Politecnico di Torino
Department of Control and Computing Engineering

Institute for Engineering and Architecture

Thesis

For the Degree of

Master of Science in Mechatronic Engineering

End-effector tools State of Health estimation: a data driven approach

By

Davide Zanon

Submission Date: April, 2021

Mentor: Prof. Alessandro Rizzo
Supervisor: Ing. Giovanni Guida - Brain Technologies

Abstract

Nowadays, the focus on efficient and cost saving maintenance techniques is the goal of both research and industrial areas. It is clear that the advantages coming from preserving the continuity of the industrial line can make the difference with main competitors. The main technologies available in the literature and on the field that handle this topic make relevant use of machine learning algorithms, thus requiring high computational resources. On the contrary, the **MorePRO** project aims to introduce itself in this slice of the market, proposing an innovative **edge computing** device, able to carry out an on-line prediction of the State of Health of CNC machines. The main idea concerns the realization of a new product, that combines a consistent use of the **multi-model approach**, already refined by brain Technologies in the BAT-MAN and ERMES projects (based on some techniques such as Kalman filtering and residual error analysis), along with a data driven tool used to increase the robustness of the prediction. In this way, on the one hand, an effective and short-term estimate of usury is obtained, on the other, the possibility of refining the estimate itself in the long term. The main steps followed in this Thesis work are the building of the model under assumption, the application and testing of the methods coming from the previous projects and finally the implementation of the learning algorithms for the previously mentioned purposes.

Contents

List of Figures

List of Tables

Acronyms

SoH State of health

ERMES Extendible Range MultiModal Estimator Sensing

KF Kalman filter

EKF Extended Kalman filter

RT Reset time

CNC Computer Numerical Control

IOT Internet Of Things

RUL Remaining Useful Life

EOL End of Life

PM Predictive Maintenance

ANN Artificial Neural Network

HMM Hidden Markov Model

ML Machine Learing

PCA Principal Component Analysis

LDA Linear Discriminant Analysis

QDA Quadratic Discriminant Analysis

1 Introduction

1.1 Wear estimation

The estimation in real-time of the state of a production machine is one of the most significant topic in scientific research. The estimation of the SoH together with predictive maintenance techniques are slightly becoming a relevant issue because of their direct relation that link them to production efficiency. As Industry 4.0 continues to become reality, many companies are struggling with AI algorithms implementation that can lead to to major cost savings, higher predictability, and the increased availability of the systems. Indeed, the benefits of predictive strategies are definitely very strategic. Thus, the increasing demand of monitoring systems that allows to keep track of the production as much efficiently as possible have led to the development of many predictive maintenance methods [1]. The main functions of those algorithms are:

- SoH (state of health) estimation of a machine, motor or single component.
- Calculation of patterns that can help prediction and prevention of failures.

Currently, such methods are predominantly based on machine learning algorithms that lead to very good results in terms of efficiency and precision but they often doesn't take into account of the computational effort and real-time requirements. Nevertheless, predictive maintenance does not require anything more than mathematical computation on when machine conditions are at a state of needed repair or even replacement so that maintenance can be performed exactly when and how is most effective. Moreover, when the processing has high precision requirements, the predictive algorithms are particularly useful. Nowadays, those requirements are very common in companies which rest their production on such fields as aerospace, oil & gas, automotive and so on. The complexity of those high precision processes depends on many aspects:

- Kind of processing.
- Modelling and simulation of robotic, mechanical and electronic systems.
- Wrought materials.
- Different tools such as milling machines, cutting machines, end-effectors and so on.
- Production timings requirements.

Dealing with those complexity level can be very hard and expensive for companies, consequently, it is always more present the need of a method that can be easily applied regardless of the wrought material, the kind of processes and the field

of application. To sum up, the key functionalities of prediction algorithm are consistent when there is abstraction with respect to processing types, real-time characteristics and efficiency both in terms of computational effort and naturally, in terms of cost savings.

1.2 CNC machine SoH

CNC (Computer numerical control) [2] machine are high precision machines which actuate manufacturing processes of material substraction that usually require computerized control action to guarantee high precision and efficiency. A subtracting manufacturing process typically employs machine tools to remove layers of material from a stock piece known as the blank or workpiece and produces a custom-designed part. This processing type is almost independent from the material of which the workpiece is composed: plastics, metals, foam, glass etc.. This is the reason why CNC machines finds application in most of industrial processing fields.

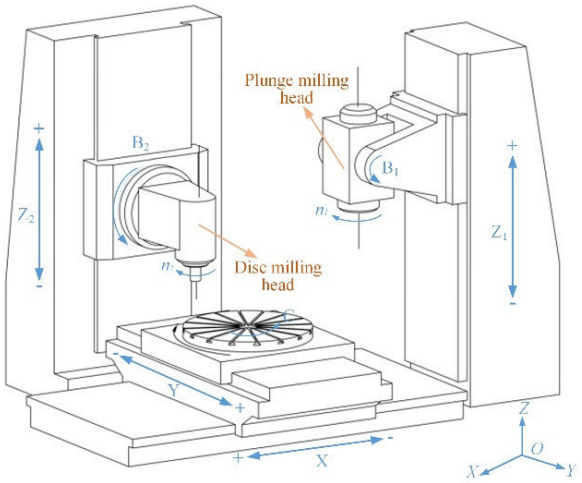


Figure 1.1. Example of CNC machine schematic diagram

As it is shown in figure 1.1 this machines have typically a SCARA or a cartesian robotic configuration with an end-effector which usually is a cutter. The modelling of the cutter contact is quite difficult because of the high number of variables that must be considered, the most relevant are:

- Robotic configuration.
- Environmental parameters.
- Wear condition of the machine: SoH.

All of those elements needs to be kept under control constantly in order to guarantee the efficiency and the precision of the machine. In particular, there is no

way to check the State of health of the end-effector in a direct way. It could be possible to install sensor to check temperature, voltage, pressure and estimate a possible SoH of the tools. However, even with the knowledge of variables that can be measured by sensors, it is difficult to extract information about actual condition of the machine, firstly because is very likely that sensors cannot be set up in the right position, secondly because the knowledge of those parameters could not be enough to understand the real condition. For instance, obviously, a temperature sensor cannot be positioned near enough to the cutter to measure the correct temperature but must be positioned further, and that definitely lead to constant and inevitable measurement errors. Nowadays, the majority of the systems which estimate the SoH commonly propose digital-twin solutions. The limitation of such system is the difficulty to isolate the tool's wear from the others monitored effects. Moreover, such monitoring systems combine machine learning techniques and digital-twin simulation to estimate SoH, not taking into account computational requirements. Digital-twin models are very useful when the variables that need to be controlled are numerous, but the most influent parameters in the SoH estimation are the ones related to the cutting process of the end-effector: the most stressed mechanical elements. Therefore, the parameters which are directly linked to the SoH are a lot and some of the most important are:

- Friction coefficients.
- Temperature.
- Chip load.

Those elements are strictly related to the contact forces, that's why understanding and modeling those element is fundamental for the estimation of the state of health of CNC machine end-effector.

1.3 Edge Computing advantages

As it was mentioned in the introduction, most of the existing architectures regarding the wear estimation and the predictive maintenance entrust the majority of their computational power in the cloud. Since to execute deep machine learning calculations there is the necessity of hardware resources, they have no choice but to rely on cloud computing solution. However, a centralized server, even if geographically far, can be definitely useful because of the potentially infinite number of resources that can be accessed and also, because of the huge data storage capacity available in the servers. On the other side, Edge Computing is an IT distributed architecture which allows to elaborate data locally, as much close as possible to the source. It is based on distributed calculation concept, which relies its principles in the separation of the code execution and in the storage of data only when it is strictly necessary. This solution compensate some of the cloud computing shortcomings and provide some advantages:

- Low-latency: Edge computing devices are installed locally and compensate latency that prevent the execution in real-time.
- Costs: Since hardware requirements are very low, the costs of those microprocessors is almost insignificant.
- Reliability and Security: Since most of the times the edge computing does not depend on internet connection and servers it offers an uninterruptible service. Users do not need to worry about network failures or slow internet connections.
- Scalability: Updates and modification on a cloud computing architecture can be very expensive. Edge computing do not require a data center to store data and it is easy to add and remove devices from the network architecture.

For sure, the limitation that characterize an edge computing device are strong constraints and developing a system which is comparable in performance with the powerful machine learning tools can be rather challenging, but definitely it is a way that it is worth to study.

1.4 MorePRO project

Considering all the thematic exposed above, the MorePRO project wants to bring on the field a new and innovative proposal, which is not present in any production system nowadays. It is basically based on a logic architecture distributed in three different levels:

- Monitoring of the SoH of machine and plant critical components through embedded sensors and, consequently, applying machine learning and data mining techniques.
- Keeping track of the SoH of the machine using digital twins tools. The goal is to combine real-time environment signals along with some estimated quantities in a specific simulation environment.
- Developing of forecast models, able to estimate the SoH of the machine and the time evolution decay of the plant/machine.

The general development architecture can be subdivided in two main levels. A first **field level** (Edge), where signals will be acquired and processed for a local supervision of the SoH. This is extremely useful to have a rapid reaction when any danger anomaly is detected. The same signals are then deployed to a second **server level**, mainly located on the cloud, which will be able to set up a proper bank of data, implement *digital twin* techniques and compute the right parameters to reconfigure the elaboration logic of every single edge device. The crucial part is the continuous inter-operability between the two levels and the possibility to reconfigure the architecture *on the fly* depending on the case problem. The figure below represents a general scheme on which the project will based on.

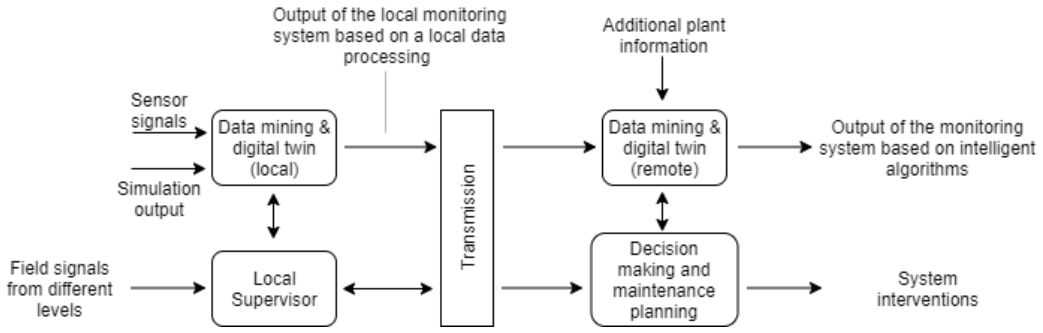


Figure 1.2. Synthetic structure of MorePRO system

With reference to the figure, the *edge device* will implement the data-mining, digital-twin and monitoring algorithm allowing the bidirectional data exchange with both the plant and the supervisor. In practice, it will process the real signal coming from the field along with the simulation output in order to compute a SoH of the considered element under monitoring. On the other side, the supervisor will be able to adjourn and perfection the algorithm of the device itself in order to re-configure and support the planning decisions. A possible physical implementation can be seen in the next figure.

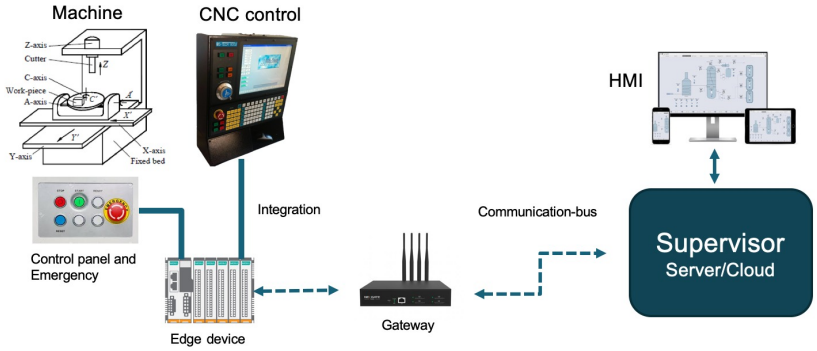


Figure 1.3. Overall system structure

1.4.1 Partnership

To the aim of this project several companies are involved. Thus, it is relevant to see how each of them is involved in the work, in order to understand how a development process is usually treated when a completely new and innovative device must be designed.

- **brain Technologies**: it will in particular contribute to the definition and design of the digital architecture (in collaboration with the other partners) and to the software development of the distributed intelligence system proposed by the project, including the architecture of the control supervisors

whose action is propagated both in the devices and in the cloud. brain Technologies will also contribute to the implementation, testing and validation phases of the final prototype.

- **MCM S.p.A**: it will contribute to the analysis of user needs and to the proposal of new or improved functionalities of the processing systems as drivers for the development of the monitoring and predictive management methods of the plants covered by the project. Other activities in which MCM will play an active role include: 1) interfacing the machines for data collection, also through the installation of new sensors; 2) supporting the integration of the new MorePRO solutions with the plant supervisor software, 3) analysis and testing of the prototype system with its validation at the production unit of CAMS, a partner in the project.
- **AL.MEC**: it will contribute to the design and manufacture of electronic boards and components necessary for data collection from machines and sensors, their mash-up and processing on board the machine and sending standardised information to predictive maintenance systems.
- **CAMS**: it is participating in the project by contributing its vision and expertise as a user of highly flexible production lines for the manufacture of complex, high value-added parts. CAMS will support in particular 1) the first phase of definition and analysis of the requirements that will guide the subsequent development of the new plant monitoring and predictive management solutions, 2) the identification and definition of its cases of industrial interest, 3) the implementation, testing and validation of the final prototype in its own production lines equipped with flexible MCM systems.

1.5 Work Organization

MorePRO project is starting out in September 2020, and the work must be organized in order to start the development as fast as possible. In this situation model-based software design can be very suitable. Model-based approaches recommend to follow precise development procedures, the so called V-shaped represents a process to be chased in order to guarantee efficiency and cost-effectiveness during such project natural advancement.

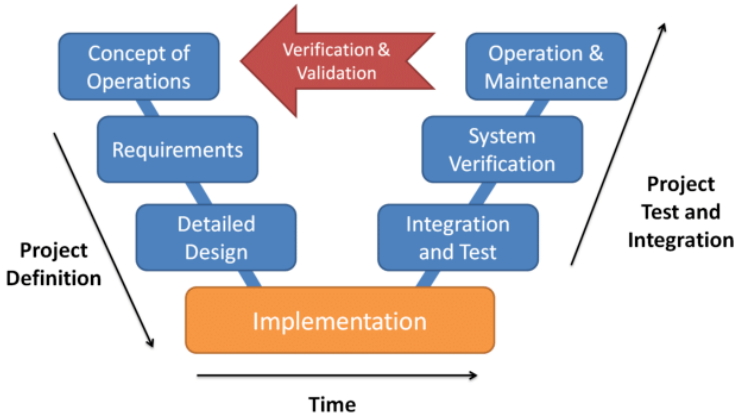


Figure 1.4. V-shape development flow.

During the first phases of the realization of scientific projects such as MorePRO, model-based approaches as the one shown in Figure 1.4 are necessary for the organization of the work. This become even more true as much as the number of people that join the project increases. Therefore, in the following paragraphs there will be a brief introduction to the team components and their aim in the V-shaped development, followed by a presentation of the work flows and the aims of the project team.

1.5.1 MorePRO team

From the collaboration between Politecnico di Torino and brain Technologies srl it is aroused a team of graduating students supervised by Giovanni Guida, Innovation Manager of brain Technologies, with the aim of developing the first phases of the project. As it was previously mentioned, the project is only at the first stage, so once it is defined the concept of operations, the aim of this team is to obtain a first implementation after the first six months of work. Despite the development flow suggests to focus first on the requirements and analysis, it has been decided to employ one member of the team to do a requirement analysis, two members working on a detailed modelling of the problem, and the three remaining members working on the core implementation. This choice comes from the necessity to get a fulfilling conclusion satisfying all time-requirements.
Therefore, there are three different sub-teams:

1. Prediction team: This team will focus the attention on the prediction analysis and parameter estimation. After the development of a simple model, the aim becomes to deeply study parameter identification through kalman-filters and residual error analysis techniques. Estimation of wear and SOH of a CNC machine is the main objective, to get to this, multi-model approach will be implemented and tested in detail, using simulating environment such as MATLAB and Simulink.

2. Modelling team: This team is created to obtain a preliminary detailed modelling of the kinematics and dynamics of a CNC Machine as first. Secondly, the main objective is to study and specify the interaction between end-effector and workpiece.
3. Requirements team: This team is in charge to carry out an overall view of the project, analyzing requirements and specifics for each part of the project. Finally, another important role of this team is to develop a design of experiment in order to opportunely test the functionalities individually and together.

After the first three months of work, the team sub-division is not valid anymore because each team-component will be focused on developing further features that are explained in detail in 1.5.3 paragraph.

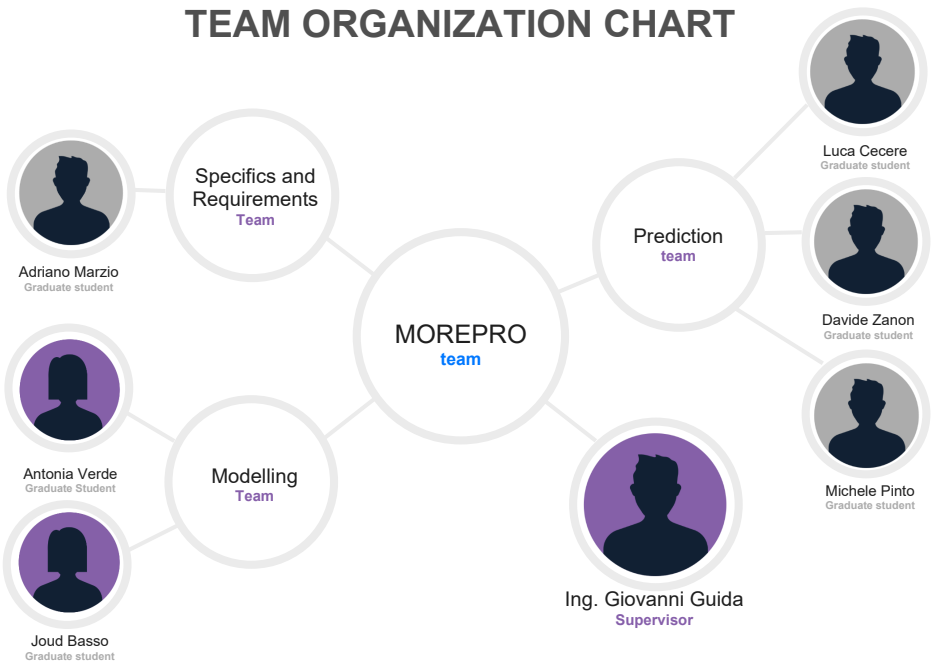


Figure 1.5. Team organization chart.

1.5.2 Work flow

An organization of the work flow is fundamental to help streamline and automate repeatable tasks, minimizing room for errors and increasing overall efficiency. The MorePRO project work flow can be synthesized in the following schematic:

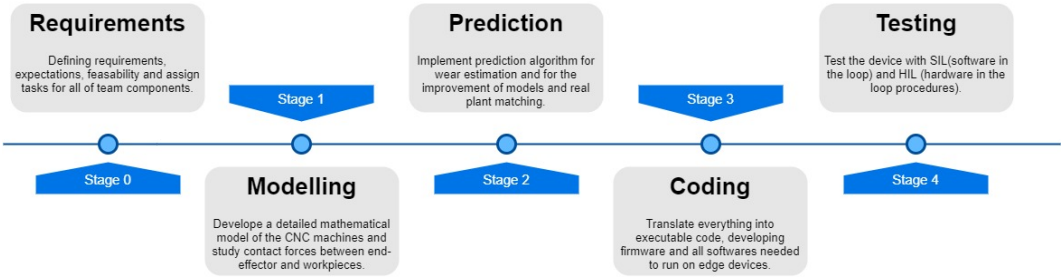


Figure 1.6. Workflow schematic.

As it was explained in the previous paragraph, the tasks have been assigned to be executed in parallel, however they are meant to be put together. Nevertheless, it is important to keep in mind a clear idea of the pre-determined work-flow.
For what regards this thesis, the work-flow is well defined and it is listed as follows:

1. State of art analysis: Comparison with all the current techniques present in the literature about parameter estimation and predictive maintenance.
2. Development of simple model for simulation scopes: In this stage the aim is to build a simulation suitable model that has to be the base for the prediction algorithm. This model is intended to be very simple in terms of physics and mathematical modelling.The idea is to test the algorithm in a simple environment as first, and then complicate the modelling until the real plant testing is ready.
3. Multi-model approach: This phase is the most crucial because it starts the actual development of the prediction algorithm. The scope of this stage is to apply the multi-model approach(only applied for Battery management systems until now) for the estimation of SoH of the end-effector.
4. Specific features: As last stage but not least, there is the implementation of specific improvements to the estimation algorithm that include machine learning techniques, reliability calculation and modelling betterment.

1.5.3 Objective of this Thesis

As part of the prediction team, the main goal of this Thesis work is to introduce data-driven techniques to support the estimation algorithm of the edge device previously mentioned. In particular, after the development of a common part carried out with other two students about the implementation of a multi-model approach on a previously designed physical model of a CNC machine, the main focus falls on the exploration of the Machine Learning world. Roughly speaking, two different types of approaches will be addressed. The first in which on the sole basis of the data coming from the sensors, an algorithm will be built that performs the same work of the edge device, i.e. estimating of the wear of the machine. In the second one, also considering the labels in output from the devices mounted on the

machines, the aim is to eliminate the small parametric variations present between different plants, obtaining a more reliable estimate.
For the first part, **Self-Organizing maps** (SOM) are used as clustering algorithms, which are essentially neural networks that allow to transform the data set into a topology-preserving 2D map. Basically, a SOM consists of a competitive layer which can classify a data set of vectors with any number of dimensions into as many classes as the layer has neurons. The neurons are arranged in a 2D topology, which allows the layer to form a representation of the distribution and a two-dimensional approximation of the topology of the data set.
For the second part, the use of **k-Nearest Neighbors** (kNN) is exploited, which basically is a classification algorithm based on the characteristics of the objects close to the one considered. The analysis is carried out considering data collected from five different machines and the responses coming from the edge-devices of each one. For this kind of approach, both cross-validation and principal component analysis are used, since they bring relevant improvements to the algorithm.

1.6 Thesis outline

The contents of this Thesis are organized as presented in the following.
In Chapter 1, an introduction related to the problem of wear estimation is reported with particular emphasis to edge computing devices and their advantages. This is followed by a brief explanation of the entire project and the main roles of each group in its realization.
In Chapter 2, an in-depth study of the state of the art literature about topics of interest is made. In particular, after a brief presentation of all the kind of industrial maintenance the problem of the State of Health estimation is analysed through the main known techniques. Finally, a summary table of all the studied methods in the preliminary phase is proposed, focusing on their potential advantages and disadvantages.
In Chapter 3, the design of a simple physical model of the plant under study is made, both for the mechanical and electrical part. Moreover, the Simulink implementation of the same is made along with a brief analysis of which parameter affects mostly the wear condition of the model.
In Chapter 4, the implementation and testing of what will be the core element of the edge device is made. Thus, the application of a Multi-model approach, based on an Extended Kalman Filters bank. First of all, a residual error analysis is carried out in order to find which parameters influence in the most significant way the results, introducing the concept of integral error reset. As final part, an accurate switching logic development for the choice of the Best Model is made.
In Chapter 5, in order to improve the efficiency of the prediction algorithm a parallel data-driven analysis is carried out. Indeed, the focus of this part is to rely on the Cloud, which must be responsible to classify/cluster the data collected from the plant through suitable machine learning techniques. In particular, two different scenarios are presented, a first one where just considering the sensors data

coming from the machines a clustering operation between different wear classes is made using Self-Organizing Maps, and a second one where, making use also of the labels that the edge device produce, a classification is carried out using a K-NN algorithm with the main goal to leverage the possible estimation errors of the various devices available on the field. Concluding remarks obtained from this Thesis will be drawn in Chapter 6. Moreover, promising new research lines that emerge from this work will be outlined as well. Additionally, this Thesis includes one appendix on boxplots that will be helpful to better understand and clarify the main analysis carried out in Chapter 4.

2 State of art

System reliability is one of the main issues in the nowadays industry, thus the development of advanced system maintenance techniques is an emerging field based on the information collected through system or component monitoring (or system state estimation) and equipment failure prognostics (or system state forecasting). According to the standard EN 13306 (2001), such techniques can be grouped into two main categories. The first one is **corrective maintenance** and it consists of replacing the component and repairing the damage after some major breakdown. This kind of approach is used when the consequences of a failure are not so critical and the intervention on the field does not require a lot of costs and time. In particular, we refer to *palliative maintenance* when the repair is provisional, and *curative maintenance* when it is definitive. The second one is **preventive maintenance** and it refers to provide an alarm before faults reach critical levels so as to prevent system performance degradation, malfunction, or even catastrophic failures. When the maintenance intervention is time-based, meaning that the components are replaced based on a predefined schedule which relies on the working hours of the component, it is referred as *predetermined maintenance*. Obviously, this approach is not optimal, since the components are being replaced before the end of their lives, therefore increasing the costs.
A possible solution is to use *condition-based maintenance*, which refer to the analysis of real-time data in order to find in the change of their characteristic a possible failure. However, this approach do not guarantee to design a maintenance policy with certainty. On the contrary, *predictive maintenance* try to estimate the SoH of the machine, relying on more dynamic algorithms. [6]

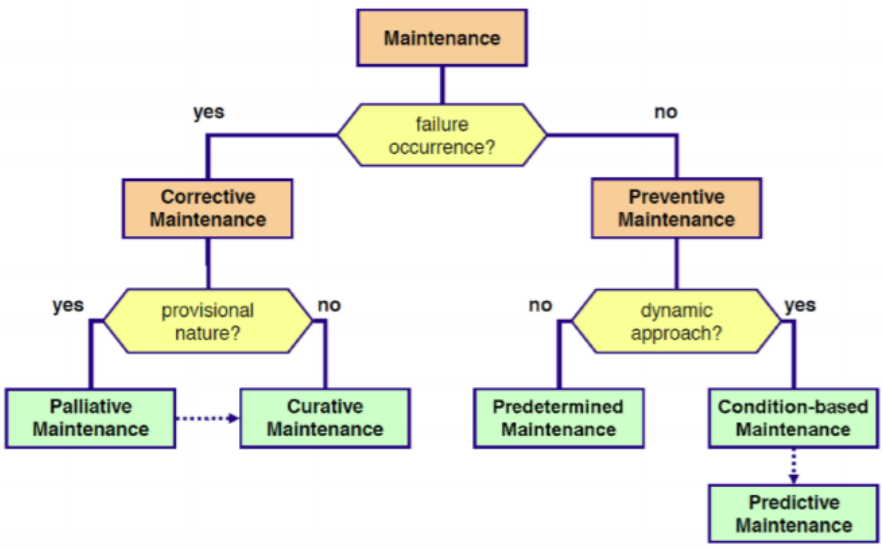


Figure 2.1. Forms of maintenance

2.1 Methods for estimating SoH

1. **Model-based approach**
   This approach makes use of physical failure model in order to predict the degradation rate of a component or its lifetime. In practice, a mathematical model able to capture the failure mechanism must be developed. It seems obvious that the more accurate and sophisticated the model is, the more precise will be the SoH estimate of the machine under control. However, it is not always possible to obtain a model that perfectly adhers to the reality, that is why a trade-off between a very precise model and an estimate that allows to hide the lack of knowledge of the plant is needed. Usually, this approach follows some prefixed steps:

   – **Critical part selection**: it is important, especially in very complex plant, to focus the study only on the part that actually contribute to the lifetime duration of the machine.

   – **Failure mechanism determination and model definition**: intuitively, this is the most difficult part, where a suitable model must be designed in order to capture the most relevant aspects.

   – **Governing loads evaluation**: it is important to understand which loads affects most the failure and how they are related to the operational usage of the system.

   – **Data collection**: once the model is defined, it is possible to collect data from the field.

   – **Failure prediction**: combining the monitored data with those one coming from the model it is possible to have an actual estimation of the health of the plant.

   – **Model validation**: finally, it is possible to determine how the model is reliable by comparing the failure prediction with actual failure data. [7]

   In particular, having a view at the models available in literature, we can distinguish between different kind of models:

   - **Electromechanical models**: in this case we have models that describe the behavior of the plant by means of equations that link macroscopic parameters such as forces, currents, torques, etc. This approach results to be very accurate but at the same type they are very time-consuming in terms of computation.

   - **Mathematical models**: these are based on the calculation of coefficients of linear and non-linear mathematical functions, needed to interpolate the data obtained experimentally through the measurement of some relevant quantities. The negative aspect is that these functions

result not to connect in a natural way the physical quantities between them, often finding relationships that have no real link with the actual dynamic of the plant. [16]

2. **State observer**
State observer is a very popular approach to system maintenance. For linear systems with additive Gaussian noise terms, KF can be used for prediction. However, when dealing with nonlinear systems with additive Gaussian noise terms EKF are more suitable. For nonlinear systems with non-Gaussian noise terms, the PF also called sequential Monte Carlo method, which are based on the sequential importance sampling (SIS) and the Bayesian theory, lead to a suboptimal solution to state estimation problem [8].

- ***KF*** is an established technology for dynamic system state estimation that is mostly used in many fields including: target tracking, global positioning, dynamic systems control, navigation, and communication. The KF covers a set of recursive equations that are repeatedly evaluated as the system operates [9]. Any causal dynamic system generates its outputs as some function of the past and present inputs. It is often also convenient to think of the system having a "state" vector (which may not be directly measurable such as the SoH of a machine) where the state takes into account the effect of all past inputs on the system. Present system output may be computed with present input and present state only, past inputs do not need to be stored. The KF can be viewed macroscopically in this way:

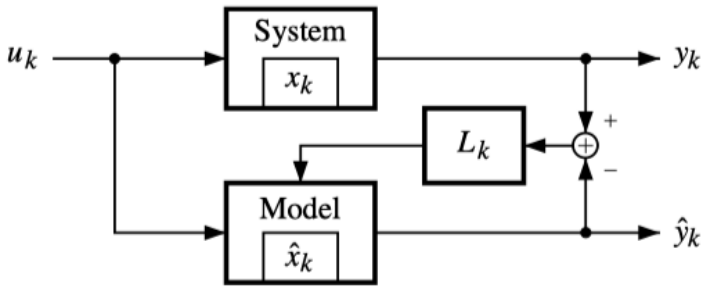


Figure 2.2. Schematic of state update

The true system has a measured input u_k and a measured output y_k . It also has an unmeasured internal state x_k . A model of the system runs in parallel with the true system, simulating its performance. This model has the same input u_k and has output $\hat{y}_k$. It also has internal state $\hat{x}_k$, which has known value as it is part of the model simulation. The true system output is compared with the model output, and the difference is an output error, or innovation. This innovation is converted

to a vector value by multiplying with the Kalman gain L_k, and used to adapt the model state $\hat{x}_k$ to more closely approximate the true system's state. The state estimate and uncertainty estimates are updated through computationally efficient recursive relationships.

- ***EKF*** (Extended Kalman Filter) is used in order to deal with nonlinear systems. In practice, it is based on a linearization of the system such that is possible to treat it as a linear time-variant (LVT). Since this algorithm will be widely used during the Thesis work it will be introduced and discussed more in detail in the next phases.

- ***Particle filters*** are nonlinear state observers that approximate the posterior state distribution using the set of weighted spots, called particles. The particles consist of samples from the states-space and a set of weights which represent discrete probability masses. A better estimate can be obtained by increasing the number of particles. Particle filtering has a wide applicability in fault prediction because of the simple implementation. The algorithm consists of two steps: the first one is state estimation, and the second one is long-term prediction. The state estimation involves estimating the current fault dimensions and changing parameters in the environment. The next step is the state prediction, which uses the current fault dimension estimate and the fault growth model, to generate state prediction from $(\tau + 1)$ to $(\tau + \mathrm{p})$. Once the long-term prediction is estimated, given the lower and upper bounds of a failure zone (H_{lb} and H_{ub}), the prognosis confidence interval can be estimated.

3. **Vibration monitoring**
   VM is a particular way of analyze the SoH of a machine by using, as obvious, vibrations as an indicator. This technique is particularly used because vibrations bring an high content of information, in the sense that a possible damage is almost instantaneously captured by them. However, vibration-based monitoring applications focus more on *diagnostic* aspects than predicting ones. Nevertheless, in some cases this method can be used and useful for making a prognosis of the system. Thus, looking at the PF-curve in the figure, it is possible to distinguish between a first part on the left, where after a certain time of inspection a point of deterioration observability (P) is used for monitoring purpose, and a second part on the right, where the objective is to predict the behavior of the curve till the failure, used for prognosis.[7, 10]

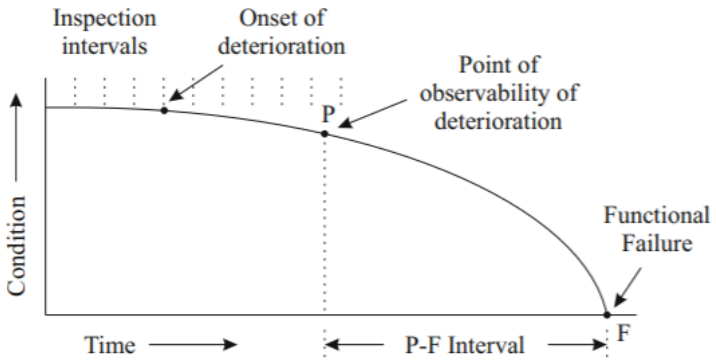


Figure 2.3. P-F curve

4. **Moving Horizon Estimation**
   MHE is a powerful technique for facing the estimation problems of the state of dynamic systems in the presence of constraints, nonlinearities and disturbances [11]. MHE is an optimization approach that uses a series of measurements observed over time, containing noise (random variations) and other imprecisions and produces estimates of unknown variables or parameters. It requires an iterative method that relies on linear programming or nonlinear programming solvers to find a solution. The basic concept is to minimize an estimation cost function defined on a moving window composed of a finite number of time stages. The cost function includes the usual output error computed on the basis of the most recent measurements and a term that penalizes the distance of the current estimated state from its prediction (both computed at the beginning of the moving window).

5. **Learning algorithm**
   These techniques use measurement signals and their statistics to create nonlinear structures which can provide desirable outcomes given the input data. These structures include a wide range of methods, such as principal component analysis (PCA), partial least squares (PLS), artificial neural networks, fuzzy-logic systems and graphical models like hidden Markov models (HMM).

   - ***ANN*** propose methodologies similar to those in the biological nervous system. For a set of available monitoring data which are used as inputs and predefined known outputs it is possible to use some of the training algorithms, such as back-propagation algorithm, to map the connection between the input and output. Neural networks are self-adaptive structures whose weights between neurons are adjusted by minimizing the criteria to match a model to desired outputs. The training procedure allows the network to learn the relationship among the data without engaging the model of the system. Once the weights are set, the ANN is ready to generate the desired output as a fault evolution prediction.

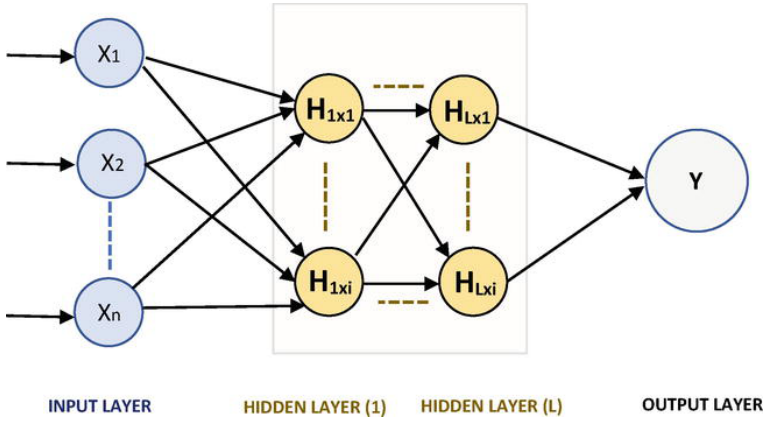


Figure 2.4. Schematic of a Artificial Neural Network

- ***Fuzzy logic*** also provide mapping between the input and output signals. It can be said to be an extension of the multi-value logic. In a wider sense, is almost synonymous with the theory of Fuzzy sets, referring to classes of objects with fuzzy boundaries, in which the concept of membership takes on a matter of degree [12]. Unlike neural networks, they are based on linguistic and reasoning human capabilities. By defining the appropriate if-then rules and adjusting membership functions, fuzzy systems can give very accurate prognosis.

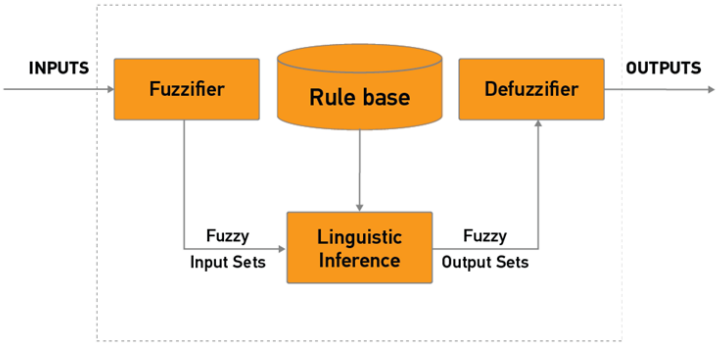


Figure 2.5. Components of a Fuzzy logic system

The common fuzzy logic system processes data in three sequential stages: fuzzification, inference and defuzzification. In the fuzzification step, a crisp, or well-defined, set of input data is gathered and converted to a fuzzy set using fuzzy linguistic variables that is, fuzzy linguistic terms. Second, an inference is made based on a set of rules. Last, in the defuzzification step the resulting output is mapped using so-called membership functions. A membership function is a curve that maps how each point in the input space is related to a membership grade. Using the wear estimation example, various levels of wear in a given set would receive

a membership grade between 0 and 1; the resulting curve would not define "new" but instead would trace the transition from worn to new [13].

- ***Hidden Markov Models*** is a statistical model which can be used to describe system transitions between states. It represents an extension of a regular Markov chain with unobservable or partially observable states. The general structure of a discrete-time HMM with N states, $S = (s_1, s_2, \ldots, s_N)$ and M observation symbols, $V = (v_1, v_2, \ldots, v_M)$ is shown in the schematic below.

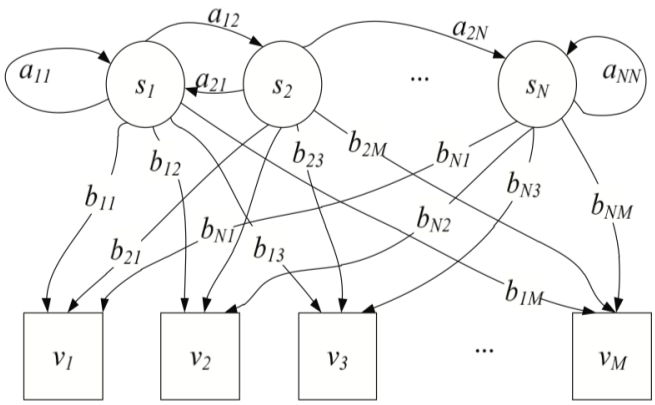


Figure 2.6. Schematic of HMM

The states are interconnected so that a transition between any two states is possible. The hidden state at time t is denoted as q_t and the state-transition rule follows the Markov property, meaning that the state q_t depends only on the state q_{t-1}. The transition matrix $A = \{a_{ij}\}$ stores the probability of state j following state i. The observation matrix $B = \{b_j(k)\}$ shows the probability of observation k being produced from the j-th state. The initial state array $\pi = \{\pi_i\}$ holds the information about initial probabilities; thus, the formulation of HMM is: $\lambda = (A, B, \pi)$.
HMMs can be used to estimate the occurrence of a breakdown, before it happens. Using the Baum-Welch algorithm, HMM can be trained in order to give desired outputs related to system health, for the monitored data inputs. HMM offer a reasonable estimation of the RUL time, meaning the time when the system will be in the specified, faulty state. Also, it is possible to estimate the probability of system being in specified state after n iterations.

6. **Frequency domain condition indicators**
Another possibility regards the analysis of frequency domain indicators. This kind of research was fundamental for the whole thesis streamline, because

it is the base of information extrapolation from signals. A deep study can be done about all frequency domain indicators but "Developing a real-time data-driven battery health diagnosis method, using time and frequency domain condition indicators" [14] perfectly sum up the main features in brief. This article is about battery health diagnosis, but the main principles can be applied also in the study-case of this thesis. The flow diagram of the construction of condition indicators which is used in the study is depicted in Figure 2.7

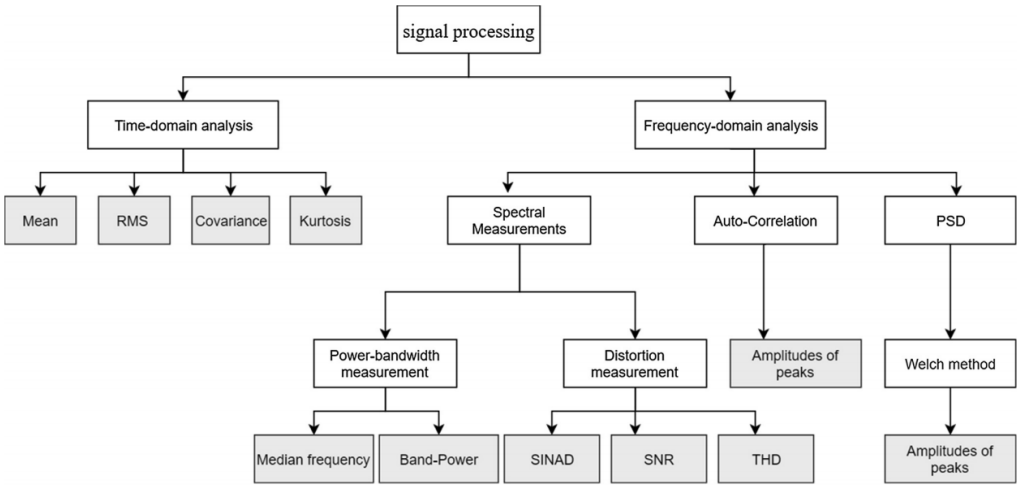


Figure 2.7. Flow diagram of condition indicator construction. The gray boxes indicate the condition indicators.[14]

7. **Hybrid algorithm**
   In the literature it is possible to find some methods that make use of some of the theories exposed so far in order to increase the estimation quality of the SoH. These is done in order to overcome the limitations of a single approach. It will be seen that also in this Thesis work a mixed/hybrid approach will be carried out, using some of techniques exposed above, such as EKF, multimodal analysis and ML.

2.1.1 Past project references

This project research, and the whole thesis work, is part of a continuing evolving series of projects handled by brain Technologies srl. Since the origin of MorePRO comes from the evolution of some ideas developed in the previous projects, it is necessary to have a preparatory overview of the ideas and the principles make up the past projects.
The projects that precede this work are:

1. The **BAT-MAN** research and development, which is an industrial project owned by brain Technologies and it is the starting point of the application

of the innovative approach based on an EKF bank and whose main goal is the realization of an electronic device capable of detecting and forecasting, in real-time, the working conditions of a Lead-Acid battery.

2. The **ERMES** (Extendible Range MultiModal Estimator Sensing), which is an algorithm designed by Brain Technologies whose innovative value is to identify the methodologies to apply to the problem of the diagnosis of an accumulation system, and in particular to the problem related to the estimation of the SoH of batteries. The proposed ERMES algorithm for the estimate the state of health (SoH) and the state of charge (SoC) is based on the model with the augmented state, which means to consider the uncertain parameters related to SoH and SoC as states and not simply as output. The algorithm involves the generation of a battery model based on an equivalent circuit and a bank of N EKF (Extended Kalman Filter) each based on a different SoH hypothesis. Since this approach is very similar to the one adopted in this thesis, a more detailed explanation of the multi-model and the residual error analysis approach is available in the dedicated chapter of this thesis (chapter 4, reference related to this project is *Virtual Sensing for the Estimation of the State of Health of batteries* [15]).

2.1.2 Comparison between the methods

	Advantages	Disadvantages
Model-Based	High reliable results when the model is accurate	High computational effort
Kalman Filters	High accuracy and online estimation	High calibration and strong hypotesis on the model
Particle Filters	Ease implementation, ability to cope with large scale system	Strong sample size dependence
Vibration Monitoring	Speed of fault detection	Hardly suitable for prognosis scope
Moving Horizon	High noise filtering	Very high computational effort, not able to cope with high dynamics
Learning algorithm	High accuracy and estimation	Need of an huge set of data
Hybrid algorithm	Online estimation, correction of disadvantages of other methods	Strongly depends on the model precision

Table 2.1. Comparison between SoH estimation methods.

3 Physical Model

Mathematical modeling is the art of translating problems from an application area into tractable mathematical formulations whose theoretical and numerical analysis provides insight, answers, and guidance useful for the originating application [16]. Nevertheless, the modeling of a CNC machine can be a very challenging objective, this is due to the complexity and the high number of elements that those technologic tools can achieve.

Starting from a blank sheet, the general idea beside this Thesis work is to develop a model able to represent in the most effective way the real condition of the plant under study. However, considering the high complexity of a CNC machine, it has been decided to start from a basic model in order to allow an embryonic prediction algorithm as soon as possible and obtain some effective results from a simple simulation environment. The main objective of the simulation is to understand and emulate the behaviour of a particular manufacturing system on a computer prior to physical production, thus reducing the amount of testing and experiments on the shop floor. By using a virtual system, less material is wasted and interruptions in the operation of an actual machine on the workplace can be avoided. The goal of the modern manufacturing technologies is to produce already the first part correctly in the shortest period of time and in the most cost effective way. Since the product complexities increase and the competitive product life cycle times are reduced, the construction and testing of physical prototypes become major bottlenecks to the successful and economically advantageous production of modern machine tools [17]. It is clear that, in this way, it is possible to discriminate better which parameter/quantity mostly affects the case under assumption. In a second moment, it will be up to the modelling team to further complicate the model in order to have a better adherence to the real case.
As for all mechatronic devices, it is possible to distinguish between a mechanical and an elctrical part, parts that are not independent but they work together to exploit the necessary tasks.

3.1 Mechanical part

As regards the mechanical part, a very simple model of a milling machine is used. In particular, having a look at the figure below, a rotational disc is considered that translates in the piece direction in order to cut it.

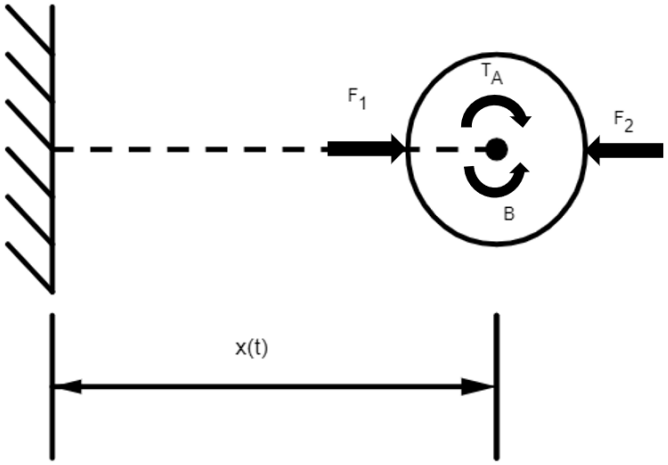


Figure 3.1. Simplified milling machine model

The state equations of the systems are obtained through a classical Newton formulation. Since the schematic is very simple, defining the various quantities:

- $\dot{\theta}$: Rotational velocity.
- $\dot{x}$: Linear velocity.
- F_1: Horizontal force that moves the cutter.
- F_2: Normal Force due to contact.
- f_c: Binary function that defines the presence of contact. Indeed, it assumes 1 value when the work piece is present or 0 otherwise.
- T_a: DC motor torque applied to the cutter.
- I_n: Inertia of the motor and the cutter.
- β: Contact rotational friction.
- Δ_x: Depth of cutting.
- *cost*: Minimum contact force (introduced in order to avoid model discontinuities).

it is very easy to trace the two Newton equations:

$$\begin{cases} \ddot{\theta} = \dfrac{T_a - \beta\dot{\theta}F_c}{I_n} \\ \ddot{x} = \dfrac{F_1 - f_c(F_2\Delta_x + cost)}{m} \end{cases}$$

3.2 Electrical part

For the electrical part instead, modern CNC machines are driven by brush-less or servo motors. The most important characteristics required for the servo motors that drive CNC machines are: fast response to instructions, good acceleration and deceleration properties, the capability to control velocity safely in all velocity ranges and to control very precise the position [18]. Machines with computer numerical control need controllers with high resolution that gives good precision. At this time, both classical and modern control techniques are used, such as PID controllers, feedback control, feedforward control, adaptive control or auto tuning methods.
In order to get a basic framework easy to manage, a DC motor is implemented to drive and interface with the mechanical part. The figure 3.2 shows a simplified DC motor circuit used to pull out the electrical equations.

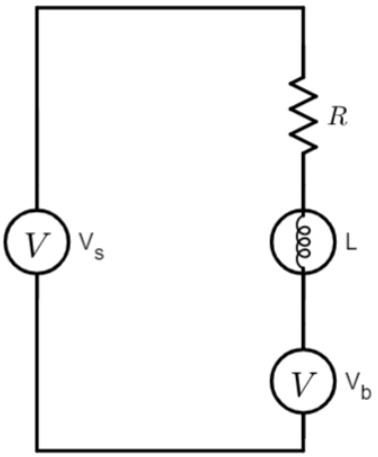


Figure 3.2. Simplified schematic of a DC motor

Defining the following quantities:

- V_s: supply Voltage.
- i_a: Armature current.
- T_a: DC motor torque applied to the cutter.
- k_t: Motor torque proportionality constant.
- L: Inductance.
- R: Resistance.
- V_b: Back E.M.F
- Att_{mot}: Engine friction.

- k: Proportionality constant.
- b: Total flux
- I_L: Motor inertia.

it is possible to derive the equations for the supply Voltage and the Torque applied to the cutter.

$$\begin{cases} V_s = Ri_a(t) + L\dfrac{di_a(t)}{dt} \\ V_b = kb\dot{\theta} \\ T_a = k_t i_a(t) - Att_{mot}\dot{\theta} \\ T_a = I_L\dot{\theta} \end{cases}$$

Thus, playing a little bit with the equations:

$$k_t i_a(t) - Att_{mot}\dot{\theta} = I_L\dot{\theta}$$
$$Vs = Ri_a(t) + L\frac{di_a(t)}{dt} + k_b\dot{\theta}$$

Since the supply Voltage is related to the angular velocity through the equation:

$$V_s = \frac{T_a}{k}R + k\omega$$

rearranged for angular velocity:

$$\omega = \frac{V_s}{k} - \frac{T_a}{k^2}R$$

Thus, it is possible to notice that two main variables affect the speed of the motor: the supply Voltage and the Load Torque.

3.3 Plant model

Finally, the electromechanical model used is mainly based on the following dynamic equations:

$$\dot{i_a} = \frac{V_s}{L} - \frac{Ri_a}{L} - \frac{k_v\dot{\theta}}{L}$$
$$T_a = k_t i_a(t) - Att_{mot}\dot{\theta}$$
$$\ddot{\theta} = \frac{T_a - \beta\dot{\theta}F_c}{I_n}$$
$$\ddot{x} = \frac{F_1 - f_c(F_2\Delta_x + cost)}{m}$$

Thus, replacing the torque equation in the angular acceleration one, the final state equations of the model are obtained:

$$\begin{cases} \ddot{\theta} = \dfrac{k_t i_a}{I_n} - \dfrac{Att_{mot}\dot{\theta}}{I_n} - \dfrac{\beta F_c \dot{\theta}}{I_n} \\ \ddot{x} = \dfrac{F_1}{m} - \dfrac{F_c(F_2\alpha + c)}{m} \\ \dot{i_a} = \dfrac{V_s}{L} - \dfrac{Ri_a}{L} - \dfrac{k_v\dot{\theta}}{L} \end{cases}$$

3.4 Most significant parameter choice

A literature research on the parameters that most influence end-effector wear has shown that the **friction coefficient** plays a key role in the interaction between the workpiece and the end-effector tool.
In this first phase of development, it is decided to adopt the friction coefficient, referred to as β, as the parameter on which to base the filter wear hypothesis.
From now on, in the multi-model approach and therefore in residual error analysis and evaluation tests, β will be used as a parameter to be estimated and from which to extrapolate the SoH of the machine.
Clearly, the interaction between the tool and the workpiece is more complex than a simple coefficient, since it depends on various factors such as temperature, the used material, relative speed, applied forces, cooling media, etc.
Thus, the key factor is that the analysis carried out during this Thesis work must work independently of the specific choice of the parameter in a way that a consequently complication of it could lead to results that are not so far from the ones obtained considering β as representative. This is the reason why, an in-depth modelling of the interaction will be the goal of another research group working on the same project.

3.4.1 Simulink implementation

The equations described to date can be translated into model through suitable simulation environment program. For this thesis work, it has been decided to implement the model in Matlab and simulink, because are suitable for this sort of simulations. Implementing such mathematical model using those tools is very intuitive, particularly if the MATLAB Guidelines are followed. Applying these guidelines one can improve the consistency, clarity, and readability of models. The guidelines also help you to identify model settings, blocks, and block parameters that affect simulation behavior or code generation. MATLAB guidelines can be found in the mathworks site [19].

In Figure 3.3 it is shown how it is implemented a simple contact logic using boolean operators, which is intended to handle the contact with the workpiece.

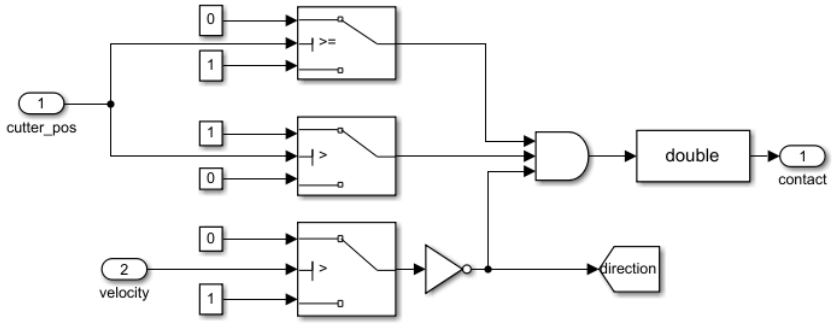


Figure 3.3. Contact logic

For the implementation of the equations described in paragraphs 3.2 and 3.1, it was decided to create a **dynamic** MATLAB function and to use integrator blocks to integrate the output and feedback where needed:

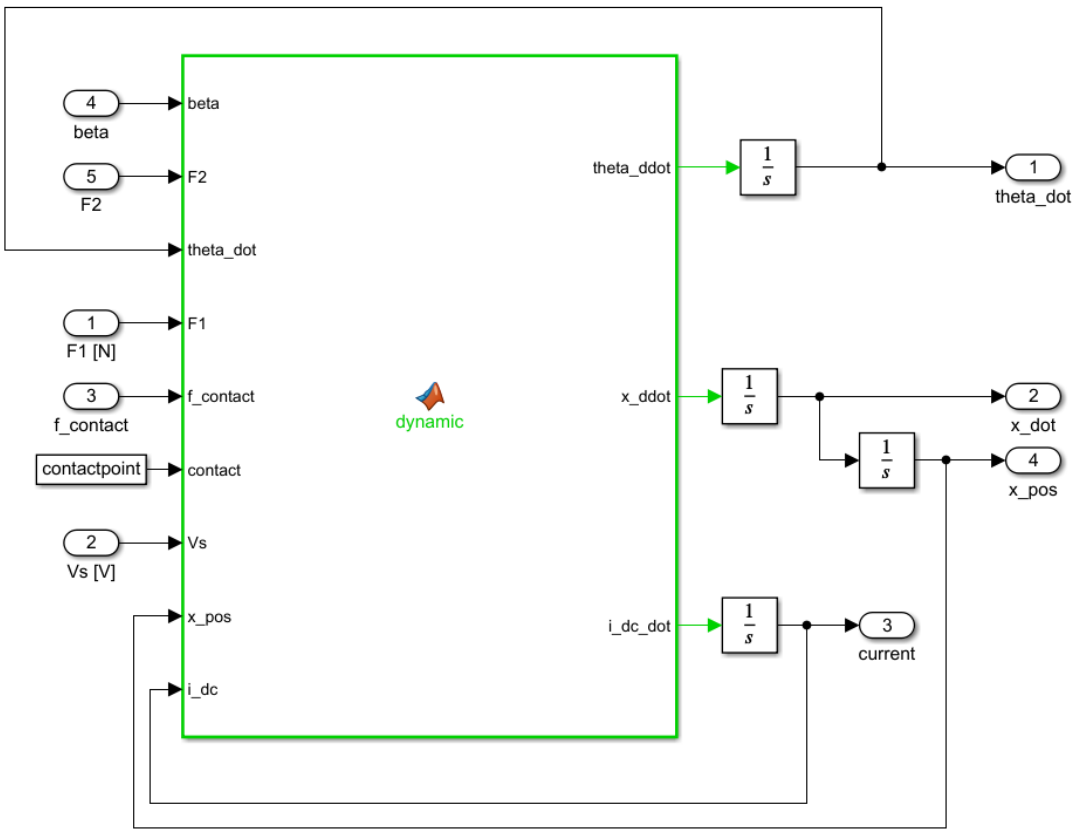


Figure 3.4. Plant Simulink.

Initial condition of the integrators are all set to 0, as well as the starting condition of the contact. The output of the plant coincide with the states of the system:

1. $\ddot{\theta}$: angular acceleration.
2. $\ddot{x}$: linear acceleration.
3. $\dot{i_{dc}}$: derivative of the current.

In the next page (figure 3.5), there is the overall simulink implementation of the model.

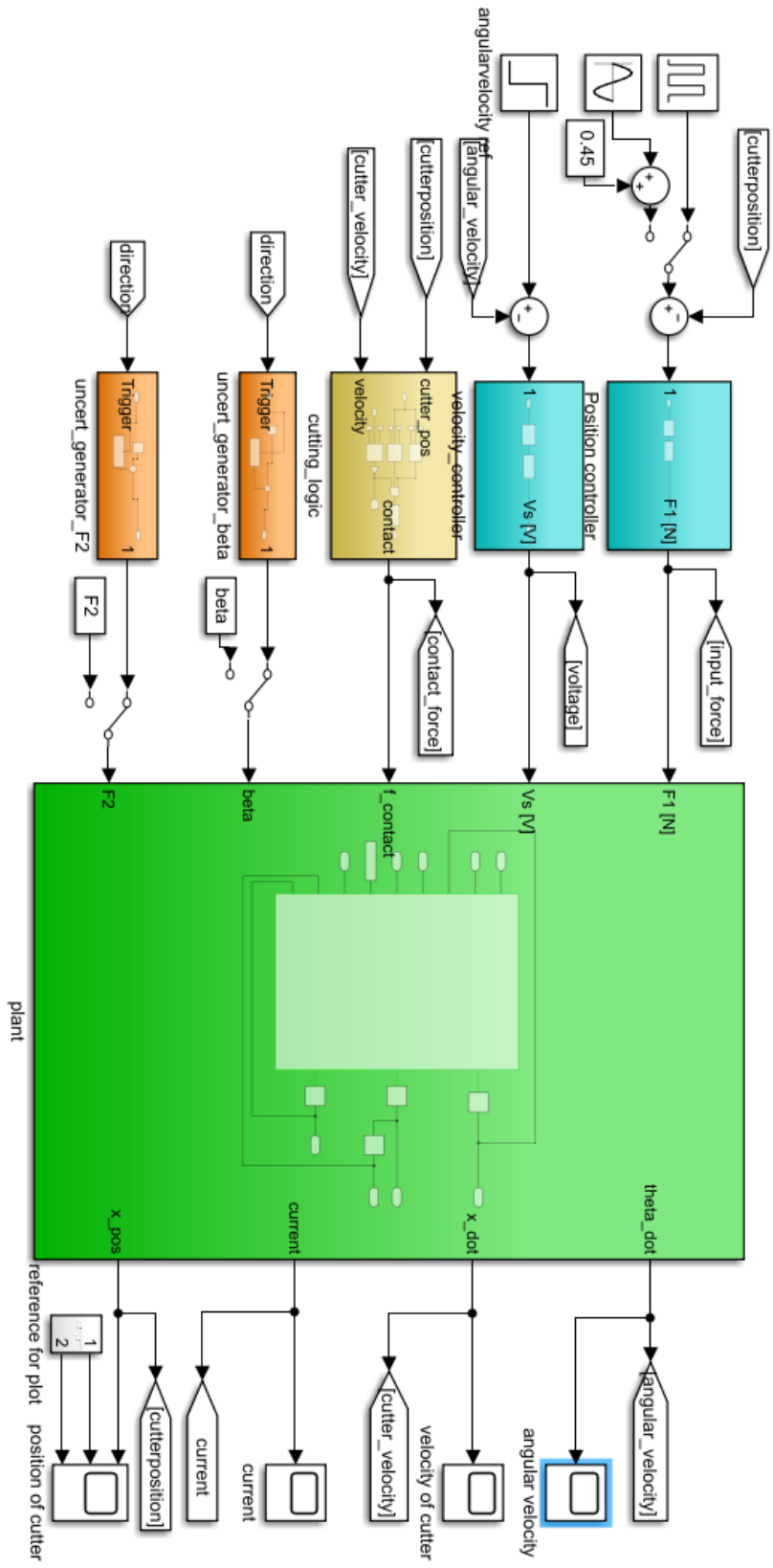


Figure 3.5. Simulink implementation of the model

To sum up, the simulink blocks are:

- Green block: plant implementation.
- Orange block: simple triggers which introduce 2% of uncertainty on the inputs.
- Cyan blocks: PID controllers on the position and angular velocity.
- Yellow block: contact logic.

For what regards the PID controllers, they have been tuned using a MATLAB predefined tool (Control System Tooolbox) in such a way to find a good balance between robustness and efficiency. Instead, the next plot represents the contact logic output:

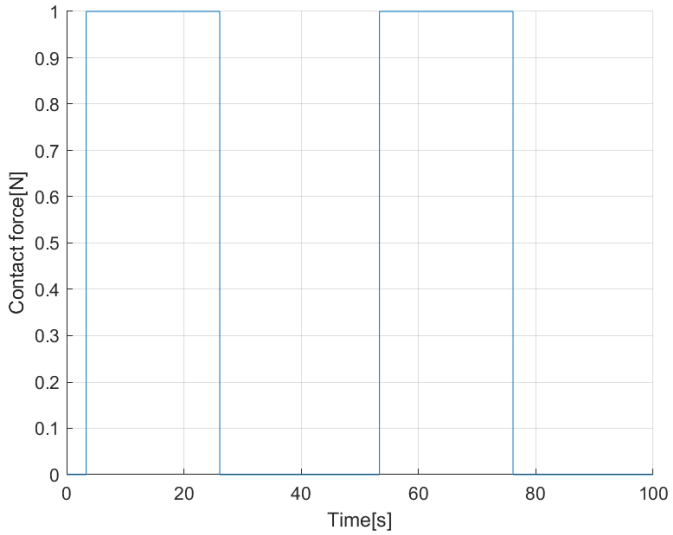
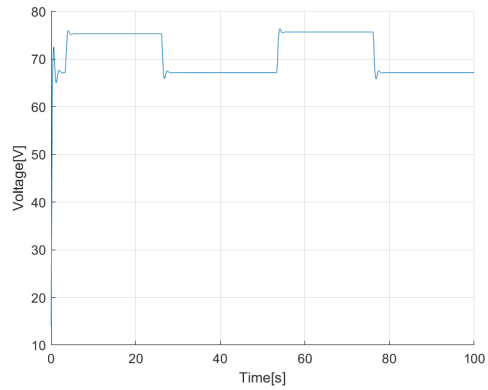
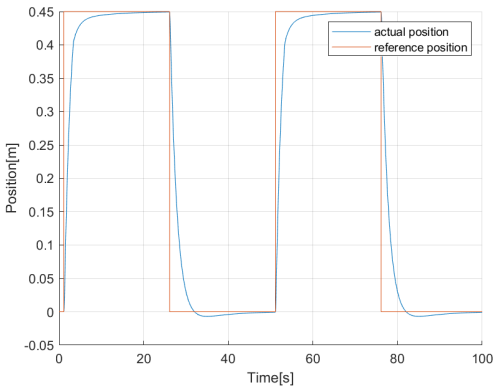


Figure 3.6. Contact force control input plot.

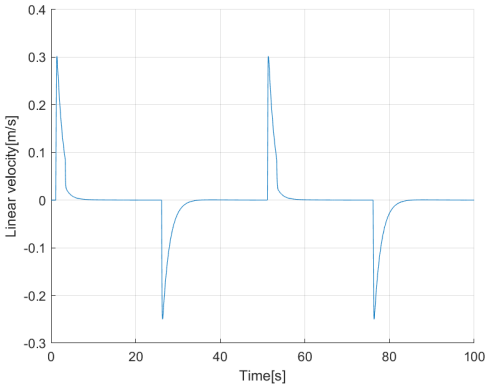
As it was expected, since the contact logic is made up so that the output is 1 when there is contact and 0 when there is no contact, the contact force input plot oscillate in a discrete way between those two values. Finally, in Figure 3.7 are depicted all outputs and main parameters of the model so that the physical behaviour is described.



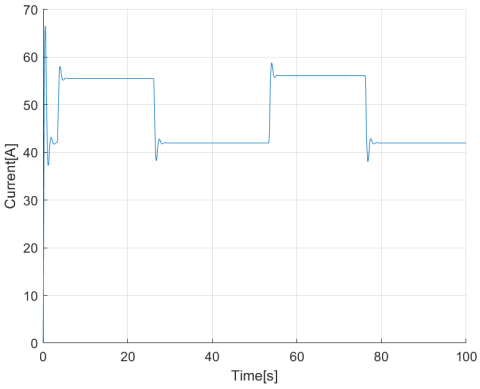
(a) Voltage plot



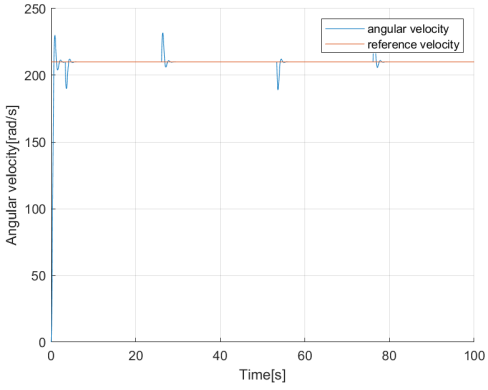
(b) Position plot



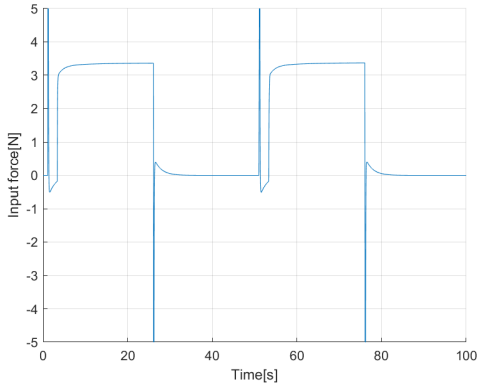
(c) Linear velocity plot



(d) Current plot



(e) Angular velocity



(f) Input force plot

Figure 3.7. Summary plots of Model's main parameters.

4 EKF Bank: Multi-model approach

State observers are mainly used to provide an estimate of the internal state of a given real system, from measurements of the input and output of the real system. This utilization is very suitable when there is noise and it is needed to be reduced, or when there is a state which cannot be measured directly and there is the necessity of have a more accurate estimation of it. Using a Kalman-filter in order to understand the state and the working condition looking at the residual error is not a common utilisation of such state-observers. What it is need to be done for this scope is a deep analysis of the residual errors. The latter, are defined as the module of the difference between a state estimation and the real state:
Supposing that x(t) is a state of a system $\mathcal{M}(x(t))$:

$$Residual\_error = |\hat{x(t)} - x(t)| \quad \forall t$$

The **residual error** can seem very similar to an **estimation error**, but there is a slight but very important difference. On one hand, the residual error is the difference between the state estimation and the state coming from the output of the real plant, so there is no need to know the internal exact formulation of the plant. On the other hand, the calculation of the estimation error suppose to perfectly know the real value of the parameter to be estimated. This difference is rather crucial, because it is not possible to suppose the real value of the internal state of the system. Moreover, it is important to mention that the residual error can be affected by measurement noise. For this Thesis work it was supposed to have a really low measurement error on the states because the approach is intended to be as much simple as possible at first.

Starting from the considerations done until now, is it possible to relate the residual error to a state or to a set of parameters that can represents the SoH?
How much the other parameters changes affect the residual error calculation?
Which is the state on whom the difference of the residual errors are more highlighted?
Is it really possible to apply the multimodel approach for the estimation of SoH?
The questions that need to be solved to answer to the last one are numerous, and during this chapter there will be the proof of concepts and some possible answer to the listed questions, mainly based on empirical approach.

The starting question is: What is the effect of a parametric variation on the residual error, and which is the weight of this variation? Initially, the focus was the search for papers and/or documents that take into account the effects of parametric variations on the residual error produced by the observer: a possibility is to consider faults as parametric variations that induce a change in system behavior. Nevertheless those kind of approach are quite time-consuming because require

strong theoretical analysis. Another available option, rather more practical, is the search for a method that foresees a sensitivity analysis with the aim of identifying which are the parameters whose variations have a relevant effect on the output and consequently these parameters could be used as criteria to do the scheduling and eventually decide which will be the partition method for the state space. Sensitivity analysis is the method most frequently used during research on this topic and seems to give the best results. This method consists in getting a many data from experiment strongly varying the condition, so that there is a strong background where a a global sensitivity analysis (GSA) can be performed[20].

4.1 EKF

The Extended Kalman filter is a method to estimate both the states of the system and also his parameters; it is an iterative procedure, composed by different equations that are fast evaluated as the system changes during time. In each step there is the estimation not only of the system states but also of the covariance matrix, indicator of the uncertainty of the states estimate. A "large" value of covariance indicates a high level of uncertainty while a "small" one indicates confidence in the estimate. As seen previously, our system is represented by the following state equations:

$$\begin{cases} \ddot{\theta} = \dfrac{k_t i_a}{I_n} - \dfrac{Att_{mot}\dot{\theta}}{I_n} - \dfrac{\beta F_c \dot{\theta}}{I_n} \\ \ddot{x} = \dfrac{F_1}{m} - \dfrac{F_c(F_2\alpha + c)}{m} \\ \dot{i_a} = \dfrac{V_s}{L} - \dfrac{Ri_a}{L} - \dfrac{k_v\dot{\theta}}{L} \end{cases}$$

We can notice the form of a classical nonlinear system $\dot{x} = f(x, u)$ and starting from the following state-space model in a discrete-time domain:

$$\begin{cases} x_{k+1} = f(x_k, u_k) + \omega_k \\ y_k = h(x_k) + v_k \end{cases}$$

where x_k are the states, u_k are the inputs, y_k is the output, ω_k is the disturbance and v_k is a measurement noise. $f(\cdot)$ is a nonlinear state transition function that describes the evolution of states x from one time step to the next. The nonlinear measurement function $h(\cdot)$ relates x to the measurements y at time step k. At each time step, $f(x_k, u_k)$ and $h(x_k)$ are linearized by a first-order Taylor-series expansion. We assume that $f(\cdot)$ and $h(\cdot)$ are differentiable at all operating points (x_k, u_k).

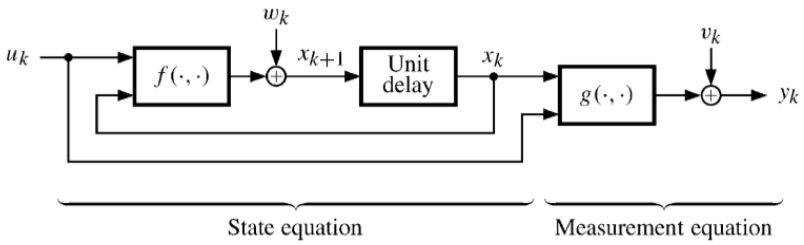


Figure 4.1. Diagram of nonlinear discrete time system in state-space form

The inputs u_k are:

- V_a: armature voltage
- F_1: horizontal force that moves the cutter
- F_c: function that define the contact with the object.

The states x_k are the same of the plant model while the output y_k we suppose to coincide with the states. We must define the following quantities:

- $F_k = \dfrac{\partial f}{\partial x_k}(x_k, u_k)$ = Jacobian of f computed in (x_k, u_k)
- $H_k = \dfrac{\partial h}{\partial x_k}(x_k)$ = Jacobian of h computed in x_k
- $\hat{x_k}$ = estimate of x_k computed at step k
- x_k^p = prediction of x_k computed at step k-1
- P_k = covariance matrix of $x_k - \hat{x_k}$
- Q^d = covariance matrix of ω_k
- R^d = covariance matrix of v_k

As regards the matrices Q^d and R^d, since we have no information on the disturbances, we chose them as diagonal matrices by a trial and error procedure. The algorithm can be summarized with the following step:

1. **Prediction**

$$x_k^p = f(\hat{x}_{k-1}, u_{k-1})$$
$$P_k^p = F_{k-1} P_{k-1} F_{k-1}^T + Q^d$$

2. **Update**

$$S_k = H_k P_k^p H_k^T + R^d$$
$$K_k = P_k^p H_k^T S_k^{-1}$$
$$\Delta y_k = y_k - h(x_k^p)$$

$$\hat{x}_k = x_k^p + K_k \Delta y_k$$
$$P_k = (I - K_k H_k) P_k^p$$

In addition, a further step was added to the algorithm to calculate the residual error for each state variable, which we recall is the modulus of the difference between the estimates produced by EKF and the data collected from the simulation of the plant.

$$Residual\_error = |\hat{x(k)} - x(k)| \quad \forall k \tag{4.1}$$

The final output of the EKF block are therefore the residual errors that is needed to carry out an analysis and establish whether a multi-model approach may be better for the final objective. In the figure 4.2 there is the Simulink implementation of the EKF.

Figure 4.2. Simulink scheme of the EKF.

4.2 Residual error analysis

Considering the residual error as raw value, it is not significant to watch the errors amount every sample time since it does not lead to any particular conclusion. To assign a meaningful sense to the residual error, it must be conducted a signal elaboration and the discrete values of the error sampled in time must be processed. Signal theory and data processing are a widely treated in today scientific literature, so there are countless articles that can be followed in order to understand the best way to treat a signal. One of the most complete article is the one cited in the state of the art chapter of this Thesis [14]. This article plainly explain how to analyze residual errors using both frequency domain and time domain indicators. Among all, some of the most simple and efficient according to the article are:

- Mean.
- Integral.
- RMS.
- Correlation.

- PSD.
- Covariance.

To verify which is the best, it can be applied and experimental approach. In particular, it is possible to set up some test to verify which of this methods, applied on the residual error, it is most suitable. It must be kept in mind that the objective is to find a method that can highlight the difference between changing of beta. That's because the aim is to make the system very sensitive to little change of beta, but confidently less sensitive to other parameters variations. So the approach will be to test 20 little variation of beta, starting from the nominal condition and increasing of 20% every step. It will also be reported a little variation on the horizontal input force of about 2%. The nominal values (calculate with beta nominal) of the errors elaborated for each state and for each considered are reported in the following table:

Method	Nom. Rot. acc.	Nom. linear acc.	Nom. curr. der.
Mean	1.7111	0.0242	0.4959
RMS	3.6821	0.0509	1.1879
Correlation	1.3979	1.0000	3.8928
covariance	10.6403	0.0020	1.1663
integral error	58.7780	1.2503	33.7761

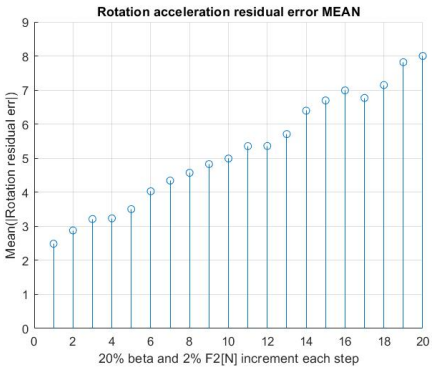
Table 4.1. Nominal values of the errors for each method.

In order to understand the results, it is also defined a FOM(Figure of merit) as a simple index that describe how far the non-nominal condition model is with respect to the nominal one. This FOM index is the ratio between the absolute value of the residual error and the absolute value of the residual in nominal condition:

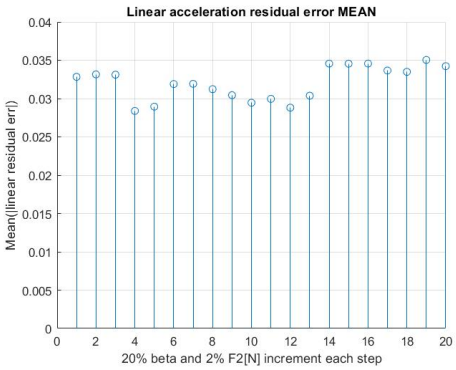
$$FOM = \frac{|\text{Residual error parameter}|}{|\text{Nominal residual error parameter}|}$$

Keeping in consideration the nominal values reported in Table 4.1, the following plot will show which signal manipulation can be considered as the most suitable. Let's start the tests from each method:

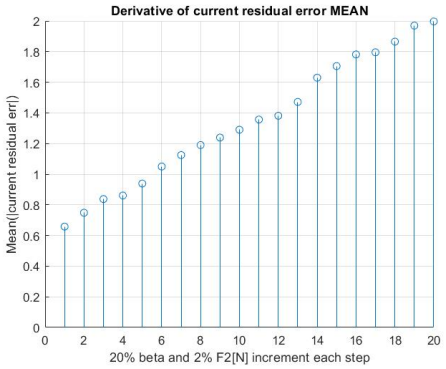
1. **MEAN:**



(a) Rot. acc.



(b) Lin. acc.



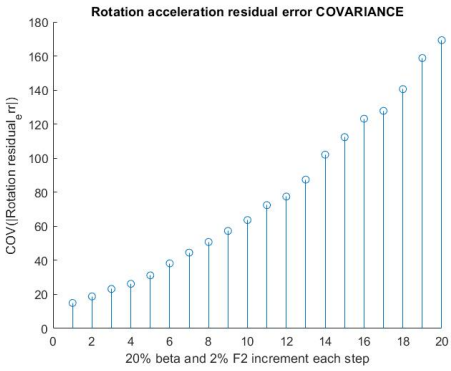
(c) Curr. der

Figure 4.3. Mean test.

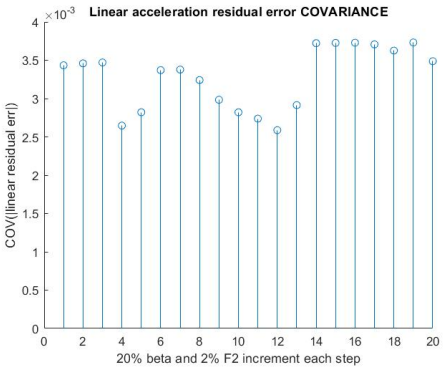
FOM angular	FOM linear	FOM current
4.6759	1.4162	4.0266

Table 4.2. FOM mean

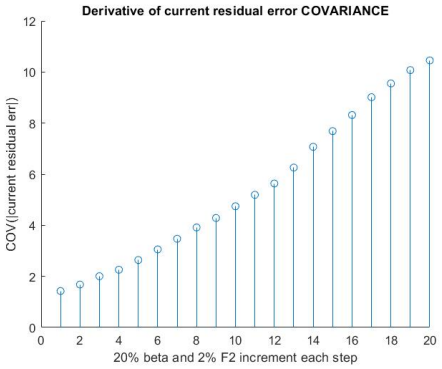
2. **Covariance:**



(a) Rot. acc.



(b) Lin. acc.



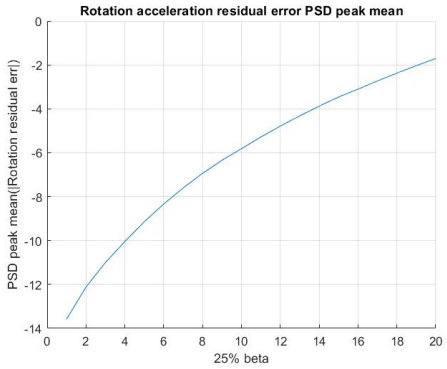
(c) Curr. der

Figure 4.4. Covariance test.

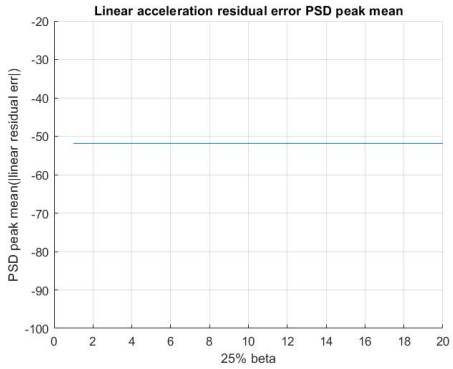
FOM angular	FOM linear	FOM current
5.9107	1.7358	7.9633

Table 4.3. FOM Covariance

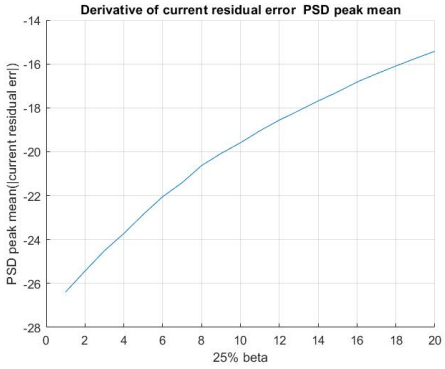
3. **PSD:** For the power spectral density, it has been considered an interpolation of the maximum peaks.



(a) Rot. acc.



(b) Lin. acc.



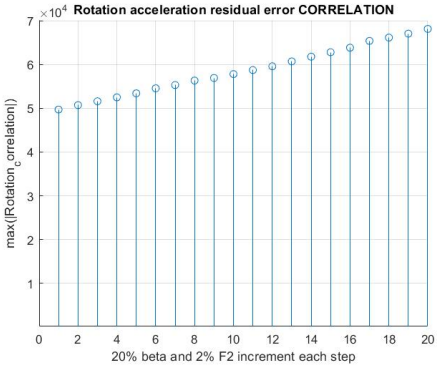
(c) Curr. der

Figure 4.5. PSD test.

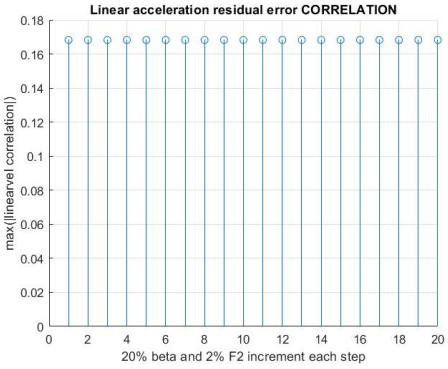
FOM angular	FOM linear	FOM current
7.527	1.001	5.743

Table 4.4. FOM PSD

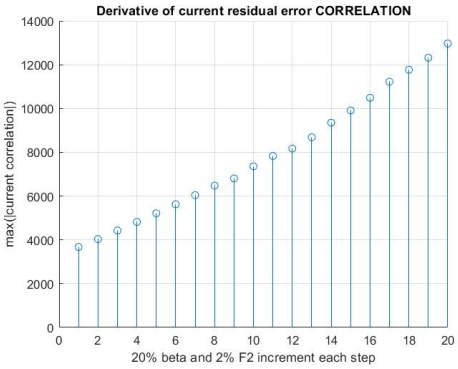
4. **Correlation:**



(a) Rot. acc.



(b) Lin. acc.



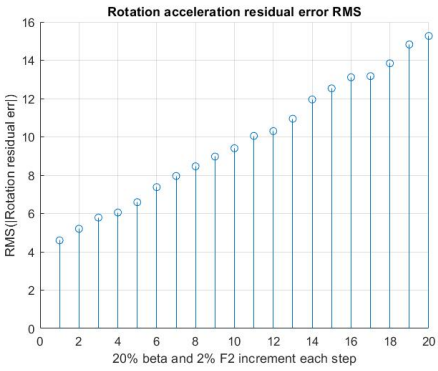
(c) Curr. der

Figure 4.6. Correlation test.

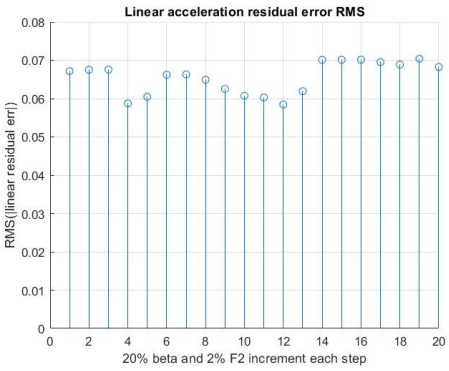
FOM angular	FOM linear	FOM current
1.3979	1.0000	3.8928

Table 4.5. FOM Correlation

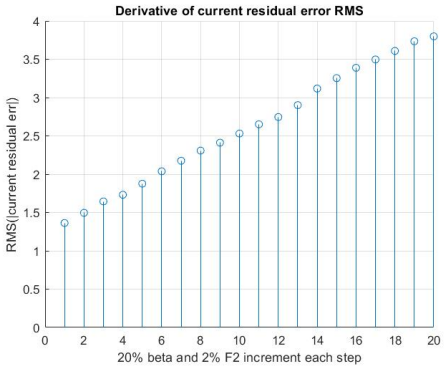
5. **RMS:**



(a) Rot. acc.



(b) Lin. acc.



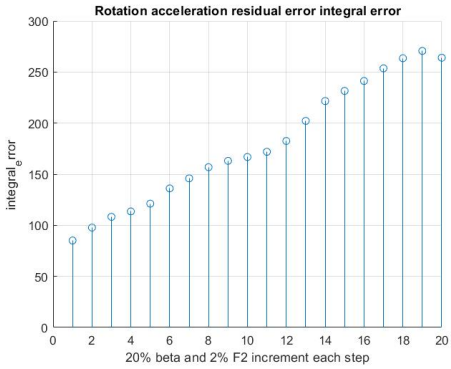
(c) Curr. der

Figure 4.7. RMS test.

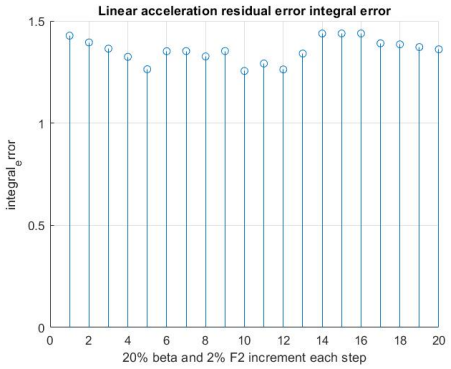
FOM angular	FOM linear	FOM current
4.1469	1.3404	3.1979

Table 4.6. FOM Correlation

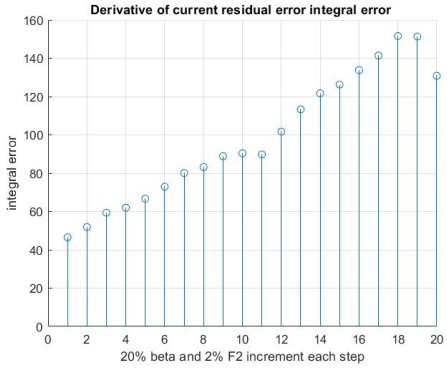
6. **Integral:**



(a) Rot. acc.



(b) Lin. acc.



(c) Curr. der

Figure 4.8. Integral test.

FOM angular	FOM linear	FOM current
4.4911	1.0886	3.8739

Table 4.7. FOM Integral

Results can be summed up in the following tables:

- Angular acceleration:

	mean	*integral*	*RMS*	*cov*	$corr_{max}$	*PSD*
	1.93	58	5.8	30	6.1	−19.9
$\beta\uparrow$	2.05	61	6	32	6.4	−18.6
	2.19	66	6.2	34	6.8	−17.4
	2.32	70	6.5	36	7.2	−16.8
$\Downarrow$	2.44	73	6.7	39	7.6	−15.9
	2.55	77	6.9	41	8	−15
	2.76	80	7.2	44	8.4	−14.2
	2.88	84	7.4	46	8.7	−13.5
	2.98	87	7.7	50	9.1	−12.7
	3.1	91	7.9	53	9.6	−12.2

- Linear acceleration

mean	*integral*	*RMS*	*cov*	$corr_{max}$	*PSD*
0.005	0.15	0.01	0.002	0.15	−63

  The results on the linear acceleration, varying F_2, differ so little from the results obtained with the nominal values that they are irrelevant.

- Current derivative

	mean	*integral*	*RMS*	*cov*	$corr_{max}$	*PSD*
	0.67	20	1.8	3.0	9.2	−25.9
$\beta\uparrow$	0.71	21	1.9	3.2	9.2	−25.8
	0.75	22	2.0	3.5	9.2	−25.2
	0.79	23	2.1	3.9	9.2	−24.9
	0.83	25	2.2	4.3	10	−24.8
$\Downarrow$	0.87	26	2.3	4.7	11	−24.3
	0.91	27	2.4	5.2	13	−23.7
	0.95	28	2.5	5.6	15	−23.2
	0.99	29	2.6	6.2	16	−22.6
	1.03	31	2.8	6.7	18	−22.6

Beyond the good results for the integral error reported above, most of the methods appear suitable for the scope. Indeed, choosing to utilize the rotation acceleration error or the current related error, there is not a method that really take advantages on the others. The RMS method and the integral are almost equivalent in terms of FOM, that indicates that are both good for the aim. So the decision to choice for

the integral error method comes from another consideration: the **integral error** is cumulative and takes into account the previous state of the system. The RMS is very good and will be used to elaborate the errors as well as the integral error. Nevertheless, considering that mechanical system states naturally needs time to evolve and change, taking into account a cumulative way to treat the residual error is definitely the best choice. The integral error behaviour will be widely treated from this point until the end of this thesis work.

4.2.1 Residual error comparison

Once decided that the integral of the residual errors is the most suitable choice to carry out a multivariate analysis, it is possible to see what of the variables available contain more information. This is done using a simulation environment composed of:

- The **Plant Model** obtained in 3.3.
- The **EKF** described in 4.1.
- A logic of management and decision of the integral of the residual errors that contains a possible integral reset as it will be seen.

The estimator allows the absolute error computation of the angular acceleration error and of the derivative of the current. Thus, exploiting a boxplot analysis (details can be found in appendix) on the integral of such errors, it is possible to decide which kind of error best describe our model. The simulation is carried out considering the nominal parameters of the machine, described in the table below.

	Nominal value
Mass [kg]	3
Radius [m]	0.3
Resistance [kΩ]	0.6
Inductance [mH]	0.1
Torque constant	1.5
Voltage constant	0.2
Motor Inertia [kgm^2]	0.001
Friction coefficient	0.1

Table 4.8. Nominal CNC parameters.

The same machine is considered to work in nominal condition when:

	Nominal value
Angular velocity reference $[\frac{rad}{s^2}]$	210
Position reference [m]	0.5
Duty cycle [%]	50
Number of cycles	4
Contact point [m]	0.4
Workpiece length [m]	0.09

Table 4.9. Nominal working conditions.

In the same environment, a variable e_c is defined and used to discriminate which of the residual errors available will be considered. In particular:

- $e_c = 1$: only the angular acceleration error is considered.
- $e_c = 2$: only the current derivative error is considered.
- $e_c = 3$: an average between the two errors is computed and considered.

Thus, considering a one hundred seconds simulation, the friction coefficient β of the plant is made to change between five different values while the filter one is keep fixed to the nominal one. In this way, analyzing the boxplots of the three different errors it is possible to see which of the three one allow a better distinction between the various friction coefficient cases.

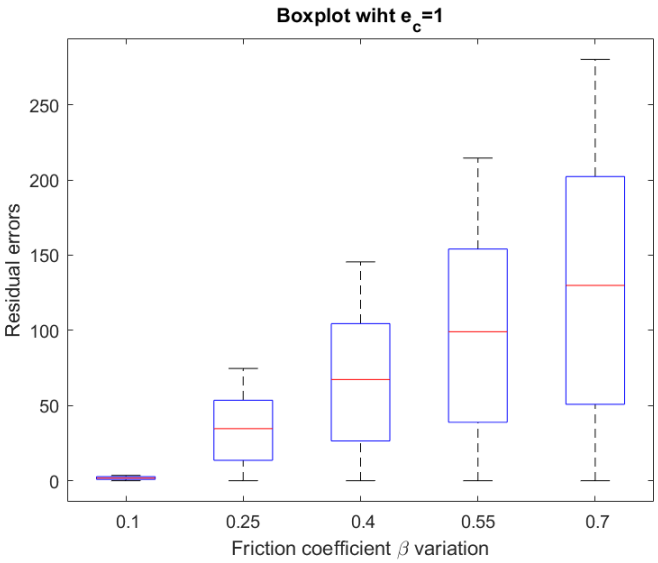


Figure 4.9. Boxplot of the angular acceleration error.

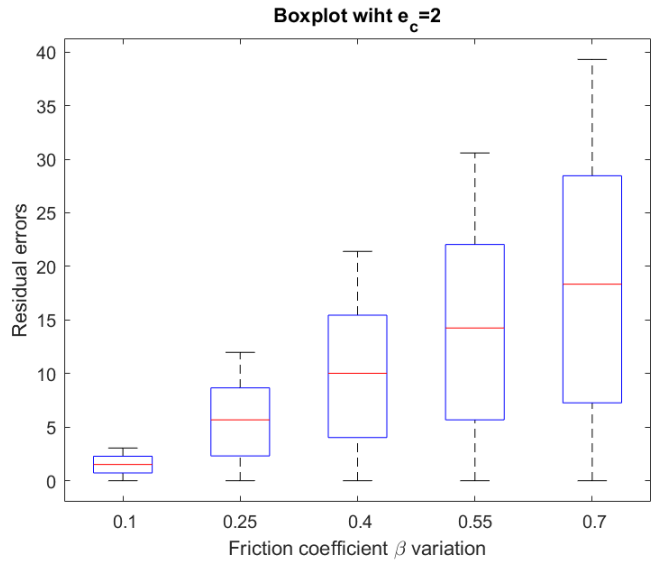


Figure 4.10. Boxplot of the current derivative error.

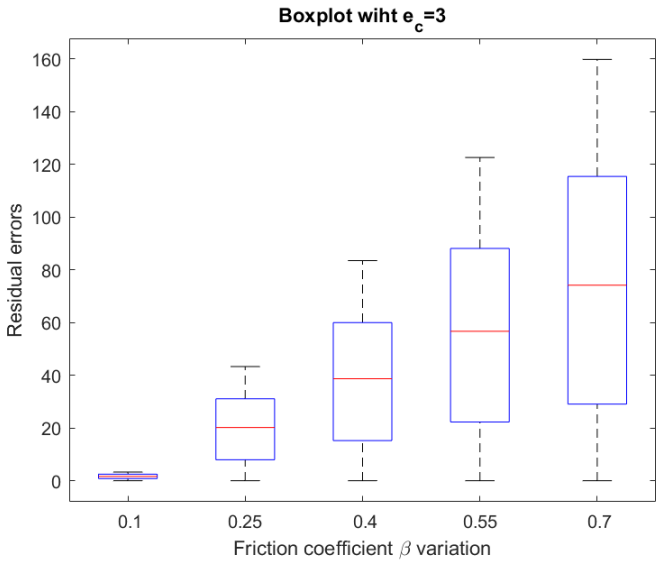


Figure 4.11. Boxplot of the average error.

Having a look at the results, it is possible to see that in the first and in the last case a better separation between the matched value and the other ones is obtained. On the contrary, considering the derivative of the current the separation is not so marked as the other ones. Thus, a first suggestion is that the e_c variable must be set to 1 or to 3 to obtain more remarkable results. Moreover, focusing just on these values, when considering the angular acceleration, looking the median values of the boxes, an higher distance between the nominal error is obtained. Finally,

it is possible to conclude that when setting e_c equal to one better results will be expected in the next.

4.3 Evaluation tests

Till now, all the simulations were carried out assuming that the machine always exploit the same kind of lavoration. Thus, it is convenient to test/stress the environment with different input conditions in order to see if the state observer works well in any case and which kind of processing affects more the algorithm. In particular, a kind of multivariate error analysis is made, changing one variable at a time:

- **Angular Velocity**
  With all the other parameters fixed, only the angular velocity is made to change:

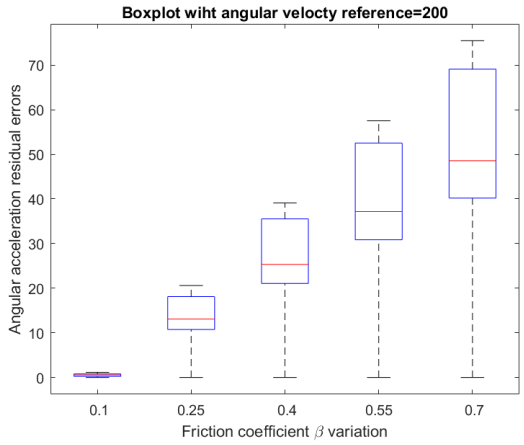


Figure 4.12. Boxplot error with 200 $[\frac{rad}{s^2}]$ angular velocity.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.10. Test with 210 $[\frac{rad}{s^2}]$ angular velocity.

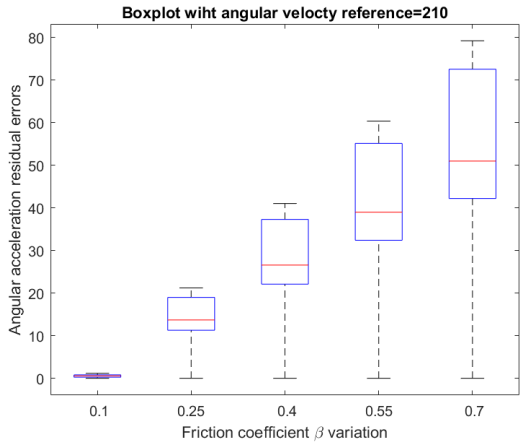


Figure 4.13. Boxplot error with 210 $[\frac{rad}{s^2}]$ angular velocity.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	220
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.11. Test with 220 $[\frac{rad}{s^2}]$ angular velocity.

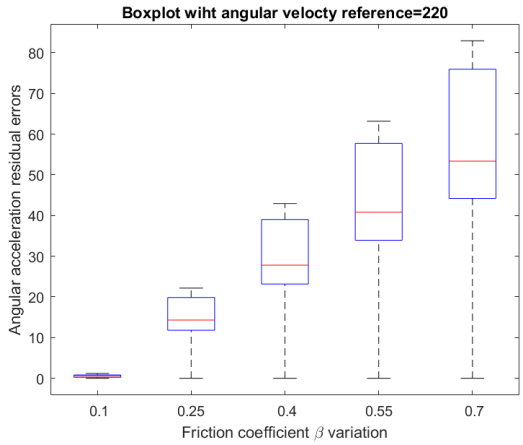


Figure 4.14. Boxplot error with 220 $\frac{rad}{s^2}$ angular velocity.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	230
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.12. Test with 230 $[\frac{rad}{s^2}]$ angular velocity.

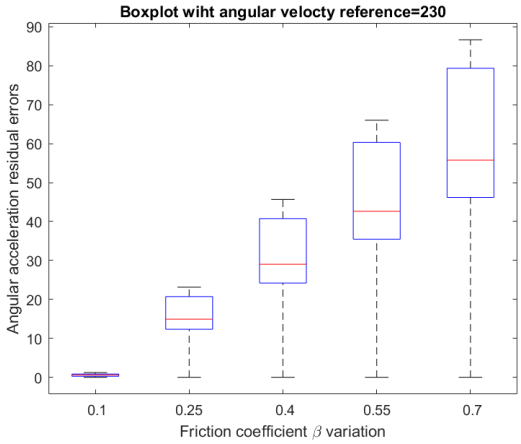


Figure 4.15. Boxplot error with 230 $[\frac{rad}{s^2}]$ angular velocity.

- **Position**
  With all the other parameters fixed, only the position reference is made to change:

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.4
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.13. Test with 0.4 $[m]$ position reference.

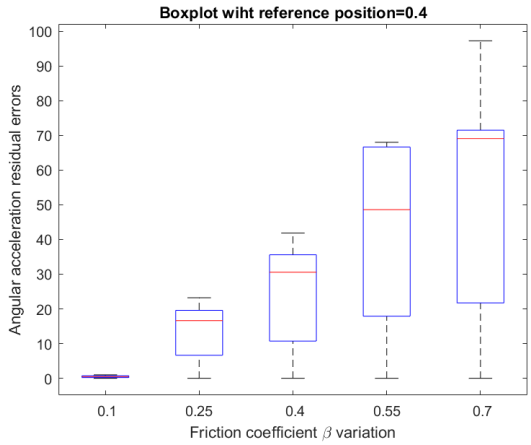


Figure 4.16. Boxplot error with 0.4 $[m]$ position reference.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.47
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.14. Test with 0.47 $[m]$ position reference.

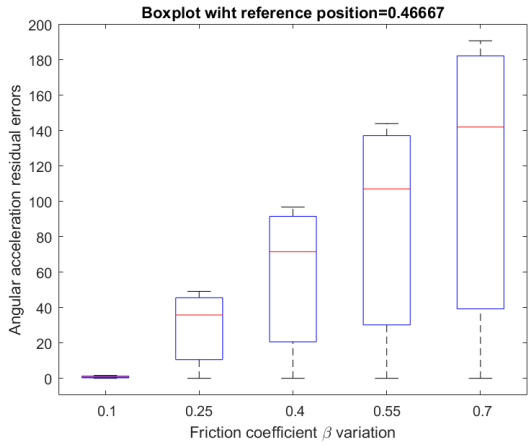


Figure 4.17. Boxplot error with 0.47 $[m]$ position reference.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.53
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.15. Test with 0.53 $[m]$ position reference.

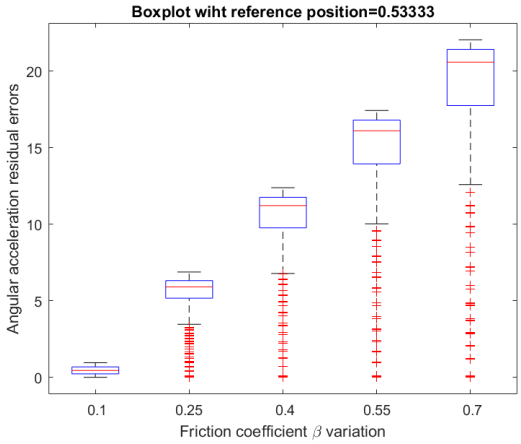


Figure 4.18. Boxplot error with 0.53 $[m]$ position reference.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.6
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.16. Test with 0.6 $[m]$ position reference.

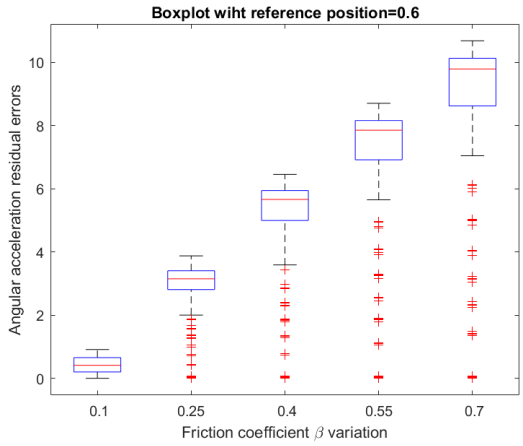


Figure 4.19. Boxplot error with 0.6 $[m]$ position reference.

- **Duty cycle**
  With all the other parameters fixed, only the duty cycle is made to change:

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	20
Number of cycles	4	4

Table 4.17. Test with 20 % duty cycle.

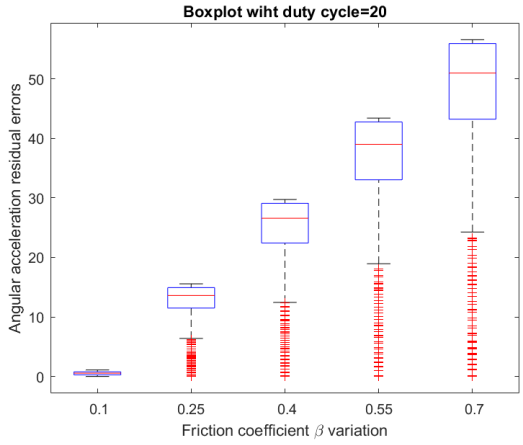


Figure 4.20. Boxplot error with 20 % duty cycle.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	40
Number of cycles	4	4

Table 4.18. Test with 40 % duty cycle.

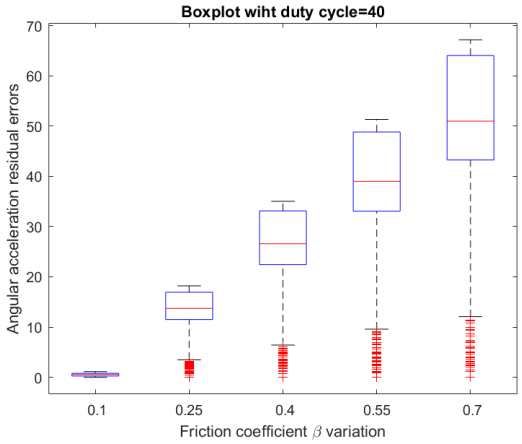


Figure 4.21. Boxplot error with 40 % duty cycle.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	60
Number of cycles	4	4

Table 4.19. Test with 60 % duty cycle.

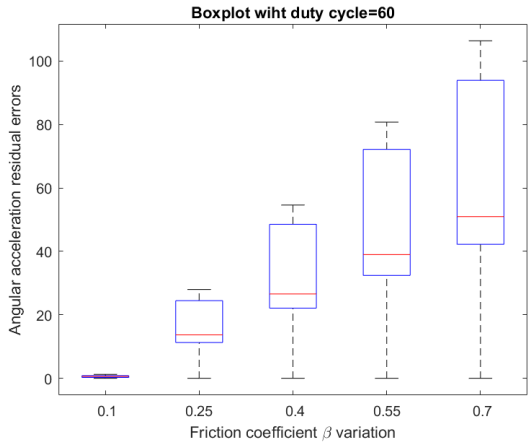


Figure 4.22. Boxplot error with 60 % duty cycle.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	80
Number of cycles	4	4

Table 4.20. Test with 80 % duty cycle.

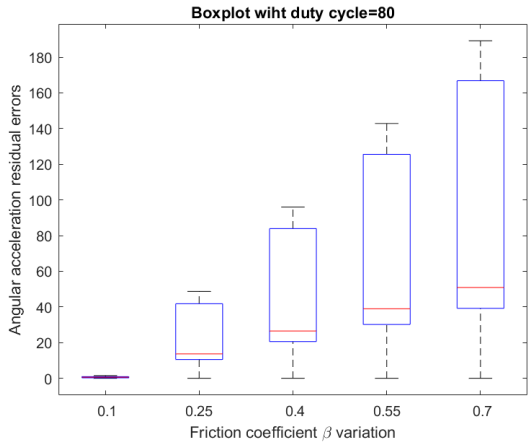


Figure 4.23. Boxplot error with 80 % duty cycle.

- **Number of cycles**
  With all the other parameters fixed, only the number of cycles is made to change:

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.21. Test with 4 number of cycles.

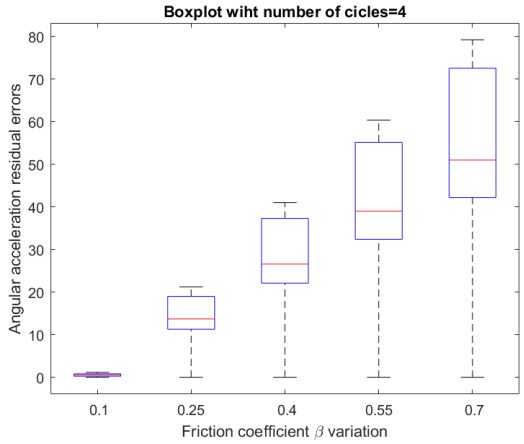


Figure 4.24. Boxplot error with 4 number of cycles.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	6

Table 4.22. Test with 6 number of cycles.

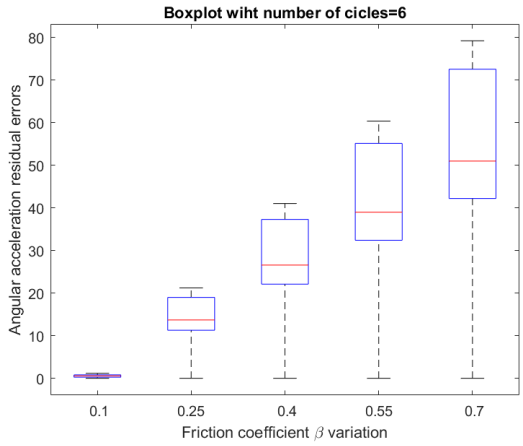


Figure 4.25. Boxplot error with 6 number of cycles.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	8

Table 4.23. Test with 8 number of cycles.

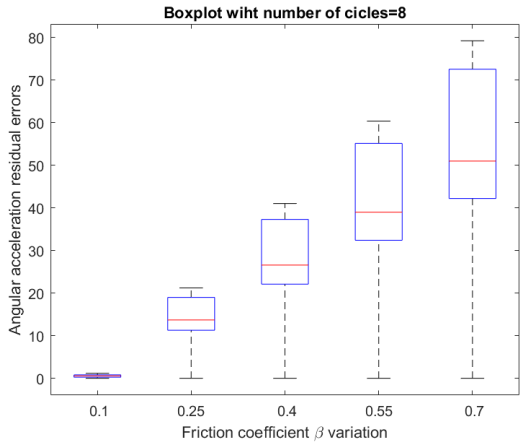


Figure 4.26. Boxplot error with 8 number of cycles.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Position [m]	0.5	0.5
Duty cycle [%]	50	50
Number of cycles	4	10

Table 4.24. Test with 10 number of cycles.

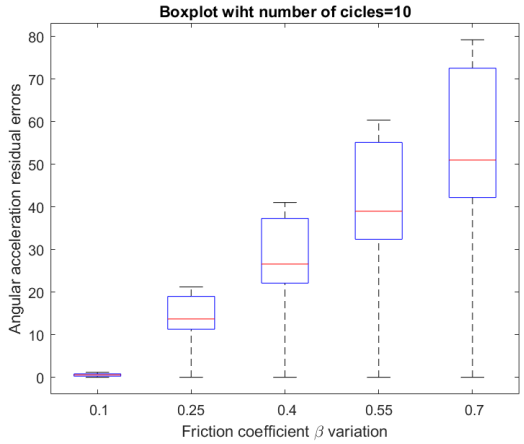


Figure 4.27. Boxplot error with 10 number of cycles.

Having a look at the various tests performed so far, it is possible to see that there are no sensible variation when considering different working conditions with respects to the nominal ones. Thus, the error associated with the correct model to estimate is always smaller then the other ones. Moreover, in some particular cases there is a better distinction between the boxplots, indicating a more accuracy on the estimation algorithm. Moreover, it is necessary to state that during these tests a reset of the integral error was considered, whose choice is justified in 4.3.1.

4.3.1 Reset time choice

A crucial aspect of residual error analysis is the choice of the integral's reset time. An integration period should be chosen mainly for two reasons:

- clearly, after a certain period of time while it is growing up, it will reach its maximum value distorting the results;
- there may be situations in which there are transient errors depending on many factors such as the work period, the type of machining process, ecc. that can influence the integral.

Thus, in order to choose an optimal reset time, a boxplot analysis was carried out by varying it through the simulation range.

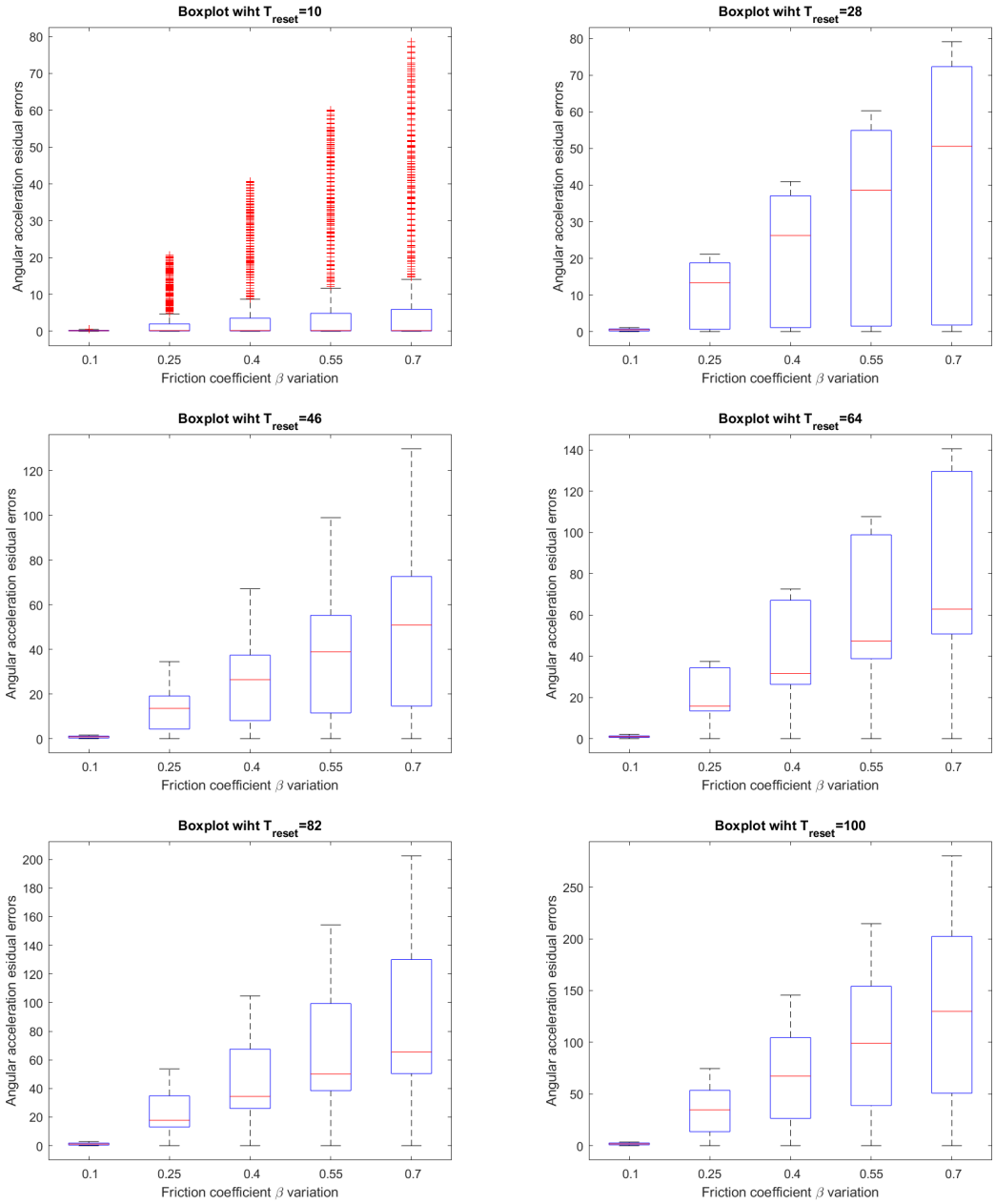


Figure 4.28. T reset analysis

From the boxplots above can be seen that with a reset time of 10s, the error variation is quite small compared to the others. This is probably due to the fact that in 10s time there are no sensible dynamic variations in the system that would capture an estimation mismatch. From 30s onwards the results are quite similar but it has been decided to investigate a narrower range right after 30s because by

increasing more and more the reset time it is possible to run into the problems listed above.

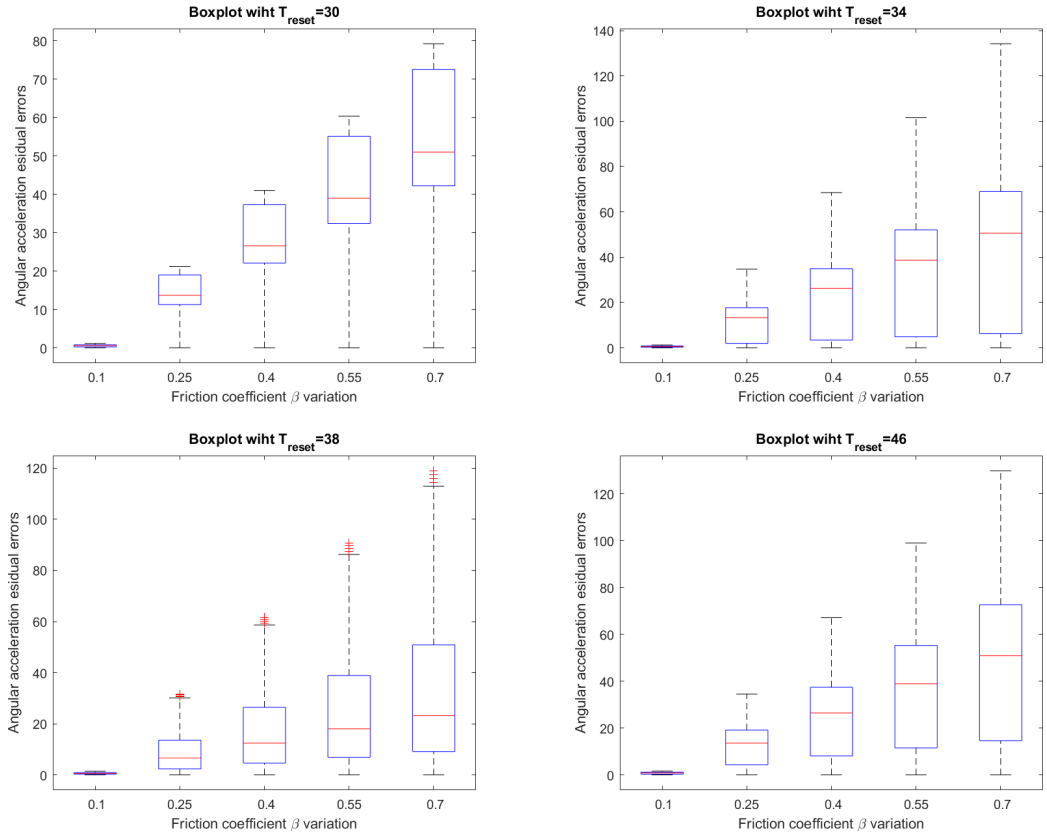


Figure 4.29. T reset choice

The results highlight that choosing a reset time of 30 seconds is the best choice for this kind of framework, because it corresponds to a processing period. In practice, since it is not possible to exactly know how long a processing period takes it is better to choose a reset time large enough to capture the dynamic variations of the system under study.
In Figure 4.30 it is possible to notice the behaviour of the angular acceleration error's integral with nominal values for all the parameters when a Reset time of 30s has been chosen.

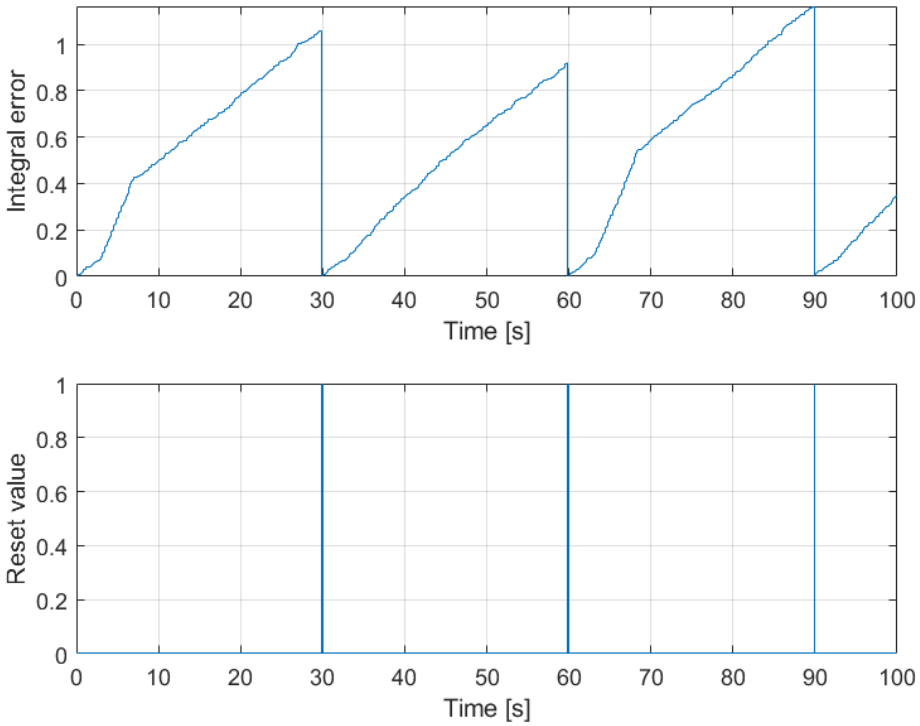


Figure 4.30. Integral error in nominal condition with Reset time of 30s.

4.4 Multi-model: Algorithm structure

With all the considerations made so far, it is possible to implement the final estimation algorithm. This part will represent the core of the MorePRO project and of the edge-device that will be implemented on the CNC machine to have an on-line SoH monitoring.
As far as the algorithm is concerned, it is mainly composed of three distinct parts:

- The **CNC model** that represents the dynamics equations governing the system;
- The **Extended Kalman Filter bank** where each filter is based on a different friction coefficient hypothesis and which get as input the same inputs applied to the model mentioned before and the outputs at the terminals produced by the latter and aims to estimate, based on the assigned β hypothesis, the acceleration at the terminals obtained by linearizing the CNC model around the specific working point.
- A **logic of decision and management** of the integral of the residual errors that include a reset, a *best model* choice and possibly the **reliability** of such choice.

In the following figure the general Simulink structure is depicted, summarizing all the components described above.

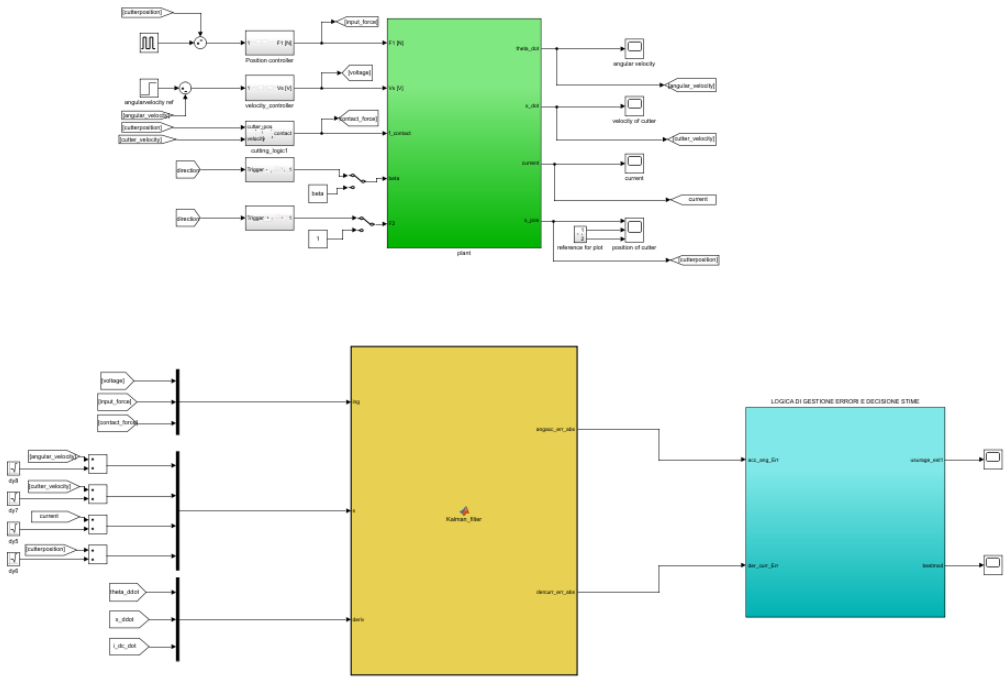


Figure 4.31. Simulink implementation of the algorithm.

In the green box are contained the dynamic behavior and the state equations of

the plant while in the yellow one is contained the entire EKF bank. The residual error estimation is than forwarded to the light blue block which represents the logic of error management, whose internal structure is represented in Figure 4.32.

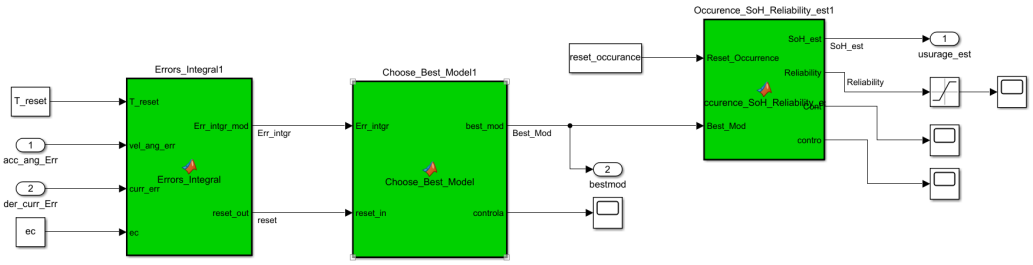


Figure 4.32. Simulink implementation of error logic.

In the next pages it is possible to find an in-depth explanation of the various mentioned structures.

4.4.1 Switching estimator

It is considered a problem of state estimation with a parameter variation in a finite range. The idea is to put N EKF in parallel, where each of them works with a different "wear condition" hypothesis. In particular, the friction coefficient β is chosen as switching parameter, obtaining N independent EKF, each with a fixed value of β.

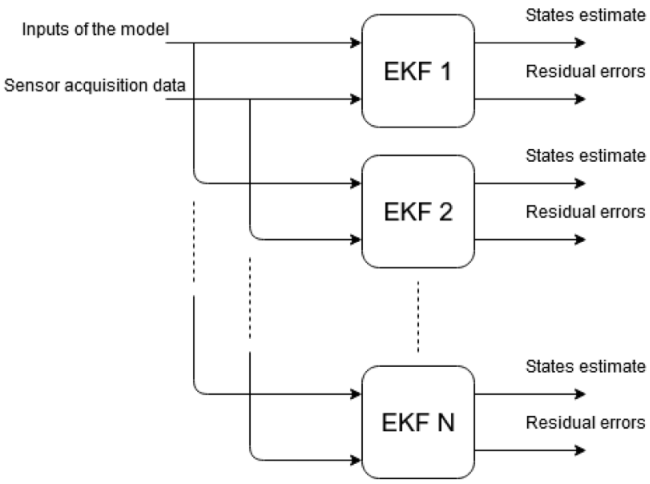


Figure 4.33. Switching estimator

The working principle of the switching estimator is:

- Each EKF will purpose its own estimate.
- Each EKF works with a different friction coefficient, switched over a finite set of values.

It has been decided to put **N=6 EKF** with a β range values from 0.1 to 1 linearly spaced as detailed in table 4.25. The aim is to identify the filter with the minimum residual errors, which means that filter which works with the friction coefficient more similar to the real one and that represents better the condition of the machine.

Filter	β value
#1	0.1
#2	0.28
#3	0.46
#4	0.64
#5	0.82
#6	1

Table 4.25. Friction coefficient associated to each filter

4.4.2 Best model choice

A key part of the algorithm is dealing with residual errors and extrapolate useful information from the, as the errors contains an intrinsic assessment of the EKF's quality. The underlying idea is to choose the model with the smallest residual error as the best model because it will have the closest friction coefficient to the real one with all other parameters unchanged.
In order to implement this management, a Matlab function has been developed which is dependent on both the input errors, the reset of the integral and also a "dwell time" which will be explained shortly. In principle, the choice works through these steps:

- Initially a **dwell time** is set, that is a period in which the function can not check the errors data because it is supposed as a period for a dynamic evolution of the system so that there are relevant data in the estimates.
- When there is a **reset of the integral**, the function cannot select the model because the data is not reliable as no past information are collected.
- After resetting the integral and the dwell time at which the transient has passed, the function analyses the errors data and assigns the model with the lowest error as the **best model**.

The Figure 4.34 shows that with the condition listed in Table 4.26 the best model is always the first because, as it should be, it is the one that has the same value

as the nominal one.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Friction coefficient	0.1	0.1
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.26. Test with nominal friction coefficient

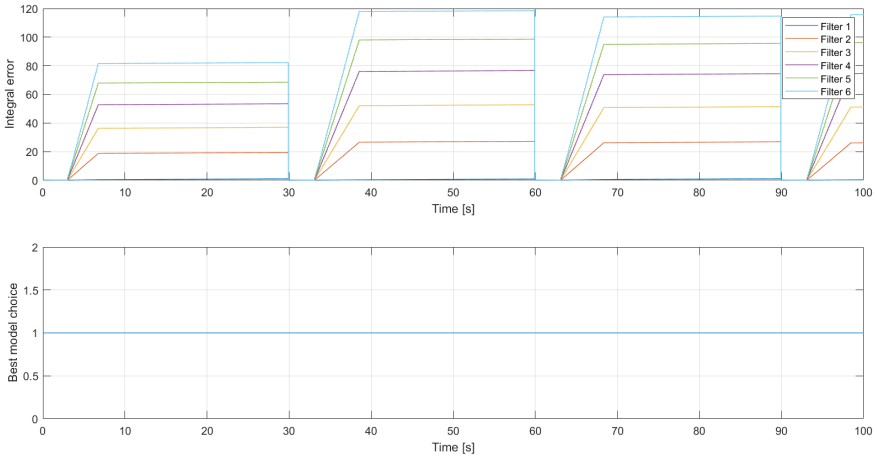


Figure 4.34. Best model choice with nominal condition

While testing a friction coefficient variation in the range $0.1 \div 0.5$ the best model changes according to the less residual error like shown in figure 4.35.

	Nominal value	Testing value
Angular velocity $[\frac{rad}{s^2}]$	210	210
Friction coefficient	0.1	0.1÷0.5
Duty cycle [%]	50	50
Number of cycles	4	4

Table 4.27. Test with friction coefficient variation

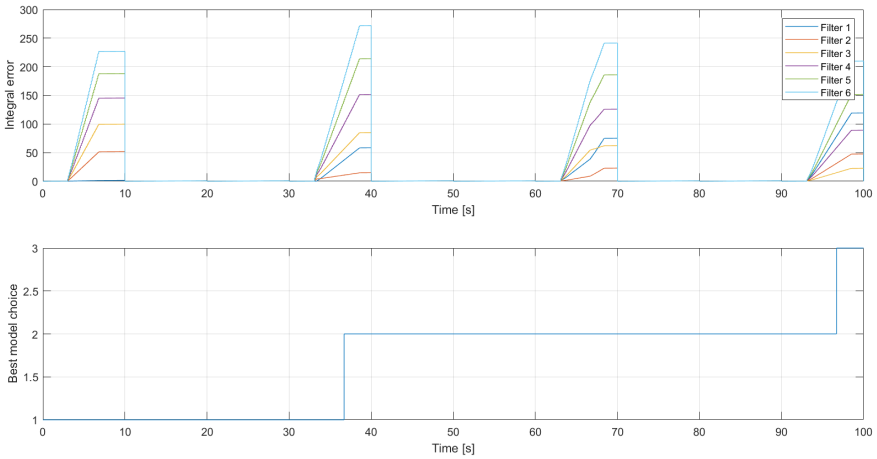


Figure 4.35. Best model choice with β variation

5 Data driven analysis

To increase the accuracy of the prediction estimate, during the development of the project it was decided to introduce a data-based analysis in parallel to what has been seen so far. In fact, the main idea is that there may be cases for which the edge device is not totally reliable due to various factors that can make the operation of the machine differ from the nominal one.
The goal is therefore to create classification/clustering algorithms that discretize between N different possible wear conditions in the inference space, choosing the most suitable for each situation. In particular, two different types of approaches will be addressed. A first in which on the sole basis of the data coming from the sensors, an algorithm will be built that performs the same work of the edge device, i.e. the estimate of the wear of the machine. A second one in which, also considering the labels in output from the devices mounted on the machines, the aim is to eliminate the small parametric variations present between different plants, obtaining a more reliable estimate.

5.1 Machine Learning

"It is said that an artificial system learns from experience E to some classes of task T and measurement of performance P it its performance in task T, measured by P, improve with experience E." [21]
As stated from the definition, Machine Learning means a set of algorithms and methodologies which goal is to provide an actual system capable of automatically learns new strategies from external stimuli, not relying on a pre-programmed logic.
All the ML algorithms are based on the same structures:

- Data source: an organized collection of field data.
- Feature: an individual and measurable property of the phenomenon under observation.
- Data element: a particular element of the data source univocally identified by the values assumed by the considered features.
- Model: a function that produces specific outputs given specific inputs.
- Model structure: a model with undefined parameters.
- Label: the specific results given by the model when a specific output is applied.
- Learner: it creates a specific model based on the class of the model structure.

The learning algorithms can be divided into two classes of belonging:

- **Supervised learning**: the algorithm learns knowledge from the data with their associated answers. In particular, we speak of regression when the tar-

get is a numerical value and of *classification* when the target is a qualitative variable, such as a class can be.

- **Unsupervised learning**: when the algorithm learns knowledge directly from the data structure, not knowing the answers. In practice, it tries to independently find specific patterns by rearranging the various features in a certain specific way. It is usually referred as *clustering*. [22]

5.1.1 Data pre-processing

Having an huge data set does not always imply having better results. In fact, bigger data brings adder complexity, that is why is important to have only significant data, organizing them in specific structures, such as matrices, filtering them and avoiding redundancies between features.[23] One of the most popular multivariate statistical technique is **Principal Component Analysis** and it is used by almost all scientific disciplines. This technique is used in order to extract the most important information from the data structure, simplify the description of the data set and to maintain only important informations compressing the data size. This is done by computing new variables, called principal components, as linear combinations of the original variables. In particular, the first principal component is required to have the largest possible variance, the second one is computed under the constraint of being orthogonal to the first component and the others are computed likewise. The values of these new variables for the observations are called factor scores, and these factors scores can be interpreted geometrically as the projections of the observations onto the principal components. [24]

5.1.2 Cross-validation

When resorting to a ML algorithm the best thing would be to have a different set of data in order to test the learner with respect to the training one. However, in industrial fields this is often unusual, since waiting for new data is infeasible in terms of costs and time. That is why a common random split between the data set is made, using a certain percentage fore training purposes (usually from 70 to 75) and the remaining one for testing. Nevertheless, a randomic split of the data could bring to bias problems. That is where cross-validation based on **k-folds** comes in, where a subdivision on equal size folds is made in order to use each fold at a turn for testing scope and the remaining ones for the training. The main idea is represented in Figure 5.1.

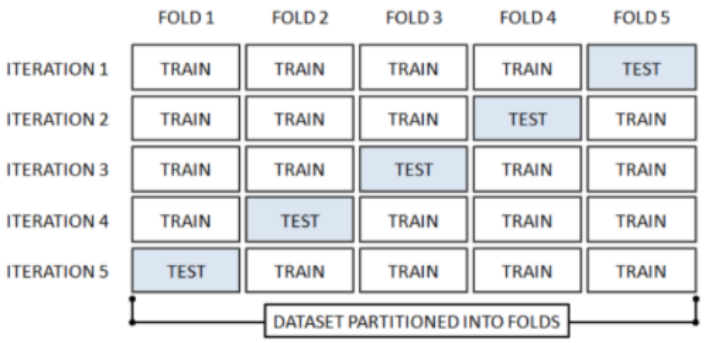


Figure 5.1. K-fold cross validation. [22]

5.2 Unsupervised SoH estimation

Recalling what stated above, in the first part of the work the goal is to obtain a clustering of the data coming from the sensors in different classes. Not having the real machine data available due to too long working/company times compared to a Thesis work, it has been decided to generate the data autonomously from the Simulink environment described in the previous paragraphs. It is clear that this approach will not be the same that will be pursued in the real implementation of the algorithm later on, however it can be useful to have a good starting point for future developments.

Therefore the starting point of the work is concentrated on a data generation that can be as similar to the possible data collected on the machine. It was therefore assumed to have sensors capable of measuring the angular and linear speed of the tool and the current of the DC motor as significant quantities. Furthermore, to have a quality of the data more similar to reality, a white error with zero mean and with a variance of 5% associated with these sensors was considered.

The data collection was made considering the work plan shown in chapter 5.1, where the only variable parameter between the various cases is the friction coefficient β (for obvious reasons described in Table 3.4) in order to have machines that are working with a different state of wear between them.

	Working value
Angular velocity $[\frac{rad}{s^2}]$	210
Position reference [m]	0.5
Friction coefficient	0.1-0.43-0.77-1.1
Duty cycle [%]	90
Number of cycles	200
Simulation time [s]	10000

Table 5.1. CNC working conditions.

5.2.1 Self-organizing maps

Self Organizing maps (SOM) are used as first clustering algorithm, which are essentially neural networks that allow to transform the data set into a topology-preserving 2D map. Basically, a SOM consists of a competitive layer which can classify a data set of vectors with any number of dimensions into as many classes as the layer has neurons. The neurons are arranged in a 2D topology, which allows the layer to form a representation of the distribution and a two-dimensional approximation of the topology of the data set. In Figures 5.2 and 5.3 the neurons topology of the used SOM and the neural network layers are represented.

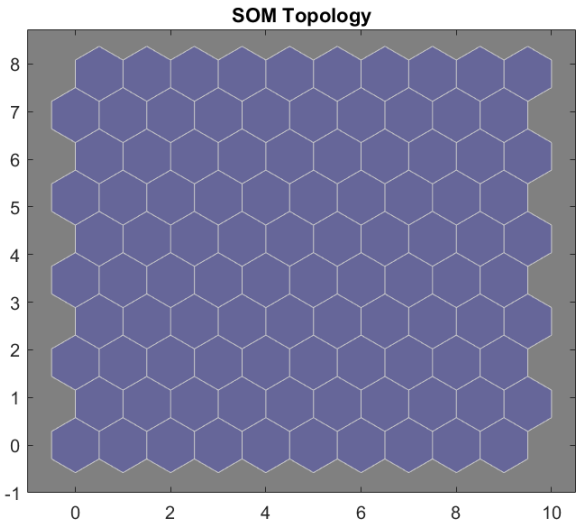


Figure 5.2. SOM topology.

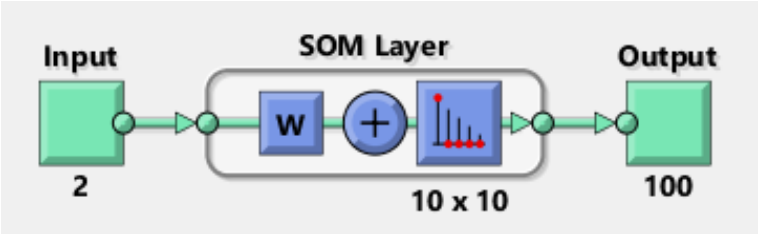


Figure 5.3. SOM layers.

Once implemented the neural network, it is possible to train it using unsupervised weight and bias learning rules with batch updates, where the weights and biases are updated at the end of an entire pass through the input data. For SOM training, the weight vector associated with each neuron moves to become the center of a cluster of input vectors. In addition, neurons that are adjacent to each other in the topology should also move close to each other in the input space, therefore it is possible to visualize a high-dimensional inputs space in the two dimensions of the network topology. Once trained the algorithm for 200 epochs, it is possible to evaluate the quality of the network by having a look at some of the main relevant plots of SOM:

- **SOM input planes**: This plot shows a weight plane for each element of the input vector. They are visualizations of the weights that connect each input to each of the neurons, with darker colors that represent larger weights. If the connection patterns of two inputs were very similar, you can assume that the inputs are highly correlated. In this case, input 1 (the angular speed) seems to have the same wright on almost all neurons thus it could be a good idea to remove this input since it does not brings relevant information.

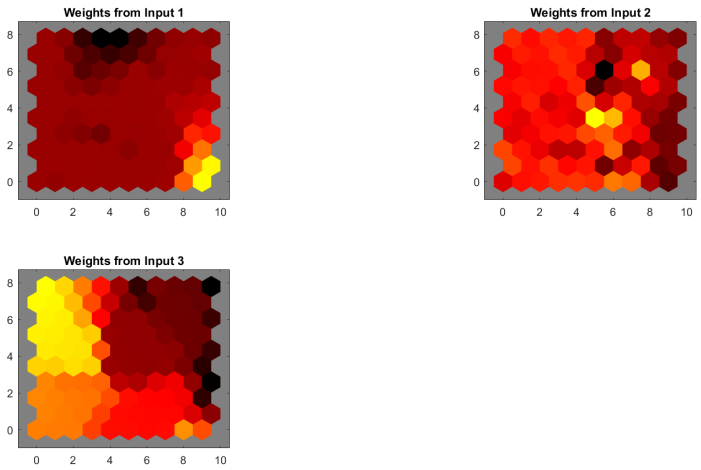


Figure 5.4. SOM input planes with all features.

- **SOM sample hits**: This plot shows the neuron locations in the topology, and indicates how many of the training data are associated with each of the neurons (cluster centers). In this case it is already possible to distinguish between four main agglomerates, which basically represents the four wear conditions.

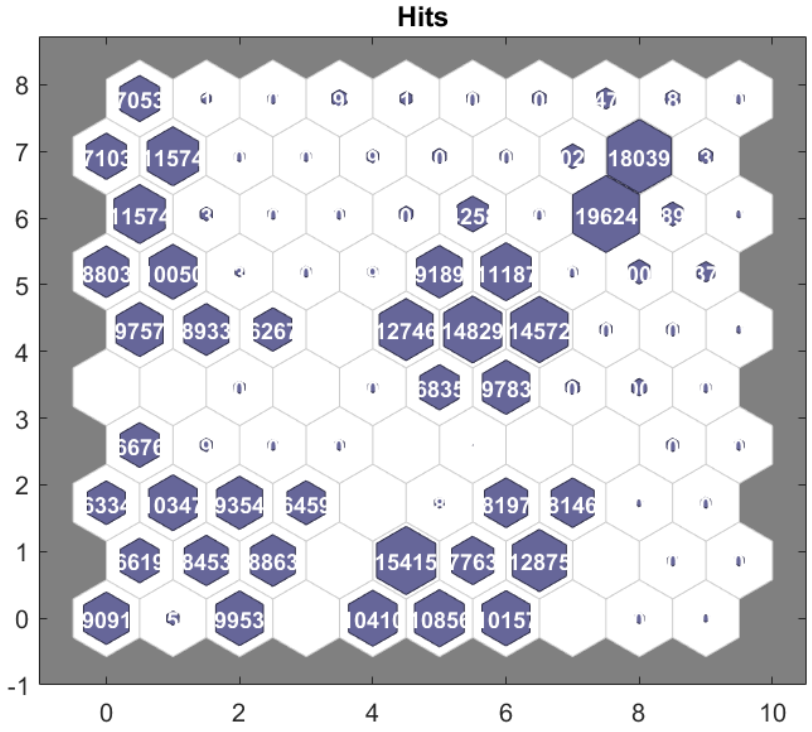


Figure 5.5. SOM sample hits with all features.

- **SOM neighbor distances**: In this figure, the blue hexagons represent the neurons. The red lines connect neighboring neurons. The colors in the regions containing the red lines indicate the distances between neurons. The darker colors represent larger distances, and the lighter colors represent smaller distances. In this case a band of darker segments seems to create four yellow cluster, even if in some region the separation is not so marked.

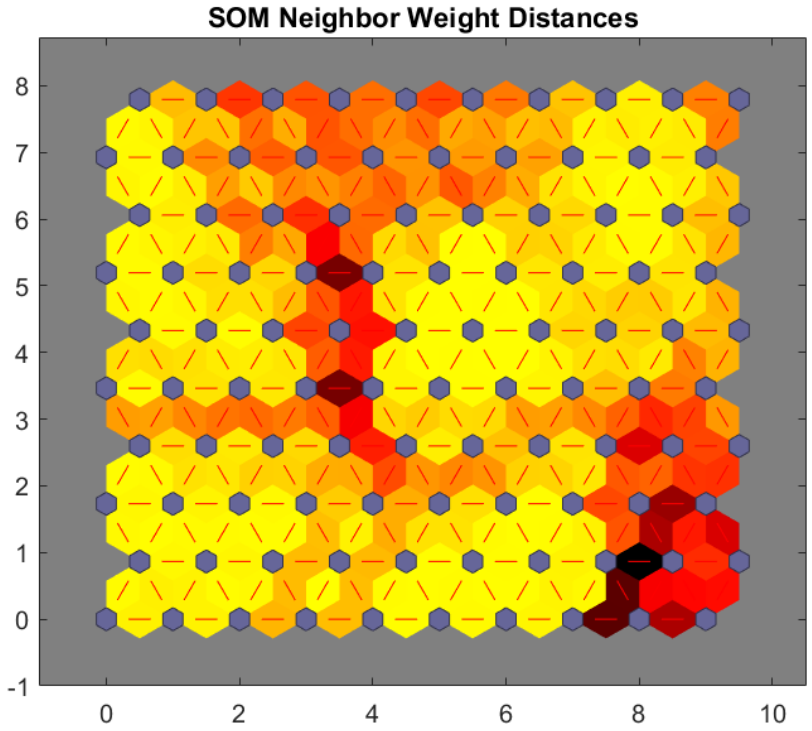


Figure 5.6. SOM neighbor distances with all features.

Recalling what just seen, a second retraining phase using only linear speed and current as features of the experiment is made in order to see if better performance of the network are obtained. The same series of plots as before are used as a measure of quality:

- **SOM input planes**: in this case the two feature brings very uncorrelated information, thus what will be expected are better results.

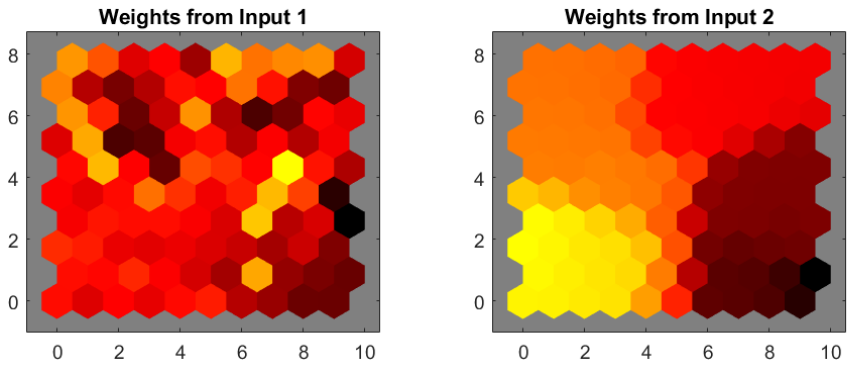


Figure 5.7. SOM input planes.

- **SOM sample hits**: also in this case it is possible to see a clear separation between four main clusters, one for each corner.

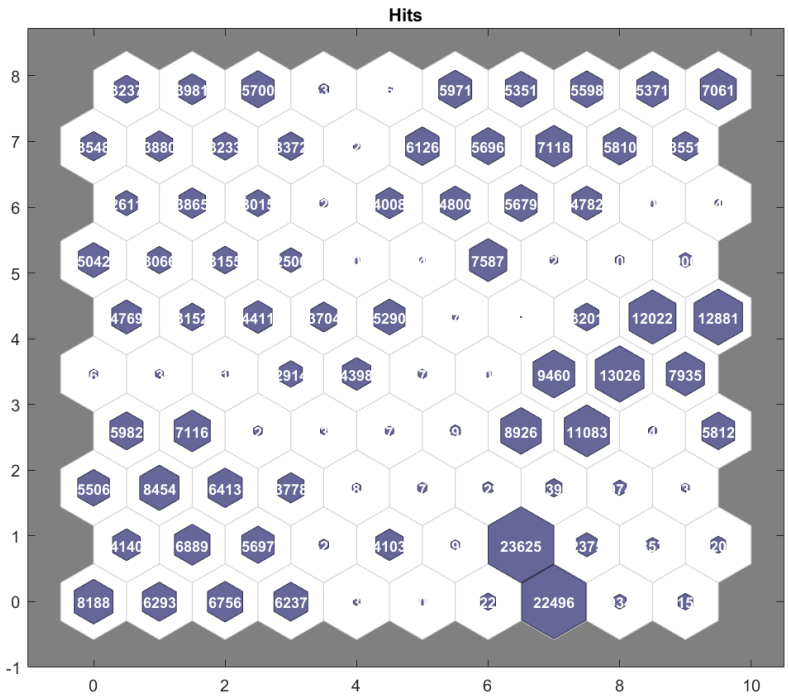


Figure 5.8. SOM sample hits.

- **SOM neighbor distances**: with respect to the previous case a better distance between the four clusters is obtained, which allows to state that a better forecast of the wear will be expected.

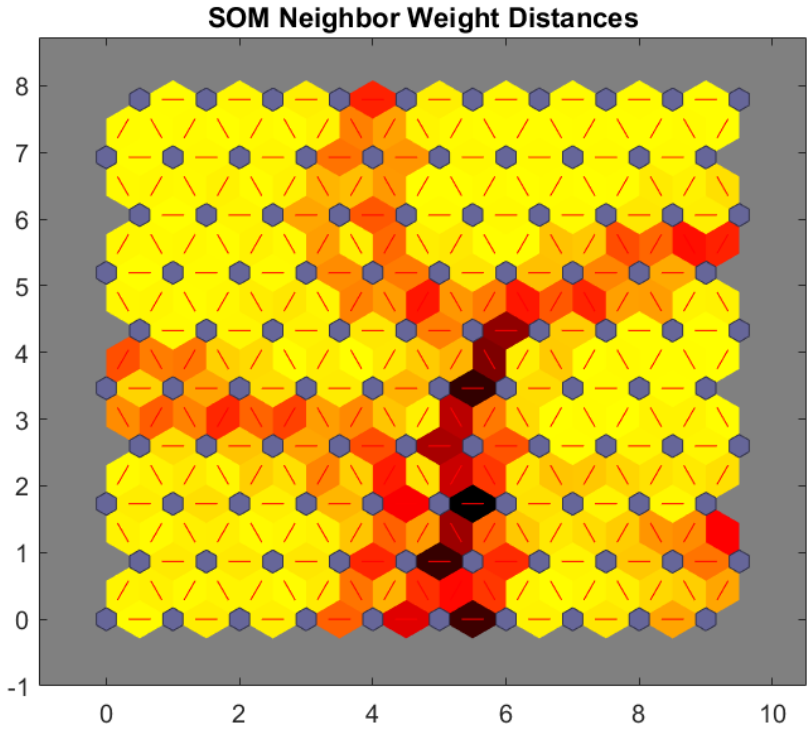


Figure 5.9. SOM neighbor distances

In order to see if more accurate network can be built it is possible to try to use PCA. Thus, with respect to the previous data set a product with the coefficients of table 5.2 is made, where each column of contains coefficients for one principal component. The columns are in descending order in terms of component variance.

$1.8 * 10^{-4}$	0.99
0.99	$-1.8 * 10^{-4}$

Table 5.2. PCA coefficient.

As before an evaluation analysis of the various plot, in order to see if a better implementation of the clustering neural net can be obtained:

- **SOM sample hits**: no relevant differences are obtained with respect to the previous case in terms of sample hitting.

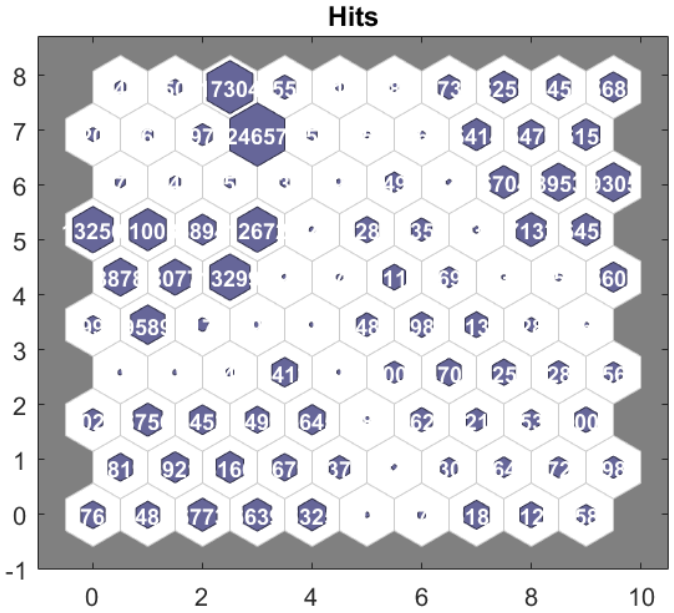


Figure 5.10. SOM sample hits with PCA.

- **SOM neighbor distances**: in some way the distinction between neighbor clusters is less clear.

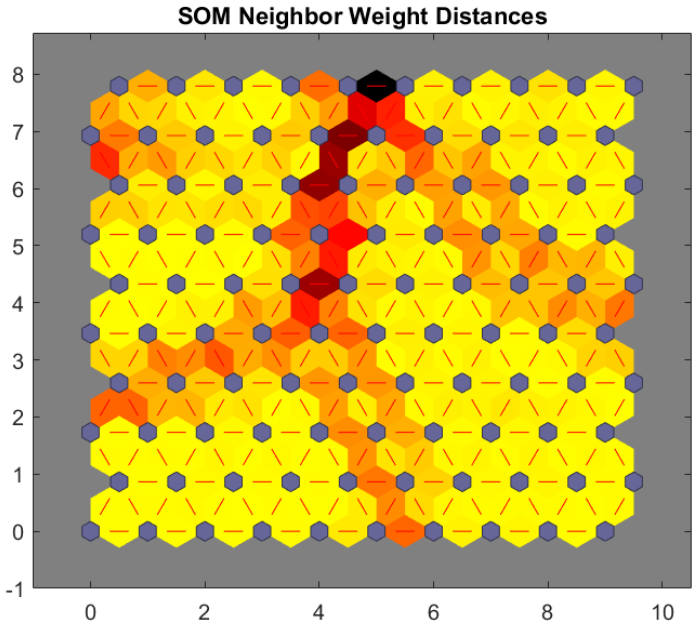


Figure 5.11. SOM neighbor distances with PCA.

After this analysis it can be concluded that SOMs are an excellent tool for clustering a data set not knowing the answers. Furthermore, it is convenient not to consider the angular velocity, thus obtaining both a better algorithm and less heavy from the computational point of view. Moreover, since the PCA analysis does not bring any actual improvement, it is advisable not to take it into account in the development phase for the same reasons mentioned above.

5.2.2 K-means approach

As second clustering algorithm **K-means** is exploited, which basically is a partitioning method that treats observations in the data set as objects having locations and distances from each other. It divide the objects into K mutually exclusive clusters, in a way that data within each cluster are as close to each other as possible, and as far from objects in other clusters as possible. Each group is characterized by its centroid, or center point. Analyzing in more detail the operation of the algorithm step by step:

- After setting the number of clusters, the algorithm chooses k random examples which will be the centroids.
- The algorithm assigns all available data to the available clusters based on the Square Euclidean distance from each centroid and assigning them to the nearest one.
- After assigning all the examples to the various clusters, the algorithm recalculates the centroids by leveling all the data present in the various groups.
- As a last step, if the position of the centroids has not changed much the algorithm returns the solution. Otherwise, he repeats his previous steps until he reaches a better solution.

Of course, the algorithm is trained using the same data collected for the SOMs in order to compare the results between the two clustering algorithms in the best possible way.
Before starting the actual training, it is useful to analyze through a silhouette plot how much the various clusters are really separated. Indeed, this plot allows to see how close each point in one cluster is to points in neighboring clusters. Taking a look at Figure 5.12 it is possible to see how most of the points in all four clusters have a silhouette value greater than 0.8, indicating a well-marked separation between the various clusters.

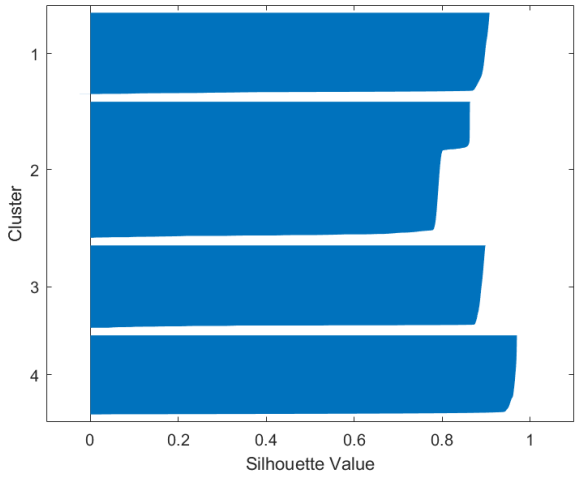


Figure 5.12. Silhouette plot.

Thus, after setting the number of clusters to four, to avoid the fact that the solution strongly depends on the starting point it is possible to replicate the training of the algorithm for a certain number of times, i.e. 5, and then pick the solution that gives the best results in terms of lowest total sum of distances among all the replicates.

It is possible now to look at the results of the training process through a scatter plot, which basically allows to represent all the experiments and their subdivision, as can be seen in Figure 5.13, where each color represents one of the fours different clusters.

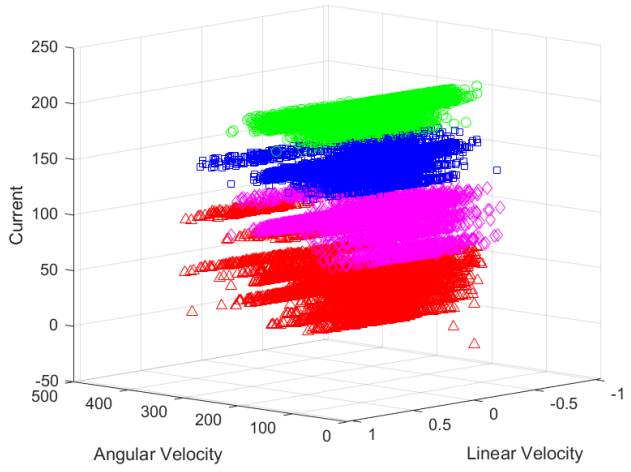


Figure 5.13. Scatter plot.

Since, as stated before, data are generated from a Simulink environment, it is

possible to estimate the accuracy of the algorithm by comparing the "real" subdivision of the experiments with the obtained one. In this case, the algorithm is capable of finding the correct match-up with the 88.4%

5.3 Supervised learning algorithm

As seen before, the main difference of having a supervised algorithm is having available not only the features of the system but also its answers. In this case, therefore, in addition to the angular and linear velocity and current coming from the sensors, the estimate of wear from the edge device is considered as the system output.

The main idea behind using a supervised algorithm for this job is to collect data from various machines in the production plant, considering the fact that each machine is slightly different on a parametric level than the others. This assumption allows to have more realistic data and to eliminate the estimation errors of the edge device due to a mismatch between plant and filter bank. Furthermore, as before it is assumed that for each CNC machine data are taken for four different configurations of wear (and therefore of the friction coefficient). The algorithm will therefore be based on the five machines listed in Table 5.3..

	1st	2nd	3rd	4th	5th
Mass [kg]	3	2.9	3.1	2.85	3.05
Resistance [$k\Omega$]	0.6	0.7	0.8	0.8	0.5
Radius [m]	0.3	0.29	0.28	0.31	0.32
Torque constant	1.5	1.5	1.3	1.6	1.4
Inductance [mH]	0.1	0.15	0.05	0.12	0.11
Voltage constant	0.2	0.2	0.25	0.15	0.3

Table 5.3. CNC parameters of different machines.

5.3.1 K-Nearest Neighbor

Given a set X of n points and a distance function, k-nearest neighbor (kNN) search lets you find the k closest points in X to a query point or set of points Y. The kNN search technique and kNN-based algorithms are widely used as benchmark learning rules. The relative simplicity of the kNN search technique makes it easy to compare the results from other classification techniques to kNN results. As distance metrics to categorize query points based on their distance to points in the training data set the standardized euclidean is used, where given an m_x-by-n data matrix X, which is treated as m_x (1-by-n) row vectors $x_1, x_2, ..., x_{mx}$, and an m_y-by-n data matrix Y, which is treated as m_y (1-by-n) row vectors $y_1, y_2, ..., y_{my}$

it is computed as:

$$d_{st}^2 = (x_s - y_t)\, V^{-1}\, (x_s - y_t)' \tag{5.1}$$

where V is the n-by-n diagonal matrix whose j_{th} diagonal element is $S(j)^2$, where S is a vector of scaling factors for each dimension.

For this approach the cross-validation exposed in chapter 5.1.2 will be used, making use of 10 sub-folds of the data set. In the first part all the three features will be used for the training/testing of the algorithm and by using a **confusion matrix** an evaluation of the quality of the classification will be made, which is a special matrix where the rows correspond to the true class and the columns correspond to the predicted class. Diagonal and off-diagonal cells correspond to correctly and incorrectly classified observations, respectively. Moreover, in order to improve the quality of the algorithm several tests are exploited in order to find the best number of neighbors (**k**). This parameter has a very high impact on the network, indeed it indicates how many of the objects around the considering point will be used to assign the class. In example, with reference to figure 5.14, if k=3 only the three nearest object will be considered, assigning the value to the triangle class, on the contrary, if k=5 the five nearest objects will be view, classifying the value as a square.

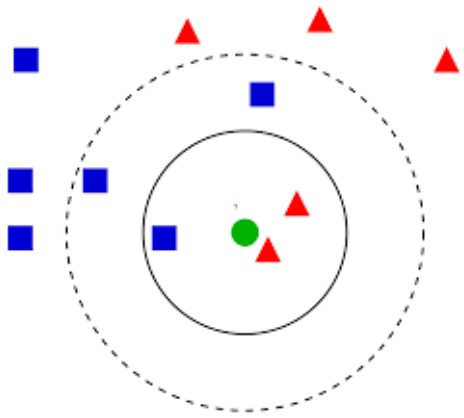


Figure 5.14. kNN algorithm approach.

- **k=1**

  Using one as number of neighbors an accuracy of the 91.2% is obtained, as can be seen in the figure below.

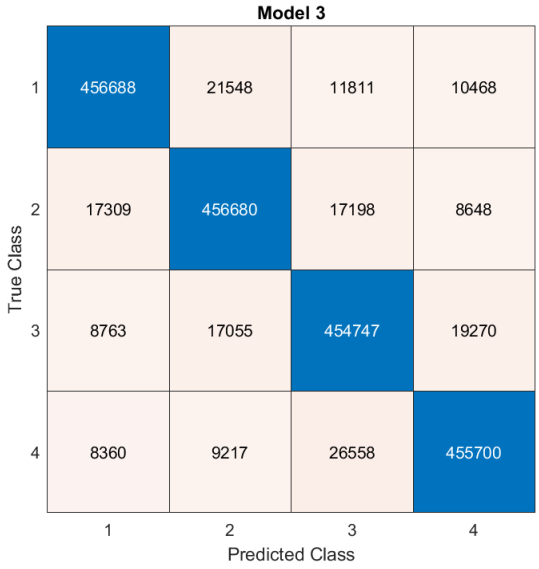


Figure 5.15. kNN with k=1.

- **k=2-3**

  When considering 2/3 number of neighbors an almost similar results is obtained, having a 91% of confidence.

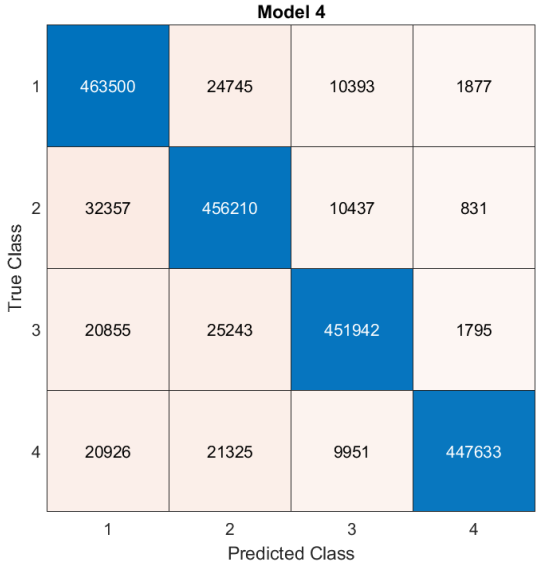


Figure 5.16. kNN with k=2-3.

- **k=4**

  Increasing the parameter k to four better results are obtained (91.1%) with respect to the previous one. However, the better classification it is still that one considering only one neighbor for the algorithm.

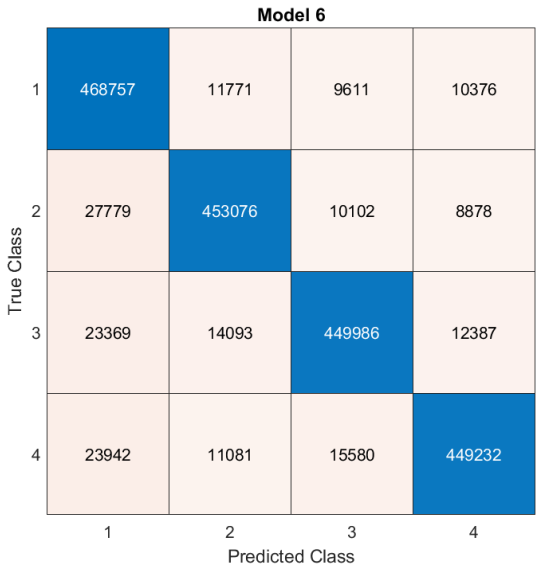


Figure 5.17. kNN with k=4.

- **k=5**

  Keeping on increasing the parameter a degradation of the performance is obtained. Indeed, the accuracy is now 90.6%, the lowest one.

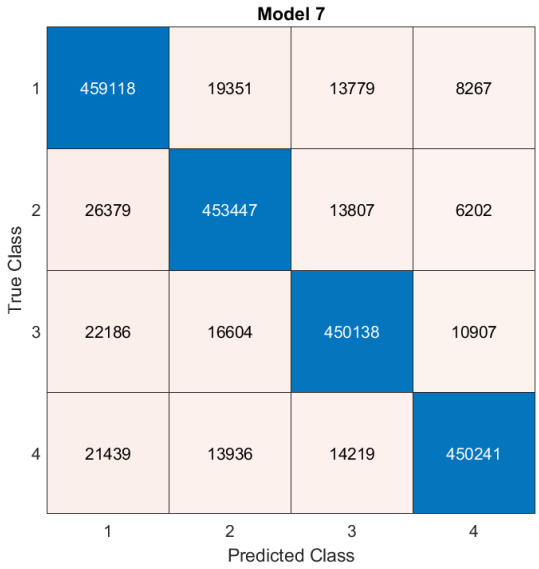


Figure 5.18. kNN with k=5.

As a last step for this algorithm it was decided to try to increase the performance using a PCA. In particular, it was decided to keep a sufficient number of components to cover 95% of the variance. By doing so, only the first 2 features will be considered, with the first that explains the 92.7% of the variance, and the second 7.9%. Once this preliminary analysis of the data has been made, it is possible to proceed as before to find with which value of k the algorithm works more accurately and if, by applying the PCA, there are significant improvements on the classification.

- **k=1**

  Using one as number of neighbors an accuracy of the 93.3% is obtained, as can be seen in the figure below.

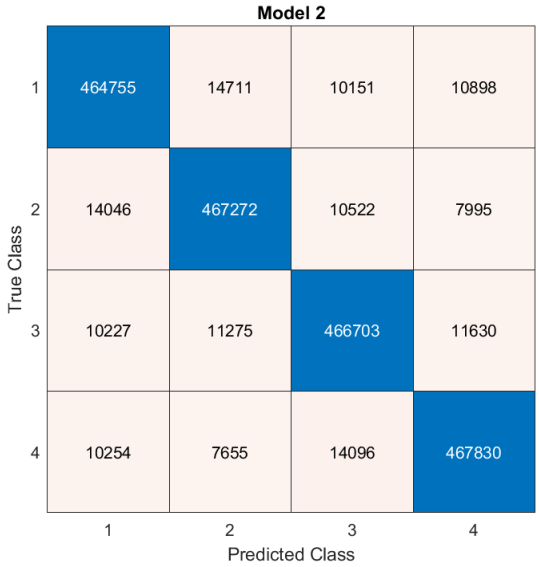


Figure 5.19. kNN with PCA and k=1.

- **k=2**

  When k is set equal to 2 no relevant differences are seen with respect to the previous case. Indeed, the accuracy is still 93.3%.

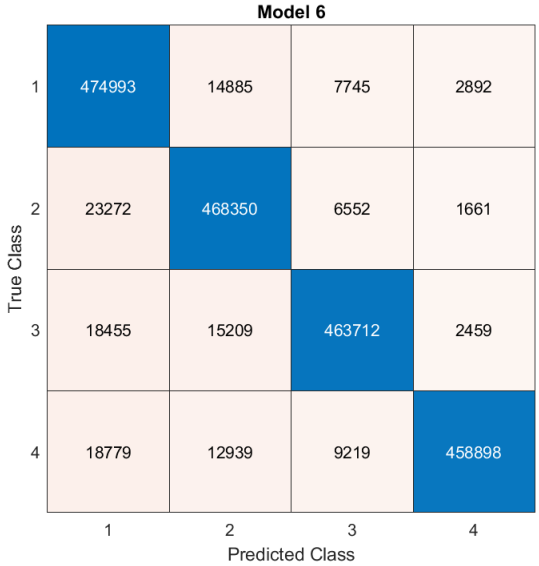


Figure 5.20. kNN with PCA abd k=2.

- **k=3**

  Increasing to three, a very small improvement is obtained (93.4%).

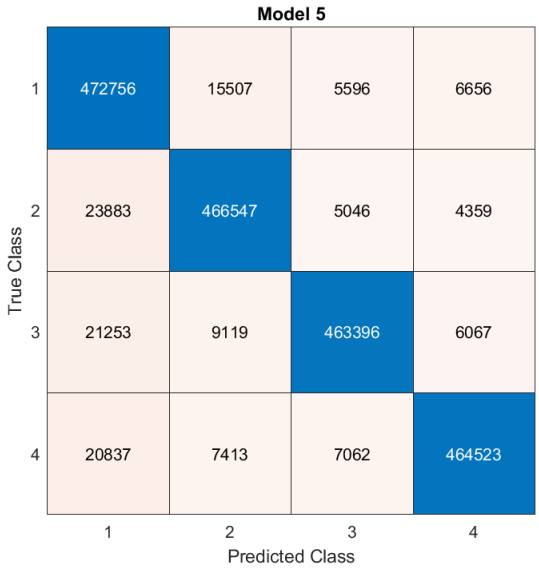


Figure 5.21. kNN with PCA and k=3.

- **k=4**

  The same results as before as obtained, not having any better classification.

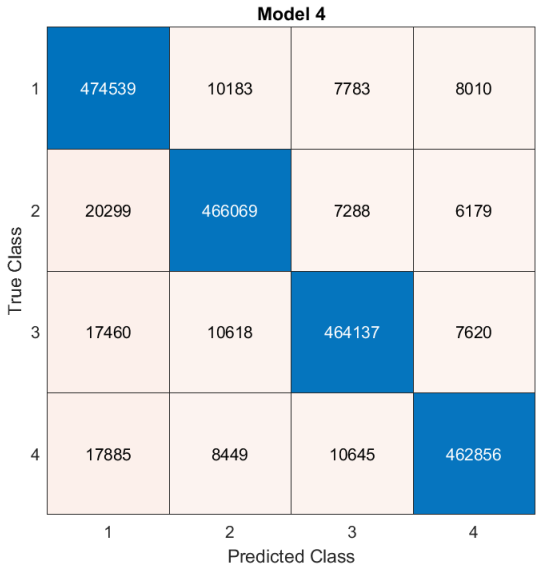


Figure 5.22. kNN with PCA and k=4.

- **k=5**

  Increasing more the number of neighbors a worst classification is obtained (92.8%).

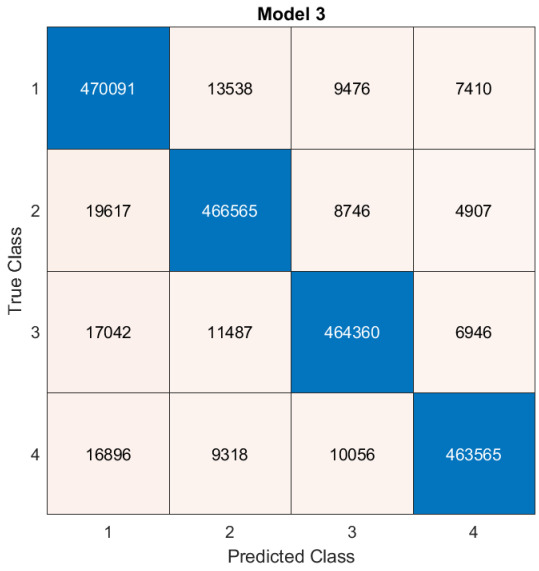


Figure 5.23. kNN with PCA and k=5.

To conclude, it can be said that it is certainly advisable to apply a PCA to the data in question, as the reliability of the algorithm seems to increase by a couple of percentage points. Furthermore, the best results are obtained when k is equal to three/four, it is therefore advisable to set it to three in order to have a less demanding algorithm from the computational point of view.

5.3.2 Other classification algorithms

Before identifying KNN as the best classification algorithm for the purpose in question, a brief analysis was made of other possible algorithms. All this was possible using the tools that MATLAB makes available. In particular, the analysis was carried out considering for each case the relative confusion matrix, as well as the percentage of reliability of the same.

- **Decision Trees**

  This is a non-parametric supervised learning method whose goal is to predict the value of target variables using a tree-like model, based on very simple rules deduced from data itself. Usually this kind of classification algorithm is used due to its simplicity, since it is able to handle both numerical and categorical data and does not require to manage data before feeding them to the algorithm. On the contrary, it is possible that a over-complex tree is created, not generalising data in a correct day, as well as they are not able to deal with small variations in a correct way. Thus, considering the data generated by the five machines, the training of the algorithm generate the confusion matrix in Figure 5.24, where an accuracy of 90.3% is reached.

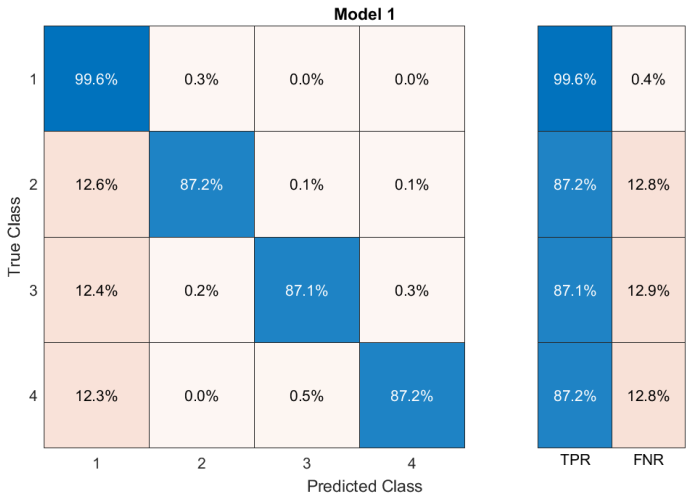


Figure 5.24. Decision tree confusion matrix.

In the same figure the **TP** and **FN** rates are reported, which respectively are the proportion of data that are correctly identified and incorrectly assigned for each class.

- **Linear & Quadratic Discriminant Analysis**

  Linear and Quadratic Discriminant Analysis are two classifier that make use, as their name suggest, a linear and a quadratic decision surface, respectively.

Both LDA and QDA make strong assumption for the construction of their algorithms, in particular:

- The predictor variables are drawn from a multivariate Gaussian distribution.
- In LDA equality of covariances among the predictor variables across each class is assumed.

Furthermore, it is very important to know that QDA is much more flexible than LDA, and so has substantially higher variance. Thus, DA is recommended if the training set is very large, so that the variance of the classifier is not a major concern, or if the assumption of a common covariance matrix is clearly untenable. In contrast, LDA tends to be a better bet than QDA if there are relatively few training observations and so reducing variance is crucial.
Indicating with x each training sample, with y the response variable and with k a class, predictions can be obtained by using Bayes' rule:

$$P(y = k \mid x) = \frac{P(x \mid y = k)P(y = k)}{P(x)} = \frac{P(x \mid y = k)P(y = k)}{\sum_l P(x \mid y = l) \cdot P(y = l)} \quad (5.2)$$

selecting the class k which maximizes this posterior probability. Thus, since $P(x \mid y)$ must be modeled as a multivariate Gaussian distribution with density:

$$P(x \mid y = k) = \frac{1}{(2\pi)^{d/2} \left|\Sigma_k\right|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k)\right) \quad (5.3)$$

where d is the number of features, Σ_k the covariance matrix and μ_k a class-specific mean vector. Thus, according to what stated, the log of the posterior is:

$$\begin{aligned} \log P(y = k \mid x) &= \log P(x \mid y = k) + \log P(y = k) + Cst \\ &= -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k) + \log P(y = k) + Cst \end{aligned} \quad (5.4)$$

The predicted class is the one that maximises this log-posterior, where Cst corresponds to the denominator of $P(x)$. Moreover, since for LDA each class is assumed to share the same covariance matrix, this reduce to:

$$\log P(y = k \mid x) = -\frac{1}{2}(x - \mu_k)^t \Sigma^{-1} (x - \mu_k) + \log P(y = k) + Cst \quad (5.5)$$

Thus, applying to the case study of this Thesis, the following confusion matrices are obtained for LDA and QDA, where an accuracy of 77.8% and 85.9% is reached, respectively.

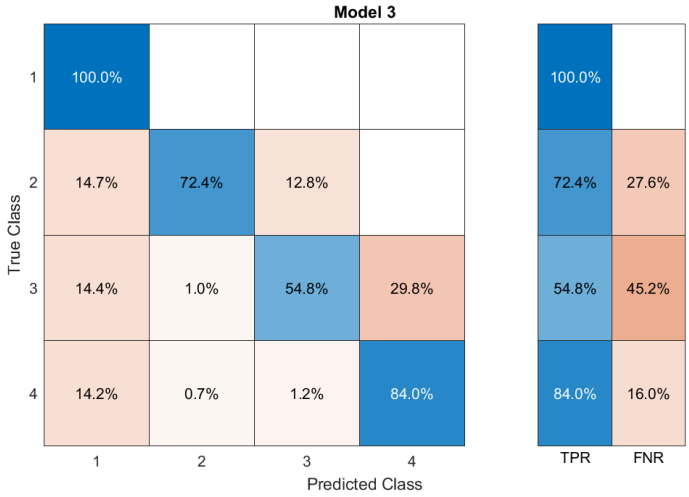


Figure 5.25. Linear discriminant confusion matrix.

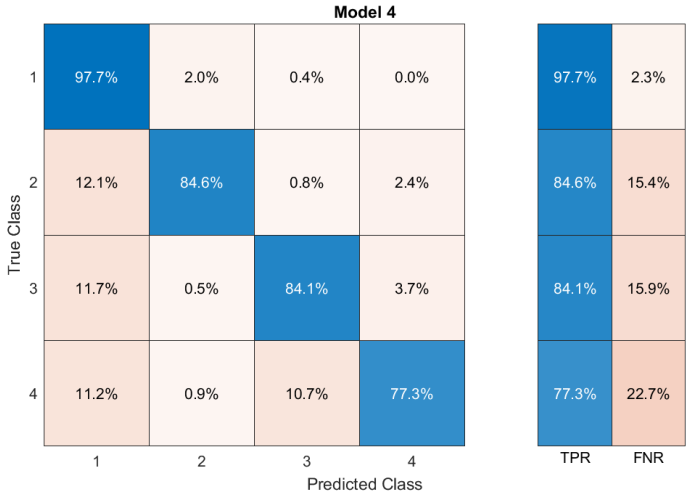


Figure 5.26. Quadratic discriminant confusion matrix.

- **Gaussian Naive-Bayes**

  Naive-Bayes are essentially based on the application of the Bayes' theorem with the "naive" assumption of strong independence between the features. In particular, given the class variable y and dependent feature vector x_1 through x_n, the Bayes' theorem states the following relationship:

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y) P\left(x_1, \ldots, x_n \mid y\right)}{P\left(x_1, \ldots, x_n\right)} \tag{5.6}$$

where, assuming the conditional independence:

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y)\prod_{i=1}^{n} P\left(x_i \mid y\right)}{P\left(x_1, \ldots, x_n\right)} \tag{5.7}$$

finally, since $P\left(x_1, \ldots, x_n\right)$ is constant given the input, it is possible to use the following classification rule:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right) \tag{5.8}$$

In this case of study, since the data available are continuous ones, it is possible to assume that the values associated to each class are distributed according to a Gaussian distribution. Thus, after the segmentation, it is possible to compute the mean and the variance in each class and getting to the final likelihood of the features:

$$P\left(x_i \mid y\right) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{\left(x_i - \mu_y\right)^2}{2\sigma_y^2}\right) \tag{5.9}$$

Applying the theory seen above, it is possible to finally train the algorithm obtaining the confusion matrix in Figure 5.27, where an accuracy of 85.9% is reached.

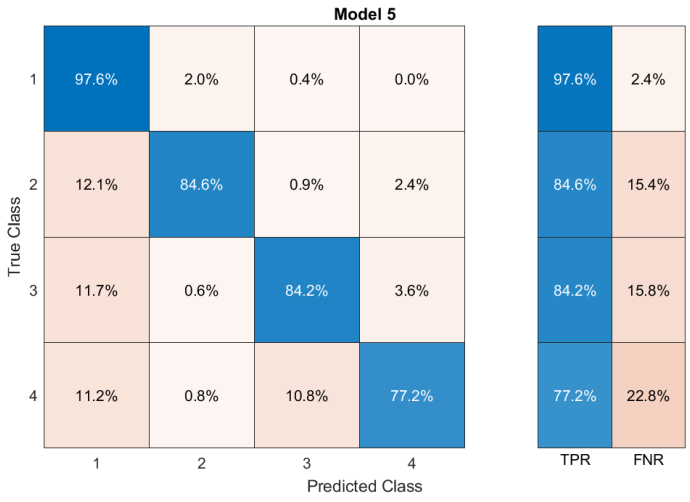


Figure 5.27. Gaussian Naive-Bayes confusion matrix.

6 Conclusions and Future works

Finally, it is possible to draft all the principal results obtained during this Thesis work developed as part of the MorePRO team at brain Technologies srl, whose main objective was the development an edge-computing device, capable of providing the most reliable on-line estimate of the state of wear of CNC end-effectors. In particular, going to follow the same path described in the various chapters the following contributions and conclusions were obtained:

- Firstly, it has been seen how the application of the algorithm already developed by brain Technologies in the ERMES project for the SoH estimation of batter system by adopting a new switching multi modal estimation technique is suitable also for a mechanical case, which concerns the wear prediction of a dynamical model resembling the work carried out by a CNC machine. This has be proven by choosing the friction coefficient as main representative for discriminating different wear conditions of the plant under assumption, and carrying out a brief analysis on the residual errors of the Extended Kalman Filters bank implemented.
- Secondly, it has been proven how relying on Machine Learning algorithms can be very useful for this case study. Indeed, a first implementation of unsupervised techniques, such as Self-Organizing Maps, could really help to have a practical feedback on how correct the estimation coming from an edge-device using the multi modal approach is. Furthermore, the application of supervised techniques, i.e. K-Nearest Neighbour, could help to leverage errors coming from a single device when more machines are working on the same process.

6.1 Future works

This Thesis has shown the potential benefits of applying the proposed estimation techniques in the field of end-effectors state of health.
As regards the first part of the work, it is clear that one of the main key points of the algorithm is the implementation of the physical model. As a consequence one of the possible line of action is the research of a dynamical model that better describes the working machine on which device will be mounted and at the same time not complicate it too much to keep the overall computational costs of the algorithm as low as possible.
In addition, another improvement that could be made, which has been the subject of study of other other students working on the same project, is to find a way to better describe the friction coefficient, not considering it as a global parameter β but going to see which are the parameters that mostly affects the wear condition, such as temperature, chip load, cutting angle, and so on could be.
Moreover, recalling the fact that all the data-driven analysis was carried out assuming auto-generated data from a Simulink environment, the main consequent

line of action must be to collect real data from the plant in order to validate/refute all the hypothesis/results gained during this Thesis work. In this way, it is possible to see if a Machine Learning approach is suitable to improve the reliability of the algorithm developed on the edge-device.

Appendix A

Boxplots

Boxplots are graphic representation used to describe the distribution of a sample by means of dispersion and position indices.

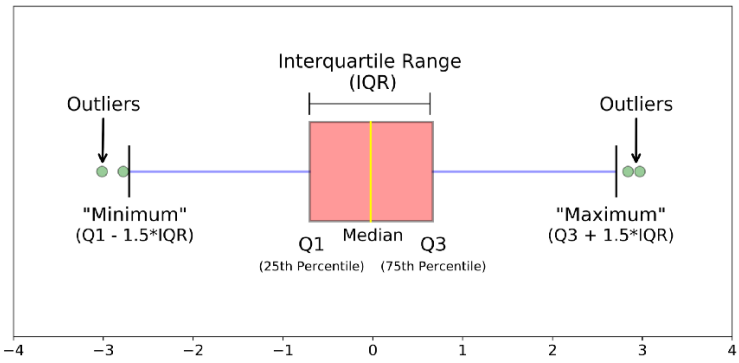


Figure 6.1. Boxplot represntation.

As can be seen in figure, the box range goes from the first to the third quartile and contains the median. Two segments come out from the rectangle and they are delimited by the minimum and maximum values, also called outliers.

References

[1] Hashemian, H. M ; Bean, W. C *State-of-the-Art Predictive Maintenance Techniques.*

[2] Li, Yulong; Zhang, Xiaogang; Ran, Yan; Zhang, Wei; Zhang, Genbao. *Reliability and Modal Analysis of Key Meta-Action Unit for CNC Machine Tool.*.

[3] Knuth: Computers and Typesetting,
`http://www-cs-faculty.stanford.edu/~uno/abcde.html`

[4] Preet Joy, Adel Mhamdi, Alexander Mitsos. *Optimization-based observability analysis.* Aachener Verfahrenstechnik - Process Systems Engineering (SVT), RWTH Aachen University, Forckenbeckstrasse 51, Aachen 52074, Germany

[5] Feng, Guohu ; Huang, Xinsheng *Observability analysis of navigation system using point-based visual and inertial sensors* ISSN: 0030-4026 EISSN: 1618-1336 DOI: 10.1016/j.ijleo.2013.08.004

[6] Gouriveau Rafael, Medjaher Kamal, Zerhouni Noureddine *From prognostic and health systems management to predictive maintenance 1: Monitoring and Prognostics* Newark: John Wiley & Sons, Incorporated, 2016

[7] Aleksandra Marjanović, Goran Kvaščev, Predrag Tadić, Željko Đurović. *Applications of Predictive Maintenance Techniques in Industrial Systems* SERBIAN JOURNAL OF ELECTRICAL ENGINEERING Vol. 8, No. 3, November 2011, 263-279

[8] Sayed-Mouchaweh Moamar, Lughofer Edwin. *Predictive maintenance in dynamic systems: advanced methods, decision support tools and real-world applications* Springer, 2019

[9] Gregory L. Plett. *Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 1. Background* Journal of Power Sources 134 (2004) 252–261

[10] Daga A. P., Garibaldi L. *Machine vibration monitoring for diagnostics through hypothesis testing* MDPI AG, 2019

[11] Bibin Pattel, Hoseinali Borhan, Sohel Anwar. *An evaluation of the moving horizon estimation algorithm for online estimation of battery state of charge and state of health* Proceedings of the ASME 2014 International Mechanical Engineering Congress and Exposition, IMECE2014, November 14-20, 2014, Montreal, Quebec, Canada

[12] Pier Giuseppe Giribone, Ottavio Caligaris, Simone Fioribello, Simone Ligato. *Implementazione della Fuzzy Logic per la gestione ottimale del portafoglio: la modellizzazione dell'avversione al rischio di un investitore attraverso tecniche di softcomputing* September 2016, DOI: 10.47473/2020rmm0061

[13] *Reshaping the world with fuzzy logic* WorldQuant Perspectives, April 2018

[14] S. Khaleghi, Y. Firouz, J. Van Mierlo, P. Van den Bossche. *Developing a real-time data-driven battery health diagnosis method, using time and frequency domain condition indicators*;Department of Mobility, Logistics and Automotive Technology Research Centre, Vrije Universiteit Brussel, Pleinlaan 2, Brussels 1050, Belgium Flanders Make, 3001 Heverlee, Belgium

[15] Davide Faverato. *Virtual Sensing for the Estimation of the State of Health of batteries* October 2020, Politecnico di Torino

[16] J. Kallrath, ed. *Modeling Languages in Mathematical Optimization*, Applied Optimization, Vol. 88, Kluwer, Boston 2004.

[17] Zoran Pandilov, Filip Gorski, Damian Grajewski, Damir Ciglar, Tihomir Mulc, Miho Klaic, Andrzej Milecki, Amadeusz Nowak *Virtual modelling and simulation of a CNC machine feed drive system*, TRANSACTIONS OF FAMENA XXXIX-4 (2015),ISSN 1333-1124, eISSN 1849-1391

[18] Dora Sabau, Mirela Dobra *System identification and control analysis for CNC machines*, 978-1-5386-2205-6/18/2018 IEEE

[19] *MATLAB guidelines*, Application-specific guidelines for model architecture, design, and configuration R2020b. URL[https://it.mathworks.com/help/simulink/modeling-guidelines.html]

[20] Harvey M. Wagne. *Global Sensitivity Analysis*; 1 Dec 1995h.

[21] Mitchell Tom M, Carbonell Jaime G., Michalski Ryszard S. *Machine Learning: a guide to current research.* Norwell, Kluwer, 1988

[22] John Paul Mueller, Luca Massaron. *Machine Learning for dummies.* John Wiley & Sons, Inc., 2016.

[23] MathWorks. *Applying unsupervised learning.*

[24] Abdi Hervé, Williams Lynne J. *Principal Component Analysis.* Hoboken, USA: John Wiley & Sons, Inc. Wiley interdisciplinary reviews. Computational statistics, 2010-07, Vol.2 (4), p.433-459