Machine and Component Residual Life Estimation through the Application of Neural Networks

by

Michael Andreas Herzog

Submitted as partial fulfilment of the requirements for the degree

Master of Engineering
(Mechanical Engineering)

Department of Mechanical and Aeronautical Engineering
Faculty of Engineering
University of Pretoria

Supervisors: Professor T. Marwala and Professor P.S. Heyns

November 2006

Acknowledgements

I would like to express my sincere gratitude towards those that assisted and supported me in this endeavour:

- Prof. T. Marwala, Prof. P.S. Heyns and Dr. Corné Stander for their guidance, support and patience as my study leaders.
- Mr. Klaus-Peter Müller, for championing this project in every way right from the start, thereby making it possible.
- Prof. Hector Dreyer, for the important mentoring role he played.
- Mr. Sybrand Visagie, for his assistance in the manufacture of the laboratory test pieces.
- Mr. Herman Reiners, for creating the engineering drawings of the laboratory test pieces.
- My colleagues and friends, for their encouragement and understanding.
- My parents, for their love and inspiration.
- My heavenly Father, for the ever present hope when times were tough.

The Author
November 2006

Summary

Machine and Component Residual Life Estimation through the Application of Neural Networks

by

Michael Andreas Herzog

Supervisors: Professors T. Marwala

P.S. Heyns

Master of Engineering
Department of Mechanical and Aeronautical Engineering
University of Pretoria

Analysis of reliability data plays an important role in the maintenance decision making process. The accurate estimation of residual life in components and systems can be a great asset when planning the preventive replacement of components on machines. Artificial intelligence is a field that has rapidly developed over the last twenty years and practical applications have been found in many diverse areas. The use of such methods in the maintenance field have however not yet been fully explored.

With the common availability of condition monitoring data, another dimension has been added to the analysis of reliability data. Neural networks allow for explanatory variables to be incorporated into the analysis process. This is expected to improve the quality of predictions when compared to the results achieved through the use of methods that rely solely on failure time data. Neural networks can therefore be seen as an alternative to the various regression models, such as the proportional hazards model, which also incorporate such covariates into the analysis.

For the purpose of investigating their applicability to the problem of predicting the residual life of machines and components, neural networks were trained and tested with the data of two different reliability related datasets. The first dataset represents the renewal case where repair leads to complete restoration of the system. A typical maintenance situation was simulated in the laboratory by subjecting a series of similar test pieces to different loading conditions. Measurements were taken at regular intervals during testing with a number of sensors which provided an indication of the test piece's condition at the time of measurement. The dataset was split into a training set and a test set and a number of neural network variations were trained using the first set. The networks' ability to generalize was then tested by presenting the data from the test set to each of these networks.

The second dataset contained data collected from a group of pumps working in a coal mining environment. This dataset therefore represented an example of the situation

encountered with a repaired system. The performance of different neural network variations was subsequently compared through the use of cross-validation.

It was proved that in most cases the use of condition monitoring data as network inputs improved the accuracy of the neural networks' predictions. The average prediction error of the various neural networks under comparison varied between 431 and 841 seconds on the renewal dataset, where test pieces had a characteristic life of 8971 seconds. When optimized the multi-layer perceptron neural networks trained with the Levenberg-Marquardt algorithm and the general regression neural network produced a sum of squares error within 11.1% of each other for the data of the repaired system. This result emphasizes the importance of adjusting parameters, network architecture and training targets for optimal performance

The advantage of using neural networks for predicting residual life was clearly illustrated when comparing their performance to the results achieved through the use of the traditional statistical methods. The potential of using neural networks for residual life prediction was therefore illustrated in both cases.

Key Words: Neural Networks, Condition Monitoring Data, Residual Life

Opsomming

Die Vooruitskatting van die Lewensverwagting van Masjiene en Komponente deur middel van die toepassing van Neurale Netwerke

deur

Michael Andreas Herzog

Leiers: Professore T. Marwala
P.S. Heyns

Magister in Ingenieurswese
Departement van Meganiese en Lugvaartkundige Ingenieurswese
Universiteit van Pretoria

Die analise van betroubaarheidsdata speel 'n belangrike rol in die besluitnemingsproses vir instandhouding. Die akkurate vooruitskatting van lewensverwagting is 'n waardevolle aanwins wanneer komponente op masjiene voorkomend vervang moet word. Kunsmatige intelligensie is 'n veld wat vinnig ontwikkel het oor die laaste twintig jaar en toepassings is op 'n verskeidenheid terreine daarvoor gevind. Die gebruik van sulke metodes vir doeleindes van instandhouding is egter nog nie ten volle ontgin nie.

Met die algemene beskikbaarheid van kondisie-moniteringsdata is 'n nuwe dimensie tot die analise van betroubaarheidsdata gevoeg. Neurale netwerke laat toe dat verklarende veranderlikes in die proses geïntegreer kan word. Dit kan verwag word dat hierdie stap die kwaliteit van vooruitskattings sal verhoog wanneer dit vergelyk word met metodes wat alleenlik op tyddata staatmaak. Neurale netwerke kan dus gesien word as 'n alternatief vir die verskeidenheid regressiemodelle, soos byvoorbeeld die proporsionele gevaarkoersmodel, wat ook die gebruik van verklarende veranderlikes toelaat.

In hierdie verhandeling word daar 'n beskrywing gegee hoe neurale netwerke met betroubaarheidsdata geleer is en hul waarde bepaal is vir die vooruitskatting van die lewensverwagting van komponente en masjiene. Die werk is gedoen met twee datastelle wat betrekking het op die instandhoudingsveld.

Die eerste datastel het bestaan uit metings wat geneem is op 'n reeks enerse komponente wat by faling verwyder is. 'n Tipiese instandhoudingsituasie is in die laboratorium gesimuleer deur hierdie toetsstukke aan verskillende lastoestande bloot te stel. Metings is met gereëlde intervalle met verskeie sensors geneem gedurende elke toetslopie om die toetsstuk se toestand te bepaal. Die datastel is opgedeel in 'n opleidingsstel en 'n toetsstel waarvan die eerste gebruik is om 'n verskeidenheid neurale netwerke op te lei. Die tweede stel is vervolgens gebruik om die netwerk se reaksie op data te toets wat nog aan daaraan onbekend was.

Die tweede datastel het inligting bevat wat op ‘n groep pompe gemeet is wat in die mynbou veld opereer. Die datastel was dus ’n voorbeeld van die situasie wat by herstelde stelsels gevind word.

Daar is bewys dat die gebruik van toestandsdata in die meeste gevalle die neurale netwerk se akkuraatheid verbeter het. Die gemiddelde afwyking in die vooruitskattings van die verskillende neurale netwerke teenoor die werklike lewensverwagting was tussen 431 en 841 sekondes vir die hernuwings datastel wat ’n karakteristieke lewensverwagting van 8971 sekondes getoon het. Wanneer die netwerke wat met die Levenberg-Marquardt algoritme opgelei is, asook die algemene regressie neurale netwerk geoptimiseer is, was die som van die kwadrate van hul afwykings binne 11.1% van mekaar vir die herstelde stelsel se datastel. Hierdie resultaat bewys die belangrikhied daarvan om deur middel van die aanpassing van parameters, netwerkargitektuur en netwerkopleidingstykens die optimale akkuraatheid te bereik.

In beide gevalle is die voordeel wat neurale netwerke bied in vergelyking met statistiese metodes duidelik uitgewys. Die potensiaal wat die gebruik van neurale netwerke vir die vooruitskatting van lewensverwagting is dus geïllustreer.

Sleutelwoorde: Neurale Netwerke, Toestandmoniteringsdata, Lewensverwagting

Table of Contents

Summary ____ *i*

Opsomming ____ *iii*

Table of Contents ____ *v*

Abbreviations ____ *vii*

Symbols ____ *viii*

Chapter 1: Problem Statement ____ *1*

1.1 Maintenance Planning ____ **1**

1.2 Renewal and Repaired Systems Theory ____ **6**

1.3 Probabilistic Models ____ **9**

1.3.1 RP – Renewal Process ____ 9

1.3.2 NHPP – Non-homogenous Poisson Process ____ 10

1.4 Advanced Regression Models ____ **12**

1.4.1 Multiplicative Intensity Models ____ 15

1.4.2 Additive Hazard/Intensity Models ____ 16

1.4.3 Models with Mixed or Modified Timescales ____ 16

1.5 Combined Advanced Failure Intensity Models ____ **17**

1.6 Residual Life Estimation ____ **18**

1.7 Neural networks for Reliability Data Analysis. ____ **19**

1.8 Reliability Data Considerations ____ **21**

1.8.1 Failure ____ 21

1.8.2 Censored Data ____ 22

1.8.3 Series and Parallel Systems ____ 22

1.8.4 System Complexity ____ 23

1.8.5 Comparability ____ 23

1.8.6 Time and Age ____ 24

1.8.7 Cumulative Damage and Degree of Repair ____ 25

1.8.8 Significant Covariates ____ 25

1.8.9 Data Integrity ____ 27

1.9 Scope and Contribution ____ **27**

Chapter 2: Neural Networks ____ *30*

2.1 Introduction ____ **30**

2.2 Brief History ____ **31**

2.3 Multi-Layer Perceptron Networks ____ **32**

2.3.1 Network Structure ____ 33

2.3.2 Transfer Functions ____ 34

2.3.3 MLP Network Training ____ 37

2.3.4 Network Generalization and Overfitting ____ 42

2.3.5 Ill-Conditioning ____ 47

2.4 Radial Basis Function Networks ____ **48**

2.4.1 Training of RBF Networks ____ 51

2.5 Conclusion ____ **52**

Chapter 3: Renewal Dataset ____ *54*

3.1 Introduction ____ **54**

3.2 Test Motivation ____ **54**

3.3 Test Piece ____ **57**

3.4 Laboratory Equipment and Test Procedure ____ **61**

3.5 Sensors and Measurements ____ **63**

3.5.1 Strain Gauge ____ 64
3.5.2 Accelerometer ____ 66
3.5.3 Thermocouple ____ 66
3.5.4 Load Cell ____ 69

3.6 Discussion ____ **69**

Chapter 4: Life Prediction for the Renewal Dataset ____ *71*

4.1 Neural Network Selection ____ **71**

4.2 Pre-Processing and Network Inputs ____ **73**

4.3 Weibull Distribution ____ **75**

4.4 Gradient Descent Backpropagation Algorithm ____ **77**

4.5 Levenberg-Marquardt Algorithm ____ **79**

4.6 Levenberg-Marquardt Algorithm with Bayesian Regularization ____ **81**

4.7 General Regression Neural Network ____ **82**

4.8 Discussion ____ **84**

Chapter 5: Repaired Systems ____ *88*

5.1 Description of the dataset ____ **88**

5.2 Comparison of neural networks and statistical methods ____ **90**

5.2.1 Case 1 Trained with failure time data ____ 91
5.2.2 Case 2 Trained with an explanatory variable based on MTTF ____ 91
5.2.3 Case 3: Trained with two Vibration Covariates ____ 92

5.3 Results ____ **92**

5.4 Network comparison by cross-validation ____ **96**

5.4.1 Neural networks, training and cross-validation ____ 96
5.4.2 Levenberg-Marquardt Algorithm ____ 99
5.4.3 Levenberg-Marquardt Algorithm with Bayesian Regularization ____ 101
5.4.4 General Regression Neural Network ____ 101

5.5 Discussion ____ **102**

Chapter 6: Closure ____ *105*

Appendix A ____ *107*

Appendix B ____ *108*

Appendix C ____ *114*

Appendix D ____ *123*

Appendix E ____ *125*

References ____ *146*

Abbreviations

AIM	Additive Intensity Model
ARIMA	Auto Regressive Integrated Moving Average
AFTM	Accelerated Failure Time Model
BAO	Bad as Old
BPP	Branching Poisson Process
BRP	Branching Renewal Process
CM	Condition Monitoring
EHRM	Extended Hazard Regression Model
FAA	Federal Aviation Administration
FEM	Finite Element Methods (Check)
FOM	Force of Mortality
GAN	Good as New
GRNN	General Regression Neural Network
HPP	Homogenous Poisson Process
i.i.d.	Independent and Identically Distributed
LM	Levenberg-Marquardt Algorithm
MLP	Multi-Layer Perceptron
MTTF	Mean Time to Failure
MTTR	Mean Time to Repair
NHPP	Non-Homogenous Poisson Process
PAR	Proportional Age Reduction
PAS	Proportional Age Setback
PHM	Proportional Hazards Model
PMIM	Proportional Mean Intensity Model
POM	Proportional Odds Model
PWP	Prentice Williams Peterson Model
RBF	Radial Basis Function
ROCOF	Rate of Occurrence of Failure
RP	Renewal Process
SGBP	Steepest Gradient Backpropagation Algorithm
SRP	Superimposed Renewal Process

Symbols

Statistical Methods (Chapter 1)

x	Local time
t	Global time
f(x)	Failure probability density function
F(x)	Cumulative failure distribution function
h(x)	FOM
$\lambda(t)$	ROCOF
n(x)	Number of surviving units at time x
Δn	Number of failures within the interval Δx
N(t)	Number of failures up to time t
L	Laplace variate
T_k	Arrival time of failure k measured in global time.
X_i	Length of the i^{th} life of a unit.
β	Weibull parameter that determines the shape of the distribution
η	Weibull scale parameter also known as the characteristic life
γ	Weibull parameter that shifts the distribution in time
$\rho_1(t)$	Log-linear process
$\rho_2(t)$	Power law process
$\bar{z}$	Vector of covariates that may be dependent on time.

MLP Neural Networks (Chapter 2)

N	Number of training patterns
C	Number of network outputs.
$y_k(\bar{x}^n, \bar{w})$	Output of unit k as a function of the input vector and the weight vector.
t_k^n	Target vector for the unit k.
$\bar{g}_k$	Calculated gradient for the k^{th} iteration
$\Delta\bar{w}_k$	Change in weight vector for the k^{th} iteration
$\bar{w}_k$	Weight vector for the k^{th} iteration
α_k	Gradient descent backpropagation learning rate for the k^{th} iteration
β	Gradient descent backpropagation momentum constant

$J(\overline{w}_k)$	Jacobian matrix
$H(\overline{w}_k)$	Hessian matrix
μ	Levenberg-Marquardt parameter
$e(\overline{w}_k)$	Vector of network errors for the kth iteration

Conjugate Gradient Algorithm (Chapter 2)

$\overline{d}_j$	Search direction
α_j	Step size coefficient
β_j	Search direction coefficient

Regularization and Pre-Processing (Chapter 2)

Ω	Penalty term introduced during regularization
υ	Parameter controlling the size of the penalty
x_i	Variables to be transformed
$\overline{x}_i$	Mean of the variables
N	Number of patterns
σ_i^2	Variance of the variables
$\tilde{x}_i^n$	Transformed variables

RBF Neural Networks (Chapter 2)

u_j	Output of the the j^{th} node in the RBF hidden layer
$\overline{c}_j$	Position vector of the j^{th} hidden node
σ_j^2	Width of the j^{th} hidden node
$\overline{x}$	Input vector
$\overline{w}_k$	Weight vector of the the k^{th} node in the linear output layer
y_k	Output of the the k^{th} node in the linear output layer

Strain Measurement (Chapter 3)

ε	Effective strain at point of measurement
U_A	Bridge output voltage
U_E	Bridge excitation voltage
k	Gauge factor

Chapter 1: Problem Statement

The purpose of this chapter is to discuss a variety of concepts that are relevant to the subject of reliability data analysis and place the current work in context. The areas of maintenance planning and strategy are briefly discussed to paint a background picture for further discussion. A brief glance at statistical methods and previous research using neural networks for reliability data analysis serves to place the current work into context.

1.1 Maintenance Planning

Pintelon, Gelder and Van Puyvelde (1997) identify two decision making levels concerning the maintenance function, namely strategic and tactical planning. Strategic planning takes place at a high level in the organization and is usually dealt with by the discipline of engineering economics, though there are synergies with the more technical aspects which are addressed by tactical planning. Strategic planning relates to the making of long term decisions which include equipment replacement decisions, life-cycle costing, availability, capacity, flexibility, impact of technology changes, economic factors and investment criteria.

Tactical planning, in comparison, concentrates on such aspects as reliability, maintainability and optimal maintenance policy. The goal is to maintain the equipment responsible for production in such a condition to ensure maximum availability. Figure 1 gives a general summary of the numerous strategies which are available to the maintenance practitioner to help him with the achievement of this aim.

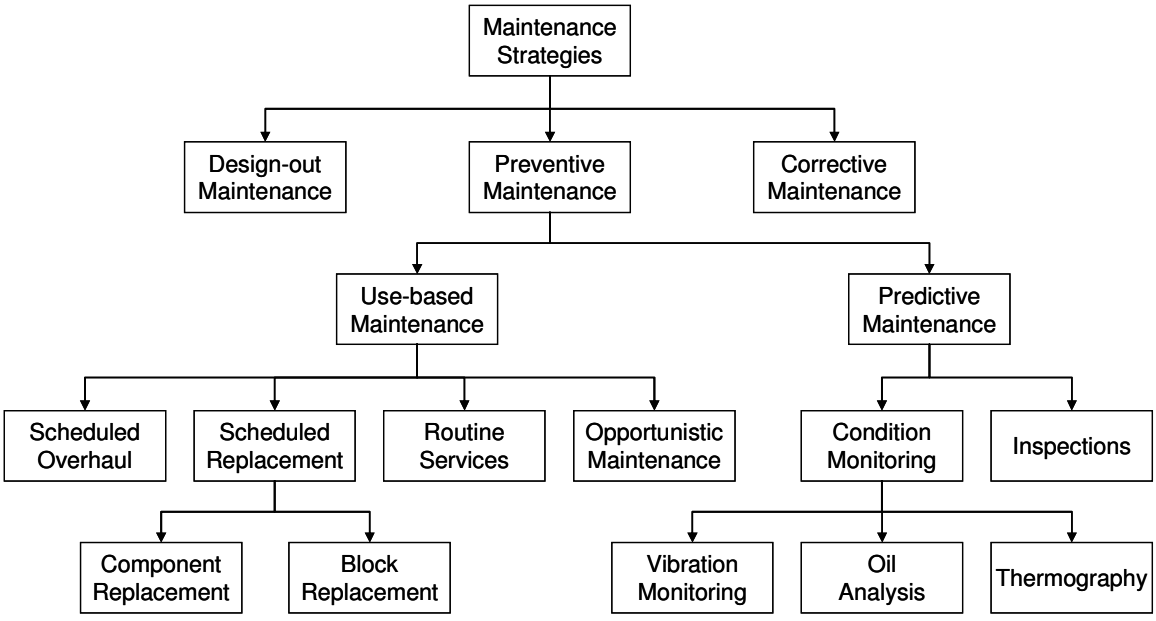

Figure 1: Maintenance strategies (Source: Coetzee (1997))

Maintenance policy in the early years rested on the principle of corrective maintenance. When a component failed, it was replaced and the system was repaired. The risks of undesirable safety and environmental consequences which may result from failures of modern systems have made the prevention of failure events a priority today. Apart from these hazards, there are also increasingly compelling economic reasons for the shift from a repair to a prevention centred approach. The secondary damage resulting from the failure event and the loss of availability caused by the unexpected failure may carry a prohibitive cost and result in a loss of competitiveness in the market which necessitates preventive intervention.

These economic and social pressures encouraged the search for methods to help prevent failures from occurring and led to the introduction and development of preventive maintenance philosophy. Research and development was mainly centred on high risk activities such as nuclear power generation and the aircraft industry, where failure would result in severe consequences and loss of life. Failure prevention was initially achieved through inspections and timely replacement of components at scheduled intervals, but techniques of measurement were subsequently developed that allowed preventive replacement on the basis of system condition. Despite these developments, failure prevention is not applied everywhere due to its high cost. Breakdown maintenance still remains in common use.

Apart from focusing on a failure prevention approach, greater availability may also be ensured by measures encompassing the design of the plant (see Figure 1) or the logistic arrangements relating to maintenance. For reducing the adverse effect of breakdowns on plant and equipment availability, redundancy may be introduced through the installation of a standby system. The downtime suffered due to failure is thereby minimised, as the plant production is switched to the standby system while repairs are affected. Such a standby system does however increase the complexity and the initial capital cost of the plant. A reduction in downtime resulting from failure can also be achieved by increasing the number of spares held in the plant inventory. This eliminates the delivery time of the equipment suppliers on critical items. Unfortunately high inventory levels come with the burden of high expenditure on assets that are not directly contributing to production.

The bathtub curve (see Figure 2) was initially based on human mortality trends in a developing country, but has also been useful when applied to equipment to illustrate the pattern of failure. It consists of three distinct phases, namely infant mortality, a constant failure rate phase and finally age related failures. The infant mortality phase is attributed to manufacturing error and faulty installation, while random failures occur due to system overloading and operator error. Age related failures can be assumed to result from the normal wear and tear introduced during operation.

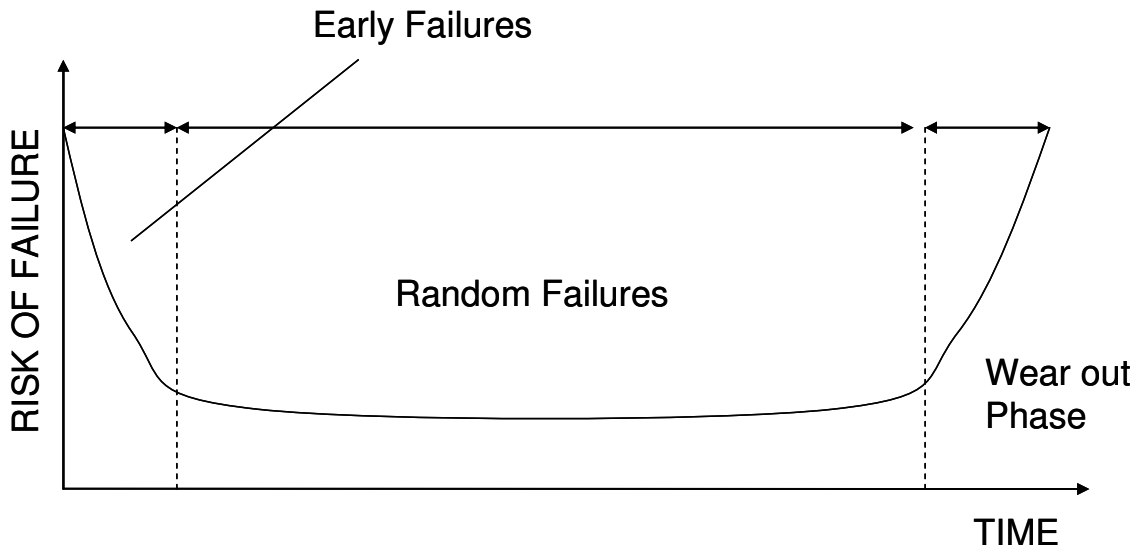

Figure 2: Bathtub curve (Sources: Jardine (1973) and Coetzee (1997))

The aim of use based maintenance is to replace units before failure occurs and the replacement decision is based on the age of the unit. An optimal time for making such replacements is chosen through the analysis of historic data or perhaps less scientifically it may be based on the experience of the maintenance practitioner. Use based maintenance is therefore only applicable to systems that have a pronounced phase of wear-out failures, where a time limit can prevent the largest component of age related failures.

Based on the notes of Bradley (1993), use-based preventive maintenance can be applied if the following conditions are met:

1. The failure rate must be increasing.
2. There must be a cost saving if replacement takes place before failure as opposed to after the failure.
3. The effect of the replacement must be a reduction in failure rate.
4. The maintenance intervention must not introduce faults which will result in an increase the failure rate.
5. It must be possible to predict the approximate time of failure, should preventive action not be taken.

An accurate prediction of failure time therefore forms one of the core requirements for the successful implementation of a preventive maintenance strategy. The introduction of preventive maintenance consequently generated a need for methods which could assist with effective failure time prediction. The optimization of preventive replacement decisions led to the introduction of statistical methods for determining the most favourable replacement intervals of machine components. By gaining a more accurate picture of failure trends it was hoped that the risk of unexpected failure could be reduced.

It is not cost effective to simply keep increasing the level of preventive action. Once the most costly failure modes have been addressed, the further reduction in breakdowns does not justify the expense of prevention (see Figure 3). An important function of tactical planning involves the finding of the optimum ratio between the costs of preventive and corrective maintenance policies. The trade-off between downtime and repair related costs associated with failures and the costs associated with excessive preventive maintenance, such as the resulting compromise in availability, must be balanced through management planning and intervention. Maintenance activities can, as a result, be structured in such a way as to minimize cost.

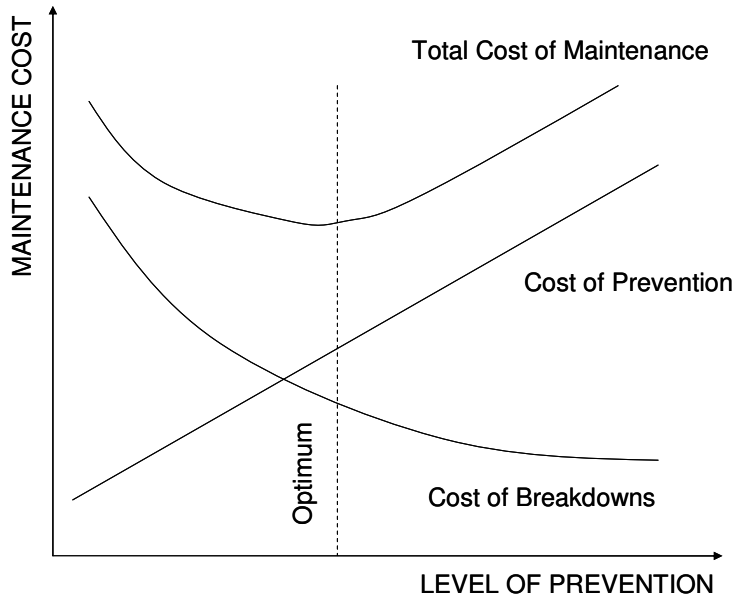

Figure 3: The influence of failure prevention levels on maintenance cost.
(Sources: Jardine (1973) and Coetzee (1997))

The replacement interval in a preventive policy is usually chosen according to the statistical distribution of previous failures. The allowable operating lifespan of all units is therefore conservatively dictated by the weaker units that have failed first. Replacement decisions based on age therefore lead to the discarding of components that are still in fairly good condition, which in some cases could have run substantially longer without the risk of failure. It is clear that the conservative approach revolving around traditional statistical methods limits the useful life of components.

While preventive replacement reduces wear and tear related failures, regular maintenance intervention introduces faults into the system due to human error, the opening of units and the handling of components. An undesirable hyper-exponential spike of failures often occurs shortly after an overhaul. Unnecessary maintenance intervention was therefore found to reduce the reliability in some systems.

Coetzee (1997) refers his readers to the work of Nowlan and Heap (1978) who show that the traditional bathtub curve only applies to 4% of the cases that they considered. It was further found that 89% of the cases in their study would not benefit from use-based replacement. Irrespective of how representative these results are, they give a good indication that use-based replacements are not suitable for a large proportion of

practically encountered applications. The only way to prevent failure in such systems is to measure the actual condition of the system and affect replacement once a clear deterioration is identified.

For the previously mentioned reasons, condition based maintenance (see Figure 1) has over the last two decades come to play an ever increasing role in the maintenance field. A number of techniques have successfully been developed to monitor the actual deterioration of the system and its components. Such techniques provide a warning when a component reaches a point where failure is imminent and the machine can subsequently be stopped for the replacement of the components in question. The risk of unexpected failure is thereby greatly reduced, while components can also be run close to the end of their useful life. Virtually the full component life can be exploited, and the required components can be ordered well in advance.

The usefulness of condition monitoring data extends beyond the detection of worsening system condition. Three main categories of information contained in condition monitoring data are identified:

- Signs of imminent failure
- Faulty installation
- The nature of operating conditions

The first category encompasses the traditional activity that is commonly labelled as condition monitoring. Regular measurements are taken in order to identify and monitor machine deterioration with the intention of replacing or overhauling faulty components/units before actual failure occurs.

The second category involves the early identification of faulty installation that would lead to increased wear and/or premature failure. Rapid identification of such a problem after a maintenance event and timely intervention can help to prevent or reduce consequential damage to the system. An example of such a problem could be the identification of a misalignment problem through vibration monitoring, or the detection of contamination in gearbox oil caused by a faulty breather or lack of cleanliness during installation.

The data collected by means of condition monitoring also contains information with regards to the operating conditions of the equipment. Operating conditions have a direct influence on the rate at which a system's condition deteriorates and are therefore valuable when attempting failure prediction. The life of machines subjected to a corrosive environment or a harder working material is usually reduced by these factors. In a similar way the forces resulting from a specific mode of operation may reduce the life of one machine when compared with an identical unit employed in a different application.

Condition monitoring data contains a wealth of information that is not only useful to prevent failure, but provides a comprehensive picture of the system that can be put to use to enhance the prediction of failure. The present study investigates the combination of the traditional statistical life estimation concept and the measurement of component condition and degradation. By bringing these concepts together in analysis, it is envisaged that reliability related information of greater quality could be generated for the purpose of facilitating maintenance decision making.

1.2 Renewal and Repaired Systems Theory

Renewal theory was built around the assumption that both scheduled reconditioning and scheduled replacement leaves the maintained item in the 'good-as-new' (GAN) condition after preventive action. It therefore applies to the situation where preventive actions based on equipment usage lead to complete restoration. The modelling of this situation is most regularly approached through the fitting of statistical distributions, though other mathematical theory bases may also be applied.

The force of mortality (FOM) or hazard rate of an item, gives the probability of failure within a short time window, from x *to* $x + \Delta x$, provided that the item survived up to that time. Equation 1.1 appears in a similar form in Vlok (2001)

$$h(x) = \lim_{\Delta x \to 0} \frac{\Pr[x \le X < x + \Delta x \mid X \ge x]}{\Delta x} \tag{1.1}$$

Coetzee (1997) presents the FOM more simply, as:

$$h(x) = \frac{1}{n(x)} \cdot \frac{\Delta n}{\Delta x} \tag{1.2}$$

A graph illustrating the concept of a good-as-new (GAN) system, as assumed by renewal theory, is shown in Figure 4. After failure it is returned to its original state through repair and the FOM is reduced to zero. This assumption only holds true if no trend can be observed in the subsequent failure times of the repaired system. According to Pijnenburg (1991) and Coetzee (1997), renewal theory only applies when the data is independent and identically distributed (i.i.d.). This means that the failures can be assumed to come from the same statistical distribution which implies that the one failure does not influence the next. The existence of a data trend means that this assumption does not hold and the various lifetimes cannot be lumped together as one dataset.

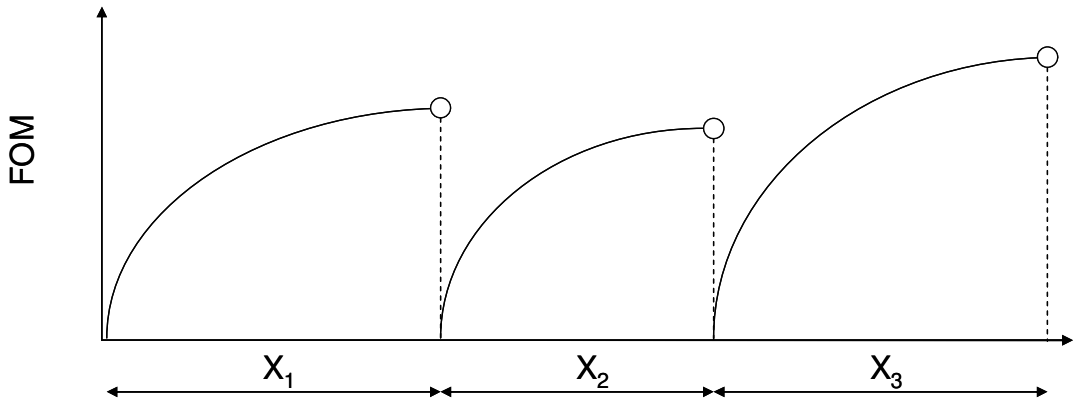

Figure 4: Illustration of the good-as-new concept. (Source: Vlok (2001))

Renewal theory is unquestionably useful for analysing the reliability data of components that are discarded upon failure. Systems in practice are mostly repaired systems and it has only been through recent work that attention has been drawn to the fact that renewal theory cannot be applied to most of these systems. According to Crowder et al. (1991) the distinction between repairable and non-repairable systems was neglected until the publications of Ascher and Feingold (1984) and the Ansell and Phillips (1989) Royal Statistical Society paper. The reader is also referred to Ascher (1981) which specifically addresses this issue. While the renewal situation can be represented by a distribution function, the repaired case is a stochastic process. The necessity that repairable systems must be analyzed with different techniques than the existing methods used for non-repairable systems must therefore be highlighted. According to most literature on this subject, the incorrect use of renewal theory still remains a common practice in the maintenance field.

In the paper published by Ascher (1981), he highlights the fact that there is no connection between the wearing out of a part and the deterioration of a system. The failure characteristics of individual components that make up a system can therefore be modelled using renewal theory, while regression is used to model the long term life trend of the whole repairable system. For non-repairable systems the lifetime of each individual unit is of interest. In contrast, for a repaired system the point process of the failure times of a particular unit is of interest. A stochastic point process is defined by Crowder et al. (1991) as a sequence of highly localized events, in this case failure, distributed in a continuum, which we take as time, according to some probabilistic mechanism.

Instead of the FOM applicable to renewal system, a rate of occurrence of failure (ROCOF) which is also referred to as the peril rate and denoted by $\lambda(t)$ has to be defined to describe the repaired system. The equation for the ROCOF is given in a similar format as previously presented by Vlok (2001).

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{\Pr[N(t+\Delta t) - N(t)]}{\Delta t} \tag{1.3}$$

The peril rate is therefore the probability of the failure event within a time interval, where t represents global time and $N(t)$ represents the number of failures up to time t.

The bad-as-old (BAO) assumption (shown in Figure 5) was introduced to describe the behaviour of repaired systems. According to this assumption, the system is brought back to the state in which it was before the failure or preventive maintenance action occurred. It is therefore not restored to its original condition, as is the case with GAN. In contrast to the FOM for the GAN case (Figure 4) which returns to zero upon repair, it can be seen from Figure 5 that the ROCOF immediately before and after the failure is the same. Repairable systems theory, based on BAO, therefore takes into account system degradation during operation which is time dependant. It follows that the ROCOF function is dependent on global system time, rather than the time that has elapsed since the last failure. The most common class of model that is used on repairable systems is probably the non-homogenous Poisson process (NHPP) model.

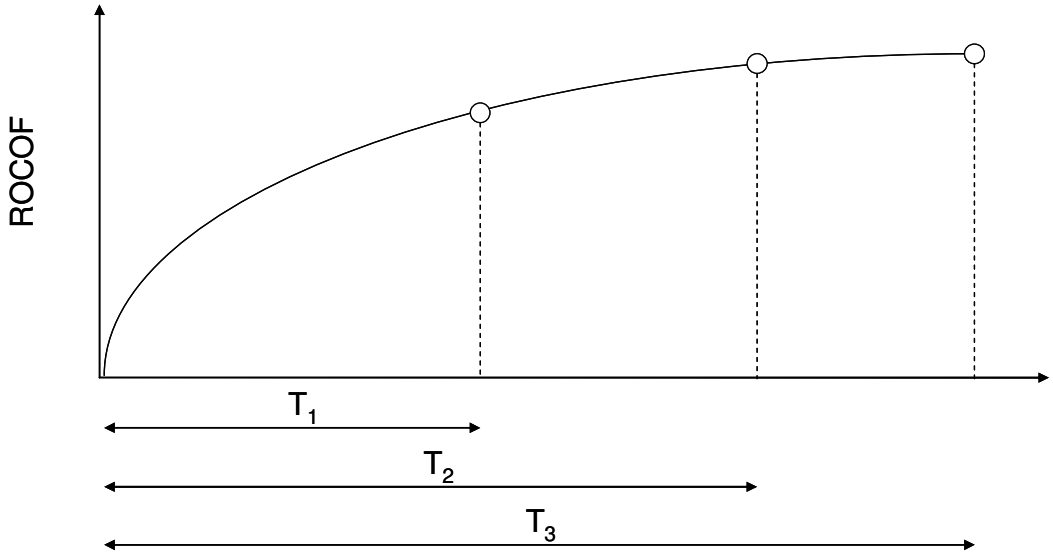

Figure 5: Illustration of the bad-as-old system (Source: Vlok(2001))

The choice of the most suitable assumption, whether it is GAN or BAO, requires the data to be first tested for dependence and trend. The absence of a trend and dependence in the data means that renewal theory can be used. Trend testing can only be done if the failure times are in order of occurrence. If X_1, X_2, X_3 ... X_n are successive lives of a unit, and T_1,T_2,T_3 ... T_n are the respective arrival times of these failures measured from a common starting point then:

$$T_k = \sum_{i=1}^{k} X_i \qquad (1.4)$$

These failure arrival times can be evaluated with the Laplace trend test, which is a popular method for testing such reliability data. According to Coetzee (1997), the Laplace test takes the following form:

$$L = \frac{\frac{1}{n-1}\sum_{i=1}^{n-1} T_i - \frac{T_n}{2}}{T_n\left[\frac{1}{12(n-1)}\right]^{1/2}} \qquad (1.5)$$

A negative result with a large absolute value indicates that the sample mean of the first *n-1* lives is small when compared to the midpoint of the observation interval. The latter failures are thus further apart indicating an improvement in reliability. A large positive result indicates that the sample mean of the first *n-1* lives is significantly larger than the midpoint of the observed interval and the reliability of the system is deteriorating. In both cases renewal theory does not hold and analysis methods must be used that can take such a trend into account. If the calculated answer is close to zero, it indicates that randomness exists in the failure patterns and renewal theory may be applied.

1.3 Probabilistic Models

The probabilistic modelling approach assumes that the failure times of a specific renewal process have the same underlying distribution. Time is the only variable on which the reliability function is based, and other failure related information is therefore ignored. A number of these models are described briefly, to give the reader an overview.

1.3.1 RP – Renewal Process

The renewal process formed the basis of statistical analysis of reliability data. Its use is only appropriate for repaired systems when the assumption that the lifetimes are i.i.d. holds true. The assumptions imply that the system is as good as new after each repair which allows the fitting of lifetime distributions such as the normal, exponential, Weibull, lognormal and gamma distributions to the data. Bain and Engelhardt (1991) discuss the application of probabilistic modelling in detail, with attention given to the techniques that are available for fitting these distributions to reliability data. Coetzee (1997) emphasizes the importance of the Weibull distribution for maintenance related work due to its versatility. Weibull (1951) suggested this empirical formula, which has the flexibility to represent the hazard function of most real life failure data and can therefore adapt to most failure situations found in maintenance practice.

According to Coetzee (1997), the hazard rate of the Weibull distribution with two parameters can be expressed as follows:

$$h(x) = \frac{\beta}{\eta}\left(\frac{x}{\eta}\right)^{\beta-1} \tag{1.6}$$

In this equation the parameter β is responsible for the shape of the curve while η is a scale parameter. The value of the parameter η is important because it represents the characteristic life of the population, with 63% of failures occurring before and 37% after this time.

To illustrate why the Weibull distribution is suitable for application to such a wide spectrum of reliability data, a diagram similar to an example given in Coetzee (1997) is shown in Figure 6. It can be seen that all the more commonly encountered shapes of the hazard function can be attained with different values of the shape parameter β.

When $\beta<1$, the Weibull function represents the decreasing hazard rate of infant mortality, as encountered during the first phase of the bathtub curve (Figure 2). A constant hazard rate is indicated when $\beta=1$ which means that failures occur at random. This corresponds to the second phase of the bathtub curve when failures are not age related. The case where the hazard increases proportionally with use can be represented by a Weibull function with $\beta=2$, while an increasing hazard rate with a convex shape corresponds the range $1<\beta<2$. The concave increasing hazard rate of the last phase of the bathtub curve can be represented when the parameter $\beta>2$. Use based maintenance can only be applied successfully in cases where $\beta>1$. Age based replacement does not offer any advantage when the hazard rate is constant or decreasing, as the risk of failure is not increasing with time.

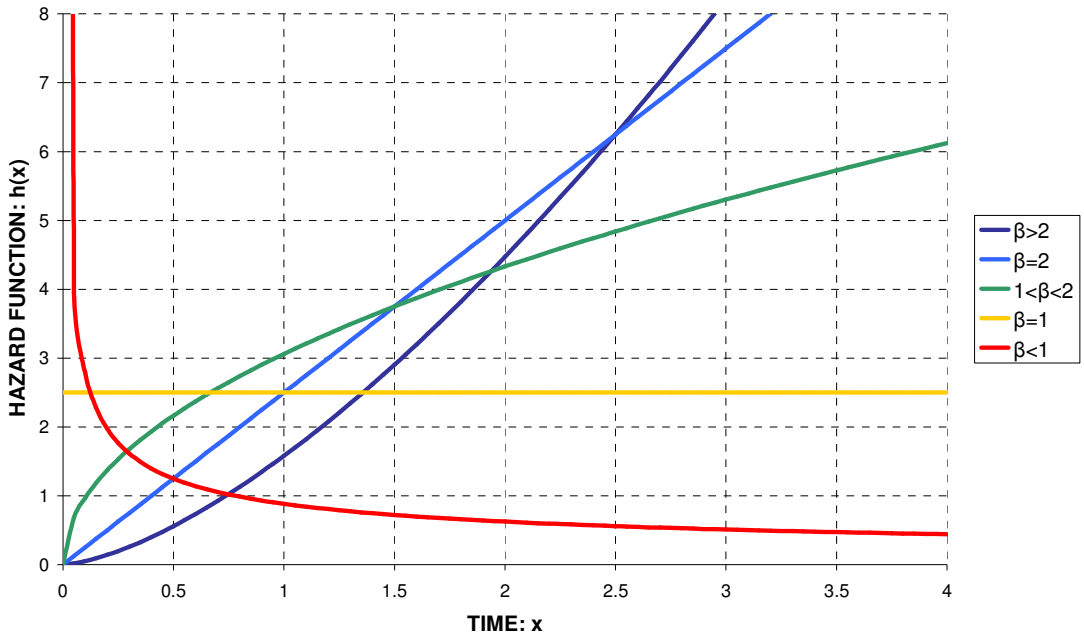

Figure 6: An example illustrating the variance in the shape of the hazard rate of the Weibull function depending on the value of the parameter β (Source: Coetzee (1997))

The Weibull distribution can also be applied in a three-parameter form. An additional parameter (γ), which represents the minimum life of the component, serves to shift the whole distribution in time.

1.3.2 NHPP – Non-homogenous Poisson Process

Pijnenburg(1991) states that the homogenous Poisson process (HPP), together with the renewal process, are the most commonly used methods in reliability data analysis. Their popularity is ascribed to their mathematical tractability, despite the fact that they are unsuitable for repaired systems that deteriorate with time. The lifetimes are exponentially distributed and i.i.d., resulting in a constant hazard rate. This assumption is often unrealistic because of the lack-of-memory property of the exponential distribution. The HPP can only be used for repairable systems if no trend is detected, in other words the GAN case.

The non-homogenous Poisson process is a generalization of the HPP where the ROCOF may vary with time instead of being constant. It therefore becomes possible to model a trend such as the deterioration of reliability encountered in the BAO case. The NHPP is a natural development from the use of a hazard function for non-repairable systems and is popular due to its mathematical tractability.

Two different NHPP models have gained common acceptance, accommodating ROCOF functions with different shapes. The equations for both these methods can be found in Coetzee (1997).

The first method is called the log-linear process and has the form:

$$\rho_1(t) = e^{\alpha_0 + \alpha_1 t} \qquad (1.7)$$

The shape of this function is upwards concave and applications for which it is suitable are not regularly encountered in practice. It represents the extremely fast increasing failure intensity rate.

The second method, which is more commonly applied, is called the "power law process":

$$\rho_2(t) = \lambda \cdot \beta \cdot t^{\beta-1} \qquad (1.8)$$

This function has different shapes depending on the value of the parameter β. A β with a value less than 2 results in a concave shape, while the function is convex if β is larger than 2. The power law function can therefore be applied to a wide range of failure processes with the exception of those with an intensity that increases rapidly. In such cases, the log-linear process can however be used, which means that these two intensity function types can cover a comprehensive range of failure processes which display an increasing rate.

An example of the application of the NHPP is found in Leung and Cheng (2000) who used the power law process to model the failures of engines in a fleet of Hong Kong buses. The Laplace trend test was used to establish if a trend of reliability deterioration was present. The model parameters were established by means of the maximum-likelihood estimation method. Cramer-von Mises was used to check the goodness of fit. The authors remarked in their paper that they did not have sufficient information "to conduct a real statistical analysis and bring a significant conclusion to the considered application."

Coetzee (1997) illustrated the selection process of a suitable technique with the diagram shown in Figure 7. Model selection for a given dataset takes place on the basis of tests for trend and dependence. If the data exhibits a trend, the non-homogeneous Poisson process (NHPP) is used. If no trend is found and the data is independent, renewal theory applies.

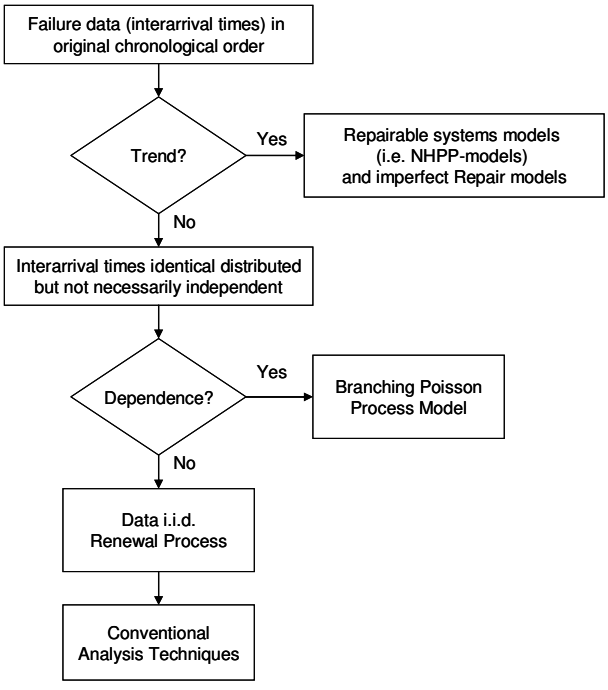

Figure 7: Data analysis framework as shown by Coetzee (1997).

1.4 Advanced Regression Models

Most of the traditional reliability studies are based solely on failure-time data where time is assumed to be the only variable. There are clearly shortfalls to such an approach, as conditions under which systems operate in practice are changeable and a number of factors exert an influence and contribute to failure. Time data is not sufficient to allow a model to distinguish between the alterations in reliability patterns attributable to each of these factors.

Guo, Love and Bradley's (1994) paper highlights the failings of the bad-as-old assumption, when maintenance and repair work is performed on a system. Very rarely is a system returned to the exact state in which it was before failure. Chan and Shaw (1993) model the failure rate of a repaired system by introducing a reduction in the failure rate after preventive maintenance. Similarly the proportional age reduction (PAR) model which was introduced by Malik (1979) is able to cater for the influence of maintenance action on the condition of the system through the introduction of virtual age which is dependant on maintenance. A maintenance action which improves the system's condition is therefore said to reduce its virtual age, while a worsening of the system's condition through maintenance intervention is said to increase the system's virtual age. System reliability henceforth is dictated by this virtual age, rather than the actual age which is used in the NHPP model.

The effect of repairs on the system's condition, changing its virtual age, can be illustrated by an example. We take a set of gears in a gearbox, where the pinion is damaged through wear. If only the damaged pinion is replaced, the effect on future reliability of the system will be notably different from the case where the complete set of gears is replaced. Running a worn gear together with a new pinion means that the system was not returned to GAN condition by the maintenance action. Renewal theory will not hold in such a situation, and the system will most probably deteriorate more rapidly than in the case where the complete set was replaced. The partial renewal of the gear set however implies that the system is in better condition than before the replacement, and the BAO assumption consequently also doesn't apply.

Venturing further on the subject of covariates, the variables that determine a particular system's reliability, the effects of failure and maintenance intervention on a repaired system deserves further discussion. The graph in Figure 8 shows the behaviour that would be observed on a simple system with units, for example light bulbs, that are replaced upon failure. The first set of units start their life together and fail according to a normal distribution. Not all failed units are replaced at the same time, however, thereby increasing the standard deviation of the distribution for the second set of failures. In time, the failure rate becomes constant as these distributions increasingly flatten out and the failures of different generations overlap. Such a constant failure rate means that the system exhibits a negative exponential failure distribution, which indicates that the probability of failure is independent of age and failures are occurring at random.

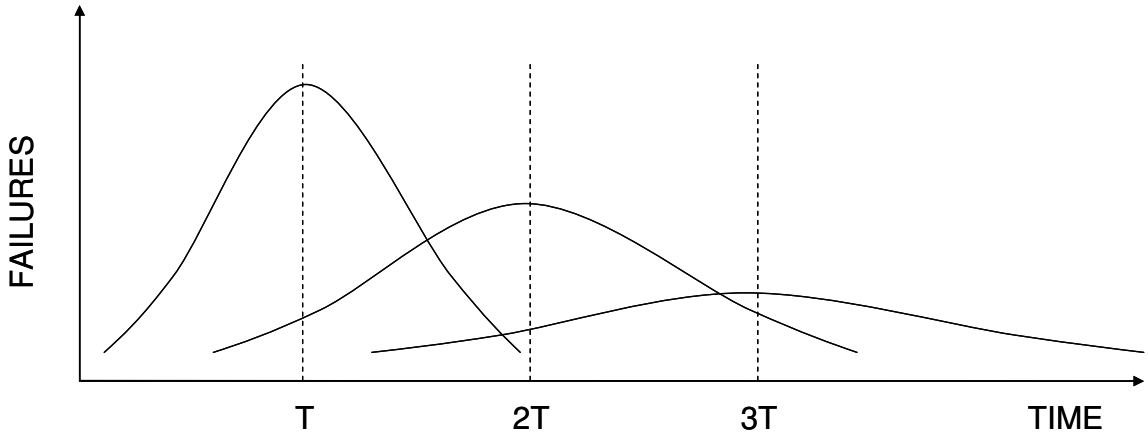

Figure 8: Change in the failure distribution over a number of generations. (Adapted from the original in Bradley (1993))

This tendency also applies to more complex systems. Bradley (1993) refers to research by Davis of Rand Corporation, who found that bus engines exhibit a constant failure rate after the second overhaul. When maintenance intervention takes place, it tends to cause unpredictability in future system reliability behaviour. Machine operation and maintenance efforts lead to a unique machine condition when individual systems diverge in terms of actual condition, as each one is influenced by its own individual history. The predictability of system reliability decreases with every intervention and it is found that it finally degenerates into random failure. This phenomenon makes use-based preventive maintenance action totally ineffective with time. Apart from maintenance intervention, and the resulting effect of virtual system age, the effects caused by operating the equipment also tends to affect reliability beyond the realms of mere time based analysis.

Condition based maintenance however will still be successful, even though a system has reached the point where it is failing at random. The actual system condition is determined from the measured data, giving warning of deterioration. The abovementioned shortcoming of use-based maintenance also reflects upon statistical models which do not incorporate the advantages and robustness gained through the use of condition based data.

As the probabilistic models discussed previously are clearly insufficient to cope with some of these factors influencing repaired system reliability, a number of regression models with different improvements have been developed. According to Kumar and Klefsjö (1994) the presence of covariates was generally ignored in the past, or the dataset was broken into a number of groups in order to accommodate the major differences. They suggest the use of such advanced regression models for estimating reliability characteristics due to the possibility that they accommodate the variances experienced in the real life situation by taking such covariates into account.

The advantage of incorporating covariates into a model is not limited to repaired systems. Even simple systems are subject to different operating conditions and may have several failure modes. Such data can no longer be accurately represented by one statistical distribution as the failure distribution curve is a composite of sub-distributions caused by the different failure modes. Figure 9 shows a number of different failure distributions which result from variability in maintenance, manufacture and operation to illustrate why a global model, solely based on failure

time data is not suitable to analyse such a system. The mean failure time of all the data when lumped together is substantially different to the mean of the data subject to certain conditions.

The suggested process of differentiating between data subject to factors of different nature and intensity can be done manually, as mentioned previously, or by means of a regression or neural computation method that uses covariates to make a distinction between the influences of these underlying factors. All the major significant covariate inputs are required to fully describe the system and give the mathematical model the opportunity to discriminate between different situations.

If a curve fit is therefore made to the overall distribution, a great deal of accuracy is lost in the process. By using a model that allows covariates to be taken into account, the incorporation of condition monitoring information into the analysis makes it possible to differentiate between various failure modes. A more accurate solution therefore becomes possible when calculating residual life.

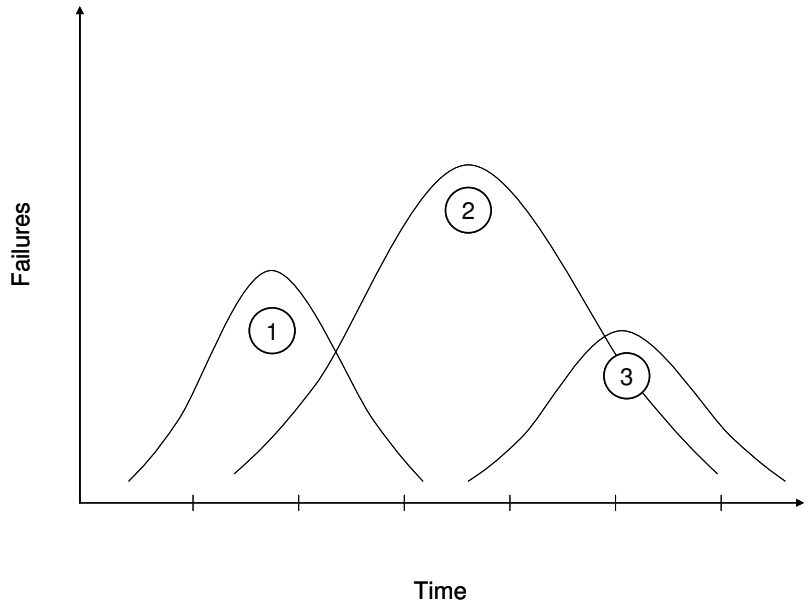

Figure 9: Illustration of different failure distributions caused by variation in the factors that influence a system.

The simplest way of quantifying the influences on a system or component, is through the measurement of various condition related or operational parameters. With modern condition monitoring methods, it is possible to establish the actual state of equipment and could be used to provide the level of degradation as a function of time. System failures can be attributed to various failure modes which can be traced back to a characteristic underlying degradation process. A specific mode of failure is also often induced by certain conditions under which the machine operates and the way in which the equipment is used. Such operational influences and environmental factors be measured, the model can use this information to differentiate between failure modes and different rates of system degradation.

As an added advantage, condition monitoring data provides insight into the degree of repair on a system. Additional variables are introduced when a failure is experienced by a repairable system and components are replaced. The next failure of the system depends on the state of the system before the current failure and the nature of the

repair work that was done. It is possible to quantify the quality of the repair and the condition of the system by using the variety of condition monitoring procedures that are available to maintenance staff. Such measurements can serve as an explanatory variable in a reliability model. It could be used to associate a particular set of readings with measurements done in the past to provide a more accurate estimate the influence on remaining life.

Regression models form a combination of the probabilistic and non-probabilistic approaches to reliability analysis. An underlying failure distribution is not the primary assumption, as is the case for probabilistic methods. Regression models utilise the hazard function, thereby resembling non-probabilistic methods. In addition, time is not the only parameter that is modelled. Provision is also made for the incorporation of covariates related to the failure event into the model. A brief discussion of such advanced models is included below.

1.4.1 Multiplicative Intensity Models

In multiplicative intensity models (illustrated in Figure 10), covariates have a multiplicative effect on the FOM and ROCOF, whichever applies. The intensity of a failure process is represented as a product of a baseline intensity, which is a function of time, and a functional term, which may be a function of time and covariates. Thus the effect of the covariates is to scale the baseline intensity up or down.

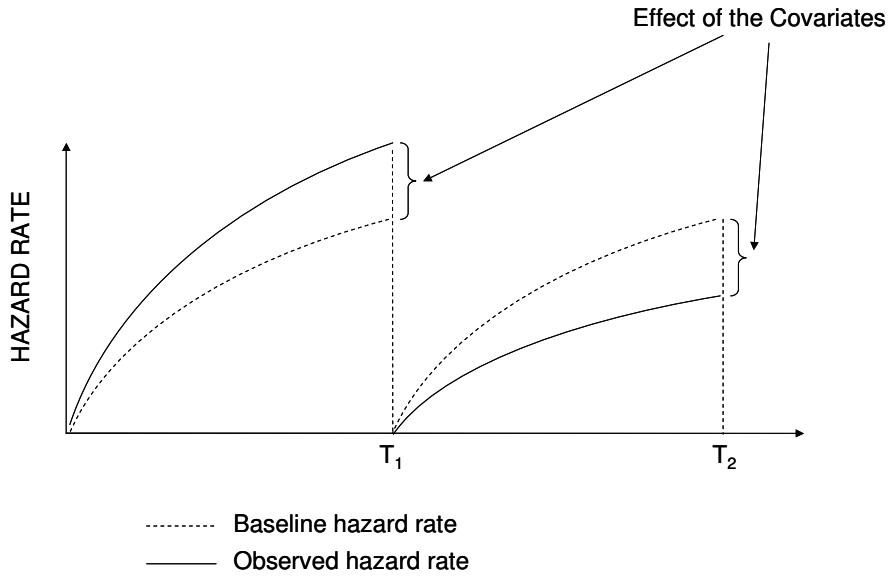

Figure 10: Illustration of a Multiplicative Intensity Model (Source: Kumar and Klefsjö (1994))

Cox (1972) introduced the proportional hazards model (PHM) in which the covariates act in a multiplicative way on a baseline hazard function such that the hazard functions are proportional to each other over time for different values of the covariates. The PHM has been approached in two ways. A parametric approach builds on the renewal model by using the Weibull, log-normal and gamma distributions as a baseline. The approach introduced by Cox, however, does not assume a particular form for the baseline hazard function. The form of the hazard function is estimated from the data. Cox used an exponential term which incorporates the effects of the covariates. The use of this latter form of the proportional hazards model has gained

much popularity because no assumption is required on the form of the hazard function (FOM).

Dale (1985) provides examples where the PHM is applied to both repairable and non-repairable systems. Advantages of this method are amongst others that the analysis can be done despite the presence of nuisance factors and that the importance of various factors is estimated as part of the process. Censoring, tied values and zero values can be easily accommodated through the flexibility of the model. No assumptions are required about the shape of the baseline function or the distribution of times to failure. Due to the assumptions of the model, the application of PHM for repaired systems in the maintenance field is restricted to the good-as-new case.

The Proportional Mean Intensity Model is similar to the PHM, but uses a baseline ROCOF instead of the FOM. This makes it suitable for repairable applications where the GAN assumption does not hold. The proportional odds model (POM) has a similar structure to the PHM and was introduced by Bennet (1983) for use in biomedicine. It models the odds of an event occurring, and unlike the PHM, the effect that the covariates have, reduces as time approaches infinity. POM is therefore suitable for use in situations where the influence that outside factors exert on the system is only pronounced in the early stages of machine life. This property has however also limited its practical applicability.

1.4.2 Additive Hazard/Intensity Models

Proposed by Pijnenburg in 1991, these additive models differ from multiplicative models discussed above, because the covariates have an additive effect on the FOM or ROCOF. Figure 11 illustrates how a functional term is added to a time related hazard rate serving as a baseline function.

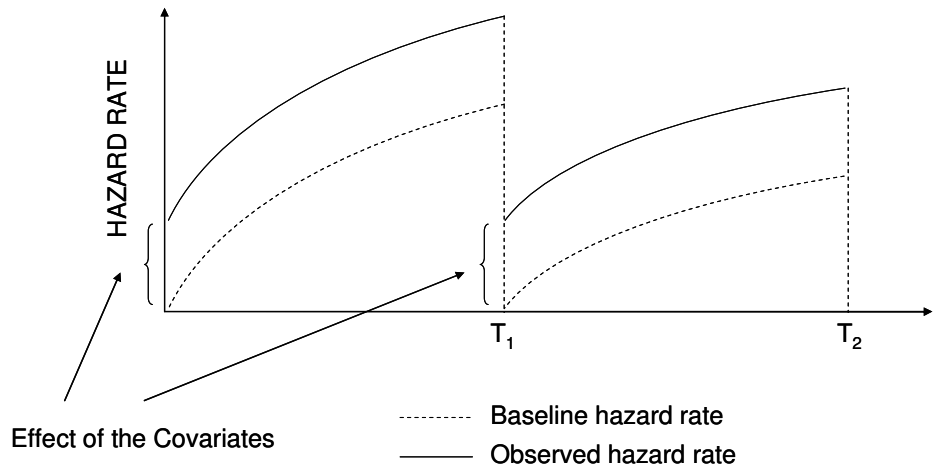

Figure 11: Illustration of an additive intensity model. (Adapted from the original found in Pijnenburg (1991))

1.4.3 Models with Mixed or Modified Timescales

The mixed or modified time scale acts as an additional covariate in the data set, providing additional flexibility. The modelled intensity of a failure process is increased or decreased by accelerating or decelerating the aging of an item.

Prentice, Williams and Peterson (1981) introduced the Prentice Williams Peterson (PWP) model which is an extension of the PHM to cater for systems where multiple

failures can occur. Ascher and Feingold (1984) emphasize the importance of the PWP model citing that it is the first model with the capability to handle most of the 18 real world factors which they had identified. It allows custom tailoring and is especially useful for datasets of limited size.

First introduced by Pike (1966), in the accelerated failure time model (AFTM) the covariates serve to accelerate or decelerate the predicted arrival of failures, allowing for circumstantial influences around the failure. Martorell, Munoz and Serradell (1996) introduced the proportional age setback (PAS) approach, where the effect of each maintenance action is assumed to shift the origin of time from where the age of the component is evaluated. The virtual age in question is based on the entire history.

Just like the PAS model, the proportional age reduction (PAR) model which was introduced by Malik (1979) and allows for the effect of maintenance to be taken into account. The maintenance action reduces the virtual age of the unit in the view of rate of occurrences of failures. It is presumed that the maintenance intervention improves the state of a repairable system by some degree which is represented by a rejuvenation factor.

The PAR model differs from the PAS approach mentioned previously, because the virtual age is only based on the survival time of the most recent lifetime. Malik's approach assumes that maintenance intervention reduces the operating time elapsed from the previous maintenance proportionally. According to Shin et al. (1996) who applied the PAR model, it is a natural generalization of the GAN and BAO assumptions. Depending on the choice of the rejuvenation factor it can cater for different levels of imperfect maintenance. As the actual effect of maintenance would differ for each individual case even though the work is performed on an identical unit, the rejuvenation factor which is used in this model is only an average over the observed period. The PAR model has been found useful in the development optimal maintenance policies because of its flexibility.

1.5 Combined Advanced Failure Intensity Models

The models briefly described above generally only cater for one type of modification with respect to reliability related predictions. Covariates are used to estimate alterations in virtual age or act to modify the FOM or ROCOF in a multiplicative or additive fashion. The situation encountered in industry is clearly more complex than can be described by only one such model, as numerous factors may exert an influence in differing ways. Even when a regression model with stratification is used which allows the separation of the effect introduced by different factors, the implementation may not be successful. The effect that these factors have on reliability may require the use of different models. Vlok (2001) identified a need to include more than one of the conventional enhancements in the same generic model and he therefore proposed a combination of advanced regression techniques, where more than one improvement is incorporated into the model.

The effort involved in manipulating data, the estimation of coefficients and refinement algorithms for any single model meant that only one type of model is normally used for modelling of data sets. The results are accepted without comparison with the performance achieved with other types of regression models. Vlok intended to find the most suitable enhancements for a particular problem. The two distinct

generic models that were developed by him are applicable respectively to the non-repairable, for the estimation of FOM, and repairable cases, for the estimation of ROCOF.

Most models only consider relative risks, as no assumptions have to be made about the underlying baseline function. Relative risks unfortunately do not provide any information with regards to absolute probabilities. Vlok however needed to determine absolute risks to enable him to estimate residual life, which was one of his aims. His models therefore had to be fully parametric to enable him to calculate the absolute risks required for his residual life calculations. To serve as a parametric baseline for these models, Vlok chose the Weibull distribution for the non-repairable case, while the log-linear NHPP was chosen for the repairable case.

Both the models include the features of frailty, a time jump/setback, time acceleration/deceleration, a multiplicative term and an additive functional term. Vlok notes in his work that the application of these generic models in their complete form is unrealistic due to data constraints encountered in practice. The models are however intended to provide a basis for the derivation of simpler models with more than one enhancement.

As covariates for these models, he decided to utilize condition monitoring data, in the form of vibration readings. Expressions were developed to calculate the parameters of the models by using the maximum likelihood method. Solution for these parameters then takes place by means of numerical optimization techniques.

The use of more than one enhancement by Vlok in his generic model illustrates the need for a method that can incorporate the effect of different factors influencing reliability, without the increasing computational complexity of regression models that have sufficient flexibility to model the real life situation.

1.6 Residual Life Estimation

When speaking of residual life, the difference between the concepts of renewal theory and repairable systems which are not returned to the GAN condition upon repair must be kept in mind. A component or system's life, when renewal theory holds, ends once it is discarded or repaired. This action may be prompted either be the need for preventing failure or as a direct result of failure. A system that is repaired to the GAN state can be theoretically described as a new system, because it is returned to a condition similar to that of a new unit. For repaired systems which do not conform to renewal theory, the situation is more complex. System life has to be dealt with alongside the concept of component life. The life of the system is dictated by safety and economics and it is discarded once it becomes expensive and unsafe to operate and uneconomic to repair. The system may therefore fail a number of times and a significant portion of its components may be replaced during its life.

Reinertsen (1996) studied a wide range of research material on work done in the field of residual life estimation, especially with respect to systems. His focus was the saving of cost by accurate prediction of residual life and extension of equipment life by considering factors influencing the life of equipment. A number of aspects are touched on, one of them being the use of neural networks by Lee and Kramer (1993) for pattern recognition and fault diagnosis.

It is noted by Reinertsen in his paper, that surprisingly little work has been done in the field of residual life estimation for repairable systems. In his opinion the differences

between repairable and non-repairable systems is not generally understood, thereby leading to false conclusions. This situation has been discussed in greater detail in previous sections. Reinertsen found that a generous number of papers have explored the estimation of residual life for non-repairable systems through statistical methods, but no corresponding work exists for repairable systems.

Reinertsen proposed that a study of life estimation techniques used on natural systems, such as human beings, should be conducted to see if these methods would also prove useful when applied to technical systems. A human body, for example, could be seen as a repairable system in which some of the components can be replaced if they cease to function. A study of research work in life estimation for natural systems could therefore compensate for the lack of publications on technical systems.

Among the research using condition monitoring data for residual life estimation is the work of Jantunen (2003) who fitted a polynomial curve to vibration data, Vlok (2001) who used regression curves and vibration data to estimate residual life of pumps and Wang and Zhang (2005) who used spectrographic oil analysis data to predict the residual life of aircraft engines.

Wang and Zhang (2005) note in their paper that the management decision making aspect of condition monitoring is a neglected field. A new methodology was adopted by them which used all past information thereby having the recursive nature of the filtering model while treating the oil data as a covariate. The right censored life data of 30 aircraft engines, with the corresponding monitored metal concentrations, was used for the research. They reduced the data to one variable through the application of principal component analysis, enabling them to fit a Weibull distribution. The curve fit was done with 21 of the 30 sets, leaving the remaining 9 sets to test the goodness of fit.

1.7 Neural networks for Reliability Data Analysis.

Research has been done to investigate the use of neural networks in maintenance and reliability related applications. A wide variety of methods, network architectures and data combinations were used in these cases and it is of interest to discuss some of the more relevant examples.

Liu, Kuo and Sastri (1995) used neural networks to identify the underlying distribution of a set of failure data. The set of data was classified as a normal, uniform, exponential or Weibull distribution. Parameter estimation of a two-parameter Weibull distribution was also performed with a neural network.

Velten, Reinicke and Friedrich (2000) investigated the use of artificial neural networks for predicting and analysing the wear behaviour of short fibre reinforced polymeric bearing materials. Regularization was used to prevent overfitting during training, which was done with the Levenberg-Marquardt training algorithm.

Amjady and Ehsan (1999) evaluated the reliability of power systems using an expert system based on neural networks. They used an architecture consisting of a number of parallel subnets, each being a multi-layer perceptron network. The training samples are distributed amongst the various parallel subnets. Very good results are obtained despite the limited number of the training sets. The two separate evaluations that were conducted looked at the generation systems and the transmission systems. Four parallel neural networks were trained to estimate various maintenance parameters,

such as scheduled maintenance time, mean time to failure (MTTF), mean time to repair (MTTR) and the forced outage rate for the generation systems. Among the inputs to the expert system were the generating unit type, the number of generating units and the generating unit output. Six subnets were required for the transmission systems expert system. The outputs to the six parallel networks were the transient outage rate, permanent outage rate, permanent outage duration, normal rating, long term rating and short term rating of transmission components. Voltage and type of transmission equipment served as inputs to these subnets.

Xu et al. (2003) tried to forecast reliability by using neural network techniques to analyse the past historical failure data information. They state that the use of neural networks for failure data analysis offers an advantage over traditional methods, as no prior assumptions are required where the parameters of the nonlinear models need to be determined. The model parameters are usually hidden within the network and are optimized through iterative adjustment in a learning process based on historical patterns. It was observed that the use of RBF neural networks should offer an advantage over traditional MLP networks due to shorter training time. Consequently, two datasets were used by the authors to train MLP and RBF neural networks to test the comparative performance of these architectures. The results were also tested against a linear benchmark that was based on the Box-Jenkins autoregressive-integrated-moving-average (ARIMA) models. The two datasets consisted of failure data of diesel engine turbochargers and car engines. The results obtained indicate that the RBF network outperformed both the MLP and the linear benchmark. It must be noted that it appears as if the failures were taken as the first failure for each unit. The study therefore does not cater for repaired systems.

Luxhøj and Shyur (1995) compared the performance of traditional reliability modelling techniques with neural networks for the fitting of a reliability curve to data of helicopter components. The Weibull distribution was used to model the data, with parameter estimation done with the maximum likelihood methods. The only input to the neural network was the part hours of the components, while the output from the network is a reliability value. The network can therefore be used to generate the reliability curve. The researchers employed a MLP neural network trained with a backpropagation algorithm with momentum. The network received one input, which was part hours, had five hidden nodes and one output node that generated a point on the reliability curve. Only one of the datasets was large enough to enable the researchers to break it up into two sets. The first set is used for the training of the network, while the second was set aside to act as a “test” set. The “test” set is used to check how well the network generalizes when it has to solve for data on which it has not seen before (see the section on generalization in Chapter 2). When the network has overfitted it will give inaccurate results between training data points. The dataset in this case consisted of 32 datapoints, which were broken into 26 “training” and 6 “test” points. Upon testing the results, the R^2 values were found to be 0.9418 for the training set and 0.9291 for the test. The authors described this result as a “robust” fit. The conclusion drawn by the research was that the results obtained from using a neural network to do curve fitting of reliability curves compares favourably with the present mathematical methods.

In his paper, Luxhøj (1999) researched the prospect of providing FAA safety inspectors with a means to evaluate and control the appropriate surveillance levels for aircraft operators through the application, amongst others, of neural networks. In his paper the performance of multiple regression and different neural network variations

are compared. The network types discussed by the author included both Multi Layer Perceptron networks (MLP) and General Regression Neural Networks (GRNN), which are a form of Radial Basis Function (RBF) network. The GRNN is a two-layer network that contains a neuron for every training pattern. Training occurs in one pass. This type of network performs well on problems which require continuous function approximation. An added advantage, according to Luxhøj, is the GRNN's ability to train on sparse data sets, which was the case. The use of a hybrid neural network was also investigated and it achieved improved levels of performance when compared to the standard networks. This hybrid network consisted of a Probabilistic Neural Network (PNN, also a form of RBF network), which was used to classify input data into groups. The results were then fed to a backpropagation neural network, which then solved for the problem.

Luxhøj used such a grouping strategy to improve the robustness of his prediction models. The available data was grouped according to age, and average flight hour and aircraft operability values were then calculated for each group. He concluded that the neural network models were easier to deal with and tended to give better results if the data pattern did not change drastically during the period of study. The classical models however remain more understandable and can be explained more easily, according to him.

Liang, Xu and Shun (2000) applied MLP neural networks to the field of condition monitoring. They put forward a method for determining the optimal number of nodes in the hidden layer of such networks. The standard MLP architecture was trained with a backpropagation algorithm using a momentum term. Unfortunately very little information is given regarding the dataset used by them, as the paper focuses on the theoretical aspects of improving network performance.

The estimation of residual life of systems and components is dependent on the utilization of various information sources that relate to factors which exert an influence. Neural networks allow such data to be taken into account in a far more efficient way than traditional methods, thereby making analysis possible even for complicated systems. One of the advantages offered by neural networks over other methods in the application to the reliability problem is that no initial assumptions have to be made about the nature of the data. Where researchers using the traditional statistically based methods, such as Vlok (1991), have been forced to reduce the number of inputs substantially in order to simplify the computation of parameters, the neural networks used for this project were shown to handle numerous inputs with ease.

1.8 Reliability Data Considerations

In the previous paragraphs the analysis of reliability data and the making of residual life predictions have come under scrutiny. It remains to discuss a number of issues, some of which are raised by Crowder et al. (1991) and Meeker and Escobar (1998), concerning reliability data, its form, and the way it is used affect the successful practical application of reliability data analysis methods. These data issues will subsequently be addressed in view of the use of neural networks for such analysis.

1.8.1 Failure

Failures may be classified according to cause, suddenness and degree. The modelling technique will be chosen in order to suit the nature of the problem. Not all models are

equally successful or suitable in dealing with all forms of data. A model that works to a clearly defined failure event will prove inaccurate if the system exhibits a gradual deterioration accompanied by a loss of performance. In some cases it may even be beneficial to work with probabilities.

Neural networks may have a number of outputs, allowing the analysis to be tailored to the given situation, depending on the analyst's objectives. It may be necessary to structure the outputs in such a fashion as to reflect such a loss of performance as has been mentioned. It may also be useful to obtain confidence intervals on a residual life estimate. The analyst may choose to receive an output reflecting a damage factor. The importance of such flexibility is especially placed into perspective when looking at two aspects that are directly linked to the manner in which failure is defined. These are the subject of censored data and of systems with redundancy.

1.8.2 Censored Data

Components may never reach actual failure and it often happens in practice that units in various states of degradation are replaced to prevent failure. Censored data is the result when a definitive starting and/or end point was not present during data collection. If the gathering of data starts at some point during a machine's life, it is said to be left censored. It may also happen that the data is processed at some point when some of the equipment has not yet failed. Such data is said to be right censored. A combination of these two conditions is known as interval censored.

A reliability data analysis method that can be applied in the real world must preferably be able to deal with such censored data. If this is not possible, this data must be adapted or eliminated during pre-processing. The use of condition monitoring data assists in limiting the effect of censored data, as the increased number of covariates makes the analysis method more independent of the time variable. It also allows the adjustment of the outputs while considering condition, and not simply an arbitrarily assigned state.

1.8.3 Series and Parallel Systems

The choice of method for the analysis of reliability data is greatly dependent on the nature of the system or component in question. For actual systems, the definition of failed versus functional is often difficult to make as it may still operate in various states below the optimum performance. Such a system is called a multi-state system. In a series system all the components must work for the system to be functional, while a parallel system may continue to operate despite partial failure of the system. Such a parallel system is said to have redundancy if a number of parallel units perform the same function and functionality is maintained despite the failure of some units. A parallel system may therefore be an example of a multi-state system if it can be found in an intermediate state, operating at reduced levels when one or more of its units are not in operation.

In load-sharing systems, the load is distributed among components. If failure occurs, the load is redistributed amongst the remaining components. Clearly the operating conditions have changed as the remaining components will now carry a greater load, are more highly stressed than before and will be subject to more rapid degradation. When systems have numerous components there is the possibility of simultaneous failures and also secondary failures. Reliability analysis may also have to deal with

the reliability of individual components, sub-assemblies or the behaviour of the overall system.

1.8.4 System Complexity

The complexity of a system defines the number of states in which it can be found. Let us say for simplicity sake that each component of a system can be in one of three states – good, moderate and bad. If such a system consists of *n* components that have a significant influence on reliability and can contribute to overall system failure, then there can be 3^n different combinations of component condition.

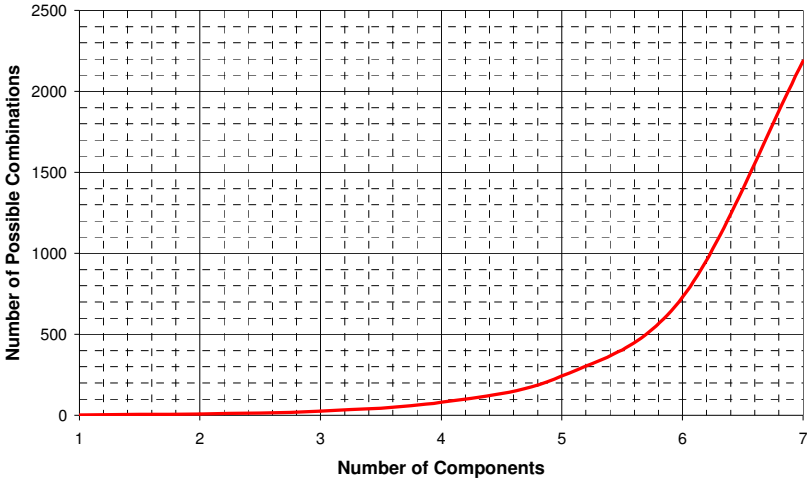

Figure 12: Graph illustrating the exponential growth in complexity as the number of significant components increases.

Looking at the overall system, the correct definition of condition corresponds to one of these 3^n states and must be identified, as each combination will be responsible for different system reliability behaviour. The complexity of a repairable systems problem therefore increases exponentially as the number of significant components increases.

The consideration of all factors that influence the reliability of a system, serves to complicate the solution of the problem. The more factors that have a significant influence on the system's health, the more complex the underlying dynamics of the system, and the greater the size of dataset required to accurately describe the system's behaviour. It is not hard to imagine therefore why so little work has been done on reliability analysis for repairable systems. Artificial neural network methods offer an advantage over existing techniques, as they require less computational effort from the analyst when faced with such a non-linear problem.

1.8.5 Comparability

The inputs of a neural network assist the network's task of distinguishing between different reliability cases in two ways. Firstly they can facilitate the network's ability to differentiate between units subject to different conditions and secondly they may help with the identification of data trends pertaining to the life of each specific unit. Components and systems in practice have different ages, are maintained in different ways, operate under different atmospheric conditions and are found in different states of modification. The use of absolute inputs therefore may adversely affect a neural

network or other analysis method's ability to generalize. In the search for a generally applicable method, the inputs of an analysis method followed must filter out the data elements that do not have relevance in the context of failure and will therefore generate confusion. Comparison becomes possible once the data from different sources can be placed on equal terms.

1.8.6 Time and Age

In the non-repairable case everything restarts from zero when a failed unit is replaced. The order in which failures occur therefore has no significance. This is not true for repaired systems where the renewal theory does not hold. The incorrect use of models with regards to local and global time has already been elaborated upon. A difficulty that is encountered with repairable systems and residual life estimation is the fact that the analyst has to work both with cumulative time and also the time since the last failure.

One repaired system generates only one set of data, especially if the analysis requires sequential information, as is the case when global time is used as an input. This places an additional burden by increasing the quantity of data that is required to achieve the desired accuracy and it must also be noted that variances in the sequence of events for different units generates totally unique data in each case.

The absolute age of a system is a function of the age of its constituent components. Components exert an influence on each other due to their interaction in the functioning of the system. It is therefore very rare that two systems can be found in exactly the same state and experiencing the same sequence of maintenance events. To this complexity is added the variations due to different failure modes. The exponential increase in combinations of condition has already been dealt with in the section on complexity. Describing a system of such complexity with any model becomes extremely difficult. Very large datasets become a necessity to allow all possibilities to be completely described and to make sure true generalization is achieved.

Table 1.1 gives an example two hypothetical cases are given as follows:

Table 1.1: Example comparing two hypothetical cases.

Description	Case 1	Case 2
Absolute Life	8000 hours	8000 hours
Previous Failures	3	1
Last Failure	6000 hours	6000 hours

The only way to distinguish between these two systems is through the use of condition monitoring data. The problem causing the additional failures in case 1, possibly misalignment, may have been sorted out. These failures should however have an effect on future life of the equipment due to a number of factors that affect each system.

On the negative side, case 1 may have suffered severe degradation due to the effects of the unexpected failures. On the other hand, it may have been completely refurbished due to these failures and therefore be running as a technically "younger" system than case 2.

It can therefore be seen that a repairable system's condition becomes very difficult to quantify as it ages, and that only methods which can use data relating to the current condition of the system have a chance of success.

1.8.7 Cumulative Damage and Degree of Repair

From the previous paragraphs it may be deduced, that although systems are very similar upon delivery, they become increasingly unique with time. The discussion on statistical methods has illustrated the fact that the effects resulting from failure, repair and maintenance action have only been taken into account in a general fashion, if at all.

The PAR regression model attempts to cater for imperfect repair, with a rejuvenation factor placing the result somewhere between the GAN and BAO assumptions. Sin, Lim and Lie (1996) however correctly stated in their paper that, "In practice, the effect of maintenance may not be uniform even though the maintenance is performed on an identical unit." They concluded that the improvement factor should be interpreted as an average effect over the observed period. These methods can be said to rely on a comparison with historic data from other systems that produced a certain chain of events. It may therefore not provide an accurate representation for predicting future events on other units. It does not take into account the uniqueness of each specific failure event, repair job and the resulting variance in system rejuvenation.

It is very difficult to describe a repaired system simply in terms of the time variable. The system deteriorates over time and the conditions and loading under which it operates are largely unique. The question of how the events in the system's history affect its reliability must therefore be quantified.

The advantage of using condition monitoring data is that it provides insight into actual system condition without the requirement of subjective input or a full set of historic data. It therefore compensates for the unknown damage caused by measuring the effect of the damage. Actual wear data from oil analysis or peaks on a vibration spectrum will allow the damage to be quantified. Actual measured condition data will probably prove to be the easiest way to differentiate between systems in a different health condition.

This also holds true for damage resulting from failure. The nature of the shutdown affects the quality of the dataset for use in network training. If the machine is stopped for a preventive replacement, equipment life is shortened when compared to a situation where the equipment is run to actual failure. The issue here is the definition of failure as both cases are different definitions for failure.

1.8.8 Significant Covariates

A system's performance and reliability is generally affected by a number of factors. Network inputs must offer full coverage of these factors that are linked to equipment failure and component life. If such a system is to be accurately represented by means of a neural network, it can be assumed that the more of these variables are presented during network training, the more accurate the result. The information given to the

neural network must be of such a nature that it can be used to identify and quantify system degradation for all the main failure modes. In other words, the data that serves as input to the network must enable the neural network to differentiate between these main modes of failure.

An arbitrary function was chosen for the purpose of illustrating the effect that comprehensiveness of input information has on the performance of a neural network. A function with four variables was used to generate outputs by assigning randomly generated values to each variable within a range from 0 to 1. The function outputs also range between zero and one.

$$f(k,l,m,n) = \frac{5 \cdot k + 3 \cdot l + 4 \cdot m + 2 \cdot n}{10} \qquad (6.1)$$

A neural network was trained with two input variables (k,l), three variables (k,l,m) and all four variables (k,l,m,n). Upon the completion of each network's training run, the error for the training data points was calculated. The results of this test were then sorted, graphed and are shown below. The error values are ranked from smallest to largest on Figure 13 in order to better illustrate network performance.

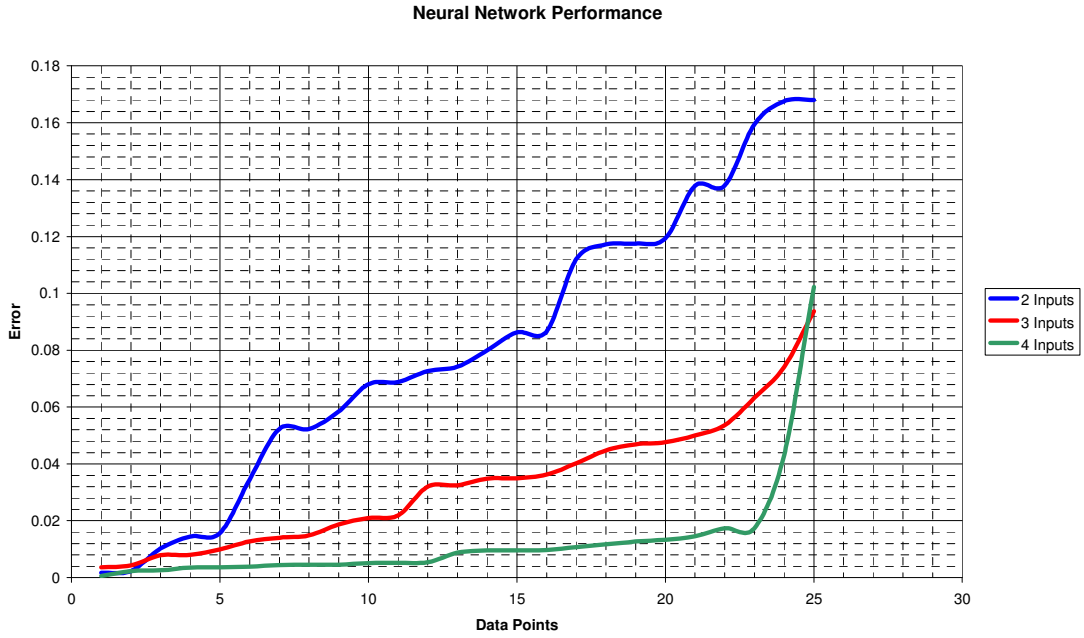

Figure 13: Comparison of the fit obtained with a different proportion of the total variables known to the network.

Each variable exerts an influence on the function's output. As expected, the ability of the network to replicate the training function improves dramatically as more of the relevant input variables become known.

The behaviour in a real life situation would unquestionably be of a more complex nature as a number of such functions could be interacting as part of a non-linear system. The experiment however still serves to illustrate the concept under discussion.

From the simple example, it can be seen that neural network performance is greatly dependent on the completeness of input information. If the inputs given to the network are indicative of the underlying mechanisms that influence the outputs, then the neural network will function well. A network can only be expected to offer a solution for failure modes that it has seen and can be clearly identified by the pattern of inputs. If the input data is outside the training range, the solution may not be satisfactory.

If all the most significant failure modes can be predicted by analyzing the data pattern generated by environmental factors and condition monitoring measurements, then the application of a neural network should be very successful. Caution is however advisable. Too many inputs may mean that the network not only splits between significant categories, but also within categories. The result is overfitting during training, accompanied by loss of the network's ability to generalize.

1.8.9 Data Integrity

Numerous datasets were investigated for use in this research project and they were invariably found to be of limited value. The main reason is a lack of data integrity, which is a critical problem when it comes to evaluating reliability data.

Among the problems experienced with datasets are readings that are skipped, variances in measurement method, vital information that is omitted from data entries and records that are incomplete. The reconstruction of events at each failure event is a difficult and time consuming task, and the tracking of repairable modules such as gearboxes which are reconditioned is sometimes almost impossible. The situation is exacerbated when the recording process lacks discipline. It was found that in some instances the serial numbers of units were removed during an overhaul or incorrectly documented, making analysis of such data well nigh impossible.

Due to the competitive environment in which they operate, condition monitoring companies were found to be reluctant to share their readings and it proved difficult, in some cases, to extract the collected data from specialised condition monitoring software packages. The reliability data is in many cases only available as a hand written hard copy and requires days of wading through paperwork on site to record the relevant information. If the equipment user switches from one service provider to another, continuity is more often than not completely lost.

Aggravating the situation for the analyst, condition monitoring has a low priority with many companies and the persons assigned with the responsibility for taking the readings do not have an active interest, adequate training and sufficient understanding for the task that they are performing. The same is also frequently true for the maintenance manager, who does not have the ability to perform analysis of the data and assigns it a low priority when compared to other tasks linked directly to the pressures of production.

1.9 Scope and Contribution

The benefits of reliable information to the maintenance manager are unquestionable. Accurate residual life estimates offer a number of benefits to tactical maintenance planning, apart from the selection of an optimal replacement strategy and the flexibility offered to the maintenance manager. Maintenance action is usually limited by production requirements, and inspections, repairs and replacement of components

must take place during periods where the plant is shut down and made available for maintenance. Advanced knowledge of pending maintenance action means that spares can be ordered and resources can be allocated. Items with a high delivery lead time are available when required; preventing costly delays and ensuring that they do not have to be kept in stock at all times to ensure minimum production losses should an unexpected failure occur. Careful planning also ensures that unusual peak loads of maintenance work can be accommodated and that the manpower and equipment for the execution of specialized jobs is in place when required. It can be concluded that planning can be optimized and resources can be employed in the most efficient and cost effective fashion, if suitable reliability related information is available.

The aim of this project is to investigate the feasibility of making reliability predictions based on failure data and the data collected through condition monitoring. The residual life estimation proposed is not merely a form of condition monitoring, but attempts longer term predictions to help the maintenance manager with his tactical maintenance decision making. Such decisions have been made in the past on the basis of information gained through the use of parametric models.

From the previous discussion it can be seen that regression models have been found to offer a significant advantage over parametric models due to their ability to take information relating to the failure of a system into account. Condition monitoring data could therefore be used to contribute to the improvement of accuracy of estimates through incorporation into such models.

A case has been developed in this chapter for investigating the use of neural networks in conjunction with condition monitoring data for failure prediction. Though a number of papers have been published on the subject, the use of neural networks to make predictions on the basis of reliability data has not yet been fully explored. As is the case with regression models, neural network methodology allows covariates to be used in such a model which is expected to improve the accuracy of failure predictions. Artificial neural networks offer numerous information processing advantages over the methods to which the reader has already been introduced. Their robustness, adaptivity, distributed and massive parallelism and their ease of implementation would make them ideally suited to implementation in the management of maintenance operations.

The work of Vlok (2001) showed that combined regression models can be used to model the reliability of a system while taking the effect of a number of covariates into consideration. Various modifiers are needed in such a regression model to ensure that the impact of the each covariate is correctly modelled. Vlok (2001) found that the optimal combination of modifiers must be found through experimentation, to ensure that the complexity of the model does not make it impractical. This process becomes increasingly difficult as the number of covariates increases, which led Vlok to discard all but the two most significant covariates from his models. In contrast, the effect of covariates is accommodated during the training process of neural networks with the result that it is easy for the analyst to introduce additional covariates. This is likely to improve the accuracy of residual life predictions, especially if an optimal input combination is found. The advances in computer technology have increased the rapidity of neural network training, making it feasible to implement changes and test different input combinations. This will be shown in later sections. Less emphasis is also placed on prior assumptions as no baseline is required when training the neural networks.

It therefore appears to be easier to train a neural network than to set up a combined regression model as was done by Vlok. Especially the much greater complexity encountered with repaired systems strongly favours the use of neural networks above regression models.

As a point of departure for this investigation, the theoretical background of Neural Network methods and literature was researched and the findings are summarized in Chapter 2. Based on this research the neural network architectures and training methods were chosen for comparison.

Two general system types have been identified which require a separate approach as they are maintained differently. Neural networks were therefore applied separately to the data collected from renewal and repaired systems.

The first dataset is an example of a renewal case. A series of tests was planned and executed at the Sasol Laboratory at the University of Pretoria to simulate a situation that might be similar to one encountered in practice. The lab testing procedure is described in greater detail in Chapter 3 and the results obtained with various neural networks are detailed in Chapter 4. Condition based measurements were taken on the test pieces and these covariates were used to improve the different networks' ability to accurately predict residual life. A number of neural network variations were trained with the laboratory generated data and their performance was compared by testing the accuracy of their predictions on a separate test set that had not been used for training the network.

The work done with the second dataset is described in Chapter 5. It was collected by Vlok (1999, 2001) and represented data collected from a number of pumps in the mining environment. This dataset was used to test the suitability of neural networks for making failure predictions for repaired systems. A comparison was made between the results achieved with neural networks and those obtained by Vlok through the use of various regression techniques. Further work involved the comparison of the performance of different neural networks by means of cross-validation, which was the most suitable method for such a sparse dataset.

The use of neural network methods was therefore investigated for the making of failure predictions for both the renewal and repaired cases. Inputs based on condition monitoring data were used as explanatory variables and their effect in the improvement of predictions was investigated. Based on these results conclusions could be drawn in Chapter 6 on the suitability of using neural networks in conjunction with condition monitoring data for reliability predictions as part of the tactical planning of the maintenance practitioner.

Chapter 2: Neural Networks

2.1 Introduction

A neural network is an artificial intelligence tool that works in a similar way to the human brain, which served as its inspiration. The construction and architecture of these networks have copied the functioning of the human brain and its capacity to learn.

The basic building block of neural networks is called the neuron, which is modelled on the synapses of the brain. It is a simple information processing unit whose activation depends on the nature of the input signals that it receives. Though the neuron has a number of input signals, it produces a single output signal upon activation, which is then transmitted to the other neurons to which it is linked. Training of such a network of neurons may involve the addition of new units, or the strengthening or weakening of these links. The combination of such simple units functioning in parallel makes such a network a powerful non-linear information-processing system.

Neural networks have been trained to perform complex analysis in various fields of application ranging from pattern recognition, identification, and classification to control systems. They are used today to solve a variety of problems that provide difficulty for conventional computers or human beings.

The neuron concept that was pioneered by McCulloch and Pitts (1943) still forms the basis of most neural networks. Figure 14, taken from Negnevitsky (2002) shows a neuron as could be found in multi-layer perceptron networks. The neuron computes the weighted sum of the input signals and evaluates a transfer function.

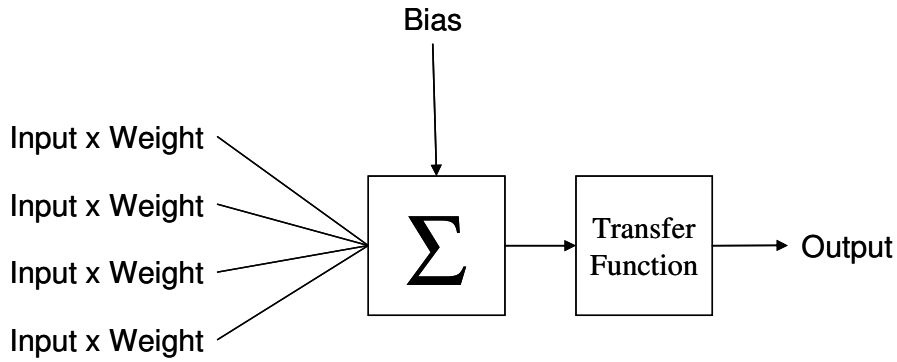

Figure 14: Diagram taken from Negnevitsky (2002) showing a neuron.

As is stated previously, a neural network is constructed by linking of a number of these simple elements together, so that they operate in parallel. The functionality of the network is dictated to a large extent by the connections between the elements. These connections have a numerical weight associated with them, and these weights form the basic means of long-term memory in the network. The weights allocate an importance to each neuron input and the learning process of a neural network takes

place through repeated adjustment of these weights. The weights and biases of a neural network are adjusted through a training algorithm (see Figure 15) so that a particular input leads to a specific target output. The adjustments are based on a comparison of the output and the target, until the network output matches the target. A number of input/target pairs are presented to the network during this process of *supervised learning*. Increasing the number of available input sets for the training of a network, allows the use of more complex network architecture, thereby improving the network performance.

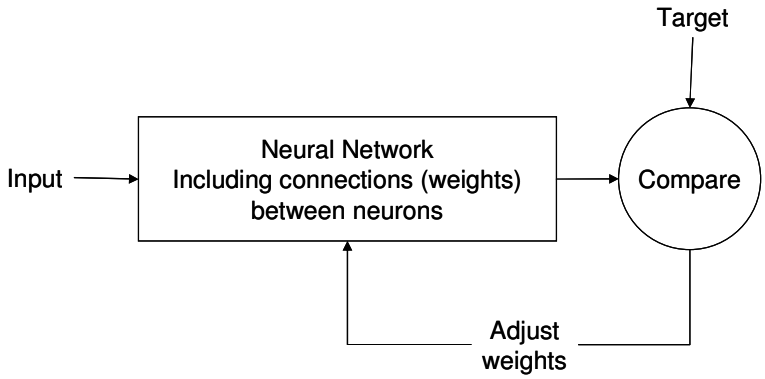

Figure 15: Training process of a MLP network (Source: MATLAB neural network toolbox manual)

2.2 Brief History

According to Hagan et al. (1996), the work of McCulloch and Pitts (1943) is commonly accepted as the origin of the neural network field. They introduced the neuron concept and showed that networks of artificial neurons could, in principle, compute any arithmetic or logical function. Following on the research by McCulloch and Pitts, Hebb (1949) proposed the first learning rule for neurons. The weight adjustment was made proportionally to the product of the outputs of the neurons before and after the weight.

Rosenblatt (1958; 1960; 1962) was responsible for the first practical application of artificial neural networks. He and his colleagues built a perceptron network with an associated learning rule and demonstrated the network's ability to perform pattern recognition. The research group focused on finding appropriate weights for specific computational tasks. A great deal of interest was generated in neural network research by this success. Rosenblatt was able to prove the convergence of a learning algorithm for simple networks, but unfortunately, it was later shown that the basic single layer perceptron network suffered from some serious limitations.

Widrow and Hoff (1960) also introduced a new learning algorithm during this period and used it to train networks that were similar in structure and capability to Rosenblatt's perceptron. These single layer networks were called adalines (adaptive linear neural networks). The Widrow-Hoff learning rule, also referred to as the delta rule, is still in use today and served as a predecessor of the backpropagation rule.

Both Rosenblatt's and Widrow's networks suffered from the same inherent limitations. These limitations were brought to light in a book by Minsky and Papert (1969). They illustrated that numerous elementary computations could not be

successfully performed with such network architecture. These single layer networks could successfully classify linearly separable patterns, but were unsuccessful on other patterns such as the exclusive or problem (XOR).

No solution could be found for these shortcomings which led to a relatively quiet period in neural network development. Rosenblatt studied network structures with multiple layers in an attempt to overcome the limitations, but could not create a suitable learning algorithm to train these networks

Minsky and Papert (1969) advocated research in other fields of artificial intelligence, as they believed that further research on neural networks was a dead end. They concluded that multi-layer networks would suffer from the same limitations as the single-layer type. Neural network development therefore went into hibernation.

During the quiet period in the 1970s, some important work continued. New networks that could act as memories were introduced separately by Kohonen (1972) and Anderson (1972). The Kohonen learning rule trains a selected neuron's weight vectors to take on the values of the current input vector. Active investigations by Grossberg (1972) also continued in the field of self-organizing networks.

Interest in neural networks was rekindled by two new concepts introduced in the 1980s. The first important development involved the use of statistical mechanics by Hopfield (1982) to explain the operation of a certain class of dynamically stable recurrent network, which could be used as an associative memory. A recurrent neural network contains feedback loops from its outputs to its inputs, but this layout previously suffered from stability problems.

Another key development of the 1980s was the backpropagation algorithm for training multilayer perceptron networks. It had been previously proposed by Bryson and Ho (1969), but no computational equipment was available to solve the demanding computations. In the mid-1980s it was re-discovered independently by several different researchers. At this time the rapid development of computers helped to overcome computational impediments, and research in neural networks was greatly facilitated. The most influential work on the subject of the backpropagation algorithm was published by Rumelhart and McClelland (1986). The backpropagation algorithm answered the criticisms that Minsky and Papert had made in the 1960s, as it could also solve the XOR problem.

For more information on the development of neural networks, the reader is referred to the collection of influential papers published by Anderson and Rosenfeld (1988).

The birth of radial basis function neural networks can be traced back to techniques which were required to perform exact interpolation of a set of data points in a multi-dimensional space. A number of modifications by Broomhead and Lowe (1988) and Moody and Darken (1989) to the exact interpolation procedure resulted in the development of the radial basis function (RBF) networks that we know today.

2.3 Multi-Layer Perceptron Networks

The brief preceding history on neural networks has already offered a short description of a number of neural network variations that have been proposed in the past, many of which are still in use. It is not the intention to embark on a detailed or exhaustive description of neural network methodology. The focus will therefore fall on those methods applicable to the purpose of this research project, which is the application of neural networks to reliability data analysis. Discussion will revolve around the two

most common network types that are used today, namely multi-layer perceptron networks (MLP) and radial basis function networks (RBF).

The following paragraphs deal in more detail with the multi-layer perceptron network, which is probably the most commonly used form of neural network. Aspects discussed include network structure, error functions and training algorithms. Emphasis will be placed on supervised learning methods, where the network parameters are adjusted to optimize the performance of the network. Important issues influencing network training, such as overfitting, pre-processing and ill-conditioning, which have an effect on neural network performance, are also examined.

2.3.1 Network Structure

Figure 16 shows the typical construction of a multi layer perceptron network. This particular example has one hidden layer, linked with weights to the input signals and the output layer.

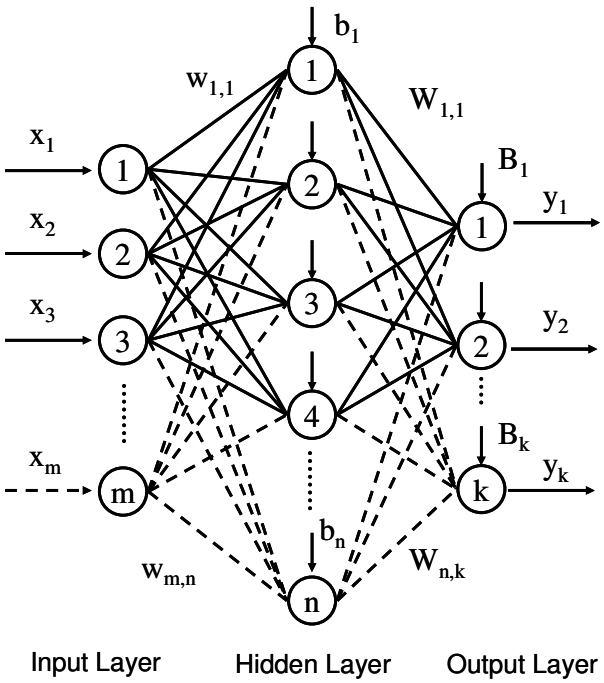

Figure 16: Illustration of the structure of a MLP network.

At each node, the sum of the weighted inputs and the bias serve as an input to the transfer function. A single output is generated which is passed on to the next layer as required. Network complexity is varied by adding or removing nodes and must be chosen to suit the training data set and the number of inputs and outputs. The choice of network size exerts an important influence on the performance of the network, as the more detailed discussion on network generalization and ill-conditioning will illustrate more clearly in separate paragraphs below.

At this stage a simple illustration will suffice and the reader is referred to Figure 17. In a situation where only a small amount of data is available, the complexity of the network must be reduced to prevent overfitting and ill-conditioning. The network does not have enough information, however, to generate an accurate representation of the situation that is being modelled. A network that is trained on enough data for a good approximation, but does not consist of enough nodes, lacks the flexibility to

generate a good fit. If the data set size is sufficient and is matched by the right amount of flexibility in the network architecture, a fit will result that gives a good representation of the system's underlying properties.

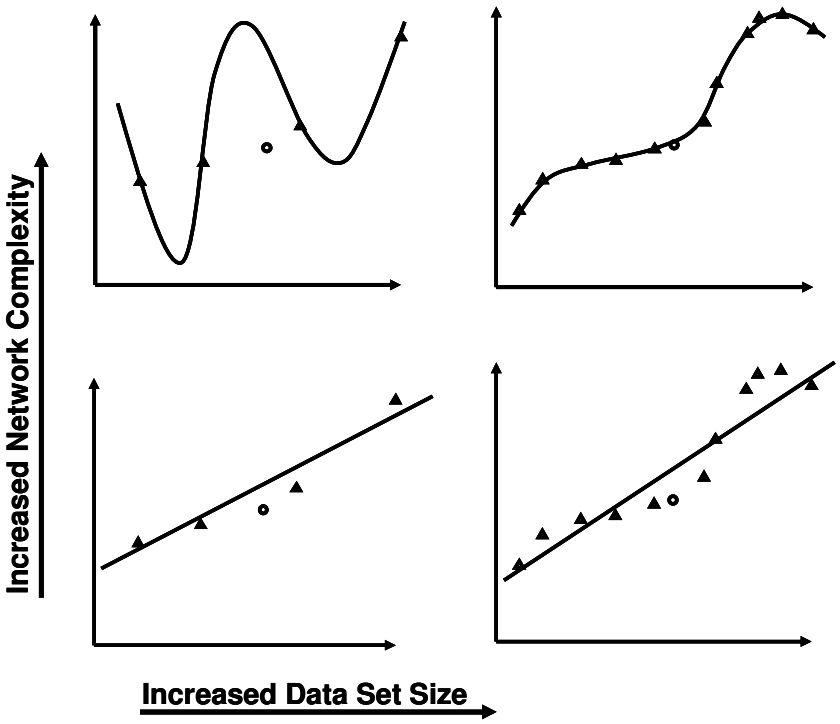

Figure 17: Four examples illustrating the relationship between data set size and network complexity

2.3.2 Transfer Functions

A number of different transfer functions can be used in such a MLP network. The choice of transfer function is dictated by the nature of the input data fed to the network and the nature of the outputs required. It must be ensured that the form of the data is compatible with the architecture that is chosen. The data can be tailored by numerous pre-processing methods to ensure that it is fit for use in conjunction with the chosen transfer functions. It is also very important that it is feasible to calculate the derivative of such a function. The three most commonly used transfer functions in the various layers of multi layer perceptron networks are the log-sigmoid, tan-sigmoid and linear functions. A brief discussion of these functions serves to introduce the reader to each of them. The equations are sourced from the work of Negnevitsky (2002).

Sigmoid units allow for smooth multivariate mapping. The sigmoid function (shown in Figure 18) transforms the input, which can have any value between plus and minus infinity, into a value in the range between 0 and 1. Neurons with this function are regularly used in the backpropagation networks, especially in the hidden layer. A sigmoid transfer function in the output layer constrains the value of the output to the range between 0 and 1. The network must therefore be trained with outputs modified to suit this range. This problem of constraint can however be overcome through the

use of post-processing for the conversion of outputs. The log-sigmoid function has the following form:

$$f(n) = \frac{1}{1+e^{-n}} \tag{2.1}$$

The use of the sigmoid function is especially convenient due to its simple derivative which is employed during the backpropagation training process.

$$f'(n) = f(n) \cdot (1 - f(n)) \tag{2.2}$$

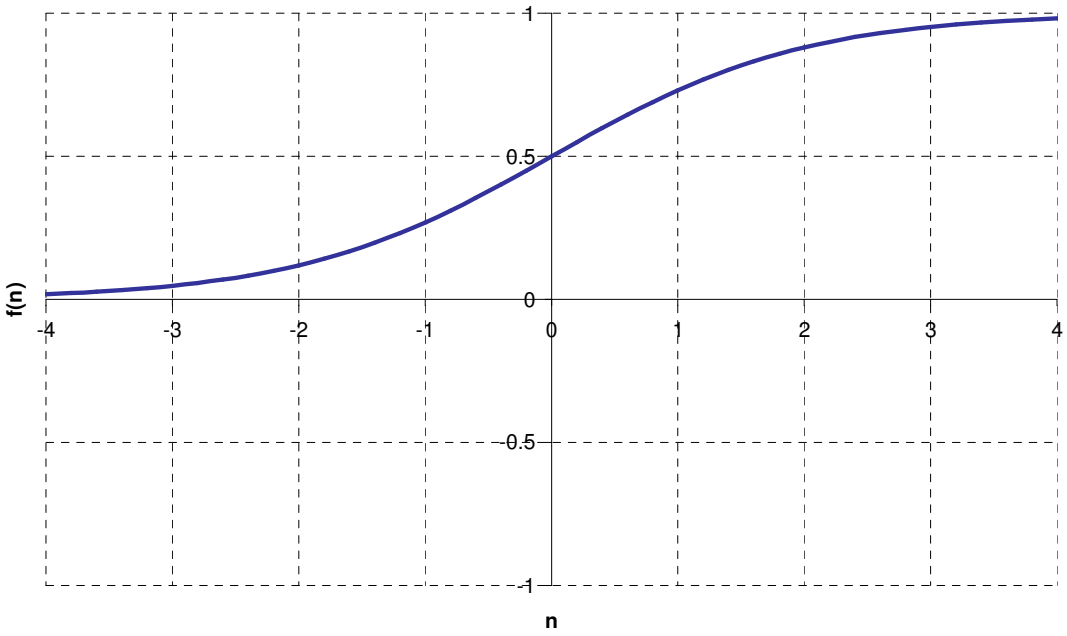

Figure 18: Sigmoid transfer function (Source: Negnevitsky (2002))

The tan-sigmoid transfer function (Figure 19) forces an output between negative and positive one. It is also mainly used for the hidden layers of feedforward neural networks, but is less popular than the log-sigmoid function. According to Bishop (1995) this function may have a slight advantage over the standard sigmoid transfer function, as it seems to result in more rapid convergence during empirical tests.

$$f(n) = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{2.3}$$

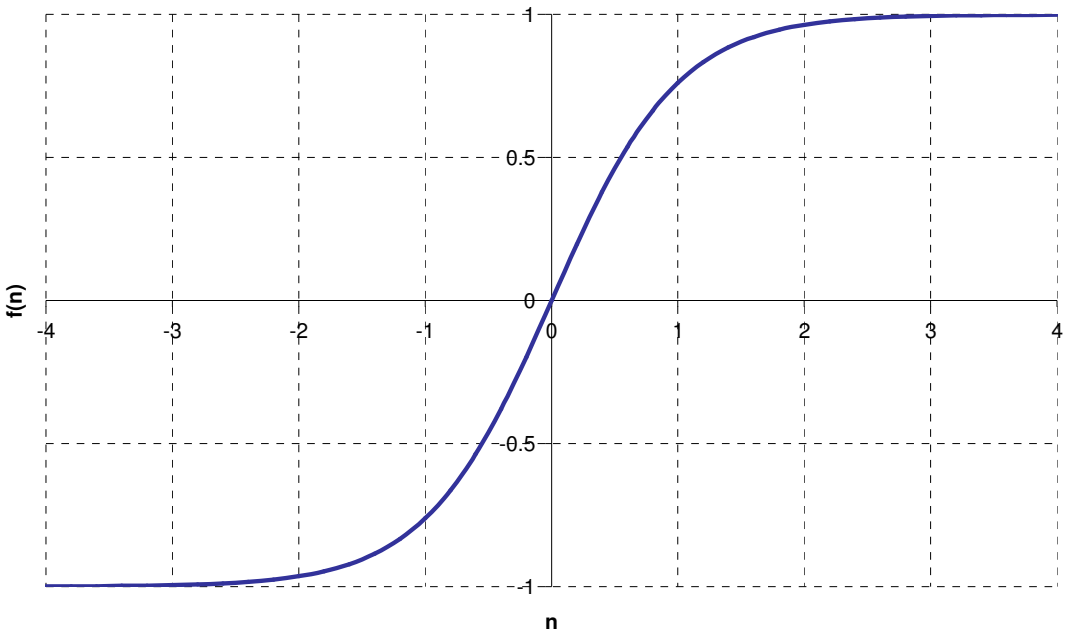

Figure 19: Tan-sigmoid transfer function (Source: Negnevitsky (2002))

The linear transfer function (Figure 20) allows outputs with an absolute value that is greater than one, hence its frequent use in the output layer. A linear activation function provides an output equal to the neuron weighted input. Neurons with the linear function are often used for linear approximation. The linear function has the following form:

$$f(n) = n \tag{2.4}$$

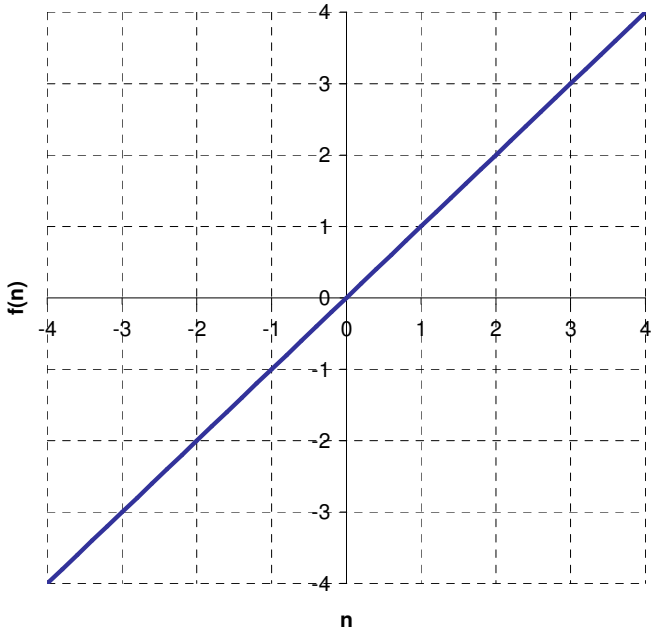

Figure 20: Linear transfer function (Source: Negnevitsky (2002))

Other transfer functions which have historic significance are the step and sign functions, which are hard limit functions and are used for decision-making neurons for classification and pattern recognition tasks.

2.3.3 MLP Network Training

Numerous algorithms have been developed for the training of neural networks. Battiti (1992) and Van Der Smagt (1994) review a number of these first and second order supervised training methods that are used for MLP neural networks. Apart from supervised learning, networks can also be trained by other methods such as associative learning, as proposed by Hebb (1949), and competitive learning, but these are beyond the scope of this discussion.

The supervised process compares the network response for a given set of inputs with the outputs that are expected for such inputs (see Figure 16). The performance of the network is measured according to a selected performance criterion. The training process aims at improving this performance index by optimizing the network through the adjustment of the network's parameters. The most commonly used error function used for this purpose is the sum of squares error which is presented by Bishop (1995) in the following form:

$$E(\overline{w}) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{C}\{y_k(\overline{x}^n;\overline{w}) - t_k^n\}^2 \qquad (2.5)$$

The sum-of-squares error function was found to be suitable for regression problems and is normally chosen because of its analytical simplicity. A shortcoming associated with the sum-of-squares error is that a small number of data points which display a very large error can dominate the training process and result in distorted network parameters. One faulty data entry could therefore nullify the whole training process.

The backpropagation algorithm serves as a convenient starting point for discussion on the minimization of the error function. Many other methods exist that are used to adjust parameters during each stage of training. A number of modifications to the original backpropagation method have been proposed, which offer improvements in performance. Apart from these heuristic techniques, standard numerical techniques have also been employed in the training of networks with excellent results. While keeping the application of neural networks to the problem of reliability data analysis in mind, the most important modifications and alternatives to the gradient descent backpropagation method are briefly discussed and compared.

Standard Backpropagation

Though Werbos (1974) and Parker(1985) had previously proposed similar ideas, the popular use of the backpropagation technique can be traced back to the paper by Rumelhart, Hinton and Williams which was published in 1986.

Supervised network training involves an iterative procedure for minimization of an error function. Weight adjustments are made in a sequence of steps, each consisting of two stages. The first stage involves the evaluation of the derivatives of the error function with respect to weights ($\overline{g}_k$), while the adjustment to the weights ($\Delta\overline{w}_k$) is calculated during the second stage. The new weights are calculated with Equation 2.6.

$$\overline{w}_{k+1} = \overline{w}_k + \Delta\overline{w}_k \qquad (2.6)$$

where

$$\Delta \overline{w}_k = -\alpha_k \overline{g}_k \tag{2.7}$$

The learning rate (α_k) dictates the step size of the weight change in the direction of the steepest gradient. Overshooting occurs when this learning rate is too large, and the network may become unstable and oscillate. A step size that is too small results in very slow convergence. The learning rate must be tailored to the specific network to get the best results.

The success and popularity of the backpropagation technique can be attributed to the fact that it is a computationally efficient method for evaluating the derivatives of the error function and adjusting the weights. Unfortunately backpropagation is a time consuming procedure and a number of variations have consequently been developed to improve on its rate of learning. Work has also centred on improving network generalization and avoiding local minima during training. These alternative techniques can be subdivided into two groups namely heuristic techniques and standard numerical optimization techniques. Faster techniques now exist that are between 10 and 100 times quicker than the basic backpropagation method.

Heuristic Improvements

Jacobs (1988) attributes the slow convergence of standard gradient descent backpropagation to two causes. The first of these flaws results in the incorrect adjustment of weight magnitudes, while the second leads to the choice of a sub-optimal direction for the weight adjustment vector. The shape of the error surface has a critical effect on the successful convergence of the steepest descent algorithm. If the error surface has a flat shape, the gradient that is calculated is low. This results in a very small step size and subsequent slow convergence. If the error surface is highly curved, the steep gradient leads to a large derivative value and therefore a large step size. The large adjustment causes oscillation as the training algorithm continually overshoots the target. The second flaw of the gradient descent method is that the direction of the calculated negative gradient is not always the same as the direction to the error function minimum. This phenomenon affects training when the shape of the error surface is elongated.

Heuristic techniques are limited to the modification of the standard gradient descent method. The four heuristics proposed by Jacobs to improve the rate of convergence of standard steepest gradient descent can be summarised follows:

1. An individual learning rate should be given to each parameter of the performance measure.
2. Each learning rate must be adjustable over time.
3. To overcome slow convergence due to small curvature, the learning rate should be increased if the derivative of the parameter keeps the same sign over a number of iterations.
4. To overcome the oscillation due to high curvature, the learning rate is decreased if the sign of the derivative changes in consecutive iterations.

Among the heuristic techniques that have found favour is the use of an added momentum term, the variation of the learning rate and resilient backpropagation.

For the momentum technique, another term is added to the weight change calculated by the gradient descent backpropagation rule. This term is the product of the momentum constant (β) and the weight change of the previous iteration. The momentum constant is a scalar factor which is chosen between zero and one. It influences the updating of the weights by taking into account recent trends in the error surface. The momentum that is introduced allows the algorithm to pass through local minima without getting stuck. Training with this technique is significantly quicker than the standard gradient descent method.

$$\overline{w}_{k+1} = \overline{w}_k + \Delta\overline{w}_k + \beta \cdot \Delta\overline{w}_{k-1} \qquad (2.8)$$

When using a variable learning rate, the learning rate is increased until instability is sensed in the calculated gradient. In this way the learning rate is maintained at a maximum level throughout the training process.

Resilient backpropagation is a third heuristic technique that can be used and relies on the sign of the gradient. The weight changes are increased or decreased, depending on the sign of the derivative and in this way the problem of a small slope at the extreme ends of the sigmoid transfer function is eliminated.

Standard Numerical Methods

Among the alternatives to the backpropagation method that are used are the conjugate gradient methods, variations on Newton's method, the quasi-Newton and Levenberg-Marquardt algorithms.

Conjugate Gradient Algorithm

The standard backpropagation method adjusts the weights in the direction of steepest descent, which is the negative of the gradient (see Equation 2.7). This is the direction of most rapid decrease in the error function but is not the optimal direction for convergence.

The conjugate gradient method, in contrast, employs the conjugate gradient direction for a line search which produces much faster convergence as each step is taken in the optimal direction resulting in much more rapid convergence (See Figure 21). The diagram illustrates the advantage of using a conjugate gradient method when compared with the gradient descent method which follows an inefficient zigzag trajectory. The accuracy of line minimizations is important to ensure the finding of the correct conjugate directions and orthogonal gradients.

Broyden (1996) found that the existence of such a large number of variations of the conjugate gradient algorithm make it difficult to obtain a comprehensive view on the methods which are available. Some of the more important conjugate gradient methods have therefore been selected for discussion with the purpose of providing basic background information on the subject.

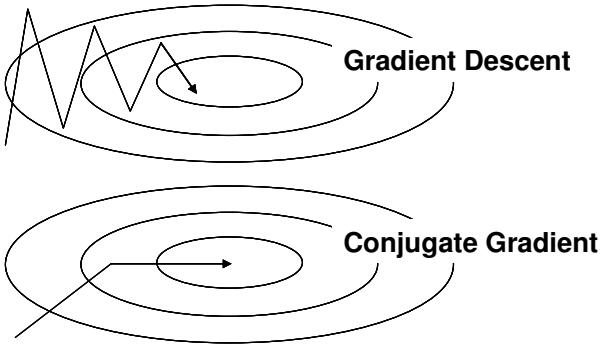

Figure 21: Illustrative comparison of the performance of the gradient descent backpropagation and conjugate gradient methods. (Adapted from Hertz et. al (1991) and Bishop (1995))

The step length taken during training is governed by a coefficient α_j which is defined according to Bishop (1995) as follows:

$$\alpha_j = -\frac{\bar{d}_j^T \bar{g}_j}{\bar{d}_j^T H \bar{d}_j}$$

Explicit calculation of this coefficient is not required as a numerical procedure which involves a line minimization along the search direction $\bar{d}_j$ is employed.

The new search direction that is used by the conjugate gradient method is defined by Equation 2.9 which is given below.

$$\bar{d}_{j+1} = -\bar{g}_{j+1} + \beta_j \bar{d}_j \qquad (2.9)$$

In this equation the scalar coefficient β_j is usually calculated with the Hestenes-Stiefel formula (Equation 2.10), the Polak-Ribiere formula (Equation 2.11) or the Fletcher-Reeves formula (Equation 2.12) which are all quoted by Bishop (1995).

Hestenes-Stiefel formula

$$\beta_j = \frac{\bar{g}_{j+1}^T (\bar{g}_{j+1} - \bar{g}_j)}{\bar{d}_j^T (\bar{g}_{j+1} - \bar{g}_j)} \qquad (2.10)$$

Polak-Ribiere formula

$$\beta_j = \frac{\bar{g}_{j+1}^T (\bar{g}_{j+1} - \bar{g}_j)}{\bar{g}_j^T \bar{g}_j} \qquad (2.11)$$

Fletcher-Reeves formula

$$\beta_j = \frac{\bar{g}_{j+1}^T \bar{g}_{j+1}}{\bar{g}_j^T \bar{g}_j} \qquad (2.12)$$

The two coefficients α_j and β_j. are therefore both calculated without the need to evaluate the Hessian matrix.

The key steps to the conjugate gradient algorithm as summarised by Bishop (1995) are as follows:

1) An initial weight vector $\overline{w}_1$ is chosen.
2) The initial search direction is set to $\overline{d}_1 = -\overline{g}_1$, where $\overline{g}_1$ is the gradient vector.
3) At each step, the error $E(w_j + \alpha \overline{d}_j)$ is minimized with respect to α such that the new weight vector is $\overline{w}_{j+1} = \overline{w}_j + \alpha_{\min} d_j$.
4) Training is now stopped if the stopping criterion is satisfied.
5) The new gradient vector $\overline{g}_{j+1}$ is calculated.
6) The new search direction can now be calculated (Equation 2.9).
7) The program returns to Step 3.

The conjugate gradient method requires only first derivative information in conjunction with line searches along a selected direction to minimize the error. As the calculation of the second derivative is not necessary, it uses much less storage space than second order methods for which the Hessian matrix must be computed.

Quasi-Newton

Where the conjugate gradient only used second order information implicitly, Newton's method makes explicit use of the Hessian matrix. The quasi-Newton algorithm is based on Newton's method, but avoids this computation by using an approximation of the Hessian matrix. Over a number of steps increasingly accurate approximations of the inverse to this matrix are generated using only information from the first derivatives of the error function. The use of an approximate Hessian matrix also overcomes the problem of a Hessian matrix that is not invertible.

The weight change of the Quasi-Newton method is as follows:

$$\Delta \overline{w}_k = -\left[J^T(\overline{w}_k) J(\overline{w}_k)\right]^{-1} J^T(\overline{w}_k) e(\overline{w}_k) \tag{2.13}$$

where the approximation of the Hessian matrix is:

$$H(\overline{w}_k) = J^T(\overline{w}_k) J(\overline{w}_k) \tag{2.14}$$

and the gradient is given as:

$$\overline{g}_k = J^T(\overline{w}_k) e(\overline{w}_k) \tag{2.15}$$

The memory usage and computation time of this method is substantial despite the use of an approximation of the Hessian matrix. This limits its application to middle-sized problems according to Robitaille et al. (1996). They therefore proposed three variations to the classical quasi-Newton approach which are aimed at reducing the computational effort associated with this method through a reduction in the size of the approximate Hessian matrix by neglecting some second order interactions.

Levenberg-Marquardt

The Levenberg-Marquardt algorithm (Levenberg, 1944; Marquardt, 1963) was designed specifically for minimizing the sum-of-squares error and is an adaptation of

the Gauss-Newton algorithm. The weight changes are calculated using the following equation:

$$\Delta \overline{w}_k = -\left[J^T(\overline{w}_k)J(\overline{w}_k) + \mu_k I\right]^{-1} J^T(\overline{w}_k)e(\overline{w}_k) \tag{2.15}$$

The key step to the algorithm is the computation of the Jacobian matrix (see Hagan et al. (1996)) which involves the calculation of the derivatives of the errors and has the form:

$$J(\overline{w}_k) = \begin{bmatrix} \frac{\partial e_1(\overline{w}_k)}{\partial w_{k1}} & \frac{\partial e_1(\overline{w}_k)}{\partial w_{k2}} & \cdots & \frac{\partial e_1(\overline{w}_k)}{\partial w_{kn}} \\ \frac{\partial e_2(\overline{w}_k)}{\partial w_{k1}} & \frac{\partial e_2(\overline{w}_k)}{\partial w_{k2}} & \cdots & \frac{\partial e_2(\overline{w}_k)}{\partial w_{kn}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_m(\overline{w}_k)}{\partial w_{k1}} & \frac{\partial e_m(\overline{w}_k)}{\partial w_{k2}} & \cdots & \frac{\partial e_m(\overline{w}_k)}{\partial w_{kn}} \end{bmatrix} \tag{2.16}$$

When the factor μ is small, the direction taken by the algorithm corresponds to the Gauss-Newton direction. When it is large, the direction corresponds to gradient descent method. Note that as μ is increased, the algorithm moves towards a small step in the direction of steepest descent. This guarantees that the algorithm will reduce the sum of squared errors.

The steps that are followed during the Levenberg-Marquardt training process may be summarised as follows (see also Hagan et al. (1996):

1) All the inputs are presented to the network, the network outputs are generated and the corresponding errors are calculated. The sum of squares error can now be calculated.
2) The Jacobian matrix is computed.
3) Calculate the weight change with Equation 2.15
4) The sum of squares error is re-computed using the updated weights. If the new sum of squares error is less than before, then $\mu=\mu/\beta$ and the program proceeds to step 1. If the sum of squares error increases, $\mu=\mu.\beta$, the relation in step 3 is re-calculated with the new value of μ.

In their work, Hagan and Menhaj (1994) tested the Levenberg-Marquardt algorithm on five function approximation problems. They compared the performance of this algorithm to the backpropagation with variable learning rate (VLBP) and conjugate gradient backpropagation (CGBP). The Fletcher-Reeves conjugate gradient algorithm was used, performing an exact line search consisting of two parts. The first was interval location, using function comparison, while the second was a golden search. The conclusion drawn by the authors was that the Levenberg-Marquardt algorithm outperformed the other two methods despite higher computational requirements. The improved performance is attributed to the increased efficiency offered by this algorithm.

2.3.4 Network Generalization and Overfitting

Generalization is the accuracy with which the network can solve for an input vector that it has not seen before during training. If a network is trained until the error on the

training set is reduced to a very small value, the error on new data that was not presented to it during training becomes large due to overfitting (see Figure 22). A network may also be unable to produce good results if it does not have the required flexibility to model a particular dataset.

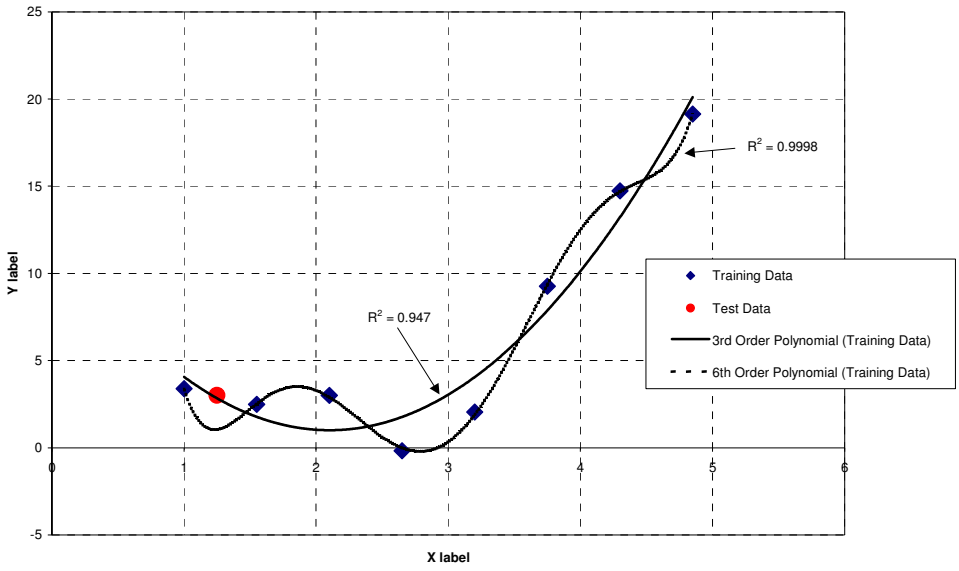

Figure 22: Graph illustrating the concept of overfitting.

The bias is a measure of the extent to which the average of the network function over all datasets differs from the desired function. A high bias means the network function is on average very different to the regression function. This condition occurs when a network does not have sufficient flexibility because it has too few parameters. The variance measures how sensitive the network function is to a particular dataset that may be chosen. A high variance therefore indicates high sensitivity of the network function with respect to a particular dataset, which is caused by excessive flexibility. The result is overfitting as shown in Figure 22. In order to achieve the optimal fit, the bias and variance of the neural network must be balanced.

Increased network size allows the modelling of more complex functions. The aim is to use the simplest network that can still adequately represent the training set.

According to Schittenkopf et al. (1997) there are generally two routes that are currently followed in order to prevent overfitting during training. The first approach involves the reduction of the size of the parameter space. This may involve the removal of weights or neurons depending on how sensitive the network error is to that particular parameter. The second alternative that can be implemented is the reduction of parameter dimensions. A penalty term in the error functions prevents the network from generating overly complex solutions.

Generalization can be improved by trimming the network of any components that are superfluous. It therefore is not powerful enough to overfit. As it is not always feasible to trim a network by trial and error, methods such as regularization and early stopping have been introduced to improve generalization.

Prechelt (1998) found that the cross-validation method which is normally used for early stopping does not allow for the complexity of the real situation. The procedure generally followed involves the division of the training data into a training set and a

validation set. The training set is used to adjust the network parameters, while the error on the validation set is continually monitored. Once the network error on the validation set increases, as shown in Figure 23, the training is stopped in order to prevent overfitting.

The outlined method may fail due to the effect of a local minimum which may cause a temporary increase in the global error without actual overfitting. This problem is dealt with by Prechelt (1998) who evaluated a number of different stopping criteria for automatic early stopping. Three classes of stopping criteria are identified. The first involves generalization loss, the second uses the quotient of the generalization loss and progress, while the third relies on the sign changes of the generalization error.

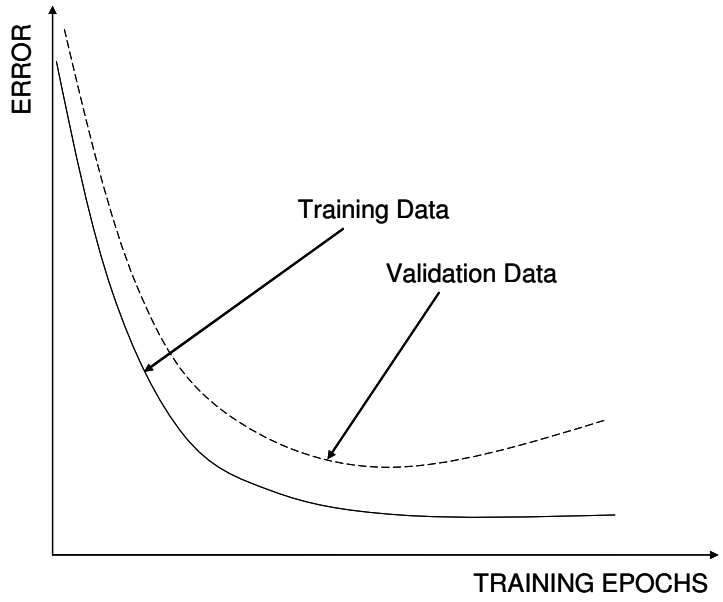

Figure 23: The graph shows how the error of the validation set starts to increase once overfitting has occurred. The error on the training set continues to decrease. (Source: Bishop (1995))

McLoone and Irwin (2001) describe their application of regularization to the training of feedforward neural networks. This is another technique that is used to smoothen the mapping of the neural network. It adds a penalty term, Ω to the error function giving the equation:

$$\tilde{E} = E + \upsilon\Omega \tag{2.17}$$

Here Ω is the penalty term, E is the standard error function and υ is a parameter which controls the size of the penalty (Bishop 1995).

The standard least squares error performance function is modified and takes the following shape:

$$MSE_{\mathrm{Re}\,g} = \gamma \cdot \frac{1}{N}\sum_{i=1}^{N} e_i^2 + (1-\gamma)\cdot\frac{1}{n}\sum_{j=1}^{n} w_j^2 \tag{2.18}$$

In this equation, γ is the performance ratio which regulates the relative weight given to the mean square of errors term and the mean square of weights term. This modified performance function generates a smoother network response by causing smaller weights and biases. It is however difficult to find the optimal value for the

performance ratio, as a performance ratio that is too large may result in overfitting, while a ratio that is too small will cause an inadequate fit.

MacKay (1992a, 1992b) outlines the Bayesian approach to regularization and describes a practical Bayesian framework for training feedforward neural networks. According to Bishop (1995), Bayesian regularization is a natural representation of the Bayesian framework that allows the values of regularization coefficients to be selected without resorting to a separate validation dataset. A large number of regularization coefficients can therefore be employed as no cross-validation is necessary.

It is noted that if insufficient data is available, the network is likely to perform inadequately despite the use of regularization, or other techniques that are designed to prevent overfitting

Pre-Processing

Neural networks have an advantage over statistical methods as they do not require such close attention to the pre-processing of data. It is however still necessary, for most practical applications, to transform the data in some way before the training is started. Training of networks on raw data causes inefficiency and generally leads to poor results. The choice of pre-processing and post-processing steps which are to be applied to data exerts a significant influence on the network's ability to generalize once training has been done. The aim of pre-processing is to try and eliminate all superfluous elements from the training data while retaining the relevant information required by the network to be effective. This makes it one of the most important stages in the development of a solution for any particular practical application. It is usually most convenient to pre-process the whole dataset before training, using the fully transformed data for training.

One of the pre-processing methods that are most commonly encountered in practice is the re-scaling of input data. The aim is to transform all the inputs in such a way that they all have a value of similar magnitude. When dealing with raw data, the various inputs may differ by several orders of magnitude. This discrepancy in size serves to distort the training process due to the dominance of large input variables. This tendency is undesirable as the average numerical size of a particular input variable may not necessarily reflect the importance of that variable. During training a relative importance is assigned to each of these variables and large size differences serve to hamper this process. Though it may not be generally necessary for pattern recognition problems, a re-scaling of the output values is usually also done when dealing with regression models.

It will be seen later that the activation of the hidden nodes in RBF networks which have spherically symmetric basis functions is dependant on the Euclidean distance between the input and the basis function centre. These networks are therefore even more sensitive to variation of size amongst the input variables, placing emphasis on the proper re-scaling of inputs.

By applying a linear transformation we can ensure that all of the inputs to have similar magnitude. During re-scaling, each input variable is treated independently, with the mean and variance calculated through the use the following formulae which appear in Bishop (1995):

$$\bar{x}_i = \frac{1}{N}\sum_{n=1}^{N} x_i^n \tag{2.19}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{n=1}^{N}\left(x_i^n - \bar{x}_i\right)^2 \tag{2.20}$$

The set of re-scaled variables then becomes:

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i} \tag{2.21}$$

Input normalization ensures that all of the input and target variables are of the order of unity. The weights will be of a similar order and can therefore be randomly initialized with reduced risk of unsatisfactory network behaviour during training. The limiting of initial weights to a well defined range results in a shortened training process with a greater chance of reaching the global minima.

A number of benefits are also associated with a dimensional reduction of the input data. Feature extraction is one way of achieving this and involves the forming of data combinations. This reduction in the number of inputs means fewer parameters to be determined during training, which reduces not only memory usage and training time but also means that smaller datasets can be used successfully.

The problem complexity can also be addressed through the grouping of input points. Feature extraction and the grouping of inputs allows for the incorporation of prior knowledge into the training process. Apart from the transformation of data, it should also be screened for deficient readings which must be discarded. Such data may cause the training algorithm to fail. Data with missing reading may however be needed for training if the available dataset is sparse. In such cases it may be decided to replace the missing readings with a mean value.

Network Testing

The usefulness of a neural network when considering it for a practical application depends on the degree to which it can generalize when confronted with data which was not seen during training. Methods have been developed to test and compare the performance of different networks with this aim in mind. Anders and Korn (1999) suggest a number of strategies for the selection of neural network models based on statistical concepts. Schenker and Agarwal (1996) identify the three most common methods for testing the relative performance of neural networks.

- A sub division of the available data into a training and test set which is termed a static split.
- Cross-validation which can be described as a dynamic split of the data.
- Statistical evaluation without splitting the data.

Testing through the use of statistical methods, according to Schenker and Agarwal (1996), is only meaningful when the data represents a true process. It can therefore be successfully applied in cases where reliable physically based models are involved. Schenker and Agarwal (1996) identify the subdivision of the dataset into separate training and test sets as the approach which is most commonly used, even though only part of the dataset can be used for training which limits this method's application to larger datasets. In their comparison of the performance of the different testing

methods, Schenker and Agarwal (1996) proved that a strategy of cross-validation generally outperformed such a static split in the search for an optimal network for a particular application.

For the purposes of comparing different neural network variations by cross-validation, the dataset is broken into a number of smaller groups which do not overlap. These groups are cyclically allocated to the training and the test sets. Each cycle in the cross-validation process represents a completely independent training run such that the networks are not tested with data used for training at a previous stage. The error on the test data is recorded for each of the network variations at the completion of each cycle. Several partially overlapping portions of the available data are therefore used for training the networks, but each group of data is used only once for testing. The recorded error values are added once the process is completed and this result forms the basis of comparison between the different neural networks.

The steps involved in cross-validation can be summarised as follows:

1) The dataset is broken into *L* parts which do not overlap.
2) *L* pairs of training and test set are formed where the data within each pair also does not overlap.
3) The different networks are retrained using the training sets and the performance is the evaluated using the test set. An error is calculated with respect to the test set.
4) Step 3 is repeated such that each of the *L* parts is used once as part of a test set and the overall network performance can be estimated by averaging the mean errors over the *L* pairs or the summation of errors for each individual point.
5) The network with the smallest error is selected as the most optimal solution.
6) The chosen network is then trained with the complete dataset.

The greatest advantage offered by the use of cross-validation is that the entire dataset can eventually be used for training the neural network once the optimal neural network layout has been found. The loss of information due to a static split of data is therefore avoided which is important for cases where the dataset is limited in size. Training does unfortunately become more cost intensive due to the repetition required by cross-validation.

A sub division into a separate training and test set is only possible when the overall dataset is of sufficient size to accommodate such a subdivision if the resulting training set is of sufficient size to allow meaningful training. Schenker and Agarwal (1996) identify cases where a substantial amount of data is easily collected through simulations and inexpensive experiments as the ideal application for this method of testing. In cases where the dataset is limited in size, however, cross-validation provides the optimal solution for the testing of network generalization.

2.3.5 Ill-Conditioning

The training process of MLP neural networks is a non-linear least squares problem solved by means of numerical optimization techniques. These methods may be rendered ineffective by ill-conditioning. In their paper on the subject, McKeown, Stella and Hall (1997) discuss the problem of ill-conditioning in great detail, noting that the issue is regularly ignored in literature. Many researchers train their networks

with insufficient data which would inevitably result in severe ill-conditioning and false solutions. Paucity of data is an important consideration, especially when working with reliability data.

The columns of the Jacobian matrix are composed of the derivatives of each residual error with respect to a single network parameter (see equation 2.16). Ill-conditioning results when there are fewer residual errors than optimization variables. Under these circumstances the Jacobian has fewer rows than columns and the product $J^T.J$ is singular. The number of residual errors (m) is defined as the product of the number of outputs from the network and the number of training sets. The number of optimization variables (n) is the total number of weights and biases of the network. When m<n, the dimension of the Jacobian matrix is of such a nature that there fewer rows than columns. The zero fitting error is easily obtained as there are fewer data points than there are variables, which means the problem is over-defined and the resulting solution is not unique. It will therefore normally be unable to generalize when faced with new input data.

The Jacobian may also be singular due to linear dependency among the columns. This problem is caused by redundant network elements when the network structure is too complex. The saturation of a specific node at each training point will have the effect that the corresponding column in the Jacobian is null and the matrix becomes singular.

McKeown et al. (1997) state that one of the problems associated with a Gauss-Newton based algorithm is that it fails if the Jacobian matrix becomes singular at an estimated point. This problem has been overcome through the addition of a small positive number to each element of the main diagonal of the approximate Hessian matrix $J^T J$. The result is the Levenberg-Marquardt algorithm (equation 2.15) which is able to escape from such regions of singularity.

With regards to the practical implications of ill-conditioning on the use of neural networks, McKeown, Stella and Hall (1997) suggest the following measures for the prevention of this phenomenon:

1) Minimise the error function using a second order method (Gauss-Newton, Levenberg-Marquardt), rather than first order method (back-propagation).
2) Apply strict tolerances to function values when attempting to identify global minima.
3) Perform sensitivity analysis on the solutions that are found. This may also help to identify the redundant connections and neurons and thereby aid in the pruning of the network.

2.4 Radial Basis Function Networks

The multi-layer perceptron network, discussed above, performs global mapping. Its units compute the value of a non-linear function of the scalar product of the input vector and a weight vector. The other major class of neural network model, the radial basis function (RBF) network, is a local network type where the activation of the hidden unit is determined by the distance between the input vector and a prototype vector.

The output of a first layer neuron of a MLP network is computed by feeding a weighted sum of inputs into a sigmoid transfer function. The radial basis network

(Figure 24), in contrast, consists of a non-linear hidden layer and a linear output layer. The hidden layer performs non-linear local mapping and its neurons have a centroid and a smoothing radius factor. The input vector is multiplied by the input weight matrix thereby producing a vector that reflects the distances between the input vector and the weights. This vector, multiplied by the bias vector, serves as an input to the radial basis function. The radial basis neuron is activated in cases where the input and weight vectors are similar.

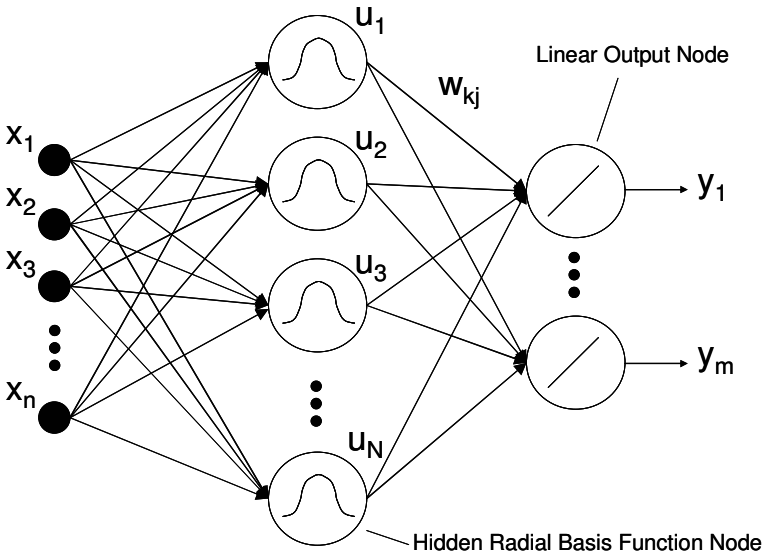

Figure 24: Layout of a radial basis function neural network.

$$u_j = \exp\left[-\frac{\left\|\bar{x}-\bar{c}_j\right\|^2}{2\sigma^2{}_j}\right] \quad j = 1,2,...,N \tag{2.22}$$

where the input vector has the form:

$$\bar{x} = [x_1, x_2, x_3, ..., x_n]^T \tag{2.23}$$

The position vector for the jth hidden node is $\bar{c}_j$, the node has the width σ_j^2 and N is the number of hidden nodes.

In the output layer the linear function is evaluated as follows:

$$y_k = \sum_{j=1}^{N} w_{kj} u_j = \bar{w}_k^T U \quad k = 1,2,...,m \tag{2.24}$$

where the weight vector is:

$$\bar{w}_k = [w_{k1}, w_{k2}, w_{k3}, ..., w_{kN}]^T \tag{2.25}$$

To understand the behaviour of radial basis function networks, it is important to first look at the shape of the function (Figure 25) used for the nodes in the layer of these networks. The Gaussian function has the following form:

$$f(n) = e^{-n^2} \tag{2.26}$$

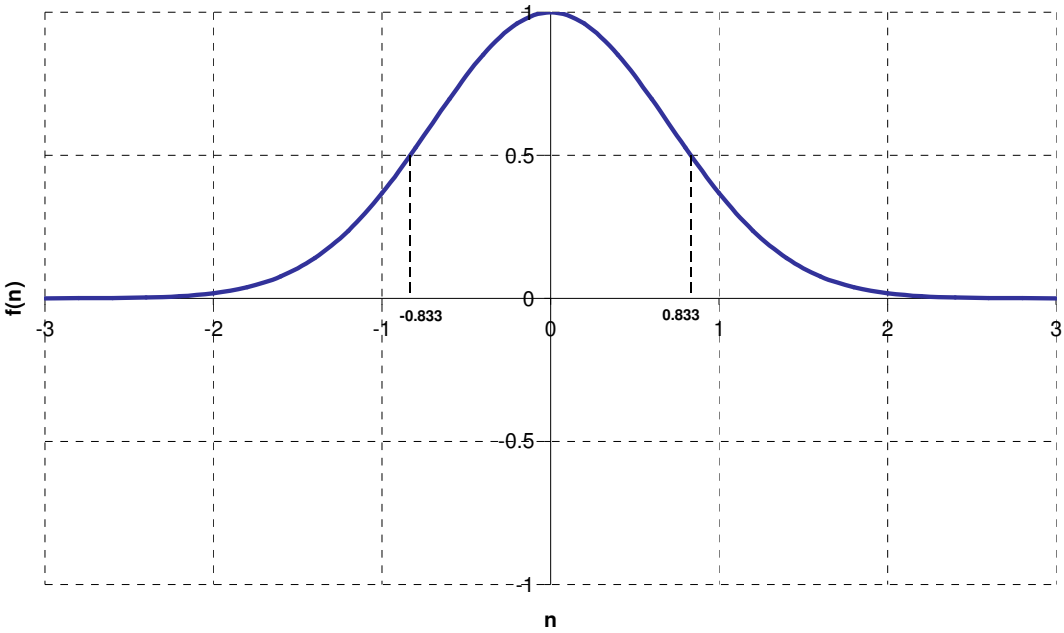

Figure 25: Radial basis function

The graph illustrates that the radial basis function has a maximum value of 1 when n is equal to zero. The shape of the function ensures that the output of the radial basis neuron increases to a maximum when the distance between the vectors approaches zero. This means that the radial basis neuron is triggered by similarity between the input and weight vectors. In cases where the weight vector is substantially different to the input vector, the radial basis neuron will have output close to zero. As the output of the RBF neuron is multiplied by the output weights in the linear neurons of the second layer, the RBF neurons that are triggered because of the similarity between their weight vector and the input vector will dominate the output of the network by transmitting their output weights.

As the hidden RBF nodes are only activated when the input vector is near the centroid of that particular neuron, these first-layer neurons are receptive only to a local region of the input space. In contrast, the use of the sigmoid transfer functions in MLP networks generates a global response. The sensitivity of RBF neurons can be adjusted by varying the radius that is assigned to the radial basis function. The rate at which the output of the neuron decreases when the input moves further away from the centroid of the function is reduced as the function radius increases. It is the function of the bias to effect this adjustment in the RBF radius, and correspondingly regulate the sensitivity of the neuron.

The bias of a RBF in MATLAB is set by defining a parameter called the spread. Every bias in the first layer of the network is set to 0.8326 divided by the spread. The radial basis functions in these neurons therefore have an output of 0.5 when the

absolute value of the distance between the input and weight vectors is equal to the spread. The area of the input space to which each neuron responds is thereby set.

When designing a RBF network, it must be ensured that the spread of the RBF neurons is large enough. If the radial basis function neurons overlap enough, several radial basis function neurons will generate significant outputs at any time. The resulting network function is smoother and better generalization is achieved for new input vectors that fall between the input vectors used in the design of the network. If the overlap is too large, however, too many neurons will then react to every input and the accuracy is lost.

2.4.1 Training of RBF Networks

A two stage training procedure is normally applied to RBF networks. The first step involves the selection of the basis function centres, while the output layer weights are determined during the second step. In the first stage the parameters governing the basis functions are normally determined using relatively fast unsupervised methods. The second stage of training involves the determination of the final-layer weights, which requires the solution of a linear problem and is therefore also fast.

It has been mentioned that the activation of a hidden unit is determined by the distance between the input vector and a prototype vector. A fixed nonlinear transformation is performed by the hidden layer where the input space is mapped onto a new output space. There are therefore no adjustable parameters in this layer. The use of a linear transfer function in the output layer of the RBF produces a weighted sum of the outputs of the hidden layer. It contains adjustable parameters in the form of weights which can be determined by using a linear least squares method.

The advantage of choosing parameters for the hidden units without having the need for a full non-linear optimization simplifies the training process of RBF networks and is also much quicker than the training of a MLP network. On the downside RBF networks will usually consist of more neurons. The sigmoid neurons of an MLP can have outputs over a large region of the input space, while RBF neurons only respond to relatively small regions of input space. As the number of inputs and the range over which each of these inputs may vary increases, the input space becomes larger leading to a corresponding increase in the number of radial basis function neurons that are required. The optimal choice of network architecture is case specific, and it will be found that for some applications a RBF network will provide the better solution.

Two variations of the RBF network are the generalized regression neural network (GRNN) and the probabilistic neural network (PNN). No iterative training process is used as the training vectors become the weight vectors in these networks. The PNN is applied in classification problems and uses a competitive output layer to select the most likely class by comparing probabilities. The output layer weights of a GRNN consist of the output target vectors. The GRNN is therefore a network type suitable for function approximation similar to standard regression methods. The GRNN and PNN are clearly memory intensive networks, as these networks basically store all input data. Training is simple, however, and merely consists of assigning the values of inputs to the first layer and target values to the second layer. For large training sets it may however be advisable to employ a technique for reducing network size.

The concern about the size of radial basis function networks and the associated computational costs and high memory requirements has led to a wide variety of methods for their optimization. Chen et al. (1991) developed a learning strategy based

on the orthogonal least squares algorithm for the construction of RBF networks. The two stage procedure of Kaminski and Strumillo (1997) where the RBF kernels are transformed into a set of orthonormal functions and the use of genetic algorithms by Whitehead and Choate (1996) to determined optimal radial basis function centres and widths are two further examples. It is beyond the scope of this project to expand in detail on the various methods that can be used to optimize such networks by choosing radial basis function parameters and centres. Only a brief description of the more important methods is therefore included for the sake of completeness.

One of the simpler methods is the selection of RBF kernels by choosing a random subset of input vectors from the training set. It is not an optimal method, but may also provide the starting values for iterative adaptive procedures. Alternatively, all the training data points may be used as centres and the centres that least affect the performance of the network are then removed.

The radius of the RBF nodes can be chosen through the application of heuristic methods, such as using the multiple of the average distance between the RBF centres. The parameter may also be set equal to the average distance from the RBF centre to a predetermined number of its nearest neighbours.

Orthogonal least squares methodology can be used to calculate which RBF centre would offer the greatest improvement in the residual sum of squares error if added to the network. The network is built by sequentially adding new basis functions, each centred on one of the data points. Overfitting is prevented by stopping this process before all data points have been used. The residual error is therefore not allowed to reach zero in the interest of achieving good generalization.

Clustering techniques may also be utilized to find a set of centres that best reflect the distribution of data points. Alternatively, the basis functions may be regarded as Gaussian mixture models, whose parameters can be optimized by using maximum likelihood.

Finally, supervised methods can also be applied to train RBF networks. The basis function parameters such as the RBF centres, radii and the second layer's weights are treated as adaptive parameters. The error function is then minimized by changing these parameters during a supervised training process. The use of supervised training means however that one of the main advantages of RBF networks, which is the fast and simple two-stage training process, is lost.

2.5 Conclusion

In the preceding overview, a number of important issues with regards to the application of neural networks have been touched upon which need to be addressed when using these methods for reliability data analysis.

Two network architectures are of specific interest in view of this project. RBF networks have an advantage over MLP networks when the available dataset is sparse. The localized nature of the RBF network means that the learning process does not involve the minimization of the global error as is normally the case with MLP networks. Such a training process places limitations on the minimum dataset size required for a given network complexity, as ill-conditioning may result.

The MLP network gains an advantage over the RBF network type as the size of the input space increases. The localized nature of RBF hidden nodes limits their coverage which means that the network complexity increases exponentially with an increase in

size. MLP networks form a global approximation of the underlying properties of the system that is modelled and therefore are more efficient as the size and complexity of the problem increases. Second order methods, such as the Levenberg-Marquardt algorithm allow much faster training than the first order methods such as the standard Gradient Descent Backpropagation method and also offer an advantage when the problem is ill-conditioned.

From the study of literature it was found that the pre-processing of network inputs is essential and affects the network's ability to generalize. Network complexity also plays an important role and may cause ill-conditioning if overdone. The ability of the network to generalize on data not seen during training is the measure of successful implementation. There are a number of avenues that can be explored in this regard.

Chapter 3: Renewal Dataset

3.1 Introduction

The performance of neural network methods on reliability data was tested through the use of two separate datasets. One of these datasets was generated through a series of laboratory tests that was planned and executed at the Sasol Laboratory for Structural Mechanics which is located at the University of Pretoria.

The aim of the laboratory work was to generate a renewal type dataset of good quality that would be suitable for testing the relative performance of different neural network architectures and training methods. The reliability scenario that was simulated during laboratory testing was modelled as far as possible on an actual reliability related problem that was encountered in industry.

A number of test pieces were subjected to a constant cyclic loading pattern which caused fatigue cracking in the region of highest stress concentration leading to eventual failure. The dataset generated conformed to the renewal assumption due to the negligible deterioration in the condition of the hydraulic test rig which was used in the laboratory. The test piece was replaced upon failure returning the system to good-as-new condition. This procedure proved to be a practically workable solution for generating a large enough dataset for the successful application of neural network techniques.

The decision to perform such a series of laboratory tests to generate a dataset was motivated by the great difficulty which was previously experienced in finding suitable equipment reliability related datasets. The training of a neural network requires a dataset of sufficient size to be successfully applied, because of the danger of ill-conditioning. This issue has been dealt with in greater detail in Chapter 2 and does not require further attention.

3.2 Test Motivation

The experimental setup was inspired by an actual maintenance situation encountered at a South African mine. Problems were experienced with a number of double toggle jaw crushers (see Figures 26 and 27) which are installed at an underground location and serve to crush feed material consisting of banded carbonite, foskorite, transgressive carbonate, dolerite and micaceous pyroxenate. Material is fed into the crusher that has a feed opening of 2100mm x 1675mm at a rate of 400m³/h. The exact details relating to the situation are beyond the scope of this work, and it will suffice to give a brief description of some of the relevant technical aspects.

The specific type of crusher (Figure 28) utilizes a large belt driven flywheel and shaft arrangement. On this shaft there is an eccentrically mounted crank, called a pitman, which serves to convert the rotational motion from the flywheel into linear motion. The force of the crank is multiplied through the scissor action of two inclined plates, called toggle plates, causing a reciprocating motion of the crusher jaw. The material is crushed increasingly finer as it flows under gravity through the tapered crusher opening between the jaws. One of the toggle plates that forms part of this linkage system that transfers forces via the crank mechanism from the large driven flywheel to

the crushing jaw is manufactured with a notch. The notched toggle plate (see Figure 28) serves as a safety device which is designed to fail in cases where large foreign objects that cannot be crushed are trapped between the jaws and the resulting forces would exceed the design parameters of the machine. Failure of the toggle plate prevents the stored energy of the flywheel from causing substantial damage to other components.

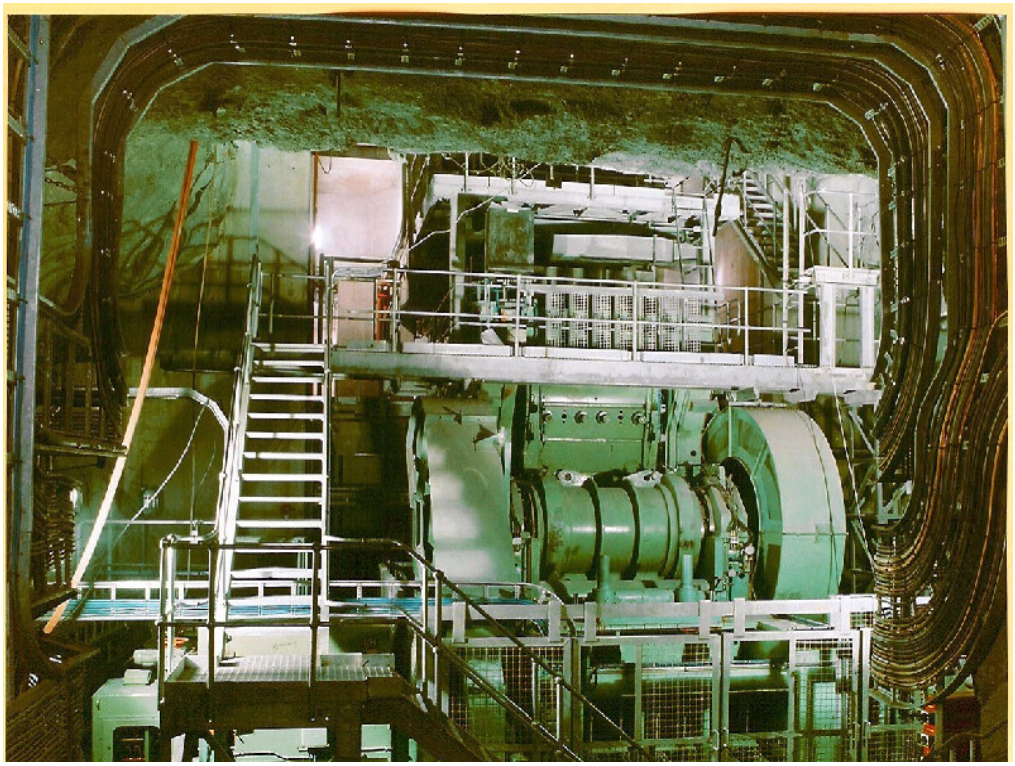

Figure 26: The Jaw Crusher in its Underground Location

The toggle plates are mounted in seats and are designed to move by a rolling action, restricting the loading on them to a purely compressive nature. Should slippage occur at these interfaces, bending moments are however introduced.

Unfortunately, in this particular application, the double-toggle jaw crusher suffered numerous failures of its toggle plates (see Figure 29), raising questions about the technical soundness of the design. Due to their enormous size, the toggle plates are replaced after each failure only at great expense.

The opportunity therefore arose for laboratory testing through which a reliability dataset could be generated for the purpose of this work. The lab testing also served the purpose of gaining experience and investigating potential measurement procedures to be used in future on this application and the verification of the toggle plate design as used in these large underground crushers. Some work had already been done previously where the actual loads on such a plate were measured. This data was however collected during normal operation and it was important to find ways of measuring and capturing the magnitude of the abnormal loads that are experienced by the toggle plate upon entry of a foreign object.

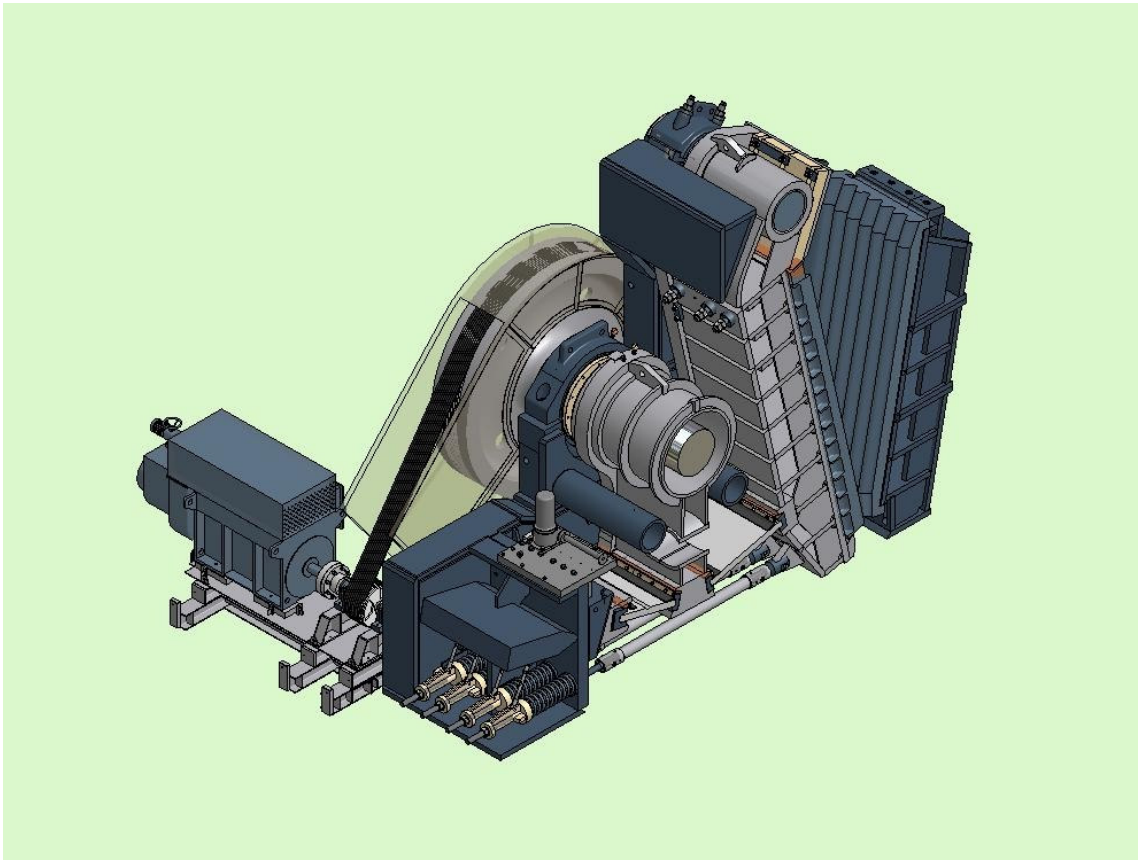

Figure 27: A three-dimensional image of the crusher

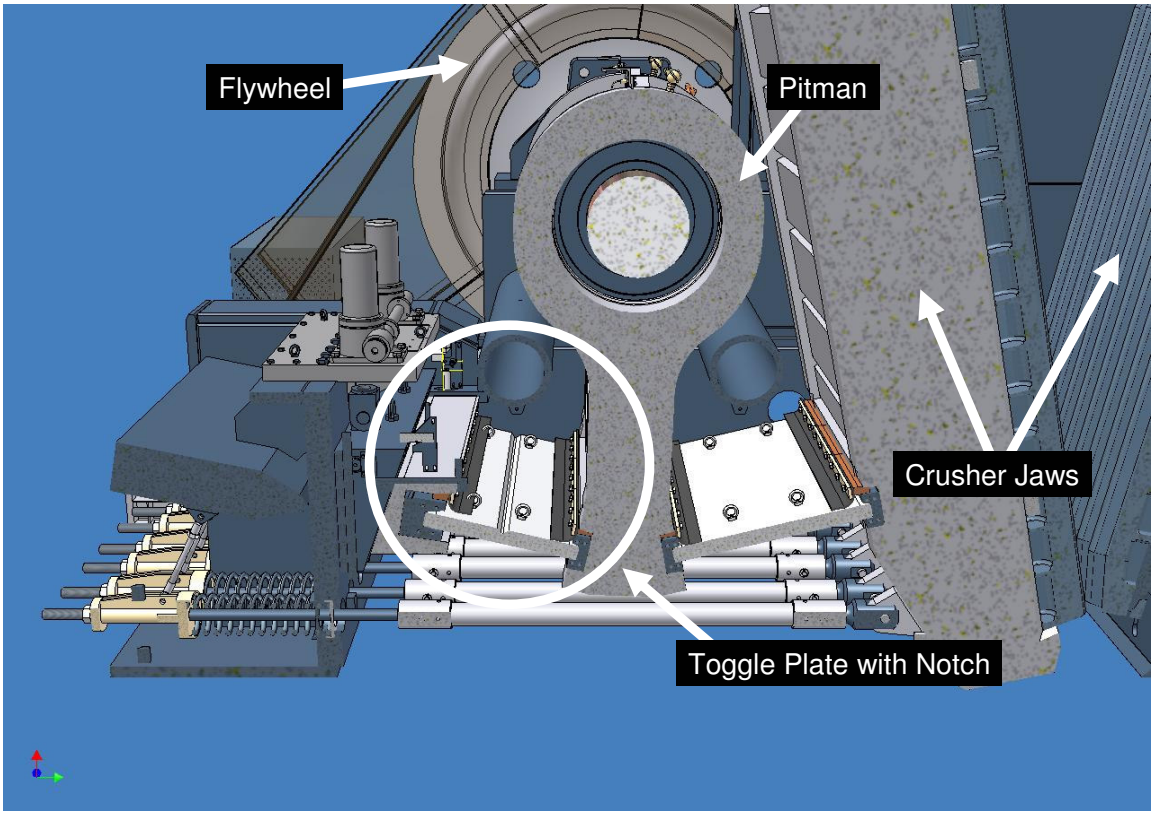

Figure 28: Position of the notched toggle plate

3.3 Test Piece

The test piece (Figure 30) and experimental setup (Figure 32) were designed to resemble the real life toggle plate installation as closely as possible. Restraints on data processing, cost aspects and time availability did however exert an influence on the design as it had to be ensured that the data generated in the process would be sufficient and could be processed to yield meaningful results.

On the crusher, the ends of the toggle plate are supported in seats that allow a rolling movement of the toggle plate ends and ensure that the forces transmitted by the toggle plate are purely compressive. The large quantity of tests that were required meant that an elaborate mounting arrangement for the test piece was out of the question and the ends of the test piece were therefore rigidly clamped by the top and bottom jaws of the test rig. This departure from the real mounting arrangement meant that the forces which were applied to the test piece also resulted in a bending moment because of the clamped ends. The realism of the lab tests is therefore limited to an extent by the nature of equipment available for testing.

It was the aim to achieve uniformity in the measurements by ensuring that all the test pieces were subjected to reasonably similar conditions. The number of variables exerting an influence during testing determines the size of the dataset that is required to give an adequate coverage of the different cases and input combinations that may occur. If the number of variable factors is therefore reduced, the number of possible combinations that have to be investigated is reduced and a corresponding reduction in the number of required test runs is achieved.

The complexity of the network architecture also depends on the number of failure modes and variations that it needs to recognize. This can be compared to the mathematical case where the number of equations that are required to solve simultaneously for a given number of variables increases as the number of variables gets larger. It was an important consideration therefore to reduce the number of variables as testing timetable was tight at all times and had to accommodate both lab availability and the work load of the student at his place of employment.

The simplicity of the test sample was pivotal in ensuring that these units could be manufactured within the limitations of a tight budget. The requirement for machining was reduced to a minimum through a simple design, thereby reducing the cost of manufacture. The notch was milled into the parent plate and then the test pieces were cut by removing strips off the plate by means of a saw. The test pieces were also manufactured of standard structural steel, which is readily available and can be machined easily.

Figure 29: The cracked notch section of an actual toggle plate

The actual toggle plates have the dimensions of 1.8 by 0.9 meters and are 90 mm thick. The design of the test samples represented a scaled-down version of the actual toggle plate used in the crushers, and were designed to fit into the laboratory test rig. It was an important criterion that the scaled-down version of the toggle plate should accurately reflect the properties of the original.

The existing toggle plate is made of high-strength steel called C60. The end caps are hardened, but the area of the notch has the original steel properties. For this type of steel, properties of sections with different thicknesses are not the same. A plate with a thickness between 40 and 100mm thickness is specified with a yield strength of 450 MPa, and ultimate tensile strength of 740 – 890 MPa. If a smaller sample with a thickness between 16 and 40 mm is used, a yield of 490 MPa and ultimate tensile strength of 780 – 930 MPa can be assumed. A test piece as the one envisaged here, would therefore have slightly different material properties to the actual toggle plate. The choice did however eventually fall on standard structural steel (300WA), due to its availability, the ease of working with this material and also its lower cost.

Apart from dictating the maximum physical sample size in that can be accommodated within the test rig, the limitations of the laboratory equipment also introduced restraints in terms of the load magnitude and load cycle frequency that could be applied. The test pieces had to break within a reasonable time period under the loading conditions generated by the actuator.

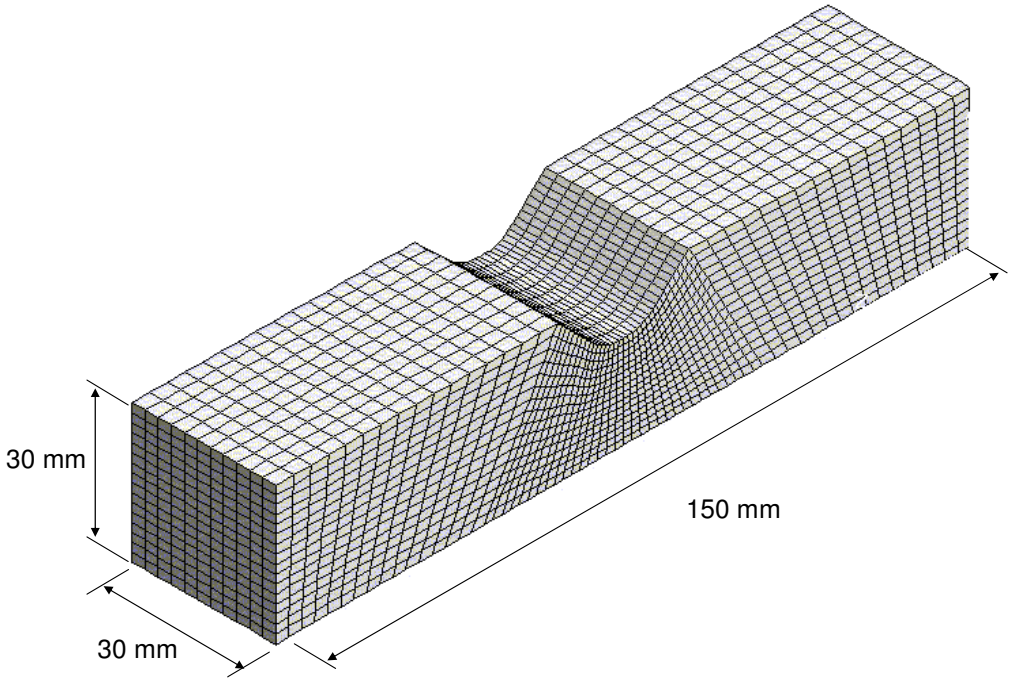

Figure 30: Dimensions of the test piece.

It was not feasible to use a process of trial and error to find the ideal test piece size, as a whole batch of test pieces had to be manufactured at once. Insight therefore had to be gained into how the test sample would respond to the testing in an alternative and cost effective way. Fatigue calculations, as per Shigley (1986), were used to generate a rough estimate the expected life, and to set the level of force applied to the test piece.

According to Shigley low cycle fatigue failure for steel is commonly accepted to take place between 0.5 and 1000 cycles. A finite life region is identified between 1000 cycles and a transition point that lies somewhere between 10^6 and 10^7 cycles. This transition point constitutes an endurance limit. When stresses drop below this level, an infinite fatigue life can be assumed. It was the intention to conduct testing within the finite life region, aiming at a failure time of less than 8 hours.

Finite element analysis (FEM) was done on different test piece designs in order to establish the magnitude of stresses in the notch area for each test piece design. The input forces were varied in order to find an equivalent loading condition to that encountered in the actual situation. Mean stress equations have been proposed by Soderberg, Goodman, Gerber and Morrow and the reader is referred to Shigley (1986) for further reading on this subject. In order to calculate finite life for a fluctuating stress condition, the endurance limit can be replaced by a fully reversed alternating stress level in any one of these methods. When the Haigh diagram is extrapolated into the region of compressive mean stresses, it is found that changes in the compressive mean stress has no effect. The stress amplitude can therefore be directly applied to the S-N curve in order to obtain an expected fatigue life. If the compressive mean stress increases sufficiently, however the component will fail due to buckling long before fatigue can play a role.

The fatigue calculation procedure requires the use of a stress concentration factor, which can be attributed to the effect caused by the notch. This may be determined

experimentally, but for the sake of this work a finite element model was generated of the test piece. The solution of the model, subjected to compressive loads similar in nature to those expected in the test rig, provided a value for the concentration of stress in the notch. This method allows for the quick and cheap calculation of a stress concentration factor and offers great flexibility in changing the component shape.

Modelling for the FEM analysis was done with thick shell quad elements. A depth dimension was assigned to elements which had been generated by meshing a two-dimensional surface. In order to check the validity of this method, a further model was generated with solid elements (Figure 31). The results were very similar, thereby validating the results achieved with the shell elements.

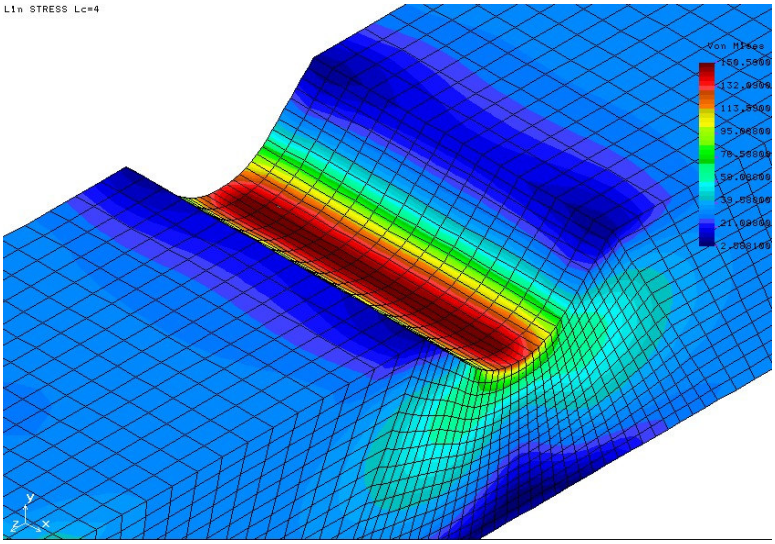

Figure 31: Finite element analysis of a test piece with a 10mm notch. The applied load is 40kN.

The shape of the notch in the test samples was scaled down from the design of the toggle plate. Finite element models were made of samples with different notch radii, to investigate the effect on the theoretical stress concentration at the base of the notch. It was found that the theoretical stress concentration factor K_t remains constant, independent of load if the ratio of notch radius to plate thickness is kept constant. Therefore the theoretical stress concentration of a notch with a radius of 10mm in a 30mm thick plate is the same as a notch with a 30mm radius in a 90mm thick plate. This conclusion can be expected as the theoretical stress concentration values given by Shigley (1986) for notches with a different shape show a similar tendency.

A stress concentration factor of approximately 3.4 was calculated for both the test sample and the actual toggle plate. For the toggle plate material (C90) a notch sensitivity of 0.88 is assumed. This factor reduces to 0.8 for the 300WA steel used to manufacture the test pieces. (Shigley (1986) Figure 7.13)

Using the following equation given in Shigley (1986):

$$K_f = 1 + q(K_t - 1) \qquad (3.1)$$

the stress concentration factor is found to be:

$$k_e = \frac{1}{K_f} \tag{3.2}$$

Fatigue calculations using the calculated stress concentration factor were subsequently used to choose the forces that were applied to the test pieces in the laboratory, which allowed the duration of each test run to be controlled.

3.4 Laboratory Equipment and Test Procedure

Quite a number of test pieces as discussed in the previous section were run to destruction in the university laboratory. The complete testing setup including the hydraulic test rig, the control system and the data collection equipment are shown in Figure 32 and Figure 33.

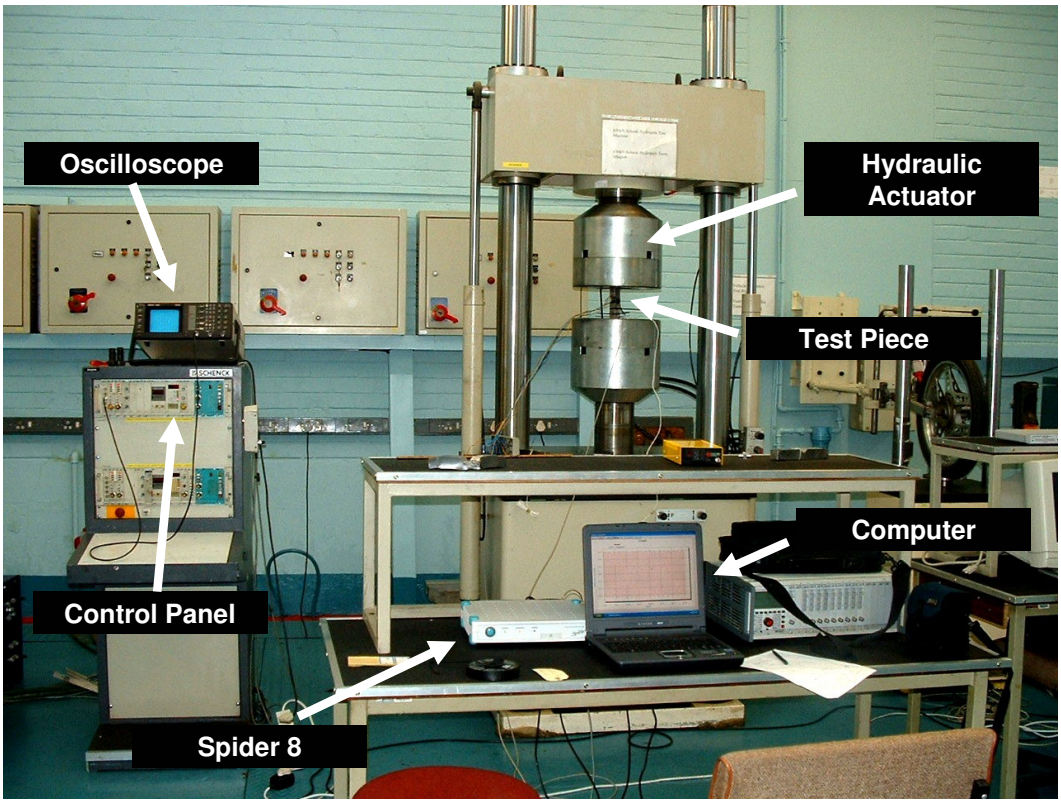

Figure 32: Photograph showing the various elements involved in the lab testing.

The test runs were performed through the use of a hydraulic test rig which applied cyclic forces to a test piece which was clamped into position between two hydraulically operated jaws. The actuator is capable of exerting a maximum force of 630 kN, though the actual load that can be effectively applied is limited by the frequency of the loading pattern. Cyclical loading was applied according to a sinusoidal pattern of which the mean and amplitude were varied by means of the actuator's control system thereby generating different operating conditions for the series of test runs and producing a varied dataset. The loading pattern was applied at a frequency of 3 Hz which was close to the upper limit of what could be achieved while still allowing the actuator to apply a suitably high load.

The actuator was controlled in displacement mode, which means that the displacement remained constant throughout each test run. A constant displacement throughout the test means that the applied load will change once plastic deformation and cracking occurs. At the outset of each test run, the desired mean and amplitude of the applied load pattern were chosen and the displacement of the machine was set to duplicate this loading pattern on the undamaged test piece. The displacement setting was performed manually by adjusting the machine settings until the desirable force reading was displayed on the oscilloscope screen. The force reading was obtained from the load cell that forms part of the test rig and measures the applied force of the actuator arrangement. Both the mean and amplitude of the input signal were adjusted in this way.

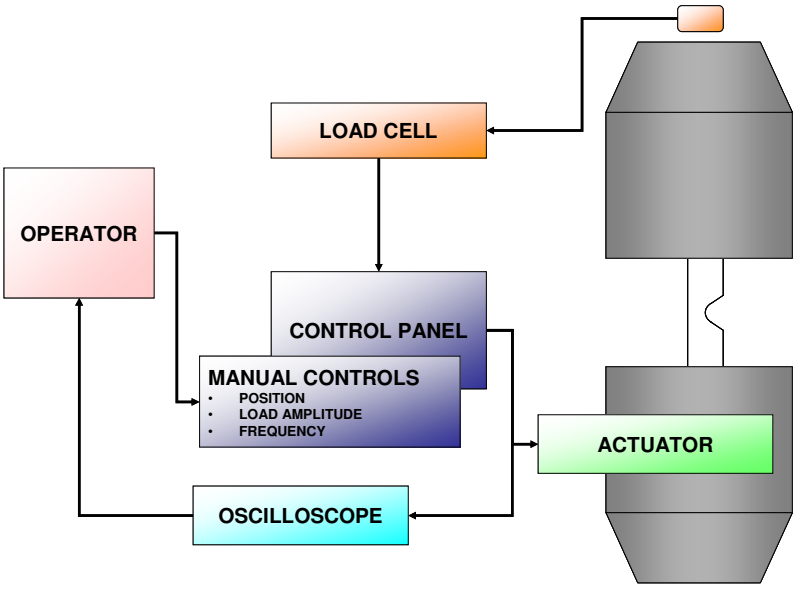

Figure 33: Control of the testing process

The benefit of using the test rig in this mode of operation is that the force amplitude decreases during the period of crack propagation, while the displacement of the jaws is maintained. The deteriorating condition of the test piece can therefore be clearly detected from a reduction in the load cell readings.

It was unfortunate that problems were sometimes experienced with the rig as it occasionally failed to maintain the desired mean of the loading pattern. The force amplitude, however, did not provide any problems and remained within the parameters set before each test. The result of the mentioned wandering of the loading pattern's mean was that the data obtained from such a test run could no longer be used.

The clamping of the test pieces in the actuator jaws was of a rigid nature. This method of fixing the test piece introduced bending stresses in the notch area, which should not occur in the crusher application during normal operation. The rolling action found in the toggle plate seats of the crusher was therefore not duplicated in the test arrangement. A pinned connection could have been used but the prohibitive cost implications of manufacturing the more complex test pieces and also the greater difficulty of installation would have been unacceptable.

3.5 Sensors and Measurements

A diagrammatic representation of the measurement setup is shown on Figure 34. The Spider 8 is a multi-channel device which allows a number of measurements to be taken in parallel. The data is subsequently fed to the laptop computer where the measurement software captures the readings and exports the data in the form of text files after each measurement cycle. These files can at a later stage be imported into a spreadsheet application where the raw data is pre-processed and transformed into a suitable format and saved again as a text file. The data is then ready for use for the training of the various neural networks.

Four different sensors (see Figure 34 and Figure 35) were selected and used for the taking of measurements during each such measurement window. The choice of sensors was aimed at tracking the test piece deterioration, but also to provide a measure of operating conditions which influence the test piece's life.

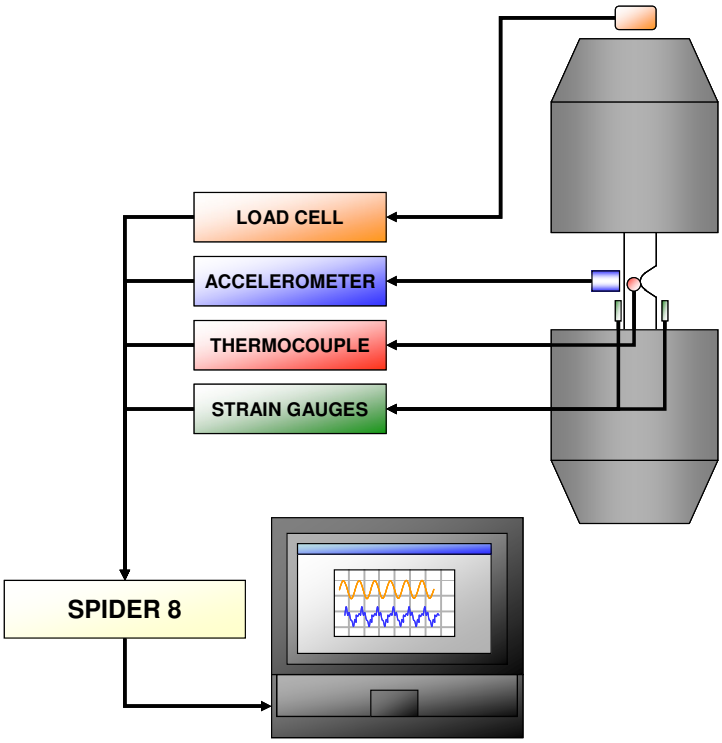

Figure 34: Diagram showing the location of sensors for measurement and the equipment used for the capture of data.

It has already been indicated that the test rig was run at a frequency of 3 Hz. When the frequency is set too high, the full force cannot be applied to the test piece due to test rig inertia. Measurements were recorded over periods of 3 seconds at 3 minute intervals and the testing proved that both the time interval between the taking of measurements and the duration of the recording window to be satisfactory. At a frequency of 3 Hz it meant that the data for a total of 9 complete actuator cycles were captured in each measurement window, in which a sequence of 1800 samples was taken during the three second period.

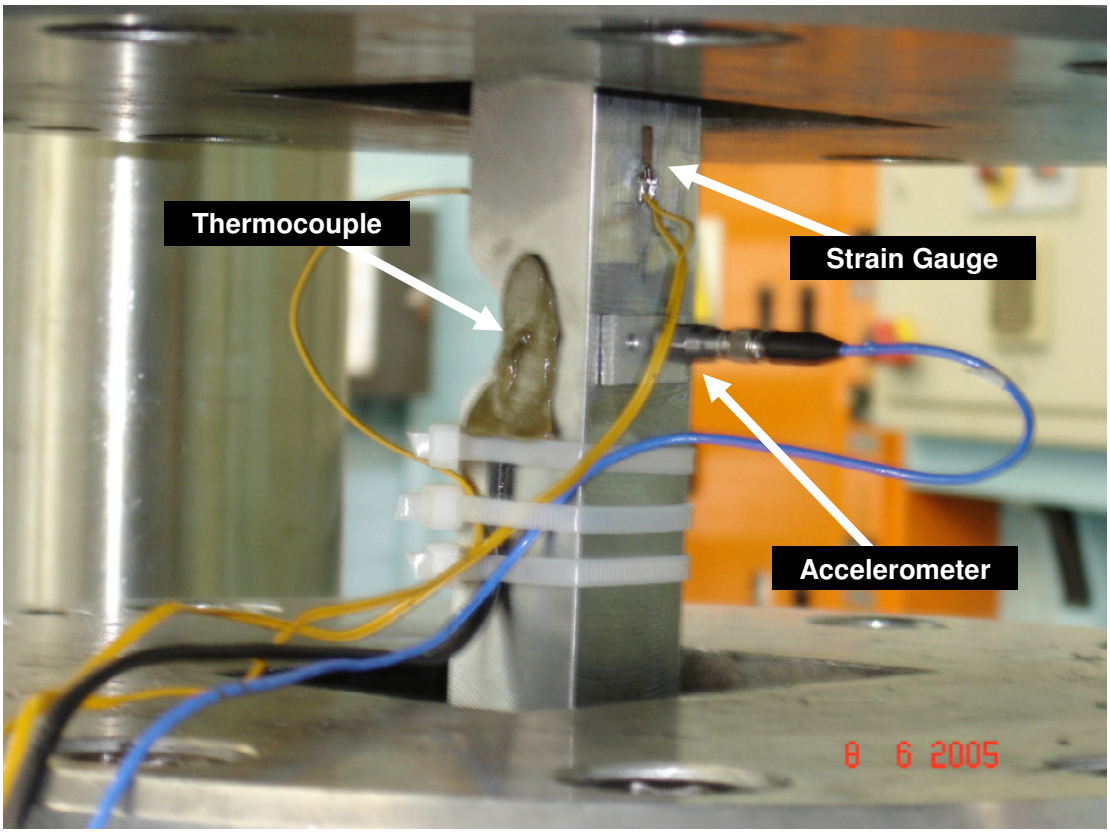

Figure 35: The location of the various sensors on the test piece.

3.5.1 Strain Gauge

In view of potential future measurements on the actual toggle plate, strain measurements were taken on the test pieces. Uni-axial strain gauges where selected above more complex units, such as rosette gauges, in an effort to reduce costs. To eliminate the effects of bending of the test piece on the strain measurement, an arrangement was chosen that serves to achieve this purpose (Van Tonder, 2004).

In Figure 36 the items labelled R_1 and R_3 are uni-axial strain gauges mounted on the opposite sides of the test piece. R_2 and R_4 on the diagram are passive gauges completing the Wheatstone Half-Bridge arrangement (Figure 36). The two uni-axial type strain gauges were applied to opposite sides of the test piece.

For the chosen layout the effective strain at the point of measurement is given by:

$$\varepsilon = \frac{1}{2} \cdot \frac{4}{k} \cdot \frac{U_A}{U_E} \tag{3.3}$$

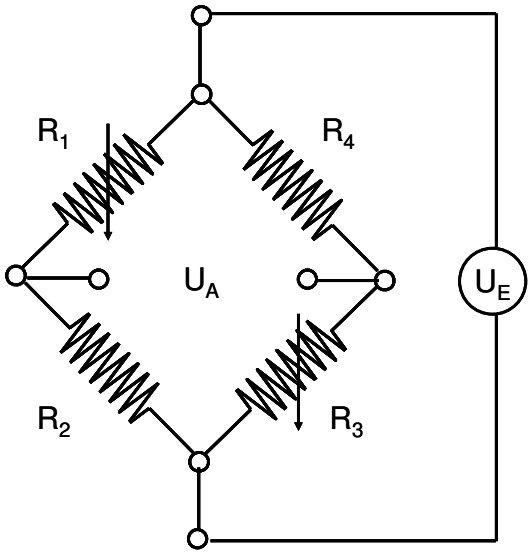

Figure 36: Diagram of the Wheatstone half-bridge arrangement that was used for the measurement of Strain.

The passive gauges that are required by the chosen type of arrangement were mounted on a spare test piece as shown in Figure 37.

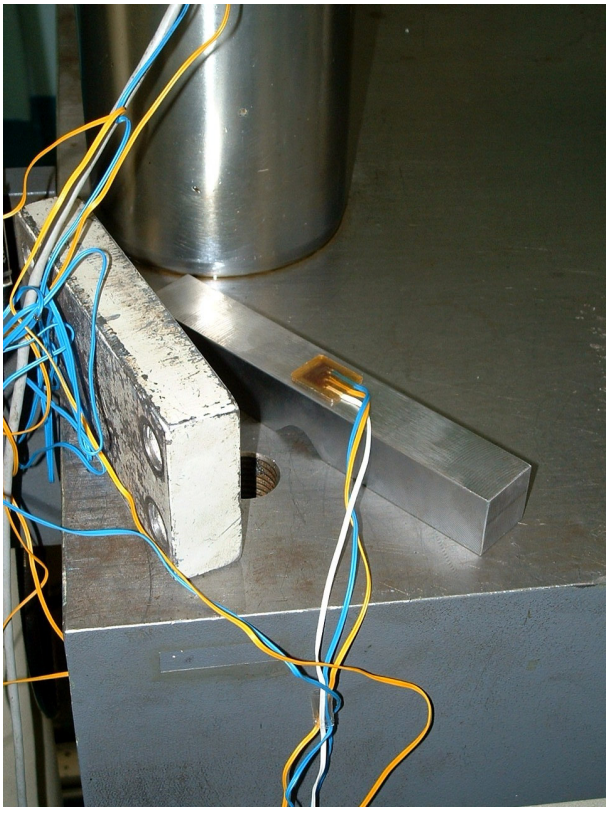

Figure 37: Test piece on which the passive gauges where mounted.

The application and connection of the strain gauges to such a large set of test pieces proved to be a laborious task which slowed down the rate at which testing could be done. This was a serious drawback, as the time available for testing was limited. Problems were also experienced with the adhesion of the gauges to the test pieces, which regularly came loose after installation in the test rig and also during the testing. As the data for all the test pieces must have a similar format when training the neural network, data collected during an abortive measurement run would have to be discarded. The first tests also proved that the load cell readings took a similar form to the stain gauge readings and it was therefore decided to continue the testing process without strain measurements.

3.5.2 Accelerometer

An accelerometer was mounted on the test piece on the opposite side to the location of the notch. The aim of this measurement was to measure the movement due to the deflection of the test piece under loading. It was also envisaged that changes would possibly occur in this movement once cracking started in the notch area. An aluminium block with a tapped hole was mounted onto the test piece by means of an adhesive gel. The accelerometer was then secured on this block.

The high measurement frequency of 600 Hz was chosen to ensure that enough data was collected for the generation of a vibration frequency spectrum from the accelerometer readings. The position chosen for mounting the accelerometer was found to be unsuitable for the intended purpose. The readings were very small and were truncated by the data recording process during initial phases of testing. The outputs captured on the data file were truncated after the third decimal place which prevented any further processing. Accelerometer readings remained problematic throughout testing, even though the switch to a more sensitive probe improved the quality of outputs. Despite the elimination of the problem through the use of an accelerometer with greater sensitivity (100mV/g), there were only meaningful readings for a reduced proportion of the test pieces. The data from this source was therefore also discarded as the number of datasets would have been less than the essential number required for the successful training of the neural networks. The repetition of test was also not an option due to the time constraints that have been mentioned above.

3.5.3 Thermocouple

Temperature on the surface of the test piece was measured by means of a thermocouple, which was mounted with adhesive putty on the side of the test piece. For the sake of convenience it was positioned halfway along the cross section and aligned with the centre of the notch. It was expected that the heat generated during the process of crack propagation could be detected at this location.

It was found that the temperature which was measured at this position increased dramatically once crack propagation started as is illustrated by Figure 38. The temperature measurement was therefore found to be a very useful indicator of test piece condition and gave a good indication of imminent failure.

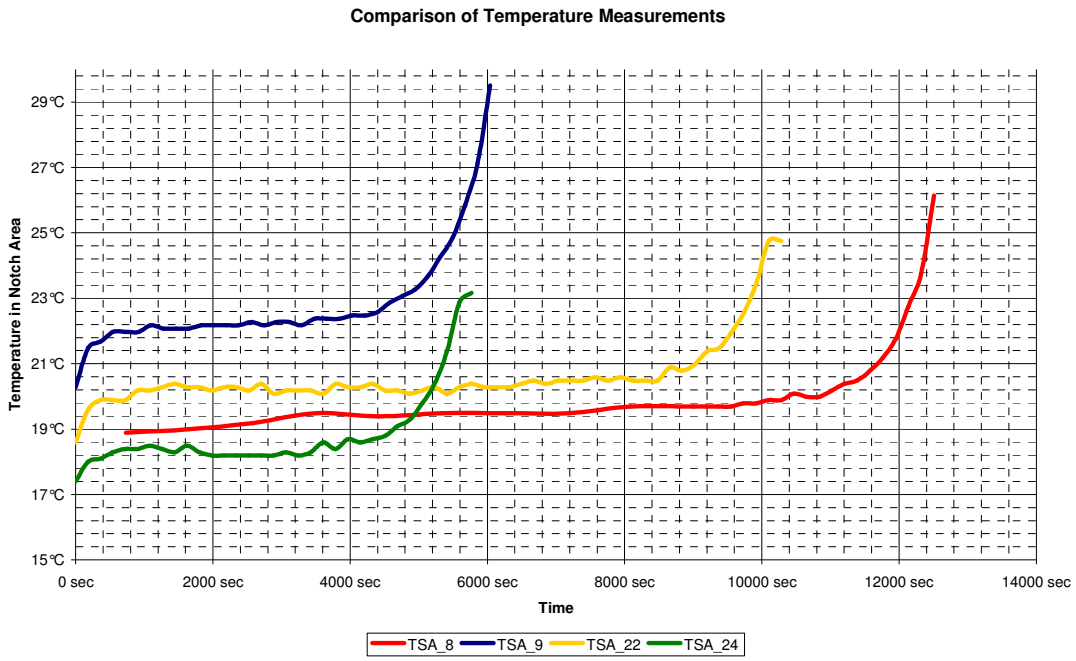

Figure 38: Graph showing the increase in temperature measured in the notch area during the period of fatigue crack propagation.

Table 3.1 gives both the initial temperature measured at the start of the test run and the maximum temperature measured during the tests. The thermocouple readings were naturally affected by seasonal temperature differences between summer and winter, as well as daily temperature fluctuations. It can be seen from Table 3.1 that there were significant differences in baseline temperatures for the various test pieces. It was found that the difference between the initial and the maximum measured temperatures ranged from 6.2 °C to 15.6°C.

The initial temperatures are a reflection of the ambient temperatures in the laboratory at the start of each test. These initial temperatures, influenced both by seasonal and daytime variance in temperature at the test run start, were found to range between 14.5°C to 25.5°C. The initial temperature does not however have a significant effect on the expected life of the component.

When the increase in temperature is plotted against the life of each component (Figure 39), it becomes clear that greater temperature increases where recorded for units that were subjected to higher loading and therefore more rapid failure. The rate of temperature increase and the magnitude of the increase above the initial measured temperatures therefore serve as an indicator of the rapidity with which failure will occur.

Table 3.1: Initial and maximum temperatures measured for each test piece during the test run.

No.	Data ID	Initial Temperature	Maximum Temperature	ΔT
1	TSA_2	23°C	38.6°C	15.6°C
2	TSA_3	25.5°C	37.5°C	12°C
3	TSA_5	17.7°C	25.6°C	7.9°C
4	TSA_8	18.9°C	26.1°C	7.2°C
5	TSA_9	20.3°C	29.5°C	9.2°C
6	TSA_20	18.5°C	24.7°C	6.2°C
7	TSA_22	18.6°C	24.8°C	6.2°C
8	TSA_23	19.4°C	26.1°C	6.7°C
9	TSA_24	17.4°C	23.2°C	5.8°C
10	TSA_25	18.5°C	30.1°C	11.6°C
11	TSA_26	14.5°C	22.3°C	7.8°C
12	TSA_27	17.7°C	24.9°C	7.2°C

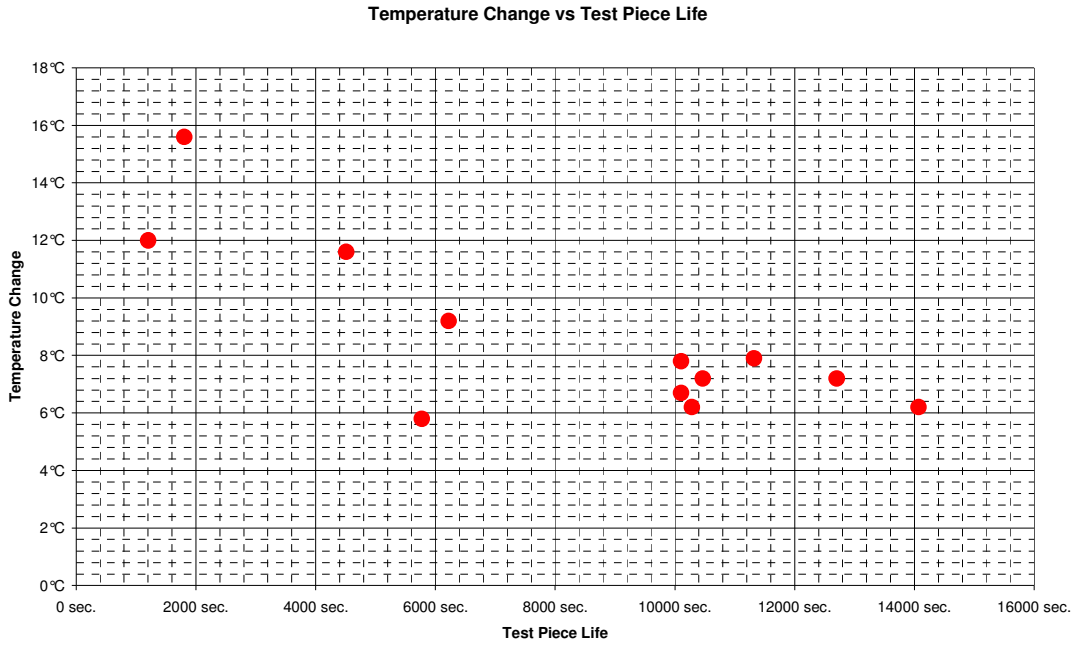

Figure 39: Graph illustration the relationship between the rapidity of failure and the magnitude of the temperature increase measured during a test run.

3.5.4 Load Cell

The actuator of the test rig was set to maintain constant amplitude in the oscillation of its jaws. The amplitude was varied for the different test runs thereby altering the operating conditions to which each test piece was subjected. A specific initial load could be applied by increasing the displacement of the jaws at the start of a test run until the required load cell reading was attained. As the cracking of the test piece in the notch area caused weakness, the force required to maintain this amplitude was reduced and this could be observed in the corresponding drop in the magnitude of the load cell measurements that were taken. Figure 40 below shows the trend of how amplitudes of the applied load are reduced and it is clear that this data is also a useful source of information on test piece condition.

Apart from the change in amplitude of the applied load, the initial amplitude of loading gives an accurate indication of what the expected life of the test piece is. In both cases, the load cell readings provide important insight into the residual life of the component that can be expected. A summary of the initial measured amplitude and mean of the load pattern exerted on each test piece and the actual life that was attained is given in Appendix A.

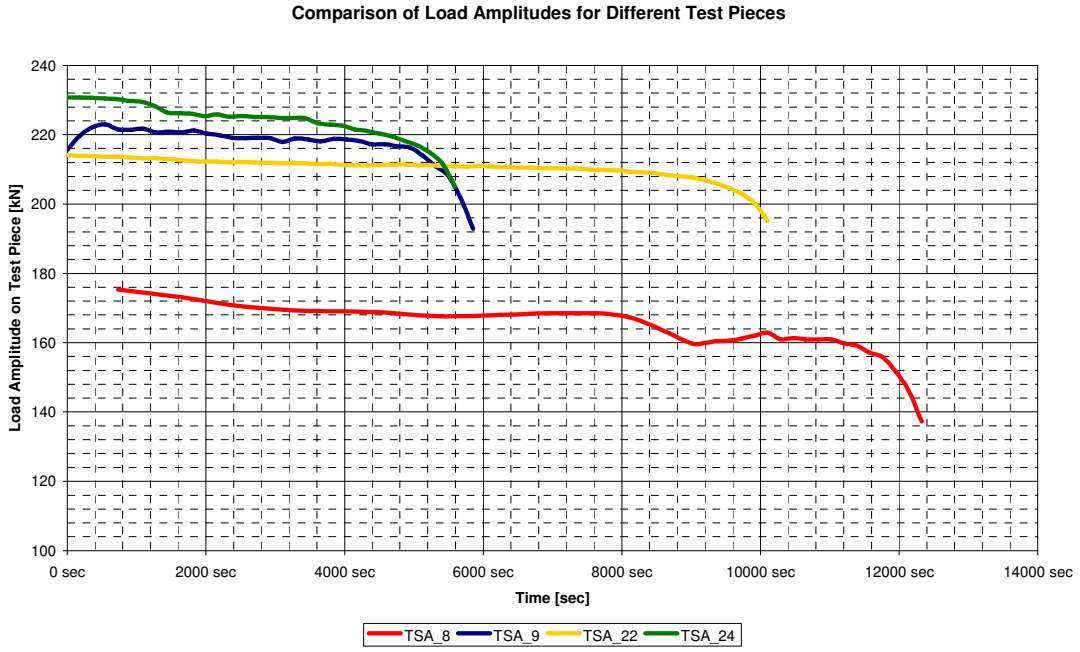

Figure 40: Graph illustrating the drop in the amplitude of the applied load pattern once cracking occurs.

3.6 Discussion

A series of test runs were conducted and a set of data was collected for each test piece that was subjected to fatigue loading in the laboratory. Among the difficulties encountered during testing was the failure of the test equipment, the discontinuation of a testing due to time constraints and in some cases the loss of data due to the failure of sensors during a test run.

It was decided at the time of testing to discard the datasets from tests that had to be discontinued. One of the main reasons for this decision was that all parameters of the

test run changed once the test equipment was restarted. The reinstallation of the test piece, the readjustment of the testing equipment during start-up and the impossibility of achieving the same loading conditions as were used before the discontinuation brought about a substantial change which would affect the expected life of the test piece. The discontinuity in the data caused by such an event also would have made it difficult for the neural network to differentiate between different cases, especially with the type of inputs chosen for this project.

Baseline measurements, taken at the start of each test run, were extensively used for the purpose of network training and these become irrelevant once testing has to be resumed after an interruption. Ambient temperature, for example, sometimes changed drastically between two test runs. Shigley (1986) recommends that an adjustment factor is introduced in fatigue calculations only when the component is subjected to temperatures above 350°C. The highest measured temperature was only 15.5°C above the initial room temperature and it can therefore be assumed that temperature related effects had no influence on the expected life of the test pieces. It would however have required adjustments to the method of data processing and network training to accommodate such changes in ambient temperature.

During some of the test sessions, the mean load exerted by the test rig did not remain constant for the entire duration of testing. This fluctuating load condition generated unpredictable data for a training set this small. Finally, errors also occurred during the measurement procedure, resulting in an unusable dataset containing spurious data.

The poor accelerometer readings were unsuitable for spectral analysis and were discarded as a result. The thermocouple readings were on occasion either unrealistically high or low to the point of saturation. As the temperature remains constant over such a short measurement interval, these anomalies could be filtered out by choosing either the minimum or the maximum in the measurement window, depending on which provided a realistic reading.

The series of tests stretched over a period of more than a year with the first test run on 11 May 2004, while the last test run only took place on 8 June 2005. This extended time span impacted negatively on testing strategy, which sometimes lacked the required cohesiveness. Some continuity was therefore lost during each of the long layoffs between testing. A total of 30 test pieces were manufactured, of which 28 where damaged or destroyed during testing. 5 test pieces where destroyed through errors that occurred during installation, while 6 test runs were interrupted or suspended. One of these interruptions was due to a failure of the hydraulic system of the test rig. A further dataset was discarded because of continuous and substantial drift in the applied loads of the test rig. Measurement and data processing errors eliminated a further three test sets while it was decided to discard the measured data of another test piece because of the large discrepancy in its operating conditions when compared to the other units. The remaining 12 datasets where used for the training and testing of the neural networks.

The laboratory testing proved successful in its main purpose of generating the required dataset that could be used for training neural networks. Both the temperature and load cell readings gave a good indication of the deterioration of the test piece's condition which was one of the most important requirements. Enough data was also generated in the laboratory to ensure that neural network methods could be effectively applied.

Chapter 4: Life Prediction for the Renewal Dataset

4.1 Neural Network Selection

After the introduction given in the previous chapters to the analysis of reliability data, a background on neural networks and the testing performed at the university laboratory, the actual application of neural network methods to reliability data is now discussed. The work was performed in two phases. The first phase involved the writing of a neural network platform using Visual Basic for Applications (VBA) and running within the spreadsheet environment of Microsoft Excel. During the second phase, use was made of the mathematical programming environment MATLAB, which contains a neural network toolbox offering access to a large variety of neural network types.

The initial work involved the generation of a MLP network with backpropagation training enhanced through the use of a momentum term. The advantage gained from such an individually tailored system was the ease with which inputs could be processed and adapted. As a result, it greatly facilitated the finding of an optimum combination of inputs and represented a useful learning process. Rapid changes could be made to the inputs and how they were normalized. The spreadsheet platform assisted with other forms of pre-processing, such as the extraction of maximum values and allowed the graphing of trends for the vast amount of readings that were collected.

This first network proved invaluable in the investigation of network behaviour as different input combinations could be tested and the effect of varying the number of hidden nodes could be determined. It was also possible to assess the effect of various pre-processing techniques on network performance and derivatives of the programme were later used to pre-process and export data for use in MATLAB.

The initial choice of standard multi-layer perceptron architecture was made because of simplicity and also the ease of implementation. Both the single hidden layer and the output layer were given sigmoid transfer functions. Training was performed with a standard backpropagation algorithm which was modified through the addition of a momentum term. Both the learning rate and momentum constant could be changed before training, but remained constant thereafter. The number of inputs, outputs and nodes in the hidden layer could also be varied in this way, thereby providing some flexibility for experimentation. The data that was fed to the neural network was pre-processed using a procedure that could be easily updated. Data was presented to the network as a batch and weights were updated after every batch presentation.

Though the spreadsheet environment proved to be an excellent basis for experimentation, it proved to be rather slow. Training which took between 20 and 40 minutes in Excel could later be accomplished in a few minutes using MATLAB. Due to this slowness of network training and also slight deficiencies in the fit that could be achieved with standard backpropagation, it was decided to change the training of the network to the Levenberg-Marquardt algorithm. The implementation of the

Levenberg-Marquardt training algorithm (see Chapter 2) failed due to software limitations. The matrix manipulations that had to be performed were very time consuming within the framework of a spreadsheet and resulted in even slower training than before. The complexity of the programming procedure also made it difficult to find the reason for the algorithm's failure to converge.

It was clear that further work in this direction was likely to prove fruitless and time consuming and an alternative platform had to be found. The neural network toolbox which is available within the MATLAB software framework was subsequently chosen for the remaining work, as the author had previous programming experience with this software. Demuth and Beale (1998) discuss the neural network types and training algorithm variations which form part of this toolbox, of which five where selected and trained on similar data for the purposes of comparison. Three MLP networks and one RBF type network were used.

The first order gradient descent learning algorithm serves here as the basis of comparison between the different neural networks due to its historic significance. Adjustments were made to the learning rate and a momentum term was introduced which increases the rate of convergence of this algorithm. The performance of the gradient descent algorithm is compared with the much faster second order Levenberg-Marquardt algorithm which outperformed other fast techniques according to the findings of Hagan and Menhaj (1994). Bayesian regularization was applied in conjunction with the Levenberg-Marquardt algorithm to investigate the effect of this method which is aimed at improving generalization. The general regression neural network (GRNN), which was also used by Luxhøj (1999) for his research, offers the advantage of rapid unsupervised training. It is also of interest because it is a network with radial basis function (RBF) architecture which contrasts with the MLP architecture of the networks that have already been mentioned.

Each network was constructed with 5 inputs and generated a single output which represented an estimate of remaining life to failure measured in seconds. The MLP networks were each constructed with 5 nodes in the hidden layer such that the basic network structure was similar for each of these networks. The size of the hidden layer was initially optimized during the first phase through an empirical process where the number of nodes in the hidden layer was varied. Table 4.1 gives a summary of the different neural networks that were used.

Table 4.1: Neural network types that were used.

No.	Neural Network Description
1	MLP neural network with a gradient descent backpropagation training algorithm with momentum term
2	MLP neural network with Levenberg-Marquardt training
3	MLP neural network with Levenberg-Marquardt training and Bayesian regularization
4	General regression neural network

4.2 Pre-Processing and Network Inputs

The importance of pre-processing for network performance and generalization has been discussed in Chapter 2. Numerous options such as grouping, feature extraction and the re-scaling of data can be used to transform the raw data in such a way that network performance is optimized.

A static split was chosen as the method for comparing network performance on the renewal dataset. This could be done due to the simplicity of the simulated maintenance setup in the laboratory, for which there was only one failure mode. The lab data collected during testing was split into two groups with nine of the datasets used for training and the remaining three comprised the testing set. The first and last readings of each test run were discarded as the first reading often still reflected transient effects from the starting up process before the system had settled, while the last reading was taken after failure.

Due to the large number of readings in each interval, the first step involved the extraction of the data for each interval. Re-scaling of the input data formed the basis of the pre-processing that was then performed. The data was first normalized and then adjusted to fit into an interval between zero and one. This helped to prevent any one input that has a much greater numerical size to the other inputs from dominating during training. The use of a sigmoid function in the output layer also requires that the outputs used for network training must fall between zero and one.

$$\text{input(new)} = \frac{(\text{input(old)} - \text{average})}{\text{variance}} \tag{4.1}$$

Equation 4.1 returns a range values with a mean of zero and a standard deviation of one. A further step is required to adjust the input values to an interval between zero and one. This is achieved by using Equation 4.2 to transform the inputs and also the single output.

$$\text{input(new)} = \frac{(\text{input(old)} - \text{minimum})}{(\text{maximum} - \text{minimum})} \tag{4.2}$$

This transformation limits the values of inputs and outputs to a range between zero and one where the minimum observed value of all the data is set equal to zero, and the maximum observed value equal to one. Such a narrow definition can be made without any risk because the absolute minimum and maximum values of all the data which the network will encounter are known. The network inputs and outputs will always fall in this range

Once pre-processing of the data had been completed, the training of various neural networks could be started. A summary of the inputs used for training of the networks is shown on Table 4.2.

Table 4.2: Inputs for the neural networks

No.	Neural Network Inputs
1	Elapsed time of the specific test at the time of the measurement (in seconds)
2	Initial average load (in Volts)
3	Initial load range (in Volts)
4	Change in load range (in Volts)
5	Change in temperature (in °C)

Elapsed time (Input 1) gives the network an indication of the component's age and allows the network to differentiate between new samples, and samples that have already fatigued. It can therefore differentiate between two samples that are subjected to the same loading but do not yet exhibit measurable signs of deterioration.

Longer term predictive capability is given to the network by providing it with information about the operating conditions to which the test piece is subjected. The initial load average and range (Inputs 2 & 3) define the conditions to which the test sample was subjected during testing. The network is therefore trained to differentiate between test pieces subjected to higher and lower loading, which is the main contributing factor to the rapidity with which failure occurs.

The changes from initial load and temperature (Inputs 4 & 5) give an indication of deteriorating test piece condition and impending failure. Due to the setting of the machine, displacement remained constant and load therefore dropped once cracking started. Temperature increased substantially as fatigue damage worsened and the crack propagated through the test piece. Adjustments could therefore be made by the network to its prediction once overt signs of impending failure become apparent. This adjustability allows the network to cope more easily with unexpected events and changing conditions.

It has already been observed that substantial increases in temperature take place as the test pieces approach failure. The tests however took place in diverse atmospheric conditions during the summer and winter months, meaning that this increase took place from a different baseline temperature. Absolute temperatures can therefore not successfully be used as a network input. For this reason it was decided to transform the temperature reading into a value reflecting the magnitude of the temperature change compared to the first reading. In a similar way the change in load range was calculated in order to reflect deterioration in the test piece. The testing was done with constant displacement amplitude for each test run, and the load therefore changed once the test piece started to crack or deform.

4.3 Weibull Distribution

The traditional way of conducting a data analysis on the reliability data originating from a renewal system was the fitting of a statistical distribution to such data. This technique, described by Coetzee (1997), was therefore chosen to form the basis of comparison to illustrate the advantage offered by neural networks. The Weibull distribution was consequently fitted to the data of the training set. This particular distribution was chosen above other statistical distributions due to its versatility, which has been discussed in Chapter 1.

The parameters to fit the two-parameter Weibull distribution to the data were found to be:

$\beta = 1.7522$ and $\eta = 8971$

The probability density function, *f(x)*, given in Figure 41 reflects the probability that failure will occur at time *x* in the component's life.

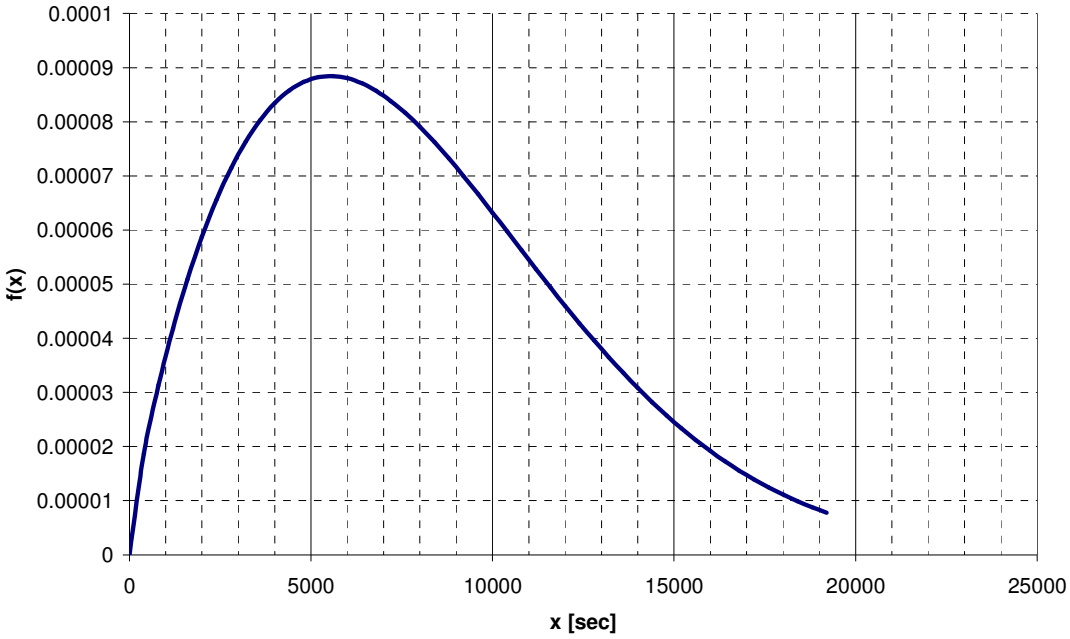

Figure 41: Probability density function

Figure 42 is the graph of the cumulative failure distribution function, *F(x)*, which indicates the probability of a component failing by a certain time *x*. The cumulative failure distribution function is obtained by integrating the probability density function. It is noted that the magnitude of *F(x)* tends to 1 as the value of *x* tends to infinity, which suggests that all units will eventually fail. On Figure 42 the actual failure times are plotted against the portion of the total number of units that had already failed by that time.

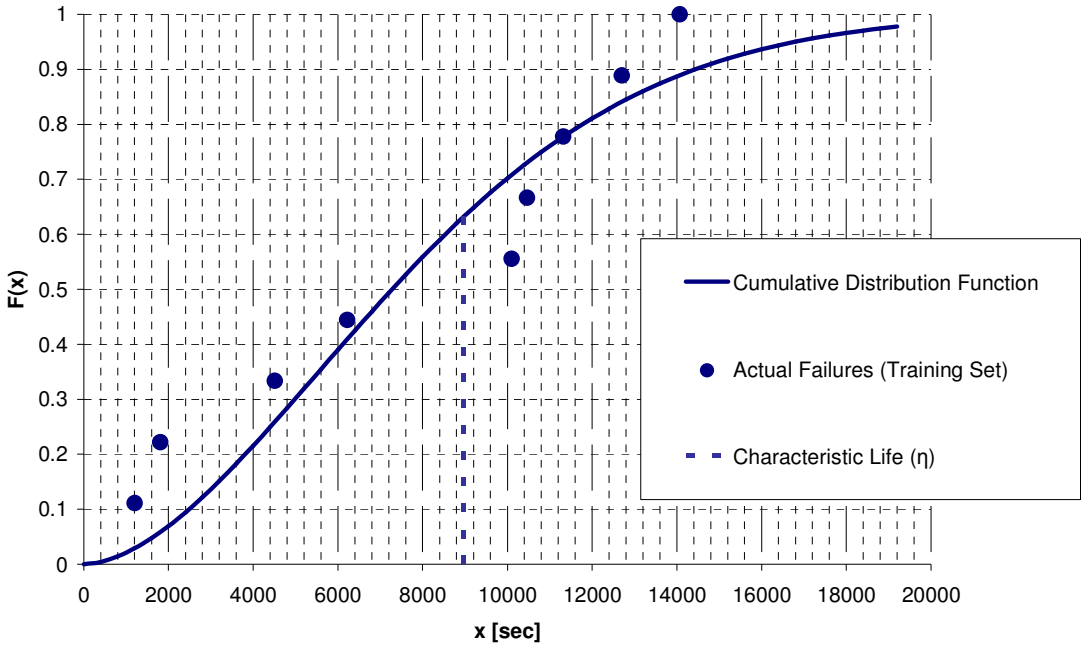

Figure 42: Cumulative failure distribution function

The Weibull parameter η is the scale parameter, which is also referred to as the characteristic life. According to Coetzee (1997), 63.2% of components fail before this time while 36.8% survive. The use of a statistical distribution means that no specific prediction can be made with respect to an individual test piece and the estimated life is therefore taken as the characteristic life of the whole population of the training set. Table 4.3 illustrates that the actual residual life for the test sets differed substantially from the characteristic life that was calculated through this statistical method.

Table 4.3: Comparison of the actual and estimated values for the residual life of the test data

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_22	10283	8971	12.8 %
2	TSA_24	5772	8971	55.4 %
3	TSA_26	10103	8971	11.2 %

According to Coetzee (1997), the most useful function to the maintenance practitioner is the hazard curve. In this case (see Figure 43), it has a convex shape (see also Figure 7) and the hazard rate is seen to be increasing. It would therefore in theory be feasible to implement a policy of preventive replacements in this case.

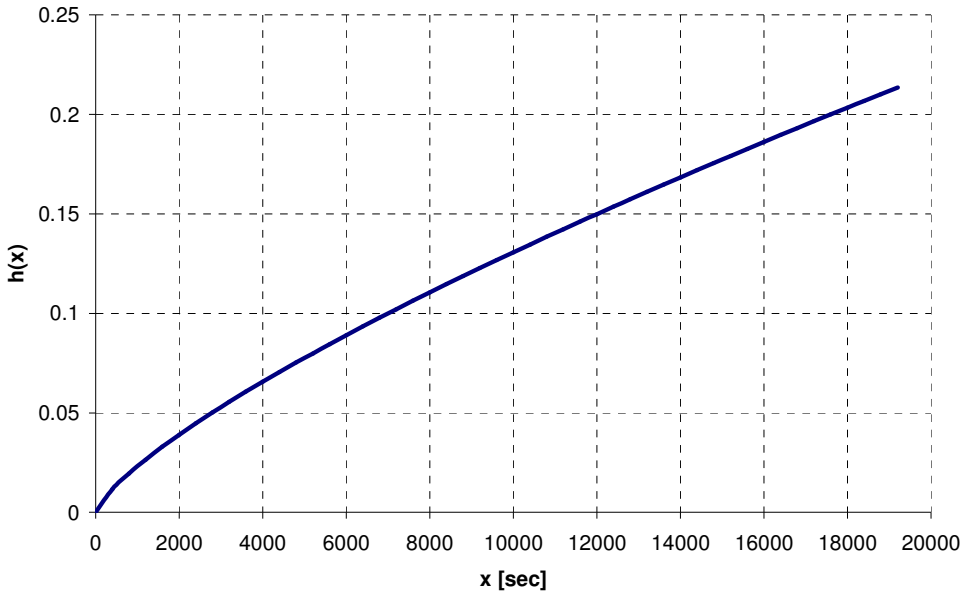

Figure 43: Hazard curve

Referring back to the criteria given by Bradley (1993) for effective preventive replacement, it was stated that it must be possible to predict the approximate time of failure. The test pieces were however subjected to different loading patterns which resulted in a corresponding variance in life expectancy. One statistical distribution can therefore not provide an adequate representation of the situation that was encountered. The results achieved through the fitting of the Weibull distribution show the disadvantages of this method when they are compared with the residual life results obtained through the use of neural networks which are discussed in the next sections.

4.4 Gradient Descent Backpropagation Algorithm

The standard backpropagation algorithm was used to train the same network architecture with nine different combinations of the learning rate (α) and momentum parameter (β). The results, which are tabulated below on Table 4.4, were used to select an optimal set of parameters. Training was in each case stopped when an error target of 1×10^{-5} was reached or after 900 epochs. This small training target meant that premature termination of training was only achieved with some of the slower training algorithms, usually those with a lower learning rate, and they were consequently prevented from overfitting and therefore provided a better response to the test data.

Figure 44 illustrates the rate of convergence of the gradient descent backpropagation algorithm with a different combination of training parameters. Oscillations become much more pronounced when a higher learning rate is used and training clearly becomes much more rapid. Should the learning rate be increased even more, the training process becomes unstable, overshoots the target and no convergence on a minimum is achieved. The training process must therefore balance the rate of convergence with the requirement for stability.

Table 4.4: Comparison of training performance using different learning rates and momentum constants.

No.	Learning Rate (α)	Momentum Constant (ß)	MSE Training Set	MSE Test Set
1	0.75	0	0.0220	0.0058
2	1.50	0	0.0117	0.0175
3	3.00	0	0.0102	0.0146
4	0.75	0.5	0.0128	0.0057
5	1.50	0.5	0.0172	0.0033
6	3.00	0.5	0.0051	0.0139
7	0.75	0.9	0.0216	0.0080
8	1.50	0.9	0.0109	0.0218
9	3.00	0.9	0.0084	0.0160

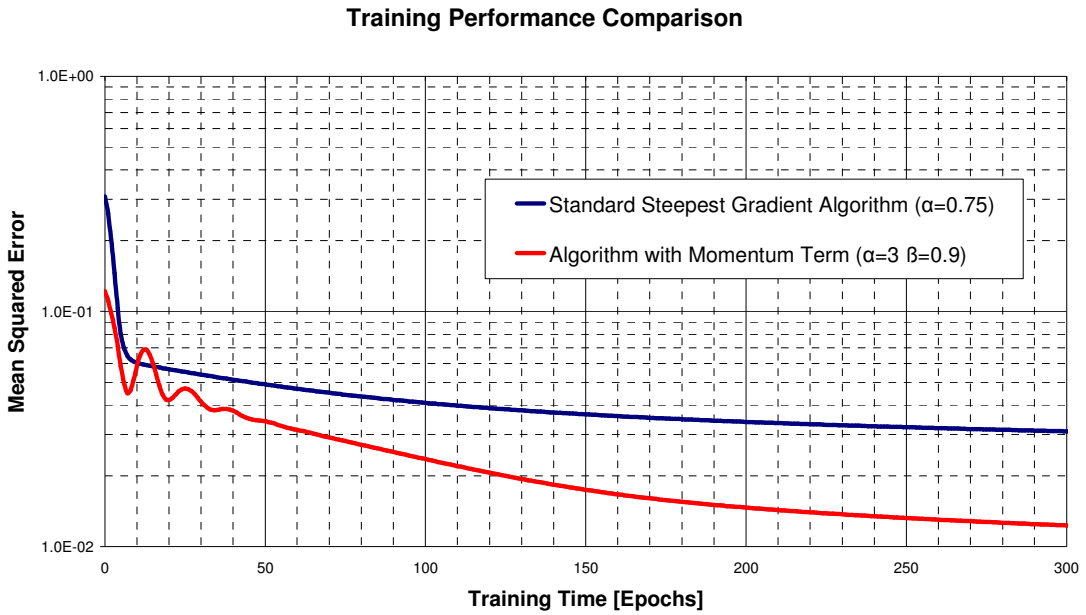

Figure 44: Comparison of the rate of convergence of the gradient descent backpropagation method using a different combination of training parameters.

It was found that a learning rate of 0.75 and a momentum constant of 0.9 provided good results and these constants were used for the comparison with other network types and training algorithms. The results obtained after the training of the network are listed on Table 4.5 and Table 4.6.

Table 4.5: Performance of the GDBP network when presented with the training data.

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_2	1685	1348	20.0 %
2	TSA_3	1083	838	22.6 %
3	TSA_5	10402	9106	12.5 %
4	TSA_8	11055	10771	2.6 %
5	TSA_9	6040	9139	51.3 %
6	TSA_20	13710	9713	29.2 %
7	TSA_23	9921	9295	6.3 %
8	TSA_25	4329	5982	38.2 %
9	TSA_27	10283	9278	9.8 %

Table 4.6: Comparison of the actual and estimated values for the residual life of the test data

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_22	10102	9174	9.2 %
2	TSA_24	5591	7542	34.9 %
3	TSA_26	9923	9521	4.0 %

The training algorithm was stopped early and was not able to accommodate some of the more isolated data points in the training set. It was therefore possible to maintain improved properties of generalization.

4.5 Levenberg-Marquardt Algorithm

Training with the Levenberg-Marquardt algorithm proved much more rapid and a much better fit was achieved after less than 300 training epochs. Table 4.7 shows the percentage difference between the actual and estimated failure times that where achieved with this network when presented with the first recorded inputs after the start of the test run.

Table 4.7: Performance of the MLP network trained with the Levenberg-Marquardt algorithm when presented with the training data.

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_2	1685	1646	2.3 %
2	TSA_3	1083	1035	4.4 %
3	TSA_5	10402	10284	1.1 %
4	TSA_8	11055	11177	1.1 %
5	TSA_9	6040	6229	3.1 %
6	TSA_20	13710	13058	4.8 %
7	TSA_23	9921	10370	4.5 %
8	TSA_25	4329	4481	3.5 %
9	TSA_27	10283	9958	3.2 %

The neural network's estimated residual life for the training data was within 5% of the actual remaining life of each component (see Table 4.8). The largest error of 449 seconds, which is the estimate for TSA_23, compares very favourably with the 180 second interval between measurements which is the band within which the failure occurred. The network performance on the training data is therefore a satisfactory result.

Table 4.8: Comparison of the actual and estimated values for the residual life of the test data

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_22	10102	9343	7.5 %
2	TSA_24	5591	4434	20 %
3	TSA_26	9923	10111	1.9 %

The result obtained by the neural network on data that was not previously seen during training was of the same accuracy in two of the cases. The largest error of 20% for TSA_24 may indicate some degree of overfitting, as the data from this test piece has the least similarity of the three test sets with the training data.

4.6 Levenberg-Marquardt Algorithm with Bayesian Regularization

It was expected that the use of Bayesian regularization would address the problem with overfitting that was encountered with the network trained with a standard Levenberg-Marquardt algorithm.

From the results which are reflected on Table 4.9 it can be seen that the neural network that was trained with Bayesian regularization did not produce such a close fit for some parts of the training set as was achieved with the standard Levenberg-Marquardt algorithm. Especially the estimates generated for TSA_2 and TSA_3 display a large error. This is expected, as the regularization technique penalises training in order to maintain the network's capability to provide acceptable results for new data. It is interesting to note that these two test pieces have a much shorter life than the other test pieces and are therefore isolated. The benefit of this regularization technique with respect to improved generalization becomes clear when looking at the results that were obtained for the test set (see Table 4.10). The largest error in a network prediction for the data of the test set was 5.1 %. The prediction of the neural network in this case was only 513 seconds adrift of the actual recorded life of 10102 seconds. The results achieved for all three of the test pieces in the test set was therefore very satisfactory and indicated good generalization.

Table 4.9: Performance of the LMBR Network when presented with the Training Data.

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_2	1685	1407	16.5 %
2	TSA_3	1083	951	12.2 %
3	TSA_5	10402	10271	1.3 %
4	TSA_8	11055	10812	2.2 %
5	TSA_9	6040	5983	0.9 %
6	TSA_20	13710	13235	3.5 %
7	TSA_23	9921	9931	0.1 %
8	TSA_25	4329	4312	0.4 %
9	TSA_27	10283	10320	0.4 %

Table 4.10: Comparison of the actual and estimated values for the residual life of the test data

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_22	10102	9589	5.1 %
2	TSA_24	5591	5392	3.6 %
3	TSA_26	9923	10232	3.1 %

4.7 General Regression Neural Network

The setting up of the GRNN is almost instant. This can be ascribed to its adaptation of the input vectors for the hidden nodes, and the use of the target vectors as weights in the output layer. No supervised training is required in its construction. Network performance can therefore only be influenced by changing the value of the spread of the radial basis function nodes in the hidden layer. The spread alters the radius of the basis functions and therefore determines the amount of overlap amongst the nodes and consequently the smoothness of the fit. A number of different spread values were tested, and the results are tabulated on Table 4.11.

Table 4.11: Performance of the GRNN for different spread values.

No.	Spread	MSE Training	MSE Test
1	0.01	9.9×10^{-7}	0.0030
2	0.02	3.8×10^{-6}	0.0027
3	0.03	8.3×10^{-5}	0.0022
4	0.04	8.5×10^{-4}	0.0026
5	0.05	2.3×10^{-3}	0.0034

Figure 45 illustrates that the larger the spread chosen for the network, the smoother the fit. The quality of the fit on the training set reduces, as a number of hidden layer neurons start to influence the output for any given input. An increase in spread however improves network generalization until an optimal balance is reached. Further increase in spread proves detrimental to network performance.

The results obtained by the GRNN with a spread of 0.03 are listed in Table 4.12 and Table 4.13. The estimated residual life for the training data when using a small spread value was closer to the actual than was achieved with any of the other networks, using the MLP architecture and supervised training. This can be attributed to the way in which the GRNN is trained and the insignificant overlapping of nodes with a small

radius. An exact fit is expected in this particular case, as the network should respond with the expected target vector if provided with a training vector.

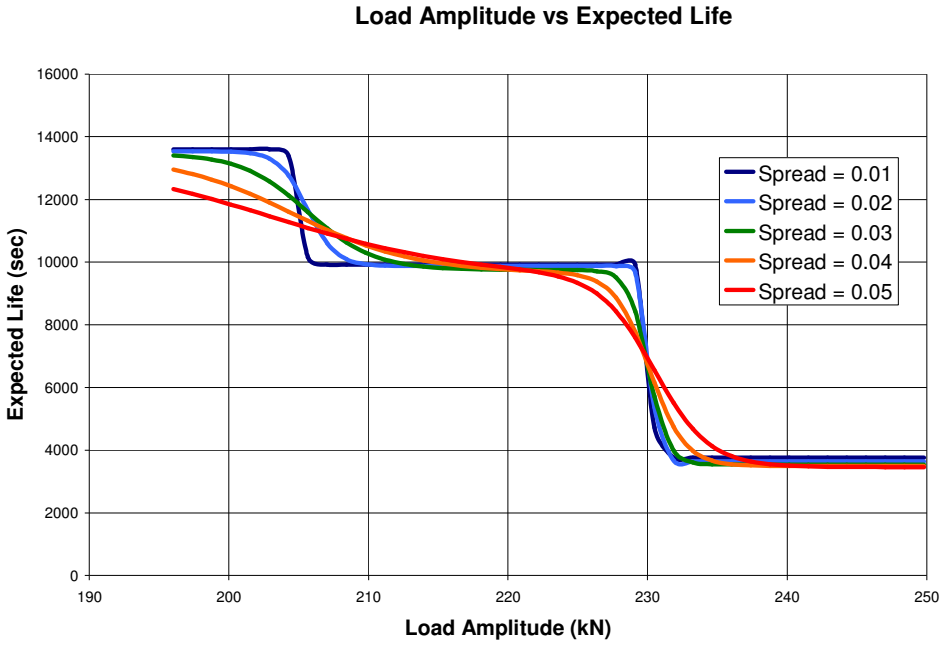

Figure 45: GRNN response when varying the load amplitude input.

Table 4.12: Performance of the GRNN when presented with the Training Data.

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_2	1685	1685	0.0 %
2	TSA_3	1083	1067	1.4 %
3	TSA_5	10402	10402	0.0 %
4	TSA_8	11055	11048	0.1 %
5	TSA_9	6040	5948	1.5 %
6	TSA_20	13710	13422	2.1 %
7	TSA_23	9921	9947	0.3 %
8	TSA_25	4329	4277	1.2 %
9	TSA_27	10283	10079	2.0 %

As was the case with the standard Levenberg-Marquardt algorithm, overfitting occurred during the design of the GRNN and a large error in the residual life estimate

for TSA_24 was observed. The problem of overfitting in RBF networks can be addressed in a number of ways and has been the topic of extensive research. Some of these methods have already been briefly discussed in Chapter 2. The application of such a method would however negate the GRNN's main advantage, which is the simplicity of its learning process.

Table 4.13: Comparison of the actual and estimated values for the residual life of the test data

No.	Data ID	Actual	Estimated	Percentage Difference
1	TSA_22	10102	9936	1.6 %
2	TSA_24	5591	9445	68.9 %
3	TSA_26	9923	10531	6.1 %

4.8 Discussion

The results that were achieved proved that neural networks are suitable for use in this type of application. Table 4.14 gives an indication of the quality of fit achieved by the different networks. The reader is however referred to the set of graphs included in Appendix C which show the fit of each neural network in greater detail. These lead to a number of interesting observations with respect to the performance of each neural network variation.

Table 4.14: Comparison of the mean squared error (MSE) on the training and test sets of the different networks.

No.	Network	MSE Training	MSE Test
1	LM with BR	5.7×10^{-5}	0.0014
2	GRNN	8.3×10^{-5}	0.0022
3	LM	8.1×10^{-5}	0.0030
4	GDBP with M	1.9×10^{-2}	0.0061

Table 4.15 shows the average prediction error while Table 4.16 gives the maximum prediction error of the networks that are compared. When considering these results it must be borne in mind that the measurements were taken at intervals of 180 seconds and the time of first measurement after failure was used as failure time for the training of the neural networks. The actual failure took place within the band spanning the last measurement cycle.

Table 4.15: Average error in the predictions made by the networks under comparison.

Network	Training Data	Test Data
LM with BR	64 sec.	455 sec.
GRNN	87 sec.	431 sec.
LM	84 sec.	616 sec.
GDBP with M	1370 sec.	841 sec.

Though the GRNN has a lower average error than the other networks, it has the highest maximum error. This serves to explain why the MLP network trained with the Levenberg-Marquardt algorithm with Bayesian regularization outperforms it in terms of the mean squared error. The early stopping of the gradient descent backpropagation algorithm meant that higher maximum and average errors were recorded for the training set. This phenomenon can be ascribed to the sparseness of the dataset which led to isolated data in the problem space. The training algorithm was stopped before it could accommodate this data and the network therefore performed well on the test data.

Table 4.16: Largest error in the predictions made by the networks under comparison.

Network	Training Data	Test Data
LM with BR	513 sec.	1065 sec.
GRNN	411 sec.	3085 sec.
LM	652 sec.	2185 sec.
GDBP with M	3997 sec.	1271 sec.

Figure 22 illustrated the effect of overfitting and the discontinuities are generated in the response of the network. The ability to generalize for data points that are located between those used for training is therefore lost. Great care is required to ensure that networks generalize, network complexity is appropriate and tests should always be performed with data not seen during training. The best fits on training data are usually achieved through overfitting. Lack of generalization may also be an indication that the network architecture is too simple.

All the neural networks performed very well on the data for TSA_26, probably due to its closeness to TSA_27. The data for TSA_24 showed the greatest variation from anything the network had seen before, and proved to be the greatest test of each network's ability to generalize. The advantage of using Bayesian Regularization to improve the network's ability to generalize is clearly illustrated when comparing the graphs of results relating to the series of data for TSA_24.

The graphs in Appendix C for the GRNN are not as smooth as those obtained with MLP networks. Especially the jagged shape of graph for TSA_22 illustrates the "local" nature of RBF networks compared with the "global" nature of MLP networks. This property may adversely affect network performance, should the information for one of data points that are used for training be corrupt. Greater overlap of the RBF nodes will counteract this situation by smoothing the transition between kernels.

The estimated residual life for the training data when using a GRNN with a small spread value was closer to the actual than was achieved with any of the other networks, using the MLP architecture and supervised training. This can be attributed to the way in which the GRNN is trained and the insignificant overlapping of nodes with a small radius. An exact fit is expected in this particular case, as the network should respond with the expected target vector if provided with a training vector. As was the case with the standard Levenberg-Marquardt algorithm, overfitting occurred during the design of the GRNN and a large error in the residual life estimate was observed for one of the test pieces.

The gradient descent algorithm was found to be significantly slower than the Levenberg-Marquardt algorithm. This is shown on Figure 46, which illustrates the advantage of using the second order above the first order method. The advantage of an unsupervised training process which was mentioned in the literature was proven by the speed with which the GRNN could be trained. Network learning in this case proved to be instantaneous.

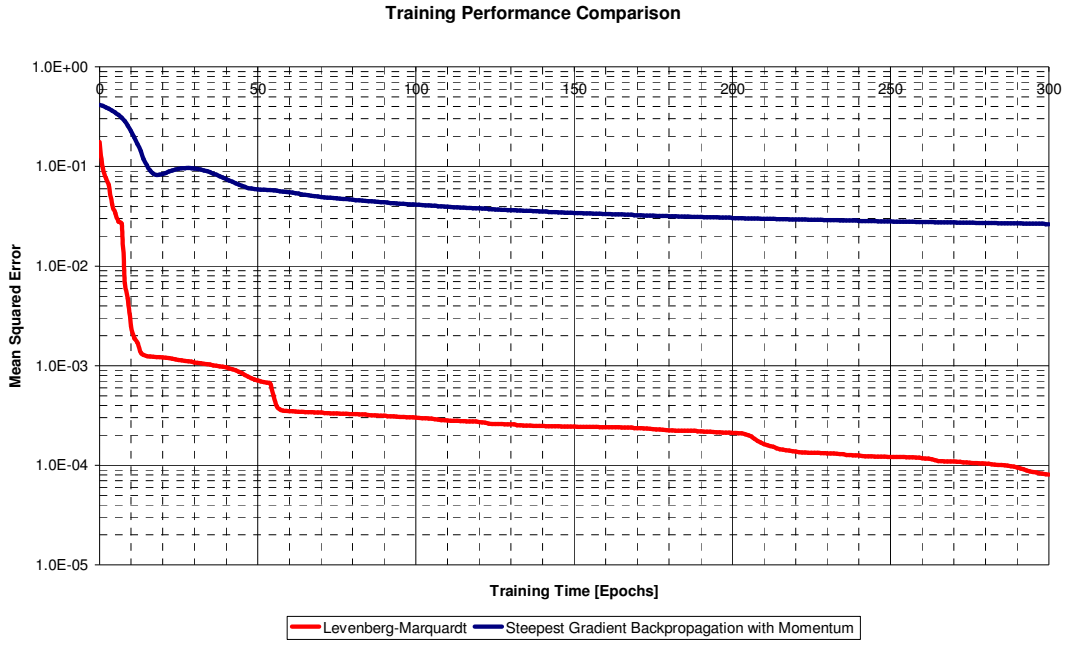

Figure 46: Graph comparing the rate of convergence of the steepest gradient backpropagation and Levenberg Marquardt algorithms.

The method of normalization (Equation 4.2) which was chosen for the re-scaling of input and output data limits the inputs and outputs of the network to a predetermined range of values. The extreme values of this range are chosen before the training of the network commences. For the present purpose this method is completely satisfactory as the extreme values of the entire dataset are known and could therefore be used. In a practical situation where a neural network is used to predict residual life of actual

components, this is generally not the case. There is a possibility that future measurements may fall outside the range which was used during training. The method discussed above would therefore become unsuitable and the network in question would provide meaningless outputs. A different method may therefore be required when re-scaling inputs and outputs in practice.

The smallness of the dataset's size did prove to be a problem. Testing was however a time consuming process, which limited the number of test runs that could be executed. Due to the long time span of the testing, continuity was lost, creating a dataset that was not structured as well as it could have been. There is little chance of the neural network generalizing to such an extent that a completely isolated test sample can be accommodated.

The definition of a baseline for a particular measurement proved to be important for this application. It allows the results of tests performed under different conditions to be used together in the training of the same neural network. In this case, where the practical difficulties led to testing being stretched over a long period, this is especially true. Temperatures and machine settings varied greatly. Lab ambient temperatures could differ by ±7 degrees during the day. Testing also spanned the summer and winter months, generating further variance. Though the change in ambient temperature had no effect on the life of the test pieces it would have been beneficial if ambient temperature had also have been measured at intervals during testing to facilitate the processing of data.

Table 4.17 indicates the accuracy of life predictions on the test set which were made by the different methods with the data available at the start of the various test runs.

Table 4.17: Accuracy of predictions for the test data with initial measurements recorded at the start of the experiments.

Approach	Type	Results for the Test Data
Statistical	Weibull	11.2% – 55.4%
Neural Network	GDBP with M	4.0% - 34.9%
Neural Network	LM	1.9% - 20%
Neural Network	LM with BR	3.1% - 5.1%
Neural Network	GRNN	1.6% - 68.9%

The results prove that neural networks can be successfully employed to make reliability predictions for a renewal system. When presented with the first set of measurements collected after the start of a test run, all the neural networks generated predictions which were more accurate than the results obtained through the traditional statistical method of fitting a Weibull distribution to the failure data. Especially the accuracy of the predictions made by the MLP trained with the Levenberg-Marquardt algorithm with Bayesian regularization would be suitable for the making of maintenance decisions in the context of the simulated situation.

Chapter 5: Repaired Systems

In the previous chapters it was described how neural networks were used to make failure predictions for a simple maintenance situation simulated in a laboratory environment. The life of each separate test piece was independent of the others and the series of tests therefore generated a dataset for which the renewal assumption is true.

Though individual components are discarded upon failure, the system of which they form part may be repaired during its life through the replacement of the damaged units. Because such repair is common practice, it would be desirable to extend the analysis of data to complete repaired systems.

The application of reliability data analysis techniques requires meaningful data and the testing of such analysis methods is dependent on the finding of a suitable and representative dataset. Such datasets, particularly for repaired systems where the renewal assumption is not likely to be true, are unfortunately hard to find. Pijnenburg (1991) comments on the extreme rarity of repairable systems datasets given in literature, where the failure times are listed in original chronological order. Ascher and Feingold (1978) could apparently only find four such sets. Trend tests cannot be performed on data if the sequence in which the failures took place is not known because no record was kept of the global time of failure. In such cases no information is available to ensure the correct choice of model, as it is impossible to establish what influence failures exert on the future reliability of the system and what contribution they have on the survival time to the next failure. In their paper, Leung and Cheng (2000) highlight the problem of insufficient data, and how it contributes to making statistical analysis difficult. Vlok (2001) confirms this view, commenting that the dataset he used had many shortcomings, such as missing observations and irregular inspection intervals, but notes that it was the best example available to him, even after an extensive search.

The process of finding data suitable for this project was faced with a similar problem. It was found that the data that was made available was generally flawed, difficult to extract, incomplete and therefore usually not suitable for analysis. The duration over which readings were taken was insufficient, usually only for two life intervals and the data in each case was almost exclusively censored in some way with little information to assist the analyst. In the light of these difficulties in finding a new dataset, the decision was made to use the existing data which had been previously collected by Vlok (2001) for the purpose of this work.

5.1 Description of the dataset

In this project the use of neural networks is investigated as an alternative to statistical methods that have been used for reliability data analysis. The application of the generic regression models as proposed by Vlok (2001) has been discussed in Chapter 1 and is of particular interest. In his work he attempted to combine the benefits of failure data analysis and condition monitoring in order to improve his residual life predictions.

The dataset used by Vlok (see Appendix C) was obtained from the Sasol Twistdraai mine plant. Measurements were taken on 8 identical Warman pumps which are used to circulate a water and magnetite solution within the plant. All these pumps therefore operated under more or less similar conditions. Measurements were unfortunately only taken sporadically and the dataset therefore has very few data points.

The measurements taken on the pumps were exclusively vibration readings, for which a spectral analysis was performed and a number of fault frequency bands (see Table 5.1) were monitored. Readings were taken at both bearings in each pump.

Table 5.1: The vibration frequency bands which were monitored on each of the pumps.

Measurement ID	Vibration sensor location	Frequency band description
RF043H	Wet-end bearing	0.4 × rotational frequency
RF13H	Wet-end bearing	1 × rotational frequency, indicative of pump unbalance.
RF23H	Wet-end bearing	2 × rotational frequency, indicative of shaft misalignment.
RF53H	Wet-end bearing	5 × rotational frequency, indicative of cavitation.
RF044H	Dry-end bearing	0.4 × rotational frequency
RF14H	Dry-end bearing	1 × rotational frequency, indicative of pump unbalance.
RF24H	Dry-end bearing	2 × rotational frequency, indicative of shaft misalignment.
RF54H	Dry-end bearing	5 × rotational frequency, indicative of cavitation.

Vlok identifies four main failure modes for these particular pumps namely:

1) Complete bearing seizure
2) Broken or defective impeller
3) Damaged or severely eroded pump housing
4) Broken drive shaft

No information is contained in the dataset with regards to the cause of failure or the reason for a condition based suspension and overhaul. Vlok states that alarm levels were used as prescribed by the pump manufacturer, but these values are not given.

During the 791 day window from the initial installation of the eight pumps, pump operation was suspended 8 times due to condition based warnings, while 11 failures were recorded. The surprisingly high percentage of failures can possibly be blamed on the inconsistent application of the condition based policy and long measurement intervals. Even though the data was apparently collected from the start of each pump's

life, the vibration measurements were taken extremely infrequently. Vlok (2001) does note that some of the failures occurred suddenly, with the deterioration taking place within a matter of hours. Clearly it is difficult to predict such a sudden deterioration with the information that is available. From the random nature of measurements, it appears that the final measurement ahead of the suspension of a pump's operation may have been prompted by clearly observable external signs of pump deterioration.

The dataset is right censored, with data collection ending at a particular date. The data which was collected between the last pump failure and the time suspensions can be used if additional work is done to estimate survival times, but this would have increased the complexity of the problem. Though such an approach may be feasible in practice, the small increase in size of the dataset was not deemed worthwhile for the purpose of this work.

Of the 8 pumps, 3 units experienced only one failure, 2 units failed twice while 3 units each failed four times. On average the pumps lasted 469 days to the first failure or preventive intervention. This can be compared with an average of 134 days, 103 days and 137 days to the second, third and fourth failure respectively. Reliability therefore deteriorated dramatically after the first failure indicating imperfect repair. A further pattern was observed with regards to the time to first failure which allows the subdivision of the pumps into two groups. Pumps which failed the first time after more than 500 days tended to fail only once or twice during the period in question. The remaining units averaged 357 days to first failure and each failed four times within the time window.

5.2 Comparison of neural networks and statistical methods

Amongst the measurements that were taken, the most significant covariates had previously been selected by Vlok (1999) using the same source of data. During this process of elimination, two measurements which provided the best results were singled out for use in the proportional intensity models that he generated. The covariates both represented the level of vibration in the fault frequency band at four times running speed, measured horizontally on the bearings. The purpose of this particular measurement in a condition monitoring context is to serve as an indicator for the occurrence of cavitation within the pump. This data is given in Appendix C, Table C2.

Vlok used the Laplace trend test on the failure data of each particular pump to establish if there was a trend in the failures. Due to the shortage of information, only the data of three of the eight pumps could be tested for trends. The data that could be tested was found to display reliability degradation. As a result of this finding, Vlok applied the proportional intensity model that he had developed for the repairable case.

Parametric approximations were generated for each of the chosen covariates, and polynomials of up to the 3rd order where fitted for each individual pump life, in terms of system (global) time. Dependent on the nature of the model, different combined proportional intensity models were fitted with Snyman's method or with the modified Newton-Raphson technique for parameter estimation. A total of 5 different variations of his combined model for repairable systems were eventually tested and compared.

The conventional NHPP model was used as a benchmark for testing the four more advanced models. Two of the models performed worse than the benchmark. The multiplicative intensity model with stratified regression coefficients resulted in the

highest sum of squared errors. The conventional model modified with stratified jump/setback coefficients proved to result in the largest confidence bound. An additive intensity model using the NHPP as baseline function and stratified regression coefficients showed an improvement over the conventional model. The best results were achieved with an additive intensity model with a time jump/setback in the baseline and stratified regression coefficients. For the purposes of comparison with neural networks, it was decided to use the NHPP model, as well as the advanced model which had provided the best results.

It was mentioned in the previous chapter that a MLP neural network platform, using a Steepest Gradient Backpropagation algorithm with momentum, was written for this project using Microsoft Excel software with VBA routines. This network was trained on combinations of inputs generated from the available data. A number of training runs were performed with these input combinations, but in the interest of simplicity and clarity, three variations where chosen for inclusion and are discussed in greater detail.

The first network was trained on basic failure time information only, to generate a baseline for measuring the performance of the other networks. The second variation was trained with an additional explanatory variable to explore the benefits of using covariates as part of such an analysis. The final case that was looked at used information gathered from vibration measurements to further improve the accuracy of the prediction, and to look at a practical application of such a network.

5.2.1 Case 1 Trained with failure time data

The basic network consisted of two input nodes, a hidden layer with two nodes, and a single output node. The life of each pump is broken into intervals, bounded by failures or termination.

The first input to the network is taken to be the age of the pump, at the start of the life interval in question. During pre-processing, the age of the pump which is expressed in days, was divided by a thousand in order to arrive at a number between zero and one. In this way the inputs were adjusted for use in the network. Amjady and Ehsan (1999) transformed their input domain in a similar way by using a pre-processing procedure which performed a linear mapping. They state that a range of [-1, 1] for input features and [0, 0.9] for output features provided the desirable results.

The second input is the interval number of the data point, with the numbering starting at one, divided by 10. This transformation is very simple, but was found to produce satisfactory results.

5.2.2 Case 2 Trained with an explanatory variable based on MTTF

Case 2 tested the effect of an additional explanatory variable on the quality of the fit. Apart from the two inputs already used for Case 1, a third input was added. A covariate was generated which had a direct relationship with the failure times of the pumps. To generate this covariate, the average time to failure was calculated for each pump and the pumps were then classified into one of two groups, depending on the calculated value. The two groups were each given a unique numerical value between zero and one that was used as an input to the network. These values were chosen in such a manner that they allowed the network to differentiate between the groups, without having a negative effect on the network's stability.

5.2.3 Case 3: Trained with two Vibration Covariates

The last case under investigation used a neural network with four inputs. The two inputs of Case 1 were combined with two inputs that were calculated from the polynomial curve fits that were made by Vlok on the vibration covariates that he had selected. The functions were evaluated at 21 days after start-up in global time. The network was optimized by trial and error, with the best results being achieved with an architecture containing three hidden nodes. The training of the third network required a longer training period of about 600 cycles to converge on a global minimum.

5.3 Results

A set of radar plots has been generated to allow the results achieved by the different models to be compared. The predicted and actual failure times at the start of each life interval are shown on the graphs.

The basic neural network (Case 1, see Figure 47) closely matched the fit achieved with the NHPP model (Figure 48) of Vlok (1999). This result was expected, as these models were generated with similar inputs and did not cater for the use of covariates.

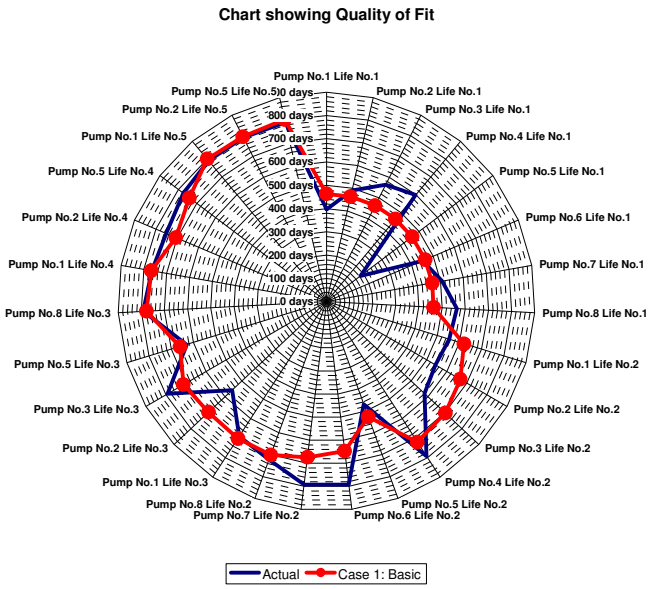

Figure 47: Results achieved with a Neural Network without covariates (Case 1)

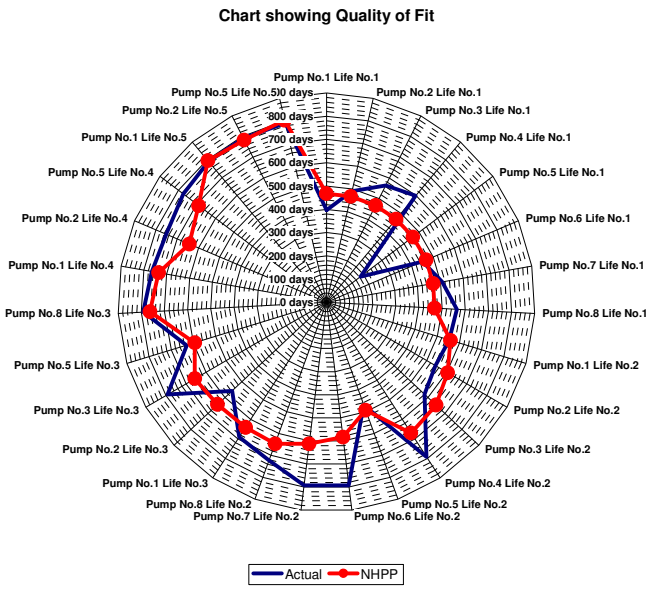

Figure 48: Graph showing the fit for NHPP model without covariates or stratifications

With an additional input available for training, chosen on the basis of average time to failure for the given pump, a significant improvement could be achieved in the performance of the MLP neural network (see Figure 49). If the radar charts of the Case 1 and Case 2 are compared, it can be seen that the network with an additional explanatory variable has much greater flexibility when fitting the data and therefore produces a better result.

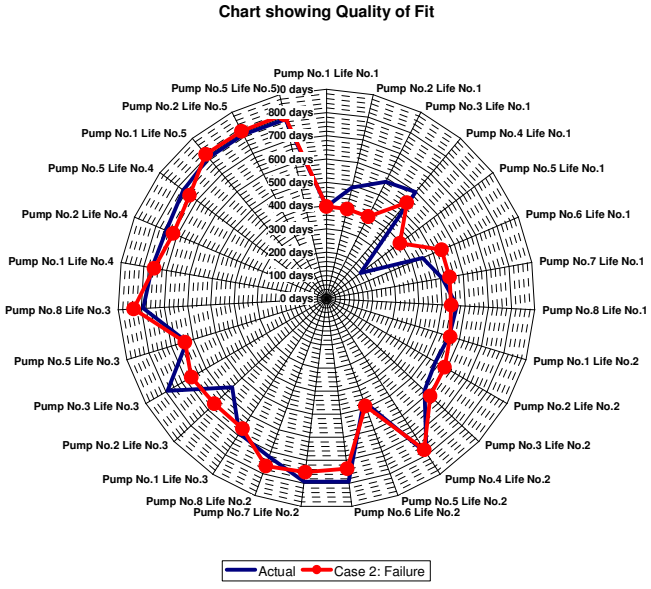

Figure 49: Results achieved with a neural network using a covariate which classifies the pumps according to the average failure time of each pump (Case 2)

This added flexibility is especially illustrated by the network predictions made for the interval leading up to the first failure of each pump. For Case 1, both network inputs would be the same for all the pumps, as the pumps start their life at zero. The basic neural network could therefore not be able to differentiate in any way between the

different pumps and therefore produces a prediction that is the average value for this input combination.

This situation also applies for the NHPP model. With the addition of the covariate, the pumps are subdivided into two groups, and hence it is made possible to get two different outputs for the first lifetime estimation. The network is therefore tailored during the learning process, to make classification according to these groups, and it is able to generate a closer fit for the data falling within each category. In practice condition monitoring data would perform the function of allowing the network to differentiate between cases subject to different operating conditions or suffering from different failure modes.

The network using the vibration data (Case 3) proved to have even greater flexibility than the network trained with the Case 2 inputs. This is reflected in the radar chart (Figure 50) plotted for its predictions. It has the freedom to generate individual outputs for each of the pumps, which is especially noticeable during their first life interval, when compared with the other models. The results obtained with this network closely approach the results of the best fit achieved by Vlok with regression models (Figure 51).

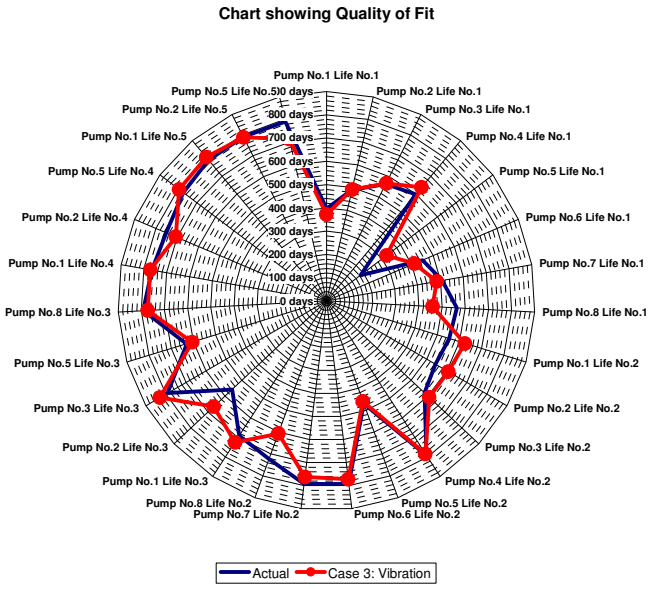

Figure 50: Results achieved with a neural network using two covariates reflecting vibration readings (Case 3)

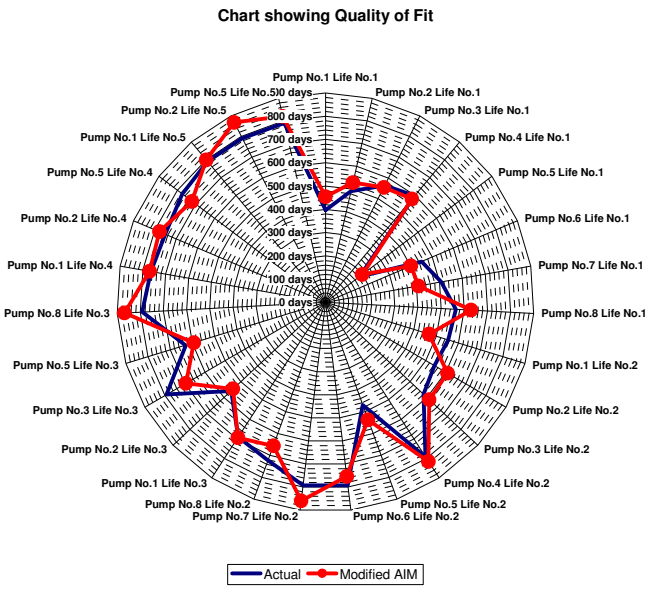

Figure 51: Graph showing the fit for an Additive Intensity Model with a time jump/setback in the baseline and stratified regression coefficients

The sum of squared errors are tabulated on Table 5.2 to display the fit achieved with each of the various methods for the first measurements of each interval.

Table 5.2: Comparison of the Results

No.	Model / Neural Network	Σ (error)^2
1	NHPP (Vlok)	2.79×10^5
2	Case 1: Failure Time Data	2.32×10^5
3	Case 2: MTTF Classification	1.36×10^5
4	Case 3: Vibration Covariates	8.36×10^4
5	Additive intensity model with a time jump/setback in the baseline and stratified regression coefficients (Vlok)	7.76×10^4

The results that were achieved through the use of neural networks are very similar to those previously achieved by Vlok. Figure 52 shows a comparison of the errors in the prediction made by the different methods. For greater clarity, the calculated errors were sorted numerically and plotted in ascending order.

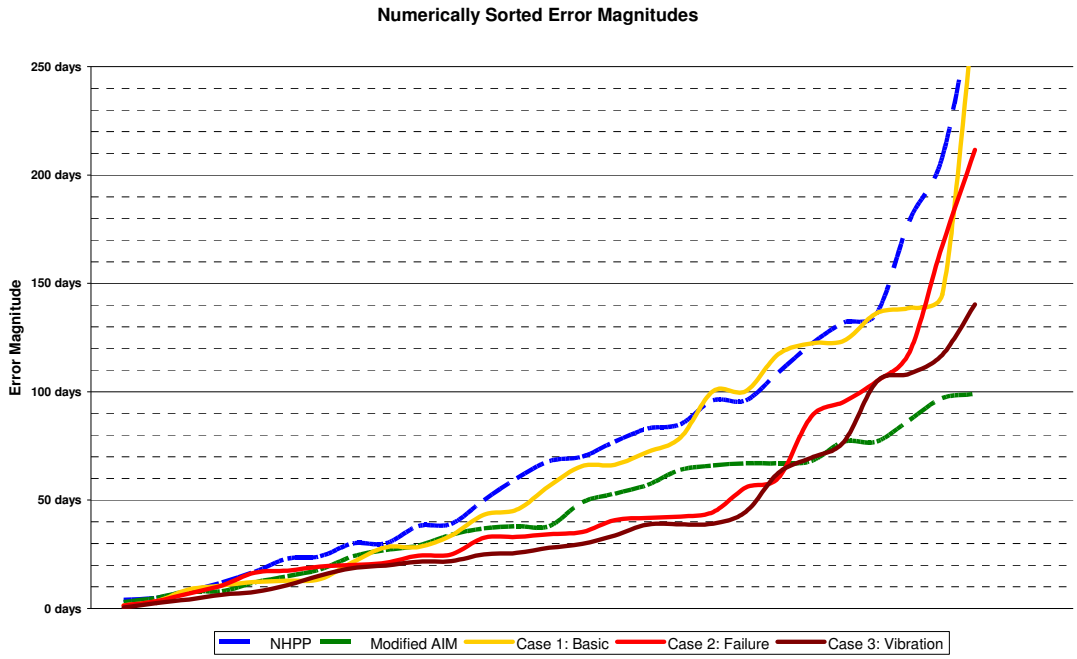

Figure 52: Comparison of the Performance of the Different Methods. The errors are sorted numerically and are plotted in ascending order from left to right.

5.4 Network comparison by cross-validation

The goal when training a neural network is not to achieve the smallest error on the training data, but to build a mathematical model of the process that generates the data. In the previous section network generalization was not tested, as only the goodness of fit achieved by various statistical and neural network methods were compared.

If the data is generated by a reliable physically based model, Schenker and Agarwal (1996) regard such a statistical comparison as sufficient. This is not the case with this dataset and further testing is required to test the neural networks' ability to generalize adequately for data that was not seen during training.

The sparseness of the dataset does not allow for a separate test set and it was therefore decided to use cross-validation to test the performance of different network designs.

5.4.1 Neural networks, training and cross-validation

For the purpose of cross-validation the dataset was broken into 8 groups, each of which represented the data from one of the pumps. In their work, Schenker and Agarwal (1996) make the assertion that individual runs should not be split when using cross-validation as it would violate the assumption that the test and training sets are independent. The total life of each pump was therefore deemed to be one run and the data was grouped accordingly. Further, it was decided to deviate from the approach which was used by Vlok which involved the fitting of polynomials to data of the

various pumps. The use of actual readings instead of values generated from a function fitted to the data was deemed to be a more realistic approach when contemplating the use of such analysis in practice.

The actual data was consequently pre-processed in a similar way to the methodology which has already been described in Chapter 4. It was found that the high values measured at an advanced stage of deterioration led to a distortion in the normalized data inputs which were used to train the neural networks. The neural networks became insensitive to the small changes which occur at the initial stages of deterioration. As the aim of this work is not to prove the use of condition based maintenance, but to improve longer term predictions of expected life, the readings taken during the last week before the occurrence of failure were discarded. This decision led to an improvement in the accuracy of predictions at earlier stages of deterioration.

On the basis of the performance of the neural networks which were trained with the renewal dataset, it was decided to focus on the network types (see Table 5.3) that could be trained more rapidly, as cross-validation involves the time consuming repetition of network training. The standard Levenberg-Marquardt algorithm, the Levenberg-Marquardt algorithm with Bayesian regularization and the GRNN were therefore chosen for comparison.

Table 5.3: Summary of the network types that were compared.

No.	Neural Network Description	Abbreviation
1	MLP neural network trained with Levenberg-Marquardt algorithm;	LM
2	MLP network trained with Levenberg-Marquardt algorithm using Bayesian regularization	LMBR
3	General regression neural network	GRNN

The use of a greater number of network inputs representing condition based information is expected to improve the network's ability to make accurate predictions. To test this hypothesis, each of the neural network layouts was trained with three, four and five inputs. The first set of training runs were done with the elapsed time since installation, the elapsed time since the last failure and a covariate that can be described as a risk variable, which is dependent on the history of the pump. Two further training runs were completed first with one and then with two additional inputs which each represented the average value of the vibration response amplitude in a chosen frequency band for the measurements on the two bearings. The second set of training runs was therefore done with inputs 1 to 4 and the third training run with inputs 1 to 5 on Table 5.4, which gives a summary and brief description of the inputs which were used for neural network training.

Table 5.4: Inputs for the neural networks.

No.	Neural Network Inputs
1	Elapsed time since the installation (in days)
2	Elapsed time since the last failure (in days)
3	Risk variable R
4	Average of RF53H and RF54H (in mm/s)
5	Average of RF043H and RF044H (in mm/s)

The dataset size which was used for cross-validation contained 53 data points. Once this was subdivided into groups, the largest group contained 13 data points which meant that smallest training set would contain 40 data points. The maximum number of hidden nodes in a MLP network with five inputs was therefore limited to 5, in order to prevent ill-conditioning as per the discussion in Chapter 2.

The dataset originates from a repaired system and its reliability is therefore affected by previous failures and repair. The influence of these factors must be taken into account even though not much of this information was recorded. Vlok (1999, 2001) states that alarm levels were used as prescribed by the pump manufacturer, but these values are not given and the cause of failure or the reason for a condition based suspension and overhaul was not indicated. An empirical risk variable was consequently based on the observed pattern which indicates that pumps that required an early repair tended to fail more frequently. For the data collected before the occurrence of the first failure, the risk variable R is set equal to 1. After the first failure, Equation 5.1 is used to calculate the value of R.

$$R = \tfrac{1}{2}\left(\frac{T_1}{T}\right)^2 \qquad (5.1)$$

The risk variable R is therefore reduced to 0.5 immediately after the first failure and its value decreases at a rate dependent on T_1 which is the time to the first failure. T is the elapsed time since the initial installation of the pump unit. A large value of R therefore corresponds to a low risk of failure, while a small value indicates a high risk. It takes into account the significant reduction in reliability after the first failure and the characteristic of a high failure rate in cases where an early first failure is recorded.

The hidden nodes of the MLP networks were varied according to the number of inputs that were presented to the networks to test the effect of such changes in network structure on network performance. Training was firstly done for networks with the same number of inputs and hidden nodes. A second training run was then performed with a hidden layer that had one node more than the input layer. MLP networks with 6 hidden nodes could however not be trained with 5 inputs due to ill-conditioning. The dataset size which was used for cross-validation contained 53 data points. Once this was subdivided into groups, the largest group contained 13 data points which meant

that the smallest training set would contain 40 data points. The maximum number of hidden nodes in a MLP network with five inputs was therefore limited to 5, in order to prevent ill-conditioning as discussed by McKeown et al. (1997), because the number of variables in the network exceed the number of inputs. The networks all generated a single output which was a prediction of the remaining life until the next failure of the pump.

5.4.2 Levenberg-Marquardt Algorithm

The neural networks trained with the Levenberg-Marquardt algorithm used nodes with the log-sigmoid transfer function in both the hidden and output layer. During initial training with a small mean squared error training target of 1×10^{-5} it was found that overfitting occurred and the neural networks failed to generalize on the test data. A series of training runs with a range of different training targets were consequently performed in order to improve generalization by an early termination of the training process. Table 5.5 lists the training targets that were used for this purpose.

Table 5.5: List of training targets which were used to stop the training of the Levenberg-Marquardt algorithm.

No.	Training Targets
1	0.05
2	0.025
3	0.01
4	0.005
5	0.00001

The complete results achieved with the various MLP networks trained with the Levenberg-Marquardt (LM) algorithm are given in Appendix D. In summary, Table 5.6 shows the five best and Table 5.7 the five worst results achieved with these networks. Please note that the target values refer to the normalized output values, while the sum-of-squares error is calculated from an error value in days.

Table 5.6: The best results achieved with the Levenberg-Marquardt training algorithm.

No.	Network Architecture	Inputs	Target	Σ (error)²
1	5 hidden nodes	5	0.05	2.70×10^{5}
2	5 hidden nodes	4	0.025	2.85×10^{5}
3	4 hidden nodes	4	0.025	2.90×10^{5}
4	3 hidden nodes	3	0.05	2.92×10^{5}
5	4 hidden nodes	4	0.05	2.92×10^{5}

The smallest error on the test data was achieved with the larger target values 5×10^{-2} and 2.5×10^{-2}. The comparatively small error on the training data indicates that the training algorithm was stopped before the overfitting which characterised the results on Table 5.7 could occur.

Table 5.7: The network results with the largest error after training with the Levenberg-Marquardt algorithm.

No.	Network Architecture	Inputs	Target	Σ (error)²
21	3 hidden nodes	3	0.00001	4.32×10^{5}
22	4 hidden nodes	3	0.00001	4.45×10^{5}
23	4 hidden nodes	3	0.01	4.49×10^{5}
24	5 hidden nodes	5	0.01	4.99×10^{5}
25	5 hidden nodes	5	0.00001	5.31×10^{5}

The network error usually did not reach the smaller target values of 1×10^{-2}, 5×10^{-3} and 1×10^{-5} and the training process was therefore terminated once the preset limit of 100 epochs was reached. These networks consequently suffered from overfitting and failed to perform well on the test data.

The effect of changing the size of the hidden layer and the number of inputs is affected by the early stopping of the training process and no clear pattern emerges. While an additional node in the hidden layer was beneficial when training towards an error target of 2.5×10^{-2}, it seemed to be detrimental when training with a target value of 5×10^{-2}. It does appear that a greater number of inputs generally improved the performance of these networks, but the result is not conclusive.

5.4.3 Levenberg-Marquardt Algorithm with Bayesian Regularization

The MLP neural networks which were trained with the Levenberg-Marquardt algorithm with Bayesian regularization (LMBR) yielded similar results to those which were trained for the optimal duration with the standard Levenberg-Marquardt algorithm. In this case the Bayesian regularization prevented overfitting during training, thereby improving the network's ability to generalize.

The log-sigmoid transfer function was used for the nodes in the hidden layer of these networks, while two different transfer functions were utilized in the output layer. Table 5.8 gives a summary of the results that were achieved with the neural networks trained with the Levenberg-Marquardt algorithm with Bayesian regularization.

Table 5.8: Levenberg-Marquardt algorithm with Bayesian regularization.

No.	Network Architecture	Inputs	Σ (error)^2
1	5 hidden nodes, linear output node	5	2.81×10^5
2	5 hidden nodes, sigmoid output node	5	2.90×10^5
3	5 hidden nodes, linear output node	4	2.95×10^5
4	4 hidden nodes, sigmoid output node	4	3.06×10^5
5	5 hidden nodes, sigmoid output node	4	3.07×10^5
6	4 hidden nodes, linear output node	4	3.15×10^5
7	4 hidden nodes, linear output node	3	3.26×10^5
8	3 hidden nodes, linear output node	3	3.35×10^5
9	3 hidden nodes, sigmoid output node	3	3.52×10^5
10	4 hidden nodes, sigmoid output node	3	3.52×10^5

The results reflected on Table 5.8 show that the choice of transfer function of the output node exerted a much greater influence on the performance of the network than the variation in the number of nodes in the hidden layer. Another observation that can be made is that in this case additional input information clearly led to more accurate predictions.

5.4.4 General Regression Neural Network

The GRNN networks were trained with RBF neurons with different sensitivities for the purpose of selecting an optimal value for this parameter. Table 5.9 reflects the results for the networks with values of 0.1 and 0.05 for the spread parameter.

In contrast to the other network types, the best results with GRNN were achieved with four inputs while the increase in input space to five inputs led to overfitting. The consequence of an increase in the number of inputs to such a network is that the outputs of the nodes are influenced by a greater number of variables and hence they become more sensitive to a particular combination of input values. Once this sensitivity becomes too great the network starts to loose its ability to generalize. For this reason, the general regression neural networks did not respond well when presented with five inputs.

Table 5.9: Cross-validation for GRNN

No.	Network Architecture	Inputs	Σ (error)^2
1	Spread = 0.1	4	3.00×10^5
2	Spread = 0.05	4	3.32×10^5
3	Spread = 0.1	3	3.56×10^5
4	Spread = 0.05	3	3.72×10^5
5	Spread = 0.1	5	3.72×10^5
6	Spread = 0.05	5	4.17×10^5

Two different values were used for the spread in the GRNN's and it was found that the less sensitive networks with a spread of 0.1 generally achieved the better results. The decreased sensitivity achieved with a larger radius for the basis function led to a reduced the degree of overfitting in the network.

The ease of implementation of the GRNN was again illustrated. The construction of this type of network is instant as no weight adjustments take place through the implementation of a backpropagation algorithm. By varying the sensitivity of the RBF neurons, adjustments can be made to optimise the network's ability to generalize. An optimal network can therefore be rapidly found through the implementation of cross-validation.

5.5 Discussion

The usefulness of neural networks as an alternative to statistical methods with respect to the estimation of residual life to failure of repairable systems was already proved in the first part of this chapter. When testing network generalization by using cross-validation, the best results obtained with the various neural networks were very similar once these networks had been optimized with respect to this particular dataset. Table 5.10 gives a comparison of the best results achieved with each network type. A further graphic representation of the results achieved by the different neural networks is given in Appendix E.

Table 5.10: Comparison of the best results achieved by the different network types.

No.	Network Architecture	Σ (error)²
1	LM	2.70×10^5
2	LMBR	2.81×10^5
3	GRNN	3.00×10^5

The comparison of the different networks by cross-validation was only based on the relative size of the sum-of-squares error obtained on the test data. If the suitability of the applied method is to be judged, the results must also be viewed in the context of the practical application. Figure 53 shows the distribution of the error magnitudes of the failure estimates for the networks of each type which provided the best results on the test data. It was found that number of very large prediction errors were made by the networks which far exceeded the actual remaining time to failure of a specific pump. The isolated points on Figure 54 serve to illustrate this problem. The average prediction error of the networks if the ten worst predictions are excluded is 39.8% for the LMBR network, 33.2% for the GRNN and 41.7% for the LM network. These averages correspond to the first peak seen on Figure 53.

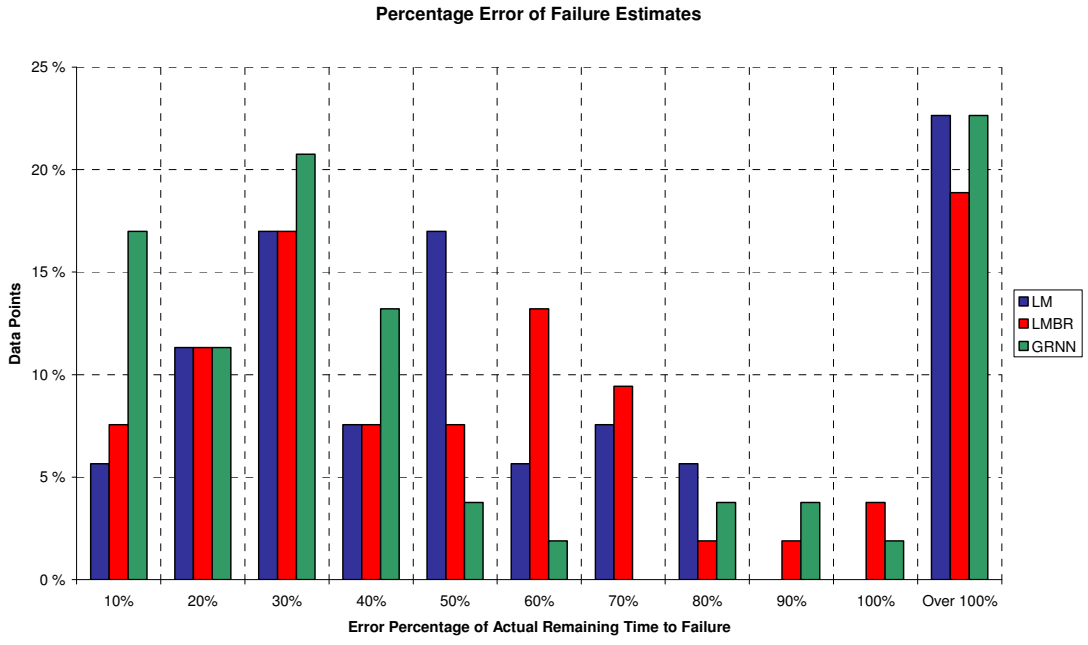

Figure 53: Comparison of the results obtained with the different neural networks in terms of percentage prediction error of the actual remaining life.

The nature of this result indicates that the networks have been able to model some but not all of the significant properties of these complex pump systems. When seen in the context of its intended application, the results therefore represent a positive point of

departure. An average prediction error of 40% is too large and does not allow these networks to be used for the making maintenance decisions while in their current form. It can therefore be said that the complexity of the problem requires a larger and more descriptive dataset for training of neural networks if more accurate results are to be obtained.

A key element in the successful practical application of neural networks is to find suitable covariates that will allow the network to differentiate between different scenarios and failure modes. The smallness of the dataset also has the result that part of the data in the test set will in some cases be significantly different to the data with which the network was trained. The network is therefore unable to deal with the data correctly and produces a spurious result. The dataset used by Vlok (1999, 2001) is not ideal for this purpose due to its sparseness and it is not believed that much more could be achieved with regards to failure prediction in view of the given data.

Despite these deficiencies it was proved that it is possible to combine the advantages of failure time data analysis and condition monitoring in a neural network platform to make more accurate predictions.

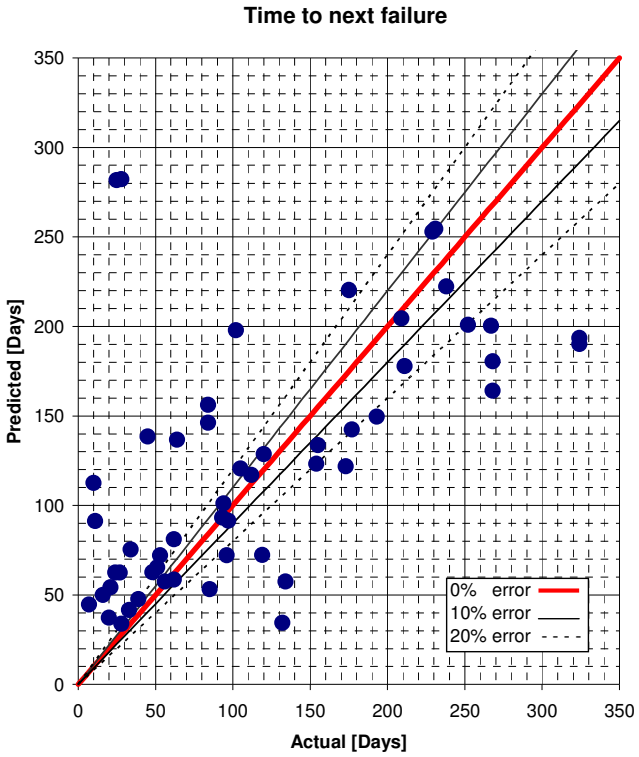

Figure 54: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

One has to bear in mind the limitations imposed on residual life predictions by the unpredictability of operations in a real life plant. The covariates chosen as inputs to a neural network must reflect the failure modes of the system. If a failure cannot be traced by one of these inputs, it will be impossible for the network to predict more accurately when the machine will fail. The results achieved here can therefore be seen as a conditional success in terms of the use of neural networks for this application.

Chapter 6: Closure

The advent of preventive maintenance increased the need for reliability information and led to the development of data analysis techniques for the purpose of residual life estimation. This work initially centred on probabilistic models which were fitted to failure time data but researchers have more recently investigated the use of regression models which allow explanatory variables to be incorporated. Condition based data is commonly available and can be used for this purpose and was found to enhance the accuracy of predictions.

Neural networks have been used previously for maintenance related applications, but their use has not yet been fully explored. They can learn the underlying relationship between various inputs and outputs and are therefore ideally suited for making predictions with respect to complex systems. In order to build upon the work done with regression models, the neural networks were designed to exploit the advantages offered by incorporating explanatory variables into the training process. Such data formed an integral component of the work and it was found that the use of covariates containing historic information and condition data helped to improve the accuracy of the predictions that were made.

The use of neural networks for making of failure predictions for both renewal and repaired cases was investigated. The estimates made by the networks with respect to the simulated renewal system proved to be very accurate, with the average error varying between 431 seconds and 841 seconds for the different neural network types. This compares well with the measurement interval of 180 seconds which was used. It was shown that much greater accuracy could be achieved with neural networks than through the use of the common probabilistic technique which involves fitting of a Weibull distribution to the failure time data. The performance of the neural networks were compared with this statistical method on the basis of predictions that were made when the networks were presented with the first set of values measured on the test pieces allocated to the test set. The MLP neural network trained with the Levenberg-Marquardt algorithm using Bayesian regularization did not exceed a prediction error of 5.1%. In comparison, the error of life estimates made using the Weibull distribution ranged between 11.2% and 55.4%. The predictions obtained with neural networks for the repaired system also compared very well with the results from the previous work done by Vlok (2001) using regression methods.

The failure predictions for the repaired systems were hampered by the combination of the system's complexity and the sparseness of the dataset. The sparseness of the dataset limits the number of inputs that can be used for MLP networks and also means that the input space is poorly mapped. Repaired systems have multiple life intervals that are not independent and are subject to numerous failure modes which place a severe challenge on the analyst. The small number of inputs and poorly mapped input space meant that the explanatory information proved to be insufficient for the network to accurately model the system and large errors were recorded on some of the test data.

The difficulties encountered in the prediction of residual life for the repaired system have highlighted some areas that require further investigation. Research should focus

on quantifying the factors that significantly influence equipment life but are hard to measure. Operating conditions may change and loads may fluctuate substantially during the life of a piece of equipment. The effects of such changes on machine condition can normally be detected through standard condition monitoring techniques, but it would be of great benefit if a cumulative effective workload can be calculated. The impact of repairs, changing machine composition and human intervention upon the reliability of machines is an important area that is not fully understood and proves difficult to measure. The study of human reliability, especially the aspect of human error, may provide some answers in this regard. An expert panel, consisting of a number of experienced maintenance practitioners and engineers, may also serve to shed light on grey areas whose effect is not directly measurable. Information from such alternative sources will serve to complement condition monitoring data in the prediction of residual life, especially at times when no clear deterioration in equipment condition can be detected. The approach followed in this work, using neural networks to estimate a single residual life, is unlikely to provide satisfactory answers for complex systems in practice. The splitting of the analysis according to failure modes and the use of competitive algorithms should form the basis of future research on the implementation of neural networks in the maintenance field.

With respect to the comparison between different neural network methods, the use of Bayesian regularization proved to be very effective in the prevention of overfitting. The use of a second order method such as the Levenberg-Marquardt training algorithm showed a significant reduction in training time when compared to the gradient descent method. The optimization of network parameters proved an important part of the training process and it was found that the performance of different network types were very closely matched once their design was adjusted to suit the specific application. GRNN are simple neural networks that are easily generated and proved to match the MLP networks very closely with a difference of 11.1% on the sum of squares error for the repaired system dataset.

The ease with which neural networks can be trained and the quality of the results that were achieved for the two datasets indicates that neural networks should become a very useful tool for the analysis of reliability data in future. Clearly the approach outlined in this project is not suitable for every application in the maintenance field, but the results show the potential of neural networks as a powerful tool for the analysis of reliability data and the prediction of residual life.

Appendix A

The table shows the average load and amplitude measured at the start of the test run of each test piece.

Table A1: Average load, load amplitude and test piece life.

No.	Data ID	Average Load	Amplitude	Life
1	TSA_2	-104 kN	280 kN	1805 sec.
2	TSA_3	-64 kN	313 kN	1203 sec.
3	TSA_5	-102 kN	190 kN	11320 sec.
4	TSA_8	54 kN	175 kN	12703 sec.
5	TSA_9	-49 kN	216 kN	6222 sec.
6	TSA_20	-33 kN	207 kN	14071 sec.
7	TSA_22	-33 kN	214 kN	10283 sec.
8	TSA_23	-33 kN	212 kN	10102 sec.
9	TSA_24	-33 kN	231 kN	5772 sec.
10	TSA_25	-34 kN	230 kN	4509 sec.
11	TSA_26	-25 kN	212 kN	10103 sec.
12	TSA_27	-25 kN	212 kN	10464 sec.

Appendix B

The Graphs illustrating Network Performance as discussed in Chapter 4.

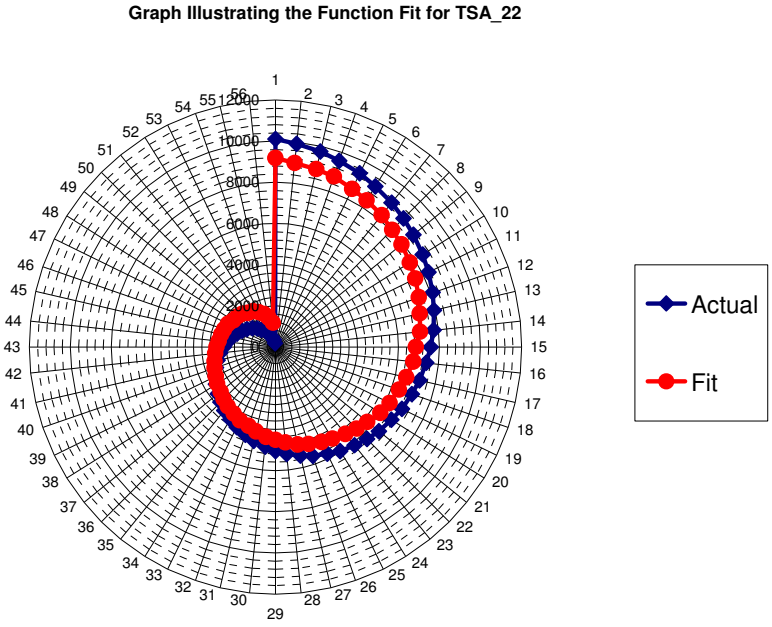

Figure 55: Results achieved in residual life estimates on the dataset of TSA_22 by the Neural Network trained with the Steepest Gradient Algorithm

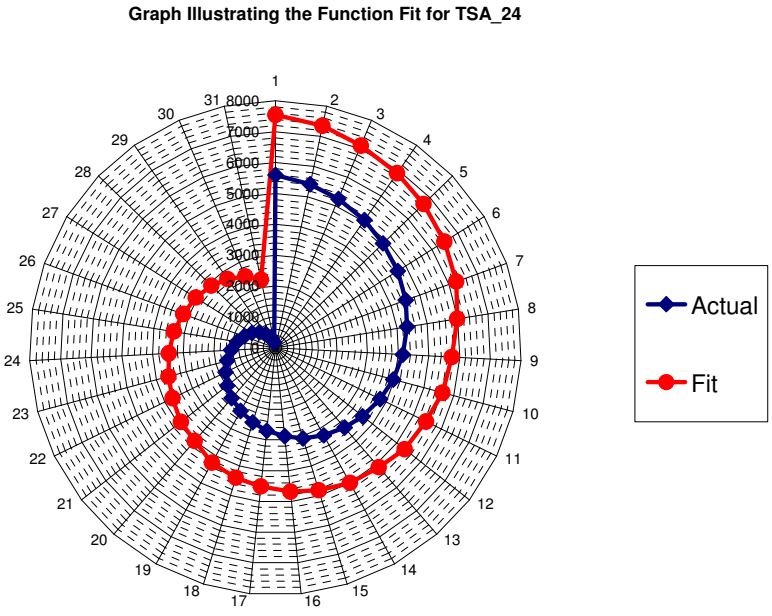

Figure 56: Results achieved in residual life estimates on the dataset of TSA_24 by the Neural Network trained with the Steepest Gradient Algorithm

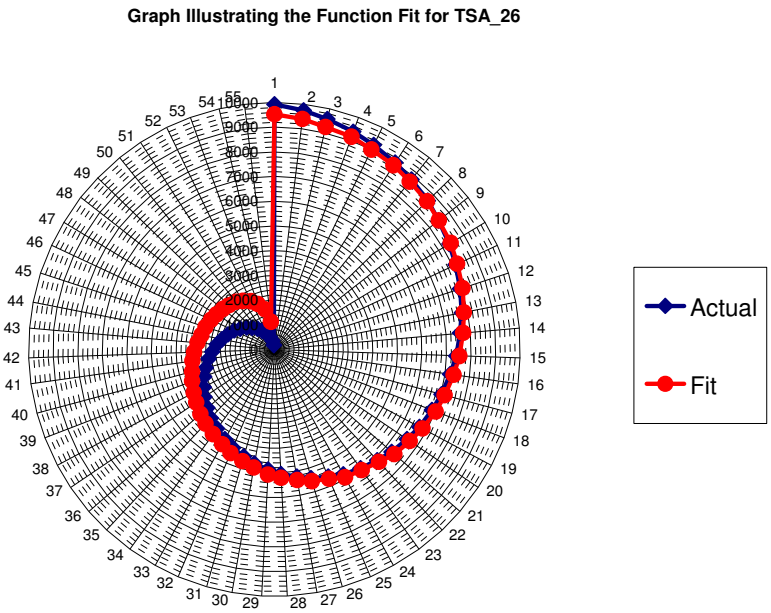

Figure 57: Results achieved in residual life estimates on the dataset of TSA_26 by the Neural Network trained with the Steepest Gradient Algorithm

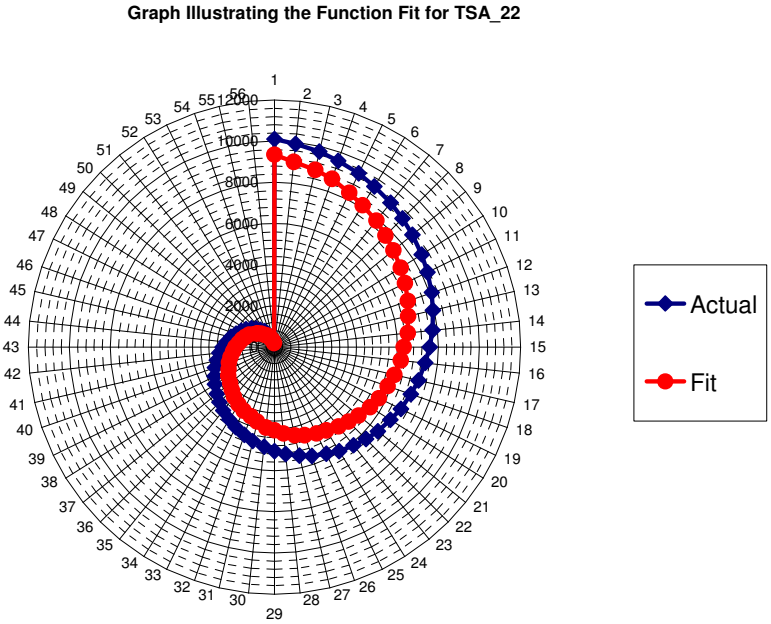

Figure 58: Results achieved in residual life estimates on the dataset of TSA_22 by the Neural Network trained with the Levenberg-Marquardt Algorithm

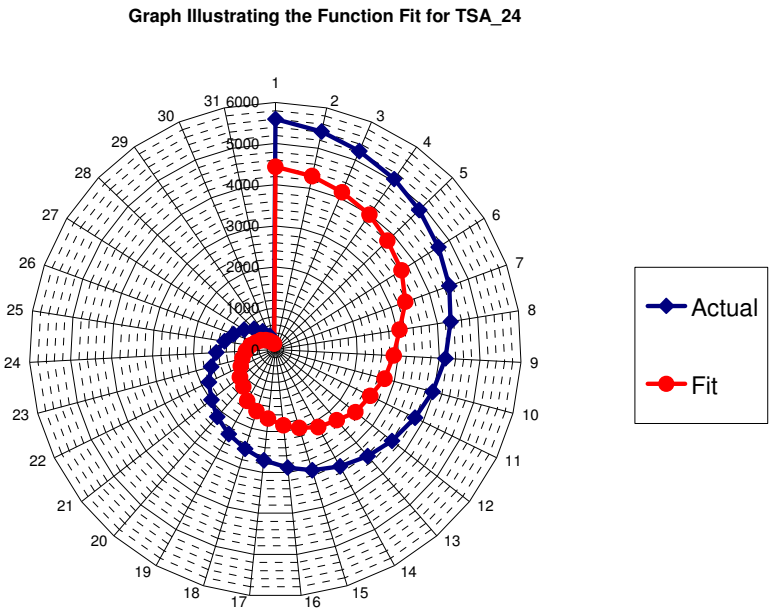

Figure 59: Results achieved in residual life estimates on the dataset of TSA_24 by the Neural Network trained with the Levenberg-Marquardt Algorithm

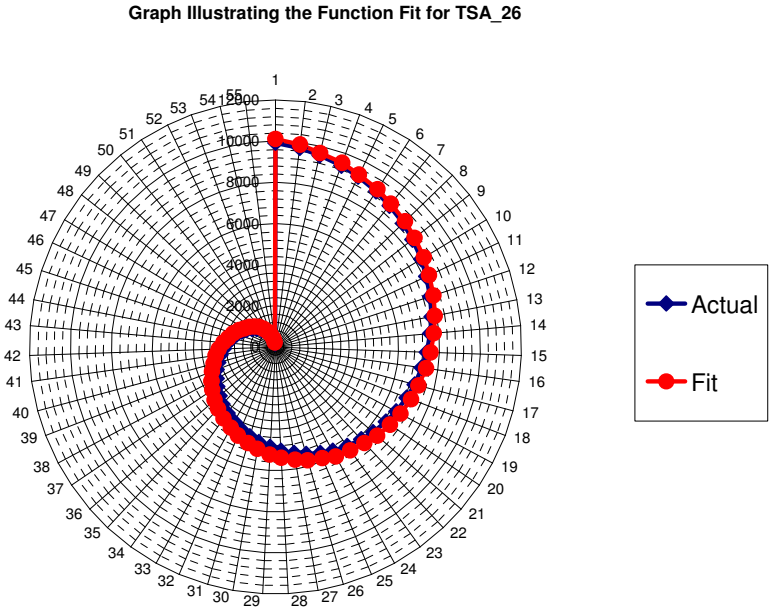

Figure 60: Results achieved in residual life estimates on the dataset of TSA_26 by the Neural Network trained with the Levenberg-Marquardt Algorithm

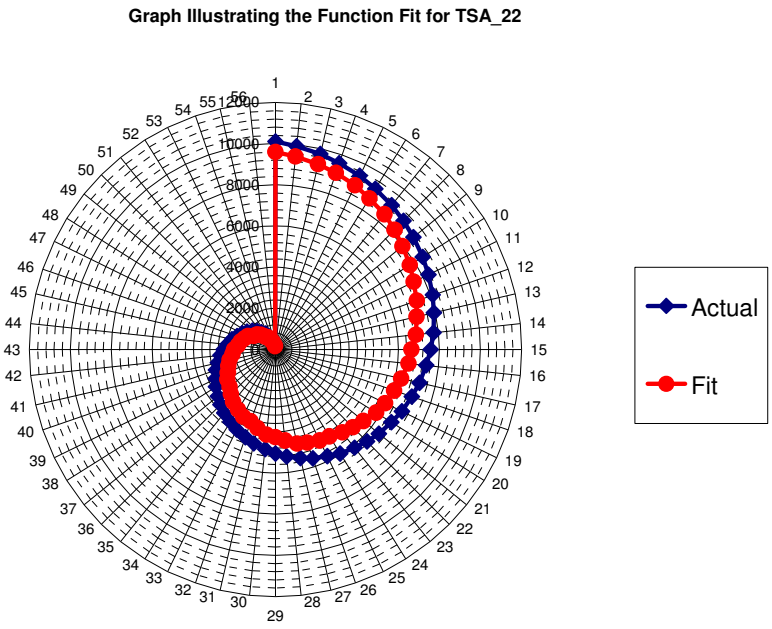

Figure 61: Results achieved in residual life estimates on the dataset of TSA_22 by the Neural Network trained with the Levenberg-Marquardt Algorithm and Bayesian Regularization

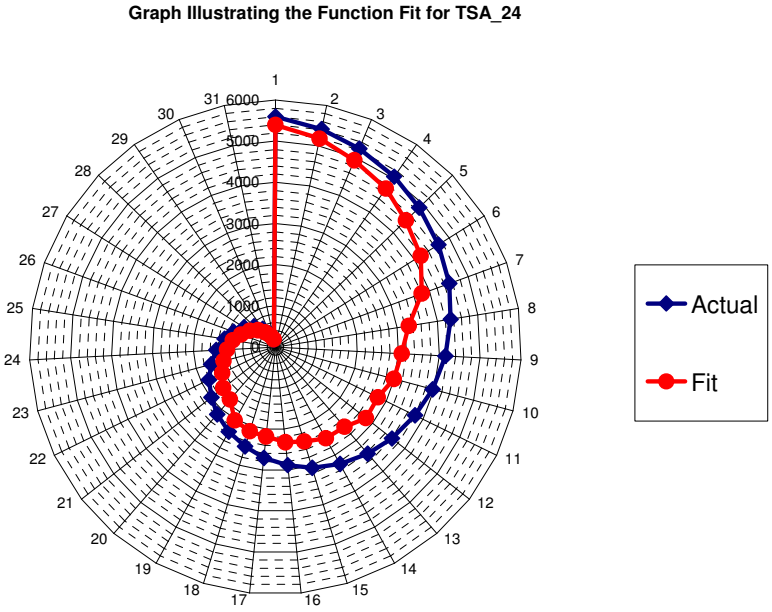

Figure 62: Results achieved in residual life estimates on the dataset of TSA_24 by the Neural Network trained with the Levenberg-Marquardt Algorithm and Bayesian Regularization

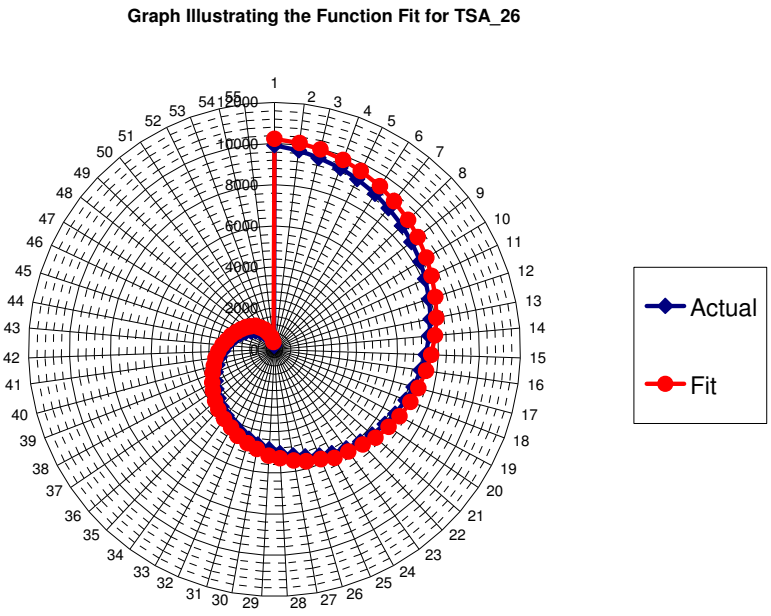

Figure 63: Results achieved in residual life estimates on the dataset of TSA_26 by the Neural Network trained with the Levenberg-Marquardt Algorithm and Bayesian Regularization

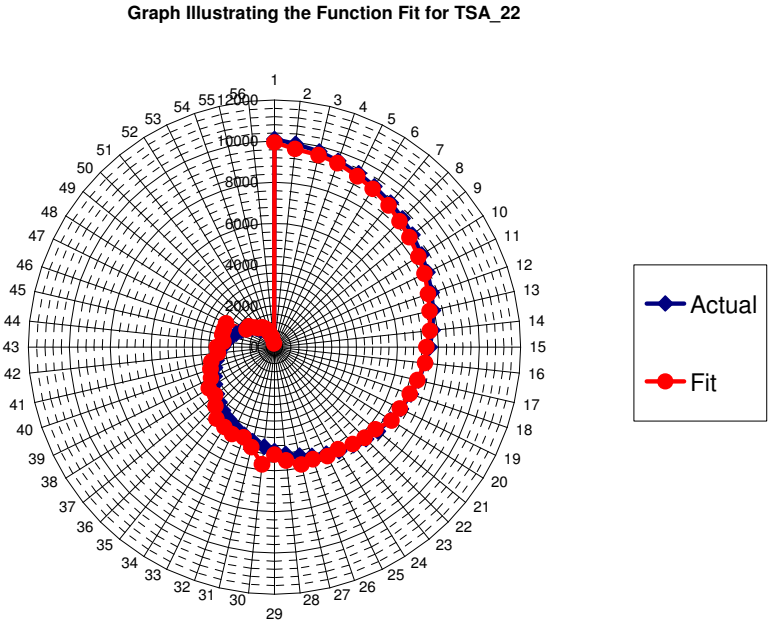

Figure 64: Results achieved in residual life estimates on the dataset of TSA_22 by the General Regression Neural Network

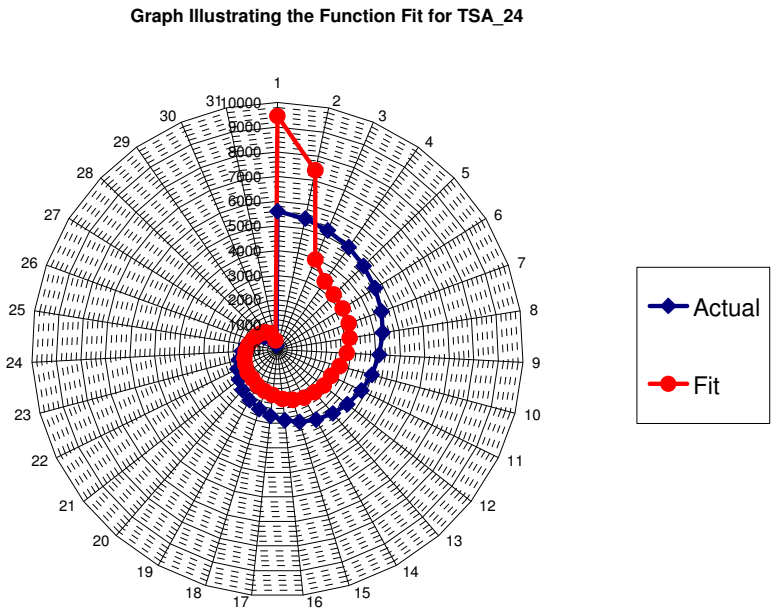

Figure 65: Results achieved in residual life estimates on the dataset of TSA_24 by the General Regression Neural Network

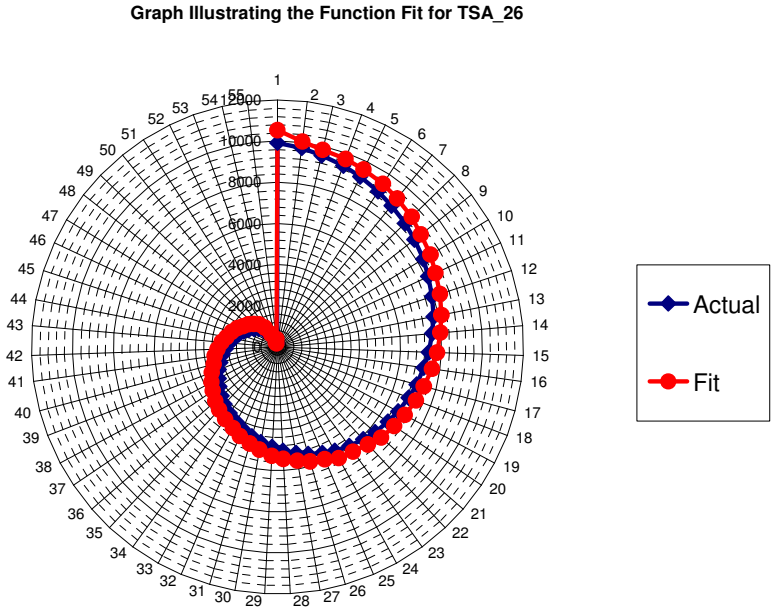

Figure 66: Results achieved in residual life estimates on the dataset of TSA_26 by the General Regression Neural Network.

Appendix C

The dataset from Vlok (1999, 2001) formed the basis of the comparison in Chapter 5. On Table C1 the pump reference numbers used for the graphs in Chapter 5 and the corresponding Pump ID used by Vlok are listed for easy reference.

Vlok (1999, 2001) identified the vibration amplitude at 5 × rotational frequency as the most significant measurement for determining the condition of the pumps. Table C2 gives the magnitude of this particular measurement at Bearing 3 (Measurement 1) and Bearing 4 (Measurement 2).

On Table C3 the maintenance events are listed for each of the pumps according to the information supplied by Vlok. Unexpected breakdowns are listed as "failure", maintenance work and repair prompted by high vibration readings is termed as "maintenance" while a calendar based truncation of the data is referred to as a "suspension".

Table C1: Pump ID Numbers

Pump Ref. No. (Chapter 5)	Pump ID
1	PC1131
2	PC1132
3	PC1231
4	PC1232
5	PC2131
6	PC2132
7	PC3132
8	PC3232

Table C2: Condition Monitoring Data

Pump ID	Global Age (Days)	Date of Measurement	RF043H [mm/s]	RF044H [mm/s]	RF053H [mm/s]	RF054H [mm/s]
PC1131	159	07/02/97	0	0.05	0.8	0.1
PC1131	295	23/06/97	0.15	0.2	0.55	0.12
PC1131	387	23/09/97	0.3	0.1	8	6.2
PC1131	394	30/09/97	0.8	2.3	12.3	5
PC1131	397	03/10/97	250	4	17	6
PC1131	530	13/02/98	0.1	0.1	11	5.5
PC1131	533	16/02/98	0.3	0.2	13	7
PC1131	554	09/03/98	0.5	0.3	16	10
PC1131	578	02/04/98	1	0.7	2	3
PC1131	597	21/04/98	0.3	0.5	1.6	5
PC1131	639	02/06/98	0.5	0.5	4	5
PC1131	689	22/07/98	0	0	0.8	1.2
PC1131	690	23/07/98	0	0	0.67	1.08
PC1131	703	05/08/98	0.05	0.2	0.2	0.4
PC1131	712	14/08/98	0.05	0.05	1.4	0.41
PC1131	765	06/10/98	0.05	0.05	2.7	0.6
PC1131	791	01/11/98	0.5	0.2	12	7
PC1132	239	28/04/97	0	0	1.5	0.72
PC1132	386	22/09/97	0.1	0.1	2.1	7.8
PC1132	394	30/09/97	0.2	0.1	11	8.2
PC1132	397	03/10/97	0.1	0.2	3	12
PC1132	491	05/01/98	0.1	1	1	30

Pump ID	Global Age (Days)	Date of Measurement	RF043H [mm/s]	RF044H [mm/s]	RF053H [mm/s]	RF054H [mm/s]
PC1132	499	13/01/98	0.1	0.1	2.5	12
PC1132	533	16/02/98	0.1	0	12	10
PC1132	543	26/02/98	5	1	9	7
PC1132	544	27/02/98	5.61	1.13	8.56	6.64
PC1132	557	12/03/98	3	1	2	2.5
PC1132	558	13/03/98	1	2	3	1
PC1132	597	21/04/98	4	1	2.6	5.4
PC1132	689	22/07/98	0.1	0.1	0.3	0.4
PC1132	712	14/08/98	0.1	0.05	0.9	0.4
PC1132	751	22/09/98	0.99	0.13	2.99	1.54
PC1132	791	01/11/98	0.08	0.15	2.01	7.68
PC1231	239	28/04/97	0.3	0	1	0.4
PC1231	295	23/06/97	1.3	0.3	1	0.3
PC1231	390	26/09/97	1	0	3	4
PC1231	530	13/02/98	0.3	0	8.5	6
PC1231	563	18/03/98	0.09	0.08	10.24	5.87
PC1231	578	02/04/98	1	2	6	9
PC1231	653	16/06/98	0.22	0	0.57	0.27
PC1231	698	31/07/98	0.68	0.22	0.61	0.15
PC1231	791	01/11/98	0.73	0	1.86	2.64
PC1232	583	07/04/98	0.5	0	4	3
PC1232	592	16/04/98	0.4	0.05	6.5	2
PC1232	597	21/04/98	0.6	1	3.5	3
PC1232	599	23/04/98	0.05	0.15	0.6	0.9

Pump ID	Global Age (Days)	Date of Measurement	RF043H [mm/s]	RF044H [mm/s]	RF053H [mm/s]	RF054H [mm/s]
PC1232	699	01/08/98	0.33	0	2.48	1.92
PC1232	791	01/11/98	0.24	0.03	4.09	1.24
PC2131	156	04/02/97	0	0	0.4	0.5
PC2131	159	07/02/97	0.1	0	0.6	0.4
PC2131	178	26/02/97	0.2	0.05	1.35	0.4
PC2131	179	27/02/97	0	0	0.9	0.4
PC2131	184	04/03/97	0	0	1	0.4
PC2131	239	28/04/97	0.09	0.05	1.55	0.7
PC2131	241	30/04/97	0.05	0.1	1.7	0.7
PC2131	295	23/06/97	0.1	0.2	1.4	0.4
PC2131	386	22/09/97	0.4	1.7	0.7	3.7
PC2131	470	15/12/97	1200	78	10	9
PC2131	535	18/02/98	0.2	0.5	4.8	7
PC2131	583	07/04/98	2	2	11	6
PC2131	597	21/04/98	2	2	6	4
PC2131	604	28/04/98	1	2	5	5
PC2131	611	05/05/98	0.01	0.01	11.6	1.4
PC2131	631	25/05/98	0.1	0.01	72.33	1
PC2131	640	03/06/98	0.6	0.2	5.9	4
PC2131	689	22/07/98	0.09	0.05	0.5	0.33
PC2131	768	09/10/98	0.1	0.05	0.66	0.2
PC2131	774	15/10/98	0.14	0.06	1.12	0.48
PC2131	791	01/11/98	0.16	0.34	3.69	5.6
PC3131	241	30/04/97	0.1	0.1	1.3	1

Pump ID	Global Age (Days)	Date of Measurement	RF043H [mm/s]	RF044H [mm/s]	RF053H [mm/s]	RF054H [mm/s]
PC3131	295	23/06/97	0.8	0.7	14	7
PC3131	386	22/09/97	0.5	2	4	7
PC3131	450	25/11/97	0.2	3.13	3	2
PC3131	550	05/03/98	0.09	0.1	1.27	1.2
PC3131	651	14/06/98	0.96	0.71	16.8	7.7
PC3131	750	21/09/98	0.59	2.4	4.16	6.58
PC3131	791	01/11/98	0.2	3.47	3.39	1.8
PC3132	239	28/04/97	0.1	0.2	0.39	0.55
PC3132	295	23/06/97	0.2	0.3	1.6	2.2
PC3132	386	22/09/97	0.2	0.05	3.5	2.4
PC3132	450	25/11/97	0.5	0	13	6.5
PC3132	506	20/01/98	0.97	0.04	26.84	12.77
PC3132	566	21/03/98	0.12	0.23	0.45	0.59
PC3132	711	13/08/98	0.19	0.37	1.82	2.35
PC3132	791	01/11/98	0.2	0.06	3.39	2.61
PC3232	239	28/04/97	0.3	0.01	0.6	0.3
PC3232	295	23/06/97	1	1	6	4
PC3232	386	22/09/97	2	1	6	3
PC3232	535	18/02/98	0	0	7	8
PC3232	563	18/03/98	0	0	7.33	9.86
PC3232	591	15/04/98	0	0	10	15
PC3232	604	28/04/98	2	0	7	8
PC3232	639	02/06/98	3	5	3	6
PC3232	722	24/08/98	0	0	1.9	0.8

Pump ID	Global Age (Days)	Date of Measurement	RF043H [mm/s]	RF044H [mm/s]	RF053H [mm/s]	RF054H [mm/s]
PC3232	723	25/08/98	0	0	1.96	0.73
PC3232	748	19/09/98	0.18	0	0.39	0.23
PC3232	783	24/10/98	0.62	0.73	4.5	3.2
PC3232	791	01/11/98	1.28	0.72	4.08	1.95

Table C3: Table of Events

Pump ID	Global Age (Days)	Date of Event	Event Description
PC1131	0	01/09/96	Start
PC1131	397	03/10/97	Maintenance
PC1131	397	03/10/97	Start
PC1131	554	09/03/98	Failure
PC1131	554	09/03/98	Start
PC1131	690	23/07/98	Maintenance
PC1131	690	23/07/98	Start
PC1131	765	06/10/98	Failure
PC1131	765	06/10/98	Start
PC1131	791	01/11/98	Suspension
PC1132	0	01/09/96	Start
PC1132	491	05/01/98	Failure
PC1132	491	05/01/98	Start
PC1132	544	27/02/98	Maintenance
PC1132	544	27/02/98	Start
PC1132	557	12/03/98	Maintenance
PC1132	557	12/03/98	Start
PC1132	751	22/09/98	Failure
PC1132	751	22/09/98	Start
PC1132	791	01/11/98	Suspension
PC1231	0	01/09/96	Start
PC1231	563	18/03/98	Failure
PC1231	563	18/03/98	Start

PC1231	578	02/04/98	Maintenance
PC1231	578	02/04/98	Start
PC1231	791	01/11/98	Suspension
PC1232	0	01/09/96	Start
PC1232	599	23/04/98	Maintenance
PC1232	599	23/04/98	Start
PC1232	791	01/11/98	Suspension
PC2131	0	01/09/96	Start
PC2131	184	04/03/97	Failure
PC2131	184	04/03/97	Start
PC2131	470	15/12/97	Maintenance
PC2131	470	15/12/97	Start
PC2131	631	25/05/98	Failure
PC2131	631	25/05/98	Start
PC2131	774	15/10/98	Failure
PC2131	774	15/10/98	Start
PC2131	791	01/11/98	Suspension
PC3131	0	01/09/96	Start
PC3131	450	25/11/97	Failure
PC3131	450	25/11/97	Start
PC3131	791	01/11/98	Suspension
PC3132	0	01/09/96	Start
PC3132	506	20/01/98	Failure
PC3132	506	20/01/98	Start
PC3132	791	01/11/98	Suspension
PC3232	0	01/09/96	Start

PC3232	563	18/03/98	Failure
PC3232	563	18/03/98	Start
PC3232	723	25/08/98	Maintenance
PC3232	723	25/08/98	Start
PC3232	791	01/11/98	Suspension

Appendix D

Comparison of the results achieved with the Levenberg-Marquardt algorithm using different target values during training as discussed in Chapter 5.

No.	Network Architecture	Inputs	Target	Σ (error)²
1	5 hidden nodes	5	0.05	2.70×10^{5}
2	5 hidden nodes	4	0.025	2.85×10^{5}
3	4 hidden nodes	4	0.025	2.90×10^{5}
4	3 hidden nodes	3	0.05	2.92×10^{5}
5	4 hidden nodes	4	0.05	2.92×10^{5}
6	4 hidden nodes	3	0.025	3.06×10^{5}
7	5 hidden nodes	5	0.025	3.18×10^{5}
8	4 hidden nodes	4	0.01	3.41×10^{5}
9	5 hidden nodes	4	0.05	3.44×10^{5}
10	3 hidden nodes	3	0.025	3.47×10^{5}
11	4 hidden nodes	4	0.005	3.48×10^{5}
12	4 hidden nodes	4	0.00001	3.59×10^{5}
13	4 hidden nodes	3	0.05	3.59×10^{5}
14	3 hidden nodes	3	0.005	3.60×10^{5}
15	5 hidden nodes	5	0.005	3.66×10^{5}
16	3 hidden nodes	3	0.01	3.81×10^{5}
17	5 hidden nodes	4	0.01	3.87×10^{5}
18	4 hidden nodes	3	0.005	4.02×10^{5}
19	5 hidden nodes	4	0.005	4.04×10^{5}

No.	Network Architecture	Inputs	Target	Σ (error)²
20	5 hidden nodes	4	0.00001	4.14×10^{5}
21	3 hidden nodes	3	0.00001	4.32×10^{5}
22	4 hidden nodes	3	0.00001	4.45×10^{5}
23	4 hidden nodes	3	0.01	4.49×10^{5}
24	5 hidden nodes	5	0.01	4.99×10^{5}
25	5 hidden nodes	5	0.00001	5.31×10^{5}

Appendix E

The graphs in this appendix serve to illustrate the cross-validation results discussed in Chapter 5.

E1: Levenberg-Marquardt algorithm

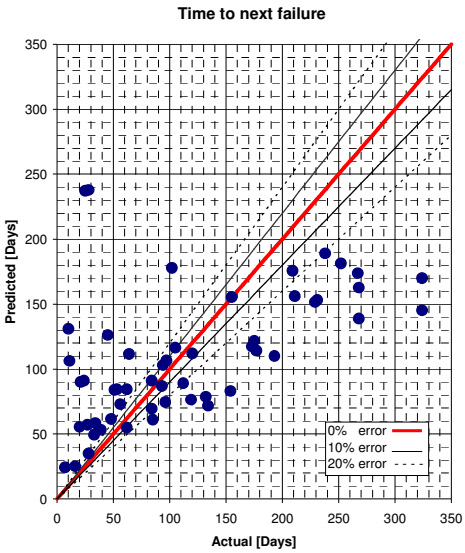

Figure 67: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-2}.

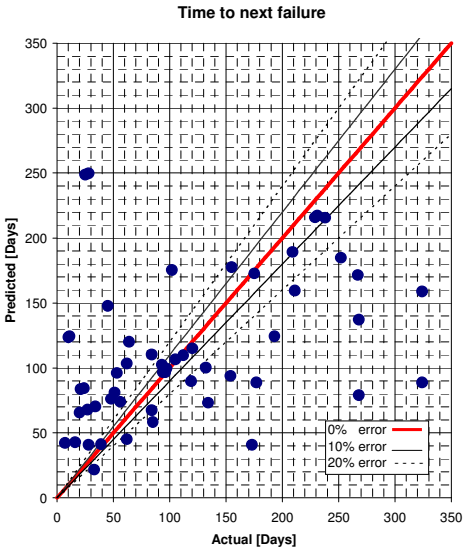

Figure 68: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-2}.

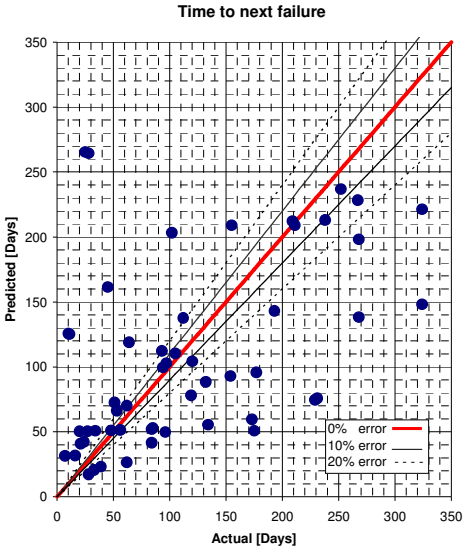

Figure 69: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 2.5×10^{-2}.

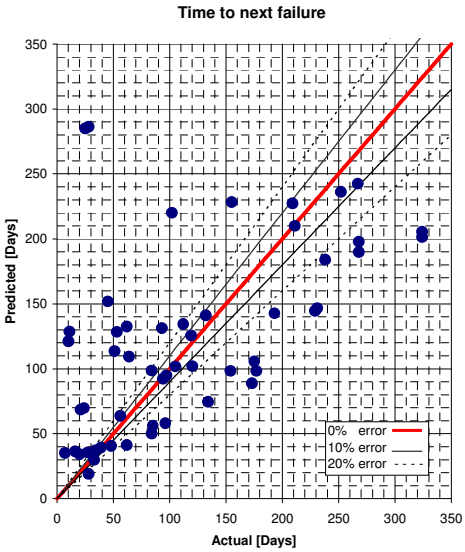

Figure 70: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 2.5×10^{-2}.

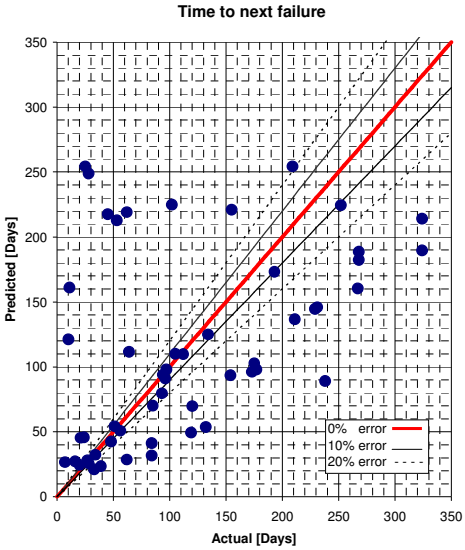

Figure 71: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-2}.

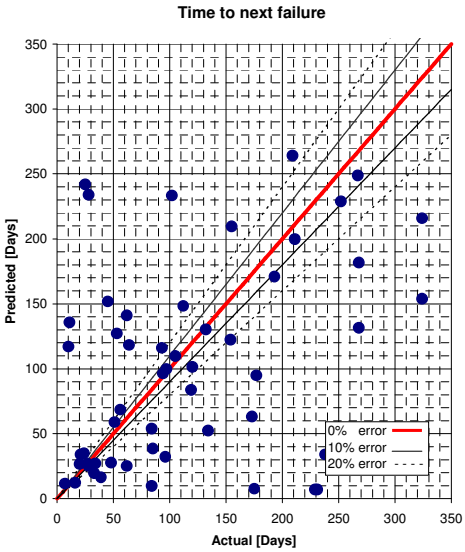

Figure 72: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-2}.

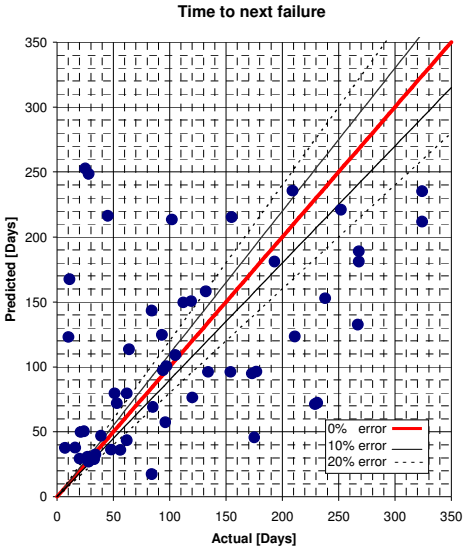

Figure 73: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-3}.

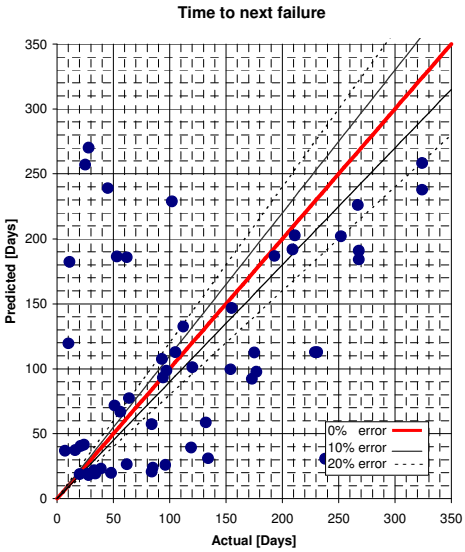

Figure 74: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-3}.

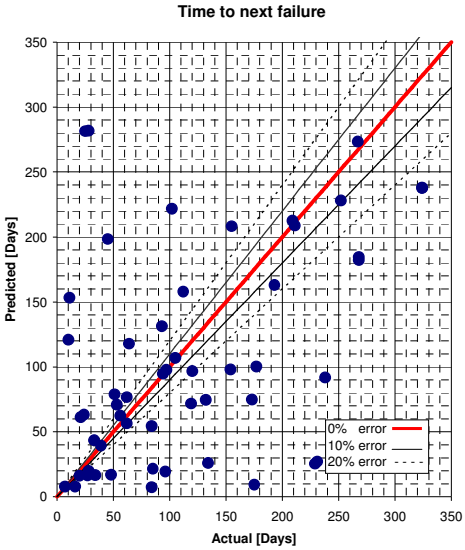

Figure 75: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-5}.

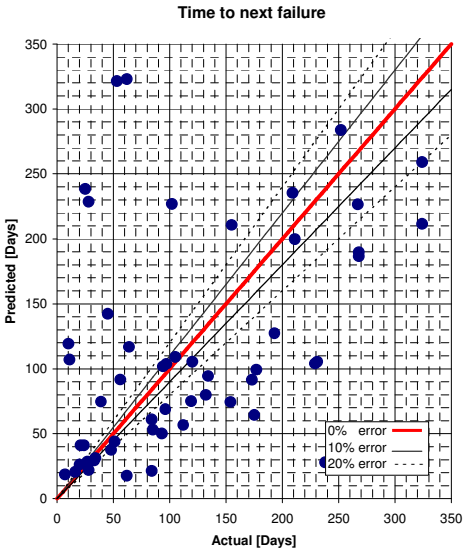

Figure 76: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-5}.

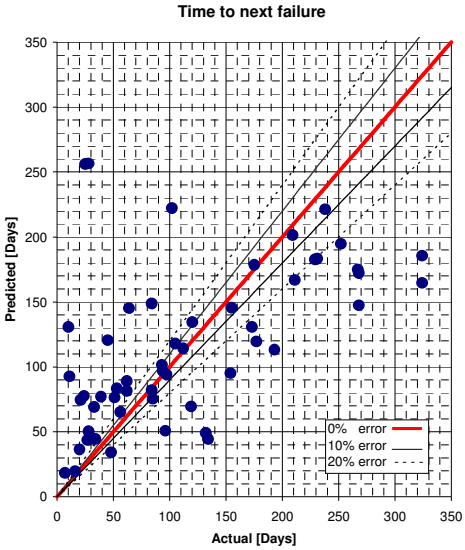

Figure 77: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-2}.

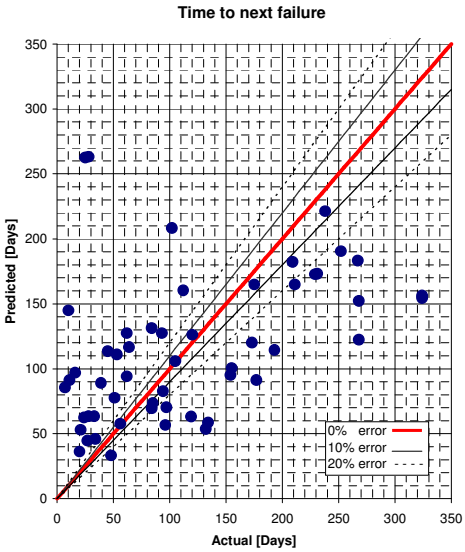

Figure 78: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-2}.

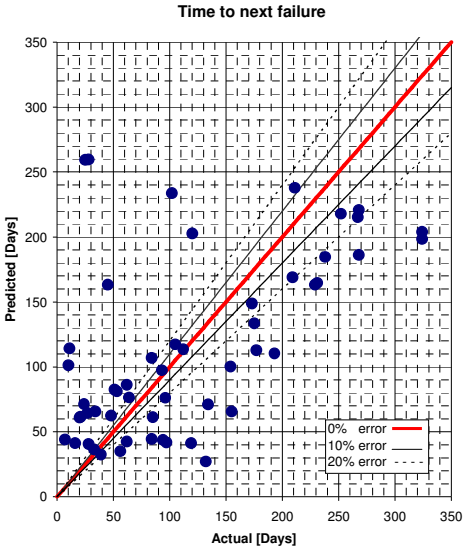

Figure 79: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 2.5×10^{-2}.

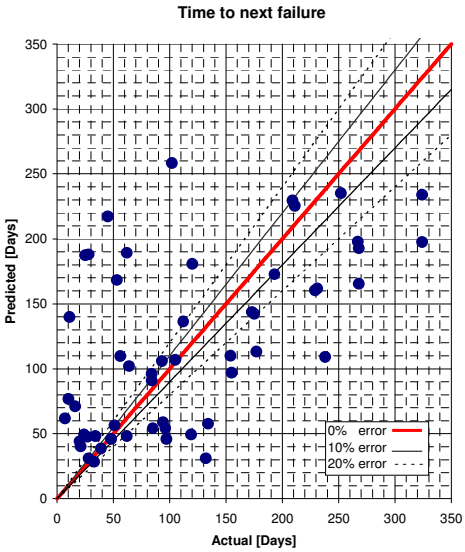

Figure 80: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 2.5×10^{-2}.

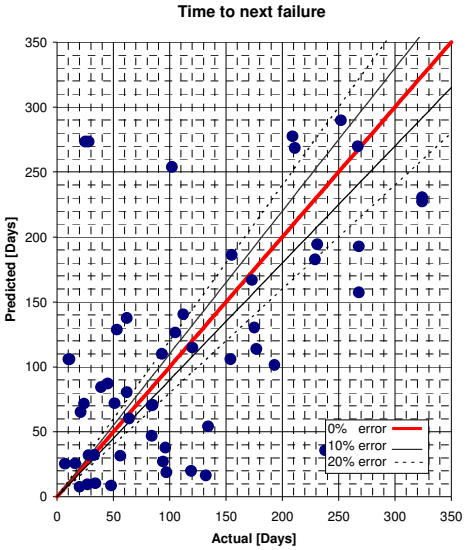

Figure 81: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-2}.

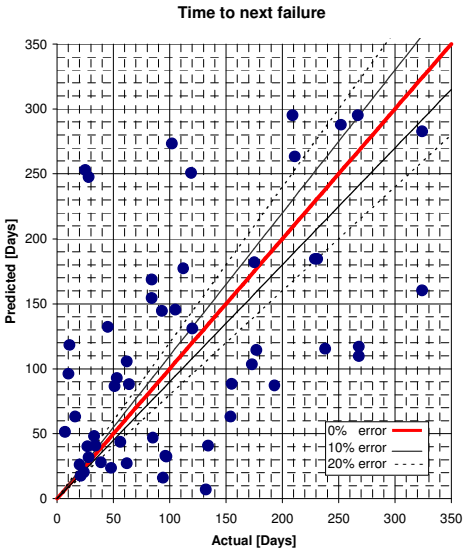

Figure 82: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-2}.

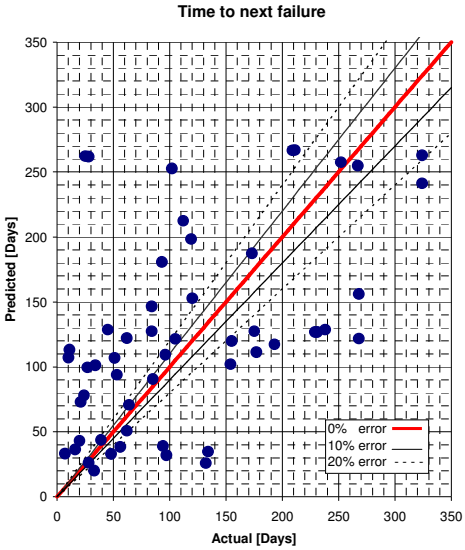

Figure 83: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-3}.

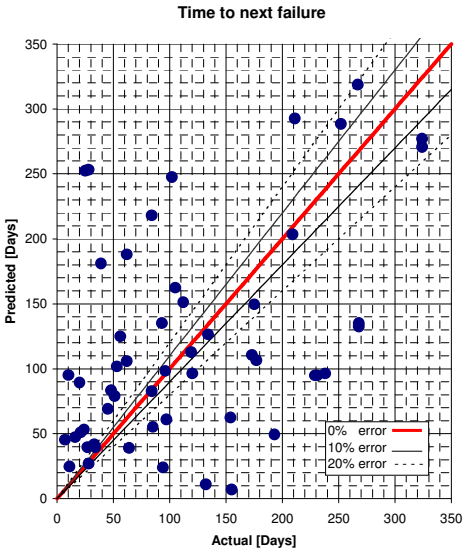

Figure 84: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-3}.

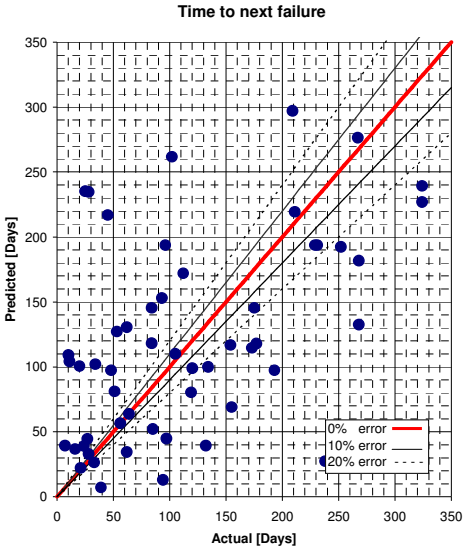

Figure 85: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-5}.

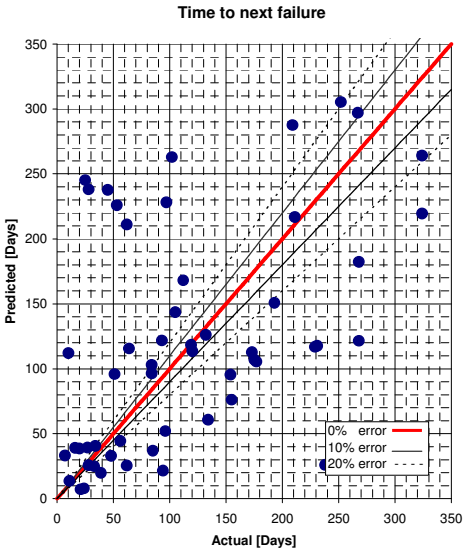

Figure 86: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-5}.

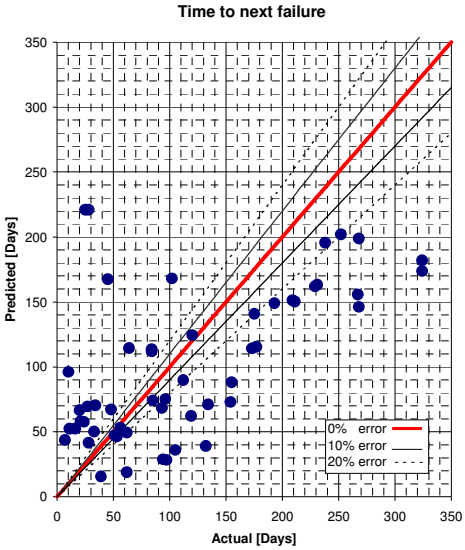

Figure 87: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-2}.

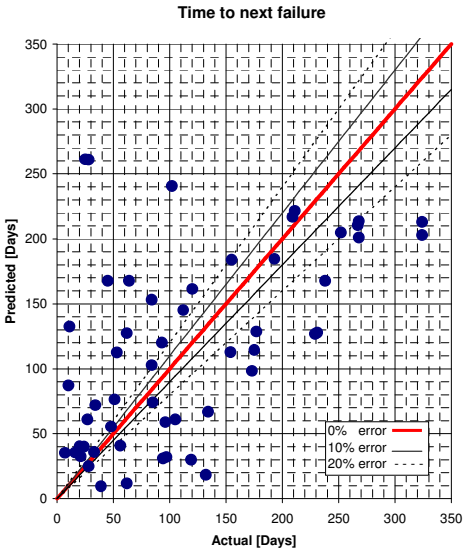

Figure 88: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 2.5×10^{-2}.

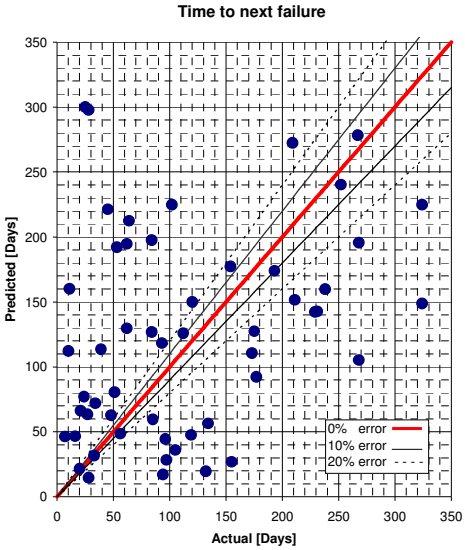

Figure 89: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-2}.

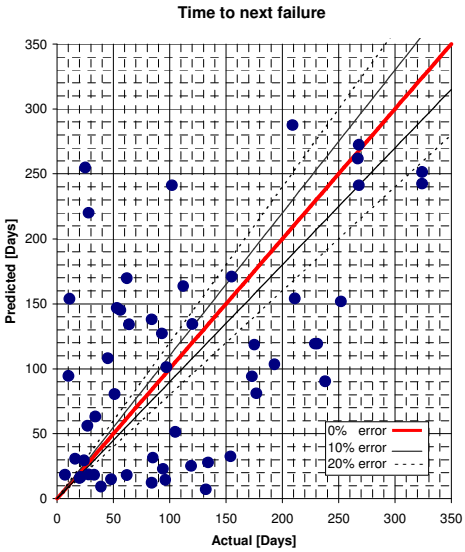

Figure 90: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 5×10^{-3}.

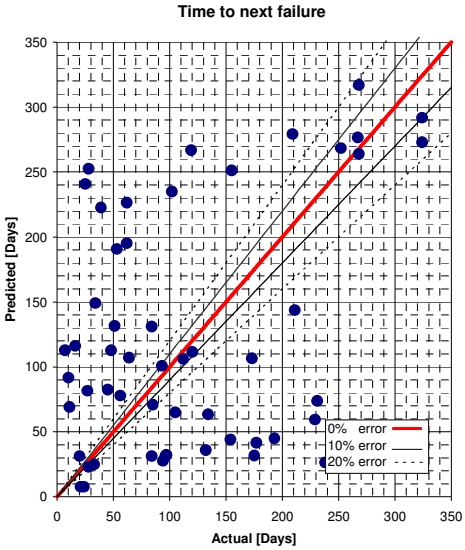

Figure 91: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with a training target of 1×10^{-5}.

E2: Levenberg-Marquardt algorithm with Bayesian regularization

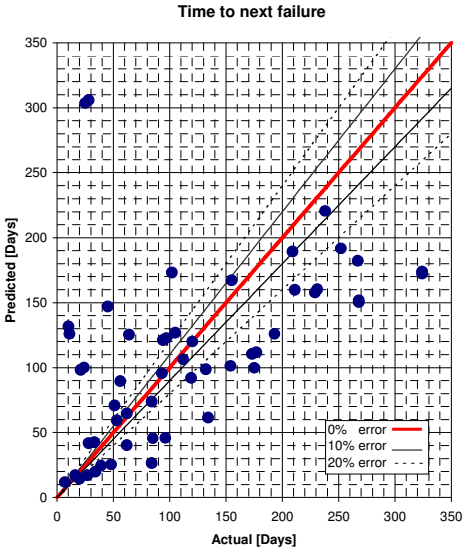

Figure 92: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

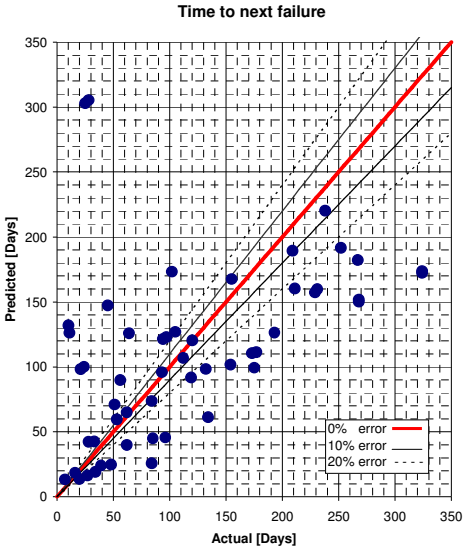

Figure 93: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

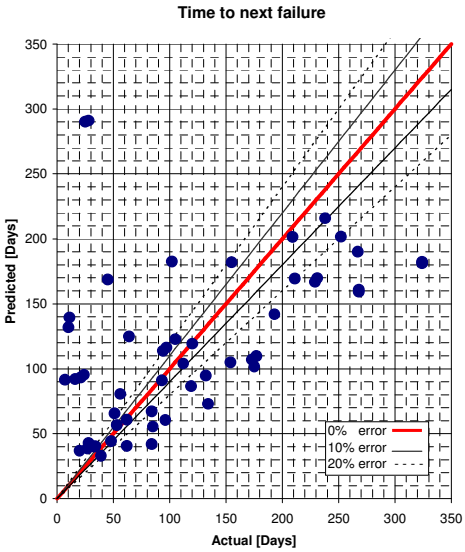

Figure 94: Cross-validation results achieved with a MLP neural network with 3 inputs, 3 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

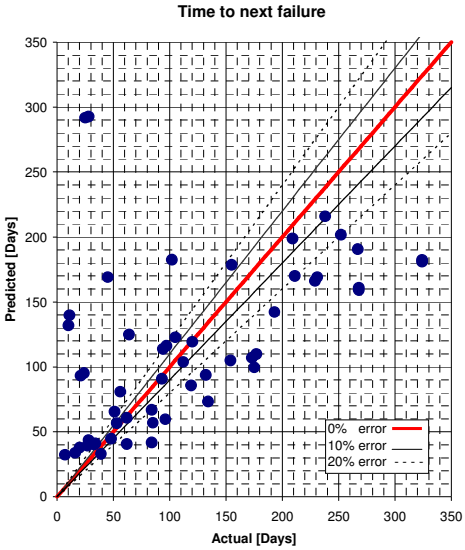

Figure 95: Cross-validation results achieved with a MLP neural network with 3 inputs, 4 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

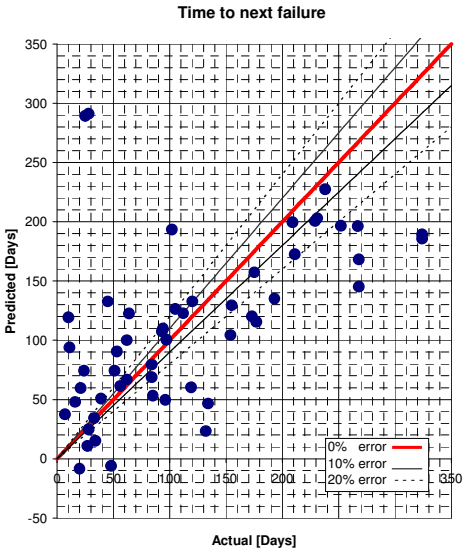

Figure 96: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

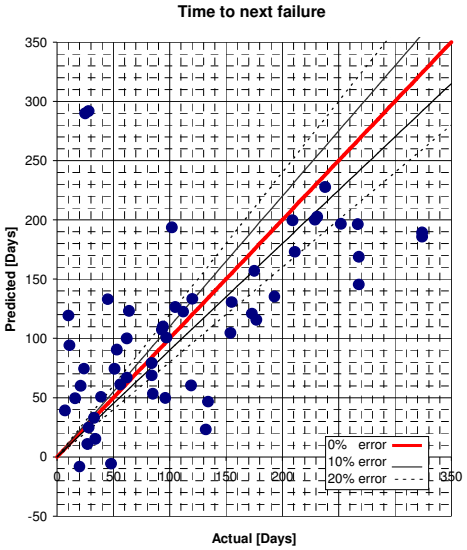

Figure 97: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

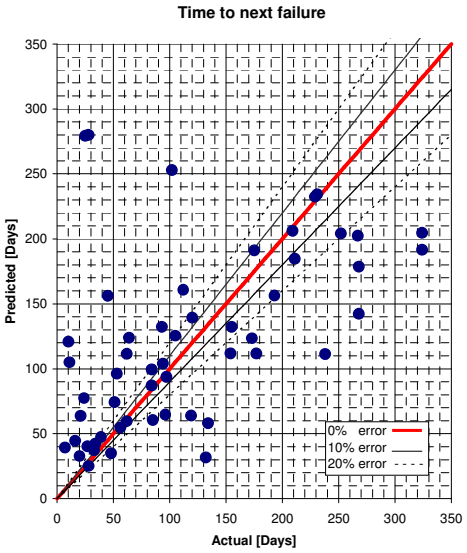

Figure 98: Cross-validation results achieved with a MLP neural network with 4 inputs, 4 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

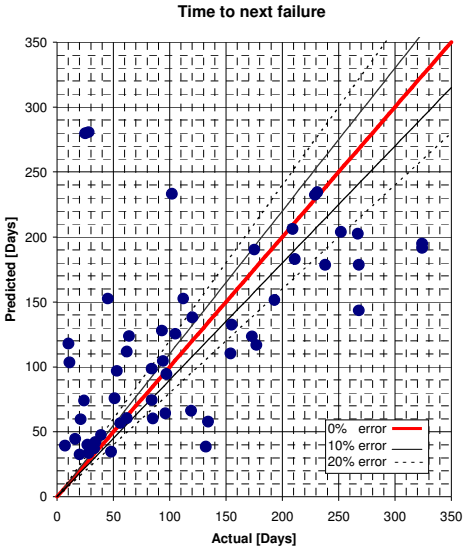

Figure 99: Cross-validation results achieved with a MLP neural network with 4 inputs, 5 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

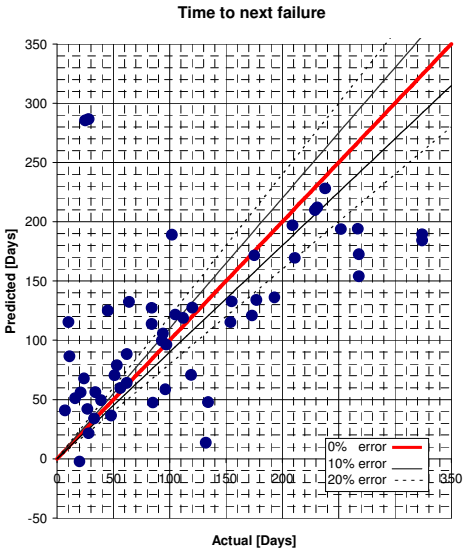

Figure 100: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a sigmoid transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

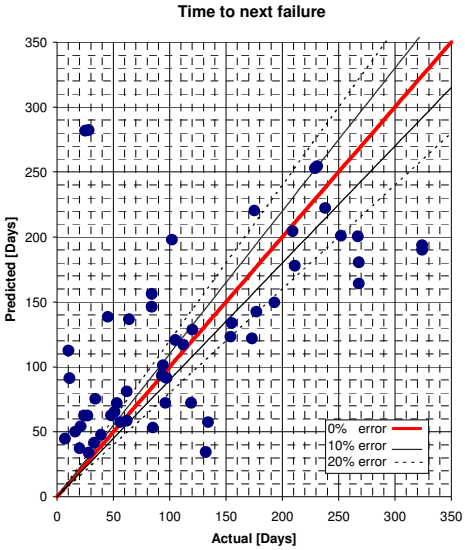

Figure 101: Cross-validation results achieved with a MLP neural network with 5 inputs, 5 hidden nodes and an output node with a linear transfer function, using the Levenberg-Marquardt training algorithm with Bayesian regularization.

E3: General regression neural network

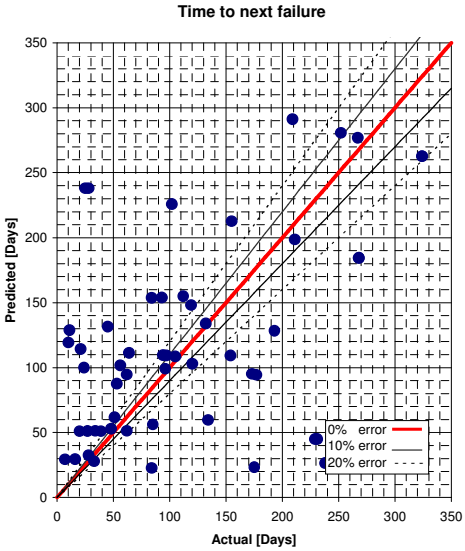

Figure 102: Cross-validation results achieved with a GRNN with 3 inputs and a spread of 0.05

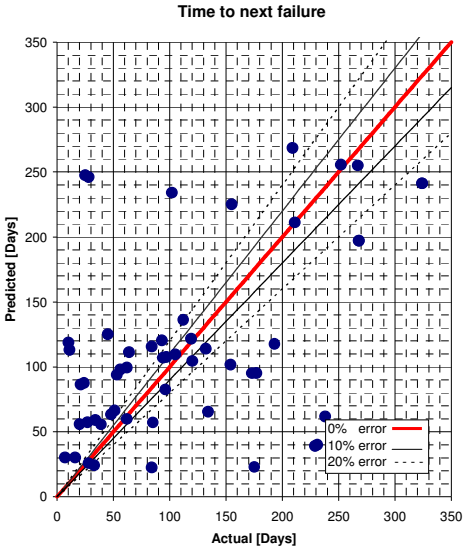

Figure 103: Cross-validation results achieved with a GRNN with 3 inputs and a spread of 0.1

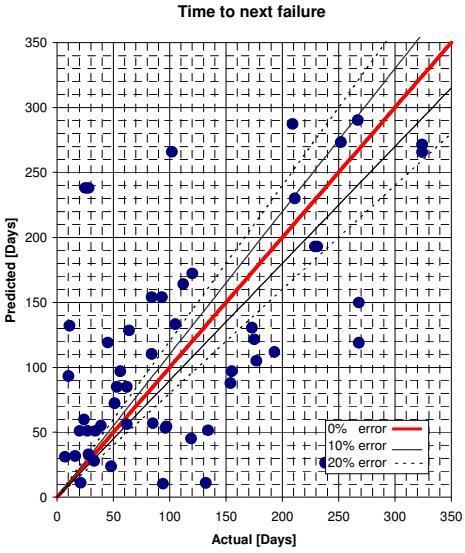

Figure 104: Cross-validation results achieved with a GRNN with 4 inputs and a spread of 0.05

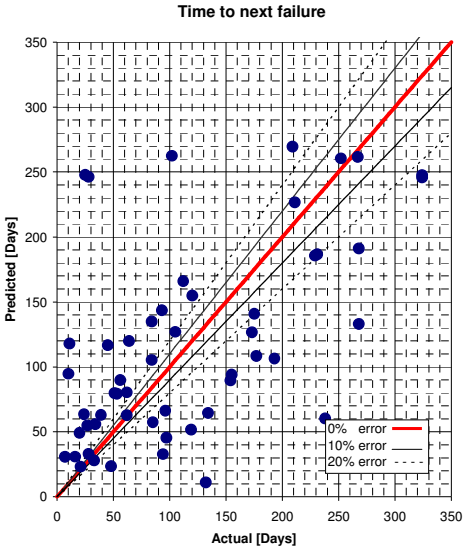

Figure 105: Cross-validation results achieved with a GRNN with 4 inputs and a spread of 0.1

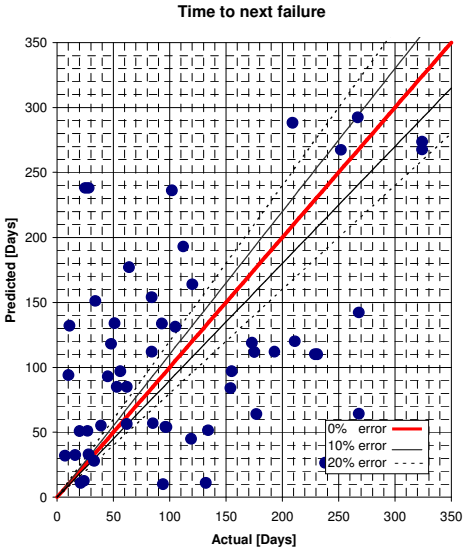

Figure 106: Cross-validation results achieved with a GRNN with 5 inputs and a spread of 0.05

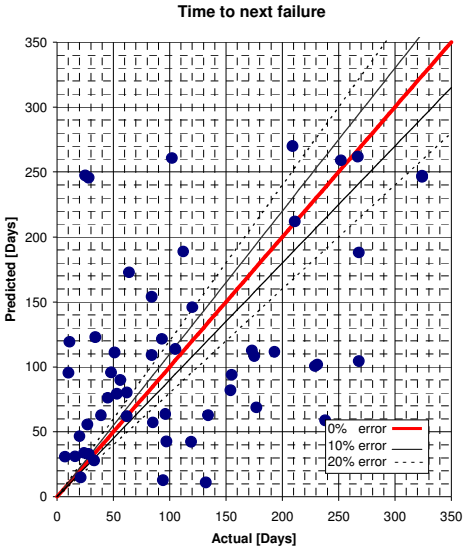

Figure 107: Cross-validation results achieved with a GRNN with 5 inputs and a spread of 0.1

References

Amjady, N. & Ehsan, M. (1999). Evaluation of power systems reliability by an artificial neural network. *IEEE Transactions on Power Systems,* **14(1)**:287-292

Anders, U. & Korn, O. (1999). Model selection in neural networks. *Neural Networks,* **12**:309-323.

Anderson, J.A. (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences,* **14**:197-220.

Anderson, J.A. and E. Rosenfeld (Eds.) (1988). *Neurocomputing: Foundations of Research.* Cambridge, MA: MIT Press

Ansell, J.I. & Phillips, M.J. (1989). Practical Problems in the Statistical Analysis of Reliability Data. *Applied Statistics,* **38(2)**:205-247

Ascher, H.E. & Feingold, H. (1978). *Is there repair after failure?* Paper presented at the 1978 Annual Reliability and Maintainability Symposium: Los Angeles, CA.

Ascher, H. (1981). Weibull Distribution vs Weibull Process. *IEEE Proc. Annual Reliability and Maintainability Symposium,* 426-431

Ascher, H.E. & Feingold, H. (1984). *Repairable Systems Reliability: Modelling, Inference, Misconceptions and their Causes.* New York: Marcel Dekker

Bain, Lee J. & Engelhardt, M. (1991) *Statistical Analysis of Reliability and Life-Testing Models. (Second Edition)* New York: Marcel Dekker Inc.

Battiti, R. (1992). First and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method. *Neural Computation.* **4**:141-166

Bennet, S. (1983). Analysis of Survival Data by the Proportional Odds Model. *Statistics in Medicine* **2**:273-277

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford, UK: Oxford University Press.

Bradley, E. (1993). Reliability Engineering Volume 1: Course Notes Rev 6. Pretoria: University of Pretoria.

Broomhead, D.S. & Lowe, D. (1988). Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems* **2** :321-355

Broyden C.G. (1996). A New Taxonomy of Conjugate Gradient Methods. *Computers Math. Applic.* **31(4/5)**:7-17

Bryson, A.E. & Ho, Y.C. (1969). *Applied Optimal Control.* New York: Blaisdell.

Chan, J-K. & Shaw, L. (1993). Modeling Repairable Systems with Failure Rates that Depend on Age & Maintenance. *IEEE Transactions on Reliability,* **42(4)**:566-571.

Chen S., Cowan, C.F.N. & Grant, P.M. (1991). Orthogonal Least squares Learning Algorithm for Radial Basis Function Networks. *IEEE Transactions on Neural Networks.* **2(2)**:302-309.

Coetzee, J.L. (1997). *Maintenance.* Hatfield, RSA: Maintenance Publishers

Cox, D.R. (1972). Regression Models and Life Tables. *Journal of the Royal Statistical Society,* **B34**:187-220

Crowder, M.J., Kimber, A.C., Smith, R.L. & Sweeting, T.J. (1991). *Statistical Analysis of Reliability Data*. London: Chapman & Hall

Dale, C.J. (1985). Application of the Proportional Hazards Model in the Reliability Field. *Reliability Engineering,* **10**:1-14

Demuth, H. & Beale, M. (1998). Neural Network Toolbox for use with Matlab: User's Guide Fifth Printing Version, The Math Works Inc.

Grossberg, S.(1972). Neural Expectation: Cerebellar and Retinal Analogs of Cells Fired by Learnable or Unlearned Pattern Classes. *Kybernetik,* **10**:49-57

Guo, R., Love, C.E. & Bradley E.A. (1994). Is Bad-As-Old A Good Model for Complex Systems with Imperfect Repaired Subsystems? Included in Bradley, E.A. *Reliability Engineering Volume 10: Course Notes Rev. 04*. Pretoria: University of Pretoria

Hagan, M.T. and Menhaj, M.B. (1994). Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks,* **5(6)**:989-993

Hagan, M.T., Demuth, H.B. and Beale, M. (1996). *Neural Network Design*. Boston, MA: PWS Publishing Company

Hebb, D.O. (1949). *The Organization of Behaviour.* New York: John Wiley

Hertz, J., Krogh, A. & Palmer, R.G. (1991). *Introduction to the Theory of Neural Computation.* Redwood City, CA: Addison-Wesley Publishing Company

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA* **79**:2554-2558

Jacobs, R.A. (1988). Increased Rates of Convergence through Learning Rate Adaptation. *Neural Networks,* **1**:295-307

Jantunen, E. (2003). Prognosis of Wear Progress based on regression analysis of condition monitoring parameters. *Tribologia,* **22(4)**:3-15

Jardine, A.K.S. (1973). *Maintenance, Replacement and Reliability*. Toronto, Canada: Pitman Publishing

Kaminski, W. & Strumillo, P. (1997). Kernel Orthonormalization in Radial Basis Function Neural Networks. *Transactions on Neural Networks,* **8(5)**:1177-1183

Kohonen, T. (1972). Correlation matrix memories. *IEEE Transactions on Computers,* **21**:353-359.

Kumar, D. & Klefsjö, B. (1994) Proportional Hazards Model: a Review. *Reliability Engineering and System Safety,* **44**:177-188.

Lee J. & Kramer B.M. (1993). Analysis of machine degradation using a neural network based pattern discrimination model. *J. Manuf. Systems,* **12**:379-387.

Leung, F.K.N. & Cheng, A.L.M. (2000). Determining replacement policies for bus engines. *Int. Journal of Quality & Reliability Management,* **17(7)**:771-783

Levenberg, K. (1944). A Method for the Solution of Certain Non-linear Problems in Least Squares. *Quarterly Journal of Applied Mathematics* **II(2)**:164-168

Liang, F., Xu, M. & Shun. Q. (2000). Competitive Supervised Learning Algorithms in Machine Condition Monitoring. *International Journal of Comadem.* **3(1)**:39-46

Liu, M.C., Sastri, T. & Kuo, W. (1995). An exploratory study of a neural network approach for reliability data analysis. *Qual. Reliabil. Eng. Int.* **11:**107-112

Luxhøj, J.T. & Shyur, H-J. (1995). Reliability curve fitting for aging helicopter components. *Reliability Engineering and System Safety,* **48**:229-234

Luxhøj, J.T. (1999). Trending of equipment inoperability for commercial aircraft. *Reliability Engineering and System Safety,* **64**:365-381

MacKay, D.J.C. (1992). Bayesian Interpolation. *Neural Computation,* **4**:415-447

MacKay, D.J.C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation,* **4**:448-472

Malik, M.A.K. (1979). Reliable Preventive Maintenance Scheduling. *AIIE Transaction,* **11**:221-228

Marquardt, D.W. (1963). An Algorithm for Least-Squares Estimation of Non-Linear Parameters. *Journal of the Society of Industrial and Applied Mathematics,* **11(2)**:431-441

Martorell, S., Munoz, A. & Serradell, V.S. (1996). Age-dependant Models for Evaluating Risks and Costs of Surveillance and Maintenance of Components. *IEEE Transactions on Reliability,* **45(3)**:433-442

McCulloch, W.S. & Pitts W. (1943). A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics,* **5**:115-133. Reprinted in Anderson and Rosenfeld (1988)

McKeown, J. J., Stella, F. & Hall, G. (1997). Some Numerical Aspects of the Training Problem for Feed-Forward Neural Nets. *Neural Networks,* **10(8)**:1455-1463

McLoone, S. & George Irwin, G. (2001). Improving Neural Network training solutions using regularization. *Neurocomputing,* **37**:71-90

Meeker, W.Q. & Escobar, L.A. (1998). *Statistical Methods for Reliability Data.* John Wiley & Sons. Inc.

Minsky, M.L. & Papert, S.A. (1969). *Perceptrons.* Cambridge, MA: MIT Press.

Moody, J. & Darken C.J. (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation,* **1(2)**:281-294

Negnevitsky, M. (2002). *Artificial Intelligence.* Harlow, England: Addison-Wesley/Pearson Education

Nowlan, F.S. & Heap, H.F. (1978). *Reliability Centred Maintenance.* National Technical Information Service, US Department of Commerce

Parker, D.B. (1985). *Learning Logic.* Cambridge, MA: MIT Center for Research in Computational Economics and Management Science

Pijnenburg, M. (1991). Additive Hazards Models in Repairable Systems Reliability. *Reliability Engineering and System Safety,* **31**:369-390

Pike, M.C. (1966). A Method of Analysis of a Certain Class of Experiments in Carcinogenesis. *Biometrics,* **22**:142-161

Pintelon, L., Gelders, L. & Van Puyvelde, F. (1997). *Maintenance Management.* Leuven, Belgium: Uitgewerij Acco

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks,* **11**:761-767

Prentice, R.L., Williams, B.J. & Peterson A.V. (1981). On the Regression Analysis of Multivariate Failure Time Data. *Biometrika,* **68(2)**:373-379

Reinertsen, R. (1996). Residual life of technical systems; diagnosis, prediction and life extension. *Reliability Engineering and System Safety,* **54**:23-34

Robitaille, B., Marcos, B. ,Veillette M. & Payre, G. (1996). Modified Quasi-Newton Methods for Training Neural Networks. *Computers chem. Engng,* **20(9)**:1133-1140

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review,* **65**:386-408

Rosenblatt, F. (1960). Perceptron simulation experiments. *Proceedings of the Institute of Radio Engineers,* **48**:301-309

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.*Washington DC: Spartan

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning Internal Representations by Error Backpropagation. In: Rumelhart, D.E., McClelland, J.L. and

the PDP research group (Eds.). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations, pp. 318-362. Cambridge, MA: MIT Press. Reprinted in Anderson and Rosenfeld (1988)

Schenker, B. & Agarwal, M. (1996). Cross-validated structure selection for neural networks. *Computers chem. Engng,* **20(2)**:175-186.

Schittenkopf, C., Deco, G. and Brauer, W. (1997). Two Strategies to Avoid Overfitting in Feedforward Networks. *Neural Networks,* **10(3)**:505-516.

Shigley, J.E. (1986). *Mechanical Engineering Design, First Metric Edition*. Singapore: McGraw Hill Book Company

Shin, I., Lim, T.J. & Li C.H. (1996). Estimating Parameters of Intensity Function and Maintenance Effect for Repairable Unit. *Reliability Engineering and System Safety,* **54**:1-10

Van Der Smagt, P.P. (1994). Minimisation Methods for Training Feedforward Neural Networks. *Neural Networks,* **7(1)**:1-11

Van Tonder, F. (2004). *Experimental Strain Measurement using Strain Gauges.* (Lecturer Notes) Pretoria: Department of Mechanical and Aeronautical Engineering, University of Pretoria

Velten, K., Reinicke, R. & Friedrich, K. (2000). Wear volume prediction with artificial neural networks. *Tribology International,* **33**:731-736

Vlok, PJ. (1999). *Vibration Covariate Regression Analysis of Failure Time Data with the Proportional Hazards Model.* Masters dissertation. Pretoria: University of Pretoria.

Vlok, PJ. (2001). *Dynamic Residual Life Estimation of Industrial Equipment Based on Failure Intensity Proportions.* PhD dissertation. Pretoria: University of Pretoria,

Wang, W. & Zhang, W. (2005). A model to predict the residual life of aircraft engines based upon oil analysis data. *Naval Research Logistics,* **52(3)**:276-284

Weibull, W.A. (1951). A Statistical Distribution Function of Wide Applicability. *Journal of Applied Mechanics,* **18**:293-297

Werbos, P.J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in Behavioural Sciences.* Ph.D. thesis. Boston, MA.: Harvard University.

Whitehead, B.A. & Choate, T.D. (1996). Cooperative-Competitive Genetic Evolution of Radial Basis Function Centres and Widths for Time Series Prediction. *IEEE Transactions on Neural Networks.* **7(4)**:869-880

Widrow, B. & Hoff M.E. (1960). Adaptive Switching Circuits. *IRE WESCON Convention Record,* **4**:96-104. Reprinted in Anderson and Rosenfeld (1988)

Xu, K., Xie, M., Tang, L.C. & Ho S.L. (2003). Application of neural networks in forecasting engine systems reliability. *Applied Soft Computing,* **2**:205-268