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

M.A. Herzog[a], T. Marwala[b] & P.S. Heyns[a]

[a]Dynamic Systems Group, University of Pretoria, Pretoria, Republic of South Africa

[b]Control and Systems Group, University of Witwatersrand, Johannesburg, Republic of South Africa

This paper concerns the use of neural networks for predicting the residual life of machines and components. In addition, the advantage of using condition-monitoring data to enhance the predictive capability of these neural networks was also investigated. A number of neural network variations were trained and tested with the data of two different reliability-related datasets. The first dataset represents the renewal case where the failed unit is repaired and restored to a good-as-new condition. Data was collected in the laboratory by subjecting a series of similar test pieces to fatigue loading with a hydraulic actuator. The average prediction error of the various neural networks being compared varied from 431 to 841 seconds on this dataset, where test pieces had a characteristic life of 8,971 seconds. The second dataset was collected from a group of pumps used to circulate a water and magnetite solution within a plant. The data therefore originated from a repaired system affected by reliability degradation. 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 renewal dataset. The small number of inputs and poorly mapped input space on the second dataset meant that much larger errors were recorded on some of the test data. The potential for using neural networks for residual life prediction and the advantage of incorporating condition-based data into the model was nevertheless proven for both examples.

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

1 Introduction

The advent of preventive maintenance has increased the need for reliable information, leading to the development of data analysis techniques for the purpose of estimating residual life. The traditional approach, described by Coetzee [1], involved the use of probabilistic models which were only a function of failure time, but more recently, researchers such as Pijnenburg [2] have investigated the use of regression models which allow explanatory variables to be incorporated. Condition-based data is commonly available and Vlok [3,4] found that its use enhanced the accuracy of the predictions made by regression models. Accurate residual life estimates have a number of benefits for tactical maintenance planning, apart from the selection of an optimal replacement strategy and the flexibility offered to the maintenance manager. Such information allows the advanced planning of shut-downs, resource allocation and the optimal holding of spares.

A different approach is required for renewal and repaired systems, which are not returned to a good-as-new condition after failure. In the renewal case, it is assumed that, once repaired, the system is returned to its original state. If a system is not repaired to its original condition, this assumption does not hold and system deterioration due to imperfect repair has to be taken into account. Reinertsen [5] states that a considerable number of papers have explored the estimation of residual life for renewal systems through the use of statistical methods, but no corresponding work has been done on repaired systems that do not conform to the assumption that they have been returned to their original state. Pijnenburg [2] comments on the extreme rarity, in the literature he reviewed, of datasets on repaired systems, in which failure times are listed in the original chronological order. Ascher and Feingold [6] could find only four such datasets. Pham and Wang [7] commented in their 1996 review that most research on imperfect repair was for single-unit systems. Despite noting a shift towards maintenance policies and models for multi-component systems, the survey of Wang [8] published in 2002 again focused mostly on single-unit systems. Langseth and Lindqvist [9] worked with a dataset of a compressor unit subject to imperfect repair. It was found that the items which caused the failure were easy to identify, but the effect of repair on the system could not always be established from the recorded information.

The research studies using condition-monitoring data for residual life estimation include the work of Jantunen [10] who fitted a polynomial curve to vibration data, Vlok [3,4] who used regression curves and vibration data to estimate the residual life of pumps, and Wang and Zang [11] who used spectrographic oil analysis data to predict the residual life of aircraft engines. Vlok [3,4] found that regression models offered a significant advantage over parametric models because of their ability to take into account the information relating to the failure of a system. Condition-monitoring data could therefore be used to improve the accuracy of the estimates made with these models.

Research has been done to investigate the use of neural networks in applications related to maintenance and reliability. A wide variety of methods, network architectures and data combinations were used in these cases. Amjady and Ehsan [12] evaluated the reliability of power systems using an expert system based on neural networks. Luxhøj and Shyur [13] compared the performance of traditional reliability modeling techniques with neural networks for the fitting of a reliability curve to the data of helicopter components. Luxhøj [14] researched the prospect of providing Federal Aviation Administration (FAA) safety inspectors with a means to evaluate and control the appropriate surveillance levels for aircraft operators through the use, among other things, of neural networks. Liang, Xu and Shun [15] applied multi-layer perceptron (MLP) neural networks to the field of condition monitoring, whereas Xu et al. [16] attempted to forecast reliability by using neural network techniques to analyze the data on past historical failures. Neural networks have therefore been employed for maintenance-related applications, but their use has not yet been fully explored in the context of residual life prediction. As these networks have the capacity to learn about the underlying relationship between various inputs and outputs, they are ideally suited to making predictions about complex systems.

This research builds upon the work done by others who employed regression models for predicting failure. As an alternative to traditional statistical methods, this study investigates the suitability of neural networks for making reliability predictions in the cases of both renewal and repair. The incorporation of covariates containing historical information and condition data into the training process is explored with the aim of improving the accuracy of the predictions that can be made. The performance of different neural network types when trained with reliability data is also of interest, and the results achieved by a selected group of networks are compared. Based on these results, conclusions can be drawn on the suitability of using neural networks in conjunction with condition-monitoring data for reliability predictions as part of the tactical planning done by the maintenance practitioner.

2 Problem description

2.1 Renewal dataset

The first dataset represents the renewal case where the system is returned to a good-as-new condition by replacing the failed component. A series of laboratory tests were conducted, using a 630kN Schenck Hydropuls hydraulic actuator (see Figure 1), which simulated an actual maintenance situation encountered in industry. A number of similar notched test pieces were manufactured with the same cross-sectional shape as a component which serves as an overload protection in jaw crushers. This toggle plate (Figure 2) is designed to fail when foreign objects become wedged between the crusher jaws, thereby preventing damage to the machine. The test pieces were placed under a cyclic loading in the hydraulic test rig until they failed as a result of fatigue. The cyclic loading was applied according to a sinusoidal pattern, where the mean and amplitude were varied by means of the actuator's control system, in this way generating different operating conditions for the series of test runs and producing a varied dataset. Though the actuator is capable of exerting a maximum force of 630 kN, the actual applied load pattern was selected to ensure that the components had a finite fatigue life.

The actuator of the test rig was set to maintain constant amplitude in the oscillation of its jaws for the duration of each test run. The amplitude was varied for the different test runs, thus 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 the amplitude was reduced and this could be observed in the corresponding drop in the magnitude of the load cell measurements that were taken. The reduction in applied force resulting from the use of displacement control provides a measurable indication of deterioration in the condition of the test piece.

The loading pattern was applied at a frequency of 3Hz which was close to the upper limit of what could be achieved while still allowing the actuator to apply a suitably high load. Measurements were recorded over periods of three seconds at three-minute intervals. The testing proved that the time interval between the taking of measurements and the duration of the recording window were both satisfactory. The data for a total of nine complete actuator cycles was captured in each measurement window, in which a sequence of 1,800 samples was taken during the three-second period.

Four different sensors were selected and used for taking the measurements during each such measurement window. The choice of sensors was not only aimed at tracking the deterioration in the test piece deterioration, but also at providing a measure of the operating conditions that influenced the life of the test piece.

Measurements were taken with the load cell forming part of the test equipment, as well as with a strain gauge attached to the test piece. These sensors provided information on the nature of the applied load. An accelerometer was mounted on the opposite side of the location of the notch. The purpose of this measurement was to measure the movement due to the deflection of the test piece under loading.

The temperature on the surface of the test piece was measured by means of a thermocouple mounted on the side of the test piece. For convenience, the thermocouple was positioned halfway along the cross-section and aligned with the center of the notch. It was found that the temperature measured at this position rose dramatically once crack propagation started. The temperature measurement was therefore found to be a useful indicator of test piece condition and gave a good indication of imminent failure. The magnitude of the rise in temperature compared with the initial measured temperatures was dependent on the applied loads and therefore also served as an indicator of the rapidity with which failure was occurring.

2.2 Repaired system dataset

As an example of a repaired system, a dataset was used which had been obtained by Vlok [3,4] from the Sasol Twistdraai mine plant. Measurements were taken on eight identical Warman pumps used to circulate a water and magnetite solution within the plant. Four main failure modes were identified for these particular pumps, namely bearing seizure, broken or defective impellers, damaged or severely eroded pump housings, and broken drive shafts. The measurements taken on the pumps were solely vibration readings, for which a spectral analysis was performed and a number of fault frequency bands were monitored. The frequency bands 0.4×RPM, 1×RPM, 2×RPM, and 5×RPM were monitored for both the bearings of these pumps. Condition-based suspensions were made on the basis of these measurements, but the reasons for intervention and the cause of failure were not recorded. Measurements were also only taken sporadically and the dataset is therefore sparse.

During the 791-day window from the initial installation of the eight pumps, pump operation was suspended eight times due to condition-based warnings, and 11 failures were recorded. The surprisingly high percentage of failures might be attributed to the inconsistent application of the condition-based policy and the long measurement intervals. For the purpose of life prediction, the suspensions due to condition-based intervention were classified as failures. Although the data was collected from the start of each pump's life, the vibration measurements were taken extremely infrequently. Vlok [3,4] does note that some of the failures occurred suddenly, with deterioration occurring in a matter of hours. Obviously, it would be difficult to predict such a sudden deterioration with the information that was available. From the random nature of the 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.

Three of the eight pumps experienced only one failure, two of the pump units failed twice, and three 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 failures or preventive interventions, respectively. Reliability therefore deteriorated dramatically after the first failure, indicating imperfect repair. A further pattern was observed with regard to the time to the first failure, and this pattern allowed the pumps to be subdivided into two groups. Pumps which failed for 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.

3 Neural network application

The MATLAB neural network toolbox was used to build and train the neural networks for the purpose of residual life prediction. A number of network variations in terms of architecture and training algorithms are available in this programming environment.

3.1 Network testing

The usefulness of a neural network in 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. Schenker and Agarwal [17] identify the three most common methods for testing the relative performance of neural networks:

- A subdivision of the available data into a training and test set, 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 [17], is only meaningful when the data represents a true process. It can therefore be successfully applied in cases involving reliable physically based models. Schenker and Agarwal [17] identify the subdivision of the dataset into separate training and test sets as the approach that 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. Schenker and Agarwal [17], in their comparison of the performance of the different testing methods, point out 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, so 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 neural networks, but each group of data is used only once for testing. The recorded error values are added once the process has been completed, and this result is the basis for comparing the different neural networks.

The greatest advantage of using 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 in cases where the dataset is limited in size. Training does unfortunately become more cost-intensive due to the repetition required for cross-validation.

3.2 Neural network for renewal dataset

The first-order gradient descent learning algorithm serves here as the basis for comparing the different neural networks due to its historical significance. Adjustments were made to the learning rate, and a momentum term was introduced that increased the rate of convergence of this algorithm. The performance of the gradient descent algorithm was compared with the much faster second-order Levenberg-Marquardt algorithm which, according to the findings of Hagan and Menhaj [18], outperformed other fast techniques. Bayesian regularization (see Bishop [19]) 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 [14] in his research, has the advantage of rapid unsupervised training. It is also of interest because it is a network with radial basis function (RBF) architecture, in contrast to the MLP architecture of the networks mentioned so far.

A static split was chosen as the method for comparing network performance on the renewal dataset. This was feasible because of 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: nine of the datasets were used for training and the remaining three comprised the testing set.

Each network was constructed with five 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 five nodes in the hidden layer, so that the basic network structure was similar for each of these networks. The size of the hidden layer was optimized through an empirical process where the number of nodes in the hidden layer was varied.

The inputs used for network training were the elapsed time of the specific test at the time of the measurement, initial average load, initial load range, change in load range, and change in temperature. (Table 1) The network inputs and outputs were normalized and transformed into values between zero and one.

Elapsed time gives the network an indication of the component's age and allows the network to differentiate between new samples, and samples that already show fatigue. Therefore the network can differentiate between two samples which are subjected to the same loading but do not yet exhibit measurable signs of deterioration.

The network is given a longer-term predictive capability by providing it with information about the operating conditions to which the test piece is subjected. The initial load average and range 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 factor contributing to the rapidity with which failure occurs. The initial load conditions can be used in this case because the loading is kept constant for the duration of each test run. In cases where the load varies, an input reflecting aggregated load would be required instead.

The changes from initial load and temperature give an indication of deterioration in the condition of the test piece and impending failure. Due to the setting of the machine, displacement remained constant and therefore the load dropped when cracking started. Temperature increased substantially as fatigue damage worsened and the crack propagated through the test piece. Therefore the network can make adjustments 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.

3.3 Neural network for repaired system

As the sparseness of the pump dataset (see Table 2) did not allow for the use of a separate test set, it was decided that cross-validation should be used to test the performance of different network designs. To this end, the dataset was divided into eight groups, each representing the data from one of the pumps. In their work, Schenker and Agarwal [17] assert that individual runs should not be split when using cross-validation, as this 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.

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

The actual data was pre-processed in a similar way to the renewal dataset. It was found that the high values measured at an advanced stage of deterioration led to a distortion in the normalized data inputs used to train the neural networks. The neural networks became insensitive to the small changes occurring in the initial stages of deterioration. As the aim of this work is not to prove the usefulness 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.

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 was done by using the elapsed time since installation, the elapsed time since the last failure, and a covariate that can be described as a risk variable dependent on the history of the pump. The risk variable served to compensate for the lack of historic information on pump loading and failure severity. Two further training runs were completed, first with one and then with two additional inputs, each of which represented the average value of the vibration response amplitude in a chosen frequency band for the measurements on the two bearings. Using the findings of Vlok [3,4] as basis, additional inputs based on condition related measurements should improve the accuracy of failure predictions.

The dataset originates from a repaired system and its reliability is therefore affected by previous failures and repair. The influence of these factors has to be taken into account, even though not much of this information was recorded. Vlok [3,4] 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 1 is used to calculate the value of R.

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

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, whereas 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 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. Then a second training run was done with a hidden layer that had one node more than the input layer. Due to ill-conditioning, however, MLP networks with six hidden nodes could not be trained with five inputs. The dataset size used for cross-validation contained 53 data points. Once this had been 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 an MLP network with five inputs was therefore limited to five, in order to prevent ill-conditioning as discussed by McKeown et al. [20], because the number of variables in the network exceeded the number of inputs. The networks all generated a single output, namely a prediction of the remaining life until the next failure of the pump.

4 Neural network results

4.1 Neural network results for renewal data

The traditional way of conducting a data analysis on the reliability data originating from a renewal system is to fit a statistical distribution to such data. This technique, described by Coetzee [1], was accordingly chosen to form the basis of comparison to illustrate the advantage of using neural networks.

A Weibull distribution was fitted to the data of the training set and the parameters of the two-parameter Weibull distribution were found to be $\beta = 1.7522$ and $\eta = 8971$. The Weibull parameter η is the scale parameter, which is also referred to as the characteristic life. Coetzee [1] notes that 63.2% of components fail before this time and 36.8% survive. The use of a statistical distribution means that no specific prediction can be made about an individual test piece. The estimated life is therefore taken as the characteristic life of the whole population of the training set. The actual residual life for the test sets differed by between 11.2% and 55.4% from the characteristic life of 8,971 seconds, that was calculated using this statistical method. The results achieved by fitting the Weibull distribution show the disadvantages of this method when comparing them with the residual life results obtained by using neural networks, which are discussed in the following paragraphs. Table 3 shows accuracy of the life predictions on the test set, using the different methods with the data available at the start of the various test runs.

The standard back-propagation algorithm was used to train the same network architecture with nine different combinations of the learning rate (*α*), and momentum parameter (*β*). Figure 3 illustrates the rate of convergence of the gradient descent back-propagation 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. If the learning rate is 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 of stability.

It was found that a learning rate of 0.75 and a momentum constant of 0.9 gave good results, so these constants were used for the comparison with other network types and training algorithms. The training algorithm was stopped early and could not accommodate some of the more isolated data points in the training set. It was therefore possible to maintain improved properties of generalization.

Training with the Levenberg-Marquardt algorithm proved much more rapid and a far better fit was achieved after less than 300 training epochs. The neural network's estimated residual life for the training data was within 5% of the actual remaining life of each component, when presented with the first recorded inputs after the start of the test run. The largest error was 449 seconds, which compares very favorably 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. The accuracy of the prediction obtained by the neural network on data that had not previously been seen during training, was similar in two of the cases. The largest error on the test set was 20%, which indicates some degree of overfit, as the data from this particular test piece was most isolated in input space when compared against the training data.

It was expected that the use of Bayesian regularization would address the problem with overfitting encountered with the network trained with a standard Levenberg-Marquardt algorithm. The neural network that was trained with Bayesian regularization did not produce as close a fit for some parts of the training set as the fit achieved with the standard Levenberg-Marquardt algorithm. In particular, the estimates generated for some of the isolated data points on the training set displayed a large error. This was expected, as the regularization technique penalized training in order to maintain the network's capability of providing acceptable results for new data. The two test pieces in question had a significantly shorter life than the other test pieces and were therefore isolated from the rest of the data. The benefit of this regularization technique regarding improved generalization becomes clear when examining the results obtained for the test set. (see Figure 4) 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 10,102 seconds. The results achieved for all three of the test pieces in the test set were therefore very satisfactory and indicated a good generalization.

Setting up the GRNN is an almost instantaneous process. The input vectors are used as the weights in the hidden layer and the target vectors as the weights in the output layer. No supervised training is therefore required in its construction. Network performance can be influenced only by changing the value of the bias of the radial basis function nodes in the hidden layer.

The bias of a radial basis function in MATLAB is set by defining a parameter, called the spread value. Every bias in the first layer of the network is set to 0.8326 divided by this spread value. 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 where the spread alters the radius of the basis functions, and therefore determines the amount of overlap and consequently the smoothness of the fit.

When designing an 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 a 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 accuracy will be forfeited. A number of different spread values were tested, and the results are tabulated in Table 4.

Figure 5 illustrates that the larger the spread chosen for the network, the smoother the fit. The quality of the fit on the training set is reduced, as a number of hidden layer neurons start to influence the output for any given input. But an increase in spread improves network generalization until an optimal balance is reached. Any further increase in spread is detrimental to network performance.

The estimated residual life for the training data when using a small spread value was closer to the actual life than was achieved with any of the other networks, when 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.

When comparing the performance of the different networks (see Table 5), it was found that the MLP network trained with a Levenberg-Marquardt algorithm using Bayesian regularizations had a clear advantage over the other models. The gradient descent algorithm was found to be significantly slower than the Levenberg-Marquardt algorithm. 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 instantaneous.

Table 6 shows the average prediction error whereas Table 7 gives the maximum prediction error of the networks being compared. Though the GRNN has a lower average error than the other networks, it has the highest maximum error. This may explain why the MLP network trained with the Levenberg-Marquardt algorithm with Bayesian regularization outperformed it in terms of the mean squared error. The early stopping of the gradient descent back-propagation 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. As the training algorithm was stopped before it could accommodate this data, the network performed well on the test data.

When considering these results, it should be borne in mind that the measurements were taken at intervals of 180 seconds, and that the time of first measurement after failure was used as failure time for training the neural networks. The actual failure took place within the band spanning the last measurement cycle. All the neural networks performed very well on the data in the test set that was closest to the training data. The data for the test piece, showing the greatest variation from anything the network had seen before, 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 (Figure 6) of the results relating to this series of data.

The results for the GRNN are not as smooth as those obtained with MLP networks. The jagged shape of the prediction graphs illustrates the “local” nature of RBF networks, compared with the “global” nature of MLP networks. This property may adversely affect network performance, if the information for one set of the data points used for training, are corrupt. A greater overlap of the RBF nodes would counteract this situation by smoothing the transition between kernels.

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. In particular, the accuracy of the predictions made by the MLP trained with the Levenberg-Marquardt algorithm with Bayesian regularization would be suitable for making maintenance decisions in the context of the simulated situation.

4.2 Results for the Repaired System

The neural networks trained with the Levenberg-Marquardt algorithm used nodes with the log-sigmoid transfer function in both the hidden and output layers. 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 the test data. A series of training runs with a range of different training targets were consequently performed in order to improve generalization by terminating the training process at an earlier stage.

The training targets used for this purpose were the values 1×10^{-5}, 5×10^{-3}, 1×10^{-2}, 2.5×10^{-2} and 5×10^{-2}.

Table 8 shows that the best results obtained with the smallest error on the test data, were 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 characterizing the worst results (shown in Table 9) could occur. The target values refer to the normalized output values, whereas the sum-of-squares error is calculated from an error value in days.

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

Changing the size of the hidden layer and the number of inputs was affected by the early stopping of the training process, so that no clear pattern emerged. Though 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 did appear that a greater number of inputs generally improved the performance of these networks, but the results were not conclusive.

The MLP neural networks trained with the Levenberg-Marquardt algorithm with Bayesian regularization (LMBR) yielded similar results to the networks 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. Figure 7 gives a comparison of actual and predicted failure times.

The log-sigmoid transfer function was used for the nodes in the hidden layer of these networks, whereas two different transfer functions were utilized in the output layer. Table 10 summarizes the results achieved with the neural networks trained with the Levenberg-Marquardt algorithm with Bayesian regularization.

The results shown in Table 10 indicate that the choice of transfer function of the output node had a far greater influence on the performance of the network than the variation in the number of nodes in the hidden layer. Another observation is that in this case, additional input information clearly leads to more accurate predictions.

The GRNN networks were trained with RBF neurons with different sensitivities so as to select an optimal value for this parameter. Table 11 lists 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 but an increase in input space to five inputs led to overfitting. The consequence of an increase in the number of inputs into such a network is that the outputs of the nodes are influenced by a greater number of variables, hence making them more sensitive to a particular combination of input values. Once this sensitivity becomes too great, the network starts to lose its ability to generalize. For this reason, the general regression neural networks did not respond well when presented with five inputs.

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

The ease of implementation of the GRNN was again illustrated. The construction of this type of network is instantaneous, as no weight adjustments are made by implementing a back-propagation algorithm. By varying the sensitivity of the RBF neurons, adjustments can be made to optimize the network's ability to generalize. An optimal network can therefore be rapidly found by employing cross-validation.

In summary, when testing network generalization by means of cross-validation, the best results obtained with the various neural networks were very similar, once these networks had been optimized in respect of this particular dataset. Table 12 gives a comparison of the best results achieved with each network type.

The comparison of the different networks by cross-validation was based solely on the relative size of the sum-of-squares error obtained on the test data. If the suitability of the applied method should be judged, the results should also be viewed in the context of the practical application. It was found that number of very large prediction errors were made by the networks on isolated points, which far exceeded the actual remaining time to failure of a specific pump. When the ten worst predictions were excluded, the average prediction error of the networks was 39.8% for the LMBR network, 33.2% for the GRNN and 41.7% for the LM network.

The nature of this result indicates that the networks were able to model some but not all of the significant properties of these complex pump systems. When seen in the context of the intended application, the results represent a positive point of departure. An average prediction error of 40% is too large and does not allow these networks to be used in their current form for making decisions about maintenance. It can therefore be said that the complexity of the problem requires a larger and more descriptive dataset for training the 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 which will allow the network to distinguish among 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 differ significantly from the data with which the network was trained. The network is therefore unable to deal with some of the data correctly, and produces a spurious result. The dataset used by Vlok [3,4] is not ideal for this purpose owing to its sparseness, and it is unlikely that more can be achieved regarding failure prediction with 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.

One should bear in mind the limitations imposed on residual life predictions by the unpredictability of operations in an actual plant. The covariates chosen as inputs into a neural network have to 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 in this study can therefore be seen as a conditional success in terms of the use of neural networks for this application.

5 Conclusion

The use of neural networks for making failure predictions for both renewal and repaired cases was investigated. The estimates that the networks made regarding the simulated renewal system proved highly accurate, with the average error varying between 431 seconds and 841 seconds for the different types of neural networks. 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 with the use of the common probabilistic technique that involves fitting a Weibull distribution to the failure-time data. The performance of the neural networks was compared with this statistical method on the basis of the predictions 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%. By comparison, the error of the residual life estimates made using the Weibull distribution, ranged between 11.2% and 55.4%.

The failure predictions for the repaired systems were hampered by the combination of the system's complexity and the sparseness of the dataset, however. 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, posing a severe challenge to the analyst. The small number of inputs and poorly mapped input space meant that the explanatory information proved insufficient for the network to model the system accurately, and large errors were recorded on some of the test data.

With respect to the comparison between different neural network methods, the use of Bayesian regularization proved very effective in the prevention of overfitting. The use of a second-order method, such as the Levenberg-Marquardt training algorithm, produced a significant reduction in training time in comparison to the gradient descent method. It was found that the optimization of network parameters was an important part of the training process and that the performance of different network types was very closely matched once their design had been adjusted to suit a specific application. GRNN are simple, easily generated neural networks and proved a close match with the MLP networks, giving 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 achieved for the two datasets indicate that neural networks should become a useful tool for the analysis of reliability data in future. Clearly the approach outlined in this paper is not suitable for every application in the maintenance field, but the results indicate the potential that neural networks have as a powerful tool for the analysis of reliability data and the prediction of residual life.

References

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

[2] Pijnenburg, M. (1991). Additive hazards models in repairable systems reliability. *Reliability Engineering and System Safety,* **31**:369-390

[3] Vlok, PJ. (1999). *Vibration covariate regression analysis of failure time data with the Proportional Hazards Model.* Master's dissertation. Pretoria: University of Pretoria.

[4] Vlok, PJ. (2001). *Dynamic residual life estimation of industrial equipment based on failure intensity proportions.* PhD dissertation. Pretoria: University of Pretoria,

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

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

[7] Pham, H. & Wang, H. (1996). Imperfect maintenance. *European Journal of Operational Research,* **94**:425-438

[8] Wang, H. (2002). A survey of maintenance policies of deteriorating systems. *European Journal of Operational Research,* **139**:469-489

[9] Langseth, H. & Lindqvist, B.H. (2006). Competing risks for repairable systems: A data study. *Journal of Statistical Planning and Inference,* **136**:1687-1700

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

[11] 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

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

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

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

[15] Liang, F., Xu, M. & Shun. Q. (2000). Competitive supervised learning algorithms in machine condition monitoring. *International Journal of Comadem.* **3(1)**:39-46

[16] 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.

[17] Schenker, B. & Agarwal, M. (1996). Cross-validated structure selection for neural networks. *Computers Chem. Engineering,* **20(2)**:175-186.

[18] Hagan, M.T. and Menhaj, M.B. (1994). Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks,* **5(6)**:989-993

[19] Bishop, C. M. (1995). *Neural networks for pattern recognition.* Oxford, UK: Oxford University Press.

[20] 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

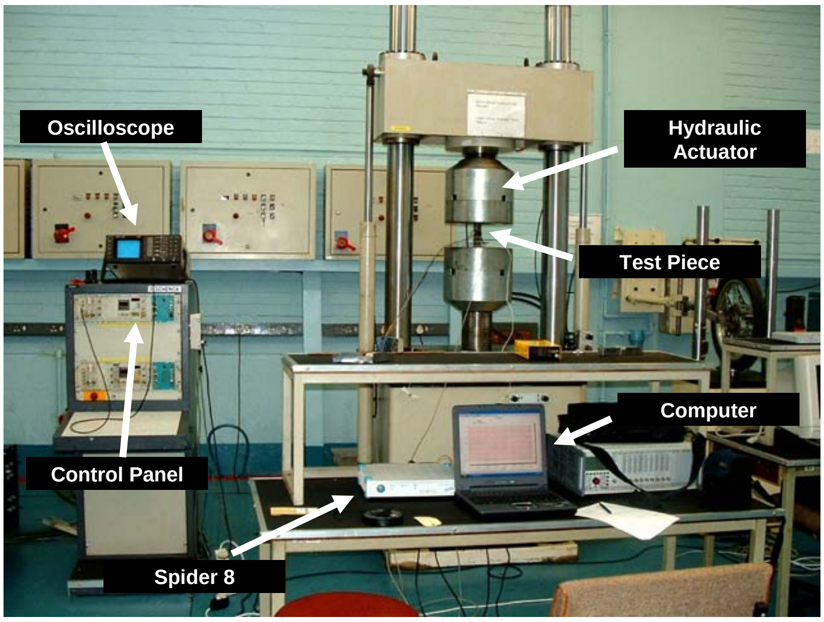

Figure 1: The equipment used for performing the laboratory work.

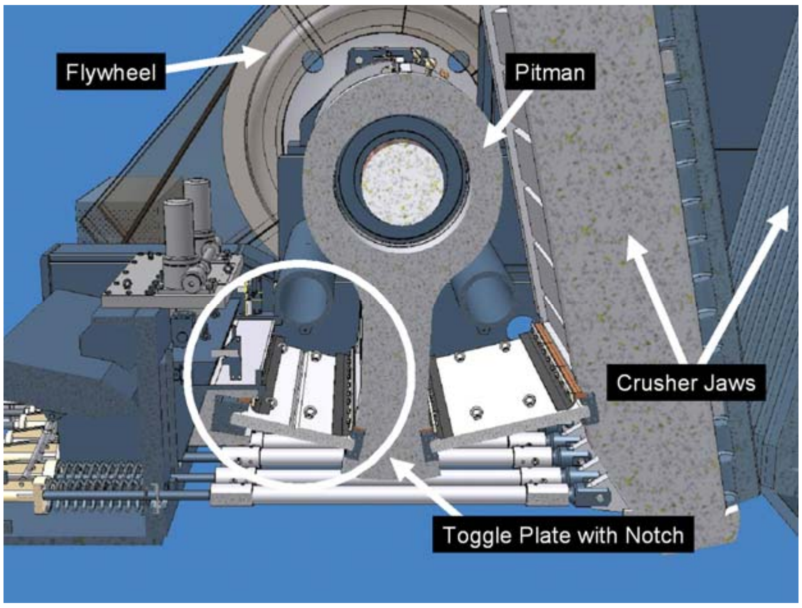

Figure 2: Crusher layout showing the position of the notched toggle plate.

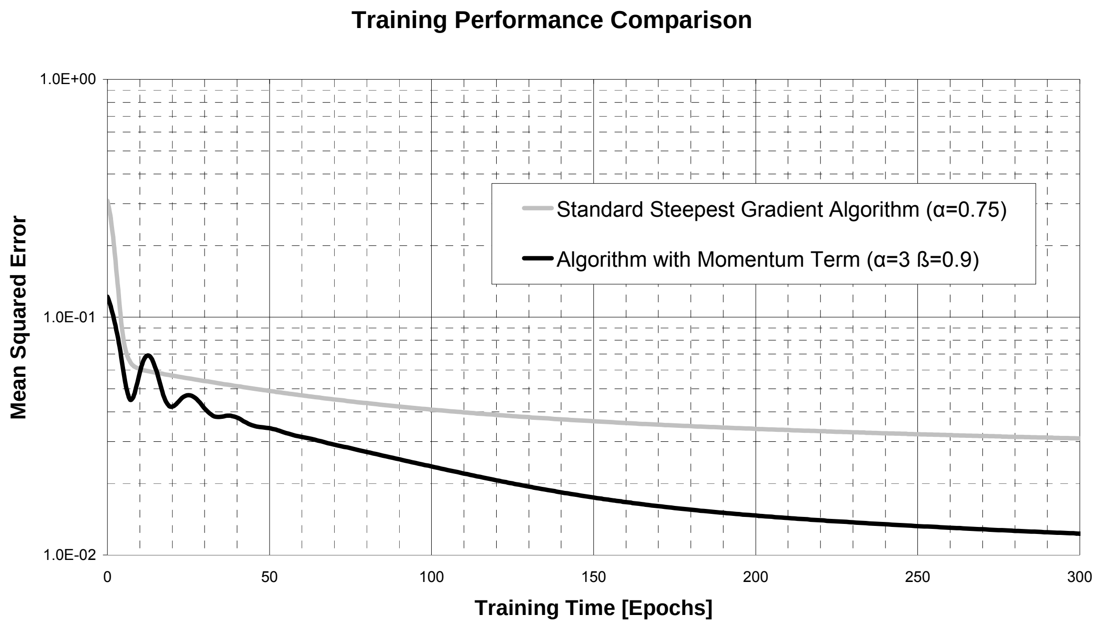

Figure 3: Comparison of the rate of convergence of the gradient descent back-propagation method, using a different combination of training parameters.

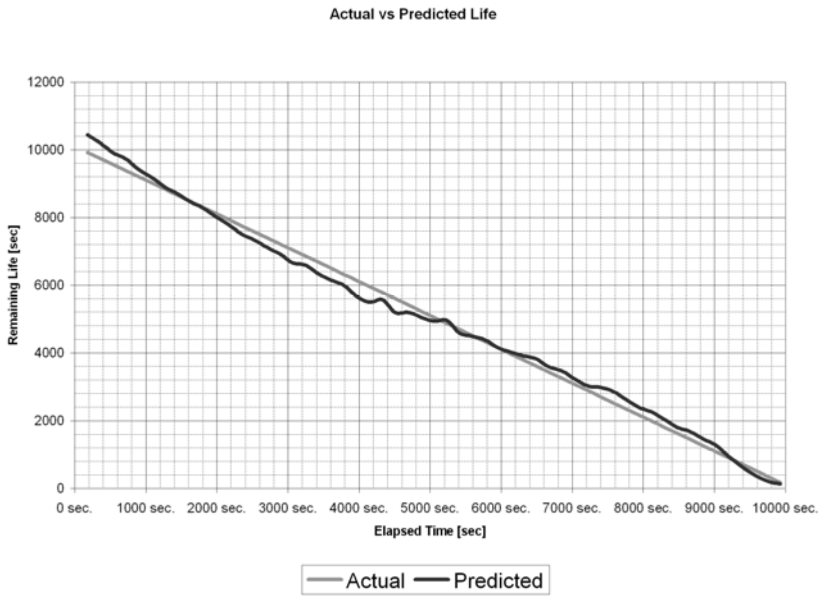

Figure 4: Comparison of actual remaining life and the prediction of the neural network trained with the Levenberg-Marquardt algorithm with Bayesian regularization.

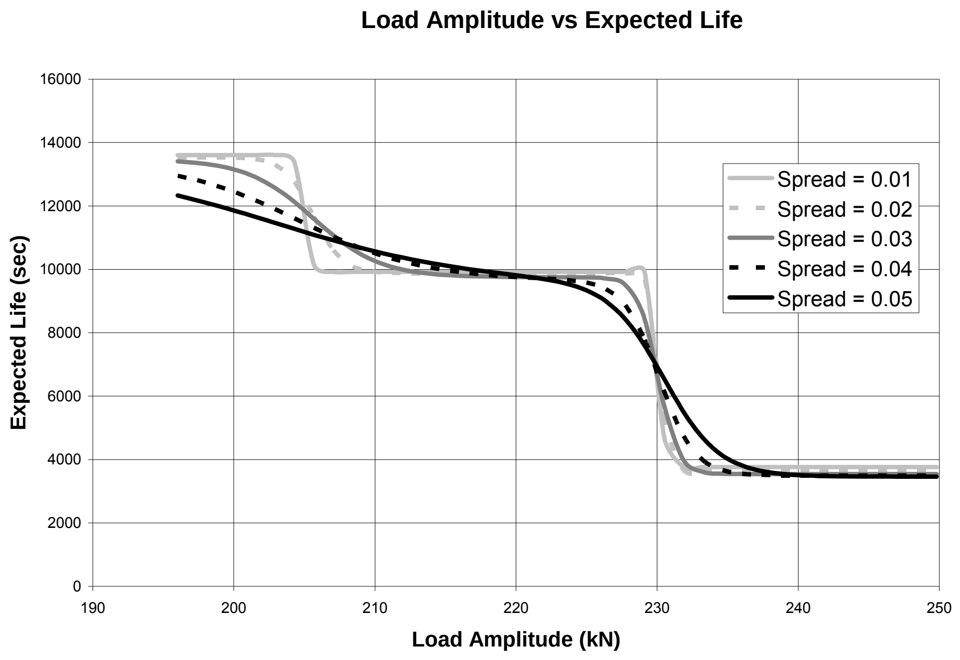

Figure 5: GRNN response when varying the load amplitude input.

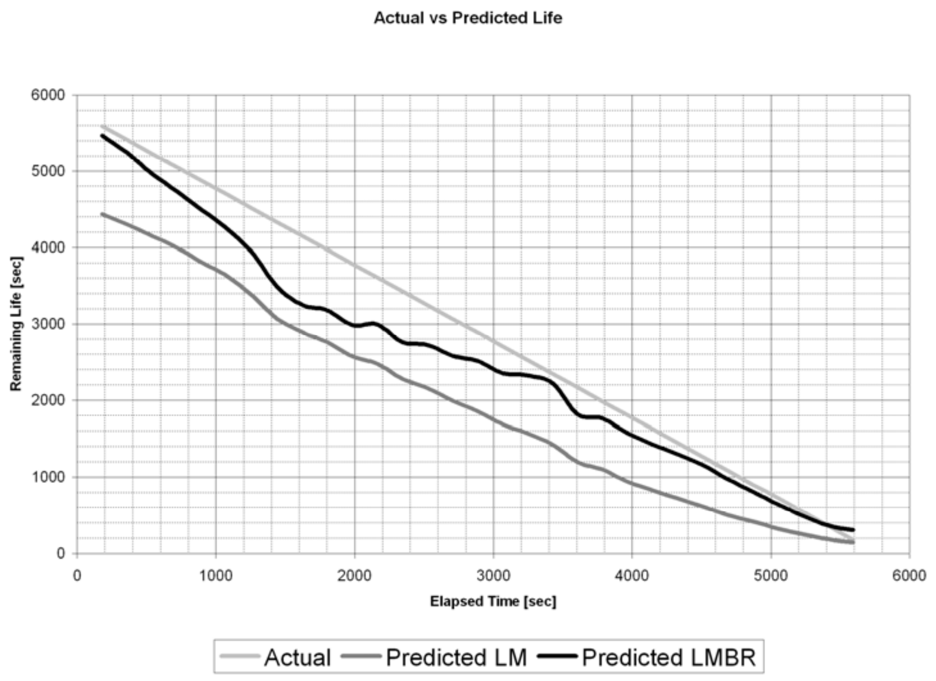

Figure 6: A comparison of predictions made by the networks trained with the Levenberg-Marquardt and Levenberg-Marquardt with Bayesian regularization algorithms. The load conditions of this particular case differed most from the examples in the training set, highlighting the advantage of the latter algorithm.

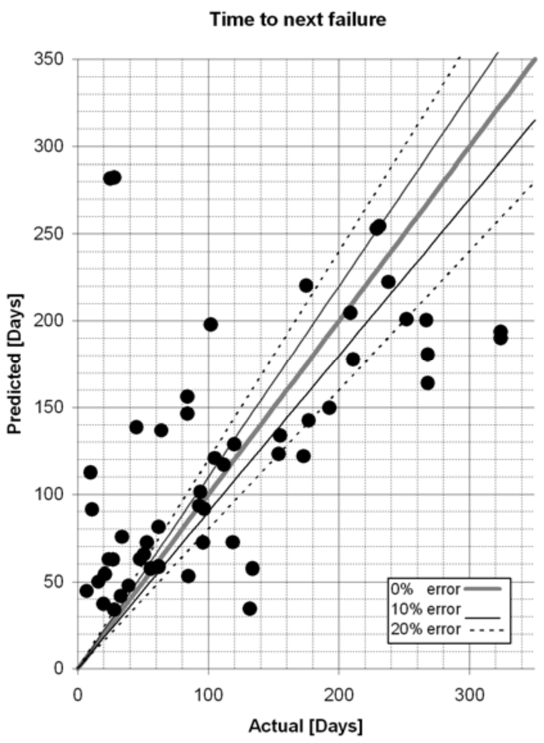

Figure 7: The actual failure times of the repaired system compared with the predictions made by a multi-layer perceptron neural network trained with the Levenberg-Marquardt algorithm with Bayestian regularization.

Table 1: Typical data collected over the life of one test piece during lab testing.

Time [sec.]	Load Range [kN]	Temperature [°C]	Load Mean [kN]
180	231	18.0	-33
361	231	18.1	-33
541	230	18.3	-33
721	230	18.4	-33
902	230	18.4	-33
1082	229	18.5	-33
1262	228	18.4	-33
1443	226	18.3	-33
1623	226	18.5	-33
1803	226	18.3	-33
1984	225	18.2	-33
2164	226	18.2	-33
2344	225	18.2	-33
2525	225	18.2	-33
2705	225	18.2	-33
2886	225	18.2	-33
3066	225	18.3	-33
3246	225	18.2	-33
3427	225	18.3	-33
3607	223	18.6	-33
3787	223	18.4	-33
3968	223	18.7	-33
4148	222	18.6	-33
4329	221	18.7	-33

Time [sec.]	**Load Range [kN]**	**Temperature [°C]**	**Load Mean [kN]**
4509	220	18.8	-33
4689	219	19.1	-33
4870	218	19.3	-33
5050	217	19.8	-33
5230	215	20.4	-33
5411	212	21.4	-33
5591	205	22.9	-33

Table 2: Extract showing the data collected from the pump PC1131. Failures occurred at the global age of 554 and 765 days, while condition based interventions were made after 397 and 690 days.

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

Table 3: 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%

Table 4: Performance of the GRNN for different spread values.

Spread	MSE training	MSE test
0.01	9.9×10^{-7}	0.0030
0.02	3.8×10^{-6}	0.0027
0.03	8.3×10^{-5}	0.0022
0.04	8.5×10^{-4}	0.0026
0.05	2.3×10^{-3}	0.0034

Table 5: Comparison of the mean squared error (MSE) on the training and test sets of the different networks.

Network	**MSE training**	**MSE test**
LM with BR	5.7×10^{-5}	0.0014
GRNN	8.3×10^{-5}	0.0022
LM	8.1×10^{-5}	0.0030
GDBP with M	1.9×10^{-2}	0.0061

Table 6: Comparison of the average error in the predictions made by the networks.

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.

Table 7: Largest error in the predictions made by the networks being compared.

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.

Table 8: The best results achieved with the Levenberg-Marquardt training algorithm.

Network architecture	Inputs	Target	Σ (error)^2
5 hidden nodes	5	0.05	2.70×10^5
5 hidden nodes	4	0.025	2.85×10^5
4 hidden nodes	4	0.025	2.90×10^5
3 hidden nodes	3	0.05	2.92×10^5
4 hidden nodes	4	0.05	2.92×10^5

Table 9: The network results with the largest error after training with the Levenberg-Marquardt algorithm.

Network architecture	Inputs	Target	Σ (error)^2
3 hidden nodes	3	0.00001	4.32×10^5
4 hidden nodes	3	0.00001	4.45×10^5
4 hidden nodes	3	0.01	4.49×10^5
5 hidden nodes	5	0.01	4.99×10^5
5 hidden nodes	5	0.00001	5.31×10^5

Table 10: Levenberg-Marquardt algorithm with Bayesian regularization.

Network architecture	Inputs	Σ (error)²
5 hidden nodes, linear output node	5	2.81×10^5
5 hidden nodes, sigmoid output node	5	2.90×10^5
5 hidden nodes, linear output node	4	2.95×10^5
4 hidden nodes, sigmoid output node	4	3.06×10^5
5 hidden nodes, sigmoid output node	4	3.07×10^5
4 hidden nodes, linear output node	4	3.15×10^5
4 hidden nodes, linear output node	3	3.26×10^5
3 hidden nodes, linear output node	3	3.35×10^5
3 hidden nodes, sigmoid output node	3	3.52×10^5
4 hidden nodes, sigmoid output node	3	3.52×10^5

Table 11: Cross-validation for GRNN

Network architecture	Inputs	Σ (error)²
Spread = 0.1	4	3.00×10^5
Spread = 0.05	4	3.32×10^5
Spread = 0.1	3	3.56×10^5
Spread = 0.05	3	3.72×10^5
Spread = 0.1	5	3.72×10^5
Spread = 0.05	5	4.17×10^5

Table 12: Comparison of the best results achieved by the different network types.

Network architecture	**Σ (error)^2**
LM	2.70×10^5
LMBR	2.81×10^5
GRNN	3.00×10^5