Evaluation of reliability of a technical system

Final project for the course of
Articial Neutral Networks 214350
part of the faculty of
Electrical Engineering and Informatics
from the
Zittau/Görlitz University of Applied Sciences

B. Eng. Automation and Mechatronics

By:

Jesús Jair Reyes Gutiérrez

Project Advisor:

Prof. Dr.-Ing. W. Kästner

Project level:

Bachelor

Date: January 2021

This report represents the work of an undergraduate student for a class part of his DHIK double degree program between the Mexican university "Instituto Tecnológico y de Estudios Superiores de Monterrey" and the German Hochschule "Hochschule Zittau/Görlitz".

Abstract

Throughout the pages of this paper, the process followed to design and develop a Hybrid Markov Model by training a multilayer perceptron with a pre-established data set will be presented, explained and detailed. DataEngine software was used for the realization of the multilayer perceptron, Excel and MatLab were used for the preparation and pre-processing of the data, and Dynstar was used for the implementation of the hybrid Markov model using a Soft Computing model.

Additionally, a brief theoretical framework will be presented with all the necessary concepts to understand each part of the process, as well as the results of each stage, to finally compare the results of the hybrid model with the classical one. At each stage, the justifications for each decision will be included, along with its corresponding implementation.

Acknowledgements

I have to start by thanking my professor of Artificial Neutral Networks, Prof. Dr.-Ing. W. Kästner. For providing me with the necessary knowledge to understand what a Markov model is and thus subsequently to understand each step of the design process. Additionally, I also thank him for his mentoring during the development of the whole project.

Contents

List of Tables

List of Figures

1 Introduction

Over time, the tools, machinery and systems of industry have facilitated mass production, thanks in large part to technological advances that have made them more complex, efficient and durable. However, this progress has also posed a great challenge for engineers and operators, since their complexity prevents the development of models that are 100% adjusted to reality and forces them to be simplified. Therefore, the development of reliable models is one of the greatest challenges of modern industry, since with them it is not only possible to determine the scope of current equipment or processes, but it is also possible to make very accurate forecasts. One of the most relevant forecasts is the one that determines how often it is necessary to perform maintenance on a tool, machinery or system, because maintenance involves not only expenses in itself, but also in production due to downtime.

1.1 Project Aim

For this reason, this paper proposes to evaluate the reliability of a model based on a hybrid Markov model against that of a classical model. For this purpose, a set of data will be presented that will go through different stages to create a multilayer perceptron, which will be subsequently implemented in a computational Markov model, which, based on a stress scenario of temperature and humidity, will provide a forecast of the time required to provide maintenance to the process or system under analysis.

The following is a brief explanation of each of the stages through which the initial data set will pass until the objective is met.

1.1.1 Data pre-processing

Through this first stage, the first filter of the information received will be carried out, because although it is information received directly from the source, the measurements may contain errors that must be eliminated for the subsequent stages

1.1.2 Data preparation

In this second filtering stage, the data must be separated for the training and testing of the multilayer perceptron, but, in order to guarantee efficiency, it is necessary to distribute the data in an intelligent way.

1.1.3 MLP design

This third stage consists directly in designing a multilayer perceptron in the selected software, under certain parameters that guarantee the maximum and minimum error requirements established, in this case, of ±3%.

1.1.4 Mathematical models

This stage presents the design of the Markov model under ideal conditions, which will lead to the calculation of the expected or desired results.

1.1.5 Simulation models

The Markov model will be implemented in a simulation software. For this purpose, the classical and hybrid models will be included in order to generate a comparison.

1.1.6 Simulation results

At this stage, the results obtained in both models, the classical and the hybrid, will be presented,and it is going to be described their empirical properties, without any further or really deep analysis.

1.1.7 Evaluation

Finally, the results obtained will be evaluated, compared, analyzed and interpreted to issue a conclusion about the time required to maintain the analyzed process or equipment. Here the two models, classical, and hybrid will be compared.

1.2 Background

It is worth mentioning that this project is the result of the Articial Neutral Networks 214350 class given by Prof. Dr.-Ing. W. Kästner. Through this class all the necessary knowledge for the realization of this project was provided, as well as all the technological tools for its development. As already mentioned, DataEngine, MatLab, Excel and Dynstar were the software used in the development of the project.

2 Theory and Methodology

First of all, we must begin by defining the importance and necessity of modelling and simulating physical phenomena. Thanks to these tools, it is possible to make prognoses about reality and generate knowledge about the causes that led to the result obtained, as well as what would have happened if something different had been done [1]. Furthermore, modelling and simulating the behavior of a system is not only restricted to a single area, but its application framework ranges from engineering to medicine and social sciences [1].

With this in mind, in order to model any system, it is first necessary to take into account how much is known about it, as well as its characteristics.

2.1 System

A system is a set of components that interact with each other and it is thanks to these interactions that a system is more than just the sum of its components [1]. In addition, a system must have a purpose that, together with the structure of its components, gives it an identity [1]. And finally, some other characteristics or parts of systems are their inputs and outputs, both coming from or going out of the system boundaries [1]. The inputs being totally independent and the outputs depending totally on the system process, which is characterized by its internal or state variables [1].

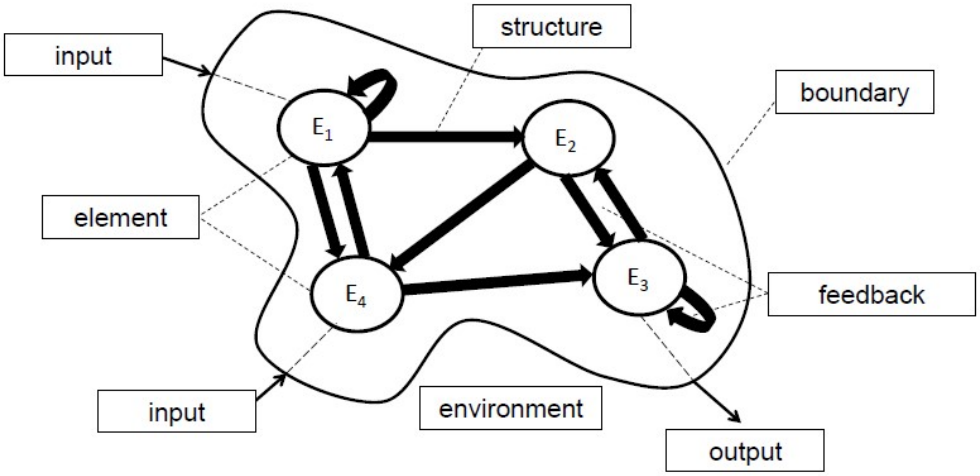


Figure 1: System representation [1]

As can be seen in Figure 1, the system is contained by a boundary, in which the inputs enter, these pass between the elements of the system according to its structure, which can have feedback or changes in the system by the same change of the state variables [1]. In general, a system works by causes and effects, an input arrives and depending on the organization and parameters of the system this is transformed into a desired output [1].

2.2 Model

On the other hand, a model is a simplified scale reproduction of some phenomenon [1]. Models can be mathematical, conceptual or physical, although these will always be limited by the simplification, neglect, abstraction and aggregation they have made of reality [1]. However, some minimum and indispensable requirements they must meet are verifiability and reproducibility in different environments [1]. That is why, in order to favor the verifiability and reproducibility of the models, models can be described or classified in different ways:

2.2.1 Linear or non-linear:

If the input variable undergoes a proportional change with respect to the output, the superposition principle is satisfied and the steady state does not depend on the initial conditions [1].

2.2.2 Static or dynamic:

If the system does not change its behavior during the observation time [1].

2.2.3 Externally influenced or autonomous:

If the system is influenced by external variables, call it inputs [1].

2.2.4 Location dependent or independent:

If the state variables and parameters are interpreted as set, were their position in the system matter[1].

2.2.5 Deterministic or stochastic:

If random external influences on the model are excluded [1].

2.2.6 Time-invariant or time-variant:

If the same behavior can be observed with the same initial conditions [1].

2.2.7 Time continuous or discrete:

If the state of the system is defined at all points in time [1].

Now that one has a better understanding of systems and models, an important step to consider before modeling is to decide whether the model will be a model of behavior or a system model. Depending on the decision, one will face different challenges that will require certain knowledge about the system itself.

2.2.8 Model of behavior

It is a reproduction of the general relationship between input and output without really knowing the internal mechanisms that lead to that change, as in a black box model.

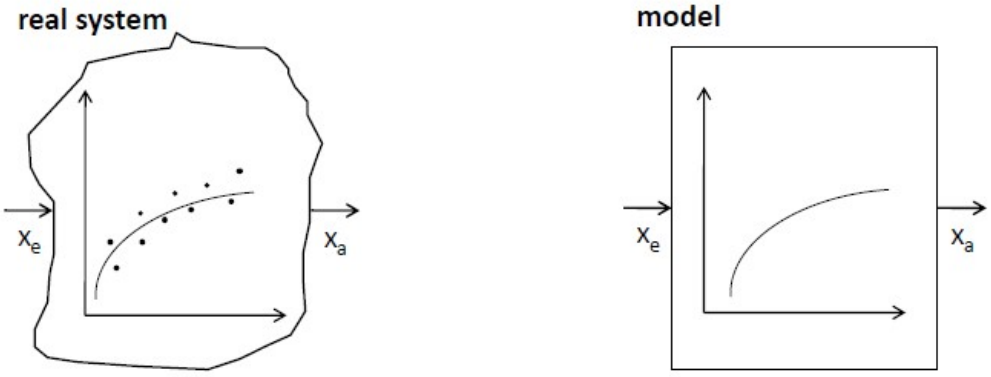


Figure 2: Real system vs. Model of behavior [1]

As can be seen in the Figure 2, the model only tries to emulate the relationship between the input and output, so it is limited to the variations that may occur [1]. To develop the model it is necessary to have a sufficiently large database of the input and output to ensure consistency and reproducibility [1].

2.2.9 System model

It is a reproduction of all the essential internal interactions of the system, so a deep knowledge of the system is required.

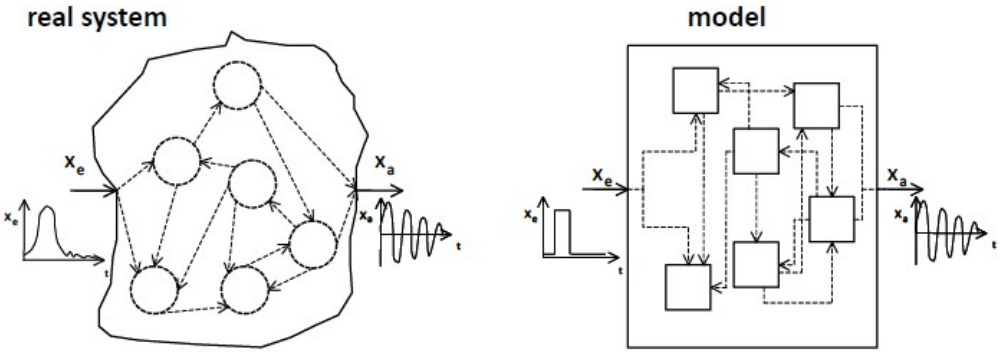


Figure 3: Real system vs. System model [1]

As can be seen in the Figure 3, the most essential internal interactions have to be reproduced in order to approximate the behavior as closely as possible, so the model is limited to the accuracy of the reproduction of the system [1]. The system model is most recommended for systems in which there are many non-linearities, the structure is very complex or unknown conditions are to be analyzed [1].

2.3 Simulation

For the next step, simulation, it is first necessary to design the model, defining the system from its purpose, structure, boundaries and interactions. Then choose the type of model, quantify it, program it and verify it. Then choose the software, implement it, analyze it and validate it. To finally analyze the behavior and look for possible optimizations. It should be clarified that errors can come from different sources, such as the model itself, data linearization, model discretization, iterations, programming or rounding [1].

2.4 Soft Computing

In the case of behavioral modeling, it is necessary to integrate certain data analysis techniques to correctly model the behavior of the system. Some of the most common techniques are artificial neural network and fuzzy set theory, as they allow the modeling of nonlinear multivariable systems [2].

2.4.1 Artificial Neural Network (ANN)

This method attempts to emulate biological neurons. Each neuron collects information which causes an excitatory or inhibitory effect that propagates through the neural network [2]. This method is also characterized by the fixed number of neurons in the network, as well as the knowledge represented by weights [2]. The data is processed by different operations.

2.4.2 Multi-Layer Perceptron (MLP)

A multilayer perceptron is a structuring of artificial neurons in a fixed network, which consists of several layers or stages of neurons that modify the inputs until they resemble the desired output [2]. Generally, multilayer perceptrons consist of a first linear input stage, a second hidden stage that processes the information and a final output stage that delivers the results [2].

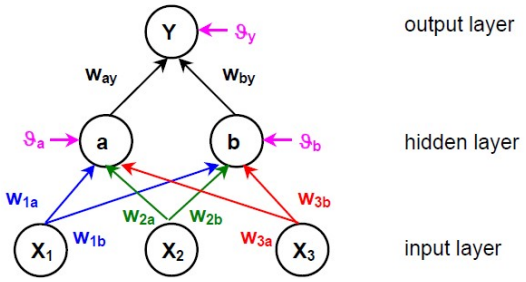


Figure 4: Multilayer perceptron [2]

In the Figure 4 it can be seen how the inputs are passed to the hidden layer, for this, the input is multiplied by a certain coefficient before entering the hidden stage, to later add a weight to it, process the data and go through the same process to pass it to the output layer [2].

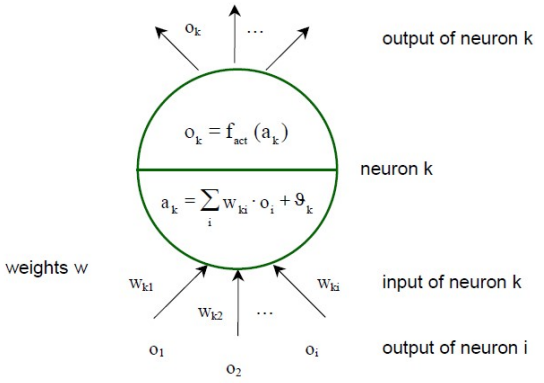


Figure 5: Artificial neuron [2]

Figure 5 shows graphically the process described above.

Multilayer perceptrons, depending on their structure, can also be classified in 3 different ways. Feed forward if neurons only point to the next layer as in Figure 6, lateral if neurons in the same layer can also communicate with each other as in Figure 7, and recurrent if different layers can interact with each other as in Figure 8 [2].

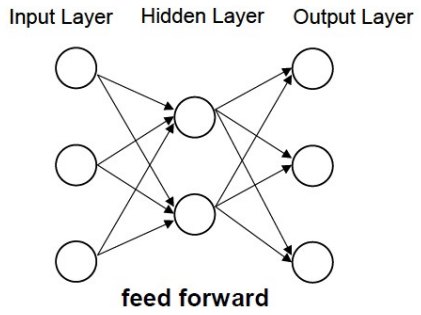


Figure 6: Artificial neural networks in a feed forward configuration [2]

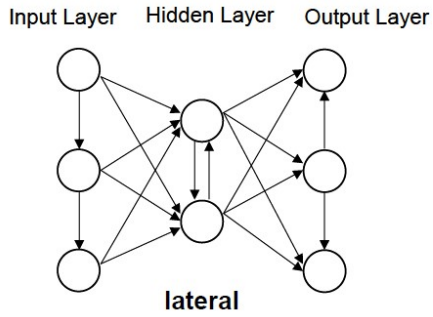


Figure 7: Artificial neural networks in a lateral configuration [2]

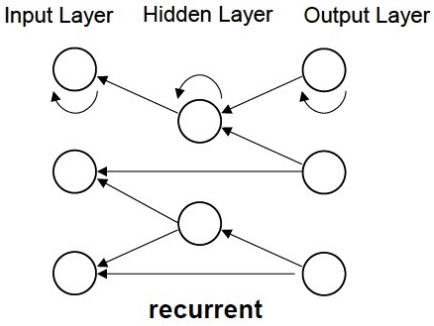


Figure 8: Artificial neural networks in a recurrent configuration [2]

2.4.3 Multi-Layer Perceptron (MLP) Design

For the design of a MLP in the selected software, DataEngine, it is necessary to first define some parameters related to the structure and the way of processing the information. First, it is important to determine the strategy for determining the values of the weights, also known as Backpropagation with Momentum-term [2]. Subsequently, it is necessary to define the Bias neuron, which is the assignment of a threshold value that acts as an additional neuron, providing the interpolation capability [2]. Next, it is necessary to define the Symmetry Breaking, which is the initialization of the variables with random values to avoid errors of identical values [2]. Then, it is also necessary to define Weight Decay, which is the process to homogenize the distribution of weights, in order to avoid critical nodes that have too much weight in the final result [2]. And finally, it is necessary to define Pruning, which is the process of eliminating non-relevant connections and simplifying the model [2].

2.5 Markov Model

However, one of the main components of this project is the Markov model. Which, in a nutshell, is a model that, depending on the stress conditions of a system, is able to estimate the lifetime. To understand the model, it is necessary to first define some concepts, such as reliability assessment, which is the prediction of system failures based on probabilities [3]. It is also important to define reliability, which is the description of system availability and the factors that influence it, such as functional capability, maintainability, etc [3].

Another important concept is availability, which is the probability that the system will be in a functional state at a defined time [3]. Non-availability is the complementary value to this and permanent availability is the parameter for the description of a repairable system [3].

Going into a little more detail, the Markov model starts from the idea that any system as in Figure 9 can be found in one of two states, either intact or defect, where to pass from one to the other there is a probability of failure or repair [3].

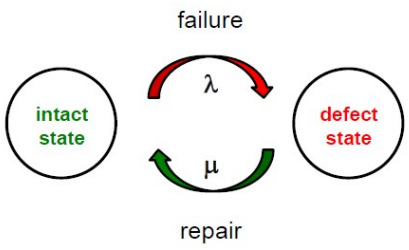


Figure 9: Binary model system state [3]

2.5.1 Failure rate

The probability of failure describes the time that a non-faulty system, after a time t, will fail [3]. This can be calculated by dividing the density of the probability of failure by the probability of survival or reliability [3].

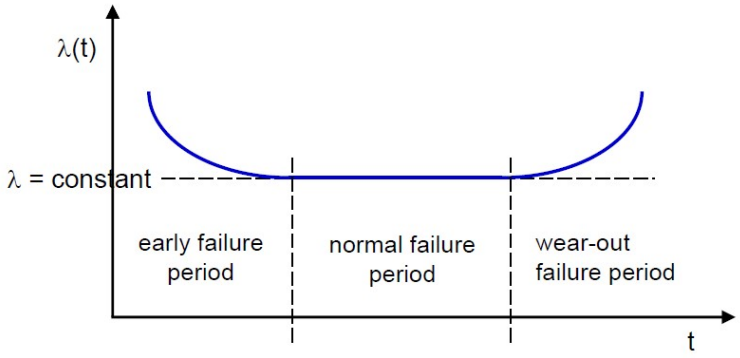


Figure 10: Failure rate curve [3]

As can be seen in Figure 10 the failure probability is a function of time describing a bath-tub curve [3] This curve can be divided into three main phases, the first one of early failure where, due to the early use of elements, they tend to fail, but then show a decline in this failure [3]. A second stage of normal failure where the highest reliability of the system is reached and this is the desired period of operation [3]. Finally, there is the wear-out phase where there is a systematic failure due to aging, fatigue, wear-out of the components and it is the phase where maintenance or corrective measures should be provided [3]. The life time of a component or system can also be obtained or visualized from the graph . Some of the most important characteristics of the probability of failure are that it has a continuous distribution over the lifetime, this being, mathematically, an exponential distribution [3].

$$\text{Failure rate} = \lambda$$

$$\text{Repair rate} = \mu$$

- Reliability or probability of survival $R(t)$

$$R(t) = e^{-\lambda t} \tag{1}$$

- Probability of failure $F(t)$

$$F(t) = 1 - e^{-\lambda t} \tag{2}$$

- Life time $\overline{T}$

$$\overline{T} = \int_0^\infty R(t)\, dt = \frac{1}{\lambda} \tag{3}$$

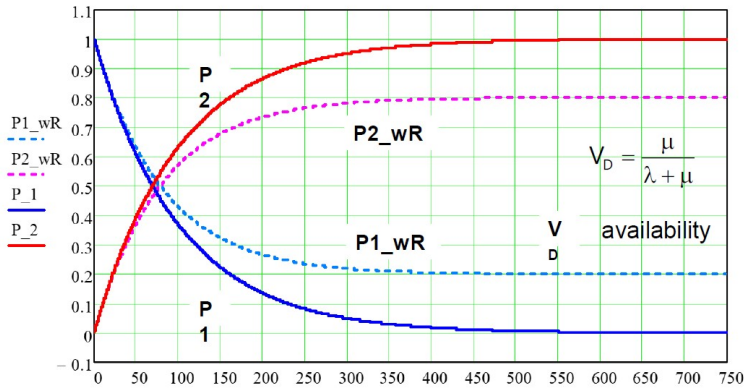


Figure 11: Failure vs survival rate on a single component w/without repair [3]

2.5.2 Redundancy

Another important concept to take into account is redundancy, which is, in general terms, the arrangement of multiple elements in the same form [3]. This provides a system with protection against random independent failures [3].

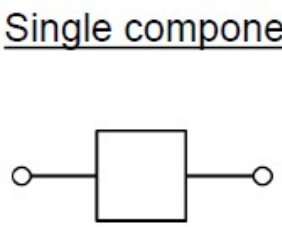


Figure 12: Single component [3]

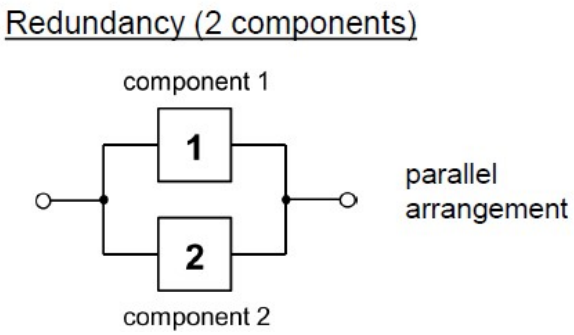


Figure 13: Two components in parallel [3]

Single component

$$W(a) = W(e) = p = 0.9$$

Two components

$$W(a) = 1 - \prod_{i=1}^{2}(1 - p_i)$$

$$W(A) = 1 - (1 - 2p + p^2)$$

$$W(a) = 2p - p^2 = 0.99 \tag{4}$$

As can be seen in Figures 12 and 13, thanks to redundancy it is possible to increase the reliability probability of a component with a reliability of 90% by an additional 9% [3].

This translated into the equations of 1, 2 and 3 gives that the lifetime of a system can be increased by up to 50% with respect to the original model [3].

- Reliability or probability of survival $R(t)$

$$R(t) = 1 - F(t) = 2e^{-\lambda t} - e^{-2\lambda t} \tag{5}$$

- Probability of failure $F(t)$

$$F(t) = (1 - e^{-\lambda t})^2 = 1 - 2e^{-\lambda t} + e^{-2\lambda t} \tag{6}$$

- Life time $\overline{T}$

$$\overline{T} = \int_0^\infty R(t)\, dt = \frac{2}{\lambda} - \frac{1}{2\lambda} = 1.5\frac{1}{\lambda} \tag{7}$$

2.5.3 Markov model: Evaluation of reliability

Within the Markov model, the probabilities of the states of the binary model are used to evaluate the probability of the state in which the system is found [3]. The Markov model is governed by the probability of failure and repair, since with these a vector of differential equations of order k is constructed, where k is the number of states of the system [3].

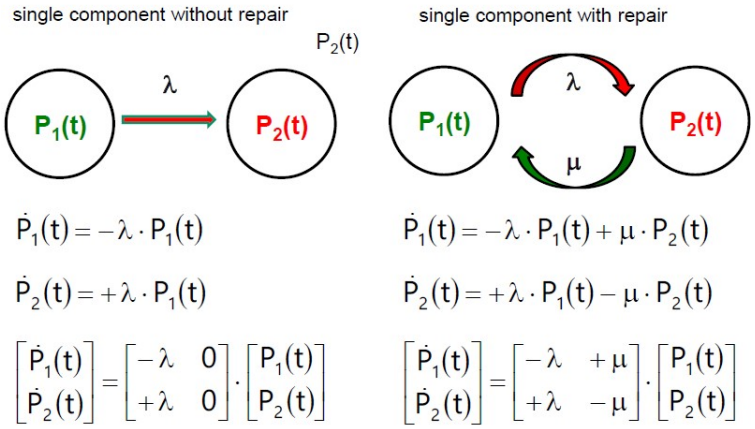


Figure 14: Markov model [3]

2.6 Methodology

Regarding the process to be followed for the construction of the multilayer perceptron and its use in the Markov model. For the construction of the MLP, it is necessary to first pre-process the data or filter it from any erroneous data. Then a second filtering stage is required to separate the data for testing and training. Then it is required to first train the MLP with pruning in order to determine a more optimized model, apply the optimizations, remove pruning and train it, then test the test data and evaluate if the results are satisfactory. Finally, it is necessary to implement the MLP in a simulation software with the connections configured according to the Markov model, simulate, obtain the results, analyze them and issue a conclusion.

3 Data pre-processing

As already mentioned, we are going to start with an initial data set, which, being directly collected from the measurement system, may or may not come with errors. The origin of these errors may well be directly the sensor, which made some erroneous detection by some internal error, these errors will be called measuring errors. Another possible origin is the processing of the sensor signal and they manifest themselves in the form of out-of-range values or sensor defect. And the last possible origin are duplicate or redundant values that only hinder the processing

3.1 Atypical values

As already mentioned, we are going to start with a data set, see table 10 in appendix C. These data were taken directly from the measurement system, so a first filtering is required in order to deliver only valid data to the next stage. To better visualize what we are talking about, in the Figure 15 and 16 it can be seen how there are certain values that stand out for having a completely different behavior from the rest and that if processed would result in an inconsistent model.

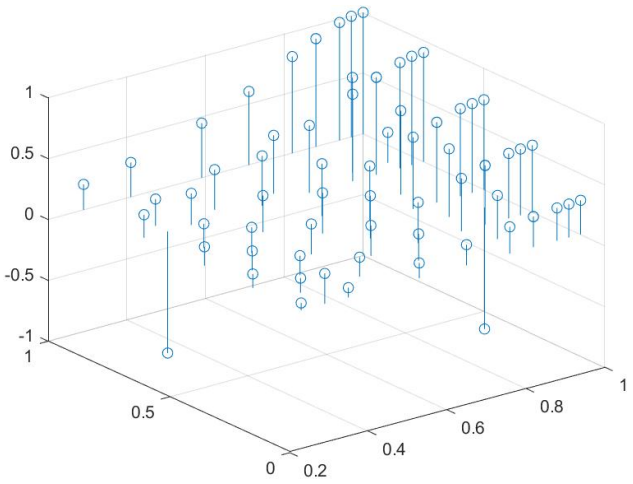


Figure 15: Raw data visualization

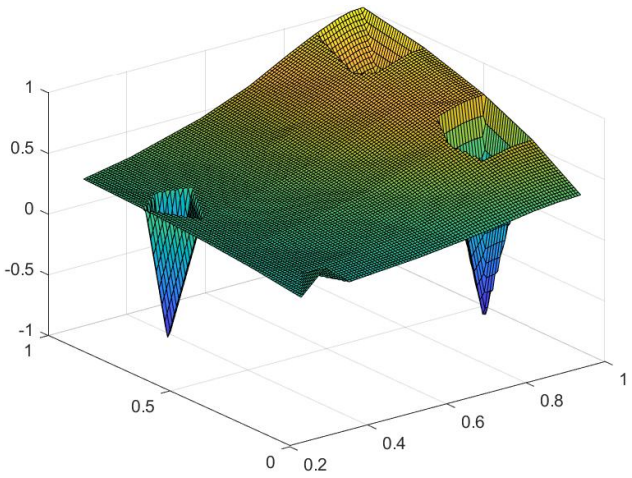


Figure 16: Raw data on a mesh-grid

3.1.1 Sensor defect

The purpose of this first filter is to eliminate all those values that are physically impossible to occur in real life. These can be easily detected by the out-of-range values, in this case, the measurements being between 0 and 1, 2 values with -1 will be eliminated. The remaining values give the behavior of Figure 17.

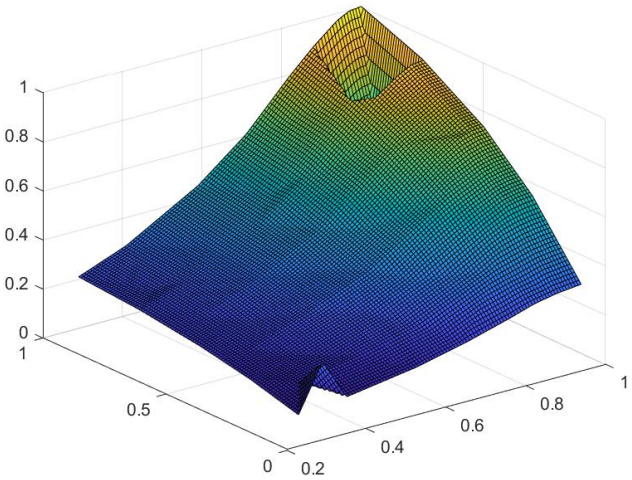


Figure 17: Data after first filter

As can be seen, in spite of having eliminated the most important outliers, the graph still has points where the general trend has a sudden change, but this will be analyzed later.

3.1.2 Redundant data

This next stage of filtering is a little more complicated to visualize graphically, as most of the graphing software eliminates these values by itself and it is only possible to detect them manually. For the given data set, there are 2 measurements for the same value of X1 = 0.88. Therefore, it will be decided to eliminate 5 repeated values, which despite having more significant figures, do not bring more precision.

3.1.3 Measuring errors

Finally, the Figure 17 shows how there are two values that bring an abrupt change to the general behavior of the data. To eliminate them, the behavior of the data is simply analyzed and if a value that changes the gradient is found, it will be eliminated. After running the code, 2 data were removed and the remaining data are shown in Figures 18 and 19 .

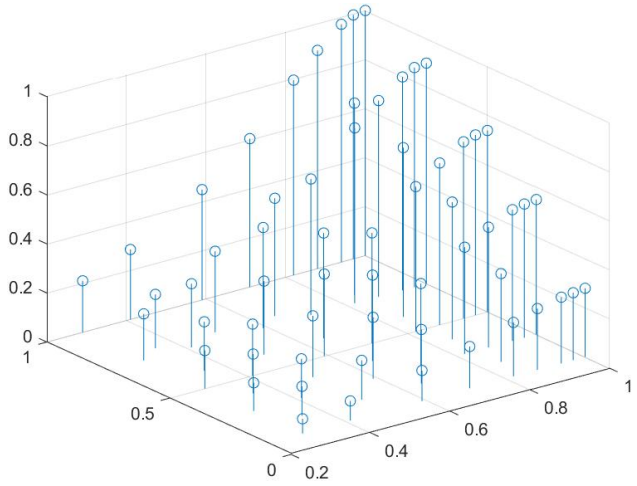


Figure 18: Final processed data graph

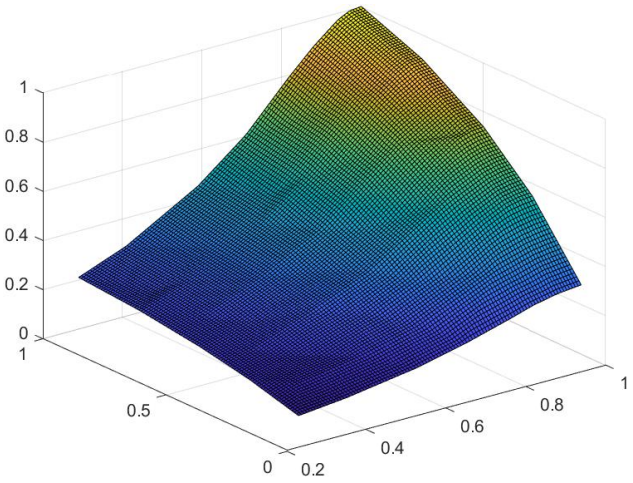


Figure 19: Final processed data mesh-grid

4 Data preparation

For both the pre-processing and data preparation sections, the Code A.1 in Appendix A was designed, programmed and tested. This code performed the filtering described in the previous section, as well as the separation of the data into two sets, one for training and one for testing.

4.1 Considerations

For this stage we sought to implement a process that would guarantee not only a maximum error within the limits of 3% for MLP training and testing, but that would also be flexible and allow several possible training and testing sets. For this purpose, a variable called tolerance was created, which, in short, sets aside the maximum and minimum values minus or plus the tolerance and leaves the remaining data available for random selection, where the user determines the amount of data for testing.

4.2 Data

In the case of this project, a tolerance of 20% was selected with a test data set of maximum 20 data. The program ran, performed the filtering and separation of the data, to finally deliver the two data sets.

4.2.1 Training data

As can be seen, the data set obtained contains the maximum and minimum data minus or plus the tolerance of the three variables, X1, X2 and Y. This decision will ensure that the largest gradients or groups of data that would provide a greater error are already considered and therefore the resulting error is minimal. There is a total of 41 data for training.

X1	X2	Y
0.29	0.1	0.06
0.41	0.1	0.08
0.29	0.3	0.115
0.59	0.1	0.125
0.29	0.5	0.155

Table 1 continued from previous page

0.71	0.1	0.17
0.29	0.75	0.19
0.29	1	0.21
0.82	0.1	0.22
0.88	0.1	0.25
0.94	0.1	0.27
0.97	0.1	0.275
1	0.1	0.28
0.41	1	0.285
0.59	1	0.45
0.88	0.3	0.49
0.94	0.3	0.535
0.97	0.3	0.545
1	0.3	0.55
0.71	1	0.605
0.88	0.5	0.66
0.94	0.5	0.72
0.97	0.5	0.735
1	0.5	0.74
0.82	1	0.795
0.88	0.75	0.8
0.94	0.75	0.87
0.88	1	0.89
0.97	0.75	0.895
1	0.75	0.9
0.94	1	0.97
0.97	1	0.995
1	1	1
0.35	0.2	0.12
0.35	0.4	0.16

Table 1 continued from previous page

0.41	0.3	0.16
0.5	0.2	0.16
0.35	0.6	0.2
0.41	0.5	0.21
0.35	0.8	0.22
0.65	0.2	0.22

Table 1: Training data set

4.2.2 Test data

As can be seen, the data obtained for the testing are average and very standard values for the three parameters X1, X2 and Y. All the maximum and minimum values together with the tolerance were separated, so that the separation of these data to form part of the test should not alter the general behavior of the system. There is a total of 20 data for testing.

X1	**X2**	**Y**
0.5	0.4	0.25
0.59	0.3	0.25
0.41	0.75	0.26
0.5	0.6	0.3
0.5	0.8	0.33
0.59	0.5	0.335
0.71	0.3	0.335
0.65	0.4	0.35
0.85	0.2	0.36
0.59	0.75	0.41
0.65	0.6	0.43
0.82	0.3	0.435
0.71	0.5	0.45
0.65	0.8	0.48
0.71	0.75	0.555

Table 2 continued from previous page

0.85	0.4	0.56
0.82	0.5	0.59
0.85	0.6	0.69
0.82	0.75	0.715
0.85	0.8	0.78

Table 2: Test data set

5 MLP design

As already mentioned, the design of an MLP requires the definition of several variables. These variables range from the number of inputs and outputs, the number of hidden layers, back-propagation, momentum, etc.... The definition of these variables is of vital importance, because as mentioned in the methodology, they will allow us to calculate a first model with pruning, which will give us the optimal number of connections, to later make a second model without pruning that really optimally models the data set without losing interpolation capacity.The overall goal that must be achieved is to create a MLP with a maximum error lower than ±3%. In Appendix B it can be seen the whole design process within the software's windows.

5.1 Parameters

There are some parameters that will not change between the models with and without pruning, these being the learning parameters:

- Learning parameters

	First hidden layer	**Second hidden layer**	**Output layer**
Learn ratio	0,1	0,1	0,9999999
Momentum	0,1	0,1	0,9999999
Weight-decay	0,1	0,1	0,9999999

Table 3: Learning parameters

- Weights initialization

Lower limit:	-0,1
Upper limit:	0,1
Initial value:	1234567890

Table 4: Weights initialization

- Stop conditions

<max. training error	0,0001
<RMS training error	0,0001
<max. training error	0,0001
<RMS training error	0,0001
number of iterations is divisible by	100.000
iterations	100.000
error measurement	0,9 RMS error + 0,1 max. error
Based on	test error

Table 5: Stop conditions

As mentioned in the methodology, the MLP development process requires a first design phase with pruning and a second phase without pruning. However, both share the same data and the parameters described above. The main differences come from the network structure itself and the learning process used.

5.1.1 Before pruning

Before pruning, the network structure must have more connections than the data used. So if by default you have two inputs and one output, that means that the other connections must come from the hidden layers, for this reason, it was also decided to use two hidden layers instead of one, because if that had been done, the hidden layer would have been very large compared to its neighbors.

	Number of neurons	**Transfer function**
Input:	2	Linear
First hidden layer:	6	Tanh
Second hidden layer:	5	Tanh
Output:	1	Linear

Table 6: Architecture with pruning

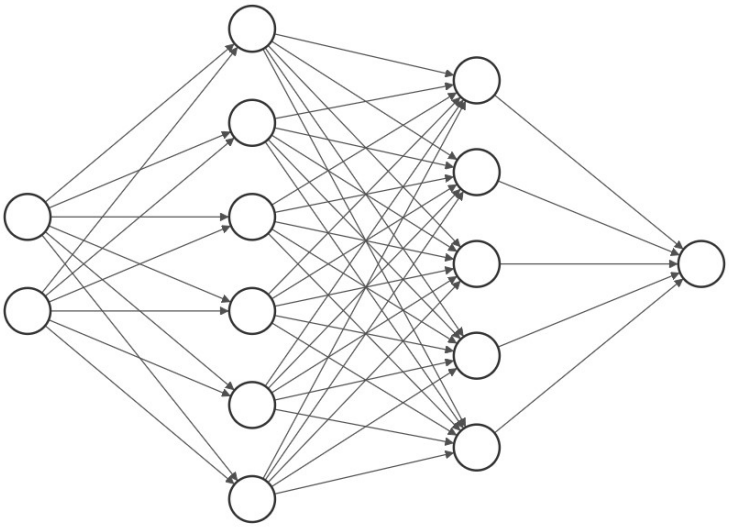


Figure 20: Graph of artificial neural network

Finally, with respect to the main topic of the section, in the learning process pruning had to be activated along with certain parameters for it, like threshold, or time constant, so the algorithm can determine which connections to eliminate. Once run, the optimal number of connections was obtained, in this case only 9 connections are really necessary. This structure was then applied to the next model and optimized.

Process	Backpropagation
Momentum	ON
Weight-decay	ON
Learning strategy	Single step (delta)
Presentation sequence	Sequential
Pruning	ON
Relevance threshold	0,02
Time constant	10

Table 7: Learning process with pruning

Once all the variables were set in the DataEngine software, the results shown in Figure 22 and 23 were obtained. As can be seen in them, the maximum and minimum errors, although they come from a single

data, are greater than expected and therefore exceed the imposed limit of 3% as maximum. Nevertheless, since the most optimal model should have 9 connections, this adjustment will be made to check if the model has improved. Also the error graph shows how this varied.

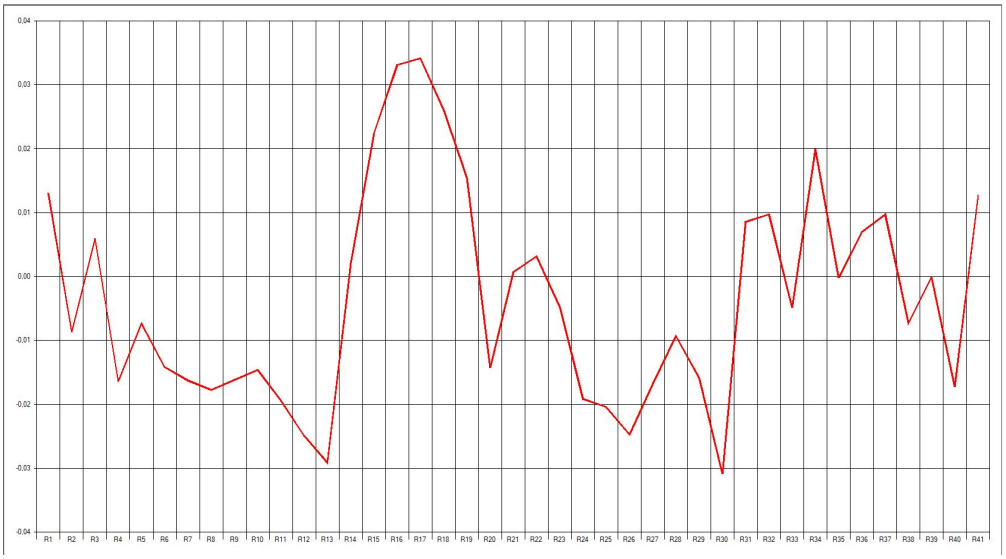


Figure 21: Error development for network with pruning

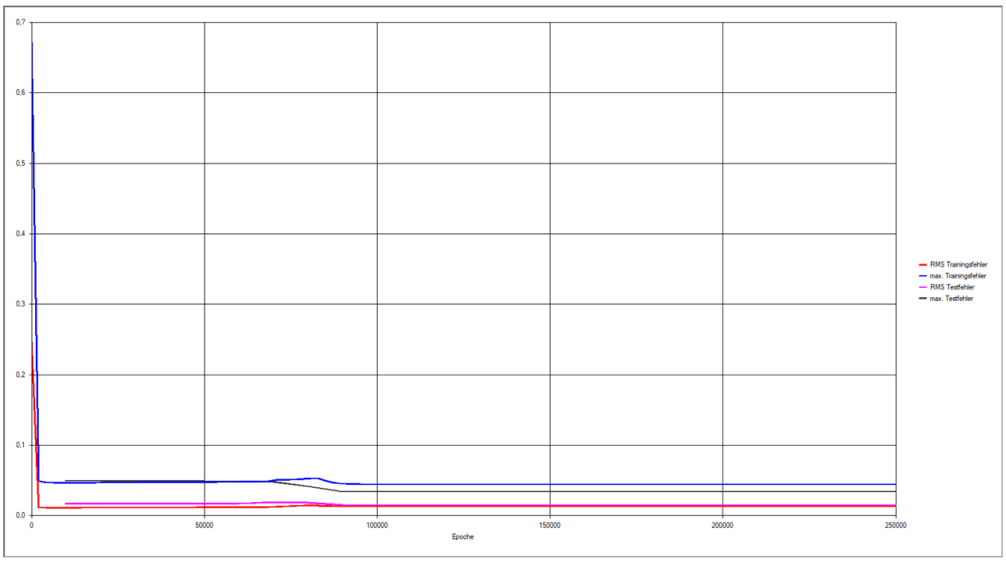


Figure 22: Results before optimization

	X1	X2	Y	MLP_Y	Fehler_MLP_Y	RMS-Fehler	*
1	0,290	0,100	0,060	0,047	0,013	0,013	
2	0,410	0,100	0,080	0,089	-0,009	0,009	
3	0,290	0,300	0,115	0,109	0,006	0,006	
4	0,590	0,100	0,125	0,141	-0,016	0,016	
5	0,290	0,500	0,155	0,162	-0,007	0,007	
6	0,710	0,100	0,170	0,184	-0,014	0,014	
7	0,290	0,750	0,190	0,206	-0,016	0,016	
8	0,290	1,000	0,210	0,228	-0,018	0,018	
9	0,820	0,100	0,220	0,236	-0,016	0,016	
10	0,880	0,100	0,250	0,265	-0,015	0,015	
11	0,940	0,100	0,270	0,289	-0,019	0,019	
12	0,970	0,100	0,275	0,300	-0,025	0,025	
13	1,000	0,100	0,280	0,309	-0,029	0,029	
14	0,410	1,000	0,285	0,283	0,002	0,002	
15	0,590	1,000	0,450	0,428	0,022	0,022	
16	0,880	0,300	0,490	0,457	0,033	0,033	
17	0,940	0,300	0,535	0,501	0,034	0,034	
18	0,970	0,300	0,545	0,519	0,026	0,026	
19	1,000	0,300	0,550	0,535	0,015	0,015	
20	0,710	1,000	0,605	0,619	-0,014	0,014	
21	0,880	0,500	0,660	0,659	0,001	0,001	
22	0,940	0,500	0,720	0,717	0,003	0,003	
23	0,970	0,500	0,735	0,740	-0,005	0,005	
24	1,000	0,500	0,740	0,759	-0,019	0,019	
25	0,820	1,000	0,795	0,815	-0,020	0,020	
26	0,880	0,750	0,800	0,825	-0,025	0,025	
27	0,940	0,750	0,870	0,887	-0,017	0,017	
28	0,880	1,000	0,890	0,899	-0,009	0,009	
29	0,970	0,750	0,895	0,911	-0,016	0,016	
30	1,000	0,750	0,900	0,931	-0,031	0,031	
31	0,940	1,000	0,970	0,961	0,009	0,009	
32	0,970	1,000	0,995	0,985	0,010	0,010	
33	1,000	1,000	1,000	1,005	-0,005	0,005	
34	0,350	0,200	0,120	0,100	0,020	0,020	
35	0,350	0,400	0,160	0,160	-0,000	0,000	
36	0,410	0,300	0,160	0,153	0,007	0,007	
37	0,500	0,200	0,160	0,150	0,010	0,010	
38	0,350	0,600	0,200	0,207	-0,007	0,007	
39	0,410	0,500	0,210	0,210	-0,000	0,000	
40	0,350	0,800	0,220	0,237	-0,017	0,017	
41	0,650	0,200	0,220	0,207	0,013	0,013	
*							

		Fehler_MLP_Y
1	Minimum	-0,031
2	Maximum	0,034
3	Mittelwert	-0,004
4	Varianz	0,000
5	Standardabweichung	0,017
6	Spannweite	0,065
7	Schiefe	0,532
8	Exzeß	-0,479
9	Summe	-0,148
10	Quadratische Summe	0,012
11	Anzahl Werte	41,000
12	Anzahl fehlender Werte	0,000

Figure 23: Error scope before optimization

For the optimization, it is also obtained, that the most relevant connections are the following:

	Von Neuron	Zu Neuron	Gewichte	Delta-Gewicht	Relevanz
1	Bias	1te Verdeckte 6	-1,287571	0,0007687997	0,780966
2	Bias	2te Verdeckte 2	1,774579	-0,0007315367	1,165866
3	Bias	Y	0,361359	0,0030341833	0,216155
4	X1	1te Verdeckte 3	2,478293	0,0000862157	0,958389
5	X1	1te Verdeckte 6	2,583372	0,0004055649	1,009902
6	X2	1te Verdeckte 2	1,783418	0,0001641247	0,492936
7	1te Verdeckte 2	2te Verdeckte 1	-0,378202	-0,0002369399	0,119224
8	1te Verdeckte 2	2te Verdeckte 2	-1,161388	-0,0000731145	0,385908
9	1te Verdeckte 3	2te Verdeckte 1	-0,268151	-0,0020386579	0,123274
10	1te Verdeckte 6	2te Verdeckte 2	-0,967444	-0,0000983847	0,404745
11	2te Verdeckte 1	Y	-0,782714	-0,0008512228	0,175015
12	2te Verdeckte 2	Y	-1,404101	0,0027620963	0,544113

Figure 24: Weights values with pruning

5.1.2 After pruning

Once the value of connections is optimized, it is implemented in the model already created, changing the values of the hidden layers, because as mentioned, the inputs and outputs are predefined.

	Number of neurons	Transfer function
Input:	2	Linear
First hidden layer:	3	Tanh
Second hidden layer:	2	Tanh
Output:	1	Linear

Table 8: Architecture after pruning

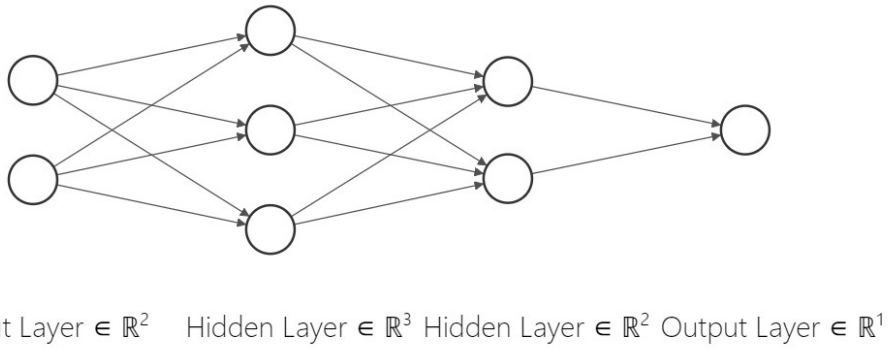


Figure 25: Graph of artificial neural network without pruning

Finally, only the pruning option is deactivated to calculate the final model.

Process	Backpropagation
Momentum	ON
Weight-decay	ON
Learning strategy	Single step (delta)
Presentation sequence	Sequential
Pruning	OFF
Relevance threshold	-
Time constant	-

Table 9: Learning process without pruning

As can be seen in Figures 27 and 28, the results of the training data are much more satisfactory and are within the established limit of ±3%, as they are of -1.0% and 1.2%. Also the error graph showed an improvement.

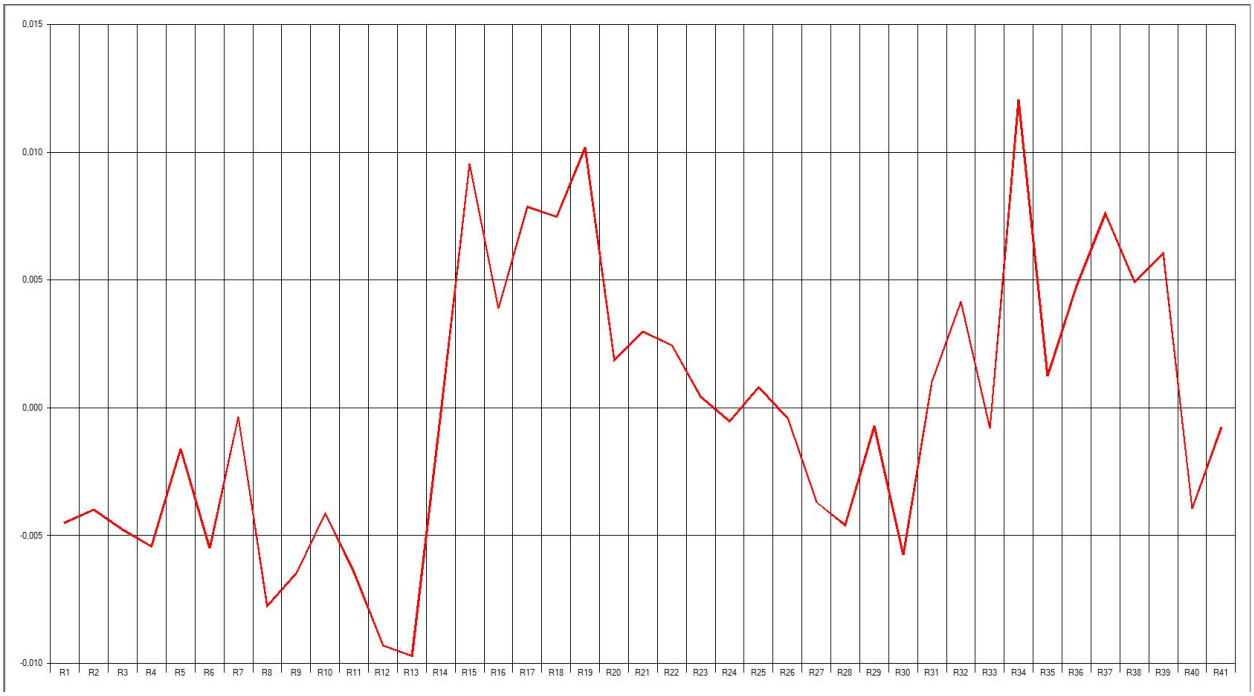


Figure 26: Error progression without pruning

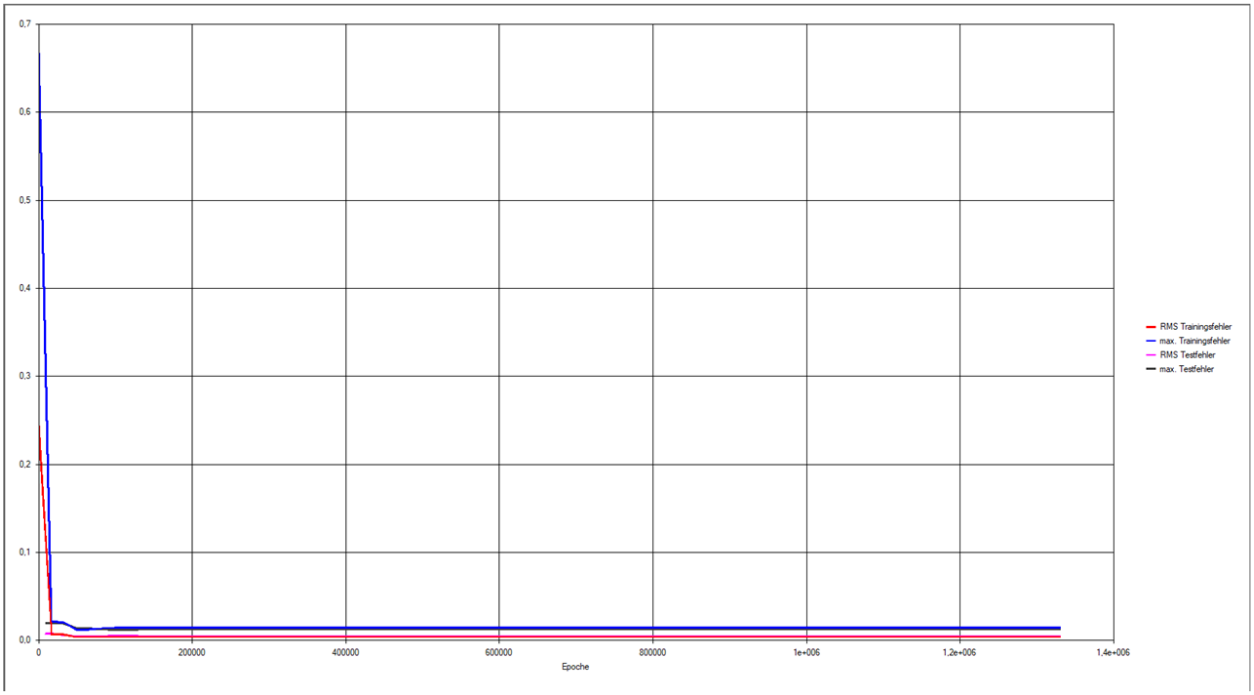


Figure 27: Results after optimization

	X1	X2	Y	MLP_Y	Fehler_MLP_Y	RMS-Fehler	*
1	0,290	0,100	0,060	0,064	-0,004	0,004	
2	0,410	0,100	0,080	0,084	-0,004	0,004	
3	0,290	0,300	0,115	0,120	-0,005	0,005	
4	0,590	0,100	0,125	0,130	-0,005	0,005	
5	0,290	0,500	0,155	0,157	-0,002	0,002	
6	0,710	0,100	0,170	0,176	-0,006	0,006	
7	0,290	0,750	0,190	0,190	-0,000	0,000	
8	0,290	1,000	0,210	0,218	-0,008	0,008	
9	0,820	0,100	0,220	0,226	-0,006	0,006	
10	0,880	0,100	0,250	0,254	-0,004	0,004	
11	0,940	0,100	0,270	0,276	-0,006	0,006	
12	0,970	0,100	0,275	0,284	-0,009	0,009	
13	1,000	0,100	0,280	0,290	-0,010	0,010	
14	0,410	1,000	0,285	0,285	-0,000	0,000	
15	0,590	1,000	0,450	0,440	0,010	0,010	
16	0,880	0,300	0,490	0,486	0,004	0,004	
17	0,940	0,300	0,535	0,527	0,008	0,008	
18	0,970	0,300	0,545	0,538	0,007	0,007	
19	1,000	0,300	0,550	0,540	0,010	0,010	
20	0,710	1,000	0,605	0,603	0,002	0,002	
21	0,880	0,500	0,660	0,657	0,003	0,003	
22	0,940	0,500	0,720	0,718	0,002	0,002	
23	0,970	0,500	0,735	0,735	0,000	0,000	
24	1,000	0,500	0,740	0,741	-0,001	0,001	
25	0,820	1,000	0,795	0,794	0,001	0,001	
26	0,880	0,750	0,800	0,800	-0,000	0,000	
27	0,940	0,750	0,870	0,874	-0,004	0,004	
28	0,880	1,000	0,890	0,895	-0,005	0,005	
29	0,970	0,750	0,895	0,896	-0,001	0,001	
30	1,000	0,750	0,900	0,906	-0,006	0,006	
31	0,940	1,000	0,970	0,969	0,001	0,001	
32	0,970	1,000	0,995	0,991	0,004	0,004	
33	1,000	1,000	1,000	1,001	-0,001	0,001	
34	0,350	0,200	0,120	0,108	0,012	0,012	
35	0,350	0,400	0,160	0,159	0,001	0,001	
36	0,410	0,300	0,160	0,155	0,005	0,005	
37	0,500	0,200	0,160	0,152	0,008	0,008	
38	0,350	0,600	0,200	0,195	0,005	0,005	
39	0,410	0,500	0,210	0,204	0,006	0,006	
40	0,350	0,800	0,220	0,224	-0,004	0,004	
41	0,650	0,200	0,220	0,221	-0,001	0,001	
*							

		Fehler_MLP_Y
1	Minimum	-0,010
2	Maximum	0,012
3	Mittelwert	-0,000
4	Varianz	0,000
5	Standardabweichung	0,005
6	Spannweite	0,022
7	Schiefe	0,326
8	Exzeß	-0,579
9	Summe	-0,002
10	Quadratische Summe	0,001
11	Anzahl Werte	41,000
12	Anzahl fehlender Werte	0,000

Figure 28: Error scope after optimization

Finally, with the new model, it is possible to run the training data and verify if the model is really valid or if it is necessary to make changes in the parameters. As can be seen in Figure 30, the errors and minimums are 0.3% and 2.6%, so they are effectively within the limit and the model is valid for the next stage.

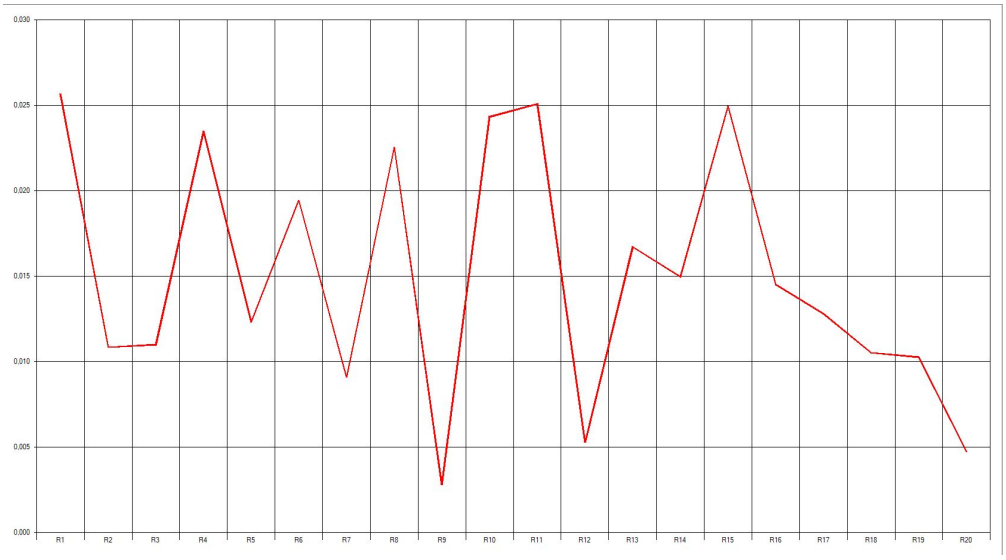


Figure 29: Error progression without pruning for test

	X1 [-]	X2 [-]	Y [-]	MLP_Y []	Fehler_MLP_Y []	RMS-Fehler []	*
1	0,500	0,400	0,250	0,224	0,026	0,026	
2	0,590	0,300	0,250	0,239	0,011	0,011	
3	0,410	0,750	0,260	0,249	0,011	0,011	
4	0,500	0,600	0,300	0,277	0,023	0,023	
5	0,500	0,800	0,330	0,318	0,012	0,012	
6	0,590	0,500	0,335	0,316	0,019	0,019	
7	0,710	0,300	0,335	0,326	0,009	0,009	
8	0,650	0,400	0,350	0,327	0,023	0,023	
9	0,850	0,200	0,360	0,357	0,003	0,003	
10	0,590	0,750	0,410	0,386	0,024	0,024	
11	0,650	0,600	0,430	0,405	0,025	0,025	
12	0,820	0,300	0,435	0,430	0,005	0,005	
13	0,710	0,500	0,450	0,433	0,017	0,017	
14	0,650	0,800	0,480	0,465	0,015	0,015	
15	0,710	0,750	0,555	0,530	0,025	0,025	
16	0,850	0,400	0,560	0,545	0,015	0,015	
17	0,820	0,500	0,590	0,577	0,013	0,013	
18	0,850	0,600	0,690	0,679	0,011	0,011	
19	0,820	0,750	0,715	0,705	0,010	0,010	
20	0,850	0,800	0,780	0,775	0,005	0,005	
*							

		Fehler_MLP_Y []
1	Minimum	0,003
2	Maximum	0,026
3	Mittelwert	0,015
4	Varianz	0,000
5	Standardabweichung	0,007
6	Spannweite	0,023
7	Schiefe	0,083
8	Exzeß	-1,198
9	Summe	0,301
10	Quadratische Summe	0,006
11	Anzahl Werte	20,000
12	Anzahl fehlender Werte	0,000

Figure 30: Results test

6 Mathematical models

For the calculations, we will first define our Markov model, obtain the equations that will model the process and determine the connections between the elements that need to be made. To begin with, we will start from the idea that the model will have parallel active redundancy, which means that two components will be responsible for delivering the final result. As a consequence, instead of there being only two states, one of failure and the other of operation, now there are 4 for the combinations of the two components, because in case one fails, the other is still in operation. It is also going to be assumed that both components cannot fail at the same time. Having said that, it is necessary to define some variables, as well as to clarify what is being assumed from the model.

6.1 Calculations

First, we begin by defining our inputs, outputs and ranges:

$$x_1 = \beta_1; \textit{ first stress parameter, which will be the enviromental temperature} \tag{8}$$

$$x_2 = \beta_2; \textit{ second stress parameter, which will be the moisture or humidity in the enviroment} \tag{9}$$

$$y = f; \textit{ Output of the system} \tag{10}$$

Also it is important to mention that according to the Markov model, it is assumed that $\tilde{\lambda}$ can be calculated through the following equation.

$$\tilde{\lambda}(t) = \lambda_0 + \lambda f(t) \tag{11}$$

$$\lambda_0 = 0.04\frac{1}{Tu}$$

$$\lambda = 0.05\frac{1}{Tu}$$

$$f(t) = Y = \textit{output of the model}$$

Which states that if the value of λ is above 0.05 then the system will be under a bigger stress than usual. Therefore, its life time could be seen shorten

Once established the variables above, now it is time to develop the Markov model under the desired

specifications of an active redundant model.

The number of states is going to be 4, since there are two components A and B, which could be on either one of two states, failure or operation.

$$State1 = P_1(t) = (A, B)$$

$$State2 = P_2(t) = (\overline{A}, B)$$

$$State3 = P_3(t) = (A, \overline{B})$$

$$State4 = P_4(t) = (\overline{A}, \overline{B})$$

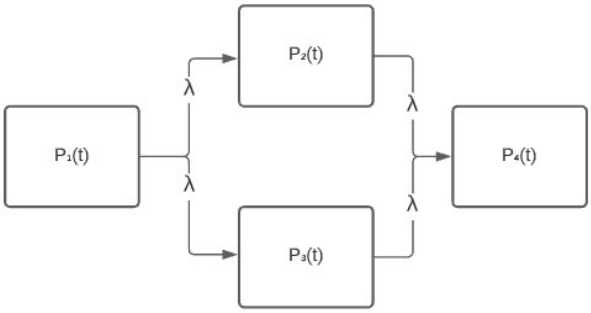


Figure 31: Model diagram

From the information above we can obtain the matrix for the model:

$$\begin{bmatrix} \dot{P_1}(t) \\ \dot{P_2}(t) \\ \dot{P_3}(t) \\ \dot{P_4}(t) \end{bmatrix} = \begin{bmatrix} -2\lambda & 0 & 0 & 0 \\ \lambda & -\lambda & 0 & 0 \\ \lambda & 0 & -\lambda & 0 \\ 0 & \lambda & \lambda & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} \tag{12}$$

If we calculate the eigenvalues of the matrix, then we will also obtain that the system is stable,since there is just one value equal to zero, and the rest are lower than zero. And, as there are no conjugated roots, it will have an aperiodic behavior.

$$\alpha = \begin{bmatrix} 0 \\ -\lambda \\ -\lambda \\ -2\lambda \end{bmatrix} \tag{13}$$

One last data that can be obtained from the matrix is the recommended simulation step. Since it is a matrix $4x4$, then the value of $n = 4$, while the maximum value of the matrix being $|a_{max}| = 2\lambda$. This will give us that the step value must be of:

$$\Delta t < \frac{1}{4(2\lambda)} = \frac{1}{8\lambda} \tag{14}$$

Then, by analyzing each variable individually, we can get a set of differential equations:

$$\frac{1}{2\lambda}\dot{P_1}(t) + P_1(t) = 0$$

$$\frac{1}{\lambda}\dot{P_2}(t) + P_2(t) = P_1(t)$$

$$\frac{1}{\lambda}\dot{P_3}(t) + P_3(t) = P_1(t)$$

$$\dot{P_4}(t) = \lambda[P_2(t) + P_3(t)]$$

The next step involves transforming into the Laplace domain the set of obtained differential equation, so they can be represented as transfer functions in the simulator.

$$\frac{1}{2\lambda}sP_1(s) + P_1(s) = \emptyset$$

$$\frac{1}{\lambda}sP_2(s) + P_2(s) = P_1(s)$$

$$\frac{1}{\lambda}sP_3(s) + P_3(s) = P_1(s)$$

$$sP_4(s) = \lambda[P_2(s) + P_3(s)]$$

$$\frac{P_1(s)}{\emptyset} = \frac{1}{\frac{1}{2\lambda}s + 1} \tag{15}$$

$$\frac{P_2(s)}{P_1(s)} = \frac{1}{\frac{1}{\lambda}s + 1}$$

$$\frac{P_3(s)}{P_1(s)} = \frac{1}{\frac{1}{\lambda}s + 1}$$

$$\frac{P_4(s)}{P_2(s) + P_3(s)} = \frac{\lambda}{s}$$

Finally, with this information we are now able to represent the model in any simulator. In our case, DynStar is going to be used.

One last point to remark is that, for the hybrid Markov model, instead of calculating λ as a constant, it is going to be directly used the output of 11, meaning that this value is going to be always dynamic.

6.2 Expected results

With all this information, parameters, variables and conditions, it is now possible to represent the calculations and model in the simulator.

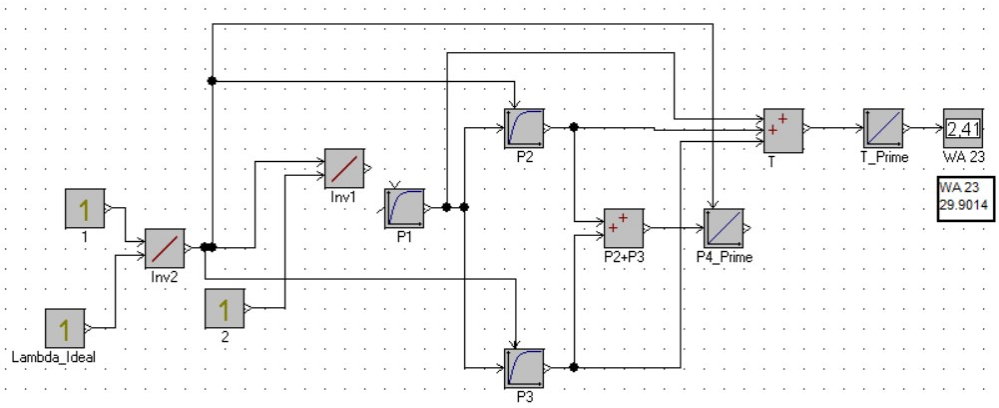


Figure 32: Ideal model diagram

As it can be seen, the result is very near to 30 time units. By applying Equation 7 with the value of $\lambda = 0.05$ it is clear that the implementation of the model is correct, and that the value should be 30. The reason behind the difference between the pure calculations and the simulation is the same simulation, as due to the numeric methods used by the software, there is a certain error.

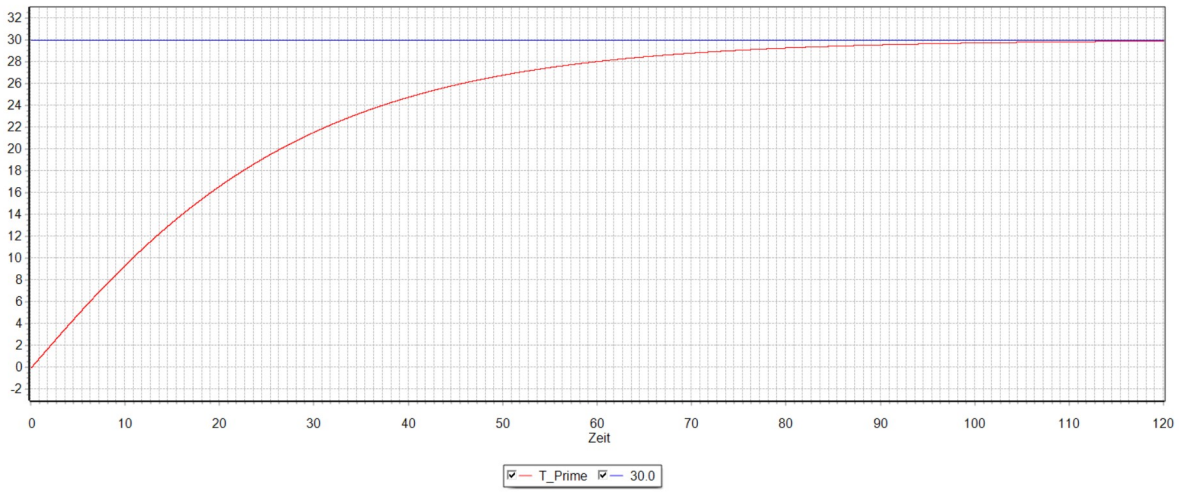


Figure 33: Ideal time life

Another important aspect or behavior that should be also seen on the final model is that the sum of the states must always be equal to 1 or 100%, and the intersection between the probabilities of the 4 states must be near to 13.8 units of time.

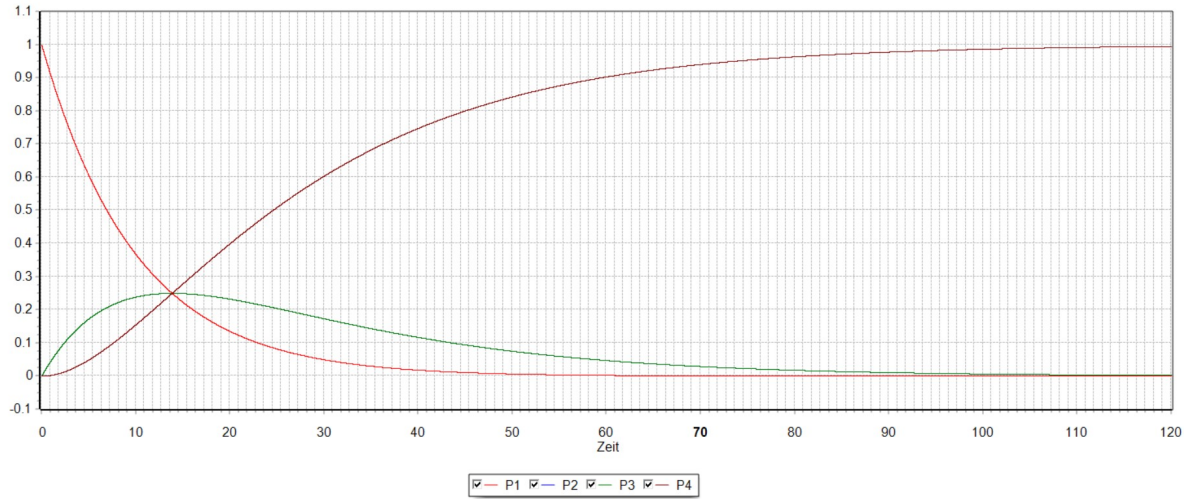


Figure 34: Ideal rates

The last aspect to consider is that, as well as in the previous graph, the failure and survival rates will complement each other, and that the intersection of these two will happen around 24.56 units of time, and the time life will come when the failure rate is at about 60%, as in Figure 36.

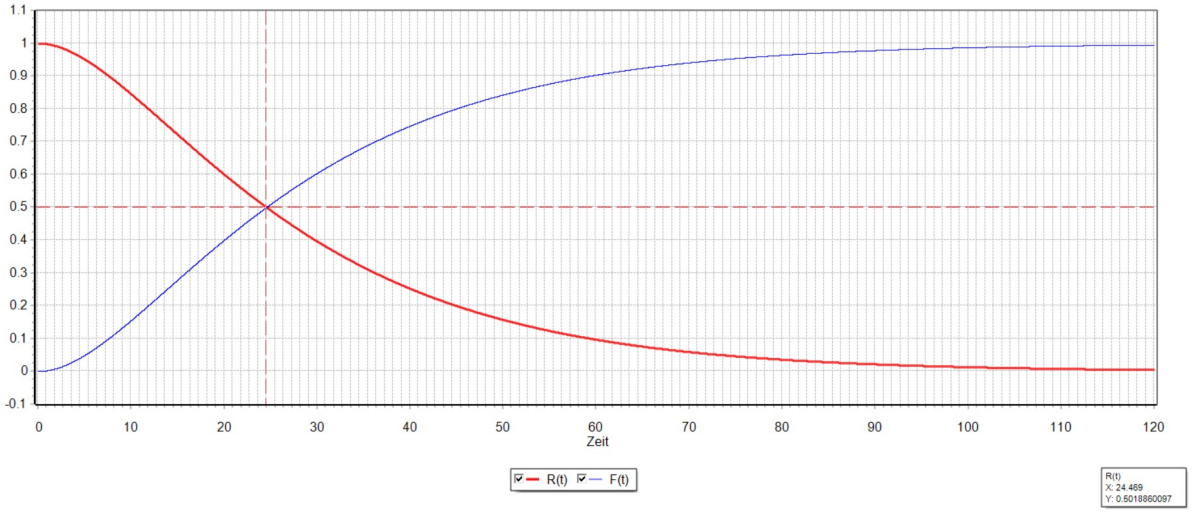


Figure 35: Intersection between failure and survival rates

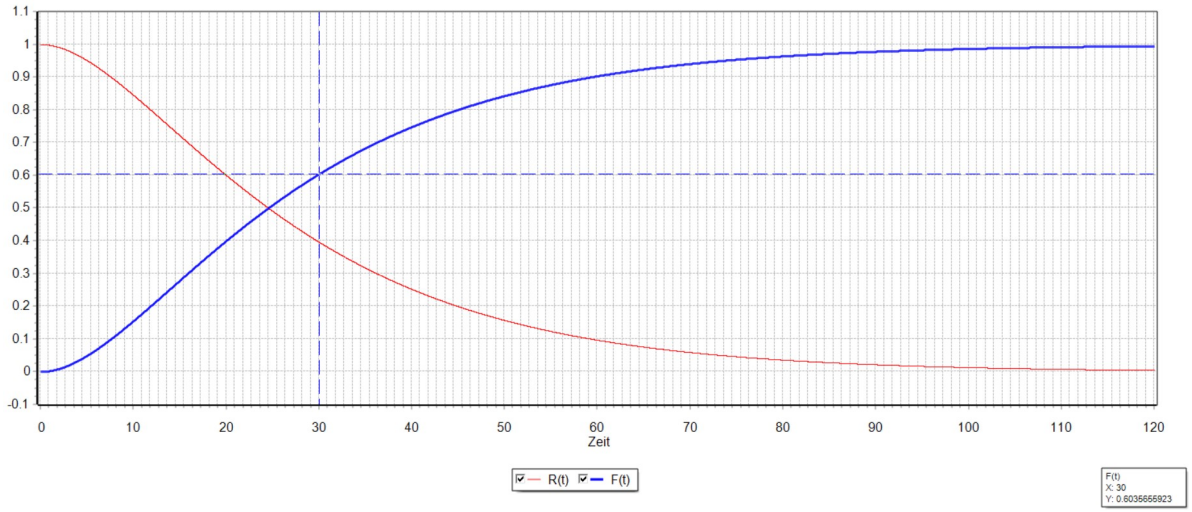


Figure 36: Failure probability at the obtained lifetime

One last thing, returning to the mathematical model, the system matrix will look as follows:

$$\begin{bmatrix} \dot{P_1}(t) \\ \dot{P_2}(t) \\ \dot{P_3}(t) \\ \dot{P_4}(t) \end{bmatrix} = \begin{bmatrix} -2(0.05) & 0 & 0 & 0 \\ (0.05) & -(0.05) & 0 & 0 \\ (0.05) & 0 & -(0.05) & 0 \\ 0 & (0.05)bs & (0.05) & 0 \end{bmatrix} \begin{bmatrix} P_1(t) \\ P_2(t) \\ P_3(t) \\ P_4(t) \end{bmatrix} \tag{16}$$

with its eigenvalues being:

$$\alpha = \begin{bmatrix} 0 \\ -(0.05) \\ -(0.05) \\ -(0.1) \end{bmatrix} \tag{17}$$

And the suggested simulation step size being of:

$$\Delta t < \frac{1}{4(2(0.05))} = \frac{1}{0.4} = 2.5TU \tag{18}$$

7 Simulation models

Once the theoretical framework, the calculations, the MLP model and the transfer functions of all the components are complete. Now we proceed to make the relevant connections. As can be seen in Figure 31, this is the configuration to be followed for the redundant active Markov model, in this configuration the transfer functions calculated in the previous step will be added and the connections will be made.

7.1 Simulation parameters

Regarding the requirements for the simulation, first of all it is necessary to define the simulation time, which will be 120 TU and the step of each pulse, this being 0.01 TU, just because practical experience tell us that it should give a precise result. The next task on the list is to define the stress conditions or the values of the inputs that will help us to determine the system lifetime. In Figures 37 and 38 we can see both inputs defined as a table and graphically. The first input or $X_1 = \beta_1$ will stand for the temperature, while the second input $X_2 = \beta_2$ will stand for the humidity or moisture of the environment; therefore, they both stand for the changing of environmental conditions.

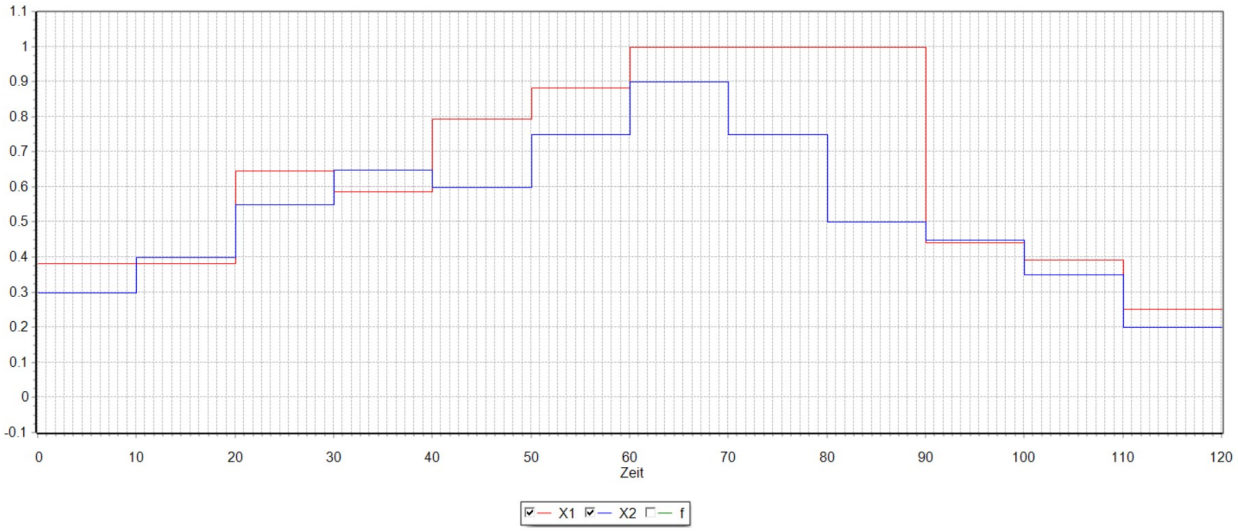


Figure 37: Input graph

H]

index	time period [TU]	value β_1	value β_2
0	0 bis 10	**0,382**	**0,3**
1	10 bis 20	**0,382**	**0,4**
2	20 bis 30	**0,647**	**0,55**
3	30 bis 40	**0,588**	**0,65**
4	40 bis 50	**0,794**	**0,6**
5	50 bis 60	**0,882**	**0,75**
6	60 bis 70	**1**	**0,9**
7	70 bis 80	**1**	**0,75**
8	80 bis 90	**1**	**0,5**
9	90 bis 100	**0,441**	**0,45**
10	100 bis 110	**0,394**	**0,35**
11	110 bis 120	**0,253**	**0,2**

Figure 38: Input table

7.2 Connections

The connections are quite simple as for Equation 11 this is represented by Figure 39, in which an integrator was also added, which helps to calculate the average value together with the value of the elapsed time, configured as a divisor of the integral result. The result of this scheme will allow us to calculate $\tilde{\lambda}$, which will give us a general idea of the stress conditions.

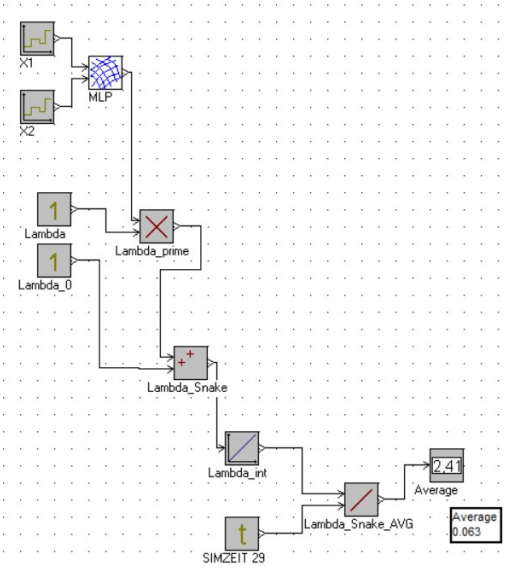


Figure 39: Simulation of the λ equation: scheme

The following section is merely the configuration of Figure 31 in the simulator. As can be seen in Figure 40, blocks of transfer functions are used to represent the different states. There is no input per se, but the system itself must tend towards a behavior. What is seen in the connections is the value of $\tilde{\lambda}$ applying

to each block.

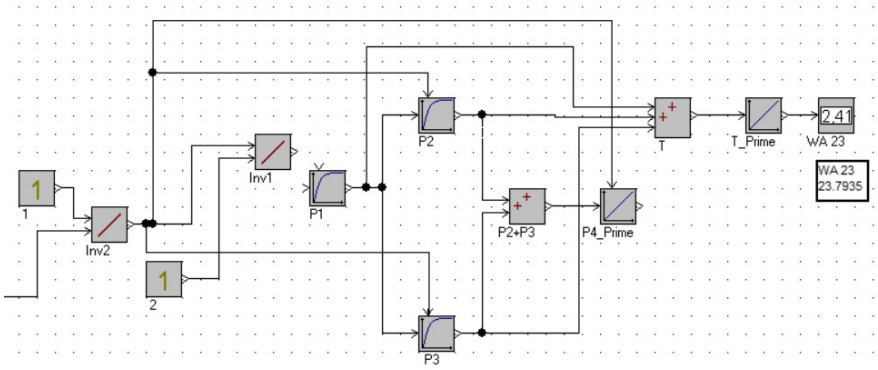


Figure 40: Simulation of the state model: scheme

7.3 Comparison

The major differences between the classical Markov model and the hybrid model come from the origin of λ, since in the case of the classical model,

lambda is a constant variable, which, is already given by the mathematical model and this is $\lambda = 0.05$. However, in the hybrid model, the life time calculation is improved by the use of a dynamic λ, which should, in principle, be much closer to reality.

7.3.1 Classical Markov model

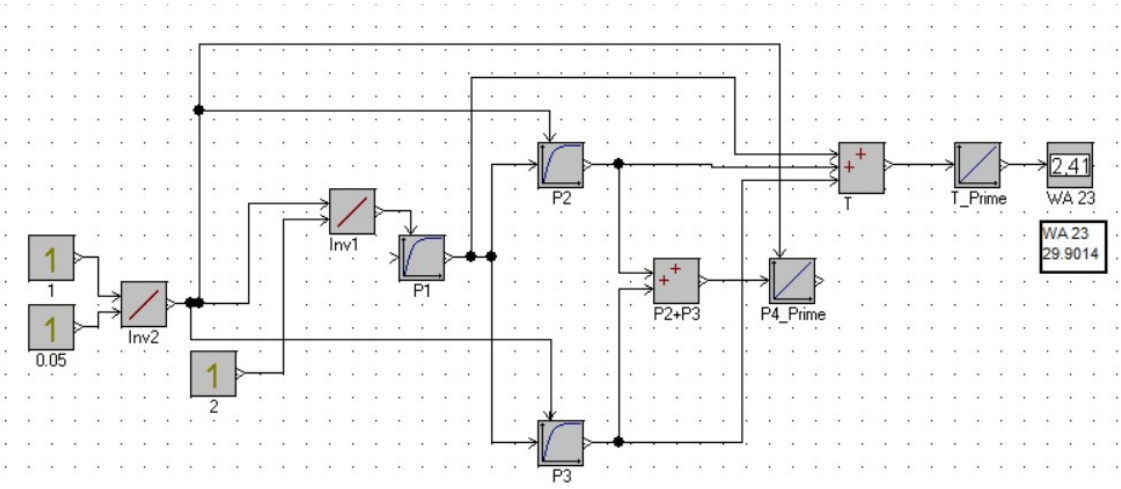


Figure 41: Classical Markov model: scheme

As for the classical model, λ is directly applied as a constant to the other blocks of the states.

7.3.2 Hybrid Markov model

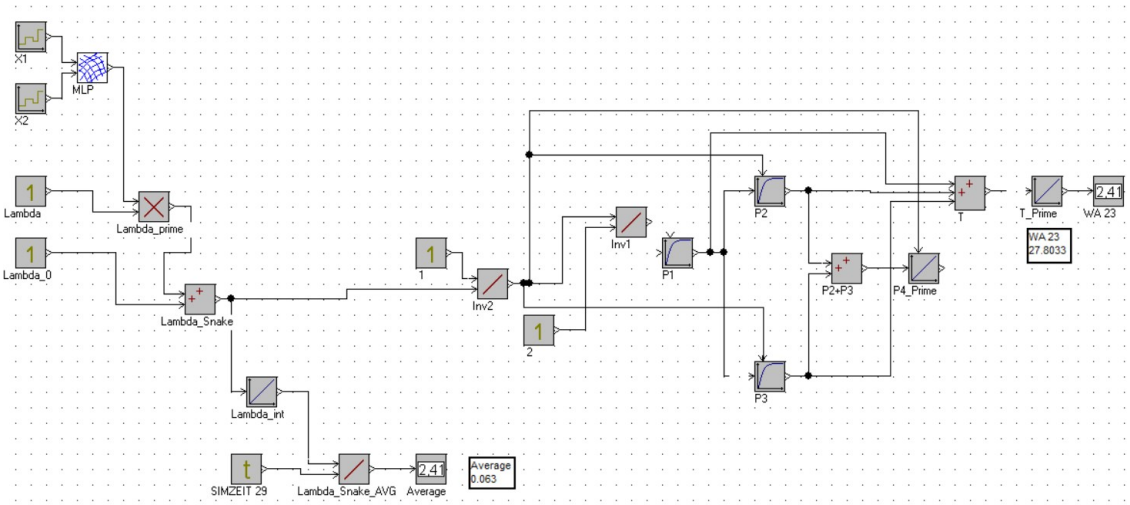


Figure 42: Hybrid Markov model: scheme

While for the hybrid model, the raw value of λ from the MLP value and the mathematical model is directly applied to the state blocks to perform the calculations.

8 Simulation results

As for the results, 5 graphs will be obtained from the simulation. The first one describes the behavior of the stress conditions or inputs as well as the output from the MLP model. A second one with the λ value together with the average $\tilde{\lambda}$, and the ideal λ of 0.05 . These two should be the same for both the classical and hybrid models. Subsequently, the third plot is of the probability that the model is in one of the four states. In this graph, the most relevant value is the intersection of all the states, since the effects of lifetime can be reflected in this value. The fourth graph represents the lifetime of the model and therefore of the system being modeled. And the fifth represents the failure and survival rates together.

8.1 Raw results

As for the results of the first plot in Figure 43, there is not much to say, it is simply how the model would react to the stress conditions introduced. It should be emphasized that the results meet the expectations of range between [0,1], so there is no error or strange situation that jumps out at first sight.

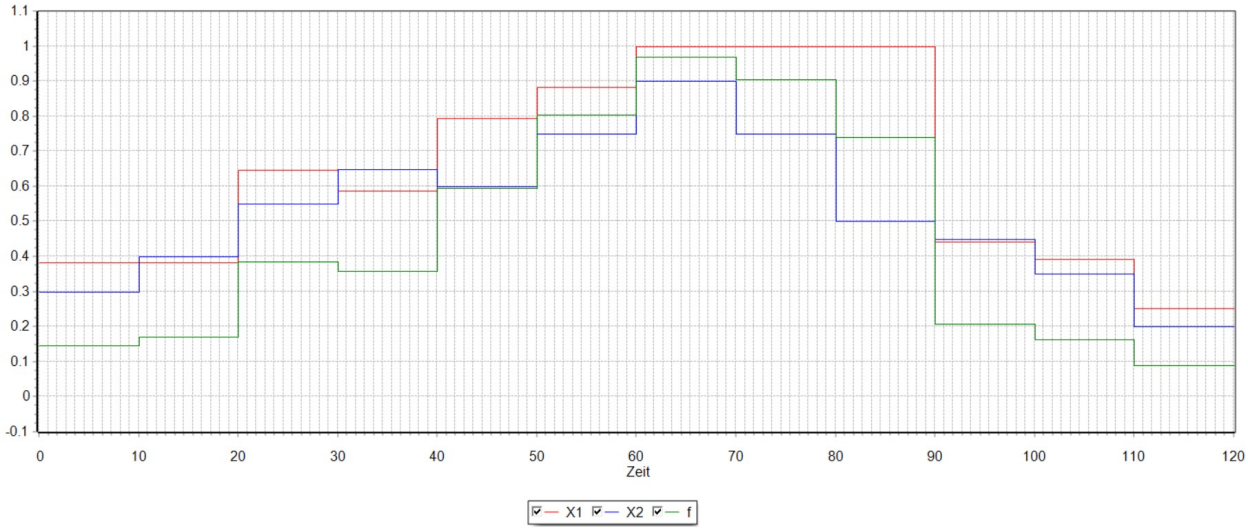


Figure 43: Stress conditions and MLP output graph

For the second plot in Figure 44, average $\tilde{\lambda}$ divides the λ output into two parts, where, precisely because of the λ values above the average $\tilde{\lambda}$ value, this last variable rises above the ideal value of 0.05, which in turn, mathematically reduces the lifetime.

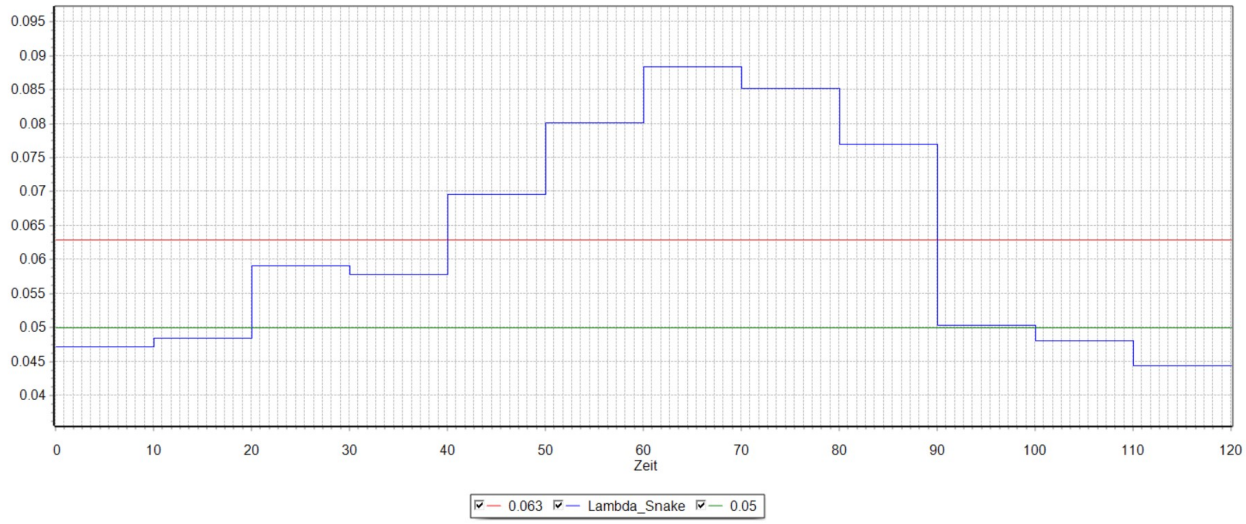


Figure 44: $\tilde{\lambda}$ calculation

8.2 Comparison

As already mentioned, the main points of interest will be the point of intersection in the third plot between the probability of the four states, since depending on where this is, the lifetime of the model can be approximated for constant λ. Another point to take into account, of course, will be the value of the lifetime of the simulations because with them the differences between one model and the other will be directly seen.

8.2.1 Classical Markov model

In the classical model, the intersection of the probabilities of the 4 states occurs around 13.8 time units, a situation that is reflected in the fourth graph, which has a lifetime of around 30 time units.

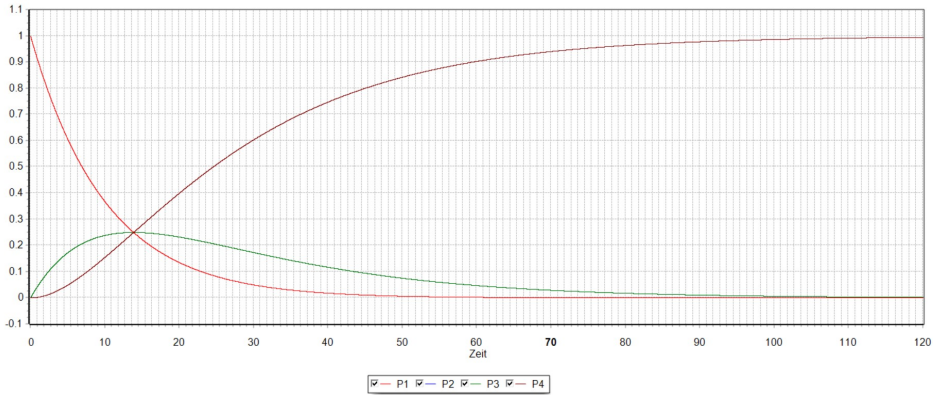


Figure 45: Classical Markov model: states probabilities

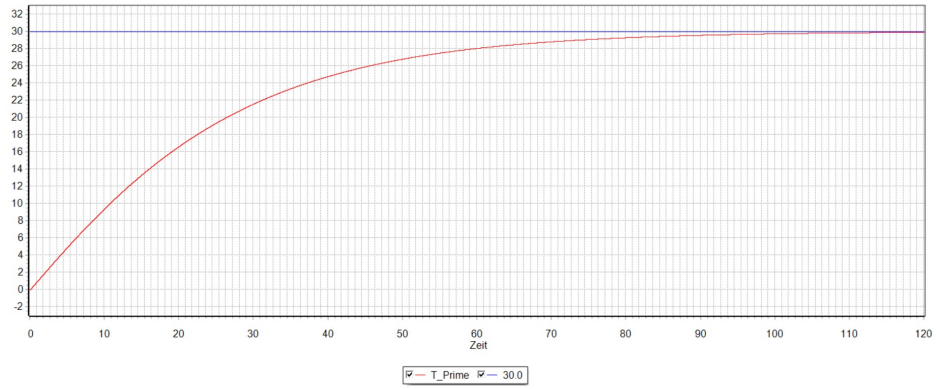


Figure 46: Classical Markov model: lifetime

For the last graph, we get that the intersection between the failure and survival rates is about 24.56 units of time, while the obtained lifetime occurs when the failure probability is at about 57.5%.

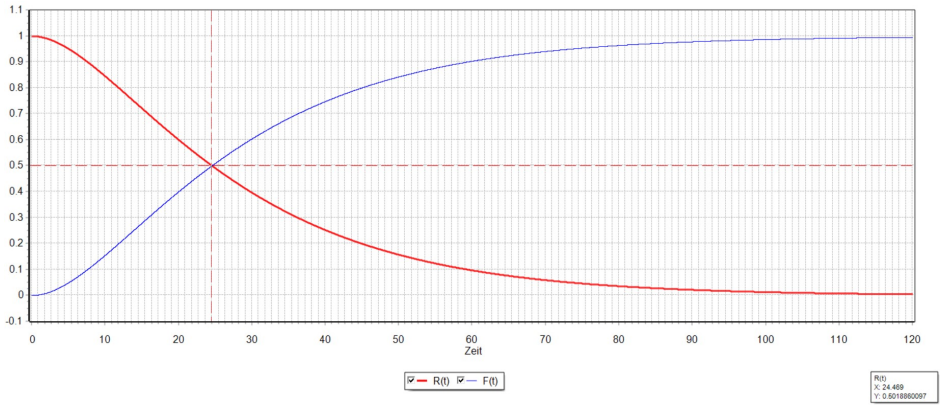


Figure 47: Classical Markov model: failure and survival rates

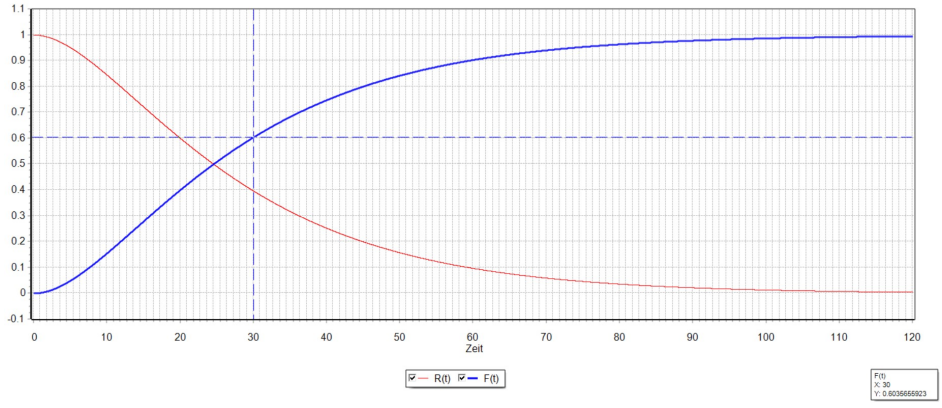


Figure 48: Classical Markov model:Failure rate at obtained lifetime

8.2.2 Hybrid Markov model

While in the classical model, the intersection of the probabilities of the 4 states is around 14.4 time units, which translates into 27.8 time units in the life time.

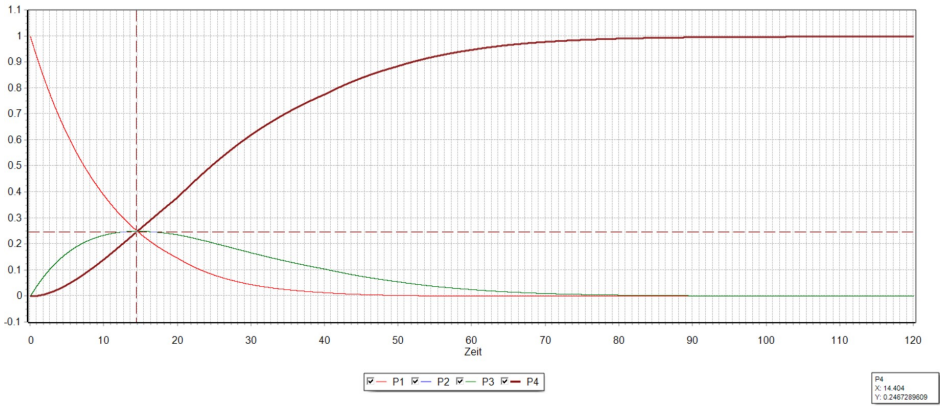


Figure 49: Hybrid Markov model: states probabilities

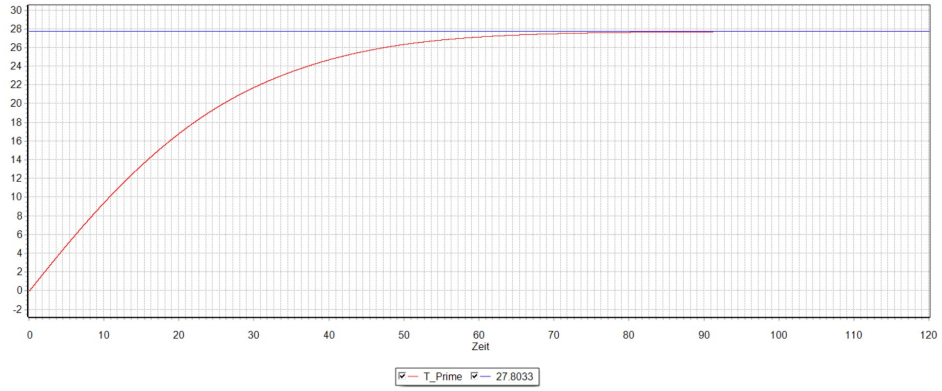


Figure 50: Hybrid Markov model: lifetime

For the last graph, we get that the intersection between the failure and survival rates is now about 24.56 units of time, while the obtained lifetime occurs when the failure probability is at about 57.5%.

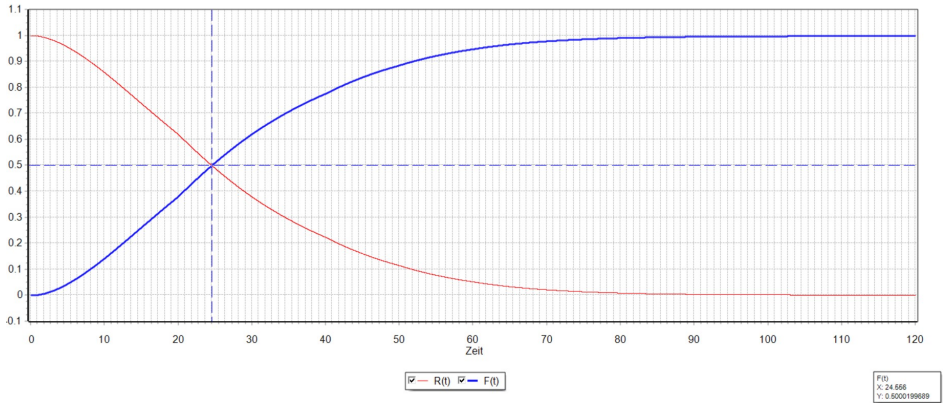


Figure 51: Hybrid Markov model: lifetime failure and survival rates

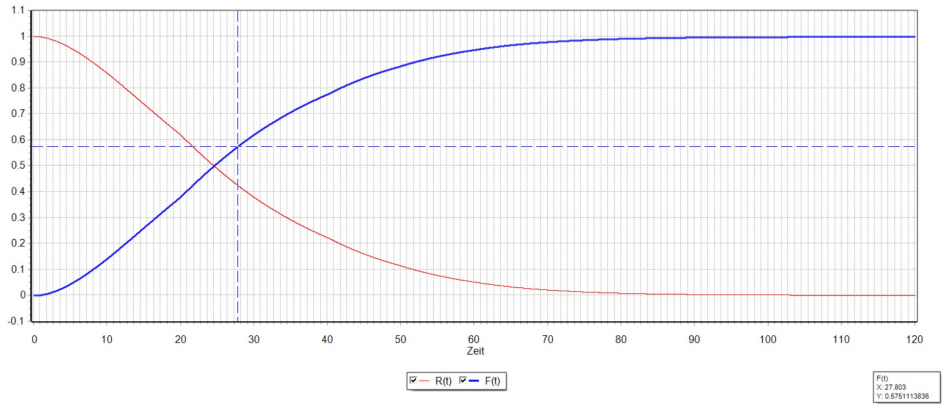


Figure 52: Hybrid Markov model: Failure rate at obtained lifetime

9 Evaluation

Finally we come to the evaluation section. After having done the corresponding simulation of the states, it was obtained that the hybrid model results in a shorter lifetime than the classical model.However, the reasons behind these results, are not yet explained.

9.1 Comparison

Compared to the classical model the hybrid model shows a decrease in lifetime. This directly means that the stress conditions are quite demanding and, by placing more demands on the model, they detract from its lifetime. Mathematically, this can be understood by the only variable contributing to the changes in the system being $\tilde{\lambda}$. While in the classical model λ is equal to 0.05, in the hybrid model the average of this value equals 0.063. In principle, the difference is not so large, but since it is an exponential distribution, small differences are amplified. The closer $\tilde{\lambda}$ is to that of the classical model, 0.05, the closer the result will resemble the theoretical results. Therefore, just with this value it can already be concluded that the system is under higher environmental conditions and will under-perform.

For the direct comparison of the graphs, we can see in Figure 53 that the states probabilities is almost the same for both models; nonetheless, as it was already stated, the hybrid model takes a longer time to reach the 25% for all states, which could indicate a higher life time, but this is not the case as seen in Figure 54.

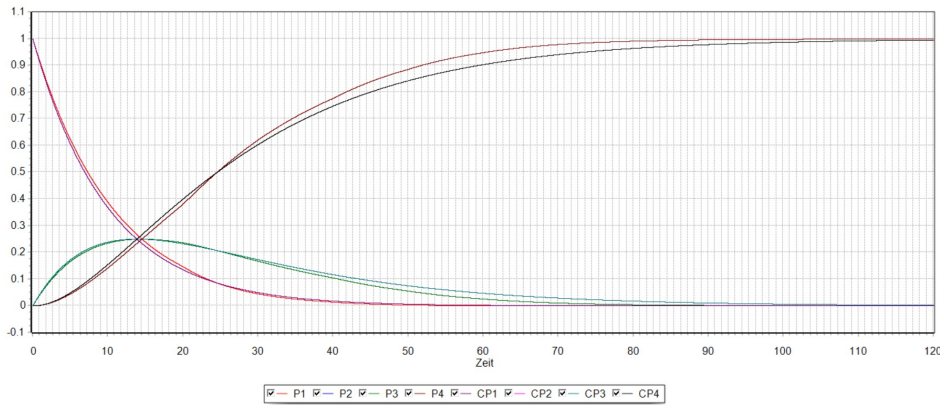


Figure 53: Comparison between the states probabilities

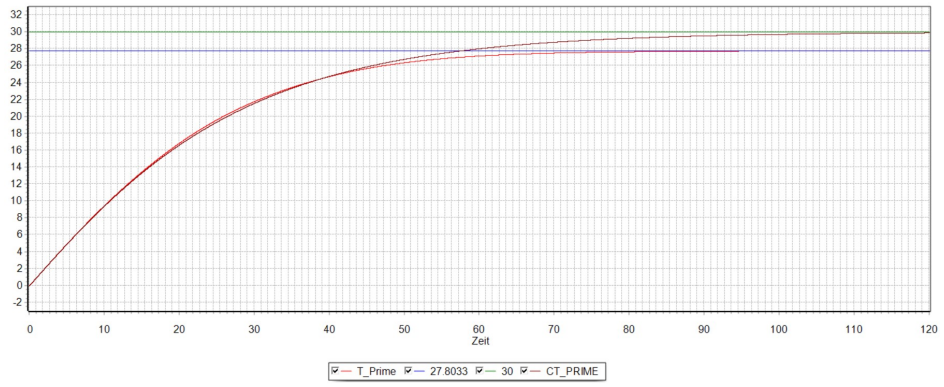


Figure 54: Life-time comparison

Visually, it can be seen in Figure 55, as the intersection between the failure and survival rate occurs earlier with the hybrid model, meaning that the failure rate when the life-time occurs is also higher than in the classical model, and that the hybrid model presents a much quicker decay than the classical one.

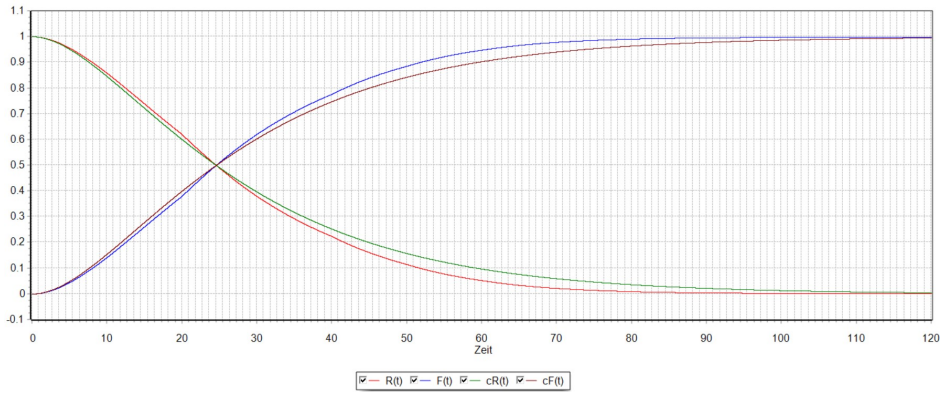


Figure 55: Failure and survival rates comparison

With all this information in can be deducted that even though the classical model presents a much more optimistic projection. The hybrid model presents a much reserved prognosis, based on the actual stress conditions, and shows a more precise model of reality.

9.1.1 Dynamics: Classical Markov model

On the part of the classical model, the value of λ does not undergo changes and the exponential function takes advantage of the 120 units of simulation time. To understand this a little better, we will use Equation 15 to calculate the time of intersection between the 4 graphs, which, being always limited to their sum equal to 1 or 100%, requires each one to have a value of 25%.

For the classical model we obtain:

$$\frac{P_1(s)}{\emptyset} = \frac{1}{\frac{1}{2\lambda}s + 1}$$

$$P_1(t) = e^{-2\lambda t}$$

$$0.25 = e^{-(2)(0.05)(t)}$$

$$t = 13.863$$

The result is pretty close to the simulated one and mathematically show the impact of a constant stress on the model, and how it reduces its lifetime with a constant λ.

9.1.2 Dynamics: Hybrid Markov model

While for the hybrid model, Figure 44 shows that the gross λ value varies over an equilibrium point, this being the average $\tilde{\lambda}$. However, the contribution of the exponents of the exponential distribution at the lowest times or those that have the greatest impact on the final result, are minimized by presenting at those times a $\tilde{\lambda}$ much lower than the classical value of 0.05. For this reason it can also be seen that the time units for the intersection of the state probability graphs are even greater than those of the ideal model, but have a shorter lifetime.

To calculate the intersection, we first need to calculate the final value before the change of exponent.

$$P_{1_{t1}}(10) = e^{-2(0.0473)(10)} = 0.388$$

Then we got to calculate the adjustment for the change of exponent, and calculate the intersection:

$$e^{-2(0.0485)(10)} = 0.379$$

$$adjustment = 0.388 - 0.379 = 0.009$$

$$P_{1_{t2}}(t) = e^{-2(0.0485)(10+t_{int})} + 0.009 = 0.25$$

$$t_{int} = 4.679$$

$$t_{total} = 10 + t_{int} = 14.679\,TU$$

The result is quite similar to the simulated one, and explains why although the intersection of the hybrid model occurs some time units after, the overall lifetime is lower than the one of the classical model. Because of the contributions of the different values of λ above the ideal value.

10 Conclusion

In conclusion, the hybrid Markov model provides greater accuracy in terms of the correct calculation of the units of time required to provide maintenance or preventive measures to a system. This is really important, since this calculation will provide results that are not only closer to reality, but also more viable in the framework of industry and finance. This is because maintenance not only represents expenses in itself, but also losses due to temporary system downtime.

Regarding the specific case study, it can also be concluded that after 27 time units it is advisable to plan maintenance, since there is still a certain cushion in case of slightly more stress or inaccurate counting of time units in comparison to the 30 time units from the classical model. Also, the instant failure rate will be at about 55.8%, which is not that high in comparison to the survival rate, but is more probable a failure than the survival of the system.

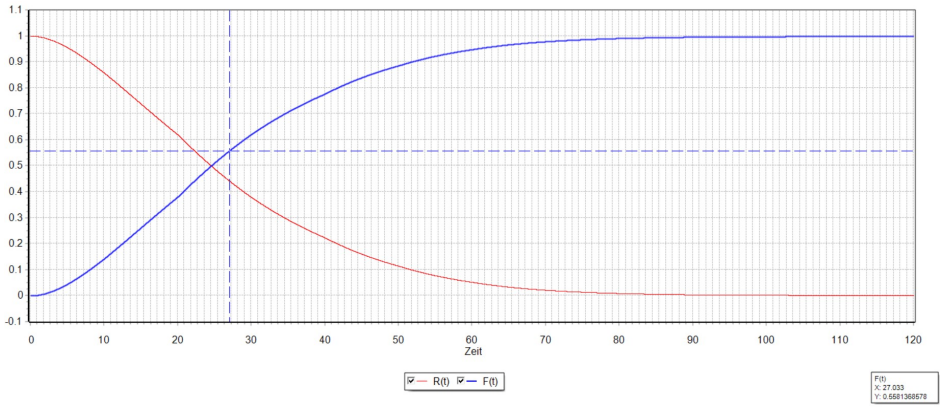


Figure 56: Failure rate at suggested maintenance of 27 units of time

Another good alternative is to consider applying maintenance since 25 units of time, as the instant failure rate is about 51% at this time, and could give us a good time range for the maintenance in case for some reason there are delays or unforeseen events.

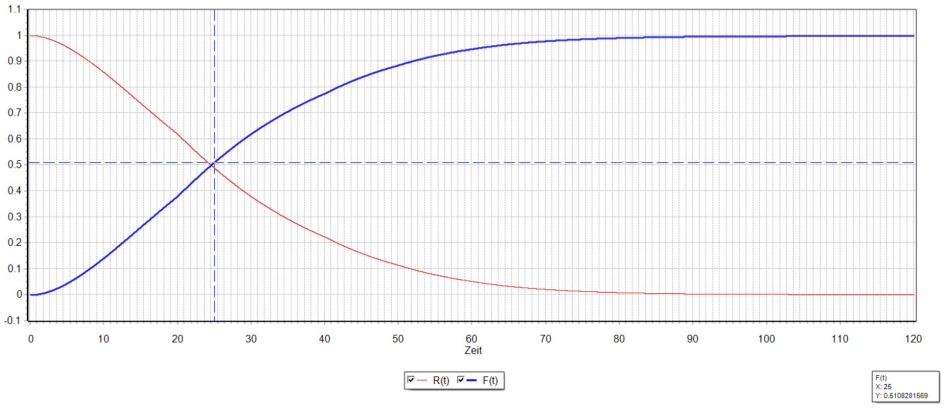


Figure 57: Failure rate at suggested maintenance of 25 units of time

Finally, care must also be taken that the initial stress conditions are not too large, since, as seen in the previous section, they have the greatest impact on the rapid increase in the probability of failure, and lower stresses at the beginning will mean that the system will operate longer with really low odds of failure.

Under all these considerations, it can be concluded that 27 time units is the most suitable time to apply maintenance to the system, since it presents a 10% safety factor in relation to the classical model, and even a 3% safety factor in relation to the hybrid model. Both these safety factors are a good cushion in case of any unexpected events and is a good rounded value.

Appendices

A Appendix of codes

A.1 Matlab code for data preparation and pre-processing

```
%% Code description:
% Author: Jair
% Date: 1.11.21
% Version: 1.0

%% Import data from the Excel file
data = readtable('Data_ANN_2021_cw.xlsx','Range','C4:E73');

%% Convert the rough data into vectors
% Conversion
x1 = data{:,1};
x2 = data{:,2};
y = data{:,3};

%% Plot the data in a 3D enviroment
% Plot
figure(1);
stem3(x1, x2, y);
grid on;

%% Create a mesh grid to construct a continous surface to plot
% Sensibility of the grid
x1v = linspace(min(x1), max(x1), 100);
x2v = linspace(min(x2), max(x2), 100);

% Grid vectors
```

```
[X1,X2] = meshgrid(x1v, x2v);
Y = griddata(x1,x2,y,X1,X2);

% Plot
figure(2);
surf(X1, X2, Y);
grid on;

%% Eliminate out of range values
% Counter of eliminated values
n = 0;

% Loop
for i = 1:length(x1)
    if (y(i-n)<0)
        y(i-n)=[];
        x1(i-n)=[];
        x2(i-n) = [];
        n=n+1;
    end
end

%% Eliminate duplicate values, as well as no-continous data behavior
% Initialize variables
point1= 0;
m = 0;
values = [];

% Loop
for j = 1:length(x1)
    if (x1(j-m)~=point1)
        point1 = x1(j-m);
```

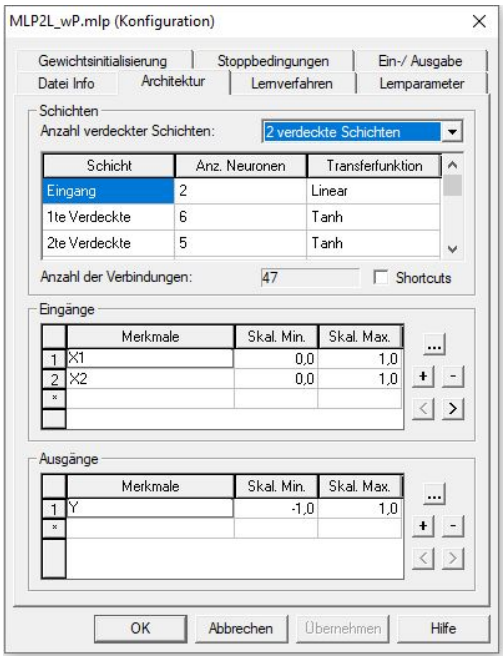
```
        values = [];
        k = 1;
    end
    if (x1(j-m) == point1)
        values(k)=x2(j-m);
        k=k+1;
        if (k>3 && j-m>1)
            for l = 1:(k-2)
                if (values(k-1)== values(l))
                    y(j-m)=[];
                    x1(j-m)=[];
                    x2(j-m) = [];
                    m=m+1;
                end
            end
        if (((j-(m+2))>1)&&(sign(y(j-(m+2))-y(j-(m+1)))~=sign(y(j-(m+1))...
            -y(j-(m)))))
            y(j-m)=[];
            x1(j-m)=[];
            x2(j-m) = [];
            m=m+1;
        end
        end
    end
end
y(10)=[];
x1(10)=[];
x2(10) = [];

%% Plot new data

% Plot data
```

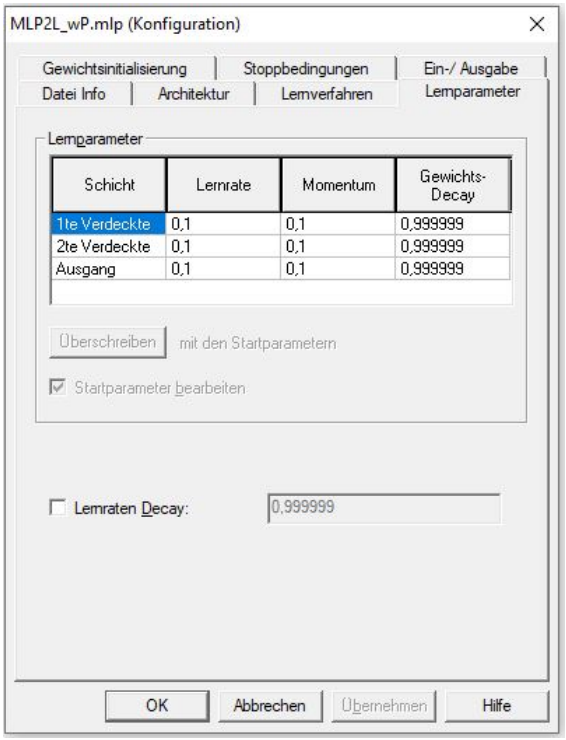
```
figure(3);
stem3(x1, x2, y);
grid on;

% Create the surface
x1v = linspace(min(x1), max(x1), 100);
x2v = linspace(min(x2), max(x2), 100);
[X1,X2] = meshgrid(x1v, x2v);
Y = griddata(x1,x2,y,X1,X2);

% Plot surface
figure(4);
surf(X1, X2, Y);
grid on;

%% Select training and test data
% Initialize variables
ss = 18;
sample = [];
counter = 1;
counter2 = 1;
test = [];

% Loop
tolerance = .2;

for h = 1:length(x1)
    if ((counter<=ss)&&(x1(h)<(1-tolerance)*max(x1))&&(x1(h)>...
            (1+tolerance)*min(x1))&&(x2(h)< (1-tolerance)*max(x2))&&...
            (x2(h)>(1+tolerance)*min(x2))&&(y(h)<(1-tolerance)*max(y))&&...
            (y(h)> (1+tolerance)*min(y)))
        test(counter,1) = x1(h);
```

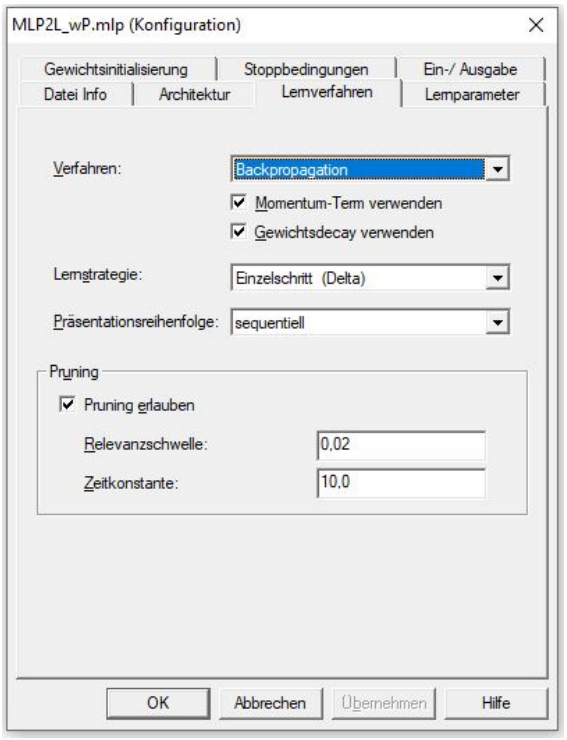
```
        test (counter,2) = x2(h);
        test(counter,3) = y(h);
        counter = counter + 1;
    else
        sample(counter2,1) = x1(h);
        sample(counter2,2) = x2(h);
        sample(counter2,3) = y(h);
        counter2 = counter2 + 1;
    end
end

%% Create new file with training and test data
%FileName = 'DataJair2.xlsx';
%xlswrite(FileName ,sample,'Train');
%xlswrite(FileName ,test,'Test');
```

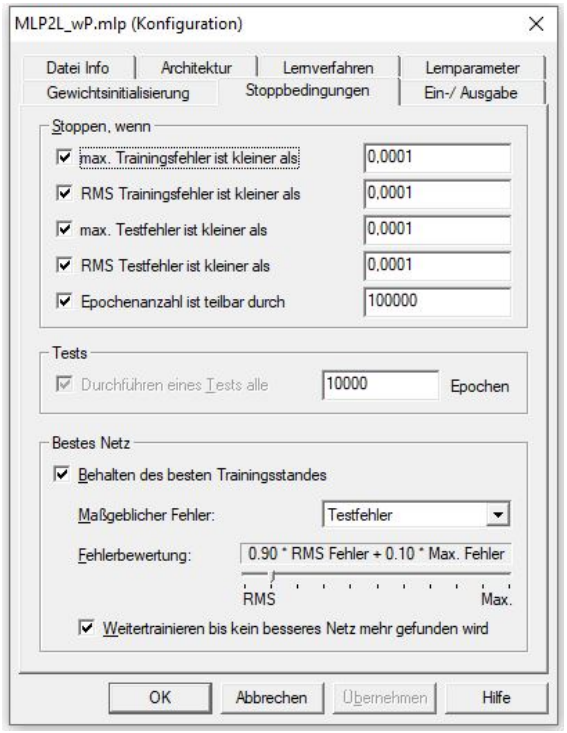
B Appendix of MLP design

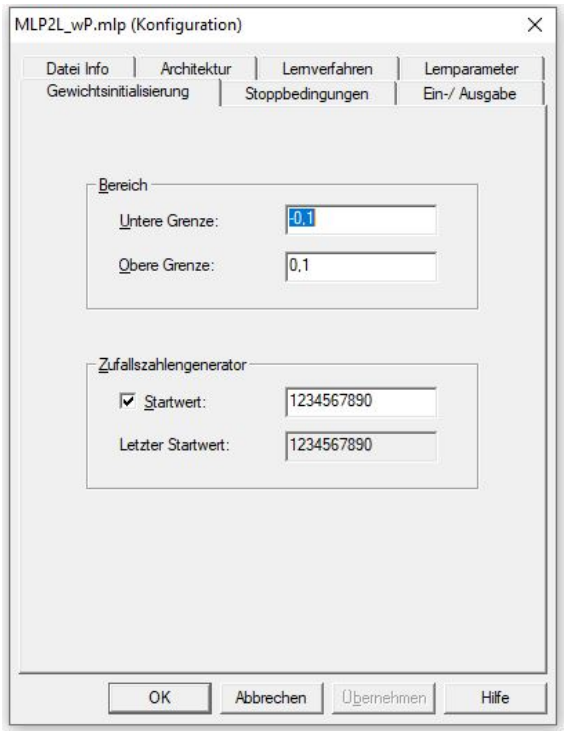
B.1 Before pruning

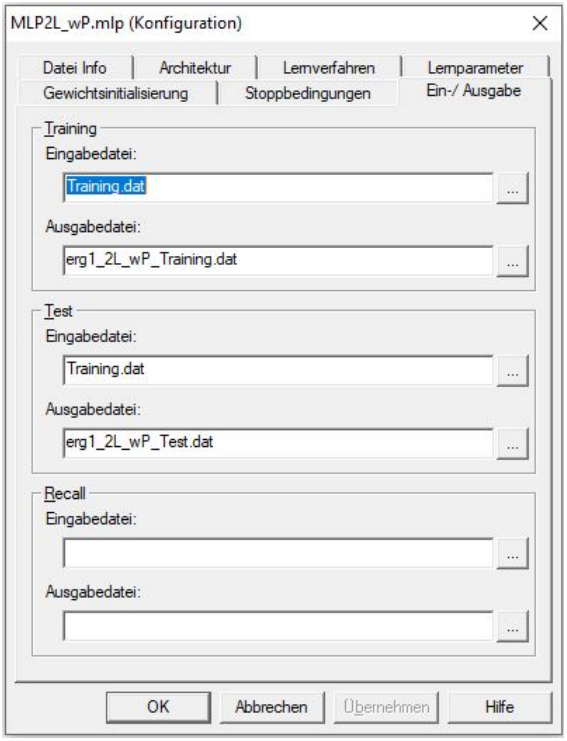




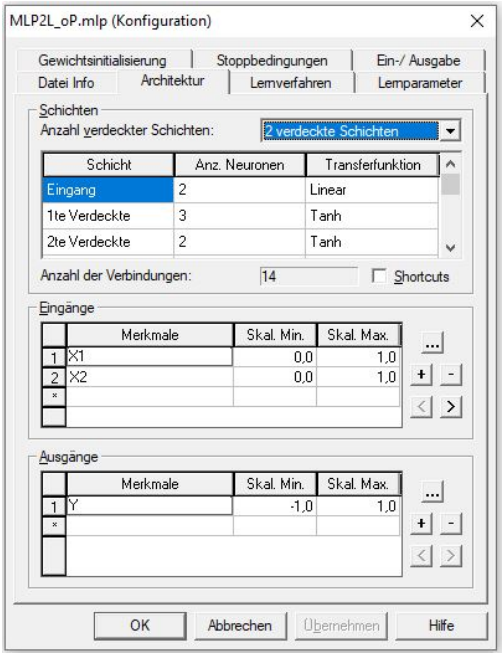

MLP2L_wP.mlp (Konfiguration)
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Datei Info
Architektur
Lernverfahren
Lernparameter
Verfahren:
Backpropagation
Momentum-Term verwenden
Gewichtsdecay verwenden
Lernstrategie:
Einzelschritt (Delta)
Präsentationsreihenfolge:
sequentiell
Pruning
Pruning erlauben
Relevanzschwelle:
0,02
Zeitkonstante:
10,0
OK
Abbrechen
Übernehmen
Hilfe

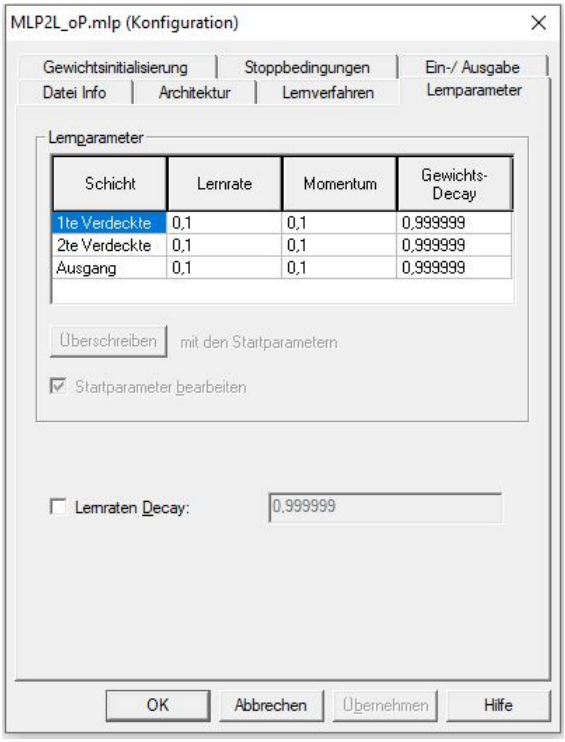

MLP2L_wP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Stoppen, wenn
max. Trainingsfehler ist kleiner als
0,0001
RMS Trainingsfehler ist kleiner als
0,0001
max. Testfehler ist kleiner als
0,0001
RMS Testfehler ist kleiner als
0,0001
Epochenanzahl ist teilbar durch
100000
Tests
Durchführen eines Tests alle
10000
Epochen
Bestes Netz
Behalten des besten Trainingsstandes
Maßgeblicher Fehler:
Testfehler
Fehlerbewertung:
0.90 * RMS Fehler + 0.10 * Max. Fehler
RMS
Max.
Weitertrainieren bis kein besseres Netz mehr gefunden wird
OK
Abbrechen
Übernehmen
Hilfe

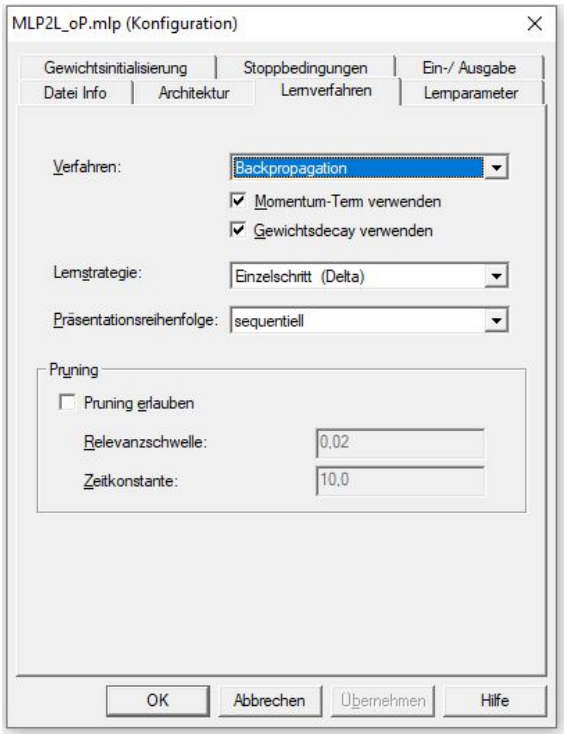

MLP2L_wP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Bereich
Untere Grenze:
-0,1
Obere Grenze:
0,1
Zufallszahlengenerator
Startwert:
1234567890
Letzter Startwert:
1234567890
OK
Abbrechen
Übernehmen
Hilfe

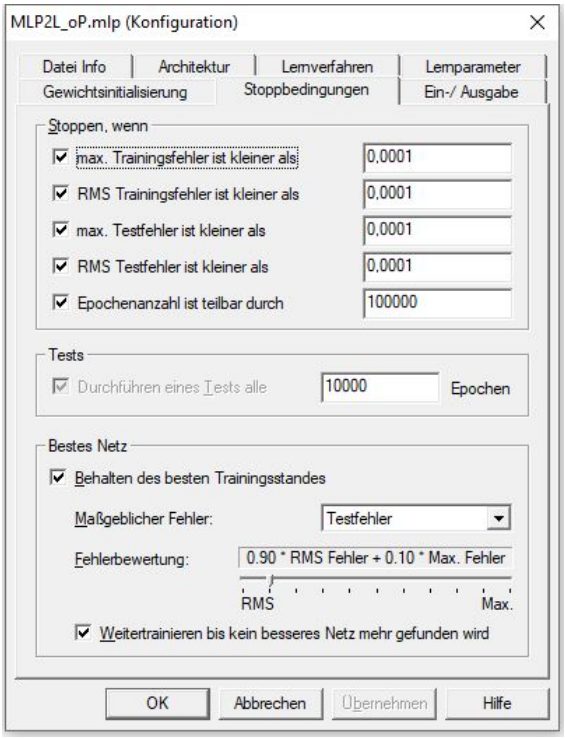

MLP2L_wP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Training
Eingabedatei:
Training.dat
Ausgabedatei:
erg1_2L_wP_Training.dat
Test
Eingabedatei:
Training.dat
Ausgabedatei:
erg1_2L_wP_Test.dat
Recall
Eingabedatei:
Ausgabedatei:
OK
Abbrechen
Übernehmen
Hilfe

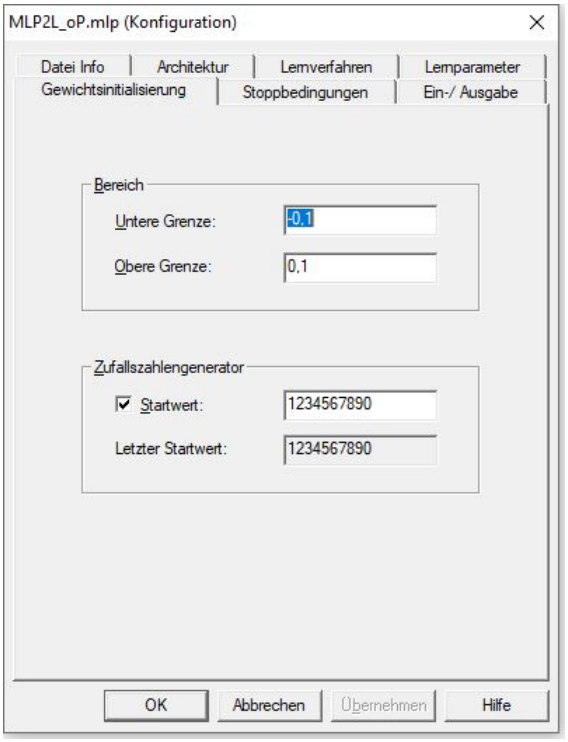
B.2 After pruning

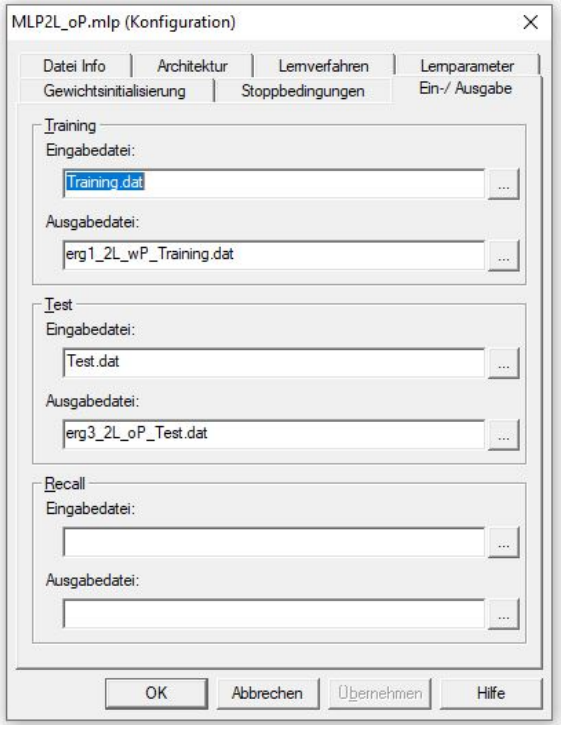





MLP2L_oP.mlp (Konfiguration)
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Datei Info
Architektur
Lernverfahren
Lernparameter
Verfahren:
Backpropagation
Momentum-Term verwenden
Gewichtsdecay verwenden
Lernstrategie:
Einzelschritt (Delta)
Präsentationsreihenfolge:
sequentiell
Pruning
Pruning erlauben
Relevanzschwelle:
0,02
Zeitkonstante:
10,0
OK
Abbrechen
Übernehmen
Hilfe


MLP2L_oP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Stoppen, wenn
max. Trainingsfehler ist kleiner als
0,0001
RMS Trainingsfehler ist kleiner als
0,0001
max. Testfehler ist kleiner als
0,0001
RMS Testfehler ist kleiner als
0,0001
Epochenanzahl ist teilbar durch
100000
Tests
Durchführen eines Tests alle
10000
Epochen
Bestes Netz
Behalten des besten Trainingsstandes
Maßgeblicher Fehler:
Testfehler
Fehlerbewertung:
0.90 * RMS Fehler + 0.10 * Max. Fehler
RMS
Max.
Weitertrainieren bis kein besseres Netz mehr gefunden wird
OK
Abbrechen
Übernehmen
Hilfe


MLP2L_oP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Bereich
Untere Grenze:
-0,1
Obere Grenze:
0,1
Zufallszahlengenerator
Startwert:
1234567890
Letzter Startwert:
1234567890
OK
Abbrechen
Übernehmen
Hilfe


MLP2L_oP.mlp (Konfiguration)
Datei Info
Architektur
Lernverfahren
Lernparameter
Gewichtsinitialisierung
Stoppbedingungen
Ein-/ Ausgabe
Training
Eingabedatei:
Training.dat
Ausgabedatei:
erg1_2L_wP_Training.dat
Test
Eingabedatei:
Test.dat
Ausgabedatei:
erg3_2L_oP_Test.dat
Recall
Eingabedatei:
Ausgabedatei:
OK
Abbrechen
Übernehmen
Hilfe

C Appendix of tables

C.1 Initial set of data

Data base	ordered	by X1	(b1)		Data base	ordered	by X2	(b1)
	b1	b2	f			b1	b2	f
data pattern	X1	X2	Y		data pattern	X1	X2	Y
1	0.29	0.10	0.060		1	0.29	0.10	0.060
2	0.29	0.30	0.115		2	0.41	0.10	0.080
3	0.29	0.50	0.155		3	0.59	0.10	0.125
4	0.29	0.75	0.190		4	0.71	0.10	0.170
5	0.29	1.00	0.210		5	0.82	0.10	0.220
6	0.35	0.20	0.120		6	0.88	0.10	0.250
7	0.35	0.40	0.160		7	0.94	0.10	0.270
8	0.35	0.60	0.200		8	0.97	0.10	0.275
9	0.35	0.80	0.220		9	1.00	0.10	0.280
10	0.35	0.10	0.250		10	0.35	0.10	0.250
11	0.35	0.75	-1.000		11	0.88	0.10	0.252
12	0.41	0.10	0.080		12	0.35	0.20	0.120
13	0.41	0.30	0.160		13	0.50	0.20	0.160
14	0.41	0.50	0.210		14	0.65	0.20	0.220
15	0.41	0.75	0.260		15	0.85	0.20	0.360
16	0.41	1.00	0.285		16	0.29	0.30	0.115
17	0.50	0.20	0.160		17	0.41	0.30	0.160
18	0.50	0.40	0.250		18	0.59	0.30	0.250
19	0.50	0.60	0.300		19	0.71	0.30	0.335
20	0.50	0.80	0.330		20	0.82	0.30	0.435
21	0.59	0.10	0.125		21	0.88	0.30	0.490
22	0.59	0.30	0.250		22	0.94	0.30	0.535
23	0.59	0.50	0.335		23	0.97	0.30	0.545
24	0.59	0.75	0.410		24	1.00	0.30	0.550

Table 10 continued from previous page

25	0.59	1.00	0.450	25	0.88	0.30	0.488
26	0.65	0.20	0.220	26	0.35	0.40	0.160
27	0.65	0.40	0.350	27	0.50	0.40	0.250
28	0.65	0.60	0.430	28	0.65	0.40	0.350
29	0.65	0.80	0.480	29	0.85	0.40	0.560
30	0.71	0.10	0.170	30	0.94	0.40	-1.000
31	0.71	0.30	0.335	31	0.29	0.50	0.155
32	0.71	0.50	0.450	32	0.41	0.50	0.210
33	0.71	0.75	0.555	33	0.59	0.50	0.335
34	0.71	1.00	0.605	34	0.71	0.50	0.450
35	0.82	0.10	0.220	35	0.82	0.50	0.590
36	0.82	0.30	0.435	36	0.88	0.50	0.660
37	0.82	0.50	0.590	37	0.94	0.50	0.720
38	0.82	0.75	0.715	38	0.97	0.50	0.735
39	0.82	1.00	0.795	39	1.00	0.50	0.740
40	0.85	0.20	0.360	40	0.88	0.50	0.661
41	0.85	0.40	0.560	41	0.35	0.60	0.200
42	0.85	0.60	0.690	42	0.50	0.60	0.300
43	0.85	0.80	0.780	43	0.65	0.60	0.430
44	0.88	0.10	0.250	44	0.85	0.60	0.690
45	0.88	0.30	0.490	45	0.29	0.75	0.190
46	0.88	0.50	0.660	46	0.41	0.75	0.260
47	0.88	0.75	0.800	47	0.59	0.75	0.410
48	0.88	1.00	0.890	48	0.71	0.75	0.555
49	0.88	0.10	0.252	49	0.82	0.75	0.715
50	0.88	0.30	0.488	50	0.88	0.75	0.800
51	0.88	0.50	0.661	51	0.94	0.75	0.870
52	0.88	0.75	0.802	52	0.97	0.75	0.895
53	0.88	1.00	0.889	53	1.00	0.75	0.900
54	0.94	0.10	0.270	54	0.35	0.75	-1.000

Table 10 continued from previous page

55	0.94	0.30	0.535	55	0.88	0.75	0.802
56	0.94	0.50	0.720	56	0.35	0.80	0.220
57	0.94	0.75	0.870	57	0.50	0.80	0.330
58	0.94	1.00	0.970	58	0.65	0.80	0.480
59	0.94	0.80	0.250	59	0.85	0.80	0.780
60	0.94	0.40	-1.000	60	0.94	0.80	0.250
61	0.97	0.10	0.275	61	0.29	1.00	0.210
62	0.97	0.30	0.545	62	0.41	1.00	0.285
63	0.97	0.50	0.735	63	0.59	1.00	0.450
64	0.97	0.75	0.895	64	0.71	1.00	0.605
65	0.97	1.00	0.995	65	0.82	1.00	0.795
66	1.00	0.10	0.280	66	0.88	1.00	0.890
67	1.00	0.30	0.550	67	0.94	1.00	0.970
68	1.00	0.50	0.740	68	0.97	1.00	0.995
69	1.00	0.75	0.900	69	1.00	1.00	1.000
70	1.00	1.00	1.000	70	0.88	1.00	0.889

Table 10: Initial set of data

References

[1] W. Kästner, "Modeling and simulation basics," 2021.

[2] ——, "Soft computing artificial neural network," 2021.

[3] ——, "Evaluation of reliability," 2021.