ISSN 0280-5316
ISRN LUTFD2/TFRT--5759--SE

Lifetime Monitoring of Wind Turbines

Lars Nilsson

Department of Automatic Control
Lund University
November 2005

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
November 2005

Document Number
ISRNLUTFD2/TFRT--5759--SE

Author(s)
Lars Nilsson

Supervisor
Rolf Johansson at Automatic Control in Lund
Florian Krug at General Electric Global Research in München.

Sponsoring organization

Title and subtitle
Lifetime Monitoring of Wind Turbines (Livstidsövervakning av vindkraftverk)

Abstract
The aim of this thesis was to design and implement a lifetime monitoring system for a GE Energy wind turbine. Monitoring the loads on the main components of a wind turbine makes it possible to keep the lifetime of the components monitored. This information can be used to enable more flexible planning of a wind turbine's maintenance. Knowing the estimated remaining lifetime makes it possible to change the components before they break. This increases both the security and availability of the wind turbine. The information from the lifetime monitoring system could possibly also be used directly by the wind turbine's main controller in order to optimize the operation of the wind turbine regarding its components lifetime and the turbine's energy capture. Several load-cycle counting methods were investigated and compared to each other and the rainflow counting method was found to be the most suitable. It was adapted, implemented and tested on a PLC (Programmable Logical Controller) mounted to a HITL (Hardware In The Loop) real-time simulation system that simulated the behavior of a GE 1.5 s/sl wind turbine. A method for calculating the fatigue, using the result from the rainflow counting, was implemented. The whole monitoring system was designed and implemented to work online, i.e., continuously calculating and displaying the lifetime of the monitored components. In order to realize an online rainflow counter, a novel approach for classification of the data-series was developed. A prototype of the system was installed on a GE Energy wind turbine in Salzbergen, Germany. Several tests were here performed in order to validate the system and to compare the simulated results with the measured results.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
83

Recipient's notes

Security classification

Acknowledgment

This work was carried out for the Department of Automatic Control at the Lund Institute of Technology, Sweden, in cooperation with General Electric Global Research in Garching, Munich, Germany.

First of all I would like to thank Professor Rolf Johansson from the Department of Automatic Control at the Lund Institute of Technology, for his helpful advices and tips for my master thesis.

Thanks, to my supervisor Dr.-Ing. Florian Krug, GE Global Research, for believing in this project and always giving me the necessary support.

I would also like to thank Dipl.-Ing. Christian Schram for his encouragement and for the very helpful technical brainstorming sessions.

A special thank you to Wilhelm Feichter, GE Global Research, for the outstanding collaboration and all the technical help.

Thanks, to Thorsten Honekamp for his useful help in questions regarding rainflow counting and classification.

Thanks, to Friedrich Loh, GE Energy, for his assistance during the prototype installation in Salzbergen.

Thanks, to all my colleagues at GE Global Research for the friendship, the inspiring atmosphere, and the pleasant coffee-breaks.

Last, but not least, I would like to thank my girlfriend Natalie Terenya for her unyielding support and love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

During the last two decades, wind turbines have evolved tremendously. The efficiency, size and complexity have all increased. More and more plants are being built off-shore in order to take advantage of the higher average wind speeds there. These put very tough demands on the components of a wind turbine. Off-shore wind turbines specifically calls for very reliable components and systems given that maintenance costs are much higher off-shore than on-shore.

Estimating the components remaining lifetime enables easier and more flexible planning of needed service of the wind turbine. By replacing the components before they break, costs can be reduced. The costs are reduced since the interruption time, caused by replacing the component, is minimized. Also the insurance costs are assumed to decrease due to the improved security achieved when installing a condition monitoring system. For instance, Allianz Versicherungs-AG requires its customers to change bearing, gearbox, generator and blades after 40.000 hours (about 4 1/2 year) of operation. However, in the case of present condition monitoring systems in wind turbines, more favorable agreements can be reached [Wind03].

This thesis has been a part of the LUM[1] project from GE Global Research. The overall aim of the LUM project was to make it possible to have a measuring system and a monitoring system installed on a single wind turbine in a wind park. Via a mathematical model it should then be possible to transfer the results from these systems to the other wind turbines in the wind park. The advantage with this approach would be that the expensive measuring and monitoring equipment would only have to be installed in one single turbine per wind park. This thesis has focused on developing the lifetime monitoring system. Future work should be invested in finding a suitable mathematical model to transfer results achieved from one wind turbine to another.

[1] LUM: Lead Unit Monitoring

1.2 Aim

The aim of this thesis was to develop a cost efficient and easy-to-integrate lifetime monitoring system for wind turbines. This task can be subdivided into the following subtasks:

- Monitoring the loads that the main components of a GE 1.5 s/sl wind turbine is exposed to
- Implementing a suitable method to extract the load-cycles, from the monitored loads, online
- Implementing a function that calculates the damage due to the fatigue, that the main components has suffered from, with the help of the result from the cycle counting method
- Visualizing the measured damage against the designed damage for each component
- Performing simulations, with a HITL real-time simulation system, in order to validate the functionality of the developed condition monitoring system
- Installing the prototype-system in a real wind turbine. Performing test measurements and evaluate the results

1.3 Outline

This thesis consists of three main parts: Methods, Results and Conclusions. The chapters of the main part, Methods, are briefly described as follows:

Chapter 2 Firstly, this chapter gives a short overview of different cycle counting methods. The rainflow counting algorithm, the cycle counting method used in this thesis, and its associated preprocessing functions are here thoroughly explained.

Chapter 3 In this chapter, the approach for calculating the fatigue damage for the monitored components, by using the result from the rainflow counting method, is described.

Chapter 4 Here, the target wind turbine is described in means of specifications, physical structure, and sensors.

Chapter 5 In chapter five, the hardware setup of the developed monitoring system is explained. The simulation environment as well as the prototype setup are here considered.

Chapter 6 The structure and functionality of the designed and implemented software is explained in this chapter.

Part I

Methods

Chapter 2

Cycle Counting

2.1 Approach Analysis

There are several methods available for cycle counting, for example level-crossing counting, peak counting, simple-range counting and rainflow counting.

All cycle counting methods reduce the amount of data coming from the load sequences. By this reduction, the information about the order of the cycles and the frequency content, gets lost [WeZe88]. Since fatigue is rate independent, see Section 3.1, this is of less interest for fatigue-analysis. By studying how the fatigue is calculated, see (3.4), it can also easily be found that the order of the load cycles is of no relevance for the fatigue. These assumptions about the fatigue are valid for most, but not all, materials.

According to the knowledge of today, the rainflow counting method is the best method to use when acquiring information, from a load sequence, relevant for fatigue-damage calculation [Johb99, WeZe88]. Contrary rainflow counting, most other methods can only start to process when the whole load history is known [DoSo82]. This is in many cases very inconvenient because it makes online calculations impossible. For this thesis, it was a requirement that the monitoring system should perform its calculations online. Rainflow counting is also a two-parametric counting method. It saves range-mean or from-to values for the found load-cycles (see Section 2.2). Usually, the cycle counting methods are only of one parameter, i.e., saving only the range of the loads.

Because the rainflow counting method fulfills all the prerequisites from the lifetime monitoring system, for a cycle counting method, it was chosen to be used throughout this work. The fact that the rainflow counting method is the state-of-the-art cycle counting method used in the industry further encouraged the use of this method.

2.2 Rainflow Counting

2.2.1 General

The rainflow counting method together with a damage accumulation model can be used to relate a load sequence to the damage it causes to the material. It allows tracking both slow and fast variations of the load by finding cycles that pair maximal loads with minimal loads even if they are separated by intermediate extremum values.

The rainflow method was originally developed by T. Endo, see [Endo74], in the late 1960 as a complicated recursive algorithm. In 1987, the first non-recursive algorithm was presented [Rych87]. A variant of this algorithm will be used throughout this work. In the following section is a description and an illustrative example of the original recursive method. This part can without loss of context be skipped by the reader in order to go directly to the non-recursive algorithm in Section 2.2.3. The recursive algorithm has also been included to assist the reader who wants to learn more about the original rainflow algorithm and how it has developed.

2.2.2 Recursive Algorithm

By turning the time-axis downwards, in the time-stress/strain diagram, the rainflow counting can be thought of as counting raindrops rolling down the roofs of a pagoda (Chinese tower). Each part of the load-curve that lies between two extremum values is a roof. Following rules are given for rainflow counting [NaAm03]:

- On the upper side of each roof a counting process is started
- A counting process is ended when:
    1. the process was started from a minimum and reaches a minimum of the same magnitude or greater
    2. the process was started from a maximum and reaches a maximum of the same magnitude or greater
    3. the process reaches the path of another raindrop
    4. the process reaches the end of the time series

In Fig. 2.1, a load sequence is given. This sequence is to be used as an example for how rainflow counting is performed.

In the example, the first counting-process is initiated at A and goes on the upper side of the curve to the maximum B. After reaching B it continues, without increasing the amplitude, until it meets the path C-D and joins it (rule 3). The second process starts at B, carries on to C, and then ends at time 3 because D has a larger maximum than B (rule 2). The third process starts at C, joins the flow from A-B and then reaches D where it continues parallel with the time-axis. The flow then stops at time 4, since the minimum E is smaller than the minimum A (rule 1). The half-cycle A-D is therefore stored in the residual. From D, the fourth flow is started. It first goes to E, then joins with F-G and subsequently it flows parallel with the time-axis until time 11. Because L has a larger maximum than D, the flow is terminated here (rule 2). The half-cycle D-G is therefore stored in the residual. The fifth flow goes from E via F and ends at

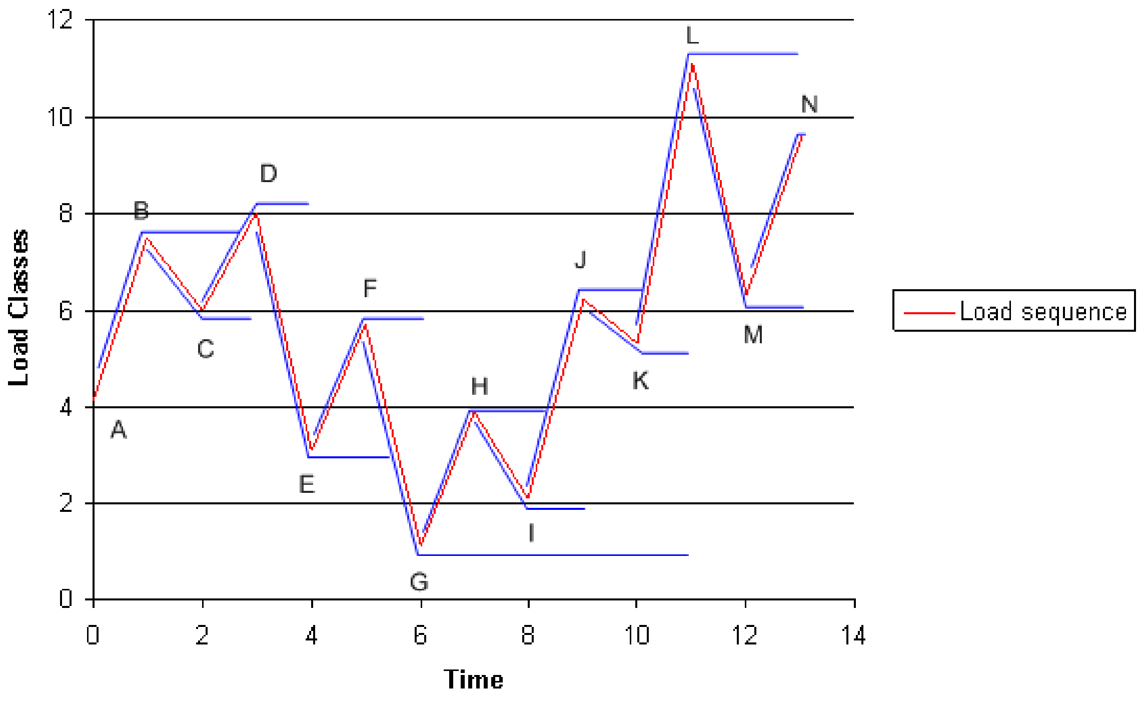


Figure 2.1: Example of how recursive rainflow counting works

time 6 since G has a smaller minimum than E (rule 1). Flow number six begins from F and then unites with the earlier mentioned flow D-E (rule 3). The seventh flow goes from G to H and then connects with I-J (rule 3). Process eight is H-I. It stops at time 9, because J is larger than H (rule 2). Flow nine, I-J, connects with G-H and continues together with flow 10 (rule 3), K-L, until time 13 where the sequence is terminated (rule 4). The half-cycle G-L is therefore stored in the residual. Flow eleven, J-K, cancels at time 11 since the maximum L has a larger maximum than J (rule 2). Also the last two flows, L-M and M-N continue without interruption to the end-time 13 (rule 4) and are therefore also stored in the residual.

The resulting residual is A D G L M N and can be seen in Fig. 2.2. The extracted cycles are B-C, E-F, H-I, J-K.

The rainflow method syntax differs between standing (positive) and hanging (negative) cycles. A standing cycle starts with a minimum and ends with maximum. For a hanging cycle, it is the other way around, i.e., starting with a maximum and ending with a minimum. In the example above, E-F is a standing cycle and B-C, H-I, and J-K are hanging cycles. There is no difference in the treatment of standing and hanging cycles, it is only a notation. Each extracted cycle leads to an increment in the rainflow-matrix.

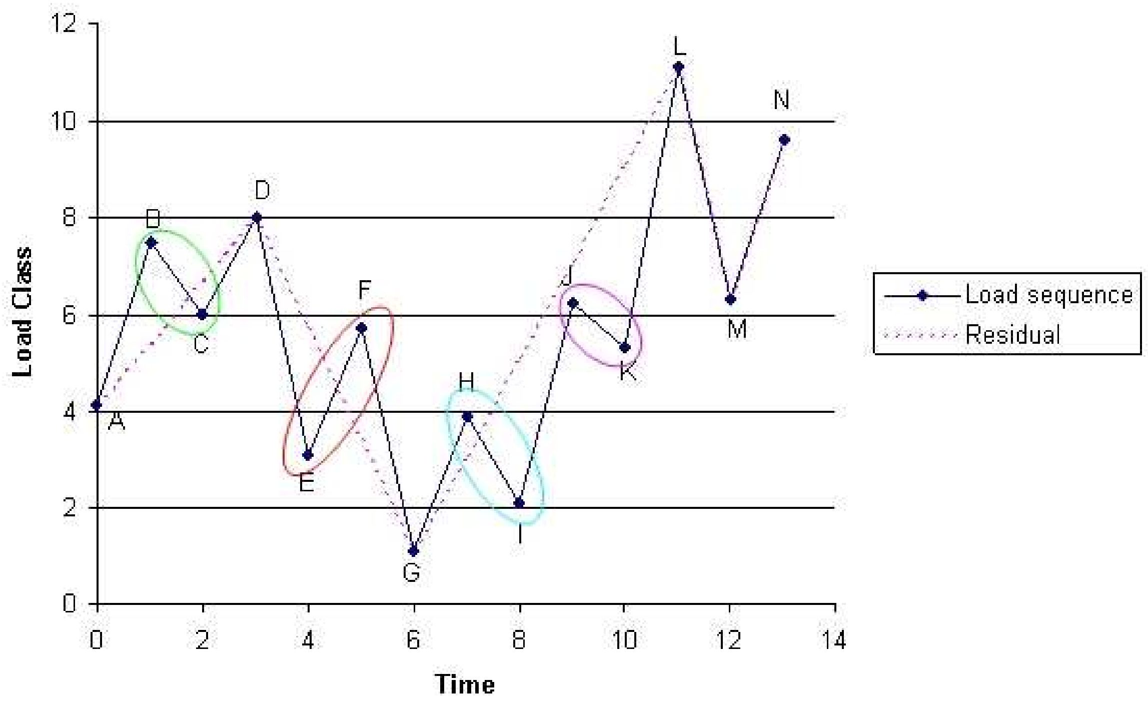


Figure 2.2: Cycles found by the rainflow method (marked with circles) and the remaining residual

The above original recursive description of the rainflow method is quite complex. However, in general it can be seen as simply counting hysteresis cycles for the loads in the stress-strain plane [Johb99]. Fig. 2.3 shows that every cycle from a load sequence leads to a hysteresis cycle in the stress-strain plane. The surrounding cycle is the remaining residual after the rainflow counting. This, of course, does not necessarily have to be closed. That is only the case if the first and last value in the residual are the same.

2.2.3 Non-Recursive Algorithm

The non-recursive rainflow counting algorithm used here comply to the standardization presented in [AGRB94].

The rainflow algorithm takes a sequence of maximum minimum load values, extracted from the complete load sequence as explained in Section 2.2.7, as an input. In order to extract the cycles from the sequence four successive points are needed (S_1, S_2, S_3, and S_4). From these, three consecutive stress ranges are determined: $dS_1 =\mid S_2 - S_1 \mid$, $dS_2 =\mid S_3 - S_2 \mid$, and $dS_3 =\mid S_4 - S_3 \mid$. If $dS_2 \leq dS_1$ and $dS_2 \leq dS_3$ then:

1. cycle $S_2 - S_3$ is extracted and stored in the rainflow matrix,

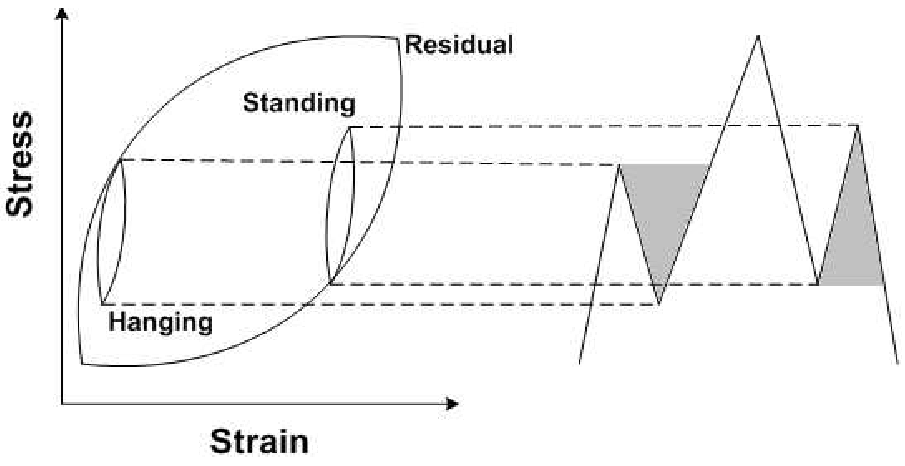


Figure 2.3: Hystereses cycles in the stress-strain plane

2. the two points S_2 and S_3 are discarded from the sequence,

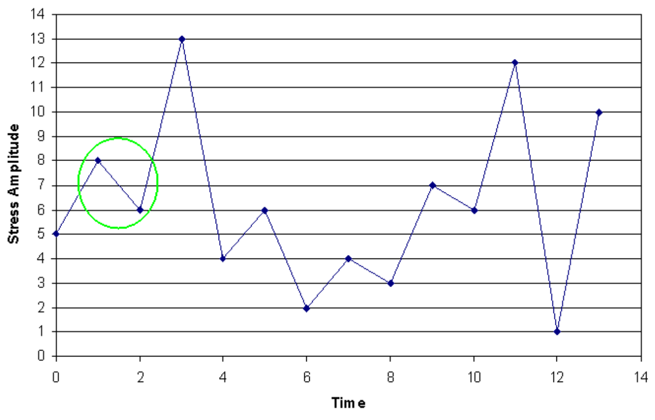
3. the two remaining parts of the sequence are connected to each other.

If this condition is not fulfilled, the next point is considered and the test is repeated, now with the points 2, 3, 4 and a new one. The procedure is continued until there are no new points and no more cycles can be extracted. This method can be performed online or offline. In Fig. 2.4 and 2.5, an example of how the non-recursive rainflow algorithm operates is presented. It demonstrates how load cycles are found and extracted from the data series and it also shows the resulting residual.

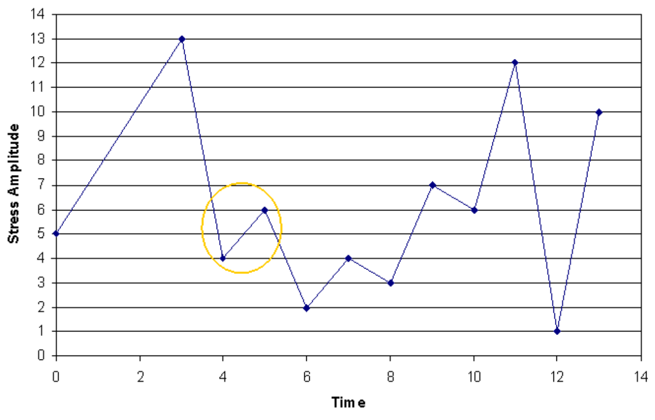
The rainflow algorithm follows the same rules as mentioned in Section 2.2.2, but expressed in a non-recursive way. The description of the recursive rainflow counting method can be seen as a theoretical approach and the algorithm above as a practical. The non-recursive algorithm is visualized in Fig. 6.4.

2.2.4 Rainflow Matrix

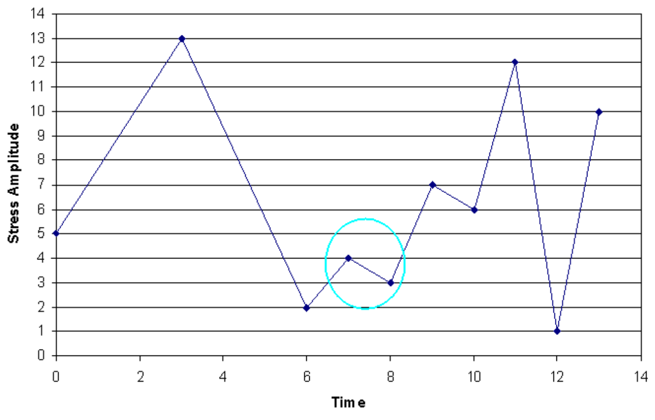
The rainflow-matrix is the result of the rainflow counting method. All extracted cycles are stored in the rainflow-matrix. The matrix can be defined in several different ways, depending on how one wants to use the matrix. Fig. 2.6 shows two different rainflow matrices that corresponds to the cycles extracted in the example of the non-recursive rainflow counter in Fig. 2.4 and 2.5. One possibility is to define the rainflow matrix as being from class x to class y, like the example matrix to the right in Fig. 2.6, i.e., the cycle starts in class x and ends in class y. Another definition states that the axes of the matrix are equal to the range of the cycle (maximum - minimum) and its mean ($| maximum + minimum \mid /2$) shown as the second example in Fig. 2.6. Either way, the matrix contains the same information. A range-mean matrix can therefore be



(a) First cycle to be extracted

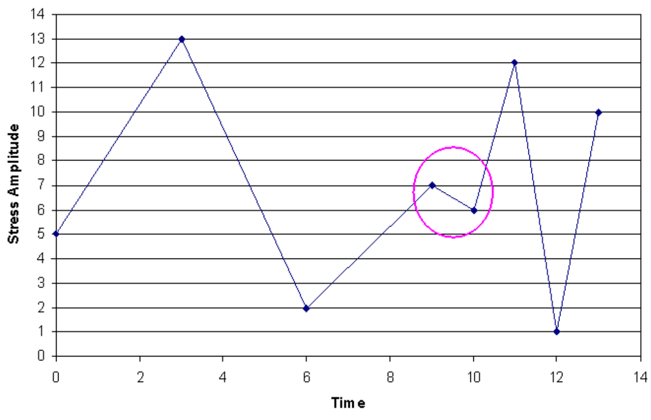


(b) Second cycle to be extracted

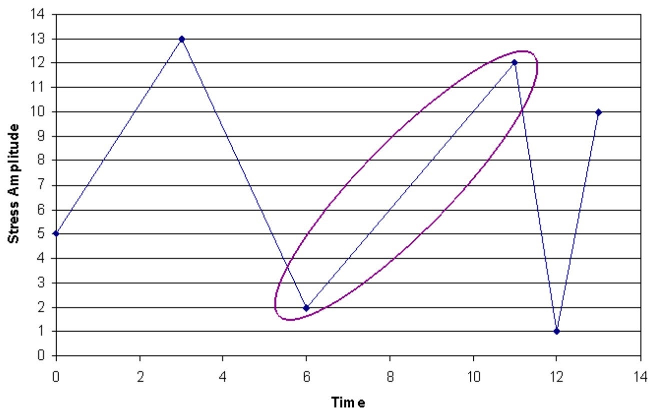


(c) Third cycle to be extracted

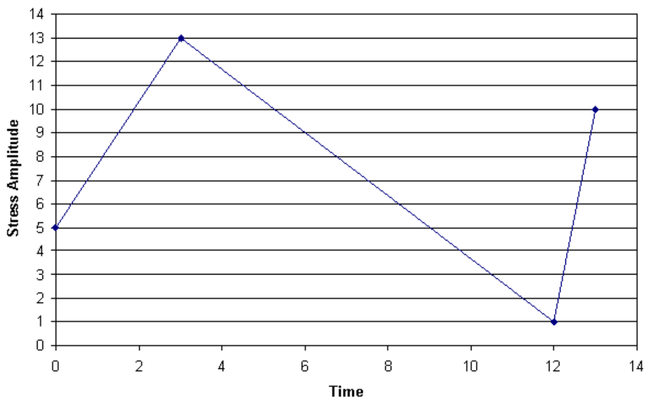
Figure 2.4: Rainflow counting example, steps 1 - 3



(a) Fourth cycle to be extracted



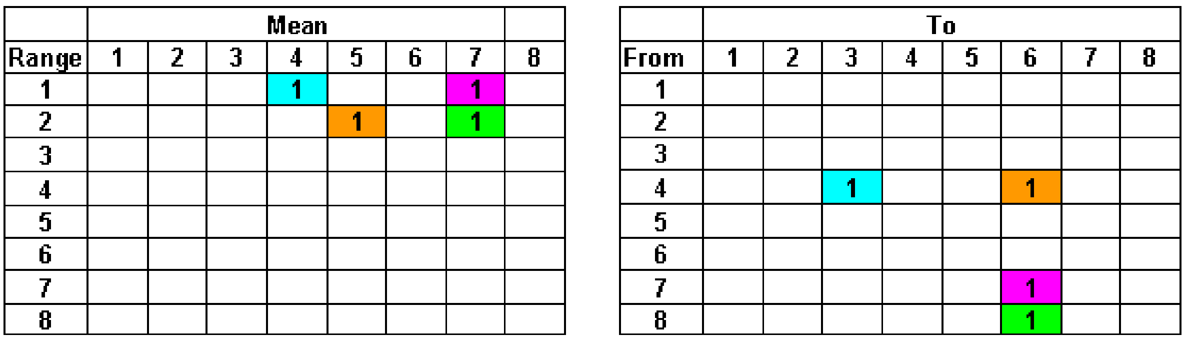
(b) Fifth cycle to be extracted



(c) Resulting residual

Figure 2.5: Rainflow counting example, steps 4 - 6

transformed in a from-to matrix and vice versa. For every cycle, with a certain range-mean or from-to, the corresponding cell in the matrix is increased by 1 [AGRB94].



	Mean							
Range	1	2	3	4	5	6	7	8
1				1			1	
2					1		1	
3								
4								
5								
6								
7								
8								

	To							
From	1	2	3	4	5	6	7	8
1								
2								
3								
4			1			1		
5								
6								
7						1		
8						1		

Figure 2.6: Rainflow matrix defined in two ways

In the examples in Fig. 2.6, only four different cycles are represented. As more cycles, with the same properties, are found, the corresponding cells in the rainflow matrix are increased. Hence, the cells in the rainflow matrix may of course also have discrete values other than 1. When performing rainflow counting on a long load time series, the rainflow-matrix starts to show symmetric characteristics. For example, if one uses the range-mean rainflow matrix, the matrix will probably be fairly symmetric around a certain mean value since most components have a natural state around which most oscillations occur.

2.2.5 Residual

The rainflow counting has been completed for a whole data series when there is no more cycles to extract. The remaining maxima and minima, turning points, are then already stored in the residual. That is because the residual works as a buffer (see Fig. 6.4) where turning points are stored by the rainflow counter as long as they have not yet been paired up with another turning point and stored in the rainflow matrix. Except from the very beginning, the residual will always contain at least two data points. The last turning point in the residual will vary during the rainflow counting process but at the end of each sequence it will always be the last turning point from the sequence. The first turning point in the residual will always be the very first turning point from the first time frame. This special property for the first and last data points in the residual depends on the structure of the rainflow counting algorithm. As mentioned in Section 2.2.3 the rainflow counting algorithm needs at minimum four data points to search for a cycle. That is why the very first and last turning points can never be extracted as a part of a cycle.

When several succeeding data series are to be processed in the rainflow counter, the residual from the previous data series continues to the next cycle in the rainflow counter together with the new incoming data series. It can then be possible to form cycles from data points from different sequences, enabling detection of low-frequency load cycles that occur over a time frame limit.

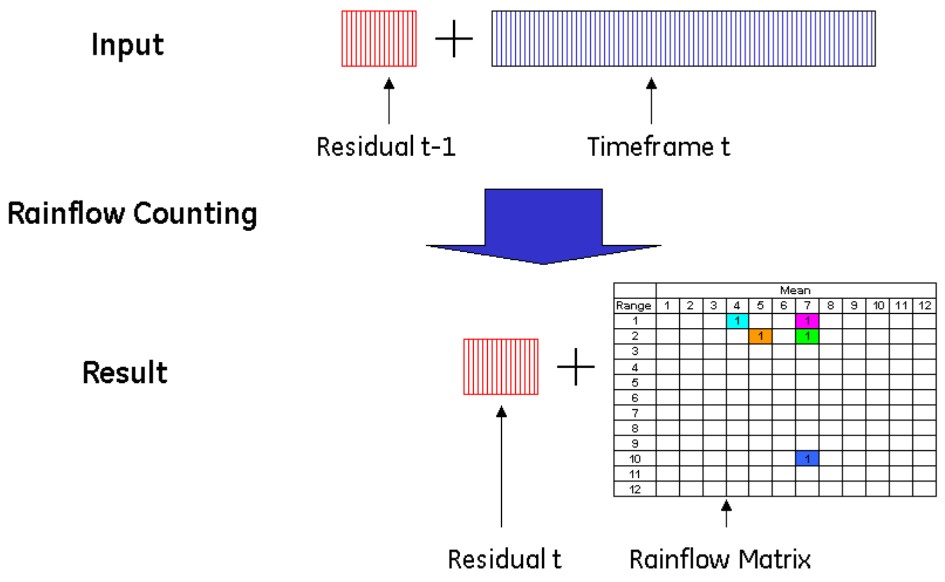


Figure 2.7: Residual's part in rainflow counting when several succeeding data series are to be taken into account

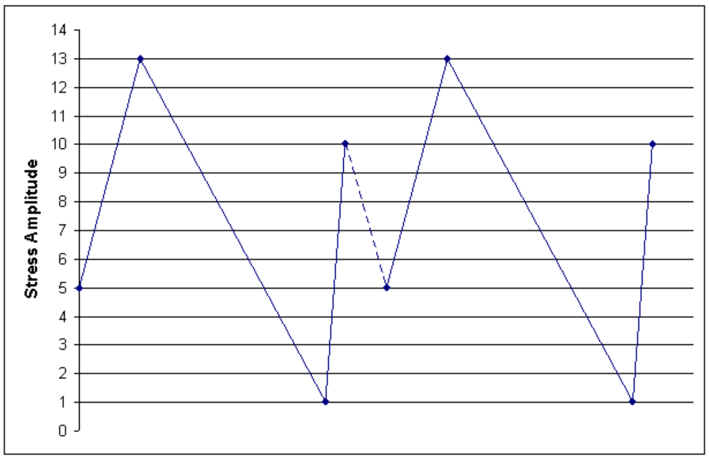
If an interruption in, or between, a time frame occurs, a special treatment of the residual is required since the time series then contain a discontinuity. In this thesis, the half-cycles that are still present in the residual when an interruption occurs are counted as full cycles. This gives a conservative treatment of the load-sequence, i.e. a worst case scenario of the amount of damage that these loads could have caused.

To extract the half-cycles from the residual as complete cycles, the residual is simply added to itself and then processed by the normal rainflow counting algorithm, as seen in (2.1) [AGRB94]. Therefore, no changes have to be made to the rainflow algorithm in order for it to process the residual, which is a big advantage of this procedure.

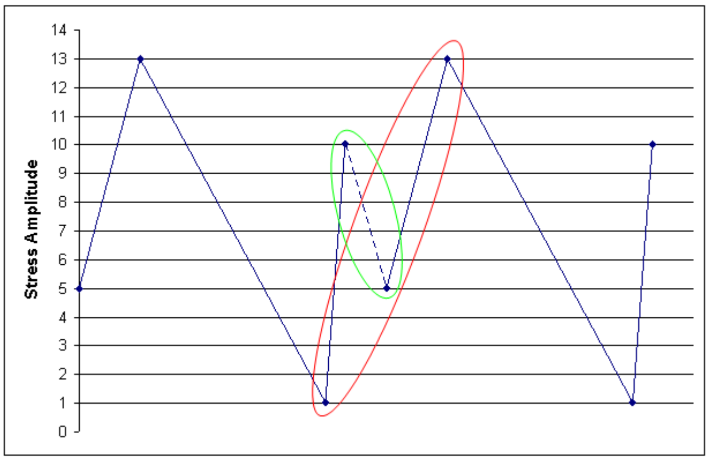
$$[residual] + [residual] \xrightarrow{rfc} [residual] + \{cycles\} \tag{2.1}$$

When joining the residual with a copy of itself, special care has to be taken if the number of elements in the residual is odd. In this case, the last element in the original residual or the first element in the copy have to be removed in order to avoid those two maxima or minima ending up as neighboring elements. Such an occurrence would corrupt the result of the rainflow counting algorithm. The rainflow algorithm only allows an input sequence of alternating maxima and minima.

In Fig. 2.8, an illustrated example is given, describing how the residual is processed. Since this residual, which originates from the example in Fig. 2.5, has an even number



(a) A copy of the residual is concatenated with the residual itself



(b) The half cycles from the residual and its copy have founded full cycles which can be extracted by the rainflow counting algorithm

Figure 2.8: Rainflow counting of the residual

of elements, no extra work has to be performed on the last element in the first residual. After concatenating this residual and its copy, the rainflow counter processes them. In this example, two cycles are extracted. Once again, the residual itself ends up as the remainder.

2.2.6 Processing Succeeding Sequences

If rainflow counting is to be performed on a signal that consists of several succeeding load-sequences, the result formed by summation of the results from the individual sequences does not, in general, correspond to the result generated when all sequences are considered together at the same time [AGRB94]. A cycle could for example originate from an extremum in one sequence and another extremum from a second sequence. In order to have the rainflow counting result for several individual sequences to comply with the global result, the following steps have to be added to the rainflow algorithm [AGRB94]:

1. Perform rainflow counting for each sequence and let the results be represented by the tuple (cycles, residual)
2. Create a sequence by merging the residuals of each sequence, in the same order as they occur in the global signal
3. Take the resulting sequence and split it up in cycles, according to the rainflow method
4. The global result of the rainflow counting is then reached by adding the extracted cycles from step 3 with the cycles from the individual sequences in step 1

The above steps are for rainflow counting that is performed offline. For online processing the sequences and residuals are not calculated separately. Instead, the residual from the rainflow counting for the previous sequence is concatenated with the present sequence and then put into the rainflow counter, as in Fig. 2.7. When concatenating the residual with the sequence, special care has to be taken to the last value in the residual and the first value in the sequence. Since it is not for certain that they are real turning points[1] an extra evaluation has to be performed for these two data points. Here, it is determined if these two data points are turning points or not. If they are not turning points they are removed from the sequences.

2.2.7 Preprocessing Functions

Classification

As seen in Fig. 2.1 and 2.2, the load-axis has been divided into classes. This discretization is done to suppress noise, reduce execution complexity and to give the rainflow matrix a more uniform appearance, independent of which sensors that are used. The classification is performed by calculating the range between the global maximum and minimum value in the time series, splitting this range in classes and then giving each data point a class-number instead of its real value. The number of classes used depends on the desired resolution, the available calculation performance and/or the amount of

[1] See the Maxima and Minima Filter in Section 2.2.7

available memory. If a larger number of classes can be used, a higher accuracy can be achieved. In the work presented here, 164 classes were used since this has shown to be good practice by the tests performed by GE Energy. The limitation of the number of classes was due to finite memory size on the PLC[2] where the program was executed. The rainflow matrix is very memory demanding since the memory on the PLC could only be statically allocated and every cell in the matrix has to be able to store an integer value that ranges up to about 10^8, assuming the worst case number of cycles. If the PLC would have had dynamic memory it would have been possible to change the declared variable size during execution, making the memory handling much more efficient and flexible.

Since the rainflow counter implemented in this thesis is to be used online, the definitive maximum and minimum value from the time series are not known in advance[3]. This is a considerable challenge since the maximum and minimum have to be known in order to make an optimized layout of the classes. To solve this, one can use experimentally found extremum values instead of the real maximum and minimum. These extremum values can be achieved by letting the wind-turbine run in normal mode and then force an emergency stop. An emergency stop exposes the wind turbine to extreme loads. It is not likely that the wind turbine will suffer from larger loads than those during emergency stop. A disadvantage of this method would be that the classes are spread over a value-space probably much larger than needed, since such extreme loads are very rare. This results in a bad resolution and makes the method less accurate. Another approach would be to ignore the extreme loads, or at least to classify them in the highest class possible, in order to increase the accuracy for the smaller and more frequent loads. However, according to [Joha99] the largest loads, even though the number of them is marginal, constitute a majority of the total damage and therefore these loads can not be neglected.

The chosen solution was to use two rainflow matrices instead of one. One of these rainflow matrices was defined to be used for a limited value-space, offering a high resolution in this area. All normal loads will be represented in this matrix. Every cell in this matrix is defined to be able to store an integer value with 32 bit resolution. Also the number of the most common cycles should therefore fit in this matrix, considering an operational lifetime of the system of 20 years. For derivation of the worst case number of cycles see Section 3.2. Extreme loads that do not fit in the value-space covered by the high resolution rainflow matrix are stored in the low resolution rainflow matrix. This matrix covers a very large value-space, however it has a low resolution since it also uses the same number of classes as the first matrix. The loss of accuracy in the low resolution matrix is of less importance since there are only a handful of cycles that will be stored there. Therefore, it is only of significance to store the approximate range and mean of these load cycles. Each cell in the low resolution matrix is only defined to store an

[2] PLC: Programmable Logic Controller

[3] For an offline rainflow-counter the definitive maximum and minimum value from the time series are known

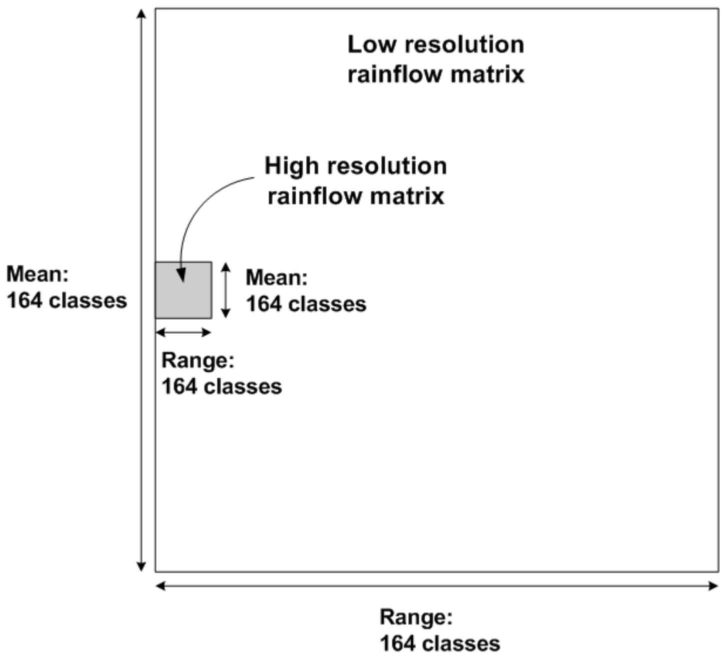


Figure 2.9: Description of a high and low resolution rainflow matrix

integer value with 16 bit resolution, which is expected to be more than enough. The low resolution matrix covers a value space ten times as large as the one covered by the high resolution matrix. For simplicity reasons, the low resolution matrix also covers the value-space of the high resolution matrix, even though this part of the matrix will always be empty since it is already represented. The real value-space that the low resolution matrix covers is for the mean-axis symmetrically located around the high resolution matrix and for the range-axis it starts right after the end of the high resolution matrix's value-space. The proportions between the high- and low-resolution rainflow matrices can be seen in Fig. 2.9. If the high resolution would be used over the entire value-space it would result in a data amount of about 11MB per rainflow matrix. With the suggested solution presented here, only 160KB of memory is needed, i.e. about 1.5% of 11MB. For every monitored sensor, the monitoring system needs a separate rainflow matrix. This means that the amount of needed memory for the rainflow matrices would increase fast when the number of monitored sensors increase. Therefore, it was important to keep the representation of the rainflow matrix as memory efficient as possible.

When performing classifications, every turning point is discretized to one of the classes $u_1 < u_2 < ... < u_n$. That means that the turning points can only be associated with n different values. Two ways of performing the classification are presented here: [Johb99].

1. Each turning point is classified to the closest class, i.e.

$$x_k^d = u_i \qquad \text{if} \qquad u_i - \frac{u_i - u_{i-1}}{2} \leq x_k < u_i + \frac{u_{i+1} - u_i}{2} \tag{2.2}$$

(where $u_0 = -\infty$ and $u_{n+1} = \infty$, i.e. the outermost borders of the first and the last class are open). The classification may cause successive values to take the same discrete class. This is dealt with by the maxima and minima filter.

2. A minimum is classified to the nearest lower value from the classes $u_1 < ... < u_{n-1}$ and a maximum to the nearest higher value from the classes $u_2 < ... < u_n$.

The advantage with the first method is that the classification errors are smaller than in the second method. This leads to smaller errors when calculating the damage. The second method, on the other hand, gives a conservative classification but with less accuracy. The reason why the second method is conservative is that it always represents a turning point that lies between two class levels as the largest one when the turning point is a maximum and as the smallest one when the turning point is a minimum [Johb99]. The damage calculations then gives a larger or equal damage result compared to if the first classification method is used. From a safety aspect, it is of course essential that the life meter always shows at least as much damage as the component has actually suffered and it must for no cases show less. For theses reasons, the second method was chosen to be used throughout this work.

Within a class, all noise will be suppressed because every value within the class gets classified as the same class. Noise occurring on a limit between two classes would on the other hand cause oscillations between those classes. In order to suppress this noise, a reset level, also called hysteresis, is introduced. With the reset level, the rainflow algorithm will only count a cycle if it is larger than the reset level. In this thesis the reset level was set to the size of one class, i.e. only cycles with a range larger than one class are counted. The reset level has been chosen to one class since practical experience has shown that this gives good noise filtering without loss of important information.

Maxima and Minima Filter

As mentioned in Section 2.2.3, the rainflow counting algorithm takes only the extremum values of the entire load sequence as an input. Therefore these have to be extracted from the rest of the data points. This is done by the max/min filter. This filter only extracts all real extremum values, i.e. no saddle-points are considered.

The first data point, in the series of turning points, will not be a turning point, but the first data point from the first input sequence. This is because it can not be determined if the first data point is a maximum or minimum since there are no previous values to compare it with. Therefore, the very first and last data point from the sequences will always be included to the series of turning points in order to be on the safe side, i.e. in order not to have any turning points neglected.

2.3 Conclusions

The rainflow counting method was chosen for extraction of load cycles from the load sequences. Best practice has shown to use the rainflow counting method when acquiring relevant information from a load sequence in order to monitor the lifetime of a component. The rainflow counting method can also be used for online processing, which was a requirement for this thesis. A non-recursive version of the rainflow counting algorithm was selected for implementation.

In case of a discontinuity in an incoming time frame, the residual has to be processed, by the rainflow counter, and be reset in order to keep the overall result valid. With this feature implemented, the system becomes very robust to handle outer disturbances to the sensors and/or the data acquisition system.

The classification of the data points in the time frames were performed in a conservative manner. The conservative classification method was chosen before the least-error method in order to ensure that the damage calculated is always larger than in reality.

To make online rainflow counting possible, a new approach had to be found for the layout of the classes. By having two rainflow matrices, one with high resolution for a small value-space and one with low resolution for a large value-space, all necessary information from the rainflow counting could be stored. This was also a very memory efficient solution, demanding only a fraction of the memory needed for other possible solutions.

Chapter 3

Fatigue Analysis

3.1 General

Fatigue, the loss of strength and other important mechanical properties as a result of cyclic loading over a period of time, is a general phenomenon occurring in most materials [DeJi03]. Usually, components are not destroyed by a single large load, but by the accumulation of many smaller loads [DrHa95].

Generally, fatigue is seen as a rate independent process [Johb99]. This means that the time interval between two loads is of no interest for fatigue, only the dimensions of the loads effects the lifetime. Therefore, the maximum and minimum values of the loads are the most important parameters to study.

In order to illustrate the fatigue properties of a material, a SN-curve (also called a Wöhler curve) is often used. A SN-curve shows the relationship between stress amplitude and cycles to failure. The SN-curve, see example in Fig. 3.1, is defined as:

$$N(S_i) = \begin{cases} \alpha S_i^{-m}, & S_i > S_\infty \\ \infty, & S_i \leq S_\infty \end{cases} \tag{3.1}$$

where N is the number of cycles to failure and S_i is the stress corresponding to load i. α and m are material parameters where α describes the fatigue strength of the material and m is the damage exponent. The higher the stress amplitude, the less number of cycles to failure. For some materials, below a certain stress amplitude, S_∞, the number of cycles to failure approaches infinity. This is called the endurance limit of the material and is a material property. As can be seen in the example in Fig. 3.1, it is state-of-the-art to present the SN-Curve with the load range on the y-axis and the number of applied load cycles on the x-axis.

In Table 3.1, the damage exponent m for different materials is presented. For some materials, after a specific cycle limit another damage exponent with a larger value should be used. This makes the SN-curve for higher number of cycles more horizontal. Hence, even load cycles of a small amplitude will still have some impact, even though it is diminutive, on the lifetime of the component since the number of cycles applied with this amplitude is enormous.

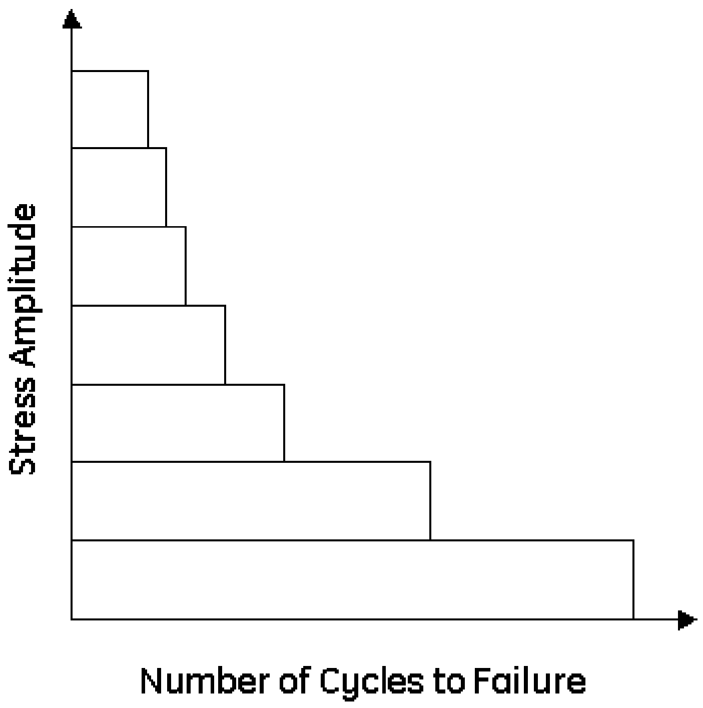


Figure 3.1: Example of a SN-curve

Material	**Used for**	**m-I**	**m-II**	**Inflection point (nbr of cycles)**
Epoxy resin	Blades	10	NA	NA
Crude steel	Tower	3	5	$5 \cdot 10^6$
Steel	Main-shaft	4.9	7.3	$7.7 \cdot 10^5$

Table 3.1: Damage exponent m-I and -II for different materials

Since the design lifetime for a wind turbine is mostly 20 years it has to endure about 10^8 cyclic loads during this time, assuming a 1 Hz oscillation [DeJi03]. This calls for a very robust construction and design of wind turbines, making the study of fatigue processes highly important[1].

3.2 Damage Calculation

Fatigue damage d is defined as the number of applied load cycles n (in this case counted with the rainflow counting method), of a certain amplitude, to the number of cycles to failure N.

[1] Wind turbine components are generally overdimensioned in order to ensure security. A more precise analysis of the actual fatigue on the different components would enable a better and more economical dimensioning of them, thus reducing material costs.

$$d = \frac{n}{N} \tag{3.2}$$

Given that a component is usually exposed to cycles of different amplitudes, the total damage can be defined by Palmgren-Miner's rule [Mine45, Palm24], which can be found in (3.3). Palmgren-Miner's rule summarizes the damage from each cycle for all different amplitudes. The accumulated damage is represented by D, and i is the discrete numbering of all different load amplitudes. The accumulated damage is less then or equal to the value 1. When D is larger than 1 the component has exceeded its expected lifetime and is, at least theoretically, broken.

$$D = \sum \frac{n_i}{N_i} \leq 1 \tag{3.3}$$

Experiments have shown that components can fail, in reality, for D-values as low as 0.2 [Echt96]. This means that Palmgren-Miner's rule has to be used with caution, e.g. by multiplying it with a safety-factor in order to keep the fatigue-damage under the critical limit. Recent research, see [JoSM05], shows that Palmgren-Miner's rule can be made conservative and secure if the SN-curve, from which the values for N_i are derived, is estimated using variable amplitudes instead of constant amplitudes. Estimating the SN-curve with constant amplitudes means that only one load amplitude is tested at a time. The result is the amount of load-cycles, with a certain load amplitude, that the component can survive before it fails. With variable load amplitudes, certain reference load spectrums are used instead of constant load amplitudes. The values in this thesis, that are derived from SN-curves, all come from SN-curves estimated with constant amplitudes.

A comfortable way of defining fatigue damage uses Damage Equivalent Load (DEL) S_{eq} and its corresponding reference number of load ranges, n_{eq}. The new definition of damage is then as follows [JoSM05]

$$D = \sum \frac{n_i}{N_i} = \sum \frac{n_i}{\alpha S_i^{-m}} = \frac{1}{\alpha} \sum n_i S_i^m = \frac{n_{eq} S_{eq}^m}{\alpha} \tag{3.4}$$

where

$$S_{eq} = \left(\sum v_i S_i^m \right)^{1/m} = \left(\frac{\sum n_i S_i^m}{n_{eq}} \right)^{1/m} \tag{3.5}$$

v_i is the relative frequency of occurrence of the load amplitude and n_{eq} is the reference-number of cycles in the time frame. When the cycle-counting result is scaled up to a time-frame of 20 years, n_{eq} is mostly chosen to be $4.74 \cdot 10^8$. This reference number is the number of oscillations resulting when one takes a 1 Hz oscillation, assuming oscillations for 20 years and taking into account an inactivity of 25 %. The inactivity comes from the turbine's *cut-in* and *cut-out* behaviour, i.e., a turbine operates if the wind speed is above the cut-in level and stops if the wind speed exceeds the cut-out level. A challenge with (3.5) is that it can not be determined if m-I or m-II is to be used for the outer exponent $1/m$ since it does not relate to a certain number of cycles, in opposite to the inner exponent that always relates to a certain n_i. The solution chosen was to keep the procedure conservative by using m-I as the outer exponent because this gives larger values for the S_{eq}.

Component	**Position at Component**	**Mechanical Load**	**m**	S_{eq_max} **(kNm)**
Blade	2.75m from hub center	Bending, edgewise	10	1089.6
Blade	2.75m from hub center	Bending, flapwise	10	979.0
Main shaft	Main-shaft flange	Torsion	4	110.7
Main shaft	Main-shaft flange	Torsion	6	192.3
Main shaft	0.1m from flange	bending 90°	4	4197.4
Main shaft	0.1m from flange	bending 90°	6	5212.4
Tower	91.77m height	Torsion	4	562.7
Tower	6.01m height	bending 0° – 180°	4	3357.6
Tower	6.01m height	bending 90° – 270°	4	4199.6

Table 3.2: Designed maximum DEL for different components and m exponents [GEWE05]

Since the designed maximal damage equivalent load, S_{eq_max}, was given for the wind turbine components used for the tests in the presented work, it was desirable to use another approach to represent the fatigue-damage. In equation 3.6, damage D is defined as the relationship between S_{eq} and S_{eq_max}, i.e., measured DEL against a designed maximum DEL. It is trivial that D, also in this definition, will be less then or equal to 1 if the component is not broken.

$$D = \frac{S_{eq}}{S_{eq\_max}} \leq 1 \tag{3.6}$$

When comparing Table 3.1 and Table 3.2, it can be observed that the damage-exponents m, for the main-shaft and tower, are not the same. The reason for this is that GE Energy, who has provided the information for these two tables, only calculates the maximum DEL for different components, positions, and for certain damage-exponents, mostly only for even integer numbers. Since this was the case, the closest lower even integer damage-exponent was chosen for the calculations because it gives a conservative result when calculating the DEL's.

3.3 Conclusions

- The fatigue properties of a component can be illustrated in a SN-curve

- With the help of the Palmgren-Miner's rule, the accumulated damage to a component can be calculated if the number of cycles to failure, for every possible load amplitude, is known.
- The Palmgren-Miner's rule has to be used with caution, for example multiplying it with safety-factors, because components can break for damage values as low as 0.2 instead of the theorethical 1.0
- A comfortable way of representing fatigue-damage is by using DEL
- The total damage to a component is calculated as the measured DEL against the designed maximum DEL

Chapter 4

Wind Turbine Setup

4.1 Structure

The structure of a GE 1.5 s/sl wind turbine hub and nacelle is presented in Fig. 4.1. This was the target wind turbine model for the installation of the prototype monitoring system. The components regarded in this thesis were blades, main-shaft, and tower.

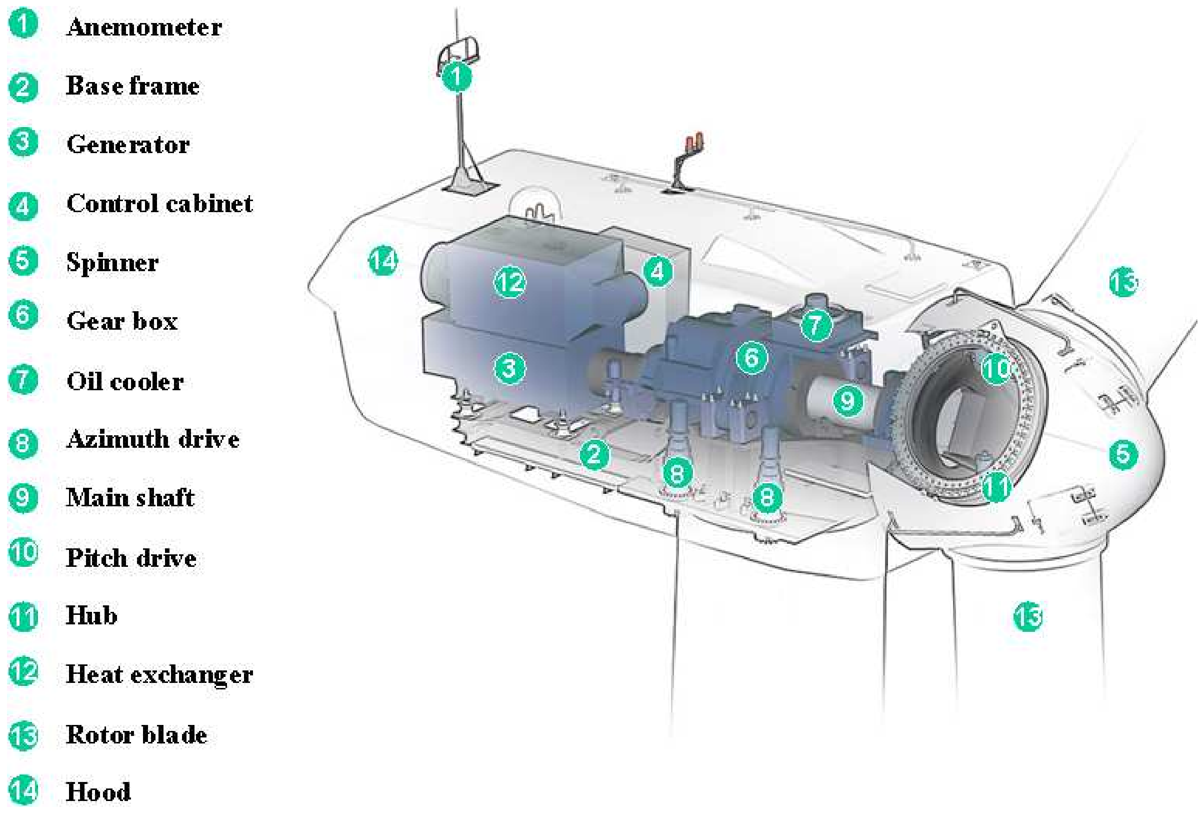


Figure 4.1: Model of a GE 1.5s/sl wind turbine [GEWE04]

Controlled parts in the wind turbine are the pitch angle of the blades, engine speed and yaw angle of the top-box. The pitch angle of the blades determines the attack angle of the blades toward the wind. 0° pitch angle means that the maximal area of the blades is turned against the wind direction. If the wind gets to strong, the turbine increases the pitch angle in order to decrease the attack area of the blades. By doing this, the turbine can operate in higher wind speeds without exceeding the security limits by overloading. The purpose of the yaw angle of the top-box is to keep the top-box positioned toward the wind direction. In case of storm, the yaw angle is changed and the top-box is turned out of the wind direction in order to avoid overloading and possible damages.

4.2 Sensors

Throughout this work, the sensors in Table 4.1 were used. These sensors were available in both the simulation environment and in the real turbine.

Component	**Value**	**Sensor Abbreviation**
blade	bending moment edgewise	blxMe
blade	bending moment flapwise	blxMf
main-shaft	torsion	mshT
main-shaft	bending 90°	mshM90
tower	torsion	toT
tower	bending 0° – 180°	toM0
tower	bending 90° – 270°	toM90

Table 4.1: Listing of the sensors used

Fig. 4.2 shows how the bending moment edgewise, respectively flapwise, at a wind turbine blade can be interpreted. The strain-gauge sensor at the blade is positioned 1.3m from the blade flange, which can also be seen in Fig. 4.3.

At the tower, the strain gauge sensors measuring the bending moments are positioned 5500mm over the fundament with 90° displacement to one another. Hence, the bending moments in 0° - 180° direction[1] and in 90° – 270° direction of the tower are measured. The tower torsion sensor is placed at 9770mm from the top of the upper tower segment. See Fig. 4.5 for a graphical description of the positionings of the sensors on the tower.

All the sensors in Table 4.1 are strain gauge sensors. Strain gauge sensors are used to measure the deformation of an object. A strain gauge sensor consists of a flexible backing with a metallic foil pattern, or a wire, etched upon it. As the object is deformed, the foil pattern is also deformed, causing its electrical resistance to change. This change in resistance, usually measured using a Wheatstone bridge circuit, can be used to calculate the exact amount of deformation with the help of the gauge factor [Efun05]. The gauge factor of a strain gauge relates strain to change in electrical resistance. The gauge factor

[1] 0° is defined to be in the same direction as the entrance door, of the wind turbine, is facing

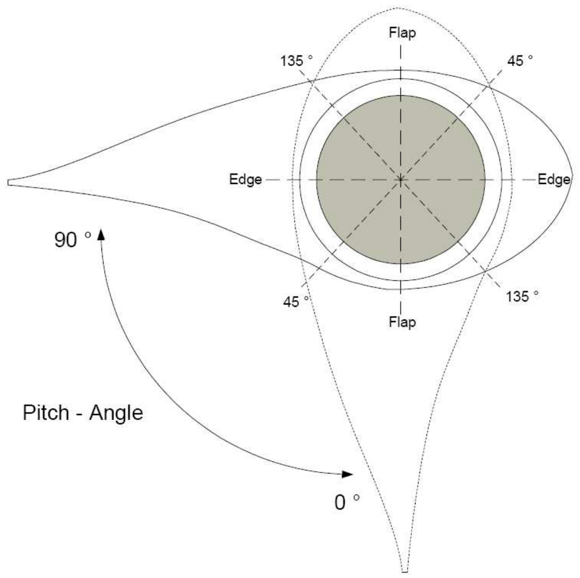


Figure 4.2: Cross section of a wind turbine blade describing how the pitch-angle, the flap direction, and the edge direction should be interpreted

GF is defined by the following equation:

$$GF = \frac{\triangle R/R_G}{\epsilon} \tag{4.1}$$

where R_G is the resistance of the undeformed gauge, $\triangle R$ is the change in resistance caused by strain, and ϵ is the strain. The gauge factor is usually provided by the manufacturer of the strain gauge.

The strain gauges on the turbine blades are calibrated by halting the turbine and putting each blade, one after another, in the vertical position. By knowing the weight of the blade and the exact position of the strain gauge on the blade, the strain gauge can then be calibrated since the load moment occurring at the actual position is known. The calibration is performed manually and only at low wind speeds.

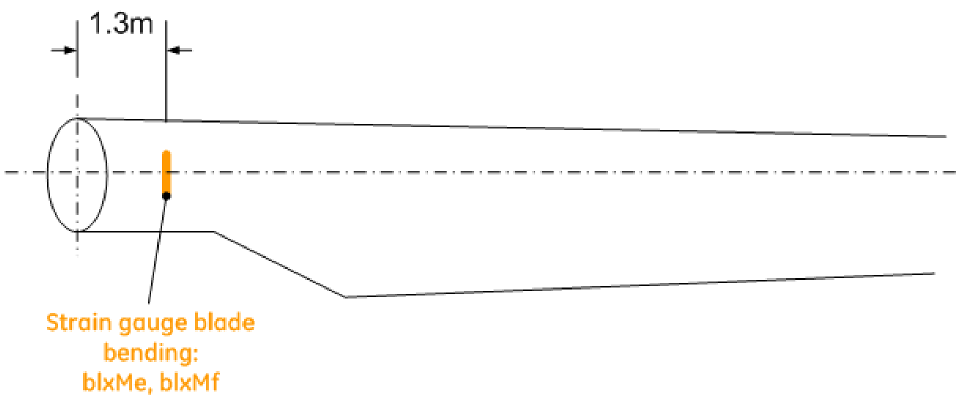


Figure 4.3: Description of the positioning of the strain gauge sensor on the blade [Feic05]

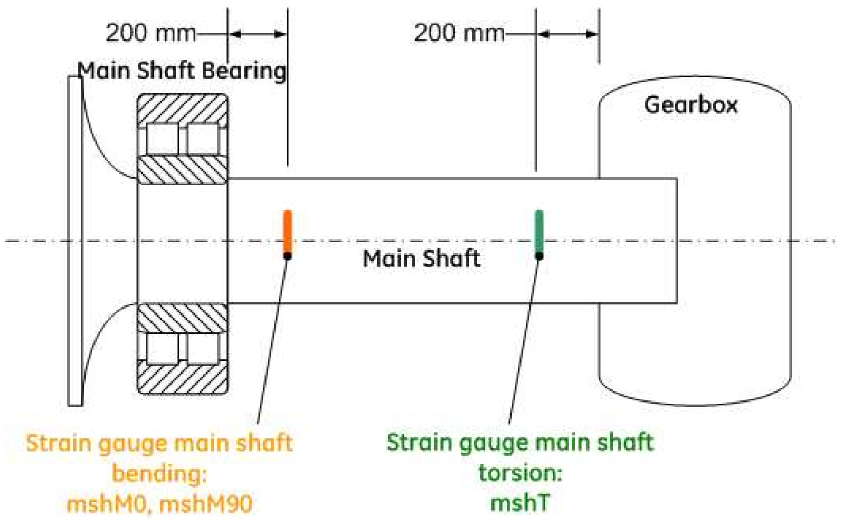

200 mm
200 mm
Main Shaft Bearing
Gearbox
Main Shaft
Strain gauge main shaft bending: mshM0, mshM90
Strain gauge main shaft torsion: mshT

Figure 4.4: Description of the positioning of the strain gauge sensors on the main-shaft [Feic05]

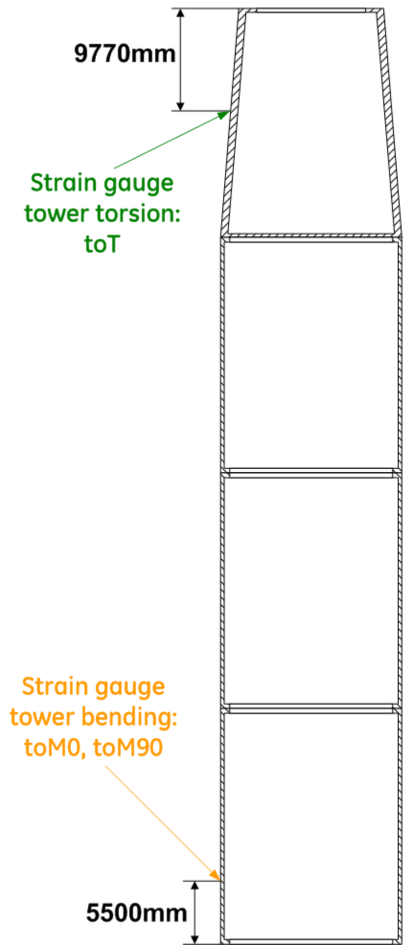

9770mm
Strain gauge tower torsion: toT
Strain gauge tower bending: toM0, toM90
5500mm

Figure 4.5: Description of the positioning of the strain gauge sensors on the tower

Chapter 5

Hardware Setup

5.1 Hardware In The Loop

For the development of the embedded software, a HITL[1] real-time simulation system, see Fig. 5.1, was used. A HITL system simulates a real system, in this case a wind turbine. The complete behavior of a wind turbine, from a controller-unit's point of view, was represented here. There was no difference for the wind turbine controller running on the HITL and running on a real turbine. It was not necessary to perform any changes to the controller when changing between these two working environments.

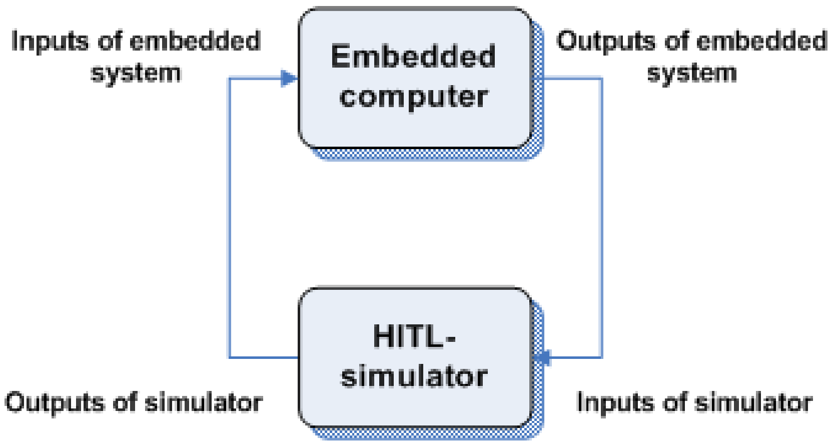


Figure 5.1: Block diagram of an embedded system connected to a HITL-simulator

The HITL simulator was built upon COTS[2] hardware, a real-time operating system (QNX) and a real-time simulation platform (RT-LAB) working with the Matlab toolbox Simulink. See Fig. 5.2 for the entire setup structure. The system consists of a RT-LAB command station, which is often called host PC, a QNX target, a workstation

[1] HITL: Hardware In The Loop
[2] COTS: Commercial-Off-The-Shelf

for SCADA[3], an interface PLC, the tested PLC and various cables for LAN[4], CAN[5], RS422[6], and digital and analog I/O's[7]. The QNX-system runs on an industrial PC[8]. All actual computations and communications takes place in this system. By using the QNX real-time operating system, the real-time performance is ensured. VisuPro is a SCADA system that is monitoring and controlling the Bachmann controller via a LAN. With VisuPro, one can change the parameters of the PLC, start/stop the wind turbine, regulate the pitch angle manually, monitoring the main parameters such as rotor speed, wind speed, pitch angle, voltage etc. VisuPro is used for both simulation purposes and for management of real GE wind turbines. [GEGR04].

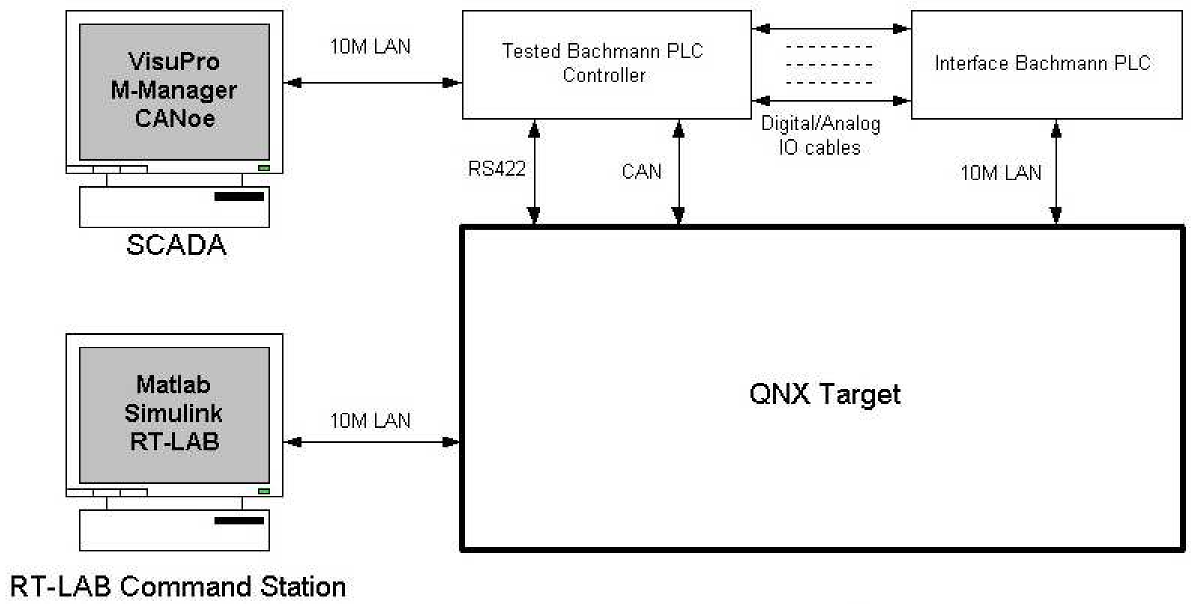


Figure 5.2: HITL system architecture [GEGR04]

Development of the software can be performed with the HITL prior to actual field-testing. This is a large advantage because it would be very expensive and precarious to develop and test new software directly on a real wind turbine. It would be precarious since errors always occur in the first few revisions of new software. When testing the software on the HITL, it does not matter if an error appears because the HITL can easily be stopped and restarted without any damage taking place. If the same error would have occurred on a real wind turbine it could, if it was a serious error, put

[3] SCADA: Supervisory Control And Data Acquisition
[4] LAN: Local Area Network
[5] CAN: Controller Area Network
[6] RS422 is a serial data communication protocol
[7] I/O: Input/Output
[8] An industrial PC is very similar to a normal PC, but more robust and stable. It is eg. more resistant to a harsh environment.

out a security system and push the turbine into an insecure state. Furthermore, more meaningful tests can be performed with a HITL than with a real system. With a HITL one can easily control all different kinds of input, which would not be the case for a true system, thus making it possible to build up an arsenal of different test cases. These can contain extreme conditions, which can be very hard to test in reality because they are too rare and/or too hazardous. The test cases can easily be run through the HITL simulator in order to determine if the developed program operates as expected.

5.2 Development Setup

Execution of the monitoring system was carried out on a Bachmann M1 Controller System. The Bachmann M1 is a modular PLC consisting of one CPU module that can be connected with several analogue or digital I/O modules.

The controller in the development setup, see Fig. 5.3, was an exact copy of the controller that is present in the wind turbine GE 1.5 s/sl from GE Energy. This improved the prospects of success for the later installation in a turbine of the same type. The HITL-system substitutes all real I/O-signals that are normally connected to the controller with simulated I/O-signals. Inputs to the controller are all sensor values. Outputs are the control signals, to all the controlled components in the wind turbine, calculated by the controller. Controlled components in the wind turbine are, as described in Section 4.1, the pitch angle of the blades, the engine speed, and the yaw angle of the top-box. The HITL-system receives the output signals from the controller and includes them in the calculations of the new input signals.

The lifetime monitoring program runs on the LUM module CPU (as seen in Fig. 5.3).

5.3 Prototype Setup

The ambition of the hardware setup was that the new monitoring system should be added to the wind turbine without any changes to the present system. This would be advantageous since the monitoring system could then be added to already existing turbines at rather low costs. Furthermore, performing changes to an existing wind turbine model would force the producer of the turbine to once again carry out security certification of the model. This would also result in large costs. Therefore, all components from the LUM-project were formed as true add-on modules, thus avoiding the necessity of changing the present system setup.

The already existing communication between the hub and the nacelle was carried out over a slip ring that was located around the main-shaft. As can be seen in Fig. 5.4, the LUM-module in the hub was connected to the nacelle module via CAN over WLAN[9]

[9] WLAN: Wireless Local Area Networ

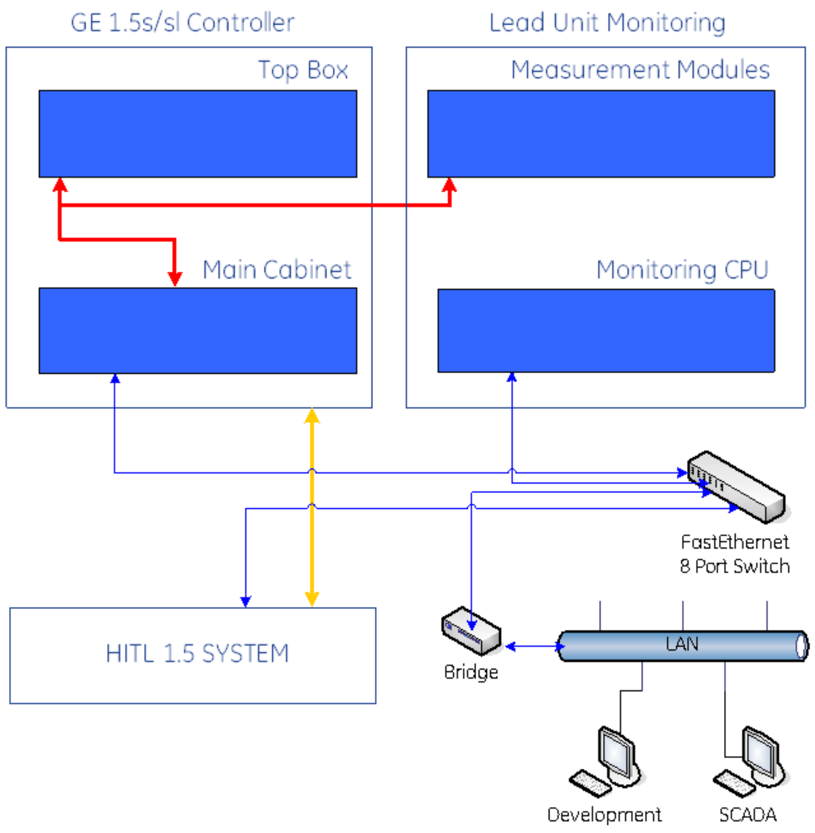


Figure 5.3: System overview of the development setup [Feic05]

with a base frequency of 2.4 GHz (IEEE 802.11 b/g). The reason for using WLAN was that all communication slots at the slip ring, on the target turbine for the prototype installation, were already full. Using WLAN in a wind turbine had only been tested once before by GE Energy[10], although never for a permanent field bus connection. There were doubts regarding installing a WLAN in the hub and nacelle of a wind turbine since the hub has thick cast steel walls that could shield or attenuate the signal. Another reason for the doubts were that there are very strong electromagnetic fields from the generator in the nacelle, that could disturb the signal. However, since the wavelengths for a WLAN with 2.4 GHz base frequency are

$$\lambda = \frac{c}{f} \tag{5.1}$$

$$\lambda_{b/g} = \frac{300 \cdot 10^6 m/s}{2.4GHz} \approx 13cm \tag{5.2}$$

[10] GE Energy has performed a field test with WLAN in a wind turbine. The test was carried out in 2004 in Magdeburg, Germany, in the wind park Wellen II.

the thickness of the material between the two WLAN access points may not exceed 13cm in order not to disturb, or even block, the communication. Using WLAN was found to be a feasible approach worth testing because there are several places at the hub where the thickness is less than 13cm [Feic05]. In practical tests, during the prototype installation, it turned out that the WLAN communication works very well, despite the harsh environment.

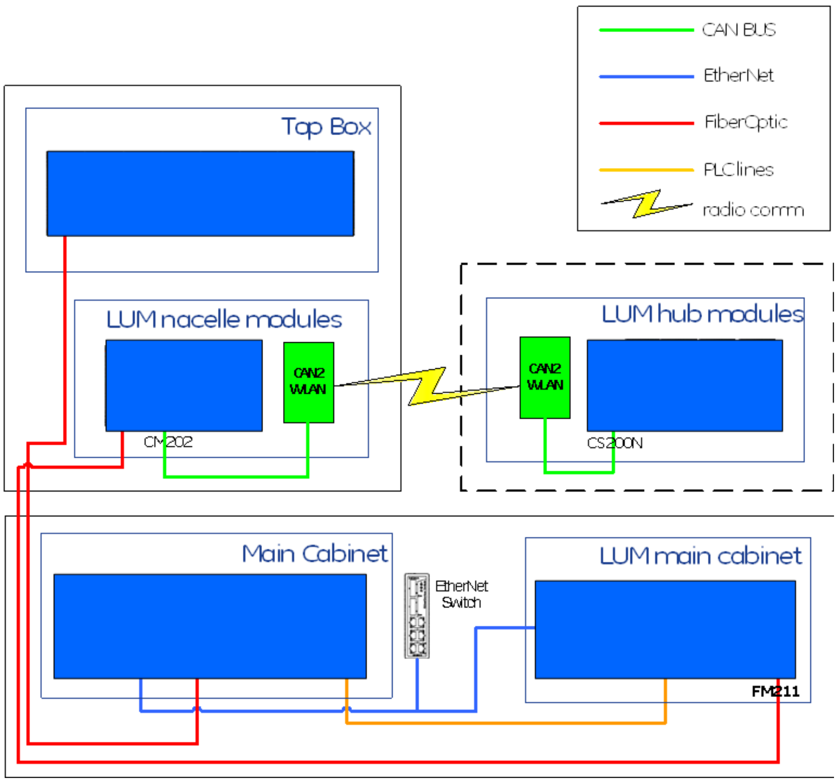


Figure 5.4: System overview of the prototype setup [Feic05]

The LUM nacelle module was connected via a fiber optic cable, installed exclusively for the LUM system, down to the LUM main station[11]. It did not use the already existing fiber optic communication line in order to ensure that the communication from the monitoring system does not disturb the ordinary communication between the top box and the main controller.

The LUM main station was connected to the main controller via an Ethernet network in order to enable the monitoring system to access some data from the main controller. As an extension, this connection is also meant to be used the other way around, i.e. that the main controller can access data from the LUM system in order to be used to optimize the lifetime of the wind turbine.

[11] The LUM main station was installed at the second platform in the wind turbine, about 10 m above the turbine foundation

5.4 Conclusions

Simulations were performed with a HITL real-time simulation system. With the HITL system, all necessary inputs for the wind turbine controller could be simulated. The chosen wind turbine controller was an exact copy of the one present in the GE 1.5 s/sl turbine. The entire software development was carried out with the help of this controller and the HITL simulator.

The monitoring system was constructed completely as an add-on in order to avoid carrying out any changes to the existing setup of the wind turbine. This makes the system more cost efficient and easy to install.

The communication between the modules in the nacelle and the hub, in the prototype system, were provided by a CAN over WLAN connection. This was a new approach that had only been tested once before by GE Energy. However, tests showed that the communication was stable and reliable.

A new fiber optic cable between the nacelle and the main cabinet was also included in the prototype setup. Since the monitoring system only uses its own physical communication connections it could be ensured that the system would not disturb any communication from the current wind turbine controller.

Chapter 6

Developed Software

6.1 Development Environment

The monitoring system was implemented with the Bachmann M-PLC 3 development environment. This enables the programmer an easy usage of the IEC-61131-3 programming languages. The system was implemented using structured text (ST), which is one of the five programming languages defined in the IEC standard for PLC-programming. The other four are instruction list (IL), ladder diagram (LD), function block diagram (FBD), and sequential function chart (SFC).

6.2 Program Organization Units

In M-PLC 3, a Program Organization Unit (POU) can be a function, a function block, or a program.

Functions normally have multiple input and only one output. Functions can only call other functions, no function blocks or programs can be called. There is no own memory reserved for functions.

Function blocks are much like functions but they can have no, or multiple, outputs and can even have combined inputs/outputs. Each function block has its own memory and therefore they have to be instanced. Having its own memory means that a function block keeps the values of its variables, after being processed, for the next process call. Thus, two calls to a function block, with the same input parameters, can generate different output values. Function blocks can call functions and other function blocks. Although, they can not call programs.

Programs are the main program organization unit. They can be globally accessed throughout the whole software project. Therefore, they can not be instanced. Programs can be called with or without parameters. Programs can only be called by other programs. As well as for a function block, a program also have its own memory.

6.3 Design

During this thesis, several design proposals were developed and taken under consideration. Two of the designs were implemented. These two designs are both described in the following two sections.

6.3.1 First Design Approach

As can be seen in Fig. 6.1, the system takes data series as input. The data series can be retrieved from an arbitrary sensor on the wind turbine that measures force or moment. This thesis concentrates on the sensors measuring tower moment and torsion, main shaft moment and torsion, and blade deflection in edgewise and flapwise direction. These components are the most central in the wind turbine construction. The sensors just mentioned were all given on both the HITL-system and on the real wind turbine that was used for testing. Thus, appropriate comparison between simulations and real measurements were possible.

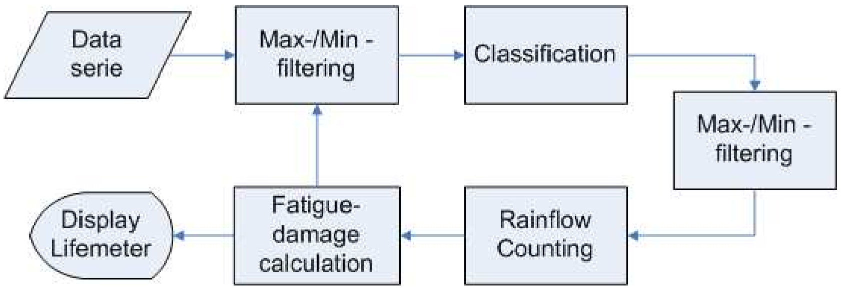


Figure 6.1: Software flowchart for the first design approach

The incoming data series were retrieved from the sensors, with the help of a data acquisition system[1], during a time-frame of 10 minutes. Approximately 27 Hz sampling frequency was used, enabling detection of oscillations up till about 13.5 Hz. Thus, each data series contain 16.384 data samples. The input data series first enters a maxima and minima filter function block. The turning points that are extracted here continues to the classification function block. Here, incoming data gets classified according to the norms for conservative classification presented in Section 2.2.7. The now classified turning points follow to a new maxima and minima filter function block that filter out any succeeding turning points that may have been discretized to the same level by the classification function block. This is necessary in order to preserve that only true maxima and minima data points enters the rainflow counter. The data points that are filtered out in the last filter are considered to be noise, since they are part of cycles occurring within the limits of one class. The reason, why a maxima and minima filter has to be performed

[1] See [Feic05] for more information on the data acquisition system

twice, instead of once after the classification, is that the classification function needs to know if the current turning point is a maximum or a minimum to be able to carry out a conservative classification of it. From the second maxima and minima filter the turning points are sent onward to the rainflow counter. The rainflow counter takes the maxima and minima and combine them into cycles, which are then stored in the rainflow matrix. The fatigue calculation block takes the rainflow matrix as input and uses it to calculate the fatigue. At the end of each processing cycle, the estimated remaining lifetime is displayed in the form of a so called life meter. After this, a whole cycle of the program, for one sensor, has finished. The program returns to its starting state and proceeds with the same process for any other sensor that is to be monitored. If all sensors have been processed, the program goes to its idle state and waits for the next time frame to arrive.

The transfer of large data-blocks, e.g. rainflow matrices and time frames, between the different function blocks gives a high processor load. To avoid this, pointers were used at all places in the system where data-blocks of significant size were needed as input parameters.

6.3.2 Second Design Approach

The second design approach was basically the same as the first one, but with one major difference, namely the classification. In opposite to the first design approach, the classification was included in the rainflow counter function block in the second design approach. Hence, the second maxima and minima filter became unnecessary. The reason why the classification was performed before the rainflow counting in the first approach was that in that way more data points were filtered out before reaching the rainflow counter. This would then lead to less execution complexity. However, by doing the classification before the rainflow counter the result becomes less accurate because the rainflow counting method is carried out with classified values. Since the result from the classification is in any case conservative this would not be a security issue. Although, the lifetime of the monitored components would be decreased slightly faster than in reality. The implication of this would be increased costs. It was also found that in the already available offline rainflow counter included in the commercial signal processing tool FAMOS, used by GE Energy, the classification was performed internally by the rainflow counter. This further encouraged the use of the second design approach since this made it possible to validate the developed system by comparing with results achieved from GE Energy's calculations.

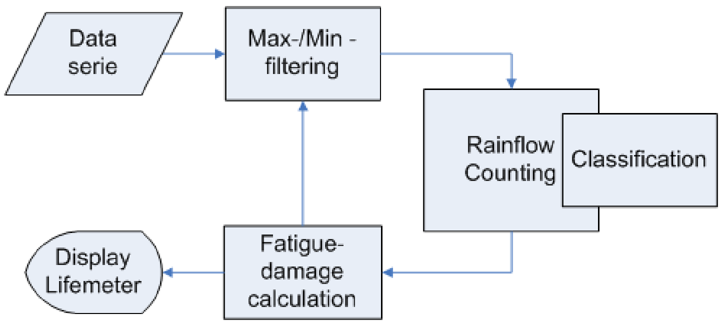


Figure 6.2: Software flowchart for the second design approach

6.4 Developed POU's

6.4.1 Program Lifetime_calc_main

The program lifetime_calc_main is the main program of the developed monitoring system. This is called for every incoming time frame from every sensor. The calls come from the main program of the data acquisition system.

In the lifetime_calc_main program there is a setting that enables the user to choose to use the implementation with the external or internal classification when performing the calculations. By setting the boolean variable *rfc_internal_classification* to TRUE or FALSE, the user can decide which approach to use. Default value of *rfc_internal_classification* is TRUE.

From the lifetime_calc_main program, all calls are made to the different function blocks used in the monitoring system. The lifetime_calc_main program is arranged in such a way that after every finished call to a function block it returns to the calling program. The reason for this is to keep execution cycles short, preventing violation of the real-time properties of the system. This means that before a time frame has been completely processed, several calls to the lifetime_calc_main program has to be made. First when the lifetime_calc_main program's output is set to true is the processing of a time frame completed.

The first task for the main program, for every new incoming time frame, is to control the integrity of the time frame. This is done by checking the time that the time frame contain. The time contained have to be exactly the same as expected, with millisecond accuracy, in order to be accepted. In this system, every time frame is supposed to be 10 minutes long, thus it should contain 600 000 ms. The time frame data structure has also an error flag that can indicate several kinds of errors. This flag is also controlled before letting the time frame pass to the calculation function blocks. In case of a corrupt time frame, the time frame is thrown away and the residual for the corresponding sensor is processed by the rainflow counter, and reset, and the program goes back to its idle state waiting for the next time frame/call.

6.4.2 Function Block Filter_max_min_REAL

The function block filter_max_min_REAL was used in the first implementation approach. In this implementation version, it was the first of two maxima and minima filter function blocks. It takes a time series of floating point values as input and filters it. The outputs from this function block are the number of turning points that the time series contained and an array with the turning points themselves. Here, a turning point means an extremum, maximum or minimum. All possible saddle points are also filtered out by this function block. If several succeeding data points in the time series have the same value and they together constitute an extremum, only one of them is kept. Thus, the output array with turning points only contain alternating maxima and minima. Fig. 6.3 shows the flowchart of the function block filter_max_min_REAL.

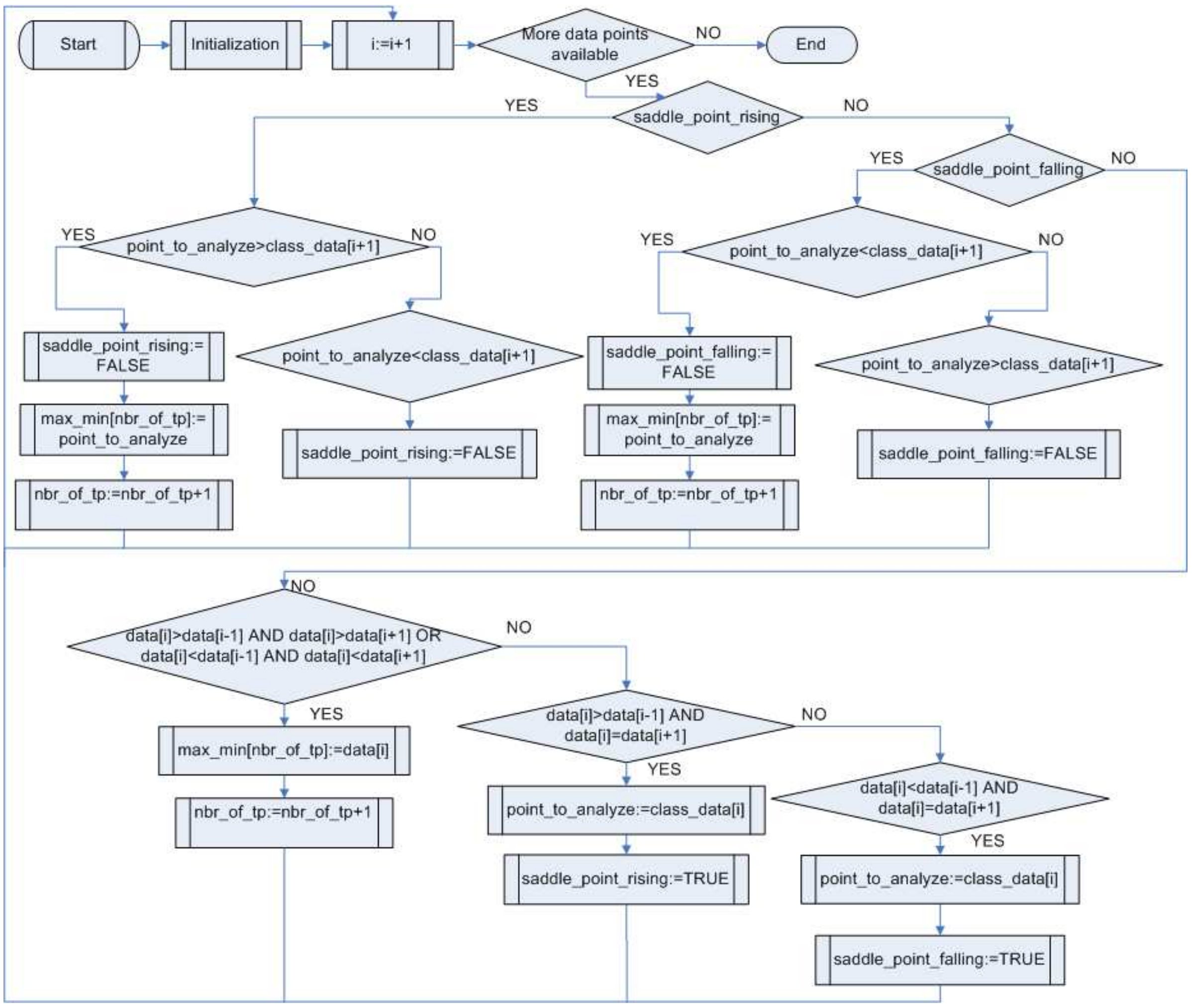

Start
Initialization
i:=i+1
More data points available
NO
End
YES
saddle_point_rising
YES
NO
saddle_point_falling
YES
NO
YES
point_to_analyze>class_data[i+1]
NO
YES
point_to_analyze<class_data[i+1]
NO
saddle_point_rising:= FALSE
point_to_analyze<class_data[i+1]
saddle_point_falling:= FALSE
point_to_analyze>class_data[i+1]
max_min[nbr_of_tp]:= point_to_analyze
saddle_point_rising:=FALSE
max_min[nbr_of_tp]:= point_to_analyze
saddle_point_falling:=FALSE
nbr_of_tp:=nbr_of_tp+1
nbr_of_tp:=nbr_of_tp+1
NO
data[i]>data[i-1] AND data[i]>data[i+1] OR data[i]<data[i-1] AND data[i]<data[i+1]
NO
YES
data[i]>data[i-1] AND data[i]=data[i+1]
NO
max_min[nbr_of_tp]:=data[i]
YES
nbr_of_tp:=nbr_of_tp+1
point_to_analyze:=class_data[i]
data[i]<data[i-1] AND data[i]=data[i+1]
YES
saddle_point_rising:=TRUE
point_to_analyze:=class_data[i]
saddle_point_falling:=TRUE

Figure 6.3: Flowchart for the maxima and minima filter algorithm

6.4.3 Function Block Filter_max_min_INT

The function block filter_max_min_INT was only used for the first implementation approach. It was used as the second maxima and minima filter function block. This function block is very similar to the function block filter_max_min_REAL. Although, it takes an array of integer values as input instead of floating point values. It also controls if the last value in the residual, i.e. the last value from the previous time frame, is really a maximum or minimum by comparing with the first data point in the present time frame. This verification has to be carried out for every time frame, because the last data point from each time frame is automatically included to the residual, to ensure that no possible maximum or minimum are lost. Except for controlling the last data point in the residual, the main functionality of the function block filter_max_min_INT was to remove possible succeeding data points with the same value that could have arisen during the external classification process, present in the first implementation approach.

6.4.4 Function Block Filter_max_min_REAL_v2

By using the function block filter_max_min_REAL_v2, the functionality of the function block filter_max_min_REAL and filter_max_min_INT are combined. It was constructed to filter a time frame of floating point values regarding its turning points, as in filter_max_min_REAL, and also for controlling the last value in the residual, as in filter_max_min_INT. The function block filter_max_min_REAL_v2 was only used in the second system implementation, with the classification performed internally in the rfc_v2 function block. Since the classification is performed internally, the filter functionality does not have to be splitted into two function blocks[2].

6.4.5 Function Block Rfc

The most central function block in this thesis was the rfc. The rfc function block performs rainflow counting on the incoming turning points according to the non-recursive rainflow algorithm presented in Section 2.2.3. The most important outputs from the rfc are the two rainflow matrices, the high resolution rainflow matrix and the low resolution rainflow matrix, and the residual.

6.4.6 Function Block Rfc_v2

An extended version to the function block rfc is the rfc_v2. The difference between rfc and rfc_v2 is that rfc_v2 has the classification functionality integrated in its own function block, according to the second design approach. This makes the results coming from rfc_v2 more accurate than the ones from rfc. Except this, there is no difference between rfc and rfc_v2.

[2] Compare with Fig. 6.1 and Fig. 6.2

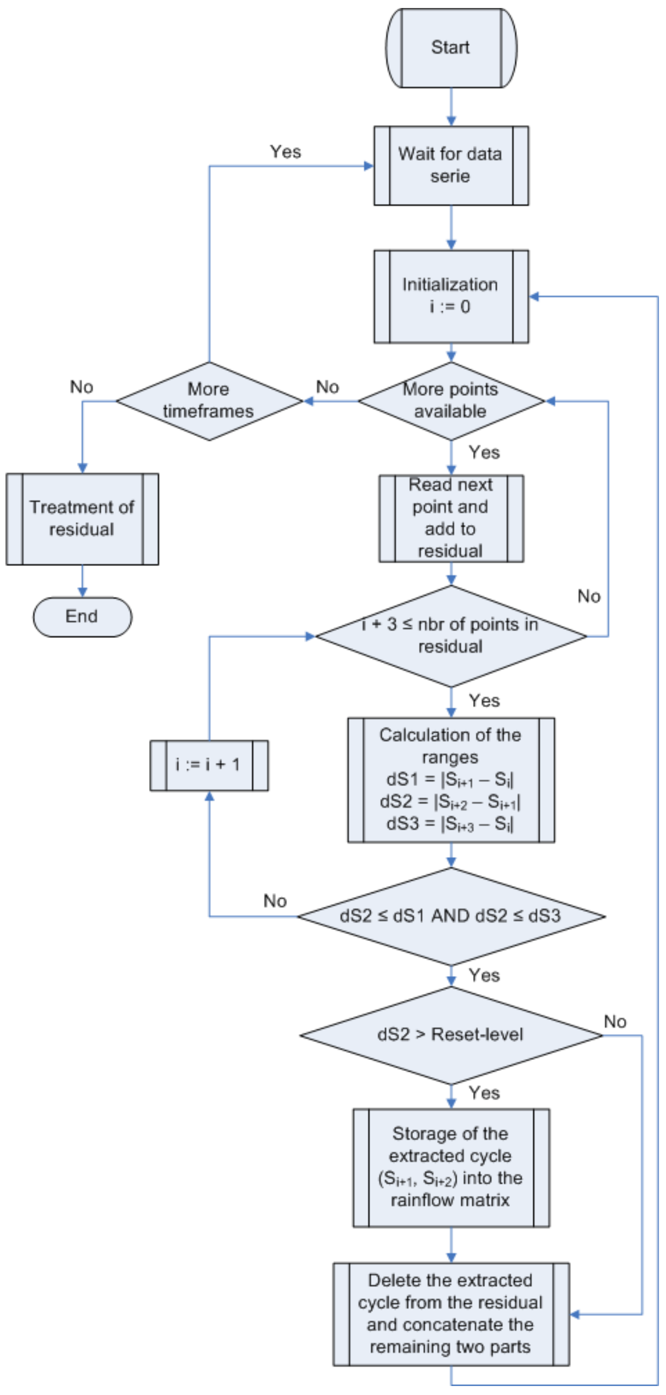

Start
Wait for data serie
Yes
Initialization i := 0
More timeframes
No
More points available
No
Yes
Treatment of residual
End
Read next point and add to residual
No
i + 3 ≤ nbr of points in residual
Yes
i := i + 1
Calculation of the ranges
dS1 = |Si+1 – Si|
dS2 = |Si+2 – Si+1|
dS3 = |Si+3 – Si|
No
dS2 ≤ dS1 AND dS2 ≤ dS3
Yes
dS2 > Reset-level
No
Yes
Storage of the extracted cycle (Si+1, Si+2) into the rainflow matrix
Delete the extracted cycle from the residual and concatenate the remaining two parts

Figure 6.4: Flowchart for the rainflow algorithm

6.4.7 Function Block Damage_calc

With the two rainflow matrices, calculated by rfc or rfc_v2, as inputs, the damage_calc function block calculates the damage that the respective component has suffered from, or to be more exact, the DEL[3]. Other inputs needed for this calculation are the material parameters and designed maximum DEL for the component that the actual time series originate from. The calculations are performed according to (3.5) and (3.6). The first result is obtained by dividing the calculated DEL with the designed maximum DEL. This gives how many percent of the total lifetime of the component that has been exhausted. The second result is received in the same way as the first one, although scaled to 20 years. This means that the second result shows how many percent of the lifetime that would have been exhausted in case that the present amount of damage would be scaled to the designed maximum lifetime of the wind turbine, which is 20 years.

6.4.8 Function Block Matrix_2_file

The function block matrix_2_file was used to store the calculated rainflow matrices onto a flash memory. The flash memory card was integrated in the CPU module on which the monitoring system executes. For the internal continuous fatigue analysis, it was enough to have the matrices saved on the internal PLC memory, but for external evaluation was it necessary to save the matrices on an external memory unit.

In order to make the storage of the matrices as memory efficient as possible, a compression algorithm was developed. The rainflow matrices are usually very sparsely occupied with values. Therefore, instead of saving the whole matrix, only the elements with values were stored.

6.5 Developed GUI's

In order to enable an easy usage of the developed monitoring system for the user, several GUI's[4] were developed. In Fig. 6.5 the GUI presenting the total damage, that has been calculated for each sensor, can be seen. This specific screenshot has been taken during offline development mode, which explains why the damage piles all have the same size. With a similar GUI as in Fig. 6.5, the damage scaled to 20 years can also be observed.

6.6 Matlab Implementation

For validation purposes, all POU's implemented with the Bachmann M-PLC 3 development environment were also translated into Matlab functions, so-called m-files. With

[3] See Section 3.2 for definition of DEL
[4] GUI: Graphical User Interface

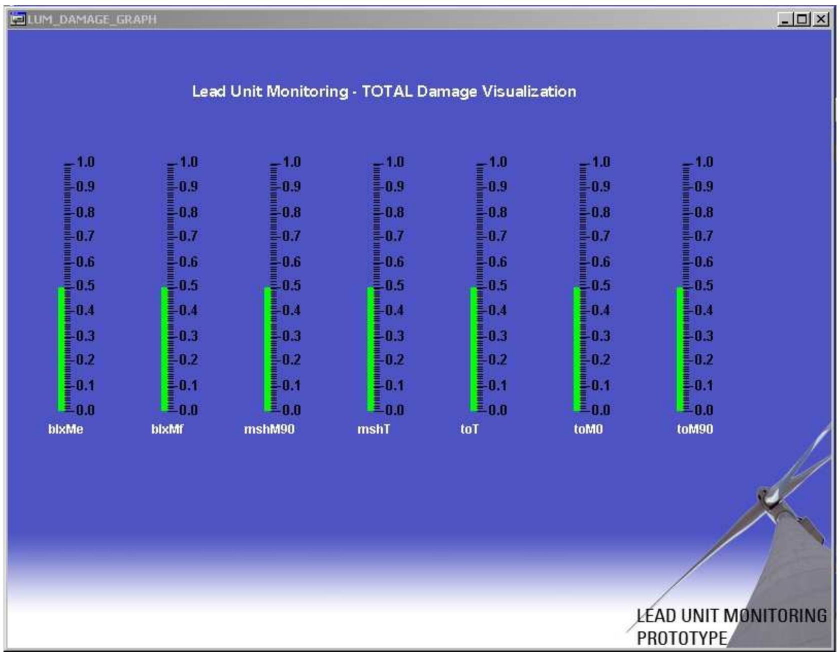


Figure 6.5: GUI showing the total damage that has been calculated for the monitored sensors

these m-files, it was possible to carry out flexible comparisons and validations as well as making graphical illustrations of the results.

6.7 Real-time Requirements

Since a new time frame arrives only every 10 minutes, the program has 10 minutes available to finish its tasks. Of course, these 10 minutes are shared with other processes, for example those performing the calculations for the time frames from the other monitored sensors and the data acqusition system. Therefore, the truly available execution time is shorter. The hardest real-time requirement was that every execution cycle in the program had to be shorter than 18 ms. The data acquisition system, running on the same CPU, samples some sensors with as much as approximately 56 Hz, thus having about 18 ms between each sample. If these 18 ms execution time slot would be exceeded by some part of the program, the integrity of the data acquisition would be violated. Therefore, *watch dogs* that controlled and secured the integrity of the time slots are implemented in the given data acquisition system.

6.8 Conclusions

Two design approaches were implemented with the Bachmann M-PLC 3 development environment. The first implementation was more time-efficient, and also more conservative, than the second one because the classification was performed before the rainflow counting. In the second implementation, the classification of the turning points was carried out internally in the rainflow counting function block. This led to more accurate, although still conservative, results but was also more time consuming. By changing a parameter in the main program, the user can decide which of the two implementations to use. Offline rainflow counting performed by GE Energy, with the commercial software FAMOS, comply to the second implementation.

All POU's implemented were also translated into Matlab functions in order to be used for offline validation purposes.

Part II

Results and Validation

Chapter 7

Simulation using HITL

The simulation was carried out with the HITL real-time simulation system. It had a duration of 40 minutes, thus 4 time frames of 10 minutes each were received, and the specifications of it are stated in Table 7.1. The sensor monitored was the one measuring edgewise bending moment for a wind turbine blade. The four received time frames are plotted in Fig. 7.1.

Wind turbine model	GE 1.5 s/sl
Mean wind speed	10 m/s
Sensor monitored	blxMe
Number of classes	164
Upper limit for the high resolution sensor coverage	+1300 kNm
Lower limit for the high resolution sensor coverage	+600 kNm
Class size	4.2683 kNm
Samples per time frame	16 384
Simulation time	40 Min

Table 7.1: Simulation specifications

The upper and lower limits for the high resolution sensor coverage was found empirically suitable. As can be seen in Fig. 7.1, none of the values from the time frames received during this simulation, reached outside these limits.

In Fig. 7.2 it can be observed how the high resolution rainflow matrix develops during the four succeeding time frames. Here, it can be seen how the rainflow matrix characteristics gets more and more distinct for each processed time frame. These rainflow matrices were calculated online by the developed monitoring system, using the implementation version with the external classification.

In Fig. 7.3, the rainflow matrix for time series number one in Fig. 7.1 can be more thoroughly investigated. This matrix was calculated offline with the help of the Matlab-functions, implemented in this thesis, with internal classification. The Matlab functions were translated from the function blocks developed with the Bachmann M-PLC 3 programming environment. In Fig. 7.4, a rainflow matrix for the same time series, but calculated by GE Energy with the FAMOS signal processing tool, is presented. When

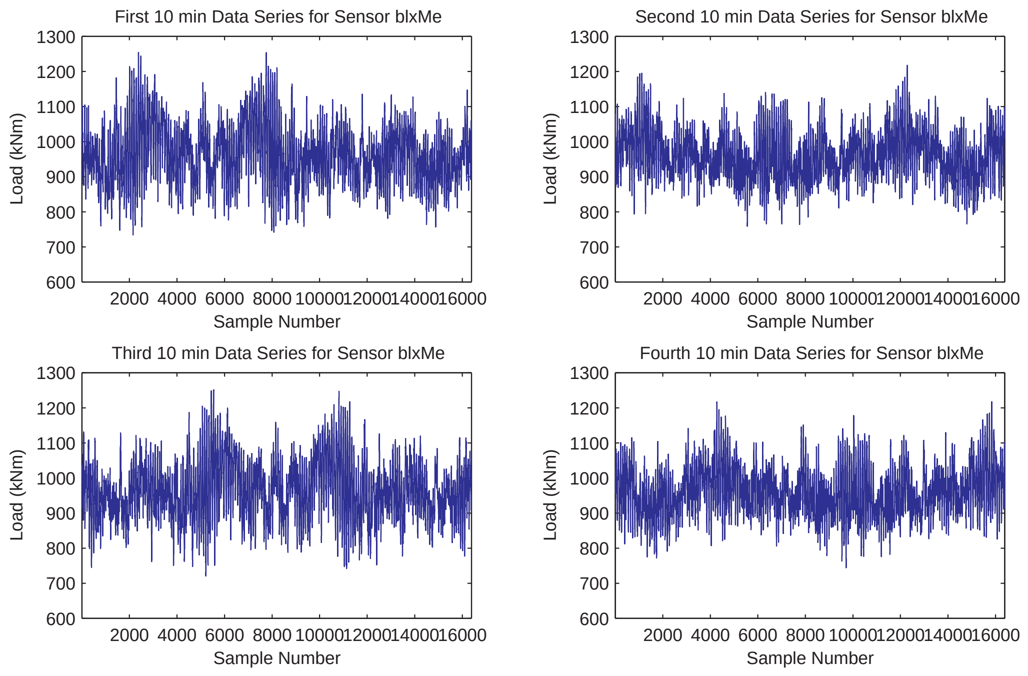


Figure 7.1: The four time frames obtained from the simulation

comparing these two matrices it can be seen that they have large similarities. Nevertheless, they differ slightly from each other, even though they originate from the same time series. The difference is most likely due to the classification that is performed in a somewhat different way. The number of load-cycles found by both calculations were almost the same, the difference was only about 0.3%. The developed system counted 692 cycles and FAMOS counted 690. When comparing the scaled damage[1], calculated with the developed Matlab functions from these two rainflow matrices, the damage calculated from the FAMOS rainflow matrix was found to be 0.15% higher than for the one from the developed monitoring system.

For the purpose of validation, the result from the monitoring system was also compared with the result received by using the WAFO[2] matlab toolbox [WAFO00]. WAFO include functions for offline rainflow calculations for time series. The number of cycles achieved from WAFO for the actual time series was exactly the same as achieved from the developed monitoring system. Although, the appearance of the rainflow matrix from WAFO was somewhat different then from the ones generated by the monitoring system and by FAMOS. It can also be observed that the WAFO rainflow matrix is stretched over a larger amount of classes then for the monitoring system and for FAMOS. The

[1] Damage scaled to 20 years.
[2] WAFO: Wave Analysis for Fatigue and Oceanography. WAFO is a freeware Matlab toolbox developed at the Department of Mathematical Statistics at Lund Institute of Technology.

reason for this is that the value-space, that should be covered by the classes, can not be fixed in WAFO. Since WAFO was developed for offline calculations, it automatically searches for the largest and smallest value of the time frame and split the value-space, between these two values, in the demanded amount of classes. Hence, this generates an optimal layout of the classes, although, this is only possible for offline mode, as discussed in Section 2.2.7. The damage calculated from the WAFO rainflow matrix was 0.45% smaller than the one calculated by the monitoring system. A comparison between the results received from the monitoring system, FAMOS, and WAFO is presented in Table 7.2. For all three calculations, the residual was always included and processed at the end of the calculations.

Method	**Nbr of Load-cycles**	**Scaled Damage**
Monitoring System	692	0.2909
FAMOS	690	0.2914
WAFO	692	0.2896

Table 7.2: Comparison between the results received with the monitoring system, FAMOS, respective WAFO for the first time frame in the simulation

The scaled damage calculated by the monitoring system for the first time frame in the simulation was about 30%. This means that if the same loads, as in this perticular time frame, would have been applied for 20 years, the total exhaustion of the blade would have been 30% of its total endurance.

Fig. 7.6 presents the SN-curve derived from the results from the monitoring system for the first time frame in the simulation. It can be observed that it has in some degree a linear appearance, since the x-axis is on a logarithmic scale, which was also expected if one considers the definition of the SN-curve in (3.1).

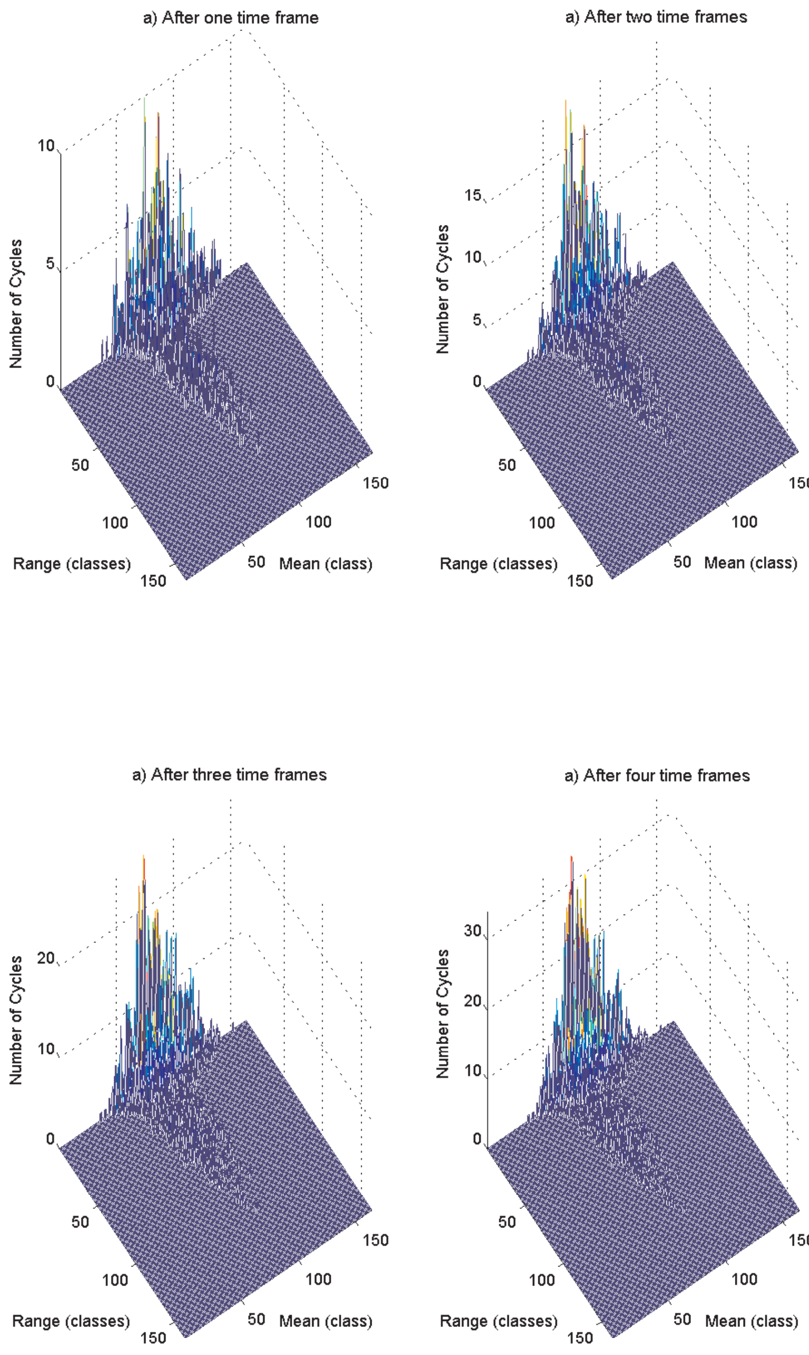


Figure 7.2: Development of the high resolution rainflow matrix, for the blxMe sensor, during four time frames

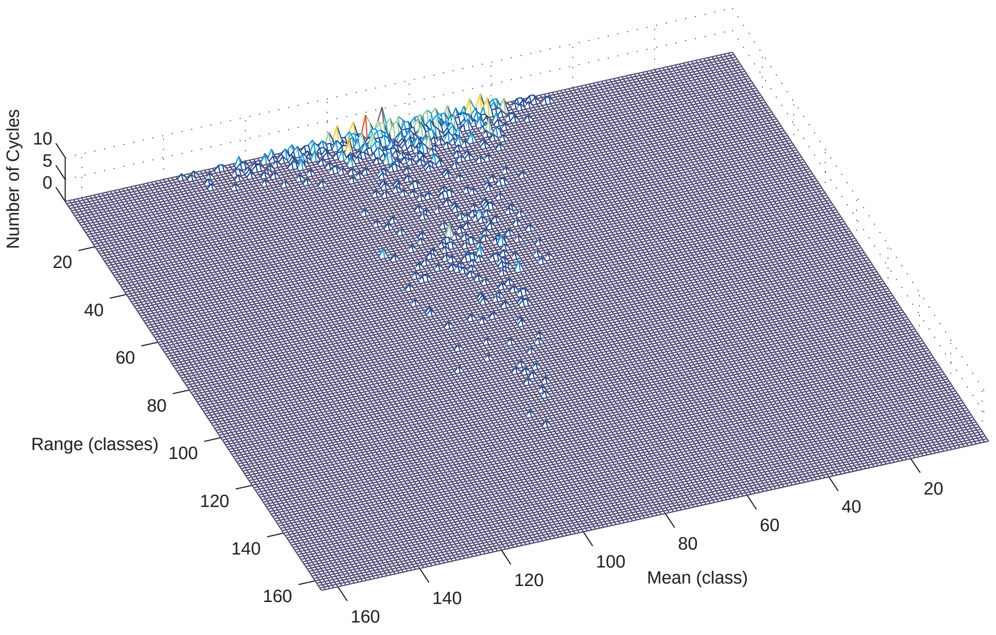


Figure 7.3: High resolution rainflow matrix calculated by the monitoring system, using internal classification, for a time frame from the blxMe sensor

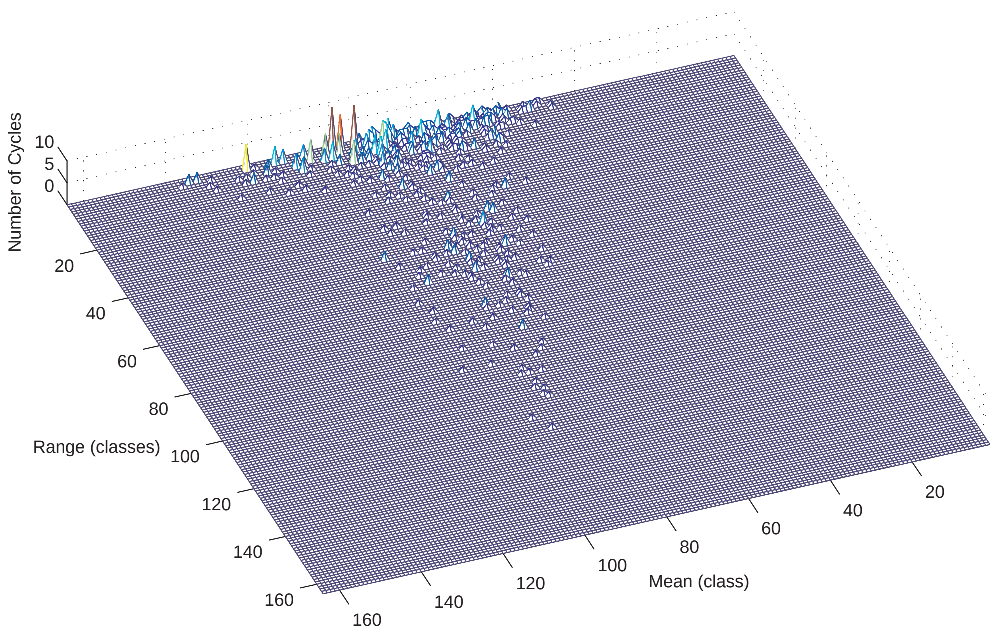


Figure 7.4: Rainflow matrix calculated with the signal processing tool FAMOS, for a time frame from the blxMe sensor

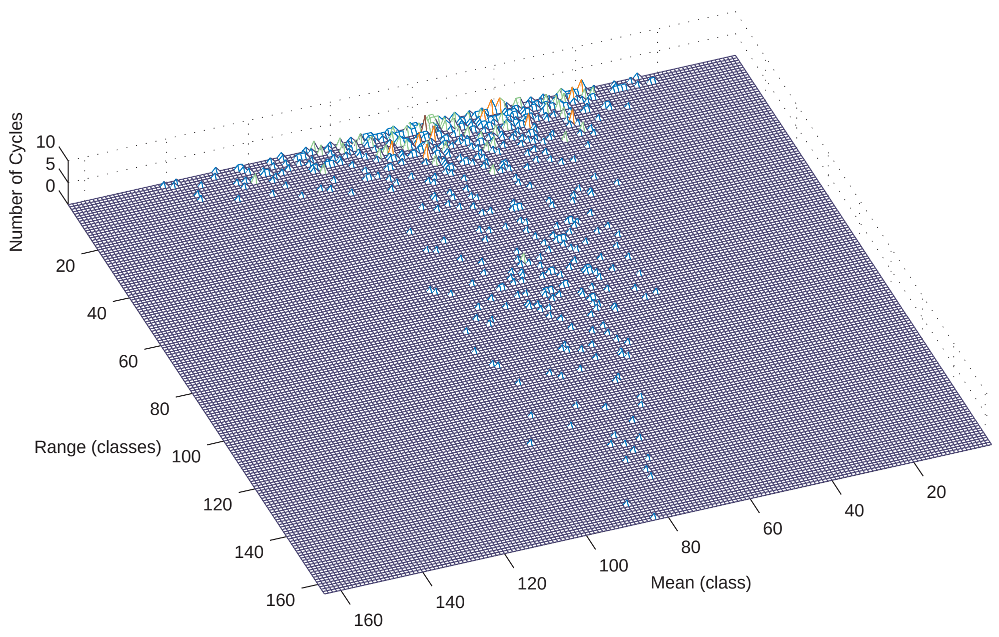


Figure 7.5: Rainflow matrix calculated with the Matlab toolbox WAFO, for a time frame from the blxMe sensor

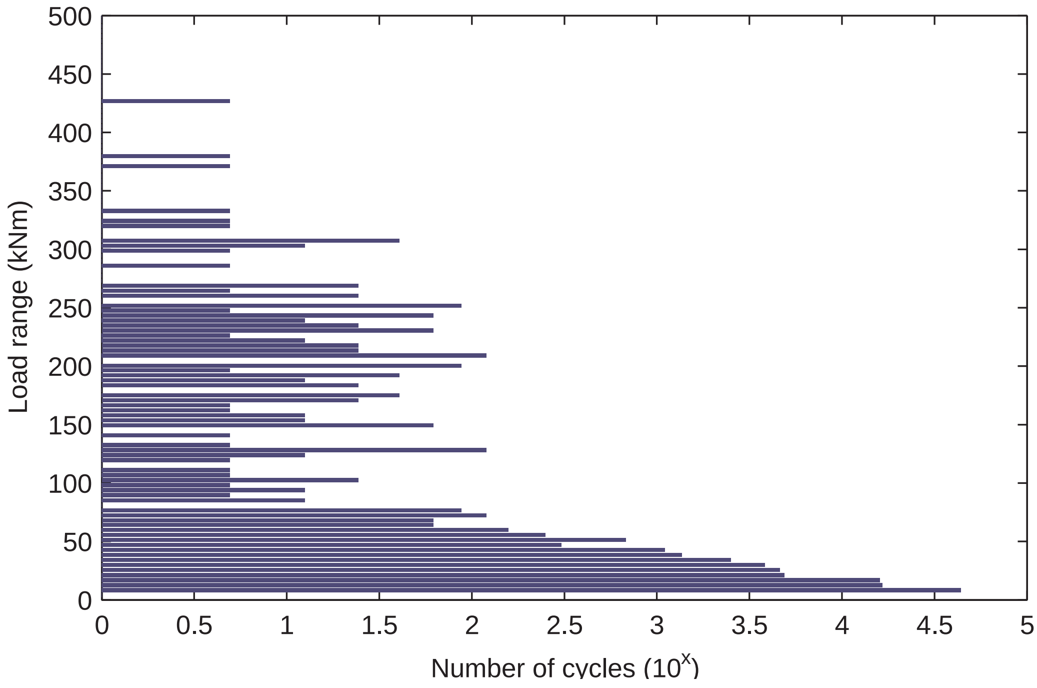


Figure 7.6: SN-curve for the first time frame in the simulation

Chapter 8

Calculations for a Measurement

The purpose of performing calculations on a measured time frame was to once more validate the developed monitoring system, thus ensuring that the system works as expected also for measured time frames. Thereby, the results from the HITL simulations could also be verified.

This measurement was carried out at the Wietmarschen wind turbine in Salzbergen, Germany. The time frame recorded came from the sensor measuring flapwise loads at one of the blades. The calculations were performed offline with the implemented Matlab functions, using internal classification, respective by GE Energy with the FAMOS signal processsing tool. The WAFO Matlab toolbox, also used for validation in the last chapter, could not be used in this case because it could not handle time frames of this size.

Since the calculations were performed offline, optimal layout of the classes could be achieved since the absolute maximum and minimum of the time frame were known.

The residual was included and processed at the end of the calculations.

Wind turbine model	GE 1.5 sl
Mean wind speed	11.15 m/s
Sensor monitored	blxMf
Number of classes	164
Upper limit for the high resolution sensor coverage	+2598.8 kNm
Lower limit for the high resolution sensor coverage	+1043.8 kNm
Class size	9.4817 kNm
Samples per time frame	30 000
Simulation time	10 Min

Table 8.1: Measurement specifications

When comparing the results achieved from the developed monitoring system with the ones achieved from FAMOS, see Table 8.2, it can be observed that they are in this case, as well as in the comparison of the results from the HITL simulation, very similar. The number of found load-cycles was in this case exactly the same for both methods. Although, the small differences in the distribution of the cycles in the rainflow matrix can still be seen when comparing Fig. 8.2 and Fig. 8.3. The difference in the scaled damage, calculated for each rainflow matrix, was 0.18%. Once more, the scaled damage calculated for the rainflow matrix from FAMOS was slightly higher than the one calculated for the rainflow matrix from the monitoring system.

For this recorded time frame, the scaled damage was considerably higher than the one calculated in Chapter 7. In this case, the exhaustion of the blade's endurance was as much as 90%. This large fluctuation can be explained by considering that the simulation and the measurement, for which the calculations were performed, were only 10 minutes long. Hence, a short, but large, fluctuation in the loads would have a large impact on the scaled damage. A 10 minutes long measurement is enough to use for validation purposes, but it is too short for making estimations of the lifetime of a component.

Method	**Nbr of Load-cycles**	**Scaled Damage**
Monitoring System	877	0.9002
FAMOS	877	0.9018

Table 8.2: Comparison between the results received with the monitoring system and with FAMOS

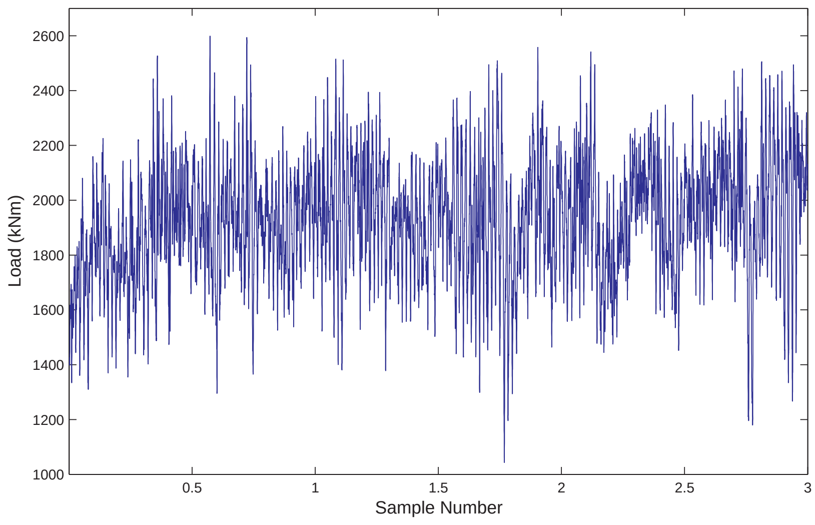


Figure 8.1: The recorded 10 minutes time frame for flapwise loads at one of the blades

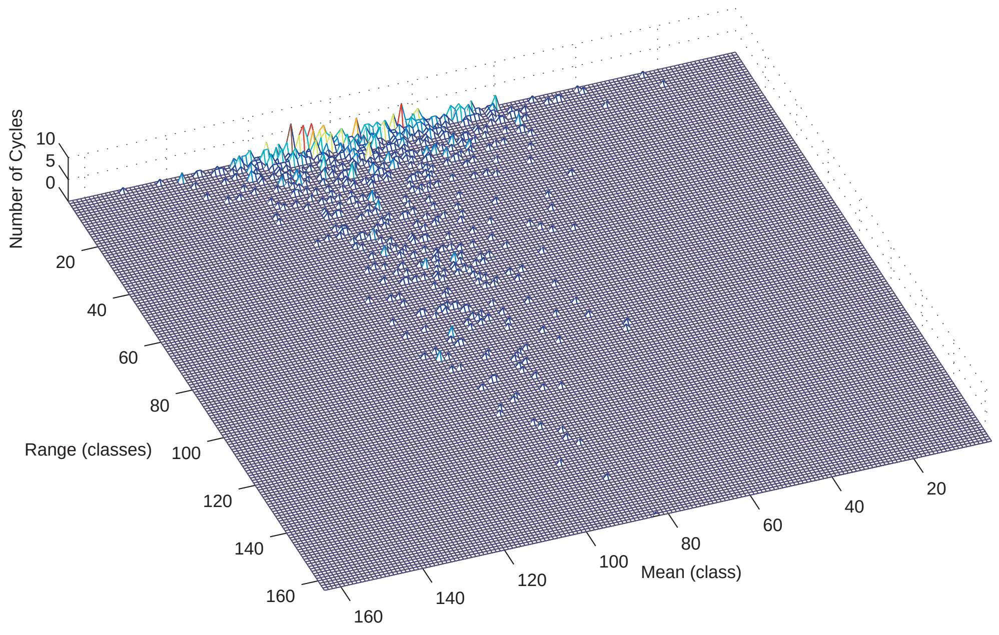


Figure 8.2: High resolution rainflow matrix calculated by the monitoring system, using internal classification, for a time frame from the blxMf sensor

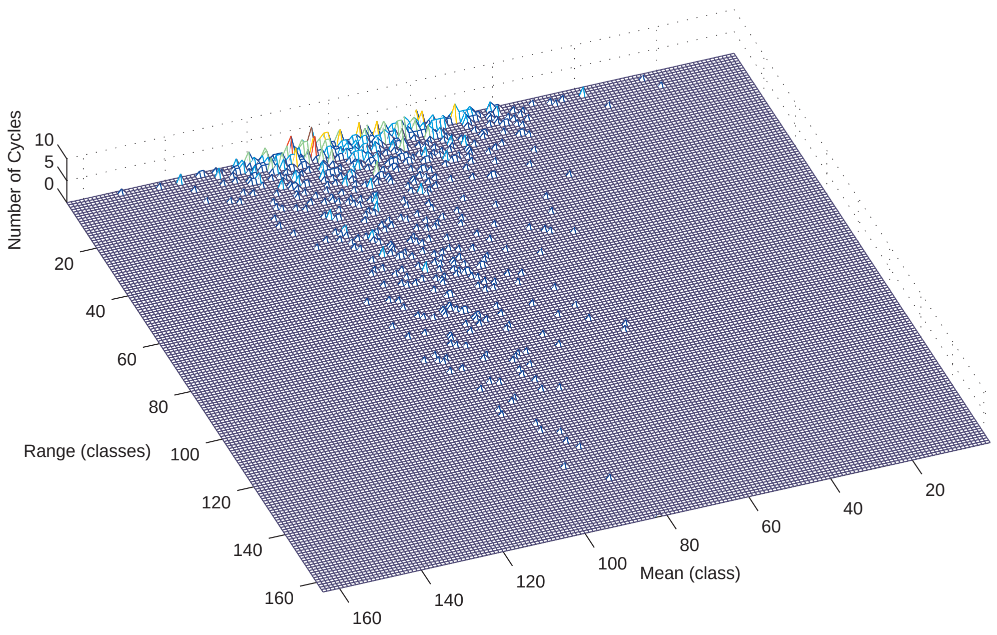


Figure 8.3: Rainflow matrix calculated with the signal processing tool FAMOS, for a time frame from the blxMf sensor

Part III

Conclusions and Perspectives

Chapter 9

Summary

The aim of this thesis was to design and implement a lifetime monitoring system for the GE 1.5s/sl wind turbine. Knowing the estimated remaining lifetime makes it possible to change the wind turbine components before they break. This increases both the security and availability of the wind turbine.

The rainflow counting method was chosen for extraction of load cycles from the load sequences. It has been shown best practice to use the rainflow counting method when acquiring relevant information from a load sequence, to monitor the lifetime of a component. The rainflow counting method can also be used for online processing, which was necessary for this thesis.

The classification of the data points in the time frames were performed in a conservative manner. The conservative classification method was chosen before the least-error method in order to ensure that the damage calculated is always at least as large as in reality, in no case less.

To make online rainflow counting possible, a new approach had to be found for the layout of the classes. By having two rainflow matrices, one with high resolution for a small value-space and one with low resolution for a large value-space, all necessary information from the rainflow counting could be stored. This was also a very memory efficient solution, demanding only a fraction of the memory needed for other possible solutions.

It was found that the life meter could be represented as the quotient between S_{eq} and S_{eq_max} in order to present measured damage equivalent loads against the designed maximum damage equivalent loads. The measured damage equivalent loads should be multiplied with safety factors in order to ensure that the components do not break before they are expected to.

Two design approaches for the monitoring system were implemented. The first implementation was more time-efficient, and also more conservative, than the second one because the classification was performed before the rainflow counting. In the second implementation, the classification of the turning points was carried out internally in the rainflow counting function block. This led to more accurate results, although, it was also more time consuming since a lower amount of data points were filtered out before the rainflow counting function block. By changing a parameter in the main program the user can decide which of the two implementations to use.

All POU's implemented were also translated into Matlab functions in order to be used for offline validation purposes.

The monitoring system was made robust to handle outer disturbances to the sensors and/or to the data acquisition system by having an internal error-detector and an error-handler implemented. The error-detector can detect faults such as sensor failure or discontinuities in or between the incoming time frames.

The validation of the lifetime monitoring system was carried out by comparing with the results achieved from the signal processing tool FAMOS and the Matlab toolbox WAFO. It turned out that the results matched each other very well. The difference in the results was always less then 0.5%. This difference was most probably due to a discrepancy between the classification methodologies used by the different systems.

A prototype of the monitoring system was succesfully installed in the Holsten-Bexten GE 1.5sle wind turbine in Salzbergen, Germany.

Chapter 10

Quality Assessments

During this thesis work, several simplifications and assumptions were, of course, made. These could make the results achieved less accurate, but are also often necessary since it is not feasible to perform empirical investigations for all unknown quantities. In this thesis, assumptions on the value of a few quantities have been made. E.g. the number of classes being used in the classification of the data points in the time series was chosen to be 164, because this had shown good practice by tests performed by GE Energy. It was a assumed that this number of classes was enough, although no own tests were performed for comparison and validation. On the other hand, it was also of interest to choose similar parameters as the ones being used by GE Energy since it made comparison of generated results possible. In this thesis, being able to do meaningful comparisons was more important than finding and using the optimal number of classes.

Another assumption was the position of the limits between the high resolution and low resolution rainflow matrices. The resolution needed was only approximately known. A one month long monitoring of the loads on a real wind turbine was studied, and the total maximum and minimum load taking place during this time were used as limits for the high resolution rainflow matrix. All loads beyond this value-space were considered to be rare and therefore a lower resolution was used for these loads. Only the approximate magnitude of these rare large loads was needed in order to be able to calculate a fairly correct fatigue. The resolution of the low resolution matrix was set to a tenth of the resolution of the high resoution matrix. This makes the total covered value space probably much larger than it would have to be. Although, it ensures that the really extreme loads will fit in the low resolution matrix.

To find accurate and unambiguous maximum DEL's to use for the damage calculations was not trivial. The maximum DEL's had to be taken from a wind turbine with the same specifications as the one where the life time monitoring took place. It was also necessary to use DEL's that had been calculated for exactly the same positions as for the monitored sensors. The DEL's used in this thesis were only given for different integer values of the material parameter m (see section 3.1). To use only a integer-value for m was an approximation since it is in reality a floating point value.

As mentioned in section 3.2, safety factors have to be multiplied to the total calculated damage in order to be sure that components do not fail before they are expected to.

The exact magnitude of these safety factors are not known and therefore such factors have not been included in the damage calculations in this thesis. In order to find out the necessary magnitude of the safety factors, several empirical tests have to be performed.

The differences between the results achieved with the, in this thesis, developed system and the ones achieved from the FAMOS software respective WAFO Matlab functions are most probably due to differences in the classification method used. Since the number of cycles found by each application were almost the same, it suggests that the rainflow counting operates in the same way in all these three methods. Although, the final fatigue calculated with the developed rainflow counter respective with the FAMOS signal-processing tool was the same. This was the most essential result in this thesis and therefore it was of most importance that this validation turned out positive.

Whenever an approximation had to be made in the calculations in this thesis, it was always ensured that it was conservative. This was of high importance since the estimated remaining lifetime of the components of a wind turbine is, for security reasons, never to be higher than it really is.

Chapter 11

Future Work

The lifetime monitoring system has been designed and implemented in this thesis work. A prototype of it has been installed into a wind turbine. Future work will now consist of carrying out tests on the prototype system over a longer period of time, in order to complete verification. By doing this, it will become clear if the accuracy of the system is satisfactory or if it has to be tuned in order to achieve long term reliability.

Multiplying the results with safety-factors has not been done in this thesis. This will also have to be considered in the future. By performing tests over a longer period of time, and for several different wind turbines, the safety-factors needed can be determined.

With the developed system, it is now possible to monitor the lifetime of several components of a wind turbine. A future task will be to develop new controller strategies that take use of the information that the monitoring system delivers in order to optimize the operation of the wind turbine regarding the lifetime of the components and the energy capture.

Bibliography

[AGRB94] C. Amzallag, J.P. Gerey, J.L. Robert, J. Bahuaud: *Standardization of the rainflow counting method for fatigue analysis*, International Journal of Fatigue, Vol. 16, pp. 287-293, 1994

[DeJi03] S. Amarnath, A. Deshpande, P. Jindal: *PP7 and Flex5 manual*, version 1.0, GE Wind Energy, 2003

[DoSo82] S.D. Downing, D.F. Socie: *Simple rainflow counting algorithms*, International Journal of Fatigue, Vol. 4, pp. 31-40, 1982

[DrHa95] K. Dressler, M. Hack: *Fatigue Lifetime Estimation on Rainflow Counted Data Using the Local Strain Approach*, 1995

[Echt96] A. T. Echtermeyer, et al.: *Method to predict fatigue lifetimes of GRP wind turbine blades and comparison with experiments*, European Union Wind Energy Conference, 1996

[Efun05] *eFunda - Engineering Fundamentals*, http://www.efunda.com/, access date 27/09/05

[Endo74] T. Endo: *Proceedings from Symposium of Mechanical Behaviour of Materials*, Society of Material Science, Japan, 1974

[Feic05] W. Feichter: *Real-time Monitoring for 1.5 Mega Watt Wind Turbine Generators*, Bachelor Thesis, Munich University of Applied Sciences, 2005

[Flei99] K. Fleischer: *LaTeX - mehr als nur schreiben*, Fernuniversität Hagen, 1999

[Germ03] *Richtlinie für die Zertifierung von Windenergieanlagen*, Germanischer Lloyd Wind Energie GmbH, Ausgabe 2003

[GEGR04] *Hardware-in-the-loop Simulator for Wind Turbine Control*, internal document, GE Global Research, Shanghai, 2004

[GEWE04] *GEWE - 3.6 Standard Europe*, GE Wind Energy GmbH, 2004

[GEWE05] *ADC-Report*, GE Wind Energy GmbH, 27 Revision, 2005

[Joha93] R. Johansson: *System Modeling & Identification*, Prentice-Hall Inc., 1993

[Joha99] P. Johansson: *Modelling of Random Vehicle Fatigue Loads through Transformed Gaussian Processed*, Department of Mathematical Statistics, Lund Institute of Technology, 1999

[Johb99] P. Johansson: *Rainflow Analysis of Switching Markov Loads*, Doctoral Thesis, Department of Mathematical Statistics, Lund Institute of Technology, 1999

[JoSM05] P. Johannesson, T. Svensson, J. de Maré: *Fatigue life prediction based on variable amplitude tests - methodology*, International Journal of Fatigue, 2005

[Jürg95] M. Jürgens: *LaTeX - eine Einführung und ein bisschen mehr...*, Fernuniversität Hagen, 1995

[Kopk04] H. Kopka: *Guide to LaTeX*, Addison-Wesley Publishing Company, 2004

[Lamp95] L. Lamport: *Das LaTeX Handbuch*, Addison-Wesley Publishing Company, 1995

[MaKa05] A. Mangold, S. Kleinhansl: *Load Assessment for the Wind Energy Turbine GE15sl, 1.5sl 50 Hz DIBt Zone II A LM37.3P2 100m*, GE Wind Energy GmbH, 2005

[MaKb05] A. Mangold, S. Kleinhansl: *Load Assessment for the Wind Energy Turbine GE15sl, 1.5sl 50 Hz IEC Type Class IIIA 7.5 m/s LM37.3P2 100m*, GE Wind Energy GmbH, 2005

[Mine45] M. A. Miner: *Cumulative damage in fatigue*, Journal of Applied Mechanics, Vol. 12, pp. A159-A164, 1945

[NaAm03] A. Natarajan, S. Amarnath: *Flex5 - Theory Manual*, Version 1.0, GE Wind Energy, 2003

[Palm24] A. Palmgren: *Die Lebensdauer von Kugellagern*, Zeitschrift des vereins Deutscher Ingenieure, Vol. 68, pp. 339-41, 1924

[Rix00] P. Rix: *Lastannahmen zur Enron Wind 1.5sl nach DIBt WZII*, Diploma Thesis, Enron Wind GmbH, 2000

[Rych87] I. Rychlik: *A new definition of the rainflow cycle counting method*, International Journal of Fatigue, Vol. 9, pp. 119-121, 1987

[Suth95] H. J. Sutherland: *Effect of Cyclic Stress Distribution Models on Fatigue Life Predictions*, SED, Vol. 16, ASME, pp. 83-90, 1995

[SöKR04] H. Söker, S. Kieselhorst, R. Royo, *Load Monitoring on a Mainshaft, a Case Study*, Proceedings from DEWEK, Wilhelmshafen, 2004

[WAFO00] The WAFO Group: *A Matlab Toolbox for Analysis of Random Waves and Loads*, Department of Mathematical Statistics, Lund Institute of Technology, 2000

[WeZe88] A. Westermann-Friedrich, H. Zenner: *Zählverfahren zur Bildung von Kollektiven aus Zeitfunktionen*, FVA-Merkblatt, Vol. 0/14, 1988

[Wind03] Windkraftjournal, Allianz Versicherungs-AG, 3/2003

[Wind05] *Positions of sensors installed on a GE 1.5sle at Bexten*, WINDTEST Kaiser-Wilhelm-Koog GmbH, 2005

Statutory Declaration

I do solemnly and sincerely declare that I have developed and written this enclosed Master Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others, or literal quotations, are clearly marked. This work has not been submitted to any other examining authority and has not yet been published.

Garching, 31. October 2005