{th}$ difference of $\tau(t)$ in (2.64) and its expression in (2.65) becomes:

$$\begin{aligned} y(t) = &-a_1 y(t-1) - \cdots - a_n y(t-n) + \\ &+ (-1)^0 \binom{k+1}{0} w(t) + (-1)^1 \binom{k+1}{1} w(t-1) + \dots \\ &+ (-1)^{k+1} \binom{k+1}{k+1} w(t-k-1), \end{aligned} \tag{2.66}$$

which in regressor form is:

$$y(t) = \varphi_y^T(t)\theta_A + \varphi_w^T(t)\theta_D \tag{2.67}$$

with

$$\varphi_y(t) = [-y(t-1) \ \dots \ -y(t-n)]^T \tag{2.68}$$

$$\varphi_w(t) = [w(t) \ \dots \ w(t-k-1)]^T \tag{2.69}$$

$$\theta_A = [a_1 \ \dots \ a_n]^T. \tag{2.70}$$

Now, if we choose the following vector of instruments

$$\bar{\varphi}_y(t) = [-y(t-k-2) \ \dots \ -y(t-k-1-n-q)]^T, \tag{2.71}$$

with $q \geq 0$, the IV estimate of θ_A, when N samples of $y(t)$ are available, is given by

$$\hat{\theta}_A = \hat{R}^+ \hat{\rho}, \tag{2.72}$$

where

$$\hat{R} = \sum_{\tau=\tau_0}^{N} \bar{\varphi}_y(\tau)\varphi_y^T(\tau), \quad \hat{\rho} = \sum_{\tau=\tau_0}^{N} \bar{\varphi}_y(\tau)y(\tau), \tag{2.73}$$

$\tau_0 = n + q$, and $\hat{R}^+$ denotes the pseudoinverse of the matrix $\hat{R}$. The choice of the instrument vector (2.71) guarantees the consistency of the estimate for $N \to \infty$ (Söderström et al., 1981; Söderström et al., 1989). Note that if $q > 0$, the solution (2.72) becomes an overdetermined IV method (Söderström et al., 1989), whose recursive solution has been presented in section 2.3. In this study, algorithm 2.5 is the one chosen for the estimation of $\hat{\theta}_A$ with the above depited vector of instruments $\bar{\varphi}_y(t)$. In the following the adopted modelling approach is tested and validated using a synthetic example.

The validation of the proposed Model-of-Signals approach in simulation is done utilizing the cam from the example (2.57) repeated 100 times, with a master working at constant speed $V_p = 1440[^\circ/s]$. We compute the torque $\tau(t)$ required to perform the task with a linear mechanism as in (2.58) with inertia $J = 0.0044[kg/m^2]$. The sampling time adopted in this simulation is $T_s = 0.001[s]$, which is commonly used on PLCs implementing electric cams in their programs. The ideal torque signal, in this case, will be composed of polynomials of the 3^{rd} degree:

$$\tau(t) = P_3^m(t). \tag{2.74}$$

The torque signal simulating a real case, as in (2.60), will also have an AR process of order $n = 2$ in addition, whose parameter vector is the following:

$$\theta_A = [1.058 \quad 0.81]^T, \tag{2.75}$$

with driving noise variance $\sigma_w^2 = 10^{-8}$ and the following couple of complex conjugate poles:

$$\sigma\left(A(z^{-1})\right) = [0.9e^{i0.7\pi}, 0.9e^{-i0.7\pi}]. \tag{2.76}$$

Finally, the simulation has been performed by applying Algorithm 2.5, with the hyperparameter $q = 2$. The Normalised Root Mean Square Error ($NRMSE$) index:

$$NRMSE = \sqrt{\frac{\|\hat{\theta}_A - \theta_A\|}{\|\theta_A\|}}, \tag{2.77}$$

has been used to evaluate the performance in obtaining a $\hat{\theta}_A$ as close as to the value set in (2.75). The results of the simulation are shown in Fig. 2.3. It is possible to observe how in the $(k+1=4)^{th}$-order difference the ideal torque contribution (orange) turns to zero, while the real torque signal (blue) becomes a zero mean ARMA process characterised by the polynomial $A(z^{-1})$ and then by the same coefficients θ_A of the AR process, see (2.65) and (2.67). The application of algorithm 2.5 to $\tau^{(4)}(t)$ results in the identification of the AR process added to the torque with an error of 0.6% ($NRMSE = 0.006$). Therefore, we are able to isolate the model of the AR signal and discard the nominal torque piece of information. In this way, given an electric cam application, the contribution of the uncertainties may be extracted and used for condition monitoring in the Model-of-Signals framework.

2.5 Identification of Noisy AR Models: the Noise-Compensated ORIV

So far, we worked under the assumption that the signals supplied to the model estimator were perfectly sampled by their respective sensing equipment. This assumption, in many cases, turns out to be untrue. Noisy measurements may be unavoidable due to the presence of EMC sources within the machinery or because of the nature or quality of the sensors themselves. For instance, when sampling vibrations onboard a machine we can make use of either piezoelectric or mems accelerometers, knowing that, in general, the former have negligible

measurement noise with respect to the latter. On the other hand, mems are typically cheaper or far cheaper than piezoelectric sensors and tackling the issue of noisy observations via software may result in a cost-effective solution. This is just an example, but it is sufficient to drive our discussion into MoS solutions when signals are sampled under not ideal conditions. In particular, we analyze here the case when those corrupted signals can be modeled under the AR plus noise framework, introducing the Noise-Compensated ORIV (NC-ORIV).

To develop a recursive version of the new algorithm, its batch version is required. As we did for the LS method, we here define a batch solution to the estimation of the AR plus noise problem and then develop its recursive form. So, let us consider the following n-th order AR process

$$x(t) + a_1\, x(t-1) + a_2\, x(t-2) + \cdots + a_n\, x(t-n) = e(t) \tag{2.78}$$

or in transfer function form:

$$x(t) = \frac{1}{A(z^{-1})} e(t) \tag{2.79}$$

where the driving noise $e(t)$ is a zero-mean white process with variance σ_e^2. The AR signal $x(t)$ is corrupted by additive noise so that the available measurement $y(t)$ is given by

$$y(t) = x(t) + w(t) \tag{2.80}$$

where $w(t)$ is a zero-mean white process with variance σ_w^2, uncorrelated with $e(t)$. In this case, we keep the notation with $y(t)$ being the system observation that we measure, when uncorrupted it is easy to revert to the AR framework since $y(t) = x(t)$.

The following assumptions will be considered.

A1. The AR process (2.78) is asymptotically stable, i.e. all roots of the polynomial

$$A(z^{-1}) = 1 + a_1\, z^{-1} + \cdots + a_n\, z^{-n} \tag{2.81}$$

lie inside the unit disc in the z-plane.

A2. The order n of the AR model is assumed as a priori known.

A3. The driving noise signal $e(t)$ is a zero-mean ergodic white process with variance σ_e^2.

A4. The additive noise $w(t)$ is a zero-mean ergodic white process with variance σ_w^2.

A5. $e(t)$ and $w(t)$ are mutually uncorrelated.

Under these assumptions, the noisy AR identification problem can be stated as follows.

Problem 2.2. *Given the set of noisy output data $y(1), y(2), \ldots, y(N)$, estimate the coefficients $a_1, a_2, \ldots, a_n$ and the noise variances σ_e^2, σ_w^2.*

This estimation problem is well known in literature and is related to the Errors-In-Variables (EIV) system identification framework(Guidorzi et al., 2008). When the AR process is corrupted by additive noise classical identification methods like least squares and Yule-Walker equations lead to biased estimates (Kay, 1979; Kay, n.d.; Zheng, 1999). Since this is a very common situation, many approaches have been proposed for identifying AR models in the presence of additive noise. Among them, the high-order Yule-Walker equations (Chan et al., 1982; Kay, n.d.), the prediction error method (Nehorai et al., 1988), the bias-compensated least squares (Zheng, 1999; Zheng, 2000; Zheng, 2005; Zheng, 2006; Mahmoudi et al., 2011; Xia et al., 2015), the errors-in-variables approach (Diversi et al., 2005a; Bobillet et al., 2007; Diversi et al., 2008b). However, none of them has a recursive version that fits MoS and edge-computing as algorithms 2.3 to 2.5 and 2.6, involving in most cases matrix inversions or even more complex operations

In the following, we will denote as $r_y(\tau)$ the autocorrelation at lag τ of the signal $y(t)$:

$$r_y(\tau) = E[y(t)\, y(t-\tau)] = r_y(-\tau), \tag{2.82}$$

where $E[\cdot]$ represents the expectation operator.

Now the following regressors may be defined in order to proceed with the discussion

$$\varphi_x(t) = [-x(t-1) \ \dots \ x(t-n)]^T \tag{2.83}$$
$$\varphi_y(t) = [-y(t-1) \ \dots \ -y(t-n)]^T \tag{2.84}$$
$$\varphi_w(t) = [-w(t-1) \ \dots \ -w(t-n)]^T \tag{2.85}$$

and the parameter vector is still

$$\theta = \left[\, a_1 \, a_2 \, \cdots \, a_n \,\right]^T. \tag{2.86}$$

The, relations (2.78) and (2.80) can be rewritten as

$$x(t) = \varphi_x^T(t)\, \theta + e(t) \tag{2.87}$$
$$\varphi_y(t) = \varphi_x(t) + \varphi_w(t). \tag{2.88}$$

From (2.80), (2.87) and (2.88) it is easy to get

$$y(t) = \varphi_x^T(t)\, \theta + e(t) + w(t), \tag{2.89}$$

then, by using (2.88) it is easy to get

$$y(t) = \varphi_y^T(t)\, \theta - \varphi_w^T(t)\, \theta + e(t) + w(t). \tag{2.90}$$

Finally, we define the extended regressor vector, which is also interpreted as the vector of instruments later on.

$$\begin{aligned} \bar{\varphi}_y(t) = [& -y(t-1) \ \dots \ -y(t-n) \\ & -y(t-n-1) \ \dots \ -y(t-n-q)]^T, \end{aligned} \tag{2.91}$$

where q is chosen such that $q \geq n$.

Problem 2.2 solution is different from the one of problem 2.1, it will involve the use Yule-Walker equations. We start by multiplying both sides of (2.90) by $\bar{\varphi}_y(t)$ and by applying the expectation operator

$$\begin{aligned} & E[\bar{\varphi}_y(t)\, y(t)] = \\ & E[\bar{\varphi}_y(t)\, \varphi_y^T(t)]\, \theta - E[\bar{\varphi}_y(t)\, \varphi_w^T(t)]\, \theta + E[\bar{\varphi}_y(t)(e(t) + w(t))]. \end{aligned} \tag{2.92}$$

By taking into account assumptions A3-A5, the above relation leads to

$$\rho = R\, \theta - \sigma_w^2\, J\, \theta \tag{2.93}$$

where

$$R = E[\bar{\varphi}_y(t)\,\varphi_y^T(t)] = \begin{bmatrix} r_y(0) & r_y(1) & \cdots & r_y(n-1) \\ r_y(1) & r_y(0) & \cdots & r_y(n-2) \\ \vdots & & \ddots & \vdots \\ r_y(n-1) & r_y(n-2) & \cdots & r_y(0) \\ r_y(n) & r_y(n-1) & \cdots & r_y(1) \\ \vdots & & & \vdots \\ r_y(n+q-1) & \cdots & \cdots & r_y(q) \end{bmatrix}, \tag{2.94}$$

$$\rho = E[\bar{\varphi}_y(t)\,y(t)] = -\begin{bmatrix} r_y(1) & r_y(2) & \cdots & r_y(n) & r_y(n+1) & \cdots & r_y(n+q) \end{bmatrix}^T \tag{2.95}$$

and

$$J = \begin{bmatrix} I_{n\times n} \\ 0_{q\times n} \end{bmatrix}. \tag{2.96}$$

Since the autocorrelations $r_y(\tau), \tau = 0,\ldots,n+q$ can be estimated directly from the available noisy measurements, relation (2.93) can be seen as a system of $n+q$ equations in the $n+1$ unknowns $a_1, a_2, \ldots, a_n$ and σ_w^2. In order to solve the system without using iterative search procedures, first consider the following partitions of R and ρ

$$R = \begin{bmatrix} R_L \\ R_H \end{bmatrix}, \qquad \rho = \begin{bmatrix} \rho_L \\ \rho_H \end{bmatrix}, \tag{2.97}$$

where R_L and R_H have dimensions $n \times n$ and $q \times n$ while ρ_L and ρ_H are a $n \times 1$ and a $q \times 1$ column vectors, respectively. The set of equations (2.93) can thus be split as follows

$$\rho_L = R_L\,\theta - \sigma_w^2\,\theta \tag{2.98}$$

$$\rho_H = R_H\,\theta. \tag{2.99}$$

As $q \geq n$, Equation (2.99) can be solved with respect to θ as follows

$$\theta_H = R_H^+\,\rho_H \tag{2.100}$$

where R_H^+ denotes the pseudoinverse of R_H. By replacing θ with θ_H in (2.98) it is possible to express the additive noise variance as follows

$$\sigma_w^2 = \frac{\theta_H^T\left(R_L\,\theta_H - \rho_L\right)}{\theta_H^T\,\theta_H}. \tag{2.101}$$

The above equation shows that σ_w^2 can be estimated directly from the available data without using iterative search procedures. Once that an estimation of σ_w^2 has been computed, the whole set of equations (2.93) may be used to obtain the final estimate of the parameter vector θ:

$$\theta = R_c^+\,\rho, \tag{2.102}$$

where R_c^+ is the pseudoinverse of the noise-compensated matrix

$$R_c = R - \sigma_w^2 J. \tag{2.103}$$

This approach is thus based on a set of noise-compensated Yule-Walker (NCYW) equations because the effect of the additive noise is "compensated" in (2.103) by using the estimate (2.101). Therefore, both the low-order and the high-order Yule-Walker equations (2.98) and (2.99) are exploited to determine an estimate of θ.

Remark 2.6. *As shown in (Davila, 2001), the condition $q \geq n$ is both necessary and sufficient to guarantee the existence of a unique solution to the NCYW equations (2.93). Therefore, if $q \geq n$, only the true parameter vector θ and the true additive noise variance σ_w^2 satisfy the set of equations (2.93).*

The AR coefficients and the additive noise variance can be estimated employing the following steps.

Algorithm 2.7 (AR + Noise estimator)**.** *When a new set $y(1), y(2), \ldots, y(N)$ of noisy data is available*

1. *Compute the sample estimates $\hat{R}_L$, $\hat{\rho}_L$, $\hat{R}_H$ and $\hat{\rho}_H$. Construct the matrices $\hat{R}$ and $\hat{\rho}$ as in (2.97).*
2. *Determine a first estimate $\hat{\theta}_H$ of θ by using the so-called high-order Yule-Walker equations (2.99):*
$$\hat{\theta}_H = \hat{R}_H^+ \hat{\rho}_H. \tag{2.104}$$
3. *Compute an estimate of the additive noise variance as follows*
$$\hat{\sigma}_w^2 = \frac{\hat{\theta}_H^T \left(\hat{R}_L \hat{\theta}_H - \hat{\rho}_L\right)}{\hat{\theta}_H^T \hat{\theta}_H}. \tag{2.105}$$
4. *Determine the final estimate $\hat{\theta}$ of the AR coefficients by using the whole set of equations (2.93) and the estimated noise variance $\hat{\sigma}_w^2$:*
$$\hat{\theta} = \hat{R}_c^+ \hat{\rho}, \tag{2.106}$$
where $\hat{R}_c = \hat{R} - \hat{\sigma}_w^2 J$.

Remark 2.7. *Because of assumptions A1, A3 and A4, the sample estimates $\hat{R}_L$, $\hat{\rho}_L$, $\hat{R}_H$ and $\hat{\rho}_H$ are consistent:*

$$\hat{R}_L \lim_{N\to\infty} R_L, \quad \hat{\rho}_L \lim_{N\to\infty} \rho_L \quad w.p.1$$
$$\hat{R}_H \lim_{N\to\infty} R_H, \quad \hat{\rho}_H \lim_{N\to\infty} \rho_H \quad w.p.1$$

As a consequence, since the condition $q \geq n$ guarantees a unique solution to the NCYW equations (see Remark 1), Algorithm 1 is consistent, i.e

$$\hat{\sigma}_w^2 \lim_{N\to\infty} \sigma_w^2, \quad \hat{\theta} \lim_{N\to\infty} \theta \quad w.p.1$$

Remark 2.8. *To increase the accuracy of the obtained solutions it is possible iterate a few times steps 3 and 4 using the result of eq. (2.106) into eq. (2.105).*

Remark 2.9. *Once that θ and σ_w^2 have been estimated, an estimate of the driving noise variance σ_e^2 can be obtained from (2.90). In fact, multiplying both sides of (2.90) by $y(t)$ and taking the expectation it is easy to get*

$$r_y(0) = -\rho_L^T \theta + \sigma_e^2 + \sigma_w^2. \tag{2.107}$$

The estimate of σ_e^2 can thus be easily computed at the end of Algorithm 1 as follows

$$\hat{\sigma}_e^2 = \hat{r}_y(0) + \hat{\rho}_L^T \hat{\theta} - \hat{\sigma}_w^2. \tag{2.108}$$

The proposed identification algorithm is computationally simpler than other existing algorithms. Indeed, it does not involve any optimization problem to be solved unlike, for example, the approach (Diversi et al., 2008b). Compared to other methods that do not involve finding the optimum (Davila, 1998), algorithm 2.7 is again more efficient since no computation of matrix eigenvalues and eigenvectors are required. Despite its simplicity, algorithm 2.7 leads to very good estimates, as shown in the following.

The performance of the proposed identification algorithm has been tested by means of Monte Carlo simulation and compared with those of other approaches that relies on noise-compensated Yule-Walker equations namely the subspace approach introduced in (Davila, 1998) and the errors-in-variables approach described in (Diversi et al., 2008b). The following AR processes have been considered and are taken from (Kay, 1980) and (Diversi et al., 2008b):

$$x(t) - 1.352\,x(t-1) + 1.338\,x(t-2) - 0.662\,x(t-3) + 0.240\,x(t-4) = e(t), \tag{2.109}$$

$$x(t) - 2.7607\,x(t-1) + 3.810\,x(t-2) - 2.6535\,x(t-3) + 0.9238\,x(t-4) = e(t), \tag{2.110}$$

$$x(t) - 2.1690\,x(t-1) + 2.8227\,x(t-2) - 2.0408\,x(t-3) + 0.8853\,x(t-4) = e(t), \tag{2.111}$$

$$x(t) - 1.6771\,x(t-1) + 1.6875\,x(t-2) - 0.9433\,x(t-3) + 0.3164\,x(t-4) = e(t). \tag{2.112}$$

The model in eqs. (2.109) and (2.112) represent broadband processes with a smooth spectrum whereas the models in eqs. (2.110) and (2.111) represent narrowband processes with two sharp peaks that are quite close. For all the above models, the driving process is white noise with unit variance. Monte Carlo simulations of $M = 500$ runs have been carried out by considering additive noise sequences with variance σ_w^2 corresponding to a defined Signal-to-Noise Ratio SNR $= 10$ dB and SNR $= 5$ dB, where

$$\text{SNR} = 20\,\log_{10}\sqrt{\frac{E[x^2(t)]}{\sigma_w^2}}\,(\text{dB}). \tag{2.113}$$

The number of available noisy output samples is $N = 4000$. For every identification approach (Davila, 1998, Diversi et al., 2008b and algorithm 2.7) three different values of the parameter q (i.e. the number of high-order YW equations in (2.99)) have been considered: $q = 5$, $q = 20$ and $q = 40$. The obtained results are summarized in table 2.1 that reports, for each experimental case, the metric used to assess the accuracy of the experiment estimates is the Normalized Root Mean Square Error defined as

$$\text{NRMSE} = \sqrt{\frac{1}{M}\sum_{i=1}^{M}\frac{\|\hat{\theta}^i - \theta\|^2}{\|\theta\|}}, \tag{2.114}$$

TABLE 2.1: Algorithm 2.7 comparison with (Davila, 1998) and (Diversi et al., 2008b) over 500 Monte Carlo runs.

	Model	(2.109)		(2.110)		(2.111)		(2.112)	
	SNR	10 dB	5 dB	10 dB	5 dB	10 dB	5 dB	10 dB	5 dB
(Davila, 1998)	$q = 5$	47.74%	76.70%	95.27%	98.44%	11.08%	37.88%	55.92%	82.78%
	$q = 20$	45.62%	75.82%	13.99%	72.01%	4.65%	16.99%	52.75%	81.85%
	$q = 40$	42.74%	74.58%	7.23%	42.39%	3.56%	12.84%	49.71%	80.78%
(Diversi et al., 2008b)	$q = 5$	6.00%	10.95%	24.85%	69.84%	1.07%	2.69%	5.19%	11.67%
	$q = 20$	5.90%	11.20%	25.00%	57.70%	0.92%	2.59%	5.42%	11.38%
	$q = 40$	6.43%	12.32%	23.00%	60.51%	1.01%	3.01%	5.01%	14.06%
algorithm 2.7 No iterations	$q = 5$	34.58%	51.34%	39.99%	74.25%	0.86%	2.16%	17.07%	41.49%
	$q = 20$	13.49%	30.82%	2.45%	18.68%	0.78%	2.01%	8.39%	29.22%
	$q = 40$	14.77%	26.82%	3.49%	24.55%	0.81%	2.80%	8.19%	28.94%
algorithm 2.7 10 iterations	$q = 5$	29.44%	43.67%	12.40%	59.52%	0.81%	1.97%	6.67%	30.84%
	$q = 20$	6.83%	23.79%	2.44%	18.67%	0.72%	2.01%	5.30%	17.26%
	$q = 40$	8.13%	18.85%	3.49%	24.55%	0.81%	2.80%	5.45%	19.58%

where M is the number of runs of each Monte Carlo simulation (500 in our case), θ is the true parameter vector (2.86) and $\hat{\theta}^i$ denotes the estimate of θ obtained in the i–th run,$i = 1,\dots,M$. In addition, algorithm 2.7 is tested performing some few more iterations (10 in this case) as pointed out in remark 2.8.

It is possible to observe that algorithm 2.7 accuracy comes in between the subspace based algorithm (Davila, 1998) and the frisch-scheme-based one (Diversi et al., 2008b). In particular, it is very close to the performance of that second one, even overcoming it in some cases. Form the result it is possible to observe how the right choice of the number of HOYW equations affects the accuracy of the estimate, suggesting that $q = 20$ and $q = 40$ and values in between provide a better estimation of θ. In the end, the addition of 10 iterations of the steps 3 and 4, as for remark 2.8, provide overall better estimation accuracy of the models.

At this point, since we have shown that the algorithm is capable in the estimation task using quite tricky synthetic models from the literature, it is time to introduce the recursive version of algorithm 2.7, starting from equation (2.93) reformulated as

$$R\,\theta = \rho + \sigma_w^2\, J\,\theta \tag{2.115}$$

In which the parameter vector θ may be obtained as:

$$\theta = R^+\,\rho - \sigma_w^2\, R^+\, J\,\theta \tag{2.116}$$

where R^+ is the pseudo-inverse of R. At this point, equation (2.116) is reformulated so to iteratively compute the estimate of $\hat{\theta}$ at time t given its value at time $t-1$:

$$\hat{\theta}(t) = \hat{R}^+(t)\,\hat{\rho}(t) - \hat{\sigma}_w^2(t-1)\,\hat{R}^+(t)\,J\,\hat{\theta}(t-1). \tag{2.117}$$

In this fashion, at each iteration we need to update $\hat{R}^+$ and $\hat{\rho}$ and rely on the last computed value of $\hat{\sigma}_w^2$ by using equation (2.101). The derivation of the recursive update of the quantities $\hat{R}^+$ and $\hat{\rho}$ is obtained in the same way as we depicted for algorithm 2.6. It is the ORIV version that takes into account the experimental autocorrelations of the quantities involved, since in this case they are required to compute the estimations of the noise variances. Following this

reasoning, we start by rewriting equation (2.117) expanding $\hat{R}^+$ into its components:

$$\hat{\theta}(t) = \hat{P}(t)\hat{R}^T(t)\,\hat{\rho}(t) - \hat{\sigma}_w^2(t-1)\,\hat{P}(t)\hat{R}^T(t)\,J\,\hat{\theta}(t-1) \tag{2.118}$$

where $\hat{P}^{-1} = \hat{R}^T\hat{R}$. In the end, eq. (2.118) in combination with algorithm 2.6 leads to the formulation of the Noise-Compensated Overdetermined Instrimental Variable for noise cossrupted AR systems:

Algorithm 2.8 (NC-ORIV)**.**

1. $w(t) = R^{\mathrm{T}}(t-1)\bar{\varphi}_y(t)$
2. $\phi(t) = \begin{pmatrix} w(t) & \varphi_y(t) \end{pmatrix}$
3. $\bar{\Lambda}(t) = \begin{pmatrix} -\bar{\varphi}_y^{\mathrm{T}}(t)\bar{\varphi}_y(t) & \tau - 1 \\ \tau - 1 & 0 \end{pmatrix}$
4. $K(t) = P(t-1)\phi(t)\left[\bar{\Lambda}(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right]^{-1}$
5. $R(t) = \frac{\tau-1}{\tau}R(t-1) + \frac{1}{\tau}\bar{\varphi}_y(t)\varphi_y^{\mathrm{T}}(t)$
6. $\rho(t) = \frac{\tau-1}{\tau}\rho(t-1) + \frac{1}{\tau}\bar{\varphi}_y(t)y(t)$
7. $P(t) = \frac{\tau^2}{(\tau-1)^2}\left[P(t-1) - K(t)\phi^{\mathrm{T}}(t)P(t-1)\right]$
8. $\hat{\theta}(t) = P(t)R^T(t)\rho(t) + \sigma_w^2(t-1)P(t)R^T(t)J\hat{\theta}(t-1)$
9. $\hat{\sigma}_w^2(t) = \frac{\hat{\theta}(t)^T\left(R_L(t)\hat{\theta}(t) - \rho_L(t)\right)}{\hat{\theta}(t)^T\hat{\theta}(t)}$
10. $\hat{\sigma}_e^2 = R_{11}(t) + \rho_L^T(t)\,\hat{\theta}(t) - \hat{\sigma}_w^2(t)$

Where the initial step is may be defined in the following way

$$\begin{array}{ll} \hat{\theta}(0) = \alpha\mathbf{1} & P(0) = \psi I \\ r(0) = \beta\mathbf{1} & R(0) = \gamma J \end{array} \tag{2.119}$$

with α, β and γ any small positive number and ψ a large positive one. While the initial value for σ_w^2 can be chosen as for algorithm 2.7.

Its implementation is indeed a bit more complex than the standard RLS and has few more computations than the standard ORIV. Here are some remark to be taken into account when implementing the algorithm:

Remark 2.10. *The algorithm does not need any matrix inversion of dimension $n \times n$, however, it requires the inverse of a 2×2 matrix at point 4. This can be easily tackled during the algorithm implementation by defining*

$$\Gamma = \left[\bar{\Lambda}(t) + \phi^{\mathrm{T}}(t)P(t-1)\phi(t)\right] = \begin{bmatrix} \Gamma_{11} & \Gamma_{12} \\ \Gamma_{21} & \Gamma_{22} \end{bmatrix}, \tag{2.120}$$

and then, by the well known formula obtain the inverse:

$$\Gamma^{-1} = \frac{1}{\Gamma_{11}\Gamma_{22} - \Gamma_{12}\Gamma_{21}} \begin{bmatrix} \Gamma_{22} & -\Gamma_{12} \\ -\Gamma_{21} & \Gamma_{11} \end{bmatrix}. \tag{2.121}$$

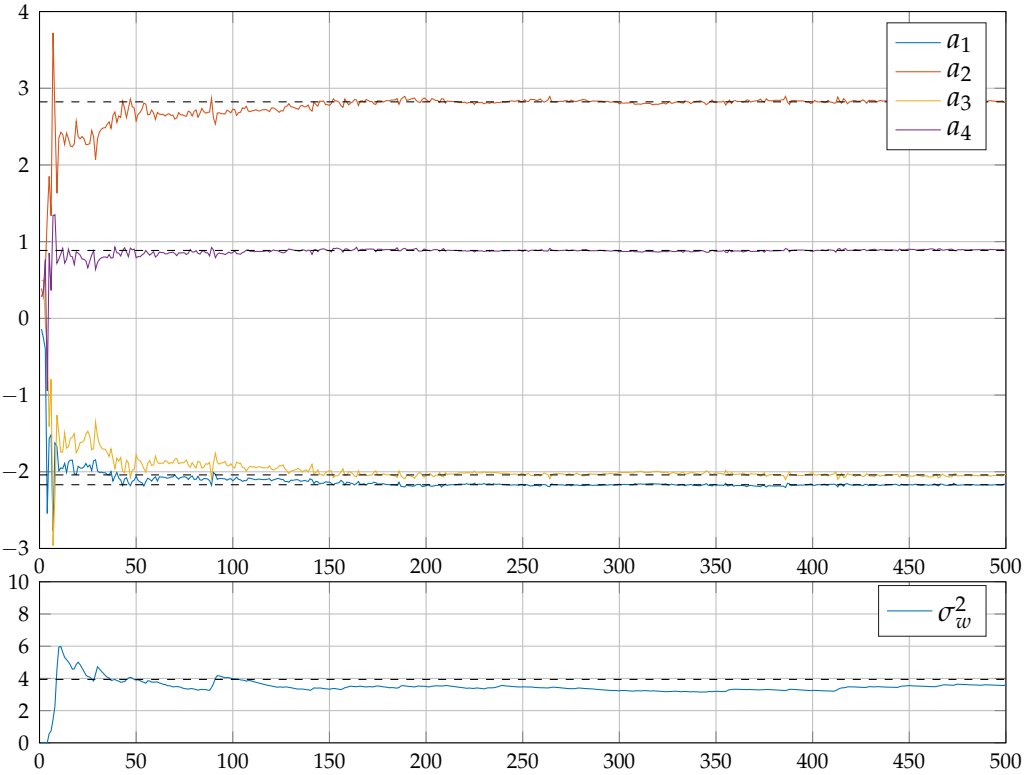

FIGURE 2.4: Estimation of Model (2.111) under SNR of 10dB with NC-ORIV, $\hat{\theta}$ and $\hat{\sigma}_w^2$ evolution during time.

Remark 2.11. *To ensure numerical consistency, the algorithm implementation should monitor the value of σ_w^2 in order to keep it at least ≥ 0. Numerical errors introduced by the deployment of the algorithm may occur, in particular when σ_w^2 is close to 0.*

Finally, fig. 2.4 shows algorithm 2.8 in action when identifying model (2.111) under an SNR of 10dB. It is possible to observe how the estimation converges despite having only 500 samples to do so.

Forgetting Factor

As pointed out for algorithm 2.3 and algorithm 2.6 is not able to track variations in time of the model underlying parameters. This happens because as t increases the elements contained in $R(t), P(t)$, and $\rho(t)$ are updated with a smaller and smaller value. On the other hand, the introduction of a forgetting factor λ may help in the estimation of a varying θ, but at the cost of losing the estimate of $\sigma_w^2(t)$. The way $R(t), P(t)$, and $\rho(t)$ are obtained is needed by the algorithm to get the output noise variance. A simple, yet effective solution to introduce a similar "memory loss" as the forgetting factor does is to fix τ to an arbitrary number of samples. This keeps $R(t), P(t)$, and $\rho(t)$ on giving an estimate of the various experimental correlations involved in the solution of the identification problem and in giving also a solid estimation of $\sigma_w^2(t)$. Here, we briefly show in fig. 2.5 what it means to fix τ to 4000 and let the algorithm follow the transition from the model in eq. (2.112) to eq. (2.111), including the evolution of $\sigma_w^2(t)$ from 0.63 to 3.08 over $15 \cdot 10^5$ samples. It is possible to observe that the algorithm can track the parameter variation as well as the variance.

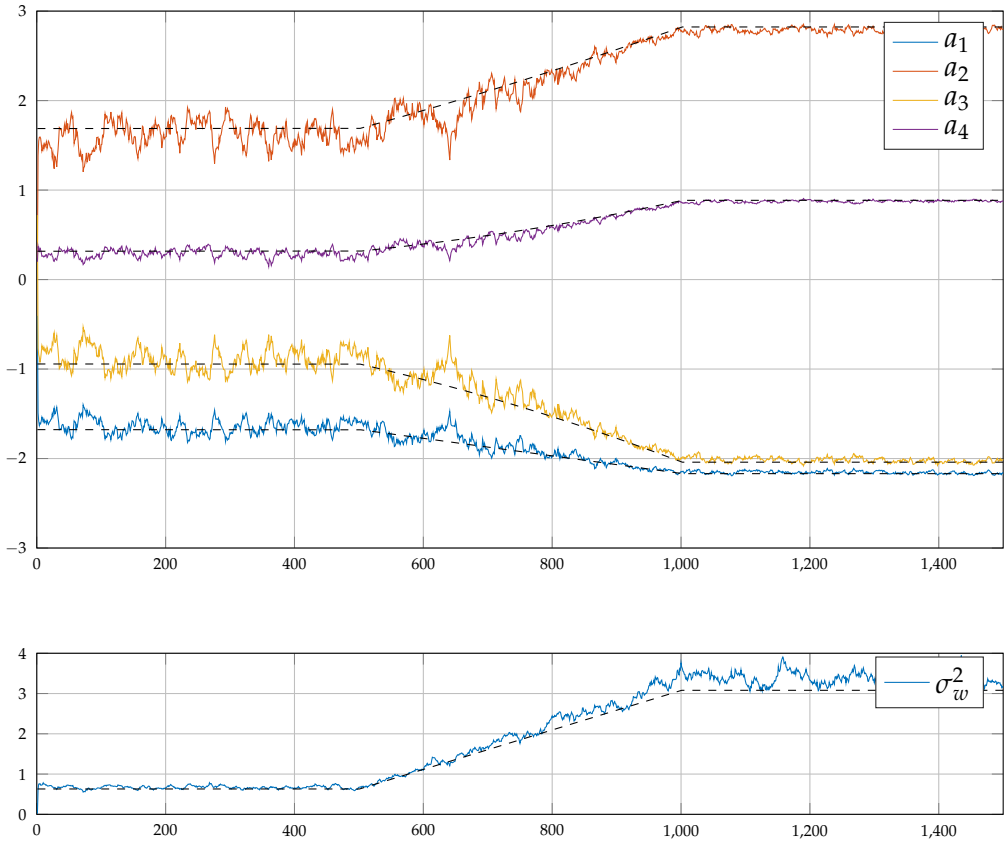

FIGURE 2.5: Estimation of eq. (2.112) to eq. (2.111) under SNR of 10dB with NC-ORIV, $\hat{\theta}$ and $\hat{\sigma}_w^2$ evolution during time.

This concludes the discussion on the estimation through recursive algorithms for the edge-computing implementation of MoS. Despite not having an actual case study of algorithm 2.8, this is the biggest theoretical advancement obtained during the studies on MoS, and it has a lot of potentialities.

2.6 Noisy Input-Output FIR estimation

As a collateral study to the analysis of system identification for MoS, and in particular in the case of corrupted measurements from the system, an algorithm for the identification of Noisy Finite Impulse Response (FIR plus Noise) systems has been developed.

The identification of finite impulse response (FIR) models plays an important role in many signal processing and control applications (Haykin, 1991; Goodwin et al., 1984; Schafer et al., 1989; Kalouptsidis, 1997; Davila, 1994; Xu et al., 1995; Diversi et al., 2005b; Cerone et al., 2013; Aljanaideh et al., 2017; Aljanaideh et al., 2018). When both the input and output of the FIR model are corrupted by noise, classical identification algorithms like least squares and prediction error methods lead to biased estimates (Söderström, 2018).

Noisy input-output models can be consistently identified using the instrumental variable (IV) method (Söderström, 2018). In this case, the vector of instruments is based on delayed input and output samples leading to a set of high-order Yule-Walker (HOYW) equations. Despite its computational efficiency, this approach may lead to a low estimation accuracy because of the poor estimates of the high-lag auto-correlations involved in the HOYW equations (Söderström, 2018). The total least squares (TLS) scheme can also be used but the ratio of the input noise variance to the output noise variance is assumed as *a priori* known (Davila, 1994; Feng et al., 1998; Feng et al., 2006).

An effective way to counteract the noise-induced bias in the Least Squares estimate consists of using the bias compensation principle (Stoica et al., 1982; Söderström, 2018). Starting from this principle, various bias-compensated least squares (BCLS) algorithms have been proposed to identify noisy FIR models (Zheng, 2003; Feng et al., 2007; Diversi et al., 2008a; Diversi, 2008; Diversi, 2009; Kang et al., 2013; Arablouei et al., 2014; Jung et al., 2017). Many BCLS methods assumes that both the input and the output noise are white processes. In (Bertrand et al., 2011; Arablouei et al., 2014), the input noise is a colored process and is correlated with the white output noise. However, the input noise covariance matrix and the cross-covariance between the input and output noise are assumed *a priori* known or already estimated.

In this case, we deal with the identification of FIR models corrupted by white input noise and colored output noise. The estimation of the FIR coefficients is performed relying on the properties of the dynamic Frisch scheme (Guidorzi et al., 2008). The idea underlying the Frisch scheme consists of finding the estimation of the input-output noise variances within a locus of solutions compatible with the covariance matrix of the noisy data. The search for a single solution inside the locus requires the definition of a suitable selection criterion. For single-input single-output infinite impulse response (IIR) models, the Frisch locus is described by a curve in the first quadrant of $\mathbb{R}^2$. Nevertheless, for the FIR case, such locus can be described by a segment of $\mathbb{R}^+$, as shown in (Diversi et al., 2008a).

Starting from the EIV framework, it makes use of the Frish scheme and HOYW equations to seek the relative identification problem solution. This is still in the early stages concerning its application to MoS, but it may be a good basis for the design of a useful recursive form. In the following, we discuss algorithm development.

Let us consider the linear, time-invariant FIR system. Its input u_0 and output y_0 are linked by the difference equation

$$y_0(t) = H(z^{-1})u_0(t), \tag{2.122}$$

where $H(z^{-1})$ is the following polynomial in the backward shift operator z^{-1}:

$$H(z^{-1}) = h_0 + h_1 z^{-1} + \ \dots \ + h_{n-1} z^{1-n}. \tag{2.123}$$

Let us assume that the available measurements of the true input and output are corrupted by the presence of additive noise $\tilde{u}(t)$ and $\tilde{y}(t)$, namely

$$u(t) = u_0(t) + \tilde{u}(t) \tag{2.124}$$
$$y(t) = y_0(t) + \tilde{y}(t). \tag{2.125}$$

The following assumptions are introduced.

A1. The FIR length n is known.

A2. The noise-free input u_0 is either a zero–mean ergodic process or a quasi-stationary bounded deterministic signal, i.e. such that the limit

$$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} u_0(t)\, u_0(t-\tau) \tag{2.126}$$

exists $\forall \tau$ (Ljung, 1999). Moreover, $u_0(t)$ is persistently exciting of sufficiently high order.

A3. The input noise $\tilde{u}(t)$ is a zero-mean ergodic white process with unknown variance $\sigma^{2*}_{\tilde{u}}$.

A4. The output noise $\tilde{y}(t)$ is an arbitrarily autocorrelated zero–mean ergodic process with unknown autocorrelation function. Its variance is denoted by $\sigma^{2*}_{\tilde{y}}$.

A5. The additive noises $\tilde{u}(t)$, $\tilde{y}(t)$ are mutually uncorrelated and uncorrelated with the noise-free input $u_0(t)$.

By defining the vectors

$$\varphi_0(t) = [-y_0(t)\ u_0(t)\ u_0(t-1)\ \dots\ u_0(t-n+1)]^T, \tag{2.127}$$
$$\begin{aligned}\varphi(t) &= [-y(t)\ u(t)\ u(t-1)\ \dots\ u(t-n+1)]^T \\ &= \left[-y(t)\ \varphi_u^T(t)\right]^T,\end{aligned} \tag{2.128}$$
$$\tilde{\varphi}(t) = [-\tilde{y}(t)\ \tilde{u}(t)\ \tilde{u}(t-1)\ \dots\ \tilde{u}(t-n+1)]^T, \tag{2.129}$$

and the vector of FIR coefficients

$$\bar{\theta}^* = [1\ h_0\ h_1\ \dots\ h_{n-1}]^T = \left[1\ \theta^{*T}\right]^T \tag{2.130}$$

it is possible to rewrite the model (2.122)–(2.125) in the following way:

$$\varphi_0^T(t)\bar{\theta}^* = 0, \tag{2.131}$$
$$\varphi(t) = \varphi_0(t) + \tilde{\varphi}(t). \tag{2.132}$$

Finally, the identification problem turns out to be the following:

Problem 2.3. *Given a set of N measurements of the corrupted input $u(t)$ and output $y(t)$, estimate the FIR coefficients $h_0, h_1, \dots, h_{n-1}$, i.e. the coefficient vector θ^*.*

The noisy input-output FIR model (2.122)–(2.125), belongs to the family of errors-in-variables models so that it can be estimated by exploiting the properties of the Frisch scheme,

as done in (Diversi et al., 2008a). The main idea behind the Frisch scheme consists in finding the solution of the identification problem within a locus of solutions that are compatible with the covariance matrix of the noisy data (Guidorzi et al., 2008). To this end, let us consider the covariance matrix

$$R = E\left[\varphi(t)\varphi^T(t)\right], \tag{2.133}$$

where $E[\cdot]$ denotes the expectation operator, and, similarly, the covariance matrices $R_0 = E\left[\varphi_0(t)\varphi_0^T(t)\right]$, $\tilde{R}^* = E\left[\tilde{\varphi}(t)\tilde{\varphi}^T(t)\right]$ of the noise-free data and noise respectively. From (2.131), (2.132) and Assumptions A2–A5 we get

$$R_0\bar{\theta}^* = 0, \tag{2.134}$$

$$R = R_0 + \tilde{R}^*. \tag{2.135}$$

From (2.134) and (2.135) we finally obtain:

$$\left(R - \tilde{R}^*\right)\bar{\theta}^* = 0, \tag{2.136}$$

where the noise covariance matrix is

$$\tilde{R}^* = \begin{bmatrix} \sigma_{\tilde{y}}^{2*} & 0 \\ 0 & \sigma_{\tilde{u}}^{2*} I_n \end{bmatrix}. \tag{2.137}$$

and I_n denotes the $n \times n$ identity matrix. Consider now the set of couples $\left(\sigma_{\tilde{u}}^2, \sigma_{\tilde{y}}^2\right)$ belonging to the first quadrant of $\mathbb{R}^2$ and satisfying

$$\left(R - \tilde{R}\right) \geq 0 \text{ and } \det\left(R - \tilde{R}\right) = 0, \tag{2.138}$$

where

$$\tilde{R} = \begin{bmatrix} \sigma_{\tilde{y}}^2 & 0 \\ 0 & \sigma_{\tilde{u}}^2 I_n \end{bmatrix}. \tag{2.139}$$

Because of (2.138), any couple of the set can be associated with a parameter vector $\bar{\theta}\left(\sigma_{\tilde{u}}^2, \sigma_{\tilde{y}}^2\right)$ satisfying

$$\left(R - \tilde{R}\right)\bar{\theta}(\sigma_{\tilde{u}}^2, \sigma_{\tilde{y}}^2) = 0. \tag{2.140}$$

Note that $\bar{\theta}\left(\sigma_{\tilde{u}}^2, \sigma_{\tilde{y}}^2\right)$ can be computed from any basis of the null space of $R - \tilde{R}$ by normalizing the first element to 1, see (2.130).

At this point, it is possible to further develop the set of equations (2.140) in order to restrict the locus of solutions, by exploiting the peculiarities of FIR models. First, the matrix R is partitioned into the following blocks

$$R = \begin{bmatrix} \sigma_y^2 & r_{yu}^T \\ r_{yu} & R_u \end{bmatrix} \tag{2.141}$$

where

$$\sigma_y^2 = E\left[y^2(t)\right] \tag{2.142}$$

$$r_{yu} = -E\left[\varphi_u(t)y(t)\right] \tag{2.143}$$

$$R_u = E\left[\varphi_u(t)\varphi_u^T(t)\right] \tag{2.144}$$

Then, (2.140) may be rewritten as

$$\begin{bmatrix} \sigma_y^2 - \sigma_{\tilde{y}}^2 & r_{yu}^T \\ r_{yu} & R_u - \sigma_{\tilde{u}}^2 I_n \end{bmatrix} \begin{bmatrix} 1 \\ \theta \end{bmatrix} = 0 \tag{2.145}$$

from which we obtain the following two sets of equations

$$\sigma_y^2 - \sigma_{\tilde{y}}^2 + r_{yu}^T \theta = 0, \tag{2.146}$$

$$r_{yu} + \left(R_u - \sigma_{\tilde{u}}^2 I_n\right) \theta = 0. \tag{2.147}$$

Finally, by solving (2.146) and (2.147) accounting for conditions (2.138), we state the following theorem.

Theorem 2.1 (Diversi et al., 2008a)**.** *The set of all diagonal matrices $\tilde{R}$ satisfying (2.138) is defined by the couples $\left(\sigma_{\tilde{u}}^2, \sigma_{\tilde{y}}^2\right)$ belonging to the first quadrant of $\mathbb{R}^2$ such that*

$$\sigma_{\tilde{u}}^2 \in \left[0, \sigma_{\tilde{u}_{max}}^2\right], \tag{2.148}$$

$$\sigma_{\tilde{y}}^2 = \sigma_y^2 + r_{yu}^T \theta, \tag{2.149}$$

with

$$\theta = -\left(R_u - \sigma_{\tilde{u}}^2 I_n\right)^{-1} r_{yu}, \tag{2.150}$$

$$\sigma_{\tilde{u}_{max}}^2 = \lambda_{min}\left(R_u - \frac{r_{yu} r_{yu}^T}{\sigma_y^2}\right). \tag{2.151}$$

Corollary 2.1 (Diversi et al., 2008a)**.** *The locus of solutions described by Theorem 2.1 contains the couple $(\sigma_{\tilde{u}}^{2*}, \sigma_{\tilde{y}}^{2*})$ that can be associated with the true coefficients of the FIR model θ^* by means of (2.150).*

It is important to note that this formulation of the solution set of Problem 2.3 leads to a locus of solutions which is a function of $\sigma_{\tilde{u}}^2$ only, with $\sigma_{\tilde{y}}^2$ and θ depending on it, namely

$$\theta(\sigma_{\tilde{u}}^2) = -\left(R_u - \sigma_{\tilde{u}}^2 I_n\right)^{-1} r_{yu}, \tag{2.152}$$

$$\sigma_{\tilde{y}}^2(\sigma_{\tilde{u}}^2) = \sigma_y^2 + r_{yu}^T \theta(\sigma_{\tilde{u}}^2). \tag{2.153}$$

with $\sigma_{\tilde{y}}^2(\sigma_{\tilde{u}}^{2*}) = \sigma_{\tilde{y}}^{2*}$ and $\theta(\sigma_{\tilde{u}}^{2*}) = \theta^*$ corresponding to the true solution. In the asymptotic case, the identification problem may thus be solved by looking for $\sigma_{\tilde{u}}^{2*}$ within the interval $\left[0, \sigma_{\tilde{u}_{max}}^2\right]$. To this end it is necessary to introduce a suitable selection criterion. The criterion proposed in (Diversi et al., 2008a), which exploits the statistical properties of the residuals of the EIV FIR model, can be applied only when both the input and output noise are white processes. The next section describes a criterion based on the properties of the high-order Yule-Walker equations that allows dealing with colored output noise.

The search for the true input noise variance $\sigma_{\tilde{u}}^{2*}$ within the Frisch locus can be performed by relying on the high-order Yule-Walker (HOYW) equations, as shown in the following.

Firstly, we define the $q \times 1$ vector of delayed input samples

$$\varphi^q(t) = \left[u(t-n)\ u(t-n-1)\ \ldots\ u(t-n-q+1)\right]^T, \tag{2.154}$$

with $q \geq 1$. Because of (2.124), $\varphi^q(t)$ can be decomposed as

$$\varphi^q(t) = \varphi_0^q(t) + \tilde{\varphi}^q(t), \tag{2.155}$$

where the elements of $\varphi_0^q(t)$ and $\tilde{\varphi}^q(t)$ are vectors of delayed samples of $u_0(t)$ and $\tilde{u}(t)$ respectively. Define the following $q \times (n+1)$ matrix

$$R^q = E\left[\varphi^q(t)\varphi^T(t)\right]. \tag{2.156}$$

From (2.155), (2.132) and Assumptions A3, A5 we get

$$\begin{aligned} R^q &= E\left[\left(\varphi_0^q(t) + \tilde{\varphi}^q(t)\right)\left(\varphi_0(t) + \tilde{\varphi}(t)\right)^T\right] \\ &= E\left[\varphi_0^q(t)\varphi_0^T(t)\right] = R_0^q. \end{aligned} \tag{2.157}$$

From (2.131) and (2.157) we can finally derive the following set of q HOYW equation:

$$R^q\bar{\theta}^* = 0 \tag{2.158}$$

Note that (2.158) does not involve the output noise variance $\sigma_{\tilde{y}}^{2*}$. In this context, it is possible to search for $\sigma_{\tilde{u}}^{2*}$ within its range of possible values $\left[0, \sigma_{\tilde{u}_{max}}^2\right]$ by means of an optimization problem. For this purpose, let us define the cost function $J(\sigma_{\tilde{u}}^2)$ as

$$J(\sigma_{\tilde{u}}^2) = \left|\left|R^q\,\bar{\theta}(\sigma_{\tilde{u}}^2)\right|\right|_2^2, \quad \sigma_{\tilde{u}}^2 \in \left[0, \sigma_{\tilde{u}_{max}}^2\right] \tag{2.159}$$

where $\bar{\theta}(\sigma_{\tilde{u}}^2) = [\,1\;\theta(\sigma_{\tilde{u}}^2)\,]^T$ and $\theta(\sigma_{\tilde{u}}^2)$ is given by (2.152). This function has the following properties:

$$\begin{aligned} J(\sigma_{\tilde{u}}^2) &\geq 0 \\ J(\sigma_{\tilde{u}}^{2*}) &= 0. \end{aligned} \tag{2.160}$$

Then, $\sigma_{\tilde{u}}^{2*}$ and consequently $\theta(\sigma_{\tilde{u}}^{2*}) = \theta^*$ can be obtained by solving

$$\text{argmin}_{\sigma_{\tilde{u}}^2 \in \left[0, \sigma_{\tilde{u}_{max}}^2\right]}\; J(\sigma_{\tilde{u}}^2). \tag{2.161}$$

Remark 2.12. *Relation (2.158) represents a set of high order Yule-Walker equations that could be directly used to obtain the parameter vector θ^* provided that $q \geq n$. This approach can also be viewed as an instrumental variable method that uses delayed inputs as instruments (Söderström, 2018). However, the accuracy of IV methods is often poor since they require the estimation of high–lag autocorrelations (Söderström, 2018).*

In practice, the number of data available from the system is finite. This means we may only rely upon an estimate of the matrices R and R^q based on the N measurements of $u(t)$ and $y(t)$ at our disposal:

$$\hat{R} = \frac{1}{N}\sum_{t=1}^{N}\varphi(t)\varphi^T(t) = \begin{bmatrix} \hat{\sigma}_y^2 & \hat{r}_{yu}^T \\ \hat{r}_{yu} & \hat{R}_u \end{bmatrix} \tag{2.162}$$

$$\hat{R}^q = \frac{1}{N}\sum_{t=q+n+1}^{N}\varphi^q(t)\varphi^T(t). \tag{2.163}$$

In this case, an estimate $\hat{\sigma}_{\tilde{u}}^2$ of $\sigma_{\tilde{u}}^{2*}$ is found by minimizing the loss function

$$J(\sigma_{\tilde{u}}^2) = \left|\left|\hat{R}^q\,\bar{\theta}(\sigma_{\tilde{u}}^2)\right|\right|_2^2, \quad \sigma_{\tilde{u}}^2 \in \left[0, \hat{\sigma}_{\tilde{u}_{max}}^2\right] \tag{2.164}$$

where

$$\hat{\sigma}^2_{\tilde{u}_{max}} = \lambda_{min}\left(\hat{R}_u - \frac{\hat{r}_{yu}\hat{r}_{yu}^T}{\hat{\sigma}_y^2}\right), \tag{2.165}$$

The solution of Problem 1 is thus given by

$$\hat{\theta} = -\left(\hat{R}_u - \hat{\sigma}^2_{\tilde{u}} I_n\right)^{-1} \hat{r}_{yu}, \tag{2.166}$$

with

$$\hat{\sigma}^2_{\tilde{u}} = \text{argmin}_{\sigma^2_{\tilde{u}} \in \left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]} \; J(\sigma^2_{\tilde{u}}). \tag{2.167}$$

Remark 2.13. *Due to the ergodicity assumption (see A2-A4) it follows that*

$$\lim_{N\to\infty} \hat{R} = R, \quad \lim_{N\to\infty} \hat{R}^q = R^q, \quad \text{w. p. 1} \tag{2.168}$$

so that

$$\text{argmin}_{\sigma^2_{\tilde{u}} \in \left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]} \; J(\sigma^2_{\tilde{u}}) \lim_{N\to\infty} \text{argmin}_{\sigma^2_{\tilde{u}} \in \left[0, \sigma^2_{\tilde{u}_{max}}\right]} \; J(\sigma^2_{\tilde{u}}) \qquad \text{w. p. 1} \tag{2.169}$$

The whole identification procedure is summarized by the following algorithm.

Algorithm 2.9. *Starting from the noisy observations $u(1), \ldots, u(N)$ and $y(1), \ldots, y(N)$:*

1. *Compute the estimate $\hat{R}$ as in (2.162), and extract the highlighted blocks. Compute also the estimate $\hat{\sigma}^2_{\tilde{u}_{max}}$ as in (2.165).*
2. *Choose the number q of HOYW equations (see (2.154)) and compute the estimates $\hat{R}^q$ as in (2.163)*
3. *Initialize to a generic value of $\sigma^2_{\tilde{u}} \in \left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]$.*
4. *Compute the relative expression for the parameter vector $\theta(\sigma^2_{\tilde{u}})$ as in (2.152) and form $\bar{\theta} = [\, 1 \; \theta(\sigma^2_{\tilde{u}})^T \,]^T$.*
5. *Compute the cost function $J(\sigma^2_{\tilde{u}})$ (2.164).*
6. *Update $\sigma^2_{\tilde{u}}$ to a new value $\in \left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]$ corresponding to a decrease in $J(\sigma^2_{\tilde{u}})$.*
7. *Repeat the steps 4-6 until the value $\hat{\sigma}^2_{\tilde{u}}$ corresponding to the minimum of $J(\sigma^2_{\tilde{u}})$ is found. The obtained estimation $\hat{\theta}$ of the FIR coefficients is then given by $\hat{\theta} = \theta(\hat{\sigma}^2_{\tilde{u}})$.*

Remark 2.14. *Steps 4-6, corresponding to the search for the solution of the optimization problem (2.167), may be performed through any numerical search procedure. In particular, we deployed the Newton-Raphson method with parameter projection, as described in Appendix A.*

Remark 2.15. *Once that the estimates $\hat{\sigma}^2_{\tilde{u}}$, $\hat{\theta}$ of the input noise variance and the FIR coefficients have been obtained, an estimate of the output noise variance can be computed by means of (2.153):*

$$\hat{\sigma}^2_{\tilde{y}} = \hat{\sigma}^2_y + \hat{r}^T_{yu}\hat{\theta}. \tag{2.170}$$

From $\hat{\sigma}^2_{\tilde{u}}$ and $\hat{\sigma}^2_{\tilde{y}}$ it is possible to evaluate the signal to noise ratio on both the input and the output.

Algorithm 2.9 is tested using Monte Carlo simulations. Two different experiments have been considered. The first refers to a synthetic FIR whereas the second one concerns a FIR model representing the truncated impulse response of an asymptotically stable IIR system. A comparison with the Frisch scheme-based algorithm (Diversi et al., 2008a) and the total least squares (TLS) method (Davila, 1994) is also considered.

Experiment 1

Firstly, to test algorithm 2.9, we select a synthetic FIR model of length $n = 5$ already used in (Davila, 1994; Feng et al., 1998; Feng et al., 2006; Zheng, 2003; Feng et al., 2007; Diversi et al., 2008a; Diversi, 2008; Diversi, 2009) with the following coefficients:

$$\theta^* = [\,-0.3 \;\; -0.9 \;\; 0.8 \;\; -0.7 \;\; 0.6\,]^T. \tag{2.171}$$

The input $u_0(t)$ is an ARMA process defined by

$$u_0(t) = \frac{1 - 0.3z^{-1}}{1 - 0.9z^{-1}}\, e(t), \tag{2.172}$$

where $e(t)$ is a unit variance white noise. A first Monte Carlo simulation of 1000 runs has been carried out by considering additive white noise on both the input and the output. The noise variances $\sigma^{2*}_{\tilde{u}}, \sigma^{2*}_{\tilde{y}}$ have been selected in order to set the signal to noise ratio (SNR) to 5 dB on both the input and the output. A second Monte Carlo simulation of 1000 runs has then been performed by considering additive white noise $\tilde{u}(t)$ on the input and additive colored noise $\tilde{y}(t)$ on the output. In particular, the output noise is the following autoregressive process:

$$\tilde{y}(t) = \frac{1}{1 - 0.8z^{-1}}\, w(t). \tag{2.173}$$

The variance of $\tilde{u}(t)$ and of the driving white process $w(t)$ have been set to get an input and output SNR of 5 dB. In both Monte Carlo simulations the number of HOYW equations is set to $q = 2$ and the number of samples used is $N = 10000$.

The performance of Algorithm 1 has been compared to that of the Frisch scheme-based algorithm introduced in (Diversi et al., 2008a) and of the TLS method. The results of both the Monte Carlo simulations are summarized in Table I that reports the true values of FIR coefficients and noise variances, the mean of their estimates, and the corresponding standard deviations. As expected, when the input and output noise are white, all the algorithms lead to unbiased estimates. In this case, the approach in (Diversi et al., 2008a) has slightly better performance because of the smaller standard deviations of the estimated parameters. The TLS approach outperforms the other methods but requires the a priori knowledge of the noise variance ratio $\sigma^{2*}_{\tilde{y}} / \sigma^{2*}_{\tilde{u}}$. When the output noise is colored, Algorithm 1 still gives good results whereas the algorithm in (Diversi et al., 2008a) clearly leads to biased estimates. Again, the TLS method leads to better results but is based on information that is seldom available in practice.

In order to show the effectiveness of the cost function $J(\sigma^2_{\tilde{u}})$, Fig. 2.6 reports its values over the interval $\left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]$ in a run of the second Monte Carlo simulation (colored noise). It is worth noting that the minimum of $J(\sigma^2_{\tilde{u}})$ occurs at a value of $\sigma^2_{\tilde{u}}$ very close to the true one. All other runs of the Monte Carlo simulations exhibit a similar behavior.

Experiment 2

The second experiment aims to identify a FIR model representing the truncated impulse response of an asymptotically stable IIR system corrupted by white input noise and colored

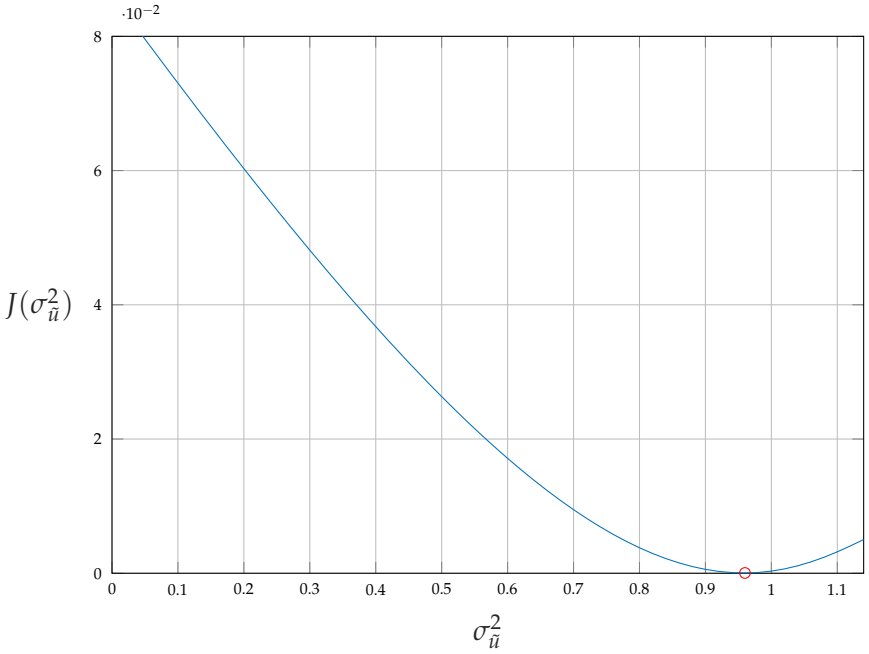

FIGURE 2.6: Values of the cost function $J(\sigma^2_{\tilde{u}})$ cost function over the interval $\left[0, \hat{\sigma}^2_{\tilde{u}_{max}}\right]$ in one of the Monte Carlo runs. The red circle corresponds to the minimum value.

TABLE 2.2: True and estimated values of the FIR coefficients and of the input and output noise variances. Monte Carlo simulation of 1000 runs performed with $N = 10000$. Synthetic FIR.

	h_0	h_1	h_2	h_3	h_4	$\sigma^{2*}_{\tilde{u}}$	$\sigma^{2*}_{\tilde{y}}$
True	-0.3	-0.9	0.8	-0.7	0.6	0.96	0.67
				$\tilde{u}(t)$ white, $\tilde{y}(t)$ white			
Algorithm 2.9	-0.302 ± 0.026	-0.892 ± 0.085	0.794 ± 0.115	-0.694 ± 0.095	0.594 ± 0.054	0.949 ± 0.065	0.678 ± 0.142
(Diversi et al., 2008a)	-0.303 ± 0.024	-0.895 ± 0.033	0.797 ± 0.043	-0.696 ± 0.034	0.597 ± 0.024	0.956 ± 0.031	0.674 ± 0.043
TLS	-0.301 ± 0.026	-0.898 ± 0.030	0.799 ± 0.029	-0.699 ± 0.033	0.599 ± 0.022	0.959 ± 0.013	0.669 ± 0.009
				$\tilde{u}(t)$ white, $\tilde{y}(t)$ colored			
Algorithm 2.9	-0.302 ± 0.025	-0.891 ± 0.089	0.792 ± 0.121	-0.693 ± 0.099	0.594 ± 0.061	0.947 ± 0.069	0.681 ± 0.153
(Diversi et al., 2008a)	-0.309 ± 0.015	-0.666 ± 0.020	0.492 ± 0.019	-0.446 ± 0.019	0.435 ± 0.017	0.713 ± 0.026	1.077 ± 0.024
TLS	-0.301 ± 0.025	-0.899 ± 0.027	0.800 ± 0.027	-0.700 ± 0.031	0.599 ± 0.020	0.959 ± 0.012	0.669 ± 0.008

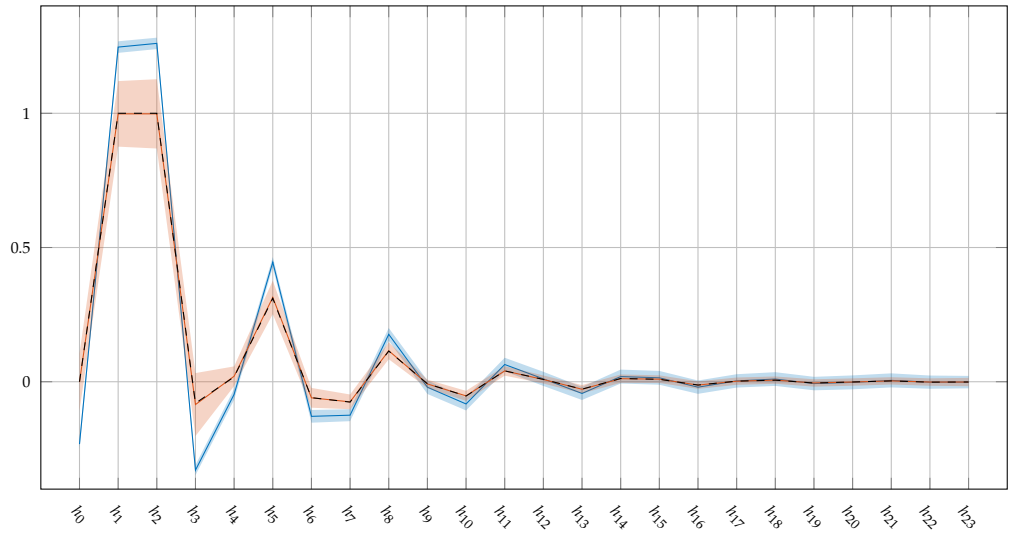

FIGURE 2.7: True truncated impulse response of the IIR system (2.174) (dashed black), mean of its estimate obtained with algorithm 2.9 in red and with Algorithm (Diversi et al., 2008a) in blue and associated standard deviations in transparency. Monte Carlo simulation of 1000 runs performed with $N = 10000$ and SNR$=$ 10dB.

output noise. We consider the following IIR model

$$y_0(t) = \frac{(z+0.6)(z+0.5)}{(z-0.5)(z^2+0.6z+0.58)}\, u_0(t), \tag{2.174}$$

where the input $u_0(t)$ is the ARMA process (2.172), the additive input noise $\tilde{u}(t)$ is a white process and the additive output noise $\tilde{y}(t)$ is the colored process (2.173). Starting from $N = 10000$ samples of the noisy input and output, $u(t)$ and $y(t)$, the truncated impulse response of the system (a FIR model with length $n = 24$) has been identified by means of algorithm 2.9 and the algorithm in (Diversi et al., 2008a). The hyperparameter q has been set to 24. Two different values of the input and output noise variances have been considered to get SNR$=$ 10 dB and SNR$=$ 5 dB respectively. For each value of the SNR a Monte Carlo simulation of 1000 runs has been performed. The results are shown in Figs. 3 and 4, reporting the first 24 coefficients of the impulse response, the mean of the obtained estimates, and the associated standard deviations. These figures confirm the good performance of the proposed approach. Again, the estimates obtained by using the approach in (Diversi et al., 2008a) are clearly biased.

Notice that in both experiments the number of HOYW equations q used to achieve consistent estimation is very small. This, together with the use of the Newton-Raphson method, leads to a relatively low computational load of the proposed approach.

2.7 Model Order Selection

In system identification, both the determination of model structure and model validation are essential aspects. An over-parametrized model structure can lead to unnecessarily complicated computations for finding the parameter estimates and using the estimated model. An under parameterized model may be wildly inaccurate. This section aims to present some

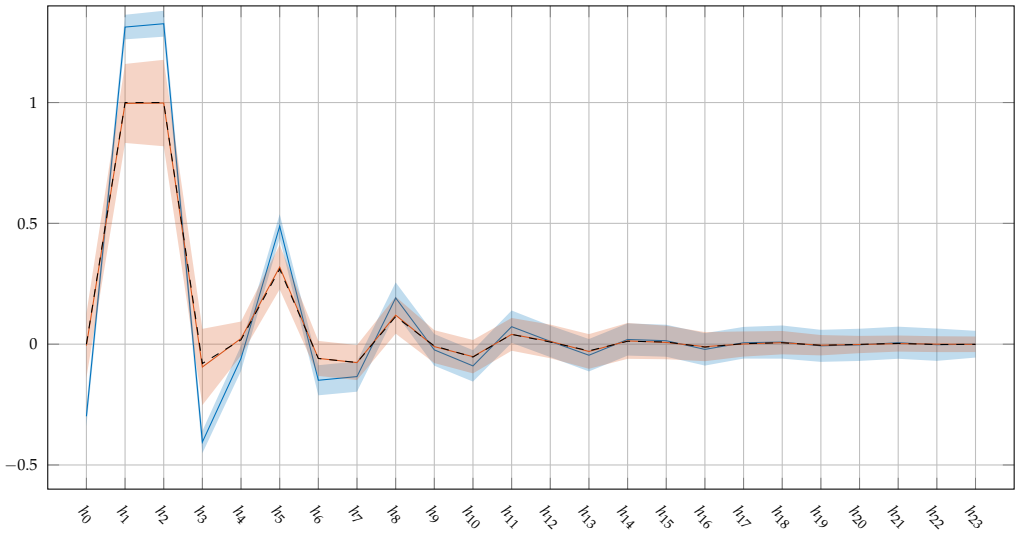

FIGURE 2.8: True truncated impulse response of the IIR system (2.174) (dashed black), mean of its estimate obtained with Algorithm 2.9 in red and with Algorithm (Diversi et al., 2008a) in blue and associated standard deviations in transparency. Monte Carlo simulation of 1000 runs performed with $N = 10000$ and SNR$=$ 5dB.

basic methods that can be used to find an appropriate model structure complexity (Söderström et al., 1989). The intended use of the model significantly influences the choice of the model structure in practice. A stabilizing regulator can often be based on a rough, low-order model. In contrast, more complex and detailed models are necessary if the model is aimed at giving physical insight into the process. In MoS, the signal underlying model structure's characterization aims at the most suitable data fitting to perform diagnostics and prognostics of faults. In practice, we often perform identification for an increasing set of model orders (or more generally, structural indices). Then we must know when the model order is appropriate, i.e., when to stop. Any real-life data set cannot be modeled precisely by a linear finite-order model. Nevertheless, such models often give good approximations of the true dynamics. However, finding the "correct" model order is based on the statistical assumption that the data come from a true system within the model class considered.

In this work, we make use of AR, ARMA, and FIR processes to represent the measured signals, which in turn requires defining a proper model order n. Many criteria are available to define the model complexity, they are based on the statistical properties of the residual of the identification, which is nothing but the equation error (2.7).

Consider one of the said model structures of order n, say the AR structure, and the associated parameter vector $\hat{\theta}_n$ identified by applying one of the previously mentioned methods to a set of N measurements $y(1), y(2), \ldots, y(N)$. Then, using the model equation in regression form

$$\varepsilon(t, \hat{\theta}_n) = y(t) - \varphi^T(t)\,\hat{\theta}_n \tag{2.175}$$

we obtain the corresponding residual sequence

$$\varepsilon(1, \hat{\theta}_n), \ldots, \varepsilon(N, \hat{\theta}_n). \tag{2.176}$$

Then, the variance of the residuals

$$J(\hat{\theta}_n) = \frac{1}{N}\sum_{i=1}^{N}\varepsilon^2(t,\hat{\theta}_n) \tag{2.177}$$

can be computed for every $n \in \mathcal{N}$ that we intend to investigate to assess the model complexity, producing a cost function that is similar to eq. (2.11), but in this case, the optimal model for the particular order is assumed computed. The catch here is that the model with the right order should produce the minimum error variance. In practice, this function alone is not enough since as we increase the order of the computed models and perform the estimation on the available dataset, overfitting may happen. Which means that the model is accurately mapping this particular set of measurement more than estimating the underlying system generating it. There are two possible ways to avoid this problem: modify eq. (2.177) with the addition of complex terms that penalize higher orders, which is what the model order estimation criteria do, or using a second set of measurements of $y(t)$ to generate a second sequence of residuals in a cross-validation fashion. Nevertheless, those two solutions can be combined to obtain more robust results. Given those considerations, and depending on the available data for monitoring, various options and criteria can be investigated, starting from the ones based on eq. (2.177). FPE, AIC and MDL are criteria with complexity terms that consist in selecting the order n leading to the minimum of the following loss functions (Söderström et al., 1989; Ljung, 1999):

$$FPE(n) = \frac{N+n}{N-n}J(\hat{\theta}_n) \tag{2.178}$$

$$AIC(n) = N\log(J(\hat{\theta}_n)) + 2n \tag{2.179}$$

$$MDL(n) = N\log(J(\hat{\theta}_n)) + n\log N, \tag{2.180}$$

where FPE refers to the Final Prediction Error criterion, AIC to the Akaike Information Criterion and MDL to the Minimum Description Length criterion.

The previous functions provide an idea of what is going to be the best model order that fits the identification problem. To validate the obtained guess, the family of statistical tests is introduce, and in involves binary hypothesis testing. The F-test method is useful to perform the comparison of two different model orders: n_1 and n_2, with $n_1 < n_2$. It consists in computing the test quantity

$$\eta_{n_1,n_2} = N\frac{J(\hat{\theta}_{n_1}) - J(\hat{\theta}_{n_2})}{J(\hat{\theta}_{n_2})} \tag{2.181}$$

and performing a χ^2 distribution test with $n_2 - n_1$ degrees of freedom upon that quantity (Söderström et al., 1989).

$$\begin{cases} H_0: & \eta_{n_1,n_2} \sim \chi^2(n_2-n_1) \\ H_1: & \text{not } H_0 \end{cases} \implies \begin{cases} H_0: & \text{accept the model order } n_1 \\ H_1: & \text{not } H_0 \end{cases} \tag{2.182}$$

In this way, the test is passed when n_1 is more suitable for the characterization of the model structures than n_2. In other words, the test is passed when the chosen model order is a better approximation of the underlying system than than the other.

On the other hand, another way to validate the estimated model and its complexity is to check if the assumptions upon its residuals are true. If we take into account an AR system, the obtained residuals should represent a zero mean white process. Thus, a way to validate the obtained models is to perform a whiteness test upon $\varepsilon(t,\hat{\theta}_n)$ (Söderström et al., 1989;

Ljung, 1999). It is based on the following quantity

$$\xi_{N,m} = N \frac{\hat{r}_\varepsilon^{m^T} \hat{r}_\varepsilon^m}{J(\hat{\theta}_n)^2}, \tag{2.183}$$

that is a function of the autocorrelation of the residuals

$$\hat{r}_\varepsilon^m = \frac{1}{N} \sum_{i=1}^{N} \begin{bmatrix} \varepsilon(t-1, \hat{\theta}_n) \\ \vdots \\ \varepsilon(t-m, \hat{\theta}_n) \end{bmatrix} \varepsilon(t, \hat{\theta}_n). \tag{2.184}$$

Besides, it possible to prove that $\xi_{N,M}$ is asymptotically chi-square distributed with m degrees of freedom:

$$\xi_{N,m} \lim_{N \to \infty} \sim \chi^2(m), \tag{2.185}$$

when the obtained $\varepsilon(t, \hat{\theta}_n)$ residuals are white. The test in this case is the following

$$\begin{cases} H_0: & \xi_{N,m} \sim \chi^2(m) \\ H_1: & \text{not } H_0 \end{cases} \Longrightarrow \begin{cases} H_0: & \text{accept the model} \\ H_1: & \text{not } H_0 \end{cases} \tag{2.186}$$

The application of MoS to real case studies that will be depicted later on will involve also this analysis about the model complexity and validation. The criteria used in each of them will be pointed out.

It is important to point out that the previous discussion is about investing in the complexity of model structures that do not involve noisy measurements. This is done for two main reasons, the case studies and applications presented later on do not involve noisy model structures and algorithm 2.8 is still in the early stages of development. However, if the reader is interested in the topic, we refer to (Diversi et al., 2008b) for a discussion on the model order of AR plus noise processes.

Chapter 3

Diagnostics and Prognostics tools

This chapter introduces to readers the theoretical foundations and definition of the tools used in combination with MoS to develop PHM procedures. First of all, we describe the commonly used health indicators that can be found or implemented inside the machine controller environment are described. Since they are suitable for edge-computing they are used as a challenger for MoS and compared in terms of fault detection and isolation performance. Then, model distance metrics are presented, since they are a valuable health indicator that can be exploited for PHM. After that, a brief introduction to machine learning and the algorithms employed in the various case studies is presented, and, finally, we conclude with the basics of particle filtering for prognostics.

3.1 Statistical Indexes and Model Distances

Statistical indexes on signals are often used for condition monitoring in automation. It is also common among vendors of real-time controllers to integrate analog sensor modules with computing units able to provide those sorts of indexes alongside the measurements. However, their use as monitoring indicators may result not robust in terms of carried information about the system. In fact, in complex machinery, it is advisable to use them together and the way they are combined needs some analysis to perform suitable feature extraction for diagnosis, as we will show later on. In this context, MoS inherently contains features linked dynamically to the modelled signal correlated with the machinery physics. This bigger amount of information about the machine state is computed and transported with a similar effort with respect to statistical indexes. In several case studies in this dissertation, we compare the usage of signal statistical quantity indicators and MoS indicators for diagnostic purposes. In table 3.1 a list of the most used statistical ones is presented, as stated in (Rauber et al., 2015), and they are:

RMS: the Root Mean Square is nothing but the square root of the arithmetic mean of the squares of the values of a given set of measurements. It is typically associated with the power contained within the signal, as happens in the electrical field.

KV: the Kurtosis Value is a measure of the "tailedness" of the probability distribution of a real-valued random variable. In this case, it quantifies the importance of the tail extremity of the signal distribution around its mean and is therefore sensitive to outliers.

PPV: the Peak-to-Peak Value is defined the difference between the maximum and the minimum signal levels. This is commonly much more sensitive to noise and to impulse disturbances, such as sudden impacts or cracks within mechanisms monitored with vibrations.

TABLE 3.1: Statistical quantities computed for the measured signals, N is the number of samples x_i with $i = 1, \ldots, N$

$X_{rms} = \left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^{1/2}$	$X_{kv} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\sigma}\right)^4$
$X_{ppv} = \max(x_i) - \min(x_i)$	$X_{sv} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_i-\bar{x}}{\sigma}\right)^3$
$X_{sra} = \left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{\lvert x_i\rvert}\right)^2$	$X_{cf} = \frac{\max(\lvert x_i\rvert)}{X_{rms}}$
$X_{if} = \frac{\max(\lvert x_i\rvert)}{\frac{1}{N}\sum_{i=1}^{N}\lvert x_i\rvert}$	$X_{mf} = \frac{\max(\lvert x_i\rvert)}{X_{sra}}$
$X_{sf} = \frac{X_{rms}}{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^2}$	$X_{kf} = \frac{X_{kv}}{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^2}$

SV: the Skewness Value is a measure of the degree of symmetry around the mean, so that symmetric distributions feature skewness 0, while the value can become either positive or negative if the mean value moves right or left with respect to the peak of the distribution (the mode), respectively.

SRA: the Square Root of the Amplitude is close to the interpretation of the RMS value (in many cases they behave equally) and is particularly useful in monitoring through vibration and periodic currents.

CF: the Crest Factor is a parameter of a waveform, such as alternating current or sound or vibration, showing the ratio of peak values to the effective value. In other words, the crest factor indicates how extreme the peaks are in a waveform. Crest factor 1 indicates no peaks, such as direct current or a square wave. Higher crest factors indicate peaks, for example, sound waves tend to have high crest factors.

IF: the Impulse Factor is a measure to compare the height of a peak to the mean level of the signal. In bearings, it is used to measure how much impact is generated from the defects. defect

MF: the Margin Factor is calculated by dividing the maximum absolute value of the signal with the RMS of the absolute value of the signal. In bearings, it measures the level of impact between rolling element and raceway.

SF: the Shape factor is the RMS divided by the mean of the absolute value. It is dependent on the signal shape while being independent of the signal dimensions.

KF: the Kurtosis Factor is the KV divided by the mean of the absolute value. It is dependent on the signal "tailedness" while being independent of the signal dimensions.

The use of model distances, on the other hand, allows for the monitoring of system conditions by relying on how far from a known healthy state model the on-line estimation is. This requires at least one reference model θ_{ref} to be computed for a given machinery condition. Its definition may be done either during the data pre-processing phase, i.e., when also the model order is defined, or during the first cycles of machine working in nominal operation.

The first metric that we want to introduce is the Normalized Root Mean Square Error (NRMSE):

$$NRMSE = \sqrt{\frac{\left\|\hat{\theta} - \theta_{ref}\right\|}{\left\|\theta_{ref}\right\|}} \tag{3.1}$$

which is about computing the distance between the coefficients of the reference model θ_{ref} and the current estimated model $\hat{\theta}$.

On the other hand, the second that we want to introduce is the symmetric Itakura-Saito spectral distance (Wei et al., 2000; Itakura, 1968)

$$IS = \frac{1}{2\,N_f} \sum_{k=1}^{N_f} \left(\frac{S_{ref}(f_k)}{\hat{S}(f_k)} - \log \frac{S_{ref}(f_k)}{\hat{S}(f_k)} + \frac{\hat{S}(f_k)}{S_{ref}(f_k)} - \log \frac{\hat{S}(f_k)}{S_{ref}(f_k)} - 2 \right), \tag{3.2}$$

where $S_{ref}(f_k)$ denotes the power spectral density (PSD) of the reference AR model θ_{ref}, $\hat{S}(f_k)$ is the PSD of the current estimated AR model $\hat{\theta}$ and N_f is the number of considered frequencies. Then, the IS distance is a measure of how close to each other are the spectra of the estimated model and of the reference one.

3.2 Machine Learning Tools

The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. In recent years, many successful machine learning applications have been developed, ranging from data-mining programs that learn to detect fraudulent credit card transactions, to information-filtering systems that learn users' reading preferences, to autonomous vehicles that learn to drive on public highways. At the same time, there have been important advances in the theory and algorithms that form the foundations of this field (Mitchell et al., 1997). Machine learning is programming computers to optimize a performance criterion using example data or past experience. We have a model defined up to some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions in the future, or descriptive to gain knowledge from data, or both (Alpaydin, 2020).

Given those premises, machine learning has gained importance also in the field of diagnostics and prognostics of machinery and their components. The data and the expertise available, and the goals of PHM fall suitably in the class of problems that can be solved by using learning algorithms. Indeed, they fall within the data-driven data processing approaches (Cerrada et al., 2018; Zhang et al., 2017; Si et al., 2011; Mosallam et al., 2016) providing viable solutions to solve the maintenance problem.

In this work, machine learning is used as a knowledge refinement tool that converts the information about the machinery working conditions contained within the models computed through MoS. Since the major interest of the whole research was about the use of MoS and its integration in industrial automation, we will describe only the tools that have been employed in combination with MoS to deliver PHM information. In general, machine learning is not edge-computing friendly since the training of the various predictors and classifiers typically cannot be done on the edge, as it is for MoS. However, it is possible to exploit the automation pyramid structure to employ supervisor and management level computer to perform this training upon the data and model the machines provide and send back to the controllers the obtained predictions and classifications, and in some cases the predictor. In the following, a brief theoretical overview of the tools employed in this work is presented. In particular, we made use of supervised learning algorithms, such as Logistic Regression (LR), Support Vector Machines (SVM) in the case of fault detection and isolation and degradation severity tracking, and Particle Filtering (PF) for prognostics.

For a detailed discussion of machine learning, we refer to (Vapnik, 1995; Mitchell et al., 1997; Alpaydin, 2020; Bishop, 2006) within the huge library available at this moment in time about the topic.

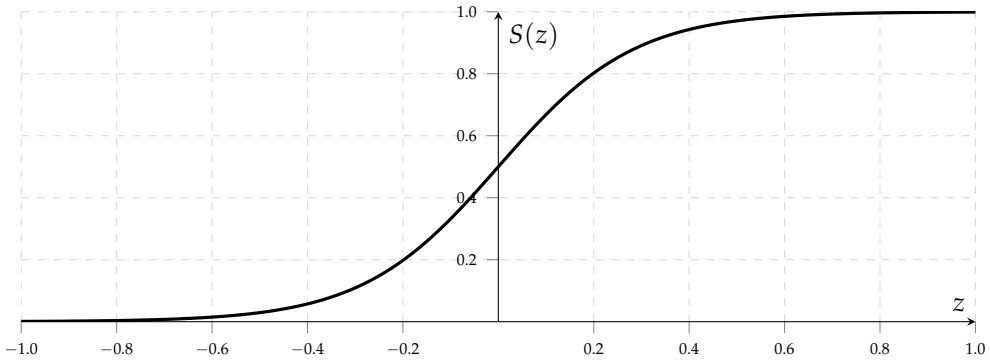

FIGURE 3.1: Sigmoid function: $S(z) = \frac{1}{1+e^{-z}}$

3.2.1 Logistic Regression

The logistic regression (LR) is one of the basic machine learning algorithms whose goal is to classify data (Shalev-Shwartz et al., 2014). It is trained by means of a set $\mathcal{X}$ of sampled features x together with their known labels y belonging to set $\mathcal{Y}$. It learns a decision boundary described by the parameter vector γ that is used by the predictor to classify new models. In particular, the predicting function h_γ should output a number in the range $[0,1]$ which is then rounded to either 0 or 1 by means of a threshold. Formally, the prediction $\hat{y}$ should be as such:

$$\hat{y} = \begin{cases} 0, \text{ if } h_\gamma < 0.5 \\ 1, \text{ if } h_\gamma \geq 0.5 \end{cases} \tag{3.3}$$

To this end, We choose the sigmoid function defined in Figure 3.1. Its argument z is taken and mapped into the range $[0,1]$. If $z = 0$ then $S(z) = 0.5$ that is also addressed as the labeling threshold. This mapping results very useful and easy to understand in the case of a linear predictor with decision boundary γ.

Cost function

Given a set of N training samples, denoted as the couples $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)})$ of features and labels, then the cost function J to be applied during the training is the following:

$$\begin{aligned} J(\gamma) = -\frac{1}{N} \Bigg[\sum_{i=1}^{N} y^{(i)} \log \left(h_\gamma \left(x^{(i)} \right) \right) + \\ + \left(1 - y^{(i)} \right) \log \left(1 - h_\gamma \left(x^{(i)} \right) \right) \Bigg], \end{aligned} \tag{3.4}$$

where $y^{(i)} \log(h_\gamma(x^{(i)}))$ is the sample contribution in the case it has known label 1 with respect to the given boundary γ, while $(1 - y^{(i)}) \log(1 - h_\gamma(x^{(i)}))$ is the case for 0 labels. The function

$$h_\gamma \left(x^{(i)} \right) = \frac{1}{1 + e^{-\gamma^T x^{(i)}}} \tag{3.5}$$

is the predictor that uses the sigmoid function to address the labeling probability.

Gradient Descent

The problem now is to get the right decision boundary γ, which corresponds to finding the minimum of the cost function $J(\gamma)$. This, formally, is a minimization problem of the form:

$$\min_{\gamma} \quad J(\gamma) \tag{3.6}$$

which, in this case, falls into the optimization framework and it can be solved by any algorithm able to find functions minima (either local or global).

In this analysis, we apply the gradient descent (Shalev-Shwartz et al., 2014): an iterative algorithm that exploits the Jacobian of the cost function to find a minimum, given some starting value of the parameter vector γ. Its update law may be synthesized in the following:

$$\gamma_{j+1} = \gamma_j - \alpha \frac{\partial J(\gamma)}{\partial \gamma_j} \tag{3.7}$$

where j is the iteration index and α is the learning rate of the algorithm. The latter has to be large enough to let the descent reach a minimum in a reasonable time and sufficiently small to guarantee that the descent actually reduced the value of $J(\gamma)$.

One Vs All

LR method may be extended to the case in which the label set $\mathcal{Y}$ contains K labels, with $K > 2$. As previously defined, the set of samples is given by $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)})$, however, this time their classes are denoted by $y^{(i)} = j$ with $j = 1, 2, \dots, K$. To tackle the increased number of classes it is still possible to rely upon the binary LR, but in a different fashion. The One vs All method is the result of the application of LR, performed by assigning label "1" to the samples labelled as j and label "0" to all the others, for every label within the label set. Therefore, the procedure requires the training of K predictors h_{γ_j}, all characterized by their related decision boundary γ_j with $j = 1, 2, \dots, K$, solving exactly the same problem as in (3.4). In the end, we may group together the h_{γ_j}s into the classifier H_γ defined in the following:

$$H_\gamma(x^{(i)}) = S\left(\Gamma^T x^{(i)}\right) \tag{3.8}$$

with $S(\cdot)$ being the element-wise sigmoid function and

$$\Gamma = \begin{pmatrix} | & | & & | \\ \gamma_1 & \gamma_2 & \cdots & \gamma_K \\ | & | & & | \end{pmatrix}_{n \times K}, \quad H_\gamma = \begin{pmatrix} h_{\gamma_1} \\ h_{\gamma_2} \\ \vdots \\ h_{\gamma_K} \end{pmatrix}_{K \times 1}. \tag{3.9}$$

The predictor results in a probability vector of dimension K in which every element h_{γ_j} expresses the probability of the feature sample to be classified by the given label j. Formally, h_{γ_j} represents $P(y = j | x^{(i)}, \gamma_j)$. Then, the label is assigned by selecting the one with the higher probability, namely:

$$\hat{y} = \arg\max_j \; h_{\gamma_j} \tag{3.10}$$

The labeling accuracy of h_γ over a set of N testing samples $x^{(i)}$ with true label y_i is the following:

$$L_s(h_\gamma) = \frac{|\{i \in [N] : \hat{y}^{(i)} = y^{(i)}\}|}{N} \times 100, \tag{3.11}$$

where $[N] = \{1 \dots N\}$.

3.2.2 Support Vector Machines

Support Vector Machine, abbreviated as SVM can be used for both regression and classification tasks, thus falling within the class of supervised learning algorithms (Vapnik, 1995; Cortes et al., 1995; Bishop, 2006). It is widely used in classification objectives and is highly preferred by practitioners as it produces significant accuracy with less computation power.

The goal of the linear SVM, similar to LR, is to find decision boundary, which in this case is called separation hyperplane. The objective in this case it to find a plane that has the maximum margin, i.e the maximum distance between data points of two defined classes. In the end, the solution of the SVM problem comes from the solution of an optimization one, as was for LR. Thus, given a set of N observations divided in two classes, denoted as the couples $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)})$, with $y_i \in \{-1, 1\}$, we want to find a separation hyperplane, defined from the parameter (w, b) that has the form:

$$D(x) = w^T \cdot x + b = 0, \tag{3.12}$$

where $b \in \mathbb{R}$ is the bias and $w \in \mathbb{R}^n$ is the vector perpendicular to the hyperplane. $D(x) = 0$ represents the separation hyperplane and the distance of a generic vector x_i, with $i = 1, \dots, N$, from it is given by

$$r_i = \frac{D(x_i)}{\|w\|}. \tag{3.13}$$

Then, labels indicate the position of x_i with respect to $D(x)$ and the parameters (w, b) should be found such that

$$y_i \left[w^T x_i + b \right] \geq 1 \text{ for } i = 1, \dots, N, \tag{3.14}$$

introducing a constraint regarding the definition of $D(x)$, where the vectors belonging to the planes $D(x) = 1$ and $D(x) = -1$ have minimum distance form the separation plane and are defined as the support vectors. Those two planes parallel to $D(x)$ are called margins and the margin

$$\tau = \frac{2}{\|w\|} \tag{3.15}$$

between them is the one that we are trying to maximize. Given those considerations, the SVM optimization problem is the following

$$\min_{(w,b)} \quad \frac{\|w\|^2}{2} \tag{3.16}$$

$$\text{subject to} \quad y_i \left[w^T x_i + b \right] - 1 \geq 0 \text{ for } i = 1, \dots, N, \tag{3.17}$$

Then, the solution of this problem is obtained by using its dual formulation with the lagrangian multipliers and exploiting the Karush-Kuhn-Tucker (KKT) conditions. The result is a quadratic optimization problem to be solved using one of the available techniques, such as the gradient descent. The final hyperplane is parameterized as follows:

$$w = \sum_{x_i \in SV} \lambda_i y_i x_i, \tag{3.18}$$

$$b = (y_i - \sum_{x_j \in SV} \lambda_j y_j x_i^T x_j, \tag{3.19}$$

where $\lambda = \left(\lambda_1 \dots \lambda_N\right) \in \mathbb{R}^N_{\geq 0}$ are Lagrangian multipliers and SV is the set of support vectors. This parameterized solution is suitable only in the case the observation of the two classes are all well separated. If this is not the case for the available dataset, the separation constraints may be relaxed using N slack variables $\xi_i > 0$ to allow for outliers between the classes. This translate into new constraints equations

$$y_i \left[w^T x_i + b\right] - 1 \geq 0 \text{ for } i = 1 - \xi, \dots, N, \tag{3.20}$$

which leads to this slightly different characterization of the problem, introducing the regularization coefficient that allows to still pursue the maximization of the separation margin trying to avoid the non-reparable items.

$$\min_{(w,b)} \quad \frac{\|w\|^2}{2} + C \sum_{xi_i \in M} \xi_i \tag{3.21}$$

$$\text{subject to} \quad y_i \left[w^T x_i + b\right] - 1 + \xi_i \geq 0 \text{ for } i = 1, \dots, N, \tag{3.22}$$

$$\xi_i > 0, \tag{3.23}$$

where C is the regularization parameter that penalizes the non-separable observations. Then, it is possible to obtain the following parameterization of the (w, b) hyperplane:

$$w = \sum_{x_i \in SV} \lambda_i y_i x_i, \tag{3.24}$$

$$b = \frac{1}{|M|} \sum_{x_i \in M} \left(y_i - \sum_{x_j \in SV} \lambda_j y_j x_i^T x_j \right), \tag{3.25}$$

where M is the set of support vectors whose corresponding λ_i are lower than the regularization parameter that penalizes classification errors in the case of non-separable classes.

In the case we have to deal with a classification problem with $k > 2$ classes, in this work we adopt the one-versus-one approach, which is known to be robust concerning this learning task (Bishop, 2006). It consists of training $\frac{K(K-1)}{2}$ binary SVMs on all the possible pair of classes and classifying the observations as belonging to the class that results in a higher number of occurrences.

3.2.3 Particle Filtering

Particle Filtering (PF) is based on the recursive Bayesian estimation framework (Gordon et al., 1993). It starts by considering the following state space model:

$$x_k = f(x_{k-1}, \omega_{k-1}) \tag{3.26}$$

$$y_k = h(x_k, v_k), \tag{3.27}$$

where x_k is the state vector; $f(\cdot)$ is the possibly non-linear state transition function; $h(\cdot)$ is the state to output mapping; ω_k and v_k are the independent identically distributed process and measurement noise respectively with known statistics; and $k \in \mathbb{N}$ index refers to the time step.

The goal is to reconstruct the Probability Density Function (PDF) of the current state x_k, defined as $p(x_k)$, exploiting the information collected through the observations sequence $y_1, y_2, \dots, y_k$, denoted $y_{1:k}$ compactly, given the conditional PDF $p(x_k|y_{1:k})$. This posterior probability can be obtained by means of a recursive procedure employing two main stages:

prediction and update. Given the initial state PDF $p(x_0) = p(x_0|y_0)$, we start the description of the aforementioned stages at time $k-1$, given the conditional PDF $p(x_{k-1}|y_{k-1})$.

The prediction stage makes use of Eq.(3.26) to obtain the prior PDF of the state vector at step k. This is used within the Chapman-Kolmogorov equation:

$$\begin{aligned} p\left(x_k|\ y_{1:k-1}\right) &= \\ &= \int p\left(x_k|x_{k-1}, y_{1:k-1}\right) p\left(x_{k-1}|y_{1:k-1}\right) dx_{k-1} \end{aligned} \tag{3.28}$$

where, if we assume the underlying process to be a first-order Markov one:

$$p\left(x_k|x_{k-1}, y_{1:k-1}\right) = p\left(x_k|x_{k-1}\right), \tag{3.29}$$

we get the following equation:

$$p\left(x_k|y_{1:k-1}\right) = \int p\left(x_k|x_{k-1}\right) p\left(x_{k-1}|y_{1:k-1}\right) dx_{k-1}. \tag{3.30}$$

Then, once the new observation y_k is available it is possible to perform the update stage by using the Bayes' Rule:

$$p\left(x_k|y_{1:k}\right) = \frac{p\left(y_k|x_k\right) p\left(x_k|y_{1:k-1}\right)}{p\left(y_k|y_{1:k-1}\right)} \tag{3.31}$$

to get the posterior PDF of the state. The normalizing constant

$$p\left(y_k|y_{1:k-1}\right) = \int p\left(y_k|x_k\right) p\left(x_k|y_{1:k-1}\right) dx_k \tag{3.32}$$

depends on the likelihood function $p\left(y_k|x_k\right)$ which is based on (3.27). The recursion of equations (3.28) and (3.31) pave the way to the optimal solution of this Bayesian framework.

The Bayesian solution is just a conceptual one, since, due to its complexity, it is possible to attain it analytically only under certain assumptions or conditions, such as Kalman Filters where all posterior PDFs are assumed to be Gaussian. Because of this reason, sub-optimal filters are required to at least approximate the solution.

One of the most used sub-optimal filters is the Particle Filter (PF), which is based on the Sequential Monte Carlo (SMC) method. The main idea of this filter is to exploit recursively MC simulations to represent the required posterior density function by a set of random samples with associated weights and to compute estimates based on these samples and weights. A suitably large number of samples, called particles, in this case, can characterize an equivalent representation of the posterior PDFs involved in the framework. A thorough explanation of such methods is presented in (Arulampalam et al., 2002).

The particle filtering approach proposed in this paper is Sampling Importance Resampling (SIR), which is used widely in the prognostic field (Tulsyan et al., 2016; An et al., 2013; Saha et al., 2011; Skima et al., 2016), and makes use of N_p weighted particles:

$$\{\langle x_k^i, w_k^i\rangle ; i = 1, \ldots, N_p\}, \text{ compactly } \{x_k^i, w_k^i\}_{i=1}^{N_p}, \tag{3.33}$$

to approximate $p(x_k|y_{1:k})$, where w_k^i is the weight of particle i at time k.

Starting from the initial distribution $p(x_0)$, approximated by the sequence $\{x_0^i\}_{i=1}^{N_p}$ with uniform weights $\{w_0^i = \frac{1}{N_p}\}_{i=1}^{N_p}$, we summarize the steps to accomplish SIR Particle Filtering within the recursive Bayesian filtering framework in the following:

Algorithm 3.1 (SIR PF)**.**

1. ***Prediction***: *compute the new a priori PDF of the state x_k by propagating through model described by Eq. (3.26) the particle set from $\{x_{k-1}^i\}_{i=1}^{N_p}$ to $\{x_k^i\}_{i=1}^{N_p}$. In this way, the approximation of $p\left(x_k|x_{k-1}\right)$, which is the state transition PDF, is obtained.*

2. ***Update***: *once the new measurement y_k is available, the likelihood function $p\left(y_k|x_k\right)$ is exploited to compute importance weights according to:*

$$\left\{ w_k^i = \frac{p\left(y_k|x_k^i\right)}{\sum_{i=1}^{N_p} p\left(y_k|x_k^i\right)} \right\}_{i=1}^{N_p} \tag{3.34}$$

3. ***State estimation***: *the estimate is given by the particle mean value:*

$$\hat{x}_k = \frac{1}{N_p} \sum_{i=1}^{N_p} x_k^i w_k^i \tag{3.35}$$

4. ***Re-sample***: *Re-sample from $\{x_k^i, w_k^i\}_{i=1}^{N_p}$, proportionally to the new computed weights, the new sequence of particles $\{x_k^i\}_{i=1}^{N_p}$ is obtained. This sequence distribution is recursively approximated by $p(x_k|y_k)$.*

Fault Prognostics

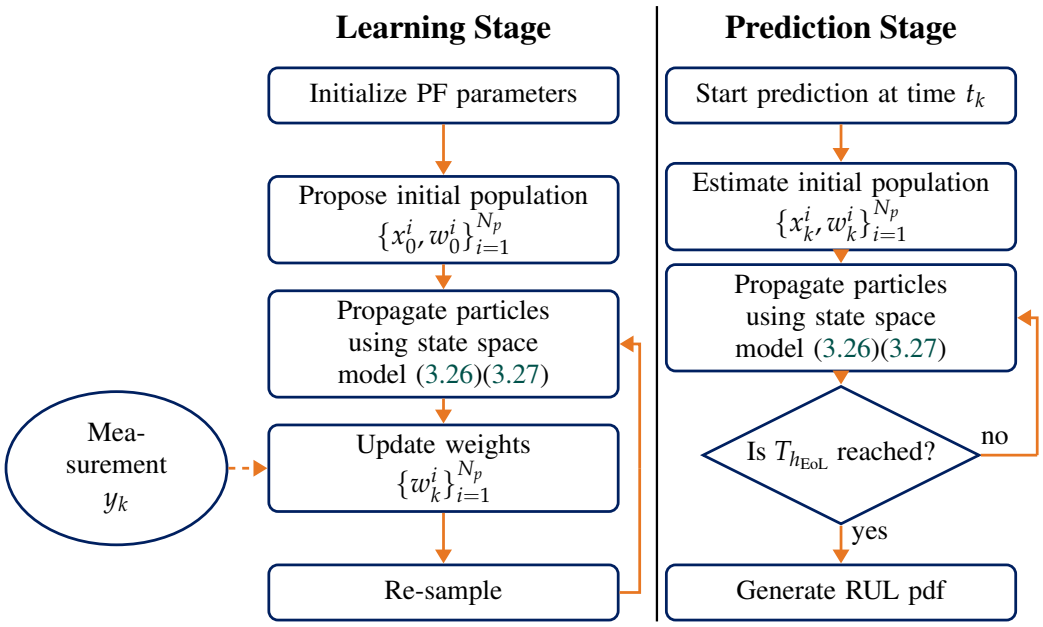

FIGURE 3.2: Application of Particle Filtering for prognostics purpose

In the previous section, we described how to employ SIR Particle Filtering to estimate the system state x_k and its unknown parameters based on the collected observation y_k at time step k (instant t_k in time units, from now on). This means that, if we apply it to the degradation model of the studied system, we can learn, through its HI observations, the underlying system failure evolution. The left column of Fig.3.2 refers to this course of action: the learning stage.

Then, using this knowledge, we are able to propagate the posterior PDF to forecast the degradation state in the future. The simplest way to perform prognostics in this framework is, at a given moment in time t_k, to reiterate the prediction step of Alg.3.1 by propagating the particles $\{x_k^i\}_{i=1}^{N_p}$ until x_k^i (or also y_k^i, obtained through eq. (3.27)) reaches the failure

threshold $T_{h_{\text{EoL}}}$ at the End-of-Life (EoL) time t^i_{EoL}. Then, the Remaining Useful Life PDF is obtained from the distribution of those failing times t^i_{EoL} with respect to t_k, i.e. $p(t^i_{\text{EoL}} - t_k)$. This is summarized in the right column of Fig. 3.2: the prediction stage.

Part II

Case Studies and Applications

Chapter 4

Diagnostics Applications

In this chapter, we present the case studies and applications in which the MoS approach is employed to perform condition monitoring and fault detection and isolation. Firstly, the idea is to show how MoS health indicators compare to the statistical ones for monitoring in a gearbox lubrication scenario. Then, the new presented modelling approach for trajectory-driven mechanisms is tested for fault detection on a laboratory setup. Finally, MoSs are compared to statistical quantities as features for machine learning when employed for fault detection and isolation.

4.1 Case Study: Gearbox Lubrication Monitoring

In this application study, the MoS approach is employed to monitor the lubricant level in a large industrial worm gear motor “Rossi MR V 160 UO2A– 132S 4 230.400 B5/28 mounting position B8 $50Hz$-Motor". The system, which is equipped with an asynchronous motor connected to a gearbox, is drawn in 16.5ℓ mineral oil and is subjected to leakages. The loss of lubricant has been usually monitored by the operator of the machinery depending mostly on the sound the gearbox makes during operations and some basic indicators on the controller HMI. On the other hand, the component is not prepared to host traditional sensors to monitor the oil level and its substitution with a new model would cost much more than applying our proposed solution doing the monitoring job.

We exploit machinery measured signals, e.g., vibration readings from accelerometers and inverter driving currents. We model those signals assuming they are AutoRegressive (AR) processes and compute their estimate through algorithm 2.3. MoS monitoring indicators are here studied in comparison to signal statistical indexes to evaluate their capabilities in addressing conditions and their evolution. Both those indicators are shown and described in section 3.1.

4.1.1 Hardware Setup

Model-of-Signals based condition monitoring focuses on the processing of sensor information, which in most cases are vibration, current, and temperature signals. The acquisition chain is composed of equipment able to convert the physical quantities to be measured in a suitable form to be read by the processing unit: Starting from the actual sensing mechanism, passing through the analog-to-digital conversion, and ending to the storage of the samples, either for online usage or long-term keeping. In this fashion, the choice of which quantities to measure, where to measure them, and with what sensors is very dependent on the design of the machinery itself, on the processing unit that will be used to elaborate the data and, obviously, on the costs of the different solutions available to perform the acquisition task.

The equipment used in this project consists of:

- Two PCB 356A15, i.e., $50g$-triaxial piezoelectric accelerometers with a measuring bandwidth of 2-$5000Hz$ and output signal of $\pm 5V$, that are acquired with three EL3632

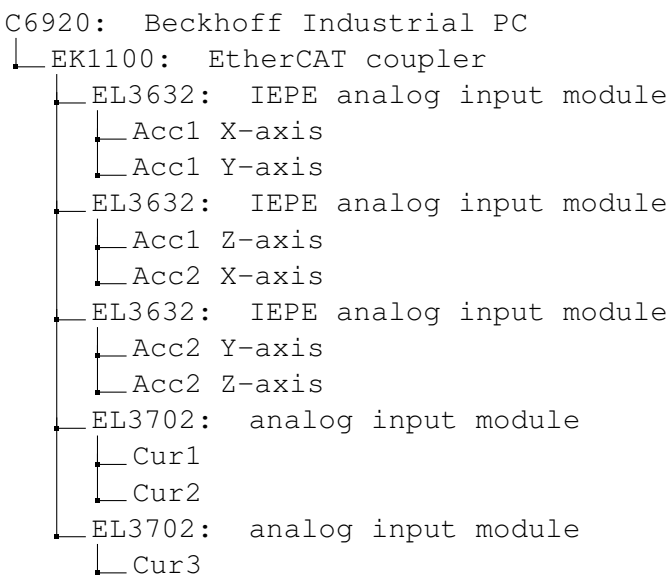

FIGURE 4.1: Components connection scheme.

modules, which supports a maximum sampling frequency of $50kHz$. The accelerometers are labeled `Acc1` and `Acc2`.

- Three LEM LF-210S, i.e., $200A$ Hall effect transducers with a frequency bandwidth of $100kHz$ and the output signal of $\pm 100mA$, that are conditioned with 50Ω-resistance and acquired with two EL3702 modules, which supports a maximum sampling frequency of $100kHz$. The current sensors are labeled `Cur1`, `Cur2` and `Cur3`.

The components connection scheme is depicted in figure 5.1.

4.1.2 Data Acquisition

The system depicted in section 4.1.1 performs the data acquisition task preparing them to be processed. The PLC and Fieldbus main cycle time is set to $1ms$, while the module sampling frequencies are set to oversample at $5kHz$. In this way, at each cycle time, the PLC handles 5 samples for each acquired signal. The acquisition task on the PLC collects the data into arrays with 15000 entries resulting in $3s$-data streams. At this point, it possible to log the acquired data or feed them to the data processing task or, if required, do both: Logging and processing. During those operations, data collection continues on a second buffer of the same size.

4.1.3 Data Processing

The data processing phase onboard the machine as defined when speaking for condition monitoring requires a pre-processing phase. The raw data streams from the sensors were acquired and logged as described in subsection 4.1.2 and used to perform the model order analysis through FPE and MDL criteria combined with the whiteness test on residual, as in section 2.7. The logging has been performed with the machinery in perfect nominal working conditions (oil level of 16.5ℓ), validated, and certified by the machine operator. The study, then, has been done in a Matlab environment and resulted in different model orders for each of the available signals. However, due to the constraints given by the implementation on a real-time platform, they were reduced to $n = 20$ for the current signals and $n = 36$ for the accelerometer signals and chosen to neither overfit nor underfit the data.

TABLE 4.1: Set of monitoring trials performed on the machinery with fixed production speed at 120 $products/min$

Trial	Oil leak	CM Duration
Nominal	0ℓ	$90min$
Delubrication 1	-0.2ℓ	$23min$
Delubrication 2	-0.4ℓ	$15min$
Delubrication 3	-0.6ℓ	$8min$
Delubrication 4	-0.8ℓ	$13min$
Delubrication 5	-1.0ℓ	$12min$
Delubrication 6	-1.2ℓ	$10min$
Delubrication 7	-1.4ℓ	$12min$
Delubrication 8	-1.6ℓ	$8min$
Delubrication 9	-2.0ℓ	$12min$

At this point, a second acquisition has been performed on the PLC in the same condition validated by the operator, this time with the model order parameter set for each signal. The computations of θ_{ref} were carried out by averaging a set of 10 models obtained by elaborating with the RLS algorithm an equal number of data arrays. The nominal reference model of every signal has been defined and retained on the system to perform the model distance calculations.

In the end, a set of subsequent trials has been performed starting from the monitoring of the machinery in nominal working conditions and, then, gradually removing discrete volumes of oil from the system until it decreased by 2ℓ, as in Table 4.1. In this way, with the help of the operator, we have been simulating the leaking process. The monitoring program has been run together with the machinery during the trials to acquire and process the sensors' data. At this stage, statistical indexes were computed together with the models and their distance indicators. Finally, those quantities were logged for each trial to analyse the monitoring results.

4.1.4 Monitoring Results

Given the whole picture about the working principles of the procedure, it is now possible to analyse the data collected. Firstly we compare the monitoring performance of the statistical indexes with respect to model distance indicators and finally we focus on the latter for the whole set of acquired sensors. Before starting, due to the lack of temperature sensors the "Nominal" and "Delubrication" trials were run when the warm-up phase of the oil was not ended yet, so their initial half is excluded in the analysis.

Regarding Figure 4.2 showing the monitoring results regarding the Y-axis of accelerometer `Acc1`, the statistical indexes RMS, SRA, and KF have similar behavior, showing a non-monotonic trend starting from "Delubrication 6", i.e, with an oil leak of -1.2ℓ. This does not permit a consistent level monitoring since "Delubrication 6" and "Delubrication 8-9" assume comparable values. While, PPV, CF, IF and MF experience high variations from their mean value, which may result in the wrong diagnosis at any leakage level.

On the other hand, those previously depicted behaviors do not find a correspondence on the current side, as shown for `Cur1` in Figure 4.3. In fact, CF, IF and MF lose completely

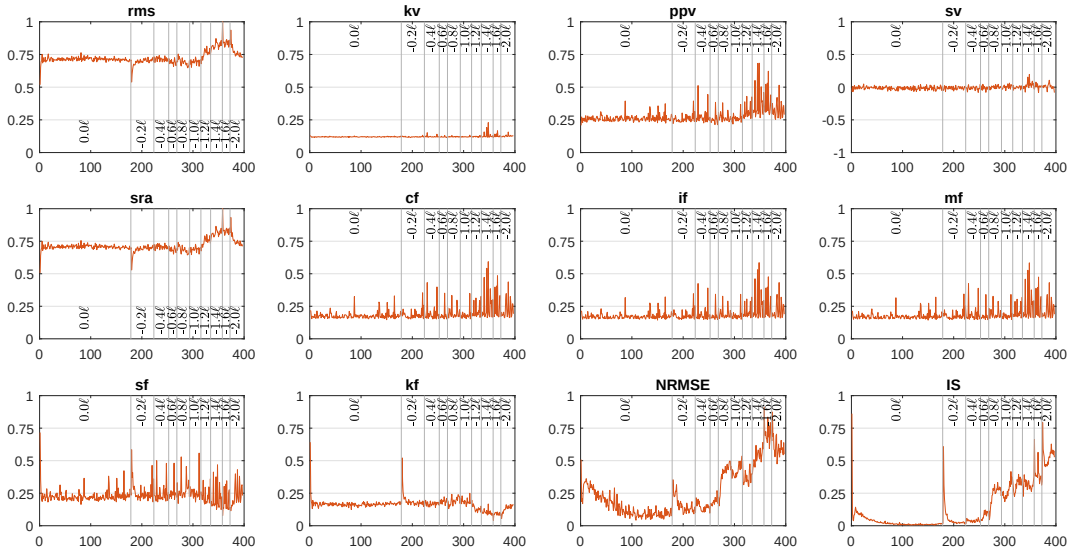

FIGURE 4.2: Monitoring indexes and distance indicators comparison: `Acc1Y` (quantities are normalized and the time axis is shrunk for the sake of clarity)

their informative content, despite their very low standard deviation. While RMS, PPV, SRA, and KF do not show a monotonic trend, they experience a jump in correspondence of "Delubrication 6" and then they either return to the nominal value or hold the acquired one. Moreover, in both current and vibration statistical quantities, the percentage variation that it is possible to observe is very low concerning their baseline value.

In general, as already stated in section 3.1, statistical indexes require further analysis to perform a suitable feature extraction for diagnosis, and this case confirms this aspect. In contrast, model distances already include such behaviour: In fact, by looking at how NRMSE and IS behave, in both current and vibration results, it is possible to see how they experience a (almost) monotonic increase correlated with the leakage level and spanning along a greater range of values. So, the variation with respect to their reference value may be straightly used for monitoring. Notice that currents are not experiencing any misbehaviour also during the warm up transient.

At this point, given the fact that distance indicators are best-suited for the monitoring task, we analyse their performance concerning each of the measured signals. Given the previous assumptions about the ending of the warm-up phase, the results indicate that accelerometers can detect oil leakages earlier than current sensors. However not all the measurements show a monotonic trend, X and Z axes show this behavior, despite being able to detect in any case a leakage of more than 0.8ℓ. Therefore, the best choice is to use accelerometer Y axes for monitoring. For what concerns currents, they detect leakages later than accelerometers, however they experience a more robust trend with respect to disturbances. Due to their similar behavior, it is possible to use just one of them for monitoring.

4.2 Case Study: Electric Cam Fault Detection

Here, we present a different application of the MoS that aims at exploiting how electric cams are executed on PLCs to monitor electric motor driven mechanism conditions employing applied torque as a source of information. The use of the torque (i.e current) measurement for monitoring electric motor components (e.g. cage, windings, resistance, bearing, and shaft faults) is long-established (Nandi et al., 2005) and typically makes use of model-based,

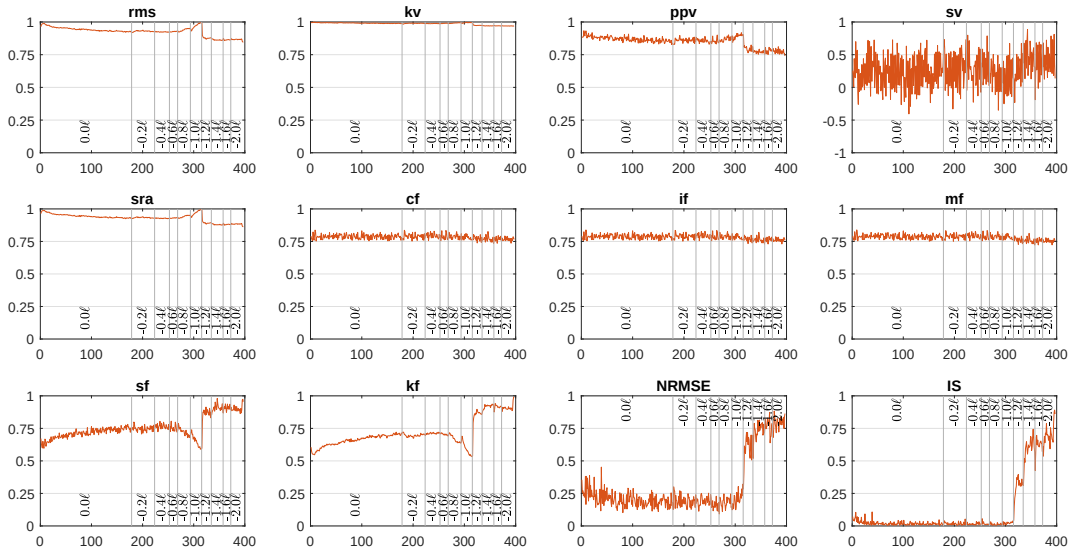

FIGURE 4.3: Monitoring indexes and distance indicators comparison: `Cur1` (quantities are normalized and the time axis is shrunk for the sake of clarity)

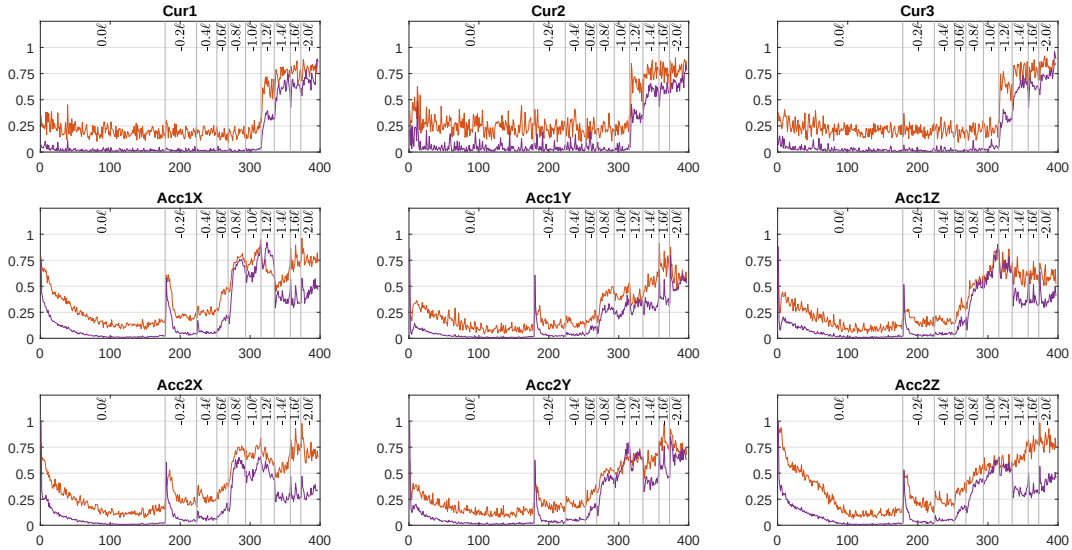

FIGURE 4.4: Distance indicators for the set of sensors, $NRMSE$ (orange) and D_{I-S} (quantities are normalized and the time axis is shrunk for the sake of clarity)

frequency-based, and data-driven techniques with high-end servo drives starting to include them within their controllers. However, due to the unknown task, they will have to accomplish in their final application, the above-mentioned techniques are only related to the motor internal health state and not to the mechanism attached to it, which is also subject to failure. The condition monitoring technique we propose aims at providing the manufacturer with a PLC practicable solution for drive-mechanism fault detection.

The main idea is the following: the majority of electric cam motion tasks for servo drives are implemented as piece-wise polynomial trajectories with order lower or equal to 7. Therefore, in the case of linear mechanisms the ideal torque demanded by their motion is linked to the second derivative of such curves. In real applications, however, another component is present alongside the ideal torque: smaller with respect to the latter, but necessary to achieve the desired motion. Our conjecture on that additional contribution is that it contains information about the mechanism health condition and it can be modelled by a set of Auto-Regressive (AR) models. Its analysis requires the ideal contribution to be removed to prevent it from masking changes within the useful one (in this domain, the ideal torque is the "noise" perturbing the informative signal). A simple subtraction of the ideal torque could be arranged in this respect, however, it would depend on the given cam trajectory and the equivalent inertia of the mechanism. In this work, we propose to compute the difference of suitable order of the torque measurement (which is linked to the order of the trajectory polynomial profile minus 2, therefore 5 at worst) to get rid of its ideal contribution without any cam and mechanism detailed knowledge. Then, the useful part of the signal is modelled as an AR process, which will be proven to become an Auto-Regressive Moving-Average (ARMA) with a particular structure when differencing.

The theoretical process undertaken to adapt the MoS framework to this typology of signals has been discussed in detail in section 2.4.

The experimental setup utilized is presented in Fig. 3 and is composed by, from right to left: electrical motor, rigid joint, shaft, and flywheel with two half-moon shaped weights, the cabled encoder has seen in the figure is not used in this experimental analysis and is not taken into account. The inertia of the system can be divided into two parts: one is fixed, with a value of $J_{fix} = 0.0015[kg/m^2]$, and one is variable $J_w = J_{w_1} + J_{w_2}$ depending on the two weights, J_{w_1} and J_{w_2}, attached. The mechanism is driven by B&R equipment: the PLC is the `Automation PC 910` connected to an `ACOPOS P3` servo drive controlling a `8LSA36 DB030S000-3` brushless motor. The electric cam utilized is the same as in example (2.57) performed using a virtual master running at constant speed $V_p = 1080°/s$. The synchronised motion task time was chosen to be $T_s = 0.0008s$ since the system only allows time-steps of $0.0004s$ or multiples and that was the recommended setting. Therefore, the measurement of the torque signal has the same resolution and is collected utilizing the tracing system provided by B&R IDE, Automation Studio, with a sampling frequency of $1250Hz$ and can be directly saved into `.mat` format. To test the proposed monitoring approach we sampled the slave drive torque during the synchronised motion of the system with both symmetric and asymmetric (i.e unbalanced) load. The former being the healthy reference operating point and the latter being the faulty one achieved by modifying one of the two half-moon-shaped weights with a slightly thicker and a slightly thinner one. In addition, in the symmetrical case, one of the two weights has been loosened by sightly unfastening the bolts that keep it in place to simulate a fault with an increased degree of severity. Those unbalanced loads should generate changes in the informative part of the torque measurement which in turn should be captured by the models. The four tested configurations are depicted below:

Config. (1) Symmetric load:

$$J_{w_1} = 7.1305 \cdot 10^{-4},\ J_{w_2} = 7.1305 \cdot 10^{-4}[kg/m^2] \tag{4.1}$$

Config. (2) Asymmetric increased load:

$$J_{w_1} = 7.1305 \cdot 10^{-4},\ J_{w_2} = 7.5030 \cdot 10^{-4} [kg/m^2] \tag{4.2}$$

Config. (3) Asymmetric decreased load:

$$J_{w_1} = 7.1305 \cdot 10^{-4},\ J_{w_2} = 6.1725 \cdot 10^{-4} [kg/m^2] \tag{4.3}$$

Config. (4) Loose Symmetric load: same as Config. (1) but with loosened bolts in one of the weights.

Various measurements of the slave torque were collected during operations in all configurations. Then, they were processed by simulating a PLC implementation via Matlab. In particular, the main steps of the finite state machine involved in the processing are the following:

1. Sample the slave torque in a buffer of $N = 10000$ elements;
2. Compute the signal model with algorithm 2.6;
3. Compute the $NRMSE$ distance index, as in (2.77), with respect to the reference model.

The reference model is computed as the mean value of the first 10 models obtained during known healthy operating conditions by exploiting steps (1) and (2). Algorithm 2.5 is executed with the following hyperparameters: $n = 2$ as the AR model order, obtained with AIC criterion (Akaike, 1974), as depicted in section 2.7, plus $q = 6$ equations in its overdetermined part and $N = 10000$ as number of samples. Each model is the outcome of the algorithm after N samples are processed. The buffer and process architecture applies to any PLC since it allows the storage of N data samples to be coded within the main priority task, in this case with a sampling time of $Ts = 0.0008s$, and the implementation of their processing in a secondary task of lower priority, without affecting significantly the system memory and the control program computational load. This keeps the condition monitoring task on-line, still able to check machine health state for degrading faults. For instance, a model is obtained every few seconds while mechanism degradation due to friction or wearing or heat typically takes minutes to hours to even days. In this fashion, the reference AR model computed while the system is in healthy working conditions is then compared with the models obtained while operating in the previously depicted load configurations.

The data collected from the depicted processing are here shown and analysed. The torque signal measured during healthy operations and its computed differences is shown in the top row of Fig.4.5. Due to the higher power of the AR part with respect to the ideal torque demand, the signals have approximately constant mean already at $\tau^{(2)}(t)$. This results in acceptable fault detection from the 2^{nd} difference if we use the $NRMSE$ indicator with a threshold of $T_h = 0.1$, as we can see in the last row of Fig.4.5. However, in $\tau^{(2-3)}(t)$ case, a false healthy condition may appear for Config. (3), despite $\tau^{(2)}(t)$ is the best in pointing out Config. (4). This can be also deduced within the model plots, in the middle row of Fig.4.5 show how close they are to the healthy ones. The best indicator in this respect is obtained in the case of $\tau^{(4)}(t)$ where faults have separate levels revealing satisfactory fault detection when T_h is applied. In particular, it turns out to be the most suitable to perform the monitoring task validating our proposition.

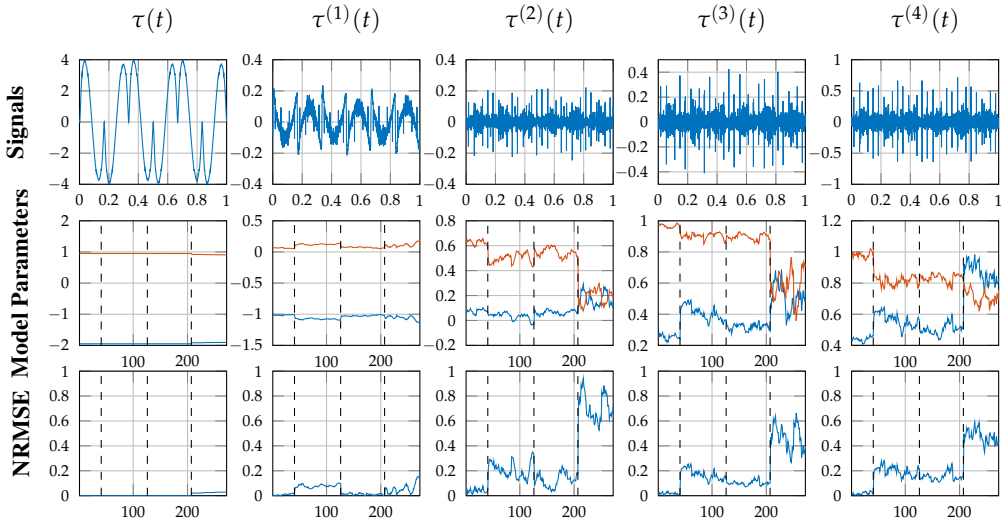

FIGURE 4.5: [Top] Signals: the healthy signal of 3 cam periods and its computed differences are shown. [Middle] Models: the collected models for the four configurations are shown. [Bottom] NRMSE: the collected indexes are shown, a_1 in blue and a_2 in orange. The dashed lines divide one configuration from the other. They are, from left to right, Config. (1) to (4) respectively.

4.3 Case Study: Bearing Fault Detection and Isolation

Bearings are one of the most common components in automatic machines. Diagnosis and prognosis of their working condition is crucial for minimization of downtime and maintenance costs. Different approaches may be adopted to either solve or mitigate the problem of identifying incipient faults during machinery operations. In this study, we propose a simple and efficient yet effective method to solve this problem by exploiting the edge-computing capabilities of PLCs. Accelerometer signals are modeled as AutoRegressive (AR) processes whose coefficients are used as features for machine learning, based on logistic regression algorithm (LR), to perform Fault Detection and Isolation (FDI). Estimation and prediction are both implementable on-board the PLC, while machine learning can be carried out remotely, from a cloud computing perspective. The exploitation of AR modelling gives a simple and inherent methodology for feature selection. We apply the procedure to the Case Western Reserve University database (CWRU, 2014), a widely known and used benchmark, to highlight its performance with respect to similar fault recognition techniques. In particular, the comparison here is done to the statistical indexes depicted in section 3.1, since they are compatible with the execution on the machine controller and are used as features for machine learning in many monitoring solutions.

4.3.1 The procedure

Our procedure is divided into three main tasks following the CBM main steps guidelines. Firstly, the measurements are collected and filtered by a suitable sensor, which in this case is a piezoelectric accelerometer measuring vibratory signals. Secondly, measured samples are treated as the elements of vector $\varphi(t)$ and fed to the RLS algorithm running on-board the machine to estimate their related AR model $\hat{\theta}$, see algorithm 2.2. Finally, we carry out FDI over those collected MOS by means of the LR algorithm in a one-vs-all fashion. We apply the depicted procedure upon the CWRU dataset to let our outcomes being compared

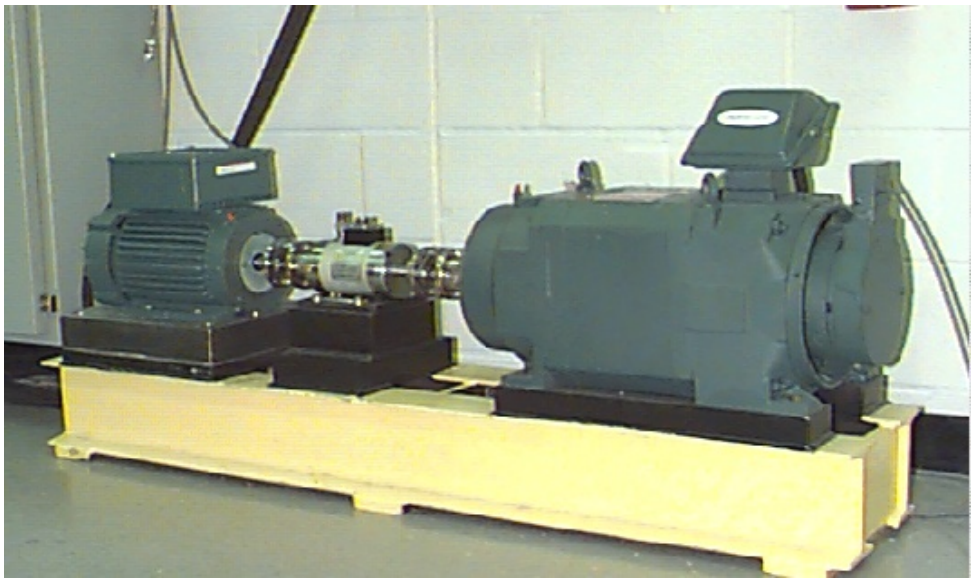

FIGURE 4.6: CWRU Database test bench.

to others and to let the user being able to reproduce our results. Because of this, we simulate the whole procedure in a Matlab environment. However, this does not invalidate our PLC implementation claim. We already showed in (Barbieri et al., 2018) that AR estimation is possible. While logistic regression prediction is nothing but a matrix product composed of a sigmoid function and a maximum function, already available on such computers. Regarding the training of the classifier, we consider it as done remotely and being sent to the PLC. It is possible to exploit the first round of MoS computed on-board the machinery for different working conditions, both healthy and faulty ones. Those data may be sent remotely to computers capable of handling the storage of features as well as the complex computations required for predictors training since PLCs are not designed to perform such functions. Moreover, this architecture allows us to combine data coming from more than one machine (provided that they are of the same typology).

4.3.2 CWRU Dataset

The Case Western Reserve University Bearing Data Center (CWRU, 2014) is often used as a benchmark for testing new procedures and techniques in bearing fault diagnosis. The test rig in Figure 4.6 consists in a 2 hp motor (left), a torque transducer/encoder (center), a dynamometer (right), and control electronics (not shown). The test bearings support the motor shaft and their vibrations are measured through piezoelectric accelerometers. Single point faults were introduced to the test bearings using Electro-Discharge Machining (EDM) with fault diameters of 7 mils, 14 mils, 21 mils, 28 mils, and 40 mils (1 mil $=$ 0.001 inches $=$ 0.0001778 mm). Faults are located in the Inner Ring (IR) and the Ball (B) rolling element and in three different points on the Outer Ring (OR). The bearings under test were two SKF ones: 6205-2RS JEM for the Drive End (DE) and 6203-2RS JEM for the Fan End (FE).

This work focuses on the capabilities of our approach to detect and distinguish incipient faults in the worst scenario allowed by the dataset. For instance, Figure 4.7 shows two samples of different faulty signals that are difficult to isolate from each other. In this fashion, we take into account the measurements regarding 7 mils faults, since it is possible to assume their working condition has suffered from degradation but it is anyway admissible for operations, and relate them to healthy ones. We employ for computations the data collected by the

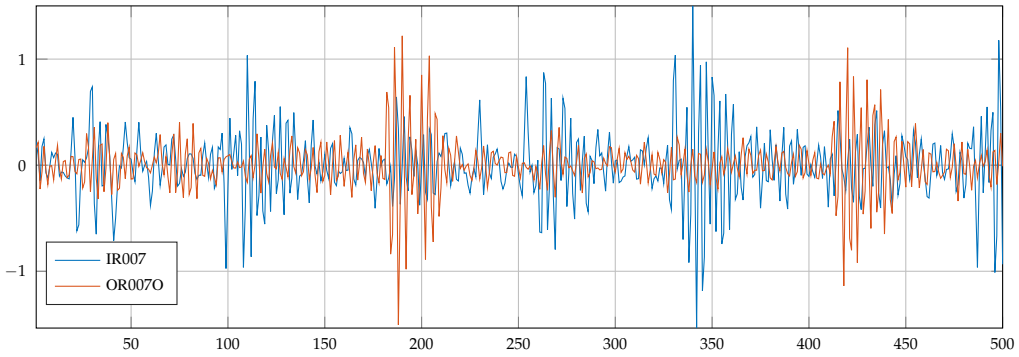

FIGURE 4.7: Vibratory signals in the presence of faults: inner ring (IR007) and outer ring positioned at the opposite side of the load point (OR007O).

DE accelerometer with sampling at 12 kHz since they are available for each condition. The bearing working state we refer are:

- Healthy: the bearing in its nominal working condition;
- B007: bearing with a ball fault of 7 mils (0.1778 mm);
- IR007: bearing with an inner ring fault of 7 mils (0.1778 mm);
- OR007C: bearing with an outer ring fault of 7 mils (0.1778 mm) positioned right under the load point;
- OR007O: bearing with an outer ring fault of 7 mils (0.1778 mm) positioned on the opposite side of the load point;
- OR007P: bearing with an outer ring fault of 7 mils (0.1778 mm) positioned right under the measuring point and in between the previous ones.

Data Processing

Starting from the aforementioned signals, we begin with the analysis of the healthy condition and determine its model order. The choice follows the parsimony principle. The combination of the FPE and MDL criteria and the F-test between consecutive model orders resulted in $n = 7$. The AR model to be estimated is thus described by the parameter vector $\hat{\theta} = [\, a_1 \, a_2 \, \cdots \, a_7 \,]^T$.

The implementation requires the order to be defined off-line due to static memory allocation requirements in real-time and constrains every condition to be addressed by an equal number of parameters. This choice allows the estimation algorithm to perform the identification task on the same set of AR coefficients independently from the condition. Moreover, the LR predictor works with a homogeneous set of feature vectors.

By following our PLC implementation objective, we carry out signals segmentation in groups of 10000 samples, since the memory available for the task on the machine cannot contain the whole set of measurements. Then, on each of the signal windows the RLS algorithm is performed and the computed $\hat{\theta}(t)$ is collected. Notice that, depending on both PLC computational power and available memory, sample windows may vary from being overlapped to being contiguous. This is due to the use of circular buffers to store accelerometer measurements. In Figure 4.8 the models computed during four windows are depicted: the algorithm converges, after a transient due to the new initialization, to almost the same model at each instance, showing consistency. The initialization of each RLS batch is done by means of the previously computed ones, the first iteration may be done by a retained stable $\hat{\theta}(0)$ and

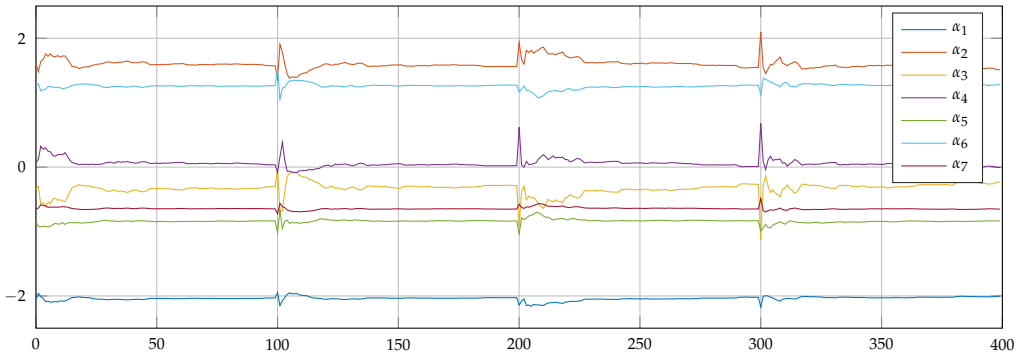

FIGURE 4.8: Healthy model parameters with no load: evolution during computations.

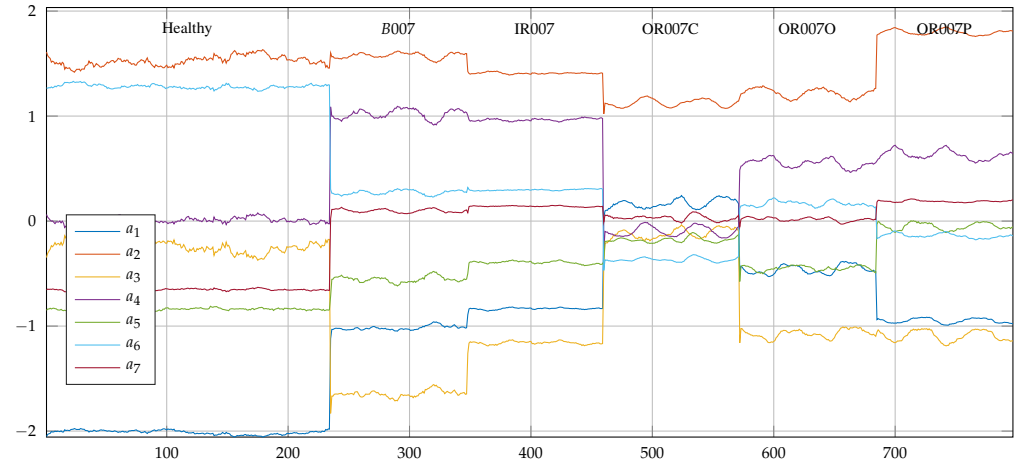

FIGURE 4.9: Model parameters with no load: collection during the various conditions.

suitable $P(0)$, see Subsection II-B. The identification procedure is then applied to the faulty

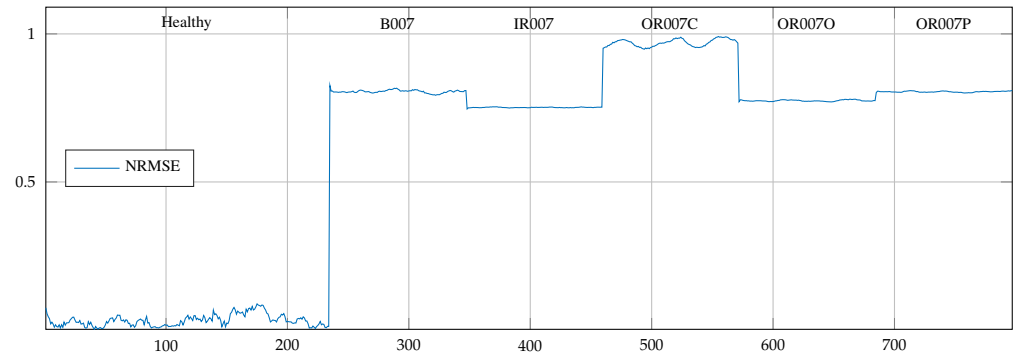

FIGURE 4.10: NRMSE with respect to θ_{Healty}

signals set as well as the healthy set by simulating PLC computations upon overlapped sample windows. This is done to collect a suitable number of models to perform the second task of our approach. Each identified AR vector is labeled with its corresponding condition. For instance, healthy models are referred as $\theta_{Healthy}$, while a ball fault one is referred as θ_{B007}. Figure 4.9 shows the gathered set of parameters in each working state.

TABLE 4.2: Prediction accuracy of the different feature sets

Training	Ψ_7	Ψ_8	Ψ_9	Θ
30%	71.8-100%	85.8-100%	85.8-100%	100%
50%	72.8-100%	85.9-100%	85.9-100%	100%
70%	71.7-100%	71.7-100%	85.8-100%	100%
90%	71.8-100%	71.8-100%	85.9-100%	100%

At this point, it is possible to evaluate how model distances may be used to address different faults. In this case, we define a reference $\theta_{Healthy}$ to which each $\hat{\theta}$ may be compared to, by means of the NRMSE. The nominal working condition $\theta_{Healthy}$, is defined as the mean value of the first 50 collected models from the healthy signal:

$$\theta_{Healthy} = [-2.00 \;\; 1.48 \;\; -0.19 \;\; 0.02 \;\; -0.85 \;\; 1.30 \;\; -0.66]^T .$$

However, as we show in Figure 5.11, it is not useful for FDI purpose, as different faults are indicated by the same level: B007, IR007, OR007O and OR007P distances from the healthy reference are very similar to each other while OR007C may be isolated. On the other hand, it may be used for detection as an unhealthy working indicator, asking for human operations to address the fault, which fails in being an automatic FDI.

FDI with one-vs-all LR

Once the identified models for each condition are available it is possible to feed them to the one-vs-all LR. Firstly, for training purposes and secondly for prediction. In this fashion, the collected $\hat{\theta}$s become the elements of the feature set Θ, being the analogous of x and $\mathcal{X}$ from section 3.2. The set of labels $\mathcal{Y}$ contains the numbers from 1 to 6, being: 1 = Healthy, 2 = B007, 3 = IR007, 4 = OR007C, 5 = OR007O and 6 = OR007P, linking each model to its condition. Together with models, the statistical features depicted in Table 3.1 were computed and collected for the same sample windows: we will denote them by the symbol ψ. Each of them is linked to the same labels depicted before. They are used as features for the machine learning algorithm, too. So, two feature sets are now available: one made of AR models (Θ) and one made of statistical quantities (Ψ). Their prediction accuracy with the logistic regression algorithm is then compared with different training set dimensions. Notice that our second set has more than n elements in a feature vector: we combine them in every possible way: from $\binom{10}{7}$ to $\binom{10}{9}$ to get the prediction results. Firstly, we train our logistic regression to get the two predictors: Γ_{AR} for the models and Γ_{Stat} for the statistical features. Then, We divide each of the feature sets labeled subgroups in training and testing with different percentages: 30-70%, 50-50%, 70-30%, and 90-10%. Results obtained by the computed predictors on the training sets are available in table 4.2. In particular, the outcomes shown for the statistical feature set Ψ are divided addressing the number of features combined to generate the predictors. For instance, Ψ_7 relates to all the possible seven-feature-out-of-ten classifiers. The accuracy in prediction obtained with Γ_{AR} on the Θ set is, no matter the training, always 100%, which is the proof that our inherent approach for features selection is also fairly robust with respect to reduction of the training set. As a matter of fact, if such a set is large enough for AR order selection, no more data are necessary. On the contrary, to get the 100% goal with the $\Psi_{7\text{-}9}$ combinations an additional pooling procedure is needed.

The outcomes obtained by applying our approach are very promising. The use of AR models to generate features for the logistic regression predictor can perform FDI upon the

CWRU dataset, resulting in perfect accuracy. This happens despite having reduced the number of training samples, proving its robustness in this particular application. The use of statistical quantities instead of AR coefficients turns out to be less accurate. The same performances may be obtained after carrying out a feature pooling at the cost of adding another machine learning step in the procedure. Our method relies on system identification principles to inherently generate a compelling feature set containing enough information to perform accurate condition labeling.

Chapter 5

Prognostics Applications

In this chapter, we depict the applications and case studies that involve actual PHM solutions, trying to track or estimate the remaining useful life of components. On the practical side, the two procedures shown in the following take into account more and more practitioners and the way MoS is implemented and deployed to perform PHM. In these applications, the monitoring technique is combined with two powerful tools from machine learning, Support Vector Machines (section 3.2.2) and Particle Filtering (section 3.2.3). The former is applied on an industrial prototype of a paper feeding machine to track its main paper picking mechanism degradation and the latter is employed for RUL prediction upon a couple of bearing benchmark run-to-failure datasets. Besides, this work is the outcome of a collaboration with Prof. Kamal Medjaher of the Production Engineering Laboratory (Laboratoire Génie de Production or LGP) of ENIT Tarbes (FR).

5.1 Case Study: Paper Feeding Mechanism degradation tracking

The machinery under test in this study is an industrial paper feeder. It is a working group within a production line that involves paper sheet insertion in several packaging typologies. The device is subject to heavy-duty cycle operations, up to 30 000 cycles per hour and we kept this maximum value constant during our testing. One electric motor drives the whole mechanism. A system of gears, cams, belts, and pulleys transmits the wanted synchronized motion to the end effectors, i.e., a combination of pliers and suction caps extracting paper sheets from a vertical stack. Even though the feeder is designed to work in high-performance conditions, its parts suffer from wearing over time. It causes the increase in play between its moving elements resulting in paper feeding quality degradation. We monitor such production deterioration by measuring device frame vibrations using two accelerometers, with an Industrial PC handling their acquisition and processing. Accelerometers are installed on the crank of the slider-crank mechanism that drives the pliers, oriented along the connecting rod, and on the shaft that releases the single paper sheets, oriented in the same direction of the motion of the vertical stack support, respectively. We cannot provide pictures of the mechanism because of the confidential nature of the project. Nevertheless, the firm we collaborate with has allowed us to share the obtained condition monitoring results.

In this study, the feeder mechanism is run to failure with its parts backlash measured at given time intervals. Accelerometer signals are modelled as AutoRegressive processes whose coefficients are then considered as features to feed to machine learning algorithms, which are employed to perform severity evaluation of the ongoing degradation. Estimation and prediction are both implementable on-board the controller, while the learning task can be carried out remotely, in a cloud computing perspective. The exploitation of AutoRegressive modelling gives a simple and inherent methodology for feature selection, serving as a foundation of the machine learning stage. Algorithm 2.3 is the tool of choice in this case to perform MoS calculations. Then, we make use of a Support Vector Machine algorithm (as in section 3.2.2) to analyze how obtained models represent the various levels of backlash in the

Cycles	Backlash [mm]
1 268 642	0.650
2 081 215	0.700
2 336 280	0.925
3 044 592	1.100
3 754 981	1.200
4 463 972	1.375
5 159 990	2.075
5 940 476	2.200
7 064 995	2.625
8 264 164	3.150
9 213 769	3.500
10 153 300	3.900

TABLE 5.1: Pliers backlash measurements with corresponding number of machine cycles reached at time of recording.

device and develop a suitable predictor of the degradation severity. In the end, the results of the application of the methodology to the case study are shown.

5.1.1 Data Acquisition

The equipment used in this project reflects the considerations we introduced about the use of industrial PCs as edge-computing units. It consists of:

- C6920-0030: Beckhoff Industrial PC, with CPU Core2 Duo 2.53 GHz and RAM of 1 GB.
- Two PCB 353B03, i.e., monoaxial, piezoelectric, 500 g accelerometers with a measuring bandwidth of 1-7000 Hz and an output signal of ± 5 V, that are acquired with one EL3632 module, which supports a maximum sampling frequency of 50 kHz. The accelerometers are labeled Acc1 and Acc2.

Components connection scheme is depicted in Figure 5.1. The PLC is responsible for the collection and conditioning of sensor measurements. Controller and Fieldbus main cycle times are synchronized and set to 1 ms, while the module sampling frequencies are configured to oversample at 5 kHz. Hence, at each cycle time, the controller receives 5 samples from each acquired sensor. The acquisition program groups accelerometer data into arrays with 18 420 entries resulting in 30 operating cycles.

We carried out this study by performing a run-to-failure test of the machine, stopping operations at given time instants to record the backlash level between parts, as shown in Table

```
C6920-0030:  Beckhoff Industrial PC
└─EK1100:  EtherCAT coupler
  └─EL3632:  IEPE analog input module
    ├─353B03:  Piezoelectric accelerometer 1
    └─353B03:  Piezoelectric accelerometer 2
```

FIGURE 5.1: Components connection scheme.

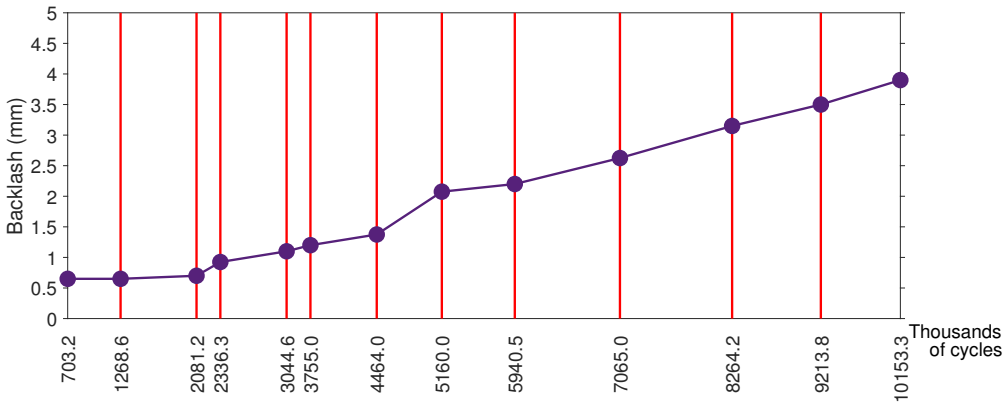

FIGURE 5.2: Plot of pliers backlash against the number of machine cycles reached at time of recording. The red lines indicate the time in cycles at which the measurement was taken.

5.1 and Figure 5.2. Despite the higher wearing registered between 4 and 5 million cycles, the degradation seems to increase constantly as time passes. The life span of such a device is typically rated for about 10 000 000 total cycles, according to the vendor. In our testing, according to the feeder technician, we reached the highest possible level of play before the performance degradation was unsustainable, feeding a random number of sheets of paper and not one at a time. In practice, the definition of the non-return level of play depends on the customer and, in particular, on the machine mounted downstream the feeding group and the feeding quality and precision it requires.

5.1.2 Data processing

The data processing stage makes use of the PLC as an edge-computing unit and its supervisor as a remote-computing one. The controller is responsible for the first information refinement, using the RLS algorithm to estimate the models of the buffered signals. The main hyperparameter to set is the model order. In this application, $n = 20$ is the result of the application of MDL and FPE on a sample signal measured during nominal working conditions with PLC available resources taken into account. Alongside this definition, the controller estimates the IS distance reference model. While performing the test, the industrial PC logs the estimated Acc1 and Acc2 models, together with the corresponding IS distance from the respective references and sends those values to the supervising computer using the MQTT communication protocol. Besides, the current backlash measure is recorded, with the operator doing this operation via Human-Machine Interface (HMI) every time the machine is stopped.

Once the test and data collection are over, the supervising PC, running MATLAB in this case, performs the machine learning task, providing the results that we will analyze in the next section. The adopted SVM algorithm exploits a linear kernel and penalises classification errors with a regularization parameter of 1. SVM labels data according to the recorded backlashes, which are however available only in correspondence of machine stops. Therefore, when the operator inserts a backlash value in the HMI, it labels all the data processed between the previous and the actual machine stop. It results in a 12-classes data partition. The algorithm partitions data with the same proportions within each class, by picking randomly 70 percent of data for the training phase, 0 percent for validation, and 30 percent for the test stage. Due to the high cost of such tests, the firm allowed us to record only one run to failure of the machine, so we had to train and test using this unique run only.

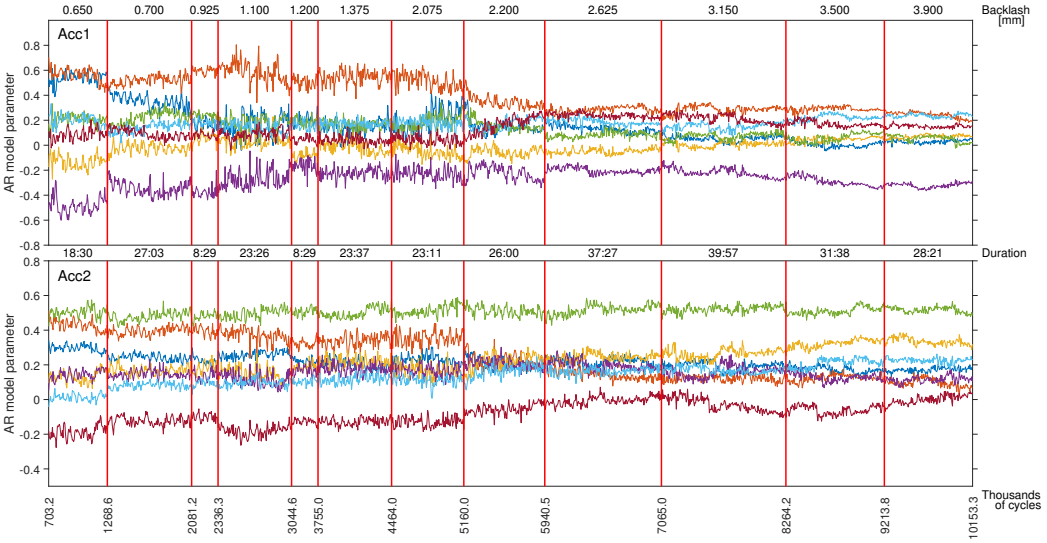

FIGURE 5.3: Model parameters evolution in time. Only the first 7 parameters of $\hat{\theta}$ are shown. Red lines separate the different working phases. The top, middle and bottom horizontal axes display the backlash level, the working phase duration and number of machine cycles reached, respectively.

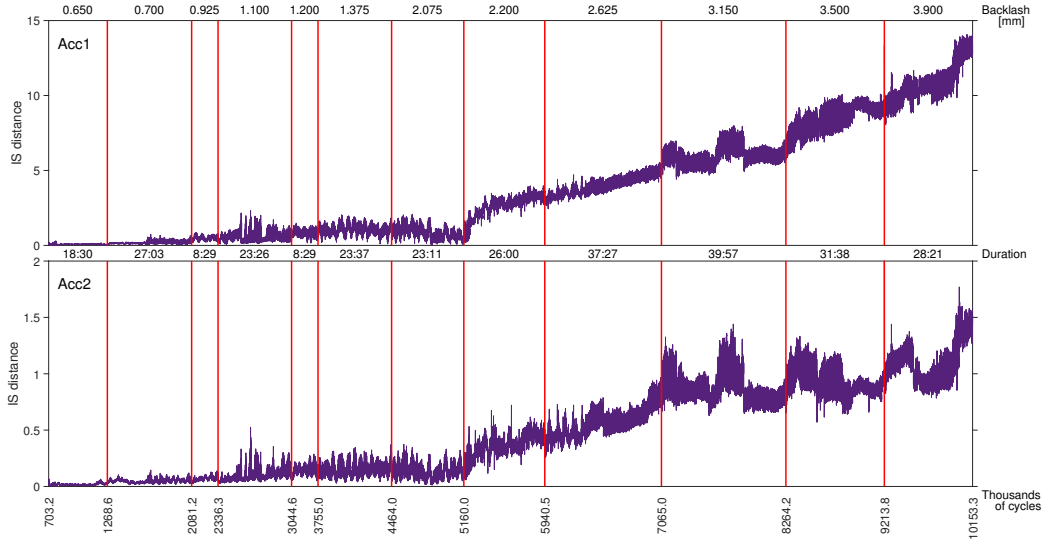

FIGURE 5.4: Itakura-Saito distance evolution in time. Red lines separate the different working phases. The top, middle and bottom horizontal axes display the backlash level, the working phase duration and number of machine cycles reached, respectively.

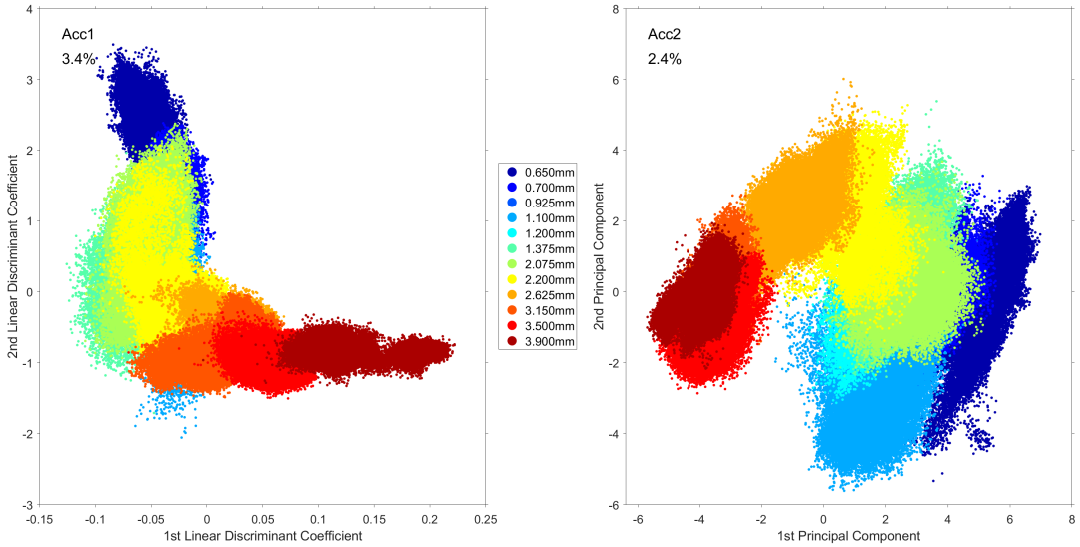

FIGURE 5.5: LDA (left) and PCA (right) of the features of the test set, with prediction coloured depending on the assigned class and accuracy displayed in the top left corner.

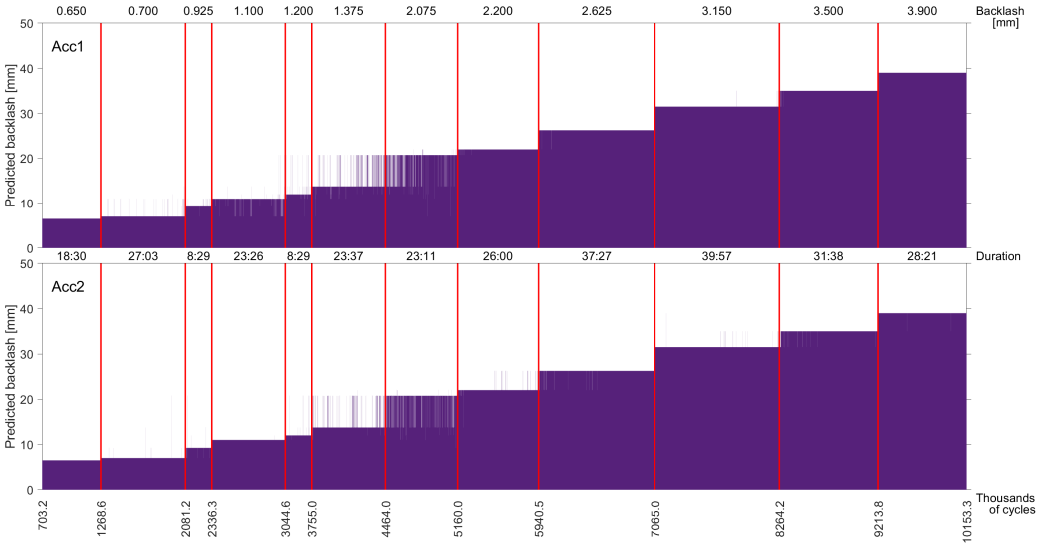

FIGURE 5.6: Label prediction in time. Red lines separate the different working phases. The top, middle and bottom horizontal axes display the backlash level, the working phase duration and number of machine cycles reached, respectively.

		Predicted labels [mm]											
		0.650	**0.700**	**0.925**	**1.100**	**1.200**	**1.375**	**2.075**	**2.200**	**2.625**	**3.150**	**3.500**	**3.900**
Actual labels [mm]	**0.650**	28 244	0	0	0	0	0	0	0	0	0	0	0
	0.700	0	39 789	18	705	21	22	9	0	0	0	0	0
	0.925	0	37	12 350	330	0	0	0	0	0	0	0	0
	1.100	2	1 122	203	33 690	178	113	76	5	0	0	0	0
	1.200	0	15	0	274	11 764	467	203	1	0	0	0	0
	1.375	0	13	1	249	268	30 363	4 521	2	0	0	0	0
	2.075	0	29	0	73	282	4 825	29 346	207	0	0	0	0
	2.200	0	0	0	0	0	0	306	38 651	6	0	0	0
	2.625	0	0	0	0	0	0	0	33	56 124	0	0	0
	3.150	0	0	0	0	0	0	0	0	0	59 808	96	0
	3.500	0	0	0	0	0	0	0	0	0	261	47 121	8
	3.900	0	0	0	0	0	0	0	0	0	0	18	42 447

TABLE 5.2: Confusion matrix of Acc1 test data. Each row contains the total number of models belonging to the relative class distributed in each column according to the predicted label.

		Predicted labels [mm]											
		0.650	**0.700**	**0.925**	**1.100**	**1.200**	**1.375**	**2.075**	**2.200**	**2.625**	**3.150**	**3.500**	**3.900**
Actual labels [mm]	**0.650**	28 244	0	0	0	0	0	0	0	0	0	0	0
	0.700	0	40 412	111	3	0	29	9	0	0	0	0	0
	0.925	0	168	12 520	10	2	7	10	0	0	0	0	0
	1.100	0	6	13	35 308	28	10	23	1	0	0	0	0
	1.200	0	0	3	13	12 407	126	173	2	0	0	0	0
	1.375	0	2	4	34	75	31 973	3 300	29	0	0	0	0
	2.075	2	14	11	55	151	3 126	31 232	171	0	0	0	0
	2.200	0	0	0	0	6	13	253	38 011	680	0	0	0
	2.625	0	0	0	0	0	0	0	672	55 485	0	0	0
	3.150	0	0	0	0	0	0	0	0	1	59 458	440	5
	3.500	0	0	0	0	0	0	0	1	0	951	46 422	16
	3.900	0	0	0	0	0	0	0	0	0	8	44	42 413

TABLE 5.3: Confusion matrix of Acc2 test data. Each row contains the total number of models belonging to the relative class distributed in each column according to the predicted label.

5.1.3 Monitoring Results

Initially, we analyze the data collected and processed on-board the PLC. Figure 5.3 presents the first 7 parameters of the signal models computed for both Acc1 and Acc2 during the run-to-failure test. The top, middle, and bottom horizontal axes show the backlash level, the time duration, and the number of cycles reached for each working period, respectively. Even if for the sake of clarity and space we do not show all of them, the graphs allow already a qualitative recognition of the different degradation stages of the mechanism. This suggests that the use of models as features for the automatic generation of PHM indicators is practicable for this condition monitoring solution.

To this extent, we provide also the data relative to IS distance measurement obtained throughout the testing in Figure 5.4. It shows that a scalar quantity, computed locally, can provide insight into the system state of health. It has an increasing trend, similar to the backlash one. The IS indicator turns out to be useful for implementing fault detection policies, with thresholds applied to its level. However, it does not permit a clear diagnosis of the severity of the degradation. The same distance value corresponds to several backlash amounts, particularly in the early stages of mechanism deterioration. In this case, a suitable threshold for fault detection may be defined at around a level of 5 for Acc1 and 1 for Acc2.

Figure 5.5 and 5.6 depicts the results of the application of the SVM algorithm to predict labels of the models' test set for Acc1 (left) and Acc2 (right). In particular, Figure 5.5 shows the projection of that feature set using Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA), on the left and right side, respectively. Points color corresponds to the predicted labels assigned by the algorithm, with the achieved classification accuracy displayed in the top left corner of each plot. A value of 3.4% for Acc1 and 2.4% for Acc2 indicates positive results for PHM. Moreover, despite being point projections, it is possible to observe the path that features follow while the machine is degrading. On the other hand, Figure 5.6 outlines predicted labels against the actual ones, revealing an almost complete stairstep graph. There, it is possible to observe that the SVM predictor mostly struggles with backlash levels 1.375 mm and 2.075 mm when giving inaccurate outcomes, and notice that most of the miss-predictions are around the class dividing line. Moreover, this can also be observed in the behavior of the IS distance in Figure 5.4, where the index holds the same value for both the said classes. In this respect, we also provide the resulting confusion matrices in tables 5.2 and 5.3 for Acc1 and Acc2, respectively. Each row represents the test set of a single class, subdivided into columns according to the predicted labels. Numbers in table rows sum up to the total model amount of the relative test set.

The SVM classifier results able to predict accurately the various degrees of wearing in the mechanism and the use of the two-level monitoring architecture is capable of providing information for predictive maintenance decision making. On the other hand, the IS distance results to be a valuable indicator for fault detection: Its computation on the controller does not require data exchange with external computers. However, it is advisable for the definition of preventive maintenance policies and not for predictive maintenance strategies.

5.2 Case Study: Bearing Prognostics Using MoS and Particle filtering

Smart manufacturing sites typically use PLCs to handle the production process and PCs to supervise their work. The technological development of those devices enables the integration of autonomous health management solutions alongside logic control tasks. Given these considerations, together with the previously mentioned brief review, we propose an efficient and practical methodology that allows integrating automated PHM modules in machine

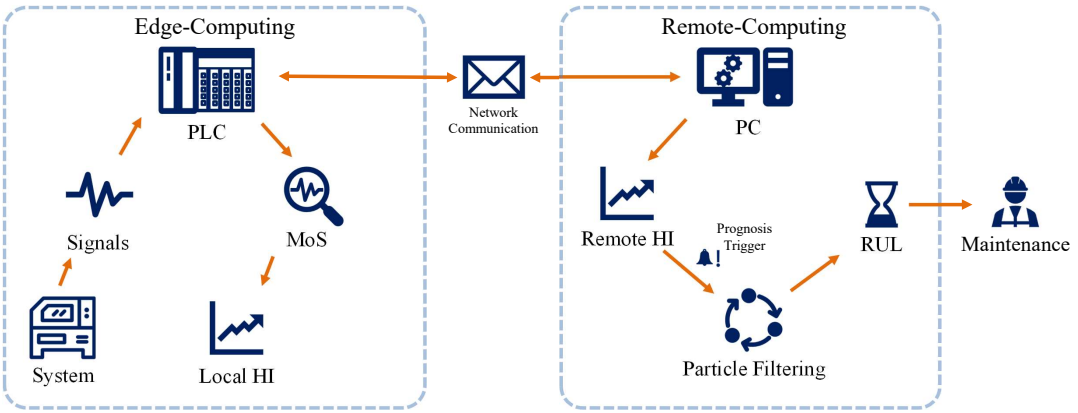

FIGURE 5.7: Schematic of the overall methodology.

controllers, using standard industrial platforms and architectures. The proposed solution, presented in Fig. 5.7, is based on a widely used architecture by industrial manufacturers: PC-supervised PLCs connected via LAN. The PLCs ensure the edge-computing unit role, providing the monitoring equipment and the first refinement of the collected data. The PC supervisor is the remote computing unit providing the final data processing to extract reliable prognostics information for maintenance-decision-making purposes.

Besides the logic control task, the PLC is capable of handling the measured signals (e.g., currents, temperatures, and vibrations), coming from the on-board sensing devices, and process them (see the left side of Fig. 5.7). In detail, the Model-of-Signals technique, implemented on PLCs, allows extracting useful information from sensor measurements. It compresses huge-data streams into models that retain the majority of the inherent signal characteristics. Its implementation can be easily performed based on main coding languages used to program machinery controllers, such as Structured Text (ST) (Barbieri, 2017).

Then, the estimated model's parameters are used to calculate Health Indicators (HIs), giving an indication of the health state of monitored parts on top of which the signals are measured. Their elaboration can be done locally or remotely, depending on the complexity of the computations and related loads. In this work, we propose the construction of both a local HI and a remote HI (see section 3.1 for the details). The first one serves as a backup for the worst-case scenario of a LAN disconnection. It allows continuous monitoring of the system's health state when it is impossible to receive information from the supervisor. The second one is the primary indicator used for health state prediction and maintenance decision-making.

The edge-computed model, built based on the Model-of-Signals technique, is sent over LAN to the remote computing unit, see Fig. 5.7 on the right side. This remote computing unit enables complex calculations that require higher computational resources for a reliable RUL prediction. To do that, the PC computes the related HI and monitors its evolution in time. Machines typically work under nominal conditions for a long time before an anomaly occurs in their components. During those healthy conditions, it is not necessary to perform prognostics. Hence, it is required the identification of a HI threshold $T_{h_{\text{prog}}}$ for anomaly detection, which is related to the moment when the degradation of the monitored systems starts. This event triggers the execution of the prognostics task performed by Particle Filtering. The PF is used to learn the degradation model and, consequently, predict its evolution toward the failure threshold. The definition of this second threshold is essential to evaluate the monitored system's Remaining Useful Life (RUL). Finally, this information provides valuable indications for maintenance optimisation and decision-making.

For this purpose, two case studies are employed to test and validate the method. The PRONOSTIA dataset is investigated to guide the industrial practitioners on how to apply the proposed methodology and lay the foundations for autonomous health management functionalities on PLCs. The procedure is described in detail from the implementation of condition monitoring through MoS on controllers to the definition of the PF degradation model, based on the HI evolution, and RUL forecasting on the supervising PC. Besides, the IMS dataset has been used to further evaluate the methodology under similar PHM conditions. Then, performance indicators have been computed for both the datasets using the prognostic horizon metric.

5.2.1 PRONOSTIA Dataset

The first case study relies on the bearing prognostics dataset PRONOSTIA (Nectoux et al., 2012), obtained from the NASA prognostics Data Center (NASA, 2019). It provides run-to-failure vibrational signals obtained from 17 ball bearings of the same typology under 3 different operating conditions, causing accelerated degradation on the component under test.

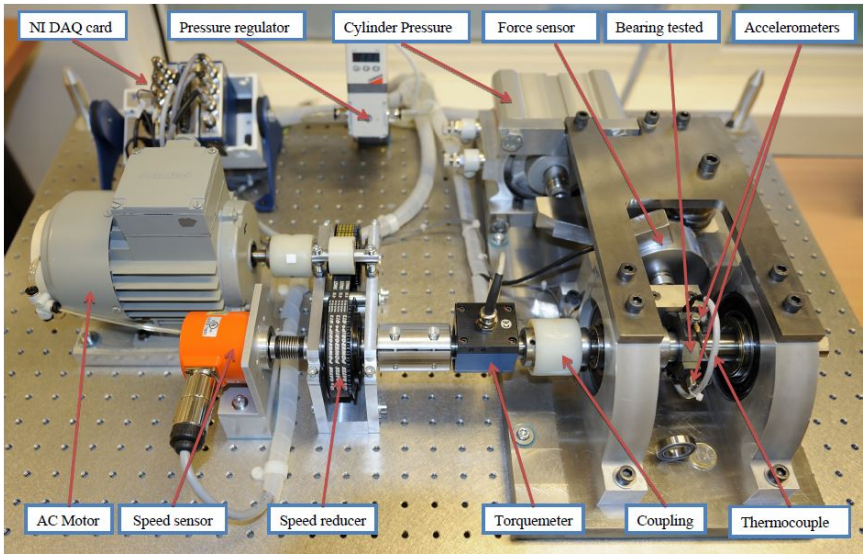

FIGURE 5.8: Experimental ball bearing diagnostics and prognostics setup: PRONOSTIA (Nectoux et al., 2012).

The test bench setup, shown in Fig.5.8, consists of an AC motor driving with timing belts. The shaft is connected to the inner ring of the bearing under test. The NI sensing system provides logging files of the data collected from two accelerometers positioned at 90° from each other: one measuring vibration in the direction perpendicular to the base, while another is set up parallel to the base. They are sampled with a frequency of $F_S = 25600$ Hz for 0.1 s of acquired signal every 10 s. Accelerated degradation is simulated by means of a pneumatic actuator pushing radially on the bearing frame/outer ring. Shaft rotating speed and actuator force determine the load conditions under which the test is performed and are defined as follows:

1. Speed: 1800 rpm - Force: 4000 N;
2. Speed: 1650 rpm - Force: 4200 N;
3. Speed: 1500 rpm - Force: 5000 N.

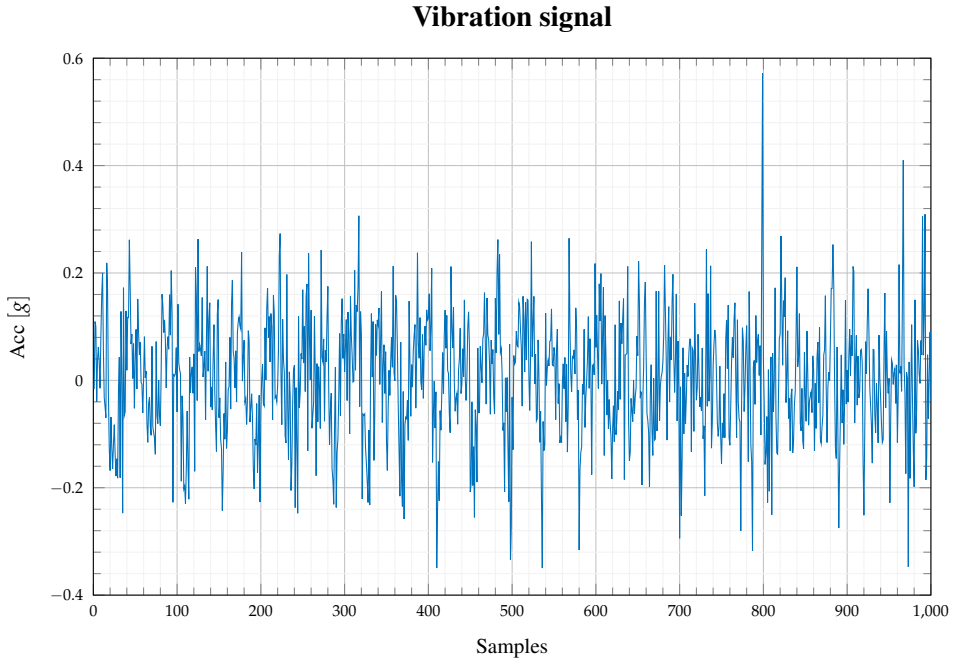

FIGURE 5.9: Vibration signal from the database.

The dataset contains run-to-failure measurements of 7 bearings under condition (1), 7 ones under condition (2), and 3 ones under condition (3). Vibration signal logs are organized in folders, one folder for one tested bearing, and divided into time-sorted `.csv` files, containing 2560 samples per file, for both accelerometers. Originally, the dataset was distributed for the *PHM Challenge* in 2012 and divided into two parts: training and validation. The former consists of 2 bearings per condition and the latter contains the remaining ones with truncated logs. Nowadays, the full data set is available for research purposes. An example of a vibration signal from the database is shown in Fig. 5.9.

The proposed methodology (Section 1.2) is applied on the described PRONOSTIA open-source database to serve as a reference for practitioners in industrial environments with PLC-controlled automatic machines. Its algorithms are coded with Matlab and can be easily implemented on the proposed PLC-Supervisor architecture. To do this, we provide practical guidelines, starting from the arrangement of measurements for MoS estimation, through the communication procedure between the logic controller and supervisor, and ending with the use of Particle Filtering for RUL prediction.

Edge-Computing on PLCs

Machinery controllers are based on real-time operating systems. They can respond to events within precise timing constraints, and they are reliable in accurately handling sensor measurements within their Fieldbus network. For instance, Beckhoff EL3632 and B&R X20CM4800X I/O modules have sampling rates of up to 25 kHz. On the other hand, bearing failing modes are a function of the driving shaft speed and typically in the range of $1 - 500$ Hz. In PRONOSTIA (Soualhi et al., 2014) those modes are around 200 Hz. Higher sampling rates require a lot of available bandwidth on the Fieldbus and also PC-based PLCs powerful enough to handle them, together with the control task. On the other hand, a sampling of 1 kHz is enough in this situation to catch the faulty behavior of the component. Thus, reducing the sampling rate allows the methodology to be feasible also in less powerful controllers

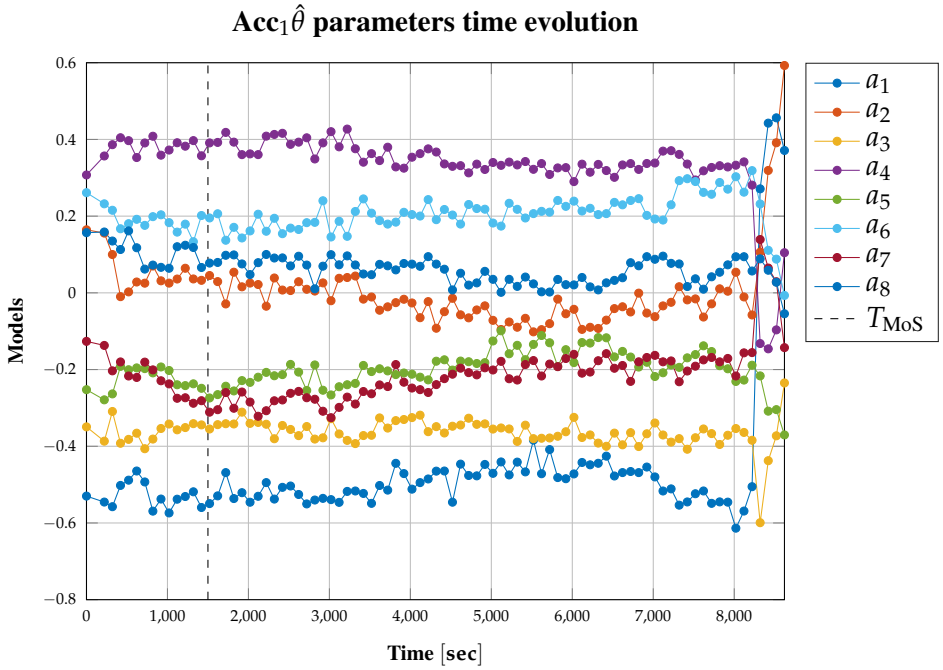

FIGURE 5.10: MoS evolution in time of bearing 2 of testing condition 1

without hindering the logic control task and without any loss of prognostics performance, reducing also the costs of its implementation.

Given those considerations, we downsampled PRONOSTIA bearing vibration signals to 2560 Hz (i.e by a factor of 10), so a log file now carries 256 usable samples per accelerometer. Since MoS identification algorithm requires a higher number of data to feasibly perform its computations, we concatenated windows of 12 files together into a circular buffer: $N = 3072$ samples over a 2 min span in PRONOSTIA time steps. This solution measures vibrations for a brief interval ($T_{meas} = 0.1$ s) every larger interval ($T_{acq} = 10$ s) and concatenates them. It facilitates the implementation of such buffering modalities in non-powerful logic controllers thanks to its low requirements in terms of computational and memory resources. Moreover, the use of a circular buffer, that updates with a FIFO policy every time a new (batch) measurement is available, allows providing the sequence $y(1), \ldots, y(N)$ to the MoS program every T_{acq}. This means, in our case, the model can be elaborated every 10 s.

Once the buffer is ready, the array of data is fed to the ORIV, algorithm 2.6. It has been configured with model order $n = 8$, which was chosen through the MDL criterion, and the hyper-parameter $q = n = 8$, see section 2.7. In the end, the algorithm is initialised with $\psi = 10$ as stated in (2.56). The model parameters obtained from Bearing 2's vibration signals under test condition 1 are shown in Fig. 5.10, where it is possible to appreciate their evolution during the component life span. We highlighted the moment at 1500 s by a dashed vertical line because that is roughly the moment at which the bearing lubricant grease reaches its temperature working point (in the range of $90 - 110°$C). This instant marks the starting of model transmission to the supervisor, and the reference model θ_{nom} is obtained by computing the mean value of the first 10 collected ones.

The computations of the local HI start after the reference model is defined. When there is a disconnection between PLC and PC supervisor, the HI_{NRMSE} is used as a backup solution for fault detection, with the failure threshold defined in the range $[1, 1.1]$. A sample of the HI_{NRMSE} evolution is shown in Fig. 5.11 where the application of threshold $T_{h_{\text{NRMSE}}} = 1$ is applied to avoid severe failing modes.

At this point, the PLC has to handle the communication of the computed quantities to the

supervisor PC. The proposed architecture does not require high payloads in data transmission because only a piece of information, i.e., model parameters, is remotely sent from the PLC to the supervisor PC every 10*s*. Various communication protocols are available to perform this task and can be easily integrated into the majority of PLCs. Various communication protocols are available together with their functions library, for instance, *OPC-UA* and *MQTT* are commonly used. Besides, logging of the computed θs and of HI_{NRMSE} can be performed locally, if feasible for the system resources.

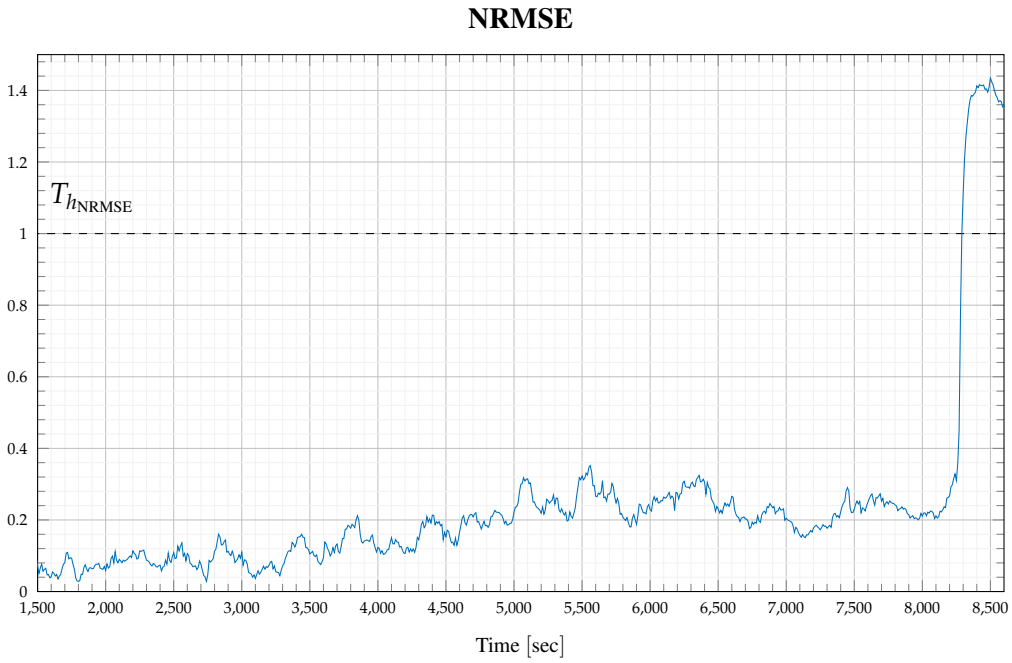

FIGURE 5.11: HI_{NRMSE} evolution over time of bearing 2 under testing condition 1.

Given the previous considerations, we can elaborate more on the "Huge-Data to Big-Data" claim. Suppose we want to continuously stream the vibration signal from the PLC to the PC via LAN. The controller collects each vibration measurement into a `REAL` type variable, which is worth $4B$ (bytes), at a frequency of 2560 Hz, meaning 10 KB/s of raw data flow not counting overheads. As the PLC simultaneously performs the logic control task, this transmission may be difficult to handle. On the other hand, in our depicted solution where only model parameters (`REAL` variables) are sent, the transmission flow is reduced and equals 3.2 B/s. Compared to signal to stream, this is 3000-times smaller, and much less demanding in terms of network and storage.

Fig. 5.12 presents the three functions previously described, i.e. measurement pre-processing, MoS generation, and network and storage, that are implemented within the PLC programs. The I/O acquisition modules have higher sampling rates with respect to PLC cycling times. This is typically handled by the Fieldbus so that a set of samples is collected and provided at each cycle. Thus, the measurement pre-processing program, that fills the data buffer for MoS, should be attached to the main priority task with the lower period (usually 1 ms). This allows managing the data flow with feasible modalities, suitable to follow the acquisition schedule previously depicted. MoS generation is the most demanding in terms of computational load on the CPU of the PLC. It should be appended to a lower priority task (e.g. a program with a $10 - 100$ ms cyclic period) than the main control program. Network and storage handling programs are usually already available on machines, for production supervision and quality control. The quantities required for transmission by the methodology should be added

effortlessly, as discussed previously.

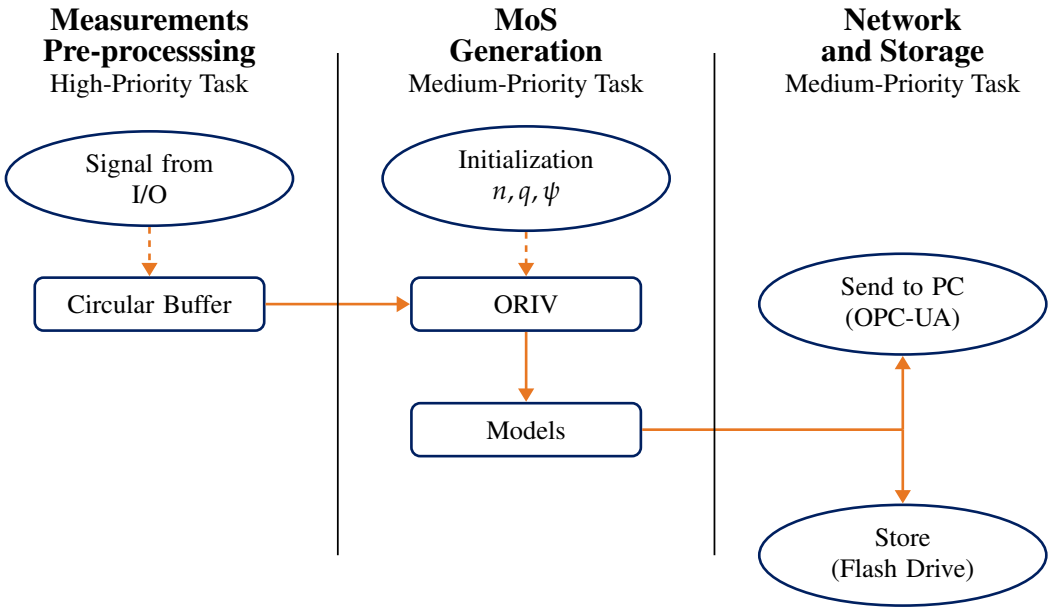

FIGURE 5.12: PLC programs and internal information flow of the procedure on the edge side.

Remote-Computing on PC

On the supervisor side, the remote computing unit, i.e., a PC, collects the data sent from the PLC and can possibly log and store them. Then, its next task is to compute the $HI_{\text{I-S}}$ health indicator, presented in eq. (3.2), on which the prognostics procedure is based. Firstly, since this index is quite noisy, a simple filtering method using the mean value of 12 samples is applied to smooth it. The obtained result is noted as $\bar{HI}_{\text{I-S}}$. Then, the index is monitored until it reaches the anomaly threshold $T_{h_{\text{prog}}} = 0.1$ where the prognostics procedure starts.

Prognostics is handled by exploiting the properties of Particle Filtering to learn the parameters of the degradation model and then to forecast its evolution. Considering Fig. 5.13, showing only the main cases for the sake of compactness and readability, $\bar{HI}_{\text{I-S}}$ is an increasing function whose behaviour may seem different from bearing to bearing. By a closer analysis, it is possible to point out that such quantity follows a piece-wise evolution starting from an exponential growth, whose parameters are to be learnt, and ending with a polynomial tail. This means that the first stage of the degradation process can be represented by an exponential function, while the second stage, that starts at a random time, T_{change}, may be characterised by a polynomial function. To capture this changing point, after the aforementioned analysis, we define a threshold for the first order difference of the health indicator, namely $T_{h_{\text{diff}}}$. In detail, the changing point is identified by the moment, at which the difference of $\bar{HI}_{\text{I-S}}$ at two consecutive times is superior than $T_{h_{\text{diff}}}$. Hence, the $\bar{HI}_{\text{I-S}}$ evolution is re-written as follows:

$$\bar{HI}_{\text{I-S}}(t_k) = \\ = \begin{cases} ae^{bt_k} & \Delta\bar{HI}_{\text{I-S}}(t_k) \leq T_{h_{\text{diff}}} \\ c(t_k - T_{\text{change}})^2 + d & \Delta\bar{HI}_{\text{I-S}}(t_k) > T_{h_{\text{diff}}} \end{cases} \tag{5.1}$$

where

$$\Delta\bar{HI}_{\text{I-S}}(t_k) = \bar{HI}_{\text{I-S}}(t_k) - \bar{HI}_{\text{I-S}}(t_{k-1}) \tag{5.2}$$

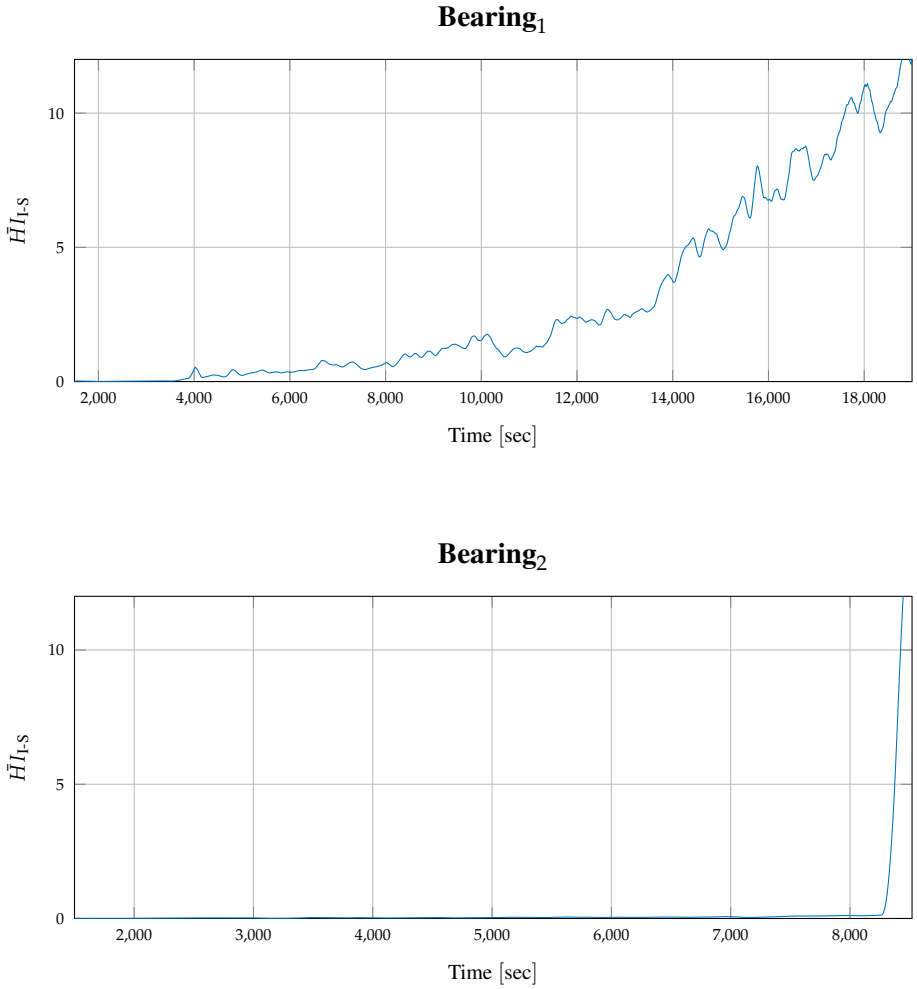

FIGURE 5.13: Evolution of $\bar{HI}_{\text{I-S}}$ of bearing 1 and 2 under testing condition 1 during their relevant run-to-failure tests.

and $T_{h_{\text{diff}}}$ is determined in this case as 0.2. Then, the coefficients a and d correspond to $T_{h_{\text{prog}}}$ and $\bar{HI}_{\text{I-S}}(T_{\text{change}})$, while b and c are positive parameters modelled as conditioned random walks:

$$b(t_k) = b(t_{k-1}) + \lambda(t_k) \tag{5.3}$$
$$c(t_k) = c(t_{k-1}) + \gamma(t_k) \tag{5.4}$$

with $\lambda \sim \mathcal{N}(0, \sigma_\lambda^2 = 10^{-5})$ and $\gamma \sim \mathcal{N}(0, \sigma_\gamma^2 = 10^{-5})$. The learning phase of the Particle Filter, in this case, is employed to estimate firstly b and secondly c. Then, the predictions are done at every time step and RUL is estimated based on the end of life threshold $T_{h_{\text{EoL}}} = 10$, which is a safe one, to stop production and to start maintenance activities before it is too late.

Fig. 5.14 and Fig. 5.15 illustrate how the Particle Filter performs when the degradation evolves only through the exponential function (e.g. Bearing 3) or when there exists the model changing point (e.g. Bearing 4). These figures describe what happens at a particular time instant t_k where the PF learning ends and the prediction starts. $\bar{HI}_{\text{I-S}}$'s actual evolution is shown in blue, while the filtered PF mean state computed until t_k is shown in yellow and the state prediction from that point on is in orange. The introduced model allows state filtering close to the observed one and its propagation fits $\bar{HI}_{\text{I-S}}$ evolution. Moreover, the final RUL probability density is shown in red together with threshold levels $T_{h_{\text{prog}}}$ and $T_{h_{\text{EoL}}}$ presented

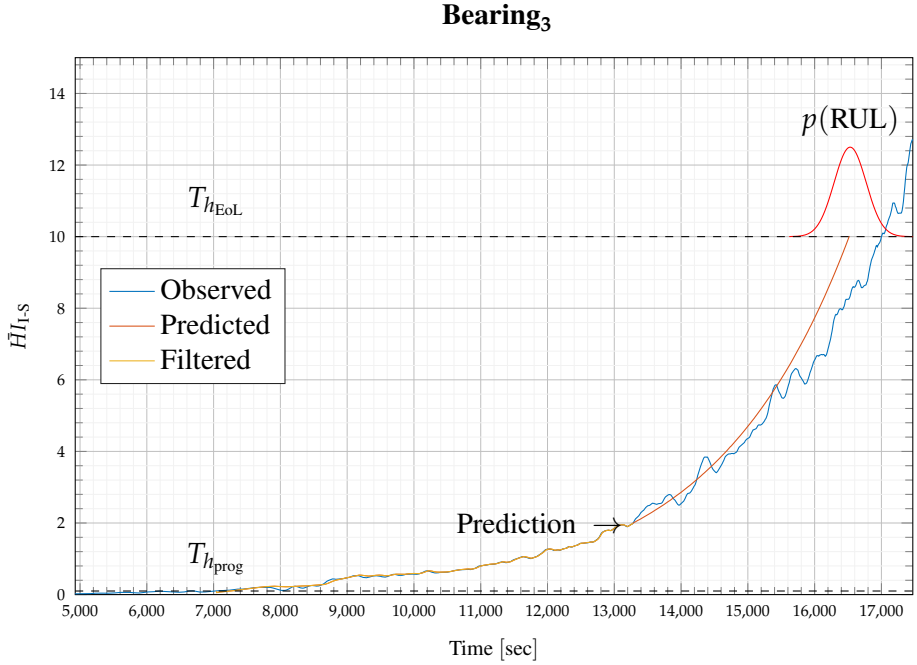

FIGURE 5.14: Evolution of $\bar{H}I_{\text{I-S}}$ of bearing 3 under test condition 1: Filtering, state estimation, and RUL prediction.

in dashed black lines. $p(\text{RUL})$ depends on how long the prediction is propagated to reach $T_{h_{\text{EoL}}}$. The longer the time it takes the greater its variance is. Moreover, in Fig. 5.14, RUL prediction helps in maintenance decision making, since, given the accelerated degradation, there is plenty of time to plan the servicing (we have 50 minutes to play with). Besides, Fig. 5.15 shows that RUL prediction can be used to avoid severe bearing degradation which may result in a critical failure.

A broader view of the RUL prediction is shown in Fig. 5.16 and Fig. 5.17. These figures represent, by the blue-dotted line, the mean value of the computed RULs of the various particles throughout the test, from the starting of prognostics to the component end of life. The orange-dashed line represents the actual RUL. Fig. 5.16 shows feasible RUL prediction when there is no change in the underlying PF model structure, after an initial drift, the more the filter "learns" the more it gets close to the actual value. On the other hand, the moment when the model structure change occurs is uncertain. From the available data, it turns out that it is not predictable, however, our procedure is capable of catching it as soon as it occurs, offering essential times for intervention. Fig. 5.17 shows how the predictor treats the incoming observation with the exponential model until it recognises the change in the evolution and gets back on track with the actual RUL value. For these reasons, the communication between PLC and PC should be in both directions. From an automated maintenance perspective, the supervisor transmits the predicted RULs to the machinery to trigger fail-safe policies in the worst-case scenario.

Performance Results

Finally, to assess the performance of the algorithm on the overall dataset we make use of the Prognostic Horizon (PH) metric (Saxena et al., 2010), a standard indicator that provides information on how the algorithm is able to suitably predict the system RUL. PH is defined as the difference between the time index t when the predictions first meet the specified performance criteria (based on data accumulated until time index t) and the time index for EoL,

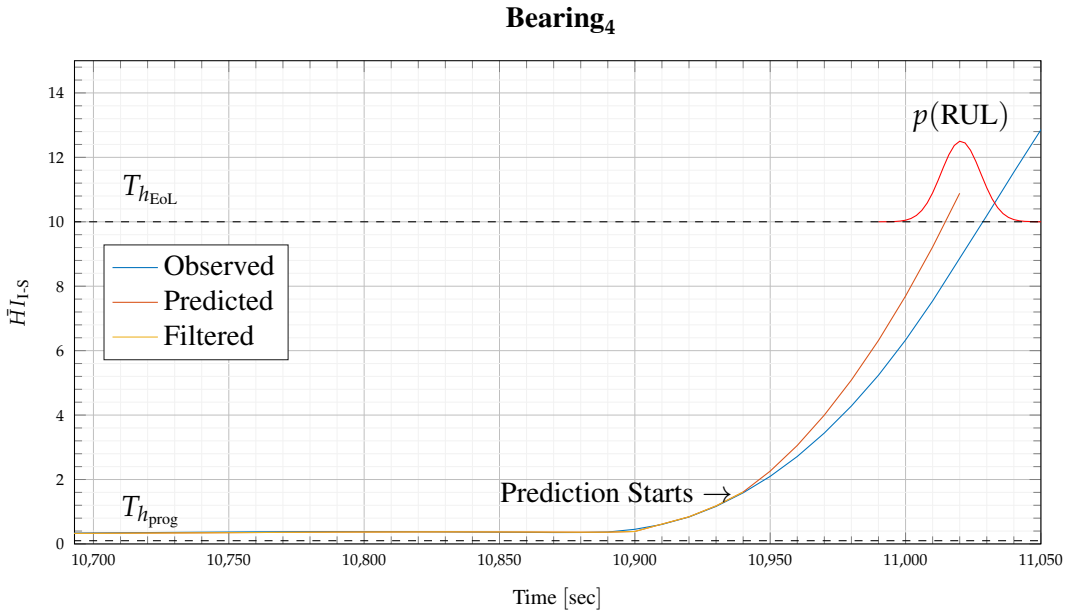

FIGURE 5.15: Evolution of $\bar{H}I_{\text{I-S}}$ of bearing 4 under test condition 1: Filtering, state estimation, and RUL prediction.

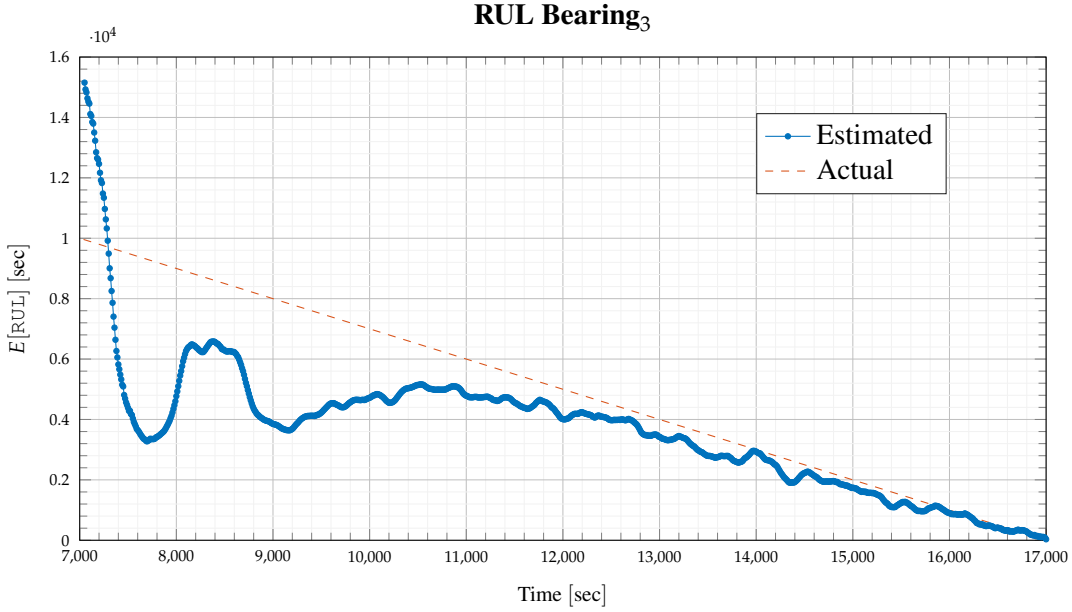

FIGURE 5.16: RUL over time of bearing 3 under test condition 1 using the proposed prognostics PF method.

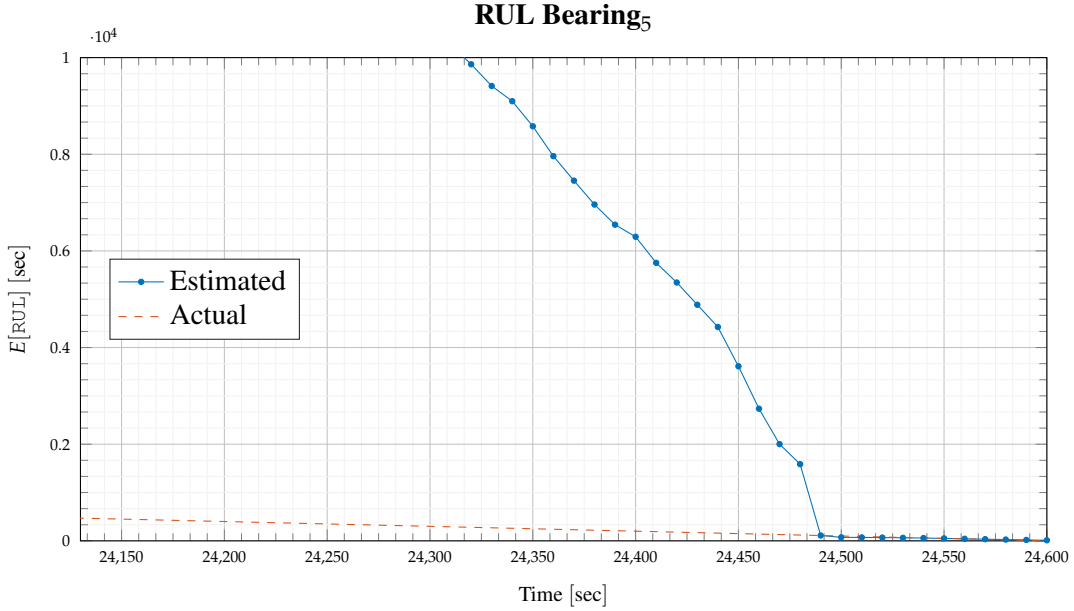

FIGURE 5.17: RUL over time of bearing 5 under test condition 1 using the proposed prognostics PF method.

TABLE 5.4: Prognostic Horizon results on PRONOSTIA dataset. Data obtained with $\alpha = 0.2$.

PH (minutes)	**Bearing**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Cond. 1	136.5	2	125.17	1.67	2.5	3.17	4.3
Cond. 2	3.5	2.17	-	2	-	1.17	1
Cond. 3	1.17	20	3.67				

t_{EoL}. The performance requirement is specified in terms of an allowable error bound (α) around the true EoL where the choice of α depends on the estimate of time required to take a corrective action. A formal definition is given in the following:

$$PH = t_{\text{EoL}} - t_{k_\alpha} \tag{5.5}$$

with

$$k_\alpha = \min\{k \mid (k \in \mathbf{P}) \wedge ((r_{*_k} - \alpha.t_{\text{EoL}}) \leq \hat{r}(t_k) \leq (r_{*_k} + \alpha.t_{\text{EoL}}))\, \forall k \geq k_\alpha\} \tag{5.6}$$

where $\mathbf{P}$ is the set of all time indices for which a prediction is made, r_{*_k} is the true RUL at time t_k and $\hat{r}(t_k)$ is the predicted RUL at time t_k (i.e., its mean value in our particular case). Moreover, to stress the methodology robustness, we choose the value of PH having also the predictions obtained after t_{k_α} within the bounds. Table 5.4 shows the obtained results with $\alpha = 0.2$. The considerations drawn before about the commuting degradation model still hold and the results show that when the model function changes to polynomial there is a minute span to handle the possible fault. On the other hand, when the degradation keeps its exponential trend, at least 20 minutes are available to program preventive actions. Notice that no results are available for bearing 3 and 5 of condition 2 because their run-to-failure data is incomplete.

TABLE 5.5: Prognostic Horizon results on PRONOSTIA dataset. Comparison with (Soualhi et al., 2014).

PH (minutes)	Bearing 1	Bearing 2	Bearing 3
(Soualhi et al., 2014)	6.8	2	1.2
Our results	136.5	2	125.17

To highlight the performance of the proposed approach, its results are compared to the outcomes obtained in (Soualhi et al., 2014) on the first three bearings under condition 1. As shown in Table 5.5, our procedure has better results for bearing 1 and 3, and the same result for bearing 2.

5.2.2 IMS Dataset

To further validate our proposition, we port it on another bearing prognostics dataset within the NASA prognostics Data Center (NASA, 2019), the IMS (Lee et al., 2007) one. It contains three run-to-failure experiments with four bearings each, constantly driven at 2000 rpm and under a radial load of 6000 lbs $(2721.554\ \text{kg})$. Vibrational data of each bearing were collected for $T_{meas} = 1$ s at a 20 kHz every $T_{acq} = 10$ min and logged into `.txt` files until bearings EoL. We applied the proposed methodology with the same course of action previously described throughout section 5.2.1. To avoid redundancy, for this case study, we briefly present in this section the modalities of data pre-processing to suit MoS estimation and PF structure, and also the definition of the method hyperparameters.

Given the data set organisation, the signal set-up for MoS estimation is similar to the previous one. The data are downsampled to 2500 Hz for the same parsimony and cost-effectiveness reasons described for PRONOSTIA. Each logged file is regarded as the data buffer (with a window size of 2500 samples) on which a model is identified. In this case, the first two bearings of the first test were used to obtain the method's hyperparameters in advance. The resulting model order was $n = 8$ and $q = 8$ more equations were used to increase the robustness of the ORIV estimation. Besides, the healthy references for the computations of the HIs are computed by averaging the models obtained in the first 2 operational hours of each bearing. Then, the threshold to start prognostics on the $\bar{HI}_{\text{I-S}}$ indicator, which is filtered in the same way as in PRONOSTIA, has been set to $T_{h_{\text{prog}}} = 0.1$. The underlying particle filtering prognostics model is unchanged from (5.1)-(5.4) except for the commutation threshold, which has been set to $T_{h_{\text{diff}}} = 0.11$. Finally, for this data set the End-of-Life threshold, on which RUL predictions are computed, is $T_{h_{\text{EoL}}} = 2$. A snapshot of the prognostic task using the particle filtering model of eq. (5.1) with the selected thresholds is shown in Fig. 5.18 for bearing 3 of the second run-to-failure test. The overall evolution of the mean value of the estimated RUL of that bearing is provided in Fig. 5.19.

Table 5.6 shows the results obtained by the methodology in prognostic performance using the PH metric. In this situation, the methodology still holds its performance and is able to grant at least a 10 minutes scope for prognosis when a model change occurs, while at least almost 6 hours when the model keeps its exponential evolution.

Even though the focus was on the development of an "industrial technology aware" methodology, we made use of bearings datasets to develop our discussion on its implementation, allowing us to cover the main technological and architectural aspects of the method application. Consequently, this choice bonded our proposition with the mechanical domain and in particular with the use of vibrational signals as a starting point. Indeed, the proposed

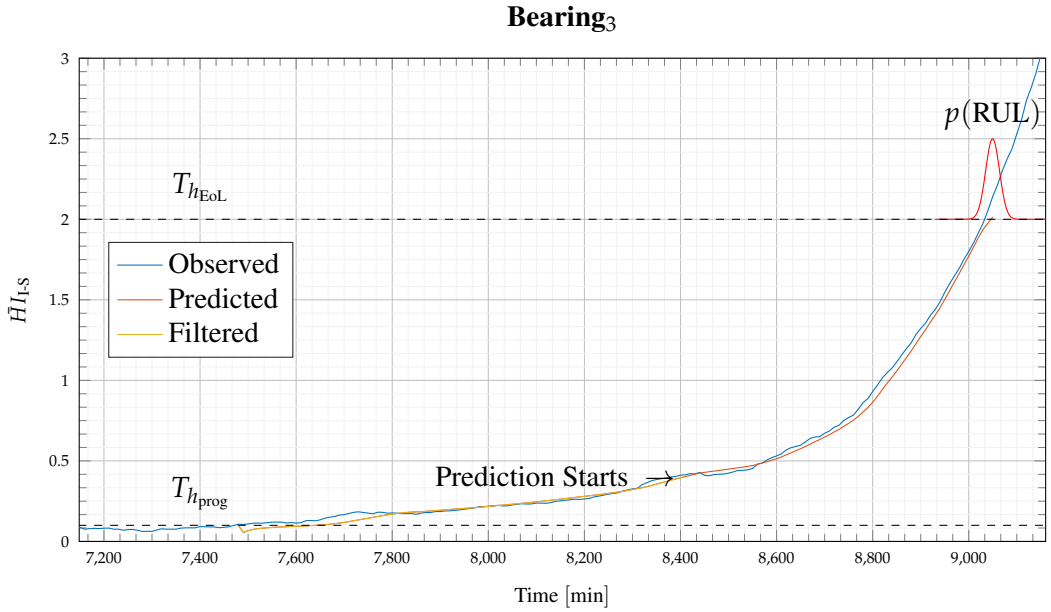

FIGURE 5.18: Evolution of $\bar{H}I_{\text{I-S}}$ of bearing 3 of test 2: Filtering, state estimation, and RUL prediction.

TABLE 5.6: Prognostic Horizon results on IMS dataset. Data obtained with $\alpha = 0.2$.

PH (minutes)	**Bearing**			
	1	**2**	**3**	**4**
Test 1	90	100	10	10
Test 2	1010	1250	1300	810
Test 3	1440	1100	350	1450

degradation model is not general and inevitably considers the nature of the component under test and the quantity to which we link its wearing.

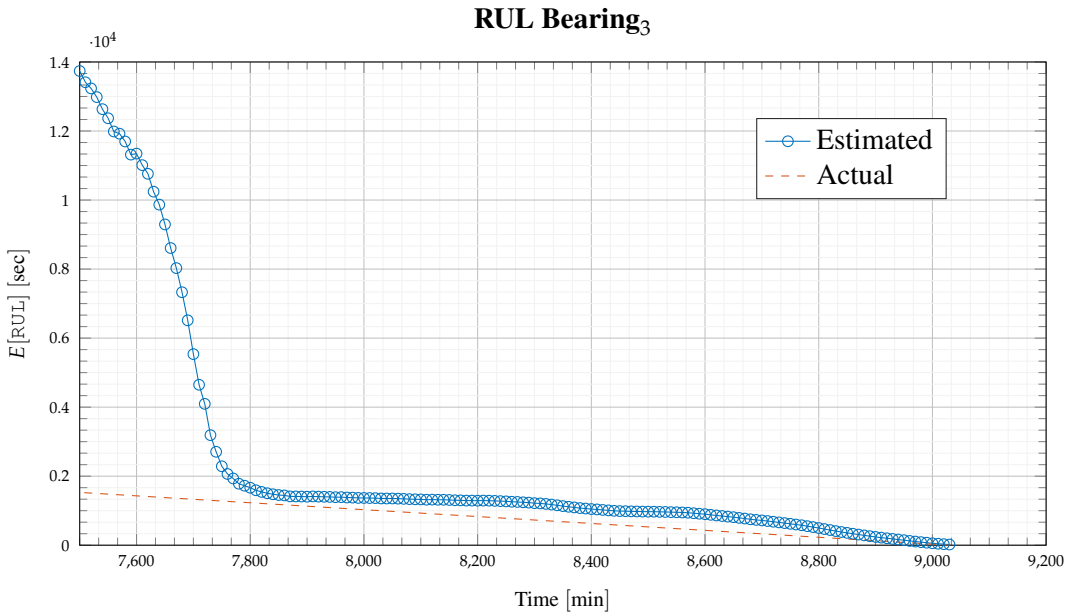

FIGURE 5.19: RUL over time of bearing 3 of test 2 using the proposed prognostics PF method.

Chapter 6

Conlusions

In this dissertation, we proposed a variety of PHM solutions based on Model-of-Signals. They focus on exploiting the increased computational power of machinery controllers and their interconnection with the automation pyramid. Briefly, the machine controller is the edge-computing unit that performs the condition monitoring task. It employs the Model-of-Signals technique to refine the information measured from onboard sensors into compact and meaningful features. The supervising PCs act as the remote-computing units collecting those computed models to produce health indicators for the PHM task. Machine learning provides a large variety of algorithms able to process and refine the models into valuable knowledge. Then, this knowledge becomes valuable information for the definition of maintenance strategies, either human-operated or automatic.

In particular, manufacturers can take advantage of this family of methodologies to integrate autonomous maintenance policies as features in their machines, increasing their competitive advantage with a solution that increases productivity and reduces costs, and keeping their expertise about standard automation platforms. The procedures discussed exploit the structure of the automation pyramid as well as the commercial equipment already in place, avoiding the addition of non-standard equipment to perform the task. Local sensor data refinement allows reducing the impact of PHM information transmission over the network and to distribute the computational load. This converts the "Huge-Data" problem of streaming raw sensor signals to remote computing units into a manageable one. This is possible because of the lightweight nature of the recursive algorithm used to produce MoS that can feasibly run alongside the logic control task.

The selection of the adequate signals upon which apply MoS and the related recursive estimation algorithm is crucial to accomplish the first stage of the methods. For instance, the expertise in the industrial automation world allowed us to develop new ways to exploit the torque of trajectory-driven mechanisms into the MoS framework. On the other hand, in-depth analysis of advanced black-box system identification allowed us to introduce more robust methods in edge-computing-based model estimation and a new recursive algorithm in the case of noisy measurements. Concerning this last accomplishment and its consequences, it provides significant help in case studies and applications that is now time to investigate.

Future steps involve the study of MoS-based procedures into more involved applications, both with components whose PHM is established with other techniques to which we want to compare and with new mechanisms and prototypes. The development of MoS is linked to the adaptation of system identification theory to the problem, with a plethora of modelling approaches to explore and exploit. Not to mention the fact that in this work we only scratched the surface of the possibilities that machine learning brings as a complementary method.

Given those considerations, this work aimed to build a first bridge between the industry and the research field. The obtained results are promising, laying the foundations for the deployment on industrial equipment of the method to "unlock" machinery smart potential, a step toward intelligent manufacturing.

Bibliography

Akaike, Hirotugu (1974). “A new look at the statistical model identification”. In: *Selected Papers of Hirotugu Akaike*. Springer, pp. 215–222.

Aljanaideh, Khaled F and Dennis S Bernstein (2017). “Closed-loop identification of unstable systems using noncausal FIR models”. In: *International Journal of Control* 90.2, pp. 168–185.

Aljanaideh, Khaled F, S Sanjeevini, and Dennis S Bernstein (2018). “Time-domain errors-in-variables identification of transmissibilities”. In: *Proceedings of 2018 IEEE Conference on Decision and Control (CDC 2018)*, pp. 3012–3017.

Alpaydin, Ethem (2020). *Introduction to machine learning*. MIT press.

An, Dawn, Joo-Ho Choi, and Nam Ho Kim (2013). “Prognostics 101: A tutorial for particle filter-based prognostics algorithm using Matlab”. In: *Reliability Engineering & System Safety* 115, pp. 161–169.

Arablouei, Reza, Kutluyıl Doğançay, and Tülay Adalı (2014). “Unbiased recursive least-squares estimation utilizing dichotomous coordinate-descent iterations”. In: *IEEE transactions on signal processing* 62.11, pp. 2973–2983.

Arulampalam, M Sanjeev, Simon Maskell, Neil Gordon, and Tim Clapp (2002). “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. In: *IEEE Transactions on signal processing* 50.2, pp. 174–188.

Atamuradov, Vepa, Kamal Medjaher, Pierre Dersin, Benjamin Lamoureux, and Noureddine Zerhouni (2017). “Prognostics and health management for maintenance practitioners-Review, implementation and tools evaluation”. In: *International Journal of Prognostics and Health Management* 8.060, pp. 1–31.

Atamuradov, Vepa, Kamal Medjaher, Fatih Camci, Noureddine Zerhouni, Pierre Dersin, and Benjamin Lamoureux (2020). “Machine Health Indicator Construction Framework for Failure Diagnostics and Prognostics”. In: *Journal of Signal Processing Systems*, pp. 1–19.

Barbieri, Matteo (Oct. 2017). “Seamless infrastructure for "Big-Data" collection and transportation and distributed elaboration oriented to predictive maintenance of automatic machines”. MA thesis. University Of Bologna.

Barbieri, Matteo, Alessandro Bosso, Christian Conficoni, Roberto Diversi, Matteo Sartini, and Andrea Tilli (2018). “An Onboard Model-of-signals Approach for Condition Monitoring in Automatic Machines”. In: *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*. Wiley – ISTE, pp. 263–269.

Barbieri, Matteo, Francesco Mambelli, Roberto Diversi, Andrea Tilli, and Matteo Sartini (2019a). “Condition Monitoring by Model-of-Signals: Application to gearbox lubrication”. In: *Proc. of the 15th European Workshop on Advanced Control and Diagnosis (ACD 2019)*. Ed. by Springer.

Barbieri, Matteo, Roberto Diversi, and Andrea Tilli (2019b). “Condition monitoring of ball bearings using estimated AR models as logistic regression features”. In: *18th European Control Conference (ECC 2019)*, pp. 3904–3909.

Barbieri, Matteo, Francesco Mambelli, Jacopo Lucchi, Roberto Diversi, Andrea Tilli, and Matteo Sartini (2020a). “Condition Monitoring of a Paper Feeding Mechanism Using

Model-of-Signals as Machine Learning Features". In: *PHM Society European Conference*. Vol. 5. 1, pp. 1–10.

Barbieri, Matteo, Roberto Diversi, and Andrea Tilli (2020b). "Condition monitoring of electric-cam mechanisms based on Model-of-Signals of the drive current higher-order differences". In: *Proc. of the 21st IFAC World Congress*. Ed. by IFAC.

Barbieri, Matteo and Roberto Diversi (2020c). "Identification of noisy input-output FIR models with colored output noise". In: *Proc. of the 21st IFAC World Congress*. Ed. by IFAC.

Barbieri, Matteo, T.P. Khan Nguyen, Roberto Diversi, Kamal Medjaher, and Andrea Tilli (2020d). "RUL Prediction for Automatic Machines: A Mixed Edge-Cloud Solution Based on Model-of-Signals and Particle Filtering Techniques". In: *Journal of Intelligent Manufacturing* Accepted for online publication.

Bertrand, Alexander, Marc Moonen, and Ali H Sayed (2011). "Diffusion bias-compensated RLS estimation over adaptive networks". In: *IEEE Transactions on Signal Processing* 59.11, pp. 5212–5224.

Biagiotti, Luigi and Claudio Melchiorri (2008). *Trajectory planning for automatic machines and robots*. Springer Science & Business Media.

Bishop, Christopher M. (2006). *Pattern recognition and machine learning*. Springer.

Bobillet, William, Roberto Diversi, Eric Grivel, Roberto Guidorzi, Mohamed Najim, and Umberto Soverini (2007). "Speech enhancement combining optimal smoothing and errors-in-variables identification of noisy AR processes". In: *IEEE Transactions on Signal Processing* 55.12, pp. 5564–5578.

Cerone, Vito, Dario Piga, and Diego Regruto (2013). "Fixed-order FIR approximation of linear systems from quantized input and output data". In: *Systems & Control Letters* 62.12, pp. 1136–1142.

Cerrada, Mariela, René-Vinicio Sánchez, Chuan Li, Fannia Pacheco, Diego Cabrera, José Valente de Oliveira, and Rafael E Vásquez (2018). "A review on data-driven fault severity assessment in rolling bearings". In: *Mechanical Systems and Signal Processing* 99, pp. 169–196.

Chan, Y and Randy Langford (1982). "Spectral estimation via the high-order Yule-Walker equations". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 30.5, pp. 689–698.

Cortes, Corina and Vladimir Vapnik (1995). "Support-vector networks". In: *Machine Learning* 20.3, pp. 273–297.

CWRU (2014). *Case Western Reserve University Bearing Data Center*. URL: http://csegroups.case.edu/bearingdatacenter/home.

Davila, Carlos E (1994). "An efficient recursive total least squares algorithm for FIR adaptive filtering". In: *IEEE Transactions on Signal Processing* 42.2, pp. 268–280.

— (1998). "A subspace approach to estimation of autoregressive parameters from noisy measurements". In: *IEEE Transactions on Signal processing* 46.2, pp. 531–534.

— (2001). "On the noise-compensated Yule-Walker equations". In: *IEEE transactions on signal processing* 49.6, pp. 1119–1121.

Diversi, Roberto (2008). "A bias-compensated identification approach for noisy FIR models". In: *IEEE Signal Processing Letters* 15.11, pp. 325–328.

— (2009). "Noisy FIR identification as a quadratic eigenvalue problem". In: *IEEE Transactions on Signal Processing* 57.11, pp. 4563–4568.

Diversi, Roberto, Roberto Guidorzi, and Umberto Soverini (2005a). "A noise-compensated estimation scheme for AR processes". In: *Proceedings of the 44th IEEE Conference on Decision and Control*. IEEE, pp. 4146–4151.

— (2005b). "Blind identification and equalization of two–channel FIR systems in unbalanced noise environments". In: *Signal Processing* 85.1, pp. 215–225.

— (2008a). "A new approach for identifying noisy input-output FIR models". In: *2008 3rd International Symposium on Communications, Control and Signal Processing*, pp. 1548–1552.

— (2008b). "Identification of autoregressive models in the presence of additive noise". In: *International Journal of Adaptive Control and Signal Processing* 22.5, pp. 465–481.

Feng, D-Z and Wei Xing Zheng (2006). "Fast approximate inverse power iteration algorithm for adaptive total least-squares FIR filtering". In: *IEEE transactions on signal processing* 54.10, pp. 4032–4039.

Feng, Da-Zheng, Zheng Bao, and Li-Cheng Jiao (1998). "Total least mean squares algorithm". In: *IEEE Transactions on Signal Processing* 46.8, pp. 2122–2130.

Feng, Da-Zheng and Wei Xing Zheng (2007). "Bilinear equation method for unbiased identification of linear FIR systems in the presence of input and output noises". In: *Signal Processing* 87.5, pp. 1147–1155.

Friedlander, B (1984). "The overdetermined recursive instrumental variable method". In: *IEEE Transactions on Automatic Control* 29.4, pp. 353–356.

Gertler, Janos J (1988). "Survey of model-based failure detection and isolation in complex plants". In: *IEEE Control systems magazine* 8.6, pp. 3–11.

Goodwin, G C and K S Sin (1984). *Adaptive Filtering, Prediction and Control*. Prentice-Hall.

Gordon, Neil J, David J Salmond, and Adrian FM Smith (1993). "Novel approach to nonlinear/non-Gaussian Bayesian state estimation". In: *IEE proceedings F (radar and signal processing)*. Vol. 140. 2. IET, pp. 107–113.

Gouriveau, Rafael, Kamal Medjaher, and Noureddine Zerhouni (2016). *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. John Wiley & Sons.

Guidorzi, Roberto, Roberto Diversi, and Umberto Soverini (2008). "The Frisch scheme in algebraic and dynamic identification problems". In: *Kybernetika* 44.5, pp. 585–616.

Haykin, Simon (1991). *Adaptive Filter Theory*. Prentice-Hall.

Isermann, Rolf (2005). "Model-based fault-detection and diagnosis–status and applications". In: *Annual Reviews in control* 29.1, pp. 71–85.

— (2006). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media.

Itakura, Fumitada (1968). "Analysis synthesis telephony based on the maximum likelihood method". In: *The 6th international congress on acoustics, 1968*, pp. 280–292.

Jardine, Andrew KS, Daming Lin, and Dragan Banjevic (2006). "A review on machinery diagnostics and prognostics implementing condition-based maintenance". In: *Mechanical systems and signal processing* 20.7, pp. 1483–1510.

Jung, Sang Mok and PooGyeon Park (2017). "Stabilization of a bias-compensated normalized least-mean-square algorithm for noisy inputs". In: *IEEE Transactions on Signal Processing* 65.11, pp. 2949–2961.

Kalouptsidis, Nicholas (1997). *Signal processing systems: theory and design*. Wiley-Interscience.

Kang, ByungHoon and PooGyeon Park (2013). "An efficient line-search algorithm for unbiased recursive least-squares filtering with noisy inputs". In: *IEEE Signal Processing Letters* 20.7, pp. 693–696.

Kay, S (1979). "The effects of noise on the autoregressive spectral estimator". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 27.5, pp. 478–485.

— (1980). "Noise compensation for autoregressive spectral estimates". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.3, pp. 292–303.

Kay, Steven M (n.d.). *Modern spectral estimation*. Pearson Education India.

Lee, J, H Qiu, G Yu, and J Lin (2007). *Bearing Data Set, NASA Ames Prognostics Data Repository*.

Lee, Jay, M Ghaffari, and S Elmeligy (2011). "Self-maintenance and engineering immune systems: Towards smarter machines and manufacturing systems". In: *Annual Reviews in Control* 35.1, pp. 111–122.

Lee, Jay, Fangji Wu, Wenyu Zhao, Masoud Ghaffari, Linxia Liao, and David Siegel (2014). "Prognostics and health management design for rotary machinery systems – Reviews, methodology and applications". In: *Mechanical systems and signal processing* 42.1-2, pp. 314–334.

Ljung, Lennart (1999). *System identification: theory for the user*. Prentice-hall.

Mahmoudi, Alimorad and Mahmood Karimi (2011). "Inverse filtering based method for estimation of noisy autoregressive signals". In: *Signal Processing* 91.7, pp. 1659–1664.

Mechefske, CK and J Mathew (1992). "Fault detection and diagnosis in low speed rolling element bearings Part I : The use of parametric spectra". In: *Mechanical systems and signal processing* 6.4, pp. 297–307.

Mitchell, Tom M et al. (1997). "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37, pp. 870–877.

Mosallam, Ahmed, Kamal Medjaher, and Noureddine Zerhouni (2016). "Data-driven prognostic method based on Bayesian approaches for direct remaining useful life prediction". In: *Journal of Intelligent Manufacturing* 27.5, pp. 1037–1048.

Nandi, Subhasis, Hamid A Toliyat, and Xiaodong Li (2005). "Condition monitoring and fault diagnosis of electrical motors - A review". In: *IEEE transactions on energy conversion* 20.4, pp. 719–729.

NASA (2019). *NASA data repository for prognostic*. URL: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/.

Nectoux, Patrick, Rafael Gouriveau, Kamal Medjaher, Emmanuel Ramasso, Brigitte Chebel-Morello, Noureddine Zerhouni, and Christophe Varnier (2012). "PRONOSTIA: An experimental platform for bearings accelerated degradation tests." In: *IEEE International Conference on Prognostics and Health Management, PHM'12*. IEEE Catalog Number: CPF12PHM-CDR, pp. 1–8.

Nehorai, Arye and Petre Stoica (1988). "Adaptive algorithms for constrained ARMA signals in the presence of noise". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 36.8, pp. 1282–1291.

Rauber, Thomas W, Francisco de Assis Boldt, and Flávio Miguel Varejão (2015). "Heterogeneous feature models and feature selection applied to bearing fault diagnosis". In: *IEEE Transactions on Industrial Electronics* 62.1, pp. 637–646.

Saha, Bhaskar and Kai Goebel (2011). "Model adaptation for prognostics in a particle filtering framework". In: *International Journal of Prognostics and Health Management* 2, p. 61.

Saxena, Abhinav, Jose Celaya, Bhaskar Saha, Sankalita Saha, and Kai Goebel (2010). "Metrics for offline evaluation of prognostic performance". In: *International Journal of Prognostics and health management* 1.1, pp. 4–23.

Schafer, Ronald William and Alan V Oppenheim (1989). *Discrete-time signal processing*. Prentice Hall.

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Si, Xiao-Sheng, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou (2011). "Remaining useful life estimation–a review on the statistical data driven approaches". In: *European journal of operational research* 213.1, pp. 1–14.

Skima, Haithem, Kamal Medjaher, Christophe Varnier, Eugen Dedu, Julien Bourgeois, and Noureddine Zerhouni (2016). "Fault prognostics of micro-electro-mechanical systems using particle filtering". In: *IFAC-PapersOnLine* 49.28, pp. 226–231.

Söderström, Torsten (2018). *Errors-in-Variables Methods in System Identification*. Springer.

Söderström, Torsten and Petre Stoica (1981). "Comparison of some instrumental variable methods – Consistency and accuracy aspects". In: *Automatica* 17.1, pp. 101–115.

— (1989). *System Identification*. Prentice Hall.

Soualhi, Abdenour, Kamal Medjaher, and Noureddine Zerhouni (2014). "Bearing health monitoring based on Hilbert–Huang transform, support vector machine, and regression". In: *IEEE Transactions on Instrumentation and Measurement* 64.1, pp. 52–62.

Stoica, Petre and Torsten Söderström (1982). "Bias correction in least-squares identification". In: *International Journal of Control* 35.3, pp. 449–457.

Tulleken, Herbert JAF (1993). "Grey-box modelling and identification using physical knowledge and Bayesian techniques". In: *Automatica* 29.2, pp. 285–308.

Tulsyan, Aditya, R Bhushan Gopaluni, and Swanand R Khare (2016). "Particle filtering without tears: A primer for beginners". In: *Computers & Chemical Engineering* 95, pp. 130–145.

Vapnik, Vladimir N., ed. (1995). *The Nature of Statistical Learning Theory*. Springer.

Varga, Andreas (2017). "Solving fault diagnosis problems". In: *Studies in Systems, Decision and Control, 1st ed.; Springer International Publishing: Berlin, Germany*.

Vogl, Gregory W, Brian A Weiss, and Moneer Helu (2019). "A review of diagnostic and prognostic capabilities and best practices for manufacturing". In: *Journal of Intelligent Manufacturing* 30.1, pp. 79–95.

Wei, B. and J. Gibson (2000). "Comparison of distance measures in discrete spectral modeling". In: *Proc. of the 9th Digital Signal Processing Workshop*.

Xia, Youshen and Wei Xing Zheng (2015). "Novel parameter estimation of autoregressive signals in the presence of noise". In: *Automatica* 62, pp. 98–105.

Xu, G, H Liu, L Tong, and T Kailath (1995). "A least–squares approach to blind channel identification". In: *IEEE Transactions on Signal Processing* 43.12, pp. 2982–2993.

Zhang, Ran, Zhen Peng, Lifeng Wu, Beibei Yao, and Yong Guan (2017). "Fault diagnosis from raw sensor data using deep neural networks considering temporal coherence". In: *Sensors* 17.3, p. 549.

Zheng, Wei Xing (1999). "A least-squares based method for autoregressive signals in the presence of noise". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 46.1, pp. 81–85.

— (2000). "Autoregressive parameter estimation from noisy data". In: *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing* 47.1, pp. 71–75.

— (2003). "A least-squares based algorithm for FIR filtering with noisy data". In: *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. (ISCAS'03)*. Vol. 4, pp. IV–444–IV–447.

— (2005). "Fast identification of autoregressive signals from noisy observations". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 52.1, pp. 43–48.

— (2006). "On estimation of autoregressive signals in the presence of noise". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 53.12, pp. 1471–1475.